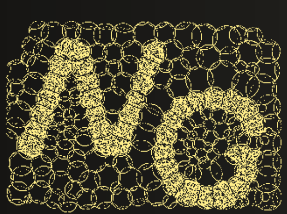
LECTURE NOTES IN COMPUTATIONAL SCIENCE AND ENGINEERING 143

Vladimir A. Garanzha
Lennard Kamenski · Hang Si *Editors*

Numerical Geometry, Grid Generation and Scientific Computing



Springer

Lecture Notes in Computational Science and Engineering

Volume 143

This series contains monographs of lecture notes type, lecture course material, and high-quality proceedings on topics described by the term "computational science and engineering". This includes theoretical aspects of scientific computing such as mathematical modeling, optimization methods, discretization techniques, multiscale approaches, fast solution algorithms, parallelization, and visualization methods as well as the application of these approaches throughout the disciplines of biology, chemistry, physics, engineering, earth sciences, and economics.

More information about this series at http://www.springer.com/series/3527

Vladimir A. Garanzha • Lennard Kamenski • Hang Si
Editors

Numerical Geometry, Grid Generation and Scientific Computing

Proceedings of the 10th International Conference, NUMGRID 2020 / Delaunay 130, Celebrating the 130th Anniversary of Boris Delaunay, Moscow, Russia, November 2020

Springer

Editors
Vladimir A. Garanzha
Dorodnicyn Computing Centre, Federal Research Center of Informatics and Control Russian Academy of Sciences Moscow
Moscow, Russia

Lennard Kamenski
Berlin, Germany

Hang Si
Weierstrass Institute for Applied Analysis and Stochastics (WIAS)
Berlin, Germany

ISSN 1439-7358 ISSN 2197-7100 (electronic)
Lecture Notes in Computational Science and Engineering
ISBN 978-3-030-76797-6 ISBN 978-3-030-76798-3 (eBook)
https://doi.org/10.1007/978-3-030-76798-3

Mathematics Subject Classification (2010): 65-xx (65Dxx, 65Fxx, 65Kxx, 65Lxx, 65Mxx, 65Nxx, 65Yxx, 65Zxx), 30Cxx, 30Fxx, 49Mxx, 52Cxx, 53xx

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Dedicated to
Boris Nikolayevich Delaunay *(1890–1980)*
on the occasion
of his 130th birthday

Foreword

This volume presents the proceedings of the NUMGRID 2020/Delaunay 130 International Conference dedicated to the 130th birthday of B. N. Delaunay (1890–1980). Since Boris Nikolayevich was my teacher, I will take this nice opportunity to have another look at certain moments in the life and work of this unique personality.

Delaunay triangulation, Delaunay partition, and their theory is the most important part of his research from an application point of view. This is, however, merely one facet in his multifaceted work, a facet to which B. N. Delaunay came not by chance. There is a deeper meaning in the fact that the name of Delaunay is forever inscribed in science next to that of G. F. Voronoi, a prominent representative of the St. Petersburg school of number theory. Even more so, because Georgy Feodosievich was a close friend of the Delaunay family. As a teenager, Boris often witnessed his father's late evening conversations with Voronoi.

Voronoi was not and could not be Delaunay's scientific supervisor: he died unexpectedly in 1908, the year Boris entered university. Nevertheless, his influence on Delaunay was considerable. Delaunay's breakthrough results on the cubic Diophantine equations, which are, by his own admission, his most outstanding work, used the famous Voronoi algorithm for finding fundamental units in cubic fields. This celebrated work of Voronoi once made a stunning impression on A. A. Markov (Voronoi's teacher) himself.

In the early 1920s, Delaunay was invited by A. A. Markov to the Petrograd[1] University as a professor and wrote a remarkable paper revealing the geometrical essence of the Voronoi algorithm. In the late 1920s, Delaunay published a major work on 4-dimensional parallelohedra, where he continued the research of H. Minkowski and G. Voronoi on the theory of parallelohedra. At the same time, Delaunay elegantly introduced the important concepts of the (r, R)-system and the L-partition corresponding to this system and published an extensive paper *"Geometry of positive quadratic forms"*. The concepts laid down in Delaunay's works in the 1920s and 1930s, influenced by Voronoi's work, proved to be useful

[1]The name of St. Petersburg in 1914–1924.

in computational geometry, crystallography, structural chemistry, biology, and other fields. As for the terms (r, R)-system and L-partition, much later, in the second half of the twentieth century, the professional community abandoned them in favor of the terms *Delaunay set* and *Delaunay partition*, largely thanks to H. S. M. Coxeter and C. A. Rogers.

As for the NUMGRID 2020 conference, I would like to acknowledge the high professional level of the participants and thank its organizers Vladimir Garanzha, Hang Si, and Lennard Kamenski.

Steklov Mathematical Institute RAS, Moscow, Russia Nikolay Dolbilin
February 2021

Preface

This volume presents a selection of papers presented at the 10th International Conference on Numerical Geometry, Grid Generation, and Scientific Computing celebrating the 130th anniversary of B. N. Delaunay (NUMGRID 2020/Delaunay 130), held November 25–27, 2020. The conference is bi-annual (since 2002) and it is one of the well-known international conferences in the area of mesh generation. The main topic of this conference, grid (mesh) generation, is about how to create a geometric discretization of a given domain. It is an indispensable tool for solving field problems in nearly all areas of applied mathematics.

The book includes an overview of the current progress in numerical geometry, grid generation, and adaptation in terms of mathematical foundations, algorithm and software development, and applications. In focus are the Voronoi-Delaunay theory and algorithms for tilings and partitions, mesh deformation and optimization, equidistribution principle, error analysis, discrete differential geometry, duality in mathematical programming and numerical geometry, mesh-based optimization and optimal control methods, iterative solvers for variational problems, as well as algorithm and software development. The applications of the discussed methods are multidisciplinary and include problems from mathematics, physics, biology, chemistry, material science, and engineering.

The presented 25 papers were selected from 31 submissions. The main selection criteria are based on the recommendations of anonymous peer reviews from experts of the corresponding fields as well as the presentation of the paper at the conference. All accepted papers are revised according to the comments of reviewers and the program committee.

The organizers would like to thank all who submitted papers and all who helped to evaluate the contributions by providing reviews for the submissions. The

reviewers' names are acknowledged in the following pages. The organizers would like to thank all participants of NUMGRID for making it a successful and interesting experience.

Moscow, Russia — Vladimir A. Garanzha
Berlin, Germany — Lennard Kamenski
Berlin, Germany — Hang Si
February 2021

Contents

Part I Delaunay-Voronoi Theory and Applications

Local Groups in Delone Sets 3
Nikolay Dolbilin

Manifolds of Triangulations, Braid Groups of Manifolds, and the Groups Γ_n^k 13
Denis A. Fedoseev, Vassily O. Manturov, and Igor M. Nikonov

A Proof of the Invariant-Based Formula for the Linking Number and Its Asymptotic Behaviour 37
Matt Bright, Olga Anosova, and Vitaliy Kurlin

The Singularity Set of Optimal Transportation Maps 61
Zhongxuan Luo, Wei Chen, Na Lei, Yang Guo, Tong Zhao, and Xianfeng Gu

Polygonal and Polyhedral Delaunay Meshing 81
Vladimir Garanzha and Liudmila Kudryavtseva

On Decomposition of Embedded Prismatoids in $\mathbb{R}^3$ Without Additional Points 95
Hang Si

Out-of-core Constrained Delaunay Tetrahedralizations for Large Scenes 113
Ziya Erkoç, Aytek Aman, Uğur Güdükbay, and Hang Si

Part II Adaptive Meshing

Size Gradation Control for Anisotropic Hybrid Meshes 127
Lucille-Marie Tenkes and Frédéric Alauzet

Adjoint Computation on Anisotropic Meshes in High-fidelity RANS Simulations 141
Francesco Clerici and Frédéric Alauzet

Moving Deforming Mesh Generation Based on the Quasi-Isometric Functional 157
Vladimir A. Garanzha and Liudmila Kudryavtseva

Adaptive Grids for Non-monotone Waves and Instabilities in a Non-equilibrium PDE Model 179
Paul A. Zegeling

RBF-VerBSS Hybrid Method for Mesh Deformation 199
Jihai Chang, Fei Yu, Jie Cao, and Zhenqun Guan

A Uniform Convergence Analysis for a Bakhvalov-Type Mesh with an Explicitly Defined Transition Point 213
Thái Anh Nhan

On a Comprehensive Grid for Solving Problems Having Exponential or Power-of-First-Type Layers 227
V. D. Liseikin, S. Karasuljic, A. V. Mukhortov, and V. I. Paasonen

Preserved Structure Constants for Red Refinements of Product Elements 241
Sergey Korotov and Jon Eivind Vatne

Part III Meshing and CAD

Global Parametrization Based on Ginzburg-Landau Functional 251
Victor Blanchi, Étienne Corman, Nicolas Ray, and Dmitry Sokolov

Parametrization of Plane Irregular Regions: A Semi-automatic Approach I 263
Pablo Barrera and Iván Méndez

A Hybrid Approach to Fast Indirect Quadrilateral Mesh Generation 281
Daniel Zint and Roberto Grosso

Hexahedral Mesh Generation Using Voxel Field Recovery 295
Alexander Sergeevich Karavaev and Sergey Petrovich Kopysov

Generation of Boundary Layer Meshes by the Enhanced Jump-and-Walk Method with a Fast Collision Detecting Algorithm 307
Jie Cao, Fei Yu, Zhonghai Gao, S.H. Lo, and Zhenqun Guan

Part IV Numerical Geometry and Applications

An Improved Algorithm for Scattered Data Interpolation Using Quartic Triangular Bézier Surfaces 327
Krassimira Vlachkova

On Integral-Based (Transfinite) Laplace Coordinates 341
Alexander G. Belyaev and Pierre-Alain Fayolle

Part V Numerical Methods

Fully-Implicit Collocated Finite-Volume Method for the Unsteady Incompressible Navier–Stokes Problem 361
Kirill M. Terekhov

Efficient Steady Flow Computations with Exponential Multigrid Methods 375
Shu-Jie Li

Exponential Time Integrators for Unsteady Advection–Diffusion Problems on Refined Meshes 391
Mikhail A. Botchev

Author Index 405

Conference Organization

Organizers

Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia
http://www.ccas.ru

Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
https://www.wias-berlin.de

Lennard Kamenski, Berlin, Germany
https://gitlab.com/lkamenski

Organizing Committee

V. A. Garanzha (Chair)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

L. Kamenski (Vice chair)
Berlin, Germany

A. I. Belokrys-Fedotov
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

I. E. Kaporin
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

L. N. Kudryavtseva
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Yu. O. Trusova
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

I. A. Zonn
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Program Committee

Yu. G. Evtushenko (Chair)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

H. Si (Vice Chair)
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany

A. Belyaev
Heriot-Watt University, Edinburgh, UK

H. Borouchaki
Inria Paris-Rocquencourt, France

N. P. Dolbilin
Steklov Mathematical Institute RAS, Moscow, Russia

V. P. Dymnikov
Institute of Numerical Mathematics RAS, Moscow, Russia

H. Edelsbrunner
Institute of Science and Technology, Klosterneuburg, Austria

S. K. Godunov
Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia

R. Jain
Argonne National Laboratory, Lemont, IL, USA

V. F. Kuropatenko
All-Russian Scientific Research Institute of Technical Physics, Snezhinsk, Russia

P. Laug
Inria Paris-Rocquencourt, France

N. Lei
Dalian University of Technology, Dalian, China

V. D. Liseikin
Institute of Computing Technologies RAS, Novosibirsk, Russia

Yu. V. Nesterenko
Moscow State University, Moscow, Russia

R. V. Polozov
Institute of Cell Biophysics, Puschino, Russia

X. Roca
Barcelona Supercomputing Center, Barcelona, Spain

D. V. Sokolov
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

S. K. Vodopyanov
Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia

Web Site

http://www.ccas.ru/gridgen/numgrid2020

Reviewers

Igor Baburin Technische Universität Dresden, Dresden, Germany

Pavel Bakhvalov Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Alexander Belyaev Heriot-Watt University, Edinburgh, UK

Matthew Bright University of Liverpool, Liverpool, UK

Michail Botchev Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Andrey Chernikov Old Dominion University, Norfolk, VA, USA

Kristian Debrabant University of Southern Denmark, Odense, Denmark

Vladimir A. Garanzha Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Alexander Golikov Dorodnicyn Computing Center of FRC CSC RAS, Moscow, Russia

Roberto Grosso Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Xianfeng Gu Stony Brook University, Stony Brook, NY, USA

Zhenqun Guan Dalian University of Technology, Dalian, China

Ronald D. Haynes Memorial University of Newfoundland, St. John's, NL, Canada

Nancy Hitschfeld University of Chile, Santiago, Chile

Rajeev Jain Argonne National Laboratory, Lemont, IL, USA

George Kamenev (The author and reviewer of the NUMGRID conferences, passed away on November 3rd 2020.) Federal Research Center of Informatics and Management RAS, Moscow, Russia

Lennard Kamenski Berlin, Germany

Jiří Kosinka Rijksuniversiteit Groningen, Groningen, The Netherlands

Na Lei Dalian University of Technology, Dalian, China

Shu-Jie Li Beijing Computational Science Research Center, Beijing, China

Alexander Linke Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany

Vladimir Liseikin Institute of Computational Technologies, Siberian Branch RAS, Novosibirsk, Russia

Yang Liu Microsoft Research Asia, Beijing, China

Xiaoming Liu Cadence Design Systems, Inc., San Jose, CA, China

Vassily Manturov Moscow Institute of Physics and Technology, Moscow, Russia

Yulong Pan University of California, Berkeley, CA, USA

Per-Olof Persson University of California, Berkeley, CA, USA

Egon Schulte Northeastern University, Boston, MA, USA

Alexander Skovpen NUMECA International, Brussels, Belgium

Kirill Terekhov Marchuk Institute of Numerical Mathematics RAS, Moscow, Russia

Vladimir Titarev Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Sergey Utyuzhnikov The University of Manchester, Manchester, UK

Yuri Vassilevski Marchuk Institute of Numerical Mathematics RAS, Moscow, Russia

Krassimira Vlachkova Sofia University St. Kliment Ohridski, Sofia, Bulgaria

Rhaleb Zayer Max-Planck-Institut für Informatik, Saarbrücken, Germany

Paul Zegeling Universiteit Utrecht, Utrecht, The Netherlands

Xiaopeng Zheng Dalian University of Technology, Dalian, China

Victor Zhukov Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Daniel Zint Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Part I
Delaunay-Voronoi Theory and Applications

Local Groups in Delone Sets

Nikolay Dolbilin

Abstract We prove that in an arbitrary Delone set X in the three-dimensional space, the subset X_6 of all points from X at which the local group has no rotation axis of order larger than 6 is also a Delone set. Here, under the *local group at point* $x \in X$ we mean the symmetry group $S_x(2R)$ of the cluster $C_x(2R)$ of x with radius $2R$, where R is the radius of the largest ball free of points of X (according to Delone's *empty sphere* theory).

The main result seems to be the first rigorously proved statement for *absolutely generic* Delone sets which implies substantial statements for Delone sets with strong crystallographic restrictions. For instance, an important observation of Shtogrin on the boundedness of local groups in Delone sets with equivalent $2R$-clusters immediately follows from the main result.

Further, we propose a *crystalline kernel conjecture* and its two weaker versions. According to the crystalline kernel conjecture, in an arbitrary Delone set, points with locally crystallographic axes only (i.e., of order 1, 2, 3, 4, or 6) *inevitably* constitute the essential part of the set. These conjectures significantly generalize the famous crystallography statement on the impossibility of a (global) fivefold symmetry in a three-dimensional lattice.

1 Introduction and Basic Definitions

This work grew out of the local theory for regular systems (Delone sets with very strong requirements). In this paper, however, we consider arbitrary Delone sets in $\mathbb{R}^3$ without any additional assumptions. For example, we do not require typical conditions of the local theory such as sameness of clusters of certain radius as we did in numerous papers (e.g., [1–3]). Another feature of the paper is as follows: for

N. Dolbilin (✉)
Steklov Mathematical Institute of the Russian Academy of Sciences, Moscow, Russia
e-mail: dolbilin@mi-ras.ru

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_1

a Delone set, we consider local groups operated over clusters of a particular radius, namely $2R$ (see the definitions below).

In [4], A.L. Mackay says:

> In an infinite crystal there may be extra elements of symmetry which operate over a limited range. These may be seen by non-space-group extinctions in diffraction pattern. . . . The local operations need not to be 'crystallographic'.

Nevertheless, the main result of this paper implies that the character of local groups is essentially predetermined for a quite wide class of Delone sets in $\mathbb{R}^3$, the so-called $2R$-isometric sets, and, in particular, for regular systems (i.e., for Delone sets with transitive groups). Here, a very significant role is played by the successful choice of the range of action of the local group. It is chosen as $2R$, R being the parameter of the Delone set (for details see below). To accurately formulate the main result and some open hypotheses we will need several definitions.

Let $|x, x'|$ denote the Euclidean distance between points x and x' in the Euclidean space $\mathbb{R}^d$ and $d(z, X) := \inf_{x\in X} |z, x|$ the distance from a point $z \in \mathbb{R}^d$ to the set $X \subset \mathbb{R}^d$.

Definition 1 (Delone Set) Given positive real numbers r and R, a point subset X of $\mathbb{R}^d$ is called a *Delone set of type* (r, R) if the following two conditions hold:

(1) an open d-ball $B^o_z(r)$ of radius r centered at any point z of space contains at most one point of X;
(2) a closed d-ball $B_z(R)$ of radius R centered at any point z of space contains at least one point of X.

Obviously, a Delone set X of type (r, R) is a Delone set of type (r', R') if $r' \le r$ and $R' \ge R$. Therefore, we can adopt the following convention: for a given Delone set X, we choose r as the largest possible value satisfying Condition (1) of Definition 1, and R as the smallest value satisfying Condition (2).

We will also need the following interpretation of the parameters r and R:

$$\inf_{x,x'\in X} |x, x'| = 2r \quad \text{and} \quad \sup_{z\in\mathbb{R}^d} d(z, X) = R. \tag{1}$$

Thus, the value of r equals to the half of the smallest (infimum) inter-point distance in X. The value of R is the distance from X to a point of $\mathbb{R}^d$ farthest away from X.

In the local theory of Delone sets, the key concept is that of a cluster.

Definition 2 (ρ-Cluster) Let x be a point of a Delone set X of type (r, R) and $B_z(\rho)$ a ball of radius $\rho \ge 0$ centered at a point $z \in \mathbb{R}^d$. We call a point set

$$C_{\mathbf{x}}(\rho) := X \cap B_{\mathbf{x}}(\rho)$$

the *cluster of radius* ρ at point x or, simply, the *ρ-cluster* at x.

Definition 3 (Equivalent Clusters) Two clusters $C_x(\rho)$ and $C_{x'}(\rho)$ of the same radius ρ at points x and x' are said to be *equivalent* if there is an isometry

$g \in \mathrm{Iso}(\mathbb{R}^3)$ such that

$$g(x) = x' \quad \text{and} \quad g(C_x(\rho)) = C_{x'}(\rho).$$

Definition 4 (Cluster Group) Given a point $x \in X$ and its ρ-cluster $C_x(\rho)$, a group $S_x(\rho)$ of all isometries $s \in \mathrm{Iso}(\mathbb{R}^d)$ which leave x fixed and the cluster $C_x(\rho)$ invariant is called the *cluster group*:

$$S_x(\rho) \coloneqq \left\{ s \in \mathrm{Iso}(\mathbb{R}^d) \mid s(x) = x,\ s(C_x(\rho)) = C_x(\rho) \right\}.$$

Groups of equivalent clusters $C_x(\rho)$ and $C_{x'}(\rho)$ are conjugate in the full group $\mathrm{Iso}(\mathbb{R}^d)$ of isometries: $S_x(\rho) = g^{-1} S_{x'}(\rho)\, g$, where g is determined by the conditions of Definition 3. Clearly, as the radius ρ increases, the cluster $C_x(\rho)$ expands but the cluster group $S_x(\rho)$ never increases and sometimes can only contract. Further, if $0 \le \rho < 2r$ then $C_x(\rho) = O_x(3)$, i.e., it is the full point group of all isometries that leave the point x fixed. On the other hand, it is well-known that the group $S_x(2R)$ is finite for any $x \in X$.

Since the main result grew up ideologically from the local theory of regular systems, we briefly recall the basic concepts of this theory. Modern in form, the following definitions of a regular system and a crystal are equivalent to those going back to E.S. Fedorov.

Definition 5 (Regular System, Crystal) A Delone set X is called a *regular system* if it is an orbit of some point x with respect to a certain space group $G \subset \mathrm{Iso}(\mathbb{R}^d)$:

$$X = G \cdot x = \{\, g(x) \mid g \in G \,\}.$$

A Delone set X is a *crystal* if it is a union of a finite number of orbits:

$$X = \bigcup_{i=1}^{m} G \cdot x_i.$$

The notion of a regular system is an essential case of a crystal (which is a multi-regular system) and generalizes the concept of a lattice. In fact, a lattice is a particular case of a regular system when G is a group of translations generated by d linearly independent translations. Moreover, due to a celebrated theorem by Schoenflies and Bieberbach, any regular system is a union of congruent and mutually parallel lattices.

The local theory of regular systems began with the Local Criterion in [1].

Theorem 1 (Local Criterion [1]) *A Delone set X is a regular system if and only if there is some $\rho_0 > 0$ such that the following two conditions hold:*

(1) all $\rho_0 + 2R$-clusters are mutually equivalent;
(2) $S_x(\rho_0) = S_x(\rho_0 + 2R)$ for $x \in X$.

In [5, 6], this criterion has been generalized for crystals (multi-regular systems).

From now on, we restrict ourselves only to the three-dimensional case. One of the central problems of the local theory of regular systems is the estimation of an upper (and a lower) bound for the regularity radius, i.e., the smallest value $\hat{\rho}_3 > 0$ such that the equivalence of $\hat{\rho}_3$-clusters in a Delone set X implies the regularity of the set $X \subset \mathbb{R}^3$. A proof of the upper bound $\hat{\rho}_3 \leq 10R$ was given in [2, 3]. The long proof starts with a selection of a special finite list of finite subgroups of $O(3)$. Groups from this list have a chance to occur in Delone sets with equivalent $2R$-clusters as local groups $S_x(2R)$. The list of selected groups is provided by the following Theorem 2. It was discovered by Shtogrin in the late 1970's but published only in 2010 [7].

Theorem 2 ([7]) *If all $2R$-clusters in a Delone set $X \in \mathbb{R}^3$ are mutually equivalent, then the order of any rotational axis of $S_x(2R)$ does not exceed 6.*

Recently [8], it was realized for the first time that an important statement about groups in Delone sets **with significant requirements on equivalent clusters** may follow from a certain statement which is true for **general** Delone sets. Namely, the following Theorem 3 was proved.

Theorem 3 ([8]) *In a Delone set $X \subset \mathbb{R}^3$ there is at least one point x with $n_x \leq 6$, n_x being the maximal order of a rotational axis in the group $S_x(2R)$.*

Obviously, Theorem 3 immediately implies Theorem 2. In fact, the subset X_6 of all points in a Delone set X with $n_x \leq 6$ is always very rich. Due to the following main result (Theorem 4), the subset X_6 is a Delone set itself.

2 Main Result and Conjectures

Theorem 4 (Main Result) *Given a Delone set $X \subset \mathbb{R}^3$ of type (r, R), let $X_6 \subseteq X$ be the subset of all points $x \in X$ such that the maximal order n_x of a rotation axis in $S_x(2R)$ does not exceed 6. Then X_6 is a Delone set of a certain type (r', R'), where $r \leq r' \leq R' \leq kR$ for some k which is independent of X.*

At the moment, we do not care for the upper bound kR on the parameter R' of the X_6. So far, it is more important to establish that the subset X_6 is a Delone set as well.

From now on, we will focus on clusters $C_x(2R)$ of radius $2R$ and their groups $S_x(2R)$. It is well-known that all $2R$-clusters for a Delone set X are full-dimensional (i.e., the dimension of their convex hulls is d). Hence, the cluster groups $S_x(2R)$ are necessarily finite. At the same time, $2R$ is the smallest radius value which guarantees the finiteness of the group of a $2R$-cluster for any Delone set with parameter R. In other words, for an arbitrary $\varepsilon > 0$, there are a Delone set X (with parameter R) and its point $x \in X$ such that the group $S_x(2R - \varepsilon)$ is infinite.

By virtue of the above, we will single out the group $S_x(2R)$ and call it a *local group at x*.

Theorem 4 immediately implies Theorem 2 about a Delone set X with mutually equivalent $2R$-clusters. For such a Delone set X, the local groups at all points are pairwise conjugate and existence of points x with $n_x \leq 6$ implies $n_y \leq 6$ for all points $y \in X$.

Now, among points of $X_6 \subseteq X$ we select points x with $n_x \neq 5$, i.e., all points of X whose local groups contain axes of only 'crystallographic' orders 2, 3, 4, or 6. We call the subset of all such points in X a *crystalline kernel* of X and denote it by K.

Conjecture 1 (Crystalline Kernel Conjecture) The crystalline kernel K of a Delone set X is a Delone subset with some parameter $R' \leq kR$, where k is some constant which is independent of X.

Let Y denote the subset of all points $x \in X$ at which local groups $S_x(2R)$ do not contain the pentagonal axis. Clearly, $K \subseteq Y$ and if K is a Delone set then Y is a Delone set, too. Therefore, Conjecture 1, if proven, immediately implies the following Conjectures 2 and 3.

Conjecture 2 (5-Gonal Symmetry Conjecture) For a given Delone set $X \subset \mathbb{R}^3$, the subset Y of points x whose groups $S_x(2R)$ are free of fivefold axes is also a Delone set.

Conjecture 2, in turn, implies the following statement which seems to be much easier to prove.

Conjecture 3 (Weak 5-Gonal Symmetry Conjecture) For a given Delone set $X \subset \mathbb{R}^3$ with mutually equivalent $2R$-clusters, the local group $S_x(2R)$ contains no fivefold axis.

Obviously, these hypotheses relate to a celebrated crystallographic theorem on the impossibility of the global fivefold symmetry in a three-dimensional lattice. Conjectures 1–3 significantly reinforce the classical statement on the famous crystallographic restrictions.

For a three-dimensional lattice, it is well-known that there is no fivefold symmetry even in the group $S_x(r_1)$, where $r_1 = 2r \leq 2R$ is the minimum inter-point distance in the lattice. In contrast to lattices, in three-dimensional regular systems the pentagonal symmetry can *locally* manifest itself on clusters of a certain radius less than $2R$. For instance, there are regular systems where even the group $S_x(r_3)$ contains the fivefold axis but the local group $S_x(2R)$ does not (r_3 being the third-smallest inter-point distance in X: $r_1 < r_2 < r_3 < 2R$). It is still unknown, however, whether there are regular systems with fivefold symmetric $2R$-clusters.

Since the regularity radius for dimension 3 is not less than $6R$ [9], there are non-regular and even non-crystallographic sets among Delone sets X with mutually equivalent $2R$-clusters. Thus, even the weakest Conjecture 3 is relevant for a wide class of these non-regular sets.

3 Proof of the Main Result

Generally speaking, in the local group $S_x(2R) \subset O(3)$, $x \in X$, there are several axes of the maximal order n_x. Bearing in mind the well-known list of all finite subgroups of $O(3)$, we see that more than one axes of the maximal order n_x in $S_x(2R)$ cannot happen provided $n_x > 5$. Let ℓ_x be one of these axes. Since $\mathrm{rk}(C_x(2R)) = 3$, i.e., the convex hull of the $2R$-cluster is three-dimensional, there are necessarily points off the ℓ_x in $C_x(2R)$.

Since X_6 is a subset of X, the minimal inter-point distance $2r'$ in X_6 (in fact, the infimum of such distances) is not less than $2r = \inf_{x,x' \in X} |x, x'|$. In order to prove that X_6 is a Delone set with a certain parameter R', we will prove that the distance from a given point $z \in \mathbb{R}^3$ to the nearest point of X_6 does not exceed R': $\min_{x \in X'} |z, x| \leq R'$ (due to the interpretation (1) for r and R in Sect. 1). We will be looking for the point $x \in X_6$ nearest to $z \in \mathbb{R}^3$ by walking along a special finite point sequence in X_6.

Definition 6 (Off-Axial Chain) A sequence of points $[x_1, x_2, x_3, \dots] \in X$ (finite or infinite) is called an *off-axial chain* if for any $i = 1, 2, \dots$ the point $x_{i+1} \in X$ is the nearest point to x_i among all points of X that are off the axis ℓ_{x_i} of the local group $S_{x_i}(2R)$ of the maximal order n_{x_i}. If the subgroup of all (orientation-preserving) rotations of $S_{x_i}(2R)$ is trivial (that is, in the local group at x_i there are no axes through x_1), any point of X nearest to x_i can be chosen as x_{i+1}.

Note that for any point $x_1 \in X$ there are off-axial sequences $[x_1, x_2, x_3, \dots]$.

Lemma 1 *Given a Delone set X and an off-axial sequence $[x_1, x_2, \dots, x_m, \dots] \subset X$, assume that $x_i \notin X_6$ for all $i \in \overline{1, m}$. Then for all m and all $i \in \overline{1, m}$*

$$|x_i, x_{i+1}| < 0.868^{i-1} \cdot 2R \quad \textit{and} \quad |x_1, x_m| < 7.6 \cdot 2R = 15.2R. \tag{2}$$

Proof Let $[x_1, x_2, , \dots, x_m, \dots]$ be an off-axial chain and assume that it belongs to $X \setminus X_6$. By construction, the length r_i^* of each link x_i, x_{i+1} in this chain is less than $2R$. Hence, $x_{i+1} \in C_{x_i}(2R)$ and the rotation $g_{x_i} \in S_{x_i}(2R)$ can be applied to x_{i+1}, too.

By assumption, $x_1 \notin X_6$, that is, $n_{x_1} \geq 7$. Let x_2 be the nearest to x_1 point that is off the axis l_{x_1} and g_{x_1} a rotation around the axis ℓ_{x_1} by $2\pi/n_{x_1}$. Since $r_1^* \leq 2R$, the cluster $C_{x_1}(2R)$ necessarily contains vertices of a regular n_{x_1}-gon P_1 which is generated by the rotation $g_{x_1} \in S_{x_1}(2R)$ applied to the point x_2. The polygon P_1 is located in a plane orthogonal to the ℓ_{x_1}. The center of P_1 is on l_{x_1} (see Fig. 1).

Denote the side-length of P_1 by a_1. Since the circumradius of P_1 does not exceed $r_{x_1}^*$ and $n_{x_1} \geq 7$, we have the following estimate for a_1 and $r_{x_2}^*$:

$$r_{x_2}^* \leq a_1 \leq 2r_{x_1}^* \sin\frac{\pi}{n_{x_1}} \leq 2r_{x_1}^* \sin\frac{\pi}{7} < 0.868\, r_{x_1}^* < 0.868 \cdot 2R. \tag{3}$$

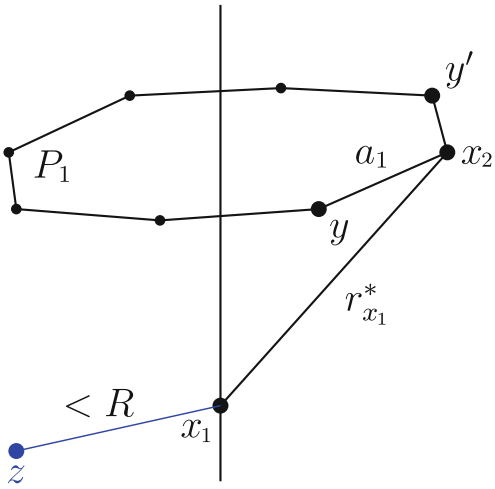


Fig. 1 Polygon P_1 and the beginning of an off-axial chain $[x_1, x_2, \dots]$

Assuming now that $x_2 \notin X_6$ (i.e., $n_{x_2} \geq 7$), we will construct the next point x_3 in the off-axial chain $[x_1, x_2, \dots]$ and obtain upper estimates for r_{x_3} and a_2 (inequalities (4) below).

The rotation g_{x_2} about the axis ℓ_{x_2} is assumed to belong to the local group $S_{x_2}(2R)$. Since $x_3 \in C_{x_2}(2R)$, g_{x_2} can be applied to the point x_3. Hence, the cluster $C_{x_2}(2R)$ necessarily contains vertices of a regular n_{x_2}-gon P_2 generated by rotation g_{x_2} applied to the point x_3. Denote the side-length of P_2 by a_2 and note that $a_2 \leq r^*_2 \leq 2R$.

Point x_3, as a vertex of the regular n_{x_2}-gon P_2, has two adjacent vertices in P_2 at a distance a_2 from x_3. Vertex x_3 and the two adjacent vertices of P_2 form a non-collinear triple. Therefore, no matter how the axis ℓ_{x_2} passes through the point x_2, at least one of the two neighboring points is off this axis. It follows that the distance $r^*_{x_3}$ from x_3 to the nearest point $x_4 \in X \setminus \ell_{x_3}$ does not exceed a_2. Since we bear in mind that, in the set $X \setminus \ell_{x_2}$, there may be points closer to x_3 than distance a_1 we have $r^*_{x_2} \leq a_2$.

Since $n_{x_2} \geq 7$, by the same argument as in (3) above, we obtain

$$r^*_{x_3} < a_2 \leq 2r^*_{x_2} \sin\frac{\pi}{n_{x_2}} \leq 2r^*_{x_2} \sin\frac{\pi}{7} < 0.868\, r^*_{x_1} < 0.868^2 \cdot 2R. \tag{4}$$

This can be repeated over and over again. Under condition $n_{x_i} \geq 7$ for all $i \in \overline{1, m}$, we obtain the off-axial chain $[x_1, x_2, \dots]$ for which the sequence of inter-point distances $r^*_{x_i} = |x_i, x_{i+1}|$ is dominated by a geometric progression,

$$r^*_{x_{i+1}} < 0.868\, r^*_{x_i} < (0.868)^i r^*_{x_1} < (0.868)^i \cdot 2R.$$

From this, we obtain the required inequalities (2). □

Now we are going to complete the Proof of Theorem 4. Given a Delone set X, Lemma 1 implies that an off-axial chain is finite if $n_{x_i} \geq 7$. Moreover, the length m of such an off-axial chain in X is bounded from above by a constant depending only on R/r. This implies that the chain $[x_1, x_2, \dots]$ has points x_m with $n_{x_m} \leq 6$, i.e., $x_m \in X_6$. By Lemma 1, the length of a segment satisfies $|x_1, x_m| < 15.2$.

Now we set up upper boundedness of the distance from an arbitrary space point z to the nearest point of the subset X_6. Let x_1 be the nearest point of X to z (see Fig. 1) and $x_m \in X_6$ (by Lemma 1). Then $|z, x_1| \leq R$ and

$$\min_{x \in X_6} |z, x| \leq |z, x_m| \leq |z, x_1| + |x_1, x_m| = 16.2\, R. \tag{5}$$

Hence, Theorem 4 is proved with $k = 16.2$.

4 Concluding Remarks

The upper bound for the parameter R' established here is far from the optimal. Soon we will be able to present a sharper bound [10, 11]. The purpose of this paper is to present the result which is, in our opinion, of a new type.

The result suggests a few conjectures which should be interesting both in itself and in the context of the theory of quasicrystals. For instance, in Penrose patterns, in structures of real Shechtman quasicrystals, the centers of $2R$-clusters with local fivefold symmetry constitute a rich Delone subset. At the same time, in these known quasicrystalline structures, there are also Delone subsets of points with local crystallographic axes (including identical). However, according to Conjecture 1, not only in these structures but in any other possible Delone sets, points with local crystallographic axes **inevitably** constitute an essential part of the structure. Thus, an arbitrary Delone set in $\mathbb{R}^3$ has a kind of a crystallographic matrix.

Acknowledgement This work is supported by the Russian Science Foundation under grant 20-11-19998.

References

1. Delone, B.N., Dolbilin, N.P., Shtogrin, M.I., Galiulin, R.V.: A local criterion for regularity of a system of points. Dokl. Akad. Nauk SSSR **227**(1), 19–21 (1976)
2. Dolbilin, N.P.: Delone sets in $\mathbb{R}^3$ with $2R$-regularity conditions. Topology and physics. volume dedicated to the 80th anniversary of academician Sergey Petrovich Novikov. Proc. Steklov Inst. Math. **302**, 161–185 (2018)
3. Dolbilin, N., Garber, A., Leopold, U., Schulte, E., Senechal, M.: On the regularity radius of Delone sets in $\mathbb{R}^3$. Discrete Comput. Geom. (2021, online). https://doi.org/10.1007/s00454-021-00292-6

4. Mackay, A.L.: Generalized crystallography. Comput. Math. Appl. Part B **12**(1–2), 21–37 (1986)
5. Dolbilin, N.P., Shtogrin, M.I.: A local criterion for a crystal structure. In: Abstracts of the IXth All-Union Geometrical Conference (in Russian), p. 99. Kishinev (1988)
6. Dolbilin, N.P., Lagarias, J.C., Senechal, M.: Multiregular point systems. Discrete Comput. Geom. **20**(4), 477–498 (1998)
7. Shtogrin, M.I.: On a bound of the order of a spider's axis in a locally regular Delone system. In: Abstracts of the International Conference "Geometry, Topology, Algebra and Number Theory, Applications" dedicated to the 120th anniversary of B.N. Delone (1890–1980) (in Russian), Moscow, August 16–20, 2010, pp. 168–169 (2010). http://delone120.mi.ras.ru/delone120abstracts.pdf
8. Dolbilin, N.P.: From local identity to global order. In: Materials of the XIII Lupanov International Seminar, MSU, June 17–22, 2019, pp. 13–22 (2019)
9. Baburin, I., Bouniaev, M., Dolbilin, N., Erokhovets, N.Yu., Garber, A., Krivovichev, S.V., Schulte, E.: On the origin of crystallinity: a lower bound for the regularity radius of Delone sets. Acta Crystallogr. Sect. A **74**(6), 616–629 (2018)
10. Dolbilin, N.P., Shtogrin, M.I.: On crystallographicity of local groups of Delone sets in Euclidean plane. Comput. Math. and Math. Physics (submitted)
11. Dolbilin, N.P., Shtogrin, M.I.: Local groups in Delone sets in 3-dimensional Euclidean space (in preparation)

Manifolds of Triangulations, Braid Groups of Manifolds, and the Groups Γ_n^k

Denis A. Fedoseev, Vassily O. Manturov, and Igor M. Nikonov

Abstract The spaces of triangulations of a given manifold have been widely studied. The celebrated theorem of Pachner (Eur J Comb 12:129–145, 1991) says that any two triangulations of a given manifold can be connected by a sequence of bistellar moves, or Pachner moves, see also Gelfand et al. (Discriminants, Resultants, and Multidimensional Determinants. Birkhäuser, Boston (1994)), Nabutovsky (Comm Pure Appl Math 49:1257–1270, 1996). In the present paper we consider groups which naturally appear when considering the set of triangulations with fixed number of simplices of maximal dimension. There are three ways of introducing this groups: the geometrical one, which depends on the metric, the topological one, and the combinatorial one. The second one can be thought of as a "braid group" of the manifold and, by definition, is an invariant of the topological type of manifold; in a similar way, one can construct the smooth version. We construct a series of groups Γ_n^k corresponding to Pachner moves of $(k-2)$-dimensional manifolds and construct a canonical map from the braid group of any k-dimensional manifold to Γ_n^k thus getting topological/smooth invariants of these manifolds.

D. A. Fedoseev (✉)
Moscow State University, Moscow, Russia

Moscow Center for Fundamental and Applied Mathematics, Moscow, Russia

V. O. Manturov
Moscow Institute of Physics and Technology, Moscow, Russia

Kazan Federal University, Kazan, Russia

I. M. Nikonov
Moscow State University, Moscow, Russia

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_2

1 Introduction

Returning back to 1954, J.W. Milnor in his first paper on link groups [9] formulated the idea that manifolds which share lots of well known invariants (homotopy type, etc.) may have different *link groups*.

In 2015, the second named author [7] defined a series of groups G_n^k for natural numbers $n > k$ and formulated the following principle:

The Main Principle
If dynamical systems describing the motion of n particles possess a nice codimension one property depending exactly on k particles, then these dynamical systems admit a topological invariant valued in G_n^k.

We shall define braids on a manifold as loops in some configuration space related to the manifold. Following G_n^k-ideology, we mark singular configurations, while moving along a loop in the configuration space, in order to get a word—an element of a "G_n^k-like" group Γ_n^k. Singular configurations will arise here from the moments of transformation of triangulations spanned by the configuration points; and in this case the good property of codimension one is "*k points of the configuration lie on a sphere of dimension $k-3$ and there are no points inside the sphere*".

In the present paper, we shall consider an arbitrary closed d-manifold M^d and for n large enough we shall construct the braid group $B_n(M^d) = \pi_1(C_n(M^d))$ (see Definitions 1–3), where $C_n(M^d)$ is the space of generic (in some sense) configurations of points in M^d which are dense enough to span a triangulation of M. This braid group will have a natural map to the group Γ_n^{d+2}. We shall consider point configurations for which the Delaunay triangulation exists, see [1].

The idea of this work comes from the second named author, and the third named author defined the constructions of Γ_n^k, $k \geq 5$, given in Sect. 3.

2 The Manifold of Triangulations

In the present section we define three types of manifold of triangulations: a geometrical one, a topological one, and a combinatorial one. We begin with geometrical approach, considering a manifold with a Riemannian metric.

2.1 Geometrical Manifold of Triangulations

Fix a smooth manifold M of dimension d with a Riemannian metric g on it. Let $n \gg d$ be a large natural number. Consider the set of all Delaunay triangulations of M with n vertices $x_1, \dots, x_n$ (if such triangulations exist). Such Delaunay triangulations are indexed by sets of vertices, hence, the set of such triangulation forms a subset of the configuration space, which we denote by $C(M, n)$. The above subset will be an open (not necessarily connected) manifold of dimension dn: if some n pairwise distinct points $(x_1, \dots, x_n)$ form a Delaunay triangulation then the same is true for any collection of points $(x'_1, \dots, x'_n)$ in the neighbourhood of $(x_1, \dots, x_n)$. We call a set of points $x_1, \dots, x_n$ *admissible* if such a Delaunay triangulation exists.

Hence, we get a non-compact (open) manifold M_g^{dn}. We call two sets of points $(x_1, \dots, x_n)$ and $(x'_1, \dots, x'_n)$ *adjacent* if there is a path $(x_{1,t}, \dots, x_{n,t})$ for $t \in [0, 1]$ such that $(x_{1,0}, \dots, x_{n,0}) = (x_1, \dots, x_n)$ and $(x_{1,1}, \dots, x_{n,1}) = (x'_1, \dots, x'_n)$ and there exists exactly one $t_0 \in [0, 1]$ such that when passing through $t = t_0$ the Delaunay triangulation $(x_{1,t}, \dots, x_{n,t})$ undergoes a flip (also called *Pachner moves* or *bistellar moves*).

The flip (Pachner move) corresponds to a position when some $d + 2$ points $(x_{i_1,t_0}, \dots, x_{i_{d+2},t_0})$ belong to the same $(d-1)$-sphere S^{d-1} such that no other point x_j lies inside the ball B bounded by this sphere.

Every manifold of triangulations described above has a natural stratification. Namely, every point of M_g^{dn} is given by a collection $(x_1, \dots, x_n)$. Such a collection is *generic* if there is no sphere S^{d-1} such that exactly $d+2$ points among x_k belong to this sphere without any points inside the sphere.

We say that a point $(x_1, \dots, x_n)$ is *of codimension* 1, if there exists exactly one sphere with exactly $d + 2$ points on it and no points inside it; analogously, *codimension* 2 *strata* correspond to either one sphere with $d + 3$ points or two spheres containing $d + 2$ points each.

In codimension 2 this corresponds to either one point of valency five in Voronoï tiling or two points of valency four in Voronoï tiling.

Hence, we have constructed a stratified (open) manifold M_g^{dn}. We call it the *geometrical manifold of triangulations*.

Note that the manifold M_g^{dn} may be not connected, i.e., there can exist non-equivalent triangulations. On the other hand, if one considers the spines of some manifold, they all can be transformed into each other by Matveev–Piergallini moves [8].

Denote the connected components of M_g^{dn} by $(M_g^{dn})_1, \dots, (M_g^{dn})_p$.

Definition 1 The *geometrical n-strand braid groups* of the manifold M_g are the fundamental groups

$$B_g(M_g, n)_j = \pi_1((M_g^{dn})_j), \quad j = 1, \dots, p.$$

2.2 Topological Manifold of Triangulations

The definition of geometrical manifold of triangulations heavily depends on the metric of the manifold M. For example, if we take M to be the torus glued from the square 1×10, the combinatorial structure of the manifold of triangulations will differ from the combinatorial structure for the case of the manifold of triangulations for the case of the torus glued from the square 1×1. Its analogue which is independent on the metric is the *topological manifold of triangulations*. We shall construct the 2-frame of this manifold. The idea is to catch all simplicial decompositions which may eventually happen for the manifold of triangulations for whatever metrics, and glue them together to get an open (non-compact) manifold.

Consider a topological manifold M^d. We consider all Riemannian metrics g_α on this manifold. They give manifolds $M^{dn}_{g_\alpha}$ as described in Sect. 2.1, which are naturally stratified. By a *generalised cell* of such a stratification we mean a connected component of the set of generic points of $M^{dn}_{g_\alpha}$.

We say that two generalised cells C_1 and C_2 are *adjacent* it there exist two points, say, $x = (x_1, \ldots, x_n)$ in C_1 and $x' = (x'_1, \ldots, x'_n)$ in C_2, and a path $x_t = (x_1(t), \ldots, x_n(t))$, such that $x_i(0) = x_i$ and $x_i(1) = x'(i)$ such that all points on this path are generic except for exactly one point, say, corresponding to $t = t_0$, which belongs to the stratum of codimension 1.

We say that two generic strata of $M^{dn}_{g_\alpha}, M^{dn}_{g_\beta}$ are *equivalent* if there is a homeomorphism of $M^{dn}_{g_\alpha} \to M^{dn}_{g_\beta}$ taking one stratum to the other.

A generalised 0-*cell* of the manifold of triangulations is an equivalence class of generic strata.

Analogously, we define generalised 1-*cells* (Fig. 1) of the manifold of triangulations as equivalence classes of pairs of adjacent vertices for different metrics $M^{dn}_{g_\alpha}$.

In a similar manner, we define generalised 2-*cells* (Fig. 2) as equivalence classes of discs for metrics $M^{dn}_{g_\alpha}$ such that:

1. vertices of the disc are points in 0-strata;
2. edges of the disc connect vertices from adjacent 0-strata; each edge intersects codimension 1 set exactly in one point;
3. the cycle is spanned by a disc which intersects codimension 2 set exactly at one point;
4. equivalence is defined by homeomorphism taking disc to disc, edge to an equivalent edge and vertex to an equivalent vertex and respects the stratification.

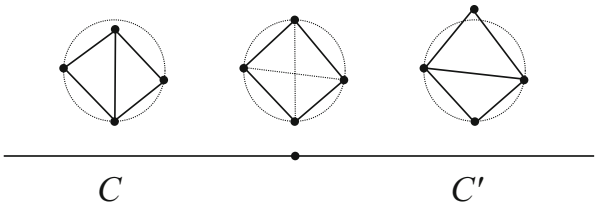


Fig. 1 Codimension 1 corresponds to a flip

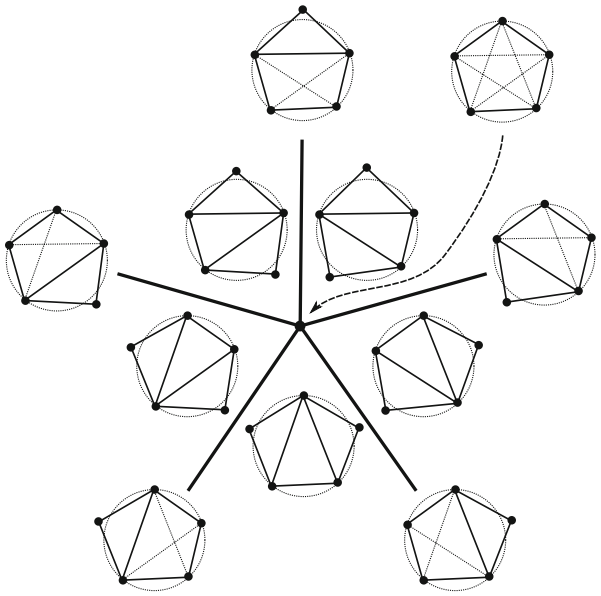


Fig. 2 Codimension 2 corresponds to a pentagon

Thus, we get the 2-frame of the manifold M_{top}^{dn}. This manifold might be disconnected.

Definition 2 The *topological n-strand braid groups* of the manifold M are the fundamental groups

$$B_t(M,n)_j = \pi_1((M_{\text{top}}^{dn})_j), \quad j = 1, \ldots, q.$$

2.3 Combinatorial Manifold of Triangulations

In Sects. 2.1 and 2.2 we constructed manifolds of triangulations basing on the notion of *Delaunay triangulation*. We did not actually discuss the *existence* of such triangulation for a given manifold and a given set of points, requesting only that the number of points is sufficiently large and points are sufficiently dense. It turns out that this condition is not always sufficient, as shows in [1]. Even for a large number of points and a Riemannian manifold with metric arbitrary close to Euclidian one, there may not exist a Delaunay triangulation.

One may impose additional restrictions on the vertex set or on the manifold itself to overcome this difficulty, but to work with the most general situation we shall consider the third notion of manifolds of triangulations: the *combinatorial*

manifold of triangulations which we denote by M^{dn}_{comb}. We construct the 2-frame of the manifold M^{dn}_{comb} which is needed to get the fundamental group.

First, we fix n points on the manifold M^d and consider triangulations of the manifold with vertices in those points. Now we do not restrict ourselves to Delaunay triangulations, but consider all triangulations of the manifold. Moreover, we work with *equivalence classes* of triangulations modulo the following relation: two triangulations T_1, T_2 with (fixed) vertices $v_1, \ldots, v_n$ are said to be *equivalent* if and only if for each pair $(i, j) \in \bar{n}$ the vertices v_i, v_j are connected by an edge of the triangulation T_1 if and only if they are connected by an edge of the triangulation T_2. From now on saying "triangulation" we mean such equivalence class of triangulations. Those triangulations are the vertices (0-cells) of the frame of the manifold M^{dn}_{comb}.

Two triangulations are connected by an edge (a 1-cell of the frame) if and only if they differ by a *flip*—a Pachner move [11].

Finally, the 2-cells of the frame are chosen to correspond to the relations of the groups Γ^k_n (see Sect. 3). There are two types of such relations. One relation corresponds to the far commutativity, i.e., for any two *independent* flips α, β (flips related to different edges) we define a quadrilateral consisting of subsequent flips corresponding to $\alpha, \beta, \alpha^{-1}, \beta^{-1}$. The other relations correspond to all possible simplicial polytopes with $d + 3$ vertices inscribed in the unit sphere. With each polytope of such sort, we associate a relation of length $d + 3$ as shown in Sect. 3.

Definition 3 The *combinatorial n-strand braid groups* of the manifold M are the fundamental groups

$$B_c(M, n)_j = \pi_1((M^{dn}_{\text{comb}})_j), \quad j = 1, \ldots, q.$$

3 The Groups Γ^k_n

In the present section we consider the case of the manifold $M^d = \mathbb{R}^d$ being a vector space. The case of Euclidian space gives groups which are important for the general case. In the present paper we deal mostly with the case of $d \geq 3$. The very interesting initial case $d = 2$ is studied in [4]. We start with the case $d = 3$.

3.1 The Case d = 3

Consider a configuration of n points in general position in $\mathbb{R}^3$. We can think of these points as lying in a fixed tetrahedron $ABCD$. The points induce a unique Delaunay triangulation of the tetrahedron: four points form a simplex of the triangulation if and only if there is no other points inside the sphere circumscribed over these points. The triangulation transforms when the points move in the space.

In order to avoid degenerate Delaunay triangulations we exclude configurations where four points lie on one circle (intersection of a plane and a sphere).

Transformations of the combinatorial structure of the Delaunay triangulation correspond to configurations of codimension 1 when five points lie on a sphere which does not contain any points inside. At this moment two simplices of the triangulation are replaced with three simplices as shown in Fig. 3 (or vice versa). This transformation is called a *2-3 Pachner move*.

To trace the evolution of triangulations that corresponds to a dynamics of the points we attribute a generator to each Pachner move. For the move that replace the simplices $iklm$, $jklm$ with the simplices $ijkl$, $ijkm$, $ijlm$ in Fig. 3 we use the generator $a_{ij,klm}$. Note that (1) we can split the indices into two subsets according to the combinatorics of the transformation; (2) the generator $a_{ij,klm}$ is not expected to be involutive because it changes the number of simplices of the triangulation.

The relations on generators $a_{ij,klm}$ correspond to configurations of codimension 2 which occurs when either (1) six points lie on the same sphere with empty interior, or (2) there are two spheres with five points on each of them, or (3) five points on one sphere compose a codimension 1 configuration.

The last case means that the convex hull of the five points has a quadrilateral face (Fig. 4). The vertices of this face lie on one circle so we exclude this configuration.

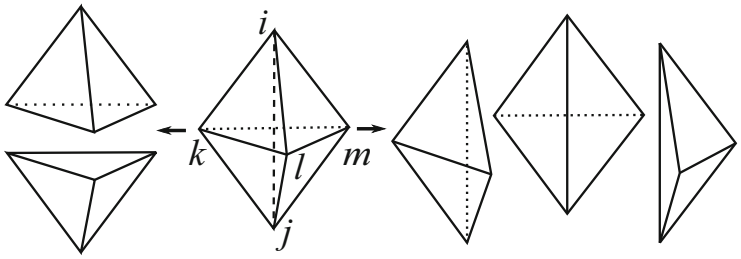


Fig. 3 A Pachner move

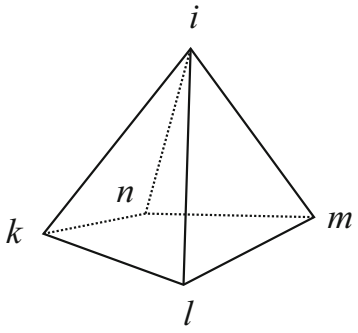


Fig. 4 A quadrilateral pyramid

If there are two different spheres with five points on each of them, then there is no simplex inscribed into the both spheres (otherwise its vertices would belong to one plane which contains the intersection of the spheres). Hence, the simplices inscribed into the first spheres and the simplices inscribed into the second one have no common internal points. For each sphere we can suppose that the faces of the convex hull of the inscribed simplices has only triangular faces; in the other case, that would be a configuration of codimension greater than two. So the convex hulls can have at most one common face, so we can transform them independently. This gives us a commutation relation

$$a_{ij,klm}a_{i'j',k'l'm'} = a_{i'j',k'l'm'}a_{ij,klm}, \tag{1}$$

where

$$\begin{aligned} \left|\{i, j, k, l, m\} \cap \left\{i', j', k', l', m'\right\}\right| &< 4, \\ \left|\{i, j\} \cap \left\{i', j', k', l', m'\right\}\right| &< 2, \end{aligned} \tag{2}$$

and

$$\left|\left\{i', j'\right\} \cap \{i, j, k, l, m\}\right| < 2. \tag{3}$$

Consider now the case of six points on one sphere. The convex hull of these points is a convex polyhedron. The polyhedron must have only triangular faces; otherwise, there is an additional linear condition (four points lie on one plane) which raises the codimension beyond 2. There are two such polyhedra (Fig. 5).

For the octahedron on the left we specify the geometrical configuration assuming that the orthogonal projection along the direction ij maps the points i and j near

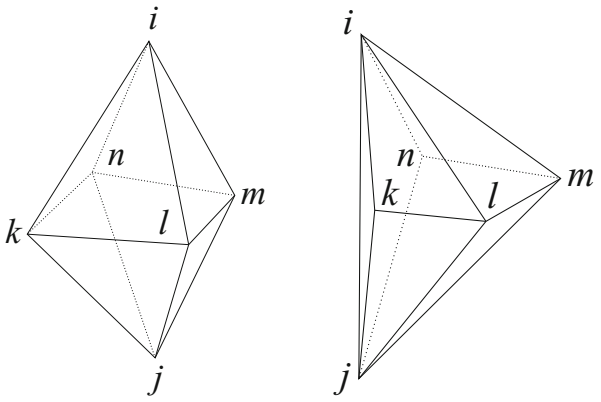


Fig. 5 Convex polyhedra with triangular faces and 6 vertices

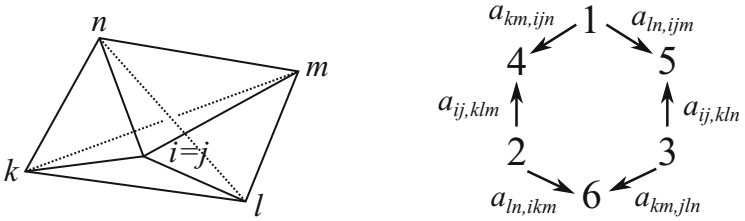


Fig. 6 Orthogonal projection and the triangulation graph for the octahedron

the projection of the edge kl and the line ln is higher than km if you look from i to j (Fig. 6 (left)). Note that we live in $\mathbb{R}^3$ which we may consider as oriented. In this case we have six triangulations:

1. $ijkl, ijkn, ijlm, ijmn$;
2. $iklm, ikmn, jklm, jkmn$;
3. $ikln, ilmn, jkln, jlmn$;
4. $ijkl, ijkm, ijlm, ikmn, jkmn$;
5. $ijkl, ijkn, ijln, ilmn, jlmn$;
6. $ikln, ilmn, jklm, jkmn, klmn$.

The first three have four simplices each, the last three have five simplices each. The Pachner moves between the triangulations are shown in Fig. 6 (right). Thus, we have the relation

$$a_{km,ijn}a_{ij,klm}^{-1}a_{ln,ikm}a_{km,jln}^{-1}a_{ij,kln}a_{ln,ijm}^{-1} = 1. \tag{4}$$

In the case of the shifted octahedron (Fig. 5 (right)) we assume the line ln lies higher than the line km when one looks from the vertex i to the vertex j (Fig. 7 (left)). Then we have six triangulations:

1. $ijkl, ijkm, ijmn$;
2. $ijkl, ijln, ilmn, jlmn$;
3. $ijkm, ijmn, iklm, jklm$;
4. $ijkn, ikln, ilmn, jkln, jlmn$;
5. $ijkn, iklm, ikmn, jklm, jkmn$;
6. $ijkn, ikln, ilmn, jklm, jkmn, klmn$.

The Pachner moves between the triangulations are shown in Fig. 7 (right). This gives us the relation

$$a_{ln,ijm}a_{kn,ijl}a_{km,jln} = a_{km,ijl}a_{kn,ijm}a_{ln,ikm}. \tag{5}$$

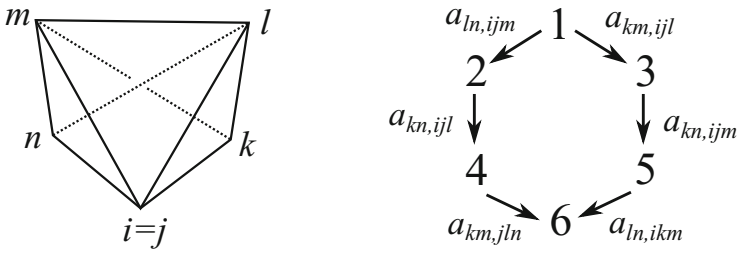


Fig. 7 Orthogonal projection and the triangulation graph of the shifted octahedron

Definition 4 The group Γ_n^5 is the group with generators

$$\left\{ a_{ij,klm} \mid \{i, j, k, l, m\} \in \bar{n},\ |\{i, j, k, l, m\}| = 5 \right\}$$

and relations

1. $a_{ij,klm} = a_{ji,klm} = a_{ij,lkm} = a_{ij,kml}$,
2. $a_{ij,klm}a_{i'j',k'l'm'} = a_{i'j',k'l'm'}a_{ij,klm}$, for

$$\left|\{i, j\} \cap \left\{i', j', k', l', m'\right\}\right| < 2, \quad \left|\{k, l, m\} \cap \left\{i', j', k', l', m'\right\}\right| < 3,$$

and

$$\left|\left\{i', j'\right\} \cap \{i, j, k, l, m\}\right| < 2, \quad \left|\left\{k', l', m'\right\} \cap \{i, j, k, l, m\}\right| < 3$$

3. $a_{km,ijn}a_{ij,klm}^{-1}a_{ln,ikm}a_{km,jln}^{-1}a_{ij,kln}a_{ln,ijm}^{-1} = 1$ for distinct i, j, k, l, m, n,
4. $a_{ln,ijm}a_{kn,ijl}a_{km,jln} = a_{km,ijl}a_{kn,ijm}a_{ln,ikm}$ for distinct i, j, k, l, m, n.

The group Γ_n^5 can be used to construct invariants of braids and dynamic systems. Consider the configuration space $\tilde{C}_n(\mathbb{R}^3)$ which consists of nonplanar n-tuples $(x_1, x_2, \ldots, x_n)$ of points in $\mathbb{R}^3$ such that for all distinct i, j, k, l the points x_i, x_j, x_k, x_l do not lie on the same circle.

We construct a homomorphism from $\pi_1(\tilde{C}_n(\mathbb{R}^3))$ to Γ_n^5. Let

$$\alpha = (x_1(t), \ldots, x_n(t)), \quad t \in [0, 1],$$

be a loop in $\tilde{C}_n(\mathbb{R}^3)$. For any t the set $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))$ determines a Delaunay triangulation $T(t)$ of the polytope $\operatorname{conv}\mathbf{x}(t)$. If α is in general position there will be a finite number of moments $0 < t_1 < t_2 < \ldots < t_N < 1$ when the combinatorial structure of $T(t)$ changes, and for each p the transformation of the triangulation at the moment t_p will be the Pachner move on simplices with vertices i_p, j_p, k_p, l_p, m_p. We assign to this move the generator $a_{i_p j_p, k_p l_p m_p}$ or $a_{i_p j_p, k_p l_p m_p}^{-1}$ and denote $\phi(\alpha) = \prod_{p=1}^{N} a_{i_p j_p, k_p l_p m_p} \in \Gamma_n^5$.

Theorem 1 *The homomorphism* $\phi\colon \pi_1(\tilde{C}_n(\mathbb{R}^3)) \to \Gamma_n^5$ *is well defined.*

Proof We need to show that the element $\phi(\alpha)$ does not depend on the choice of the representative α in a given homotopy class. Given a homotopy in general position $\alpha(\tau)$, $\tau \in [0,1]$, of loops in $\tilde{C}_n(\mathbb{R}^3)$, the transformations of the words $\phi(\alpha(\tau))$ correspond to point configurations of codimension 2, considered above the definition of Γ_n^5, and thus are counted by the relations in the group Γ_n^5. Therefore, the element $\phi(\alpha(\tau))$ of the group Γ_n^5 remains the same when τ changes. □

3.2 General Case

The braids in higher dimensional Euclidean spaces were defined by the second named author in [5, 6]. For our needs we modify that definition slightly.

Consider the configuration space $\tilde{C}_n(\mathbb{R}^{k-2})$, $4 \le k \le n$, consisting of n-point configurations $\mathbf{x} = (x_1, x_2, \dots, x_n)$ in $\mathbb{R}^{k-2}$, such that $\dim \operatorname{conv} \mathbf{x} = k-2$ and there are no $k-1$ points which lie on one $(k-4)$-dimensional sphere (the intersection of a $(k-3)$-sphere and a hyperplane in in $\mathbb{R}^{k-2}$).

A configuration $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \tilde{C}_n(\mathbb{R}^{k-2})$ determines a Delaunay triangulation of $\operatorname{conv} \mathbf{x}$ which is unique when $\mathbf{x}$ is in general position. If the vertices $x_{i_1}, \dots, x_{i_{k-1}}$ span a simplex of the Delaunay triangulation then the interior of the circumscribed sphere over these points does not contain other points of the configuration. The inverse statement is true when $\mathbf{x}$ is generic. The condition that no $k-1$ points lie on one $(k-4)$-dimensional sphere ensures that there are no degenerate simplices in the Delaunay triangulation.

Let $\alpha = (x_1(t), \dots, x_n(t))$, $t \in [0,1]$, be a loop in $\tilde{C}_n(\mathbb{R}^{k-2})$. For any t the configuration $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$ determines a Delaunay triangulation $T(t)$ of the polytope $\operatorname{conv} \mathbf{x}(t)$. If α is generic then there will be a finite number of moments $0 < t_1 < t_2 < \dots < t_L < 1$ when the combinatorial structure of $T(t)$ changes. We shall call such configurations *singular*.

For each singular configuration $\mathbf{x}(t_i)$, either one simplex degenerates and disappears on the boundary $\partial \operatorname{conv} \mathbf{x}(t)$ or the Delaunay triangulation is not unique. That means that there is a sphere in $\mathbb{R}^{k-2}$ which contains k points of $\mathbf{x}$ on it and no points of $\mathbf{x}$ inside. Assuming $\mathbf{x}$ is generic, the span of these k points is a simplicial polytope. Below we shall count only the latter type of singular configurations.

The simplicial polytopes in $\mathbb{R}^{k-2}$ with k vertices are described in [3]. Each of them is the join $\Delta_P * \Delta_Q$ of simplices $\Delta_P = \operatorname{conv}(x_{p_1}, \dots, x_{p_l})$ of dimension $l-1 \ge 1$ and $\Delta_Q = \operatorname{conv}(x_{q_1}, \dots, x_{q_{k-l}})$ of dimension $k-l-1 \ge 1$ such that the intersection $\operatorname{relint}(x_{p_1}, \dots, x_{p_l}) \cap \operatorname{relint}(x_{q_1}, \dots, x_{q_{k-l}})$ consists of one point.

We recall that the *join* of two sets $X, Y \subset \mathbb{R}^m$ is defined as

$$X * Y = \{\, \lambda x + (1-\lambda) y \mid x \in X,\ y \in Y,\ \lambda \in [0,1] \,\}$$

and the *relative interior* of a finite set $X \subset \mathbb{R}^m$ is defined as

$$\operatorname{relint} X = \left\{ \sum_{x \in X} \lambda_x x \mid \forall x \ \lambda_x > 0, \sum_{x \in X} \lambda_x = 1 \right\}.$$

The polytope $\Delta_P * \Delta_Q$ has two triangulations:

$$T_P = \left\{ \mathbf{x}_{P \cup Q} \setminus \{x_p\} \right\}_{p \in P} = \left\{ x_{p_1} \cdots x_{p_{i-1}} x_{p_{i+1}} \cdots x_{p_l} x_{q_1} \cdots x_{q_{k-l}} \right\}_{i=1}^{l}$$

and

$$T_Q = \left\{ \mathbf{x}_{P \cup Q} \setminus \{x_q\} \right\}_{q \in Q} = \left\{ x_{p_1} \cdots x_{p_l} x_{q_1} \cdots x_{q_{i-1}} x_{q_{i+1}} \cdots x_{q_{k-l}} \right\}_{i=1}^{k-l}.$$

Here $P = \{p_1, \ldots, p_l\}$, $Q = \{q_1, \ldots, q_{k-l}\}$ and $\mathbf{x}_J = \{x_j\}_{j \in J}$ for any $J \subset \{1, \ldots, n\}$.

The condition $\operatorname{relint}(\mathbf{x}_P) \cap \operatorname{relint}(\mathbf{x}_Q) = \{z\}$ implies $P \cap Q = \emptyset$. Thus, when the configuration $\mathbf{x}(t)$ goes over a singular value t_i, $i = 1, \ldots, L$, in the Delaunay triangulation simplices T_{P_i} are replaced by simplices T_{Q_i} for some subsets P_i, $Q_i \subset \{1, \ldots, n\}$, $P_i \cap Q_i = \emptyset$, $|P_i|, |Q_i| \geq 2$, $|P_i \cup Q_i| = k$. This transformation is called a *Pachner move*. The pair of subsets (P_i, Q_i) we call the *type* of the Pachner move. We assign to the transformation the letter a_{P_i, Q_i}.

Hence, the loop α produces a word

$$\Phi(\alpha) = \prod_{i=1}^{L} a_{P_i, Q_i} \tag{6}$$

in the alphabet

$$\mathcal{A}_n^k = \left\{ a_{P,Q} \mid P, Q \subset \{1, \ldots, n\}, \ P \cap Q = \emptyset, \ |P \cup Q| = k, \ |P|, |Q| \geq 2 \right\}.$$

Now let us consider a generic homotopy α_s, $s \in [0, 1]$ between two generic loops α_0 and α_1. A loop $\alpha_s = \{\mathbf{x}(s, t)\}_{t \in [0,1]}$ can contain a configuration of codimension 2. This means that for some t the configuration $\mathbf{x}(s, t)$ has two different k-tuples of points, such that each of them lies on a sphere whose interior contains no points of $\mathbf{x}(s, t)$. If these spheres do not coincide then their intersection contains at most $k-2$ points (the intersection can not contain $k-1$ points because $\mathbf{x}(s, t) \in \tilde{C}_n(\mathbb{R}^{k-2})$). Hence, the simplices involved in one Pachner move can not be involved in the other one, so the Pachner moves can be fulfilled in any order.

If the k-tuples of points lie on one sphere, there is a sphere with $k+1$ points of $\mathbf{x}(s, t)$ on it and its interior contains no points of $\mathbf{x}(s, t)$. These $k+1$ points span a simplicial polytope in $\mathbb{R}^{k-2}$. Such polytopes are described in [3]. The description uses the notion of *Gale transform*.

Let $X = \{x_1, \ldots, x_n\}$ be a set of n points in $\mathbb{R}^d$ such that $\dim \operatorname{conv} X = d$. Then $n \geq d + 1$. Let $x_i = (x_{1i}, \ldots, x_{di}) \in \mathbb{R}^d$, $i = 1, \ldots, n$, be the coordinates of the points of X. The matrix

$$M = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{d1} & x_{d2} & \cdots & x_{dn} \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

has rank $d + 1$. Then the dimension of the space $\ker M = \{ b \in \mathbb{R}^n \mid Mb = 0 \}$ of dependencies between columns of M is equal to $n - (d + 1)$. Take any basis $b_j = (b_{j1}, b_{j2}, \ldots b_{jn})$, $j = 1, \ldots, n - d - 1$, of $\ker M$ and write it in matrix form

$$B = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \cdots & \cdots & \cdots \\ b_{n-d-1,1} & \cdots & b_{n-d-1,n} \end{pmatrix}. \tag{7}$$

The columns of the matrix B form a set $Y = y_1, \ldots, y_n$, $y_i = (b_{1i}, \ldots, b_{n-d-1,i})$, in $\mathbb{R}^{n-d-1}$. The set Y is called a *Gale transform* of the point set X. Gale transforms which correspond to different bases of $\ker M$ are linearly equivalent. The vectors of the Gale transform Y may coincide.

Let $Y = y_1, \ldots, y_n$ be a Gale transform of X. The set

$$\bar{Y} = \{\bar{y}_1, \ldots, \bar{y}_n\}, \quad \bar{y}_i = \begin{cases} \frac{y_i}{\|y_i\|}, & y_i \neq 0, \\ 0, & y_i = 0 \end{cases}$$

is called a *Gale diagram* of the point set X. It is a subset of $S^{n-d-2} \cup \{0\}$.

Denote $N = \{1, \ldots, n\}$. Two subsets $\bar{Y} = \{\bar{y}_1, \ldots, \bar{y}_n\}$ and $\bar{Y}' = \{\bar{y}'_1, \ldots, \bar{y}'_n\}$ in $S^{n-d-2} \cup \{0\}$ are called *equivalent* if there is a permutation σ of N such that for any $J \subset N$

$$0 \in \operatorname{relint} \bar{Y}_J \iff 0 \in \operatorname{relint} \bar{Y}'_{\sigma(J)}.$$

Here we denote $\bar{Y}_J = \{\bar{y}_i\}_{i \in J}$ and $\bar{Y}'_J = \{\bar{y}'_i\}_{i \in J}$.

The properties of Gale diagrams can be summarized as follows [3].

Theorem 2

1. *Let X be a set of n points which are vertices of some polytope P in $\mathbb{R}^d$ and $\bar{Y}$ be its Gale diagram. Then the set of indices $J \subset N$ defines a face of P if and only if* $0 \in \operatorname{relint} Y_{N \setminus J}$.

2. *Let X and X' be sets of vertices of polytopes P and P', $|X| = |X'|$, and $\bar{Y}$ and $\bar{Y}'$ be their Gale diagrams. Then P and P' are combinatorially equivalent if and only if $\bar{Y}$ and $\bar{Y}'$ are equivalent.*
3. *For any n-point set $\bar{Y} \in S^{n-d-2} \cup \{0\}$ such that $\bar{Y}$ spans $\mathbb{R}^{n-d-1}$ and 0 lies in the interior of* conv $\bar{Y}$*, there is an n-point set X in $\mathbb{R}^d$ such that $\bar{Y}$ is a Gale diagram of X.*

The theorem implies (see [3]) that simplicial polytopes with $k+1$ vertices in $\mathbb{R}^{k-2}$ are in a bijection with standard Gale diagrams in $\mathbb{R}^2$ (defined uniquely up to isometries of the plane).

A *standard Gale diagram* of order $l = k+1$ is a subset $\bar{Y}$, $|\bar{Y}| = l$, of the vertices set $\{e^{\pi i p/l}\}_{p=0}^{2l-1}$ of the regular $2l$-gon inscribed in the unit circle S^1, such that:

1. any diameter of S^1 contains at most one point of $\bar{Y}$;
2. for any diameter of S^1, any open half-plane determined by it contains at least two points of $\bar{Y}$.

The first property means the corresponding polytope is simplicial, the second means any of the $k+1$ vertices of the polytope is a face.

The number c_l of standard Gale diagrams of order l is equal to

$$c_l = 2^{[(l-3)/2]} - \left[\frac{l+1}{4}\right] + \frac{1}{4l} \sum_{h:\ 2\nmid h \mid l} \varphi(h) \times 2^{l/h},$$

where $l = 2^{a_0} \prod_{i=1}^{t} p_i^{a_i}$ is the prime decomposition of l, and φ is Euler's function. For small l the numbers are $c_5 = 1$, $c_6 = 2$, $c_7 = 5$, see Fig. 8.

Let us describe the triangulations of the simplicial polytopes with $k+1$ vertices in $\mathbb{R}^{k-2}$.

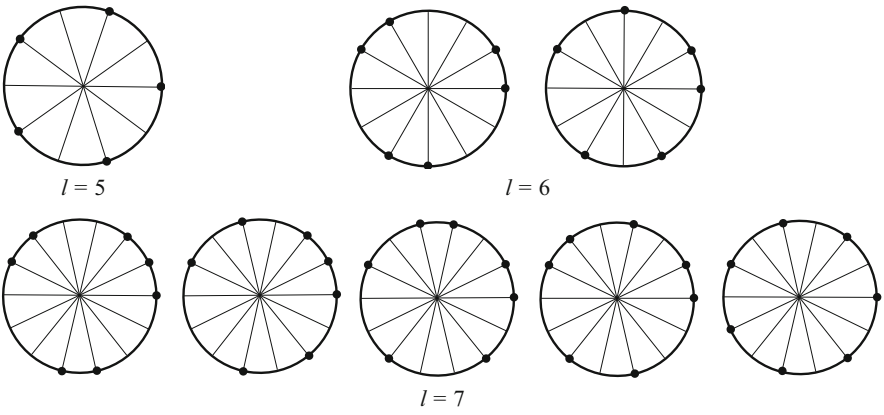


Fig. 8 Standard Gale diagrams of small order

Let $X = \{x_1, \ldots, x_n\}$ be a subset in $\mathbb{R}^d$, so $x_i = (x_{i1}, \ldots, x_{id})$, $i = 1, \ldots, n$. Let $P = \operatorname{conv} X$ be the convex hull of X, assume that $\dim P = d$. A triangulation T of the polytope P with vertices in X is called *regular* if there is a height function $h\colon X \to \mathbb{R}$ such that T is the projection of the lower convex of the lifting $X^h = \{x_1^h, \ldots, x_n^h\} \subset \mathbb{R}^{d+1}$, where $x_i^h = (x_i, h(x_i))$. This means that a set of indices $J \subset \{1, \ldots, n\}$ determines a face of T if and only if there exists a linear functional ϕ on $\mathbb{R}^{d+1}$ such that $\phi(0, \ldots, 0, 1) > 0$ and $J = \left\{ i \mid \phi(x_i^h) = \min_{x^h \in X^h} \phi(x^h) \right\}$. In case T is regular we write $T = T(X, h)$. Any generic height function induces a regular triangulation.

The Delaunay triangulation of X is regular with the height function $h\colon \mathbb{R}^d \to \mathbb{R}$, $h(z) = \|z\|^2 = \sum_{i=1}^d z_i^2$ if $z = (z_1, \ldots, z_d) \in \mathbb{R}^d$.

A height function $h\colon X \to \mathbb{R}$ can be regarded as a vector $h = (h_1, \ldots, h_n) \in \mathbb{R}^n$ where $h_i = h(x_i)$. Denote $\beta(h) = Bh \in \mathbb{R}^{n-d-1}$ where B is the matrix (7) used to define a Gale transform of X. Let $\bar{Y} = \{\bar{y}_1, \ldots, \bar{y}_n\}$ be a Gale diagram of X. Convex cones generated by the subsets of $\bar{Y}$ split the space $\mathbb{R}^{n-d-1}$ into a union of conic cells. A relation between the triangulation $T(X, h)$ and the Gale diagram $\bar{Y}$ can be described as follows.

Theorem 3

1. *If $T(X, h)$ is a regular triangulation of X then $\beta(h)$ belongs to a conic cell of maximal dimension in the splitting of $\mathbb{R}^{n-d-1}$ induced by $\bar{Y}$.*
2. *Let $J \subset N = \{1, \ldots, n\}$. The set X_J spans a cells of the triangulation $T(X, h)$ if and only if $\beta(h) \in \operatorname{concone}(\bar{Y}_{N \setminus J})$, where $\operatorname{concone}(X_{N \setminus J})$ is the convex cone spanned by the set $\bar{Y}_{N \setminus J}$.*

Let P be a simplicial polytopes with $l = k + 1$ vertices in $\mathbb{R}^{l-3}$ and $\bar{Y} = \{\bar{y}_1, \ldots, \bar{y}_l\}$ be the corresponding standard Gale diagram. By Theorem 3 there are l different regular triangulation which correspond to open sectors between the rays spanned by the vectors of $\bar{Y}$. The graph whose vertices are combinatorial classes of triangulations of P and the edges are Pachner moves, is a cycle. Let us find which Pachner moves appear in this cycle.

We change the order of vertices of P (and, hence, the order of the points of $\bar{Y}$) so that the points $\bar{y}_1, \ldots, \bar{y}_l$ appear in this sequence when one goes counterclockwise on the unit circle. For each i denote $R_{\bar{Y}}(i)$ (respectively, $L_{\bar{Y}}(i)$) be the set of indices j of vectors $\bar{y}_j$ that lie in the right (respectively, left) open half-plane incident to the oriented line spanned by the vector $\bar{y}_i$. Then the Pachner move which occurs when the vector $\beta(h)$ passes $\bar{y}_i$ from right to left, will be marked with the letter $a_{R_{\bar{Y}}(i), L_{\bar{Y}}(i)} \in \mathcal{A}_l^{l-1}$. The Pachner moves of the whole cycle of triangulation give the word $w_{\bar{Y}} = \prod_{i=1}^l a_{R(i), L(i)}$.

Thus, we have the relation $w_{\bar{Y}} = 1$ that we should impose in order to make the words like $\Phi(\alpha)$ independent on resolutions of configurations of codimension 2.

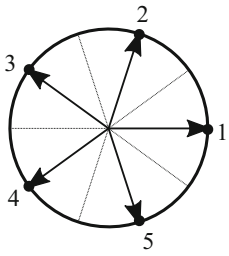


Fig. 9 Standard Gale diagram of order 5

Example 1 Consider the standard Gale diagram of order 5 (Fig. 9). Then we have $R(1) = \{4, 5\}$, $L(1) = \{2, 3\}$, $R(2) = \{1, 5\}$, $L(2) = \{3, 4\}$ etc. The corresponding word is equal to

$$w = a_{45,23}a_{15,34}a_{12,45}a_{23,15}a_{34,12}.$$

We can give now the definition of Γ_n^k groups.

Definition 5 Let $4 \le k \le n$. The group Γ_n^k is the group with the generators

$$a_{P,Q}, \quad P, Q \subset \{1, \dots, n\},\ P \cap Q = \emptyset,\ |P \cup Q| = k,\ |P|, |Q| \ge 2,$$

and the relations:

1. $a_{Q,P} = a_{P,Q}^{-1}$;
2. *far commutativity*: $a_{P,Q}a_{P',Q'} = a_{P',Q'}a_{P,Q}$ for each generators $a_{P,Q}$, $a_{P',Q'}$ such that

$$\begin{aligned} \left|P \cap (P' \cup Q')\right| &< |P|, & \left|Q \cap (P' \cup Q')\right| &< |Q|, \\ \left|P' \cap (P \cup Q)\right| &< \left|P'\right|, & \left|Q' \cap (P \cup Q)\right| &< \left|Q'\right|; \end{aligned}$$

3. $(k+1)$-*gon relations*: for any standard Gale diagram $\bar{Y}$ of order $k+1$ and any subset $M = \{m_1, \dots, m_{k+1}\} \subset \{1, \dots, n\}$

$$\prod_{i=1}^{k+1} a_{M_R(\bar{Y},i),M_L(\bar{Y},i)} = 1,$$

where $M_R(\bar{Y}, i) = \{m_j\}_{j \in R_{\bar{Y}}(i)}$, $M_L(\bar{Y}, i) = \{m_j\}_{j \in L_{\bar{Y}}(i)}$.

Example 2 Let us write the $(k+1)$-gon relations in Γ_n^k for small k.

The group Γ_n^4 has one pentagon relation

$$a_{m_4m_5,m_2m_3}a_{m_1m_5,m_3m_4}a_{m_1m_2,m_4m_5}a_{m_2m_3,m_1m_5}a_{m_3m_4,m_1m_2} = 1.$$

Our definition of Γ_n^4 differs slightly from the definition of these groups in [4]. To obtain the groups defined in [4] we should add relations $a^2_{m_1m_2,m_3m_4} = 1$ and

$$a_{m_1m_2,m_3m_4}a_{m_1m_2,m_5m_6} = a_{m_1m_2,m_5m_6}a_{m_1m_2,m_3m_4},$$
$$a_{m_1m_2,m_3m_4}a_{m_1m_5,m_2m_6} = a_{m_1m_5,m_2m_6}a_{m_1m_2,m_3m_4},$$

where $|\{m_3, m_4\} \cap \{m_5, m_6\}| \leq 1$.

The group Γ_n^5 has two hexagon relations

$$a_{m_5m_6,m_2m_3m_4}a_{m_1m_5m_6,m_3m_4}a_{m_1m_2,m_4m_5m_6}a_{m_1m_2m_3,m_5m_6}\cdot$$
$$a_{m_3m_4,m_1m_2m_6}a_{m_3m_4m_5,m_1m_2} = 1,$$

$$a_{m_5m_6,m_2m_3m_4}a_{m_1m_5m_6,m_3m_4}a_{m_1m_2m_6,m_4m_5}a_{m_1m_2m_3,m_5m_6}\cdot$$
$$a_{m_3m_4,m_1m_2m_6}a_{m_4m_5,m_1m_2m_3} = 1,$$

as we have seen in the previous subsection.

Theorem 4 *Formula* (6) *defines a correct homomorphism*

$$\Phi\colon \pi_1(\tilde{C}_n(\mathbb{R}^{k-2})) \to \Gamma_n^k.$$

Proof We need to show that the element $\Phi(\alpha)$ does not depend on the choice of the representative α in a given homotopy class. Given a generic homotopy $\alpha(\tau)$, $\tau \in [0, 1]$, of loops in $\tilde{C}_n(\mathbb{R}^{k-2})$, the transformations of the words $\Phi(\alpha(\tau))$ correspond to point configurations of codimension 2, considered above the definition of Γ_n^k, and thus are counted by the relations in the group Γ_n^k. Therefore, the element $\Phi(\alpha(\tau))$ of the group Γ_n^k remains the same when τ changes. □

4 Main Results

4.1 Γ_n^k-Valued Invariant of Braids on Manifolds

Theorem 5 *Let (M^d, g) be a manifold with a Riemannian metric and $B(M_g, n)_j$, $j = 1, \ldots, p$ be its geometrical braid groups. Then for any j there is a well defined mapping $B(M_g, n)_j \to \Gamma_n^{d+2}$.*

Proof By Definition 1, the space of geometrical braids on the manifold M^d is the fundamental group of the manifold of triangulations: $B(M_g, n)_j = \pi_1((M_g^{dn})_j)$. Hence, we need to construct a mapping from homotopy classes of loops in $(M_g^{dn})_j$ into the group Γ_n^{d+2}.

Let us fix a number j and consider a loop $\alpha = (x_1(t), \ldots, x_n(t)), t \in [0, 1]$ in $(M_g^{dn})_j$, where for each $i = 1, \ldots, n$ the point $x_i(t) \in M^d$. For each t the collection of points $(x_1(t), \ldots, x_n(t))$ defines a Delaunay triangulation $T(t)$ of the manifold M. If the loop α is in general position, then there is a finite number of moments $0 < t_1 < \ldots < t_N < 1$ when the combinatorial structure of the triangulation $T(t)$ changes. Each of those moments t_p corresponds to a codimension 1 configuration of points of the manifold M^d, and at each of them the triangulation undergoes some Pachner move, which transforms the triagulation simplices T_{P_p} into simplices T_{Q_p}, where $P_p, Q_p \subset \{1, \ldots, n\}$, $P_p \cap Q_p = \emptyset$, $\left|P_p\right|, \left|Q_p\right| \geq 2$ and $\left|P_p \cup Q_p\right| = d+2$. To each of those moments we attribute the element a_{P_p, Q_p} of the group Γ_n^{d+2}.

Thus a loop α produces a word $\Phi(\alpha) = \prod_{p=1}^{N} a_{P_p, Q_p}$.

Now we need to prove that this mapping $\Phi\colon \pi((M_g^{dn})_j) \to \Gamma_n^{d+2}$ is well defined: the element $\Phi(\alpha)$ does not depend on the choice of a representative in a given homotopy class.

Let $\alpha(\tau)$ be a generic homotopy, where $\tau \in [0, 1]$ and $\alpha(0) = \alpha$. With it we associate a family of words $\Phi(\alpha(\tau)) \in \Gamma_n^{d+2}$. Note that the word $\Phi(\alpha(\tau))$ changes whenever the loop passes through a point of codimension 2 in the manifold of triangulations M_g^{dn}. But such configurations of points in M^d exactly correspond to the relations of the group Γ_n^{d+2}. Therefore the element $\Phi(\alpha(\tau))$ of the group Γ_n^{d+2} remains the same when τ changes. □

Quite analogously, we get the following

Theorem 6 *Let M^d be a smooth manifold of dimension d and $B(M, n)_j$, $j = 1, \ldots, q$, be its topological braid groups. Then for any j there is a well defined mapping $B(M, n)_j \to \Gamma_n^{d+2}$.*

Proof By Definition 2, the topological braids group on the manifold M^d is the fundamental group of the manifold of triangulations: $B(M, n)_j = \pi_1((M_{\text{top}}^{dn})_j)$.

Consider a loop γ in the manifold M_{top}^{dn} and let γ be in general position. There is a finite number of intersections $\mathbf{x}_1, \ldots, \mathbf{x}_k$ between the loop γ and the codimension 1 stratum of the manifold of triangulations, where the triangulation undergoes a flip. Consider an intersection point $\mathbf{x}_i$ and its neighbourhood U. The intersection point $\mathbf{x}_i$ is a set of points $x_{1,i}, \ldots, x_{n,i} \in M$.

Consider two metrics g_α and g_β on the manifold M, yielding a structure of geometrical manifold of triangulations on the neighbourhood U. Let us denote the neighbourhood U equipped with the corresponding metric by U_{g_α} and U_{g_β}, respectively. As it was described in Sect. 2, equivalence of strata of the manifolds of triangulation $M_{g_\alpha}^{dn}$ and $M_{g_\beta}^{dn}$ is defined by an existence of a homeomorphism $h\colon M_{g_\alpha}^{dn} \to M_{g_\beta}^{dn}$ with certain properties.

The arc $l = \gamma \cap U \subset M_{\text{top}}^{dn}$ gives rise to an arc $l_1 \subset U_{g_\alpha}$ and an arc $l_2 \subset U_{g_\beta}$, and $l_2 = h(l_1)$ where h is the homeomorphism from the definition of cell equivalence. Likewise, the intersection point $X_i \in U$ gives a point $X_i' \in U_{g_\alpha}$ and a point $X_i'' \in U_{g_\beta}$. Naturally, those points are the intersections between the codimension 1 stratum of the corresponding manifold and the arcs l_1, l_2, respectively.

At the point $\mathbf{x}_i$ a certain Pachner move occurs. The type of this move is independent of the choice of metric on the manifold. Therefore, the Pachner moves occurring in the points $\mathbf{x}_i', \mathbf{x}_i''$ of the manifolds $U_{g_\alpha}, U_{g_\beta}$ are one and the same. Choosing the numbering of the n points which form the vertices of the triangulations, we hence obtain the same generator of the group Γ_n^{d+2} corresponding to this intersection point. Therefore the letter which appears in the word $\Phi(\gamma)$ does not depend on the choice of the metric.

Therefore the mapping Φ is defined on the fundamental group of the topological manifold of triangulations M_{top}^{dn} and due to Theorem 5 it does not depend on the choice of a representative in the homotopy class of a loop. Hence, it is a well defined mapping from $B(M,n)_j$ to the group Γ_n^{d+2}. □

Hence, we get invariants of smooth metrical manifolds and of topological manifolds: these invariants are the corresponding images of the fundamental group of the manifold of triangulations in the corresponding group Γ_n^k.

4.2 Triangulations of Polyhedra and Groups Γ_n^k

Triangulations of a given convex polyhedra are well studied. If we deal with stable triangulations, the set of those form a simplicial complex [2]. In the present section we show that they can be treated as elements of a certain group provided that some fixed triangulation plays the role of the unit element.

Let $\mathbf{x} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $n > d$, be the vertex set of a polytope W in general position, namely, any $d+1$ points of $\mathbf{x}$ do not belong to one hyperplane. Let $\mathcal{T}$ be the set of regular triangulations of W (see page 27). We can consider $\mathcal{T}$ as a graph whose vertices are regular triangulations and whose edges are Pachner moves. This graph is connected, see [8].

Choose a regular triangulation $T_0 \in \mathcal{T}$. For any other triangulation $T \in \mathcal{T}$ there is a path $\gamma = e_1 e_2 \ldots e_l$ from T_0 to T where each edge e_i is a Pachner move of type (P_i, Q_i), see page 24. We assign the word $\varphi(T) = \prod_{i=1}^{l} a_{P_i,Q_i} \in \Gamma_n^{d+2}$ to the triangulation T.

Let $Cay(\Gamma_n^{d+2})$ be the Cayley graph of the group Γ_n^{d+2} corresponding to the group presentation given in Definition 5.

Theorem 7 *The correspondence $T \mapsto \varphi(T)$ defines an embedding of the graph $\mathcal{T}$ into the graph $Cay(\Gamma_n^{d+2})$.*

Proof We must show first that the map φ is well-defined, i.e., that the element $\varphi(T)$ does not depend on the choice of a path connecting T_0 with T.

It follows from the theory of I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky [2, Chapter 7] (see also [10]) that the graph $\mathcal{T}$ is the edge graph of some polytope $\Sigma(W)$ (called the *secondary polytope* of W), see Fig. 10. The faces of $\Sigma(W)$ correspond to triangulations which coincide everywhere except for divisions of two subpolytopes with $d+2$ vertices or a subpolytope with $d+3$ vertices in W. Hence,

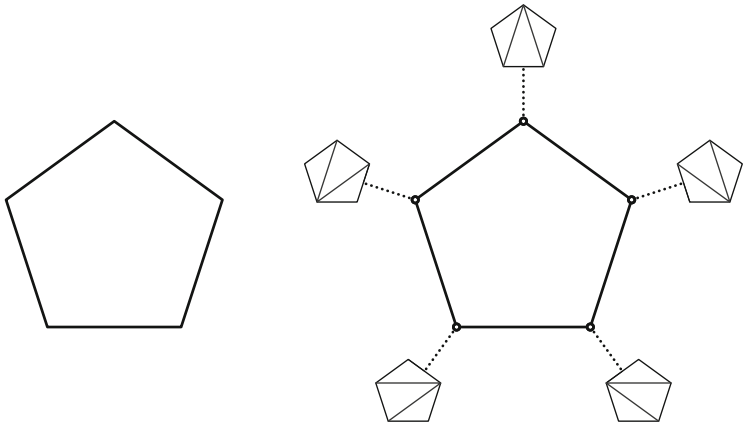


Fig. 10 A pentagon and its secondary polytope

the faces give exactly the relations in Γ_n^{d+2} which appear in Definition 5. Thus, two different paths γ and γ', connecting T_0 with some triangulation T, produce words $\varphi(\gamma)$ and $\varphi(\gamma')$ which differ by relations in Γ_n^{d+2}, so $\varphi(\gamma) = \varphi(\gamma') \in \Gamma_n^{d+2}$.

In order to proof that φ is injective we construct a left inverse map to it. A *formal l-dimensional simplex* is any subset $P \subset \{1, \dots, n\}$, $|P| = l + 1$. Let $C_l(n)$ be the linear space over $\mathbb{Z}_2$ whose basis consists of all the l-dimensional formal simplices. We can identify$C_l(n) \simeq \mathbb{Z}_2^{\binom{n}{l+1}}$ with the space of l-chains of the simplicial complex for a $(n-1)$-dimensional simplex whose vertices are marked with numbers $1, 2, \dots, n$. The differential $\partial\colon C_l(n) \to C_{l-1}(n)$ of this complex is defined by the formula $\partial(P) = \sum_{p \in P} P \setminus \{p\}$.

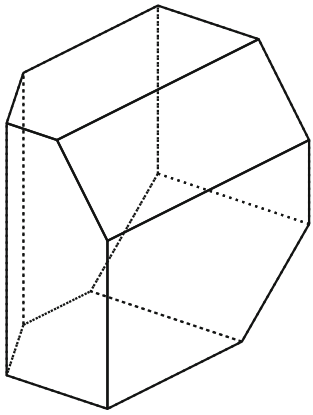
Any triangulation T of W defines a d-chain $c(T) = \sum_{P\colon \mathbf{x}_P \in T} P$ such that $\partial c(T) = \partial W$ where ∂W is the formal sum of $(d-1)$-dimensional faces forming the boundary of the polytope W. The triangulation T can be restored from its chain $c(T)$ as the support of the chain. Thus, we can identify triangulations of W with a subset of $C_d(n)$.

If a triangulation T' differs form a triangulation T by applying a Pachner move of type (P, Q) then $c(T') - c(T) = \partial(P \cup Q)$. Define a homomorphism $\psi\colon \Gamma_n^{d+2} \to C_d(n)$ by the formula

$$\psi(a_{P,Q}) = \partial(P \cup Q) = \sum_{i \in P \cup Q} (P \cup Q) \setminus \{i\}.$$

The map ψ annihilates the relations of Γ_n^{d+2} because they come from cycles of triangulations of some polytope with $d+3$ vertices (or two polytopes with $d+2$ vertices). The image of a relation is equal to the difference of the chains for the final triangulation and the initial one. But these triangulations coincide so the image is zero. Thus, ψ defines a correct homomorphism from Γ_n^{d+2} to $C_d(n) \simeq \mathbb{Z}_2^{\binom{n}{d+1}}$.

Fig. 11 Associahedron K_5



For any regular triangulation T consider a path $\gamma = e_1 e_2 \dots e_l$ which connects T_0 with T in $\mathcal{T}$. Any edge e_i of γ corresponds to a Pachner move of type (P_i, Q_i). Then

$$c(T) - c(T_0) = \sum_{i=1}^{l} \partial(P_i \cup Q_i) = \sum_{i=1}^{l} \psi(a_{P_i, Q_i}) = \psi(\varphi(T)).$$

Thus, the element $\varphi(T)$ determines the chain $c(T)$, hence, it uniquely determines the triangulation T. □

Example 3 Let $W \subset \mathbb{R}^2$ be a convex n-gon. Its secondary polytope $\Sigma(W)$ is the associahedron (Stasheff polytope) K_{n-1} [12], see Fig. 11. Thus, we have the following statement:

Corollary 1 *The graph $Cay(\Gamma_n^4)$ contains a subgraph isomorphic to the edge graph of the associahedron K_{n-1}.*

5 Other Developments: The Groups $\tilde{\Gamma}_n^k$

In the present section we provide constructions resembling the Γ_n^k groups but different in some aspects, which prove useful in certain situations where the Γ_n^k groups are not sufficient. Geometrically speaking, here we consider oriented triangulations. Therefore, the indices of the generators of the groups are not independent and do not freely commute as was seen, for example, in the groups Γ_n^5 (see Definition 4, first relation).

To be precise, we introduce the following:

Definition 6 Let $5 \le k \le n$. The group $\tilde{\Gamma}_n^k$ is the group with the generators

$$\left\{ a_{P,Q} \mid P,\ Q\text{—cyclically ordered subsets of } \{1,\dots,n\},\right.$$
$$\left. P \cap Q = \emptyset,\ |P \cup Q| = k,\ |P|, |Q| \ge 2 \right\},$$

and the following relations:

1, 2, 3. the relations 1, 2, 3 from Definition 5;

4. $a_{Q,P} = a_{Q',P'}$, where $Q = Q'$, $P = P'$ as unordered sets, and as cyclically ordered sets Q differs from Q' by one transposition and P differs from P' by one transposition.

These groups have the same connection to geometry and dynamics as the groups Γ_n^k defined above. To illustrate that, consider the following dynamical system.

Example 4 Let us have a dynamical system describing a movement of a point around the configuration of four points on one circle, see Fig. 12. Such system may be presented as the word in the group $\tilde{\Gamma}_6^5$

$$w = a_{35,164} a_{46,253}^{-1} a_{46,135} a_{35,246}^{-1}.$$

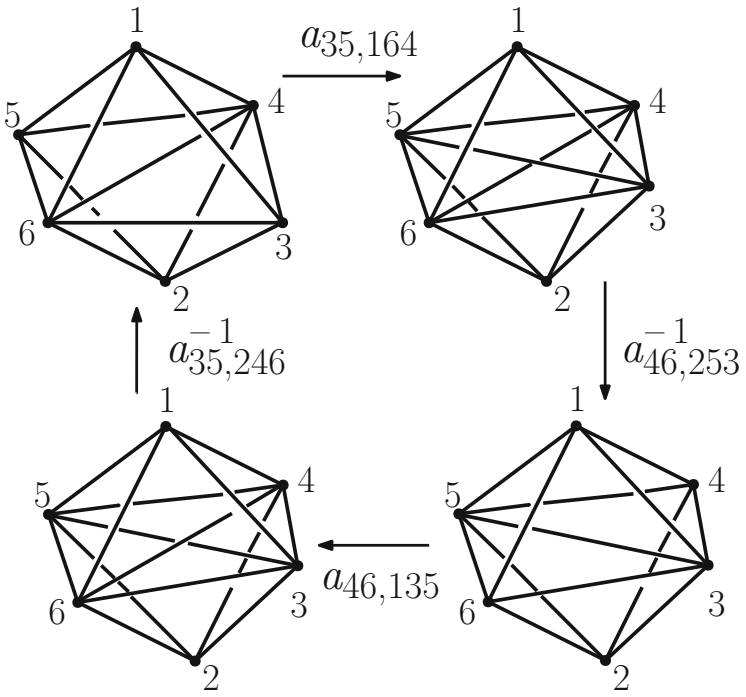


Fig. 12 A movement of a point around the configuration of four points on one circle

We can show that $\phi(\alpha)$ is nontrivial in the abelianisation $(\tilde{\Gamma}_6^5)_{ab} = \tilde{\Gamma}_6^5 / [\tilde{\Gamma}_6^5, \tilde{\Gamma}_6^5]$ of the group $\tilde{\Gamma}_6^5$. Computer calculations show that the group $(\tilde{\Gamma}_6^5)_{ab}$ can be presented as the factor of a free commutative group with 120 generators modulo $2 \times 6! = 1440$ relations that span a space of rank 90 if we work over $\mathbb{Z}_2$. Adding the element w to the relations increases the rank to 91, so the element w is nontrivial in $(\tilde{\Gamma}_6^5)_{ab}$, therefore it is nontrivial in $\tilde{\Gamma}_6^5$.

Hence, we have encountered a peculiar new effect in the behaviour of $\tilde{\Gamma}_n^5$ which is not the case of G_n^k. Certainly, the abelianisation of G_n^k is non-trivial and very easy to calculate since any generator enters each relation evenly many times. However, this happens not only for relations but for any words which come from braids. Thus, any abelianisations are trivial in interesting cases. For $\tilde{\Gamma}_n^k$, it is an interesting new phenomenon, and the invariants we have demonstrated so far are just the tip of the iceberg to be investigated further.

Note that both for the groups Γ_n^k and $\tilde{\Gamma}_n^k$ we have many invariants since the corank of those groups is big.

Acknowledgments The authors are very grateful to S. Kim for extremely useful discussions and comments.

The first named author was supported by the Russian Foundation for Basic Research (grant No. 19-01-00775-a, grant No. 20-51-53022). The second named author was supported by the Russian Foundation for Basic Research (grant No. 20-51-53022, grant No. 19-51-51004). The third named author was supported by the Russian Foundation for Basic Research (grant No. 18-01-00398-a, grant No. 19-51-51004). The work of V.O.M. was also funded by the development program of the Regional Scientific and Educational Mathematical Center of the Volga Federal District, agreement No. 075-02-2020.

References

1. Boissonnat, J.D., Dyer, R., Ghosh, A., Martynchuk, N.: An obstruction to Delaunay triangulations in Riemannian manifolds. Discrete Comput. Geom. **59**(1), 226–237 (2018)
2. Gelfand, I.M., Kapranov, M.M., Zelevinsky, A.V.: Discriminants, Resultants, and Multidimensional Determinants. Birkhäuser, Boston (1994)
3. Grünbaum, B.: Convex Polytopes, 2nd edn. Spriger, Berlin (2003)
4. Kim, S., Manturov, V.O.: Artin's braids, Braids for three space, and groups Γ_n^4 and G_n^k (2019). arXiv:1902.11238
5. Manturov, V.O.: The Groups G_n^k and fundamental groups of configuration spaces. J. Knot Theory Ramifications **26**(6), 1742004 (2017)
6. Manturov, V.O.: The groups G_{k+1}^k and fundamental groups of configuration spaces. In: Zelmanov, E.I., Shum, K.P., Kolesnikov, P.S. (eds.) Proceedings of the 3rd International Congress on Algebra and Combinatorics, pp. 310–323. World Scientific, Singapore (2019)
7. Manturov, V.O., Nikonov, I.M.: On braids and groups G_n^k. J. Knot Theory Ramifications **24**(13), 1541009 (2015)
8. Matveev, S.V.: Algorithmic Topology and Classification of 3-Manifolds. Springer, Berlin (2007)

9. Milnor, J.: Link groups. Ann. Math. **59**(2), 177–195 (1954)
10. Nabutovsky, A.: Fundamental group and contractible closed geodesics. Comm. Pure Appl. Math. **49**(12), 1257–1270 (1996)
11. Pachner, U.: P.L. homeomorphic manifolds are equivalent by elementary shellings. Eur. J. Comb. **12**(2), 129–145 (1991)
12. Stasheff, J.D.: Homotopy associativity of H-spaces. I, II. Trans. Amer. Math. Soc. **108**, 293–312 (1963)

A Proof of the Invariant-Based Formula for the Linking Number and Its Asymptotic Behaviour

Matt Bright, Olga Anosova, and Vitaliy Kurlin

Abstract In 1833 Gauss defined the *linking number* of two disjoint curves in 3-space. For open curves this double integral over the parameterised curves is real-valued and invariant modulo rigid motions or isometries that preserve distances between points, and has been recently used in the elucidation of molecular structures. In 1976 Banchoff geometrically interpreted the linking number between two line segments. An explicit analytic formula based on this interpretation was given in 2000 without proof in terms of 6 isometry invariants: the distance and angle between the segments and 4 coordinates specifying their relative positions. We give a detailed proof of this formula and describe its asymptotic behaviour that wasn't previously studied.

1 The Gauss Integral for the Linking Number of Curves

For any vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$, the *triple* product is $(\mathbf{u}, \mathbf{v}, \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) \cdot \mathbf{w}$.

Definition 1 (Gauss Integral for the Linking Number) For piecewise-smooth curves $\gamma_1, \gamma_2 \colon [0, 1] \to \mathbb{R}^3$, the *linking number* can be defined as the Gauss integral [7]

$$\mathrm{lk}(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int_0^1 \int_0^1 \frac{(\dot{\gamma}_1(t), \dot{\gamma}_2(s), \gamma_1(t) - \gamma_2(s))}{|\gamma_1(t) - \gamma_2(s)|^3} \, \mathrm{d}t \, \mathrm{d}s \,, \tag{1}$$

where $\dot{\gamma}_1(t)$, $\dot{\gamma}_2(s)$ are the vector derivatives of the 1-variable functions $\gamma_1(t)$, $\gamma_2(s)$.

M. Bright · O. Anosova · V. Kurlin (✉)
University of Liverpool, Liverpool, UK
e-mail: sgmbrigh@liverpool.ac.uk

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_3

The formula in Definition 1 gives an integer number for any closed disjoint curves γ_1, γ_2 due to its interpretation as the *degree* of the *Gauss map*

$$\Gamma(t,s) = \frac{\gamma_1(t) - \gamma_2(s)}{|\gamma_1(t) - \gamma_2(s)|} \colon S^1 \times S^1 \to S^2, \quad \text{i.e.,} \quad \deg \Gamma = \frac{\text{area}(\Gamma(S^1 \times S^1))}{\text{area}(S^2)},$$

where the area of the unit sphere is $\text{area}(S^2) = 4\pi$. This integer degree is the *linking number* of the 2-component link $\gamma_1 \sqcup \gamma_2 \subset \mathbb{R}^3$ formed by the two closed curves. Invariance modulo continuous deformation of $\mathbb{R}^3$ follows easily for closed curves—indeed, the function under the Gauss integral in Definition 1, and hence the integral itself, varies continuously under perturbations of the curves γ_1, γ_2. This should keep any integer value constant.

For open curves γ_1, γ_2, the Gauss integral gives a real but not necessarily integer value, which remains invariant under rigid motions or orientation-preserving isometries, see Theorem 1. In $\mathbb{R}^3$ with the Euclidean metric these are rotations, translations and reflections. The isometry invariance of the real-valued linking number for open curves has found applications in the study of molecules [1].

Any smooth curve can be well-approximated by a polygonal line, so the computation of the linking number reduces to a sum over line segments L_1, L_2. In 1976 Banchoff [3] has $\text{lk}(L_1, L_2)$ in terms of the endpoints of each segment, see details of this and other past work in Sect. 3.

In 2000 Klenin and Langowski [8] proposed a formula for the linking number $\text{lk}(L_1, L_2)$ of two straight line segments in terms of 6 isometry invariants of L_1, L_2, referring to a previous paper in which it was used without any detailed proof [17]. The paper [8] also does not provide details of the form's derivation.

The usefulness of an invariant based formula can be seen by considering the analogy with the simpler concept of the scalar (dot) product of vectors. The algebraic or *coordinate-based* formula expresses the scalar product of two vectors $\mathbf{u} = (x_1, y_1, z_1)$ and $\mathbf{v} = (x_2, y_2, z_2)$ as $\mathbf{u} \cdot \mathbf{v} = x_1x_2 + y_1y_2 + z_1z_2$, which in turn depend on the co-ordinates of their endpoints. However, the scalar product for high-dimensional vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ can also expressed in terms of only 3 parameters $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| \cdot |\mathbf{v}| \cos \angle(\mathbf{u}, \mathbf{v})$. The two lengths $|\mathbf{u}|, |\mathbf{v}|$ and the angle $\angle(\mathbf{u}, \mathbf{v})$ are isometry invariants of the vectors $\mathbf{u}, \mathbf{v}$. This second geometric or *invariant-based* formula makes it clear that $\mathbf{u} \cdot \mathbf{v}$ is an isometry invariant, while it is harder to show $\mathbf{u} \cdot \mathbf{v} = x_1x_2 + y_1y_2 + z_1z_2$ is invariant under rotations. It also provides other geometric insights that are hard to extract from the coordinate-based formula—for example, $\mathbf{u} \cdot \mathbf{v}$ oscillates as a cosine wave when the lengths $|\mathbf{u}|, |\mathbf{v}|$ are fixed, but the angle $\angle(\mathbf{u}, \mathbf{v})$ is varying.

In this paper, we provide a detailed proof of the invariant-based formula for the linking number in Theorem 2 and new corollaries in Sect. 6 formally investigating the asymptotic behaviour of the linking number, which wasn't ptreviously studied.

Our own interest in the asymptotic behaviour is motivated by the definition of the *periodic linking number* by Panagiotou [13] as an invariant of networks that are infinitely periodic in three directions, by calculating the infinite sum of the linking number between one line segment and all copies of another such segment. In [13]

there is a complex proof that this sum is convergent for 3-periodic structures, which could be simplified and improved by a new asymptotic analysis of the closed form.

2 The Outline of the Invariant-Based Formula and Consequences

We list key properties of $\mathrm{lk}(\gamma_1, \gamma_2)$ below which are frequently assumed without proof in other literature—we have provided a proof for these in the appendices.

Theorem 1 (Properties of the Linking Number) *The linking number defined by the Gauss integral in Definition 1 for curves γ_1, γ_2 has the following properties:*

(a) the linking number is symmetric: $\mathrm{lk}(\gamma_1, \gamma_2) = \mathrm{lk}(\gamma_2, \gamma_1)$*;*
(b) $\mathrm{lk}(\gamma_1, \gamma_2) = 0$ *for any curves γ_1, γ_2 that belong to the same plane;*
(c) $\mathrm{lk}(\gamma_1, \gamma_2)$ *is independent of parameterisations of γ_1, γ_2 with fixed endpoints;*
(d) $\mathrm{lk}(-\gamma_1, \gamma_2) = -\mathrm{lk}(\gamma_1, \gamma_2)$*, where $-\gamma_1$ has the reversed orientation of γ_1;*
(e) the linking number $\mathrm{lk}(\gamma_1, \gamma_2)$ *is invariant under any scaling* $\mathbf{v} \to \lambda\mathbf{v}$ *for* $\lambda > 0$*;*
(f) $\mathrm{lk}(\gamma_1, \gamma_2)$ *is multiplied by* $\det M$ *under any orthogonal map* $\mathbf{v} \mapsto M\mathbf{v}$.

Our main Theorem 2 will prove an analytic formula for the linking number of any line segments L_1, L_2 in terms of 6 isometry invariants of L_1, L_2, which are introduced in Lemma 1. Simpler Corollary 1 expresses $\mathrm{lk}(L_1, L_2)$ for any *simple* orthogonal oriented segments L_1, L_2 defined by their lengths $l_1, l_2 > 0$ and initial endpoints O_1, O_2, respectively, with the Euclidean distance $d(O_1, O_2) = d > 0$, so that $\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{O_1O_2}$ form a positively oriented orthogonal basis whose signed volume $(\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{O_1O_2}) = l_1 l_2 d$ is the product of the lengths, see the first picture in Fig. 1.

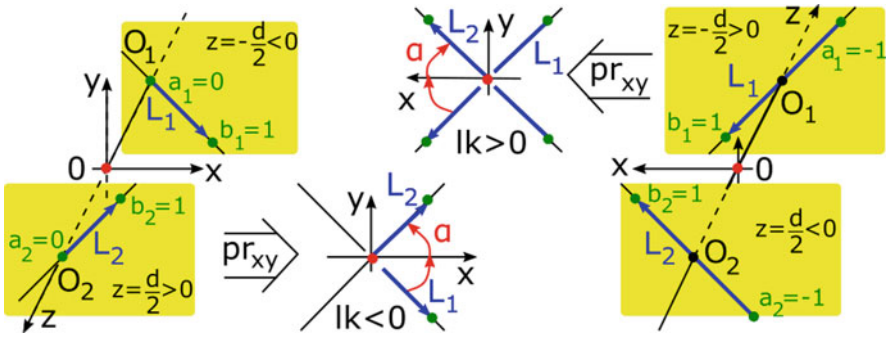


Fig. 1 Each line segment L_i is in the plane $\{z = (-1)^i d/2\}$, $i = 1, 2$. Left: signed distance $d > 0$, the endpoint coordinates $a_1 = 0$, $b_1 = 1$ and $a_2 = 0$, $b_2 = 1$, the lengths $l_1 = l_2 = 1$. Right: signed distance $d < 0$, the endpoint coordinates $a_1 = -1$, $b_1 = 1$ and $a_2 = -1$, $b_2 = 1$, so $l_1 = l_2 = 2$. In both middle pictures $\alpha = \pi/2$ is the angle from $\mathrm{pr}_{xy}(L_1)$ to $\mathrm{pr}_{xy}(L_2)$ with x-axis as the bisector

Corollary 1 (Linking Number for Simple Orthogonal Segments) *For any simple orthogonal oriented line segments $L_1, L_2 \subset \mathbb{R}^3$ with lengths l_1, l_2 and a distance d as defined above, the linking number is*

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi} \arctan\left(\frac{l_1 l_2}{d\sqrt{l_1^2 + l_2^2 + d^2}} \right).$$

The above expression is a special case of general formula (3) for $a_1 = a_2 = 0$ and $\alpha = \pi/2$.

If $l_1 = l_2 = l$, the linking number in Corollary 1 becomes

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi} \arctan \frac{l^2}{d\sqrt{2l^2 + d^2}}.$$

If $l_1 = l_2 = d$, then

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi} \arctan \frac{1}{\sqrt{3}} = -\frac{1}{24}.$$

Corollary 1 implies that the linking number is in the range $(-1/8, 0)$ for any simple orthogonal segments with $d > 0$, which wasn't obvious from Definition 1.

If L_1, L_2 move away from each other, then

$$\lim_{d \to +\infty} \mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi} \arctan 0 = 0.$$

Alternatively, if segments with $l_1 = l_2 = l$ become infinitely short, the limit is again zero:

$$\lim_{l \to 0} \mathrm{lk}(L_1, L_2) = 0$$

for any fixed d. The limit

$$\lim_{x \to +\infty} \arctan x = \frac{\pi}{2}$$

implies that if segments with $l_1 = l_2 = l$ become infinitely long for a fixed distance d,

$$\lim_{l \to +\infty} \mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi} \arctan \frac{l^2}{d\sqrt{2l^2 + d^2}} = -\frac{1}{8}.$$

If we push segments L_1, L_2 of fixed (possibly different) lengths l_1, l_2 towards each other, the same limit similarly emerges:

$$\lim_{d\to 0} \mathrm{lk}(L_1, L_2) = -\frac{1}{8}.$$

See more general corollaries in Sect. 6.

3 Past Results about the Gauss Integral for the Linking Number

The survey [15] reviews the history of the Gauss integral, its use in Maxwell's description of electromagnetic fields [12], and its interpretation as the degree of a map from the torus to the sphere. In classical knot theory $\mathrm{lk}(\gamma_1, \gamma_2)$ is a topological invariant of a link consisting of closed curves $\gamma_1 \sqcup \gamma_2$, whose equivalence relation is ambient isotopy. This relation is too flexible for open curves which can be isotopically unwound, and hence doesn't preserve the Gauss integral for open curves γ_1, γ_2.

Computing the value of the Gauss integral directly from the parametric equation of two generic curves is only possible by approximation, but this problem is simplified when we consider simply straight lines. The first form of the linking number between two straight line segments in terms of their geometry is described by Banchoff [3]. Banchoff considers the projection of segments on to a plane orthogonal to some vector $\xi \in S^2$. The Gauss integral is interpreted as the fraction of the unit sphere covered by those directions of ξ for which the projection will have a crossing.

This interpretation was the foundation of a closed form developed by Arai [2], using van Oosterom and Strackee's closed formula for the solid angle subtended by a tetrahedron given by the origin of a sphere and three points on its surface.

An alternative calculation for this solid angle is given in [14] as a starting point for calculating further invariants of open entangled curves. This form does not employ geometric invariants, but was used in [8] to claim a formula (without a proof) similar to Theorem 2, which is proved in this paper with more corollaries in Sect. 6.

4 Six Isometry Invariants of Skewed Line Segments in 3-Space

This section introduces 6 isometry invariants, which uniquely determine positions of any line segments $L_1, L_2 \subset \mathbb{R}^3$ modulo isometries of $\mathbb{R}^3$, see Lemma 1.

It suffices to consider only *skewed* line segments that do not belong to the same 2-dimensional plane. If L_1, L_2 are in the same plane Π, for example if they are parallel, then $\dot{L}_1(t) \times \dot{L}_2(s)$ is orthogonal to any vector $L_1(t) - L_2(s)$ in the plane Π, hence $\mathrm{lk}(L_1, L_2) = 0$. We denote by $\bar{L}_1, \bar{L}_2 \subset \mathbb{R}^3$ the infinite oriented lines through L_1, L_2 respectively. In a plane with fixed coordinates x, y, all angles are measured anticlockwise from the positive x-axis.

Definition 2 (Invariants of Line Segments) Let $\alpha \in [0, \pi]$ be the angle between oriented line segments $L_1, L_2 \subset \mathbb{R}^3$. Assuming that L_1, L_2 are not parallel, there is a unique pair of parallel planes Π_i, $i = 1, 2$, each containing the infinite line $\bar{L}_i$ through the line segment L_i. We choose orthogonal x, y, z coordinates in $\mathbb{R}^3$ so that

(a) the horizontal plane $\{z = 0\}$ is in the middle between Π_1, Π_2, see Fig. 1;
(b) $(0, 0, 0)$ is the intersection of the projections $\mathrm{pr}_{xy}(\bar{L}_1), \mathrm{pr}_{xy}(\bar{L}_2)$ to $\{z = 0\}$;
(c) the x-axis bisects the angle α from $\mathrm{pr}_{xy}(\bar{L}_1)$ to $\mathrm{pr}_{xy}(\bar{L}_2)$, the y-axis is chosen so that α is anticlockwisely measured from the x-axis to the y-axis in $\{z = 0\}$;
(d) the z-axis is chosen so that x, y, z are oriented in the right hand way, then d is the *signed* distance from Π_1 to Π_2 (negative if $\overrightarrow{O_1 O_2}$ is opposite to the z-axis in Fig. 1).

Let a_i, b_i be the coordinates of the initial and final endpoints of the segments L_i in the infinite line $\bar{L}_i$ whose origin is $O_i = \Pi_i \cap (z\text{-axis}) = \big(0, 0, (-1)^i d/2\big)$, $i = 1, 2$.

The case of segments L_1, L_2 lying in the same plane $\Pi \subset \mathbb{R}^3$ can be formally covered by Definition 2 if we allow the signed distance d from Π_1 to Π_2 to be 0.

Lemma 1 (Parameterisation) *Any oriented line segments $L_1, L_2 \subset \mathbb{R}^3$ are uniquely determined modulo a rigid motion by their isometry invariants $\alpha \in [0, \pi]$ and d, a_1, b_1, a_2, $b_2 \in \mathbb{R}$ from Definition 2. For $l_i = b_i - a_i$, $i = 1, 2$, each line segment L_i is*

$$L_i(t) = \left((a_i + l_i t) \cos \frac{\alpha}{2},\ (-1)^i (a_i + l_i t) \sin \frac{\alpha}{2},\ (-1)^i \frac{d}{2} \right), \quad t \in [0, 1]. \tag{2}$$

If $t \in \mathbb{R}$ in Lemma 1, the corresponding point $L_i(t)$ moves along the line $\bar{L}_i$.

Lemma 2 (Formulae for Invariants) *Let $L_1, L_2 \subset \mathbb{R}^3$ be any skewed oriented line segments given by their initial and final endpoints $A_i, B_i \in \mathbb{R}^3$ so that $\mathbf{L}_i = \overrightarrow{A_i B_i}$, $i = 1, 2$. Then the isometry invariants of L_1, L_2 in Lemma 1 are computed as follows: the lengths*

$$l_i = \left| \overrightarrow{A_i B_i} \right|,$$

the signed distance

$$d = \frac{[\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{A_1 A_2}]}{|\mathbf{L_1} \times \mathbf{L_2}|},$$

the angle

$$\alpha = \arccos \frac{\mathbf{L_1} \cdot \mathbf{L_2}}{l_1 l_2},$$

$$a_1 = \left(\frac{\mathbf{L_2}}{l_2} \cos\alpha - \frac{\mathbf{L_1}}{l_1} \right) \cdot \frac{\overrightarrow{A_1 A_2}}{\sin^2 \alpha}, \qquad a_2 = \left(\frac{\mathbf{L_2}}{l_2} - \frac{\mathbf{L_1}}{l_1} \cos\alpha \right) \cdot \frac{\overrightarrow{A_1 A_2}}{\sin^2 \alpha},$$
$$b_i = a_i + l_i, \quad i = 1, 2.$$

Lemma 3 guarantees that the linking number behaves symmetrically in d, meaning that we may confine any particular analysis to cases where $d > 0$ or $d < 0$.

Lemma 3 (Symmetry) *Let $L_1, L_2 \subset \mathbb{R}^3$ be parameterised as in Lemma 1. Under the central symmetry* $\mathrm{CS}\colon (x, y, z) \mapsto (-x, -y, -z)$ *with respect to the origin $(0, 0, 0) \in \mathbb{R}^3$, the line segments keep their invariants $\alpha, a_1, b_1, a_2, b_2$. The signed distance d and the linking number change their signs:* $\mathrm{lk}(\mathrm{CS}(L_1), \mathrm{CS}(L_2)) = -\mathrm{lk}(L_1, L_2)$.

5 Invariant-Based Formula for the Linking Number of Segments

This section proves main Theorem 2, which expresses the linking number of any line segments in terms of their 6 isometry invariants from Definition 2. In 2000 Klenin and Langowski claimed a similar formula [8], but gave no proof, which requires substantial lemmas below. One of their 6 invariants differs from the signed distance d.

Theorem 2 (Invariant-Based Formula) *For any line segments $L_1, L_2 \subset \mathbb{R}^3$ with invariants $\alpha \in (0, \pi)$, $a_1, b_1, a_2, b_2, d \in \mathbb{R}$ from Definition 2, we have*

$$\mathrm{lk}(L_1, L_2) = \frac{\mathrm{AT}(a_1, b_2; d, \alpha) + \mathrm{AT}(b_1, a_2; d, \alpha) - \mathrm{AT}(a_1, a_2; d, \alpha) - \mathrm{AT}(b_1, b_2; d, \alpha)}{4\pi}, \tag{3}$$

where

$$\mathrm{AT}(a, b; d, \alpha) = \arctan\left(\frac{ab \sin\alpha + d^2 \cot\alpha}{d\sqrt{a^2 + b^2 - 2ab\cos\alpha + d^2}} \right).$$

For $\alpha = 0$ or $\alpha = \pi$, we set

$$\mathrm{AT}(a, b; d, \alpha) = \mathrm{sign}(d)\frac{\pi}{2}.$$

We also set $\mathrm{lk}(L_1, L_2) = 0$ *when* $d = 0$.

$a^2 + b^2 - 2ab\cos\alpha$ gives the squared third side of the triangle with the first two sides a, b and the angle α between them, hence is always non-negative. Also $a^2 + b^2 - 2ab\cos\alpha = 0$ only when the triangle degenerates for $a = \pm b$ and $\cos\alpha = \pm 1$. For $\alpha = 0$ or $\alpha = \pi$ when L_1, L_2 are parallel, $\mathrm{lk}(L_1, L_2) = 0$ is guaranteed by $\mathrm{AT}(a, b; d, \alpha) = \mathrm{sign}(d)\pi/2 = 0$ when $d = 0$ holds in addition to $\alpha = 0$ or $\alpha = \pi$.

The symmetry of the AT function in a, b, i.e. $\mathrm{AT}(a, b; d, \alpha) = \mathrm{AT}(b, a; d, \alpha)$ implies that $\mathrm{lk}(L_1, L_2) = \mathrm{lk}(L_2, L_1)$ by Theorem 2. Since the AT function is odd in d, i.e. $\mathrm{AT}(a, b; -d, \alpha) = -\mathrm{AT}(b, a; d, \alpha)$, Lemma 3 is also respected.

Figure 2 shows how the function $\mathrm{AT}(a, b; d, \alpha)$ from Theorem 2 depends on 2 of 4 parameters when others are fixed. For example, if $\alpha = \pi/2$, then

$$\mathrm{AT}(a, b; d, \pi/2) = \arctan\left(\frac{ab}{d\sqrt{a^2 + b^2 + d^2}}\right).$$

If also $a = b$, then the surface

$$\mathrm{AT}(a, a; d, \pi/2) = \arctan\left(\frac{a^2}{d\sqrt{2a^2 + d^2}}\right)$$

in the first picture of Fig. 2 has the horizontal ridge

$$\mathrm{AT}(0, 0; d, \pi/2) = 0 \quad \text{and} \quad \lim_{d\to 0} \mathrm{AT}(a, a; d, \pi/2) = \mathrm{sign}(d)\frac{\pi}{2} \quad \text{for} \quad a \neq 0.$$

If d, α are free, but $a = 0$, then

$$\mathrm{AT}(0, 0; d, \alpha) = \arctan\left(\frac{d^2\cot\alpha}{d\sqrt{d^2}}\right) = \mathrm{sign}(d)\arctan(\cot\alpha) = \mathrm{sign}(d)\left(\frac{\pi}{2} - \alpha\right).$$

Similarly,

$$\lim_{d\to\infty} \mathrm{AT}(0, 0; d, \alpha) = \mathrm{sign}(d)\left(\frac{\pi}{2} - \alpha\right),$$

see the lines $\mathrm{AT} = \pi/2 - \alpha$ on the boundaries of the AT surfaces in the middle pictures of Fig. 2.

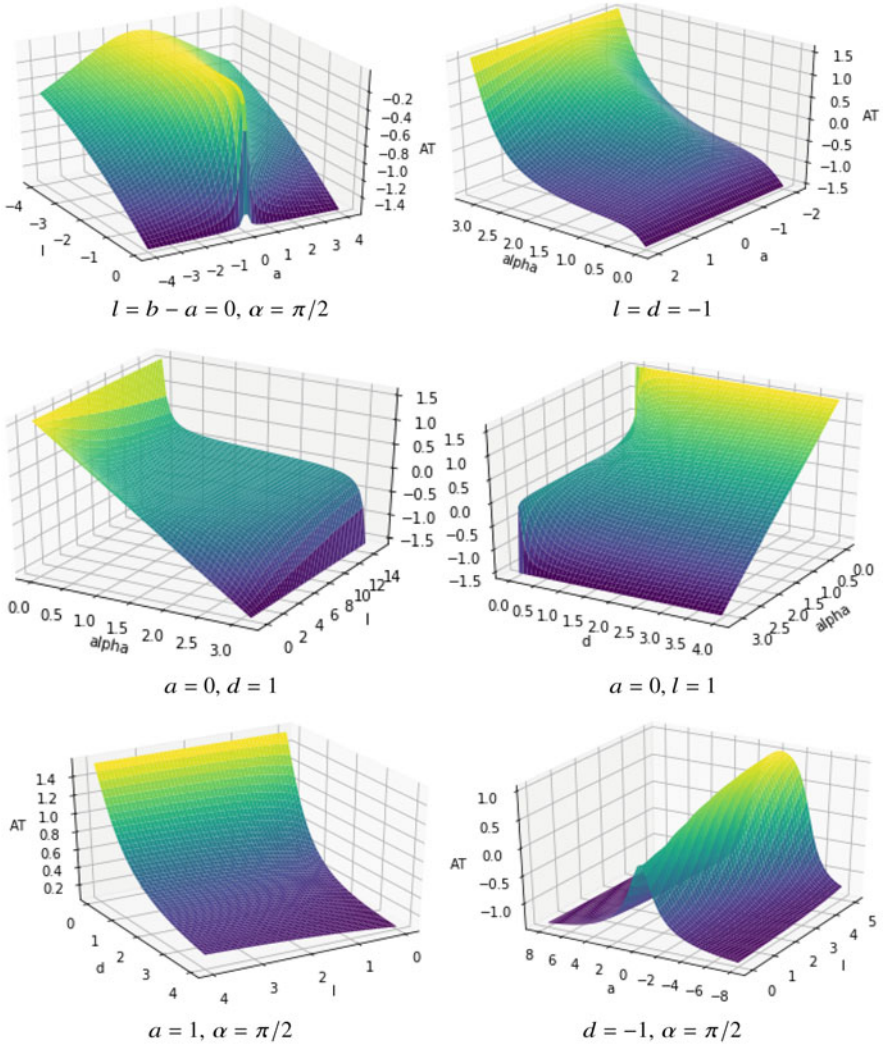


Fig. 2 Graph of $\mathrm{AT}(a, b; d, \alpha) = \arctan\Big((ab\sin\alpha + d^2\cot\alpha)/(d\sqrt{a^2 + b^2 - 2ab\cos\alpha + d^2})\Big)$, where 2 of 4 parameters are fixed

Lemma 4 ($\mathrm{lk}(L_1, L_2)$ **Is an Integral in** p, q) *In the notations of Definition 2 we have*

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi}\int\limits_{a_1/d}^{b_1/d}\int\limits_{a_2/d}^{b_2/d} \frac{\sin\alpha\,\mathrm{d}p\,\mathrm{d}q}{\left(1 + p^2 + q^2 - 2pq\cos\alpha\right)^{3/2}} \quad \textit{for} \quad d > 0.$$

Lemma 5 (The Linking Number as a Single Integral) *In the notations of Definition 2 we have*

$$\mathrm{lk}(L_1, L_2) = \frac{I(a_2/d) - I(b_2/d)}{4\pi},$$

where the function $I(r)$ is defined as the single integral

$$I(r) = \int_{a_1/d}^{b_1/d} \frac{\sin\alpha(r - p\cos\alpha)\,\mathrm{d}p}{\left(1 + p^2\sin^2\alpha\right)\sqrt{1 + p^2 + r^2 - 2pr\cos\alpha}} \quad \textit{for} \quad d > 0.$$

Lemma 6 ($I(r)$ via Arctan) *The integral $I(r)$ in Lemma 5 can be found as*

$$\int \frac{\sin\alpha(r - p\cos\alpha)\,\mathrm{d}p}{\left(1 + p^2\sin^2\alpha\right)\sqrt{1 + p^2 + r^2 - 2pr\cos\alpha}} = \arctan\frac{pr\sin\alpha + \cot\alpha}{\sqrt{1 + p^2 + r^2 - 2pr\cos\alpha}} + C.$$

***Proof** (**Theorem 2**)* Consider the right hand side of the equation in Lemma 6 as the 3-variable function

$$F(p, r; \alpha) = \arctan\left(\frac{pr\sin^2\alpha + \cos\alpha}{\sqrt{1 + p^2 + r^2 - 2pr\cos\alpha}}\right).$$

The function in Lemma 5 is $I(r) = F(b_1/d, r; \alpha) - F(a_1/d, r; \alpha)$. By Lemma 5

$$\begin{aligned}\mathrm{lk}(L_1, L_2) =& \frac{(F(b_1/d, a_2/d; \alpha) - F(a_1/d, a_2/d; \alpha))}{4\pi} \\ &- \frac{(F(b_1/d, b_2/d; \alpha) - F(a_1/d, b_2/d; \alpha))}{4\pi}.\end{aligned}$$

Rewrite a typical function from the numerator above as follows:

$$\begin{aligned}F(a/d, b/d; \alpha) &= \arctan\frac{\left(ab/d^2\right)\sin^2\alpha + \cos\alpha}{\sqrt{1 + (a/d)^2 + (b/d)^2 - 2\left(ab/d^2\right)\cos\alpha}} \\ &= \arctan\frac{ab\sin\alpha + d^2\cot\alpha}{d\sqrt{a^2 + b^2 - 2ab\cos\alpha + d^2}}.\end{aligned}$$

If we denote the last expression as $\mathrm{AT}(a, b; d, \alpha)$, required formula (3) follows.

In Lemma 4, Lemma 5 and above we have used that the signed distance d is positive. By Lemma 3 the signed distance d and $\mathrm{lk}(L_1, L_2)$ simultaneously change

their signs under a central symmetry, while all other invariants remain the same. Since $\mathrm{AT}(a, b; -d, \alpha) = -\mathrm{AT}(a, b; d, \alpha)$ due to the arcsin function being odd, formula (3) holds for $d < 0$. The formula remains valid even for $d = 0$, when L_1, L_2 are in the same plane. The expected value $\mathrm{lk}(L_1, L_2) = 0$ needs an explicit setting, see the discussion of the linking number discontinuity around $d = 0$ in Corollary 4. □

6 The Asymptotic Behaviour of the Linking Number of Segments

This section discusses how the linking number $\mathrm{lk}(L_1, L_2)$ in Theorem 2 behaves with respect to the six parameters of line segments L_1, L_2. Figure 3 shows how the linking number between two equal line segments varies with different pairs of parameters.

Corollary 2 (Bounds of the Linking Number) *For any line segments $L_1, L_2 \subset \mathbb{R}^3$, the linking number $\mathrm{lk}(L_1, L_2)$ is between $\pm 1/2$.*

Corollary 3 (Sign of the Linking Number) *In the notation of Definition 2,*

$$\lim_{\alpha \to 0} \mathrm{lk}(L_1, L_2) = 0 = \lim_{\alpha \to \pi} \mathrm{lk}(L_1, L_2).$$

Any non-parallel L_1, L_2 have

$$\mathrm{sign}(\mathrm{lk}(L_1, L_2)) = -\,\mathrm{sign}(d).$$

So $\mathrm{lk}(L_1, L_2) = 0$ if and only if $d = 0$ or $\alpha = 0$ or $\alpha = \pi$.

Corollary 4 (lk **for** $d \to 0$) *If $d \to 0$ and L_1, L_2 remain disjoint, formula (3) is continuous and*

$$\lim_{d \to 0} \mathrm{lk}(L_1, L_2) = 0.$$

If $d \to 0$ and L_1, L_2 intersect each other in the limit case $d = 0$, then

$$\lim_{d \to 0} \mathrm{lk}(L_1, L_2) = -\frac{\mathrm{sign}(d)}{2},$$

where $d \to 0$ keeps its sign.

Corollary 5 (lk **for** $d \to \pm\infty$) *If the distance $d \to \pm\infty$, then $\mathrm{lk}(L_1, L_2) \to 0$.*

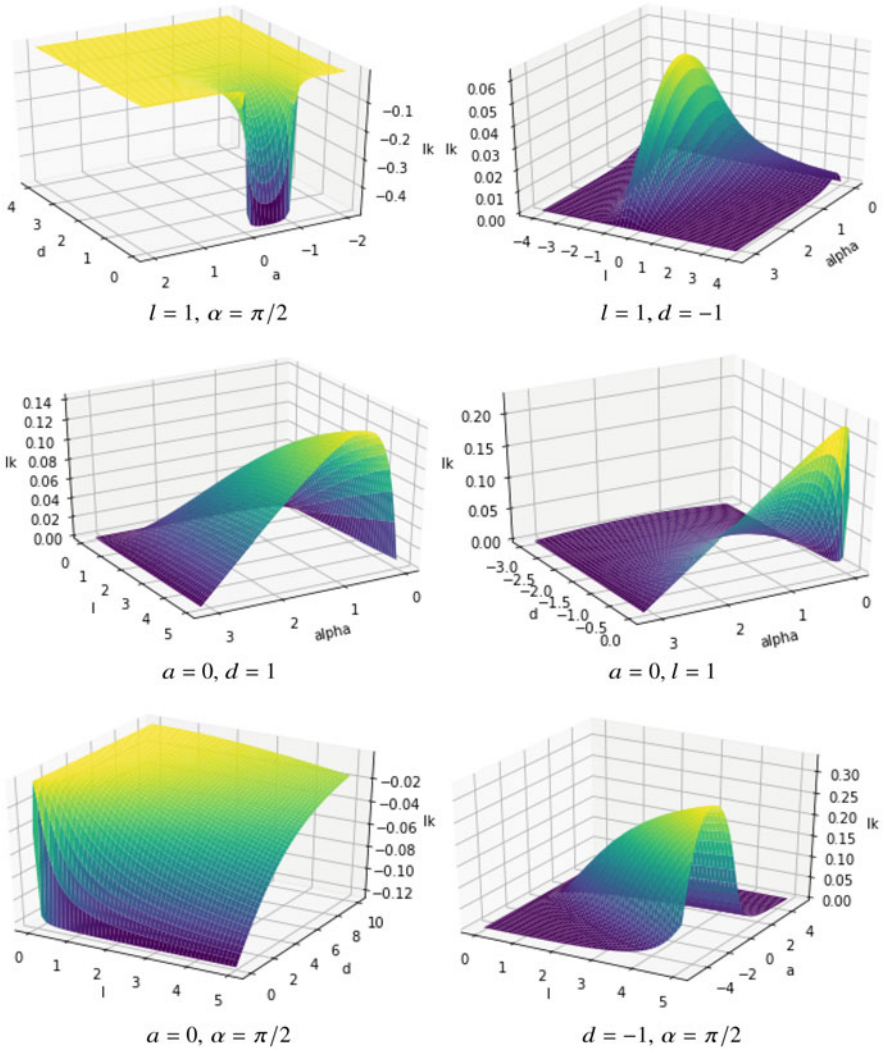


Fig. 3 The linking number $\mathrm{lk}(a, a+l; a, a+l; d, \alpha)$ from formula (3), where 2 of 4 parameters are fixed

Corollary 6 (lk **for** $a_i, b_i \to \infty$) *If the invariants* d, α *of line segments* $L_1, L_2 \subset \mathbb{R}^3$ *remain fixed, but* $a_i \to +\infty$ *or* $b_i \to -\infty$ *for each* $i = 1, 2$, *then* $\mathrm{lk}(L_1, L_2) \to 0$.

Corollary 7 (lk **for** $a_i \to b_i$) *If one of segments* $L_1, L_2 \subset \mathbb{R}^3$ *becomes infinitely short so that its final endpoint tends to the fixed initial endpoint (or vice versa), while all other invariants of* L_1, L_2 *from Definition 2 remain fixed, then* $\mathrm{lk}(L_1, L_2) \to 0$.

7 Example Computations of the Linking Number and a Discussion

If curves $\gamma_1, \gamma_2 \subset \mathbb{R}^3$ consist of straight line segments, then $\mathrm{lk}(\gamma_1, \gamma_2)$ can be computed as the sum of $\mathrm{lk}(L_1, L_2)$ over all line segments $L_1 \subset \gamma_1$ and $L_2 \subset \gamma_2$. Figure 4 shows polygonal links whose linking numbers were computed by our Python code at https://github.com/MattB-242/Closed_Lk_Form.

The asymptotic linking number introduced by Arnold converges for infinitely long curves [16], while our initial motivation was a computation of geometric and topology invariants to classify periodic structures such as textiles [4] and crystals [5].

Theorem 2 allows us to compute the *periodic* linking number between a segment J and a growing finite lattice L_n whose unit cell consists of n copies of two oppositely oriented segments orthogonal to J. This periodic linking number is computed for increasing n in a lattice extending periodically in one, two and three directions, see Fig. 5. As n increases, the lk function asymptotically approaches an approximate value of 0.30 for 1- and 3-periodic lattice and 0.29 for the 2-periodic lattice.

The invariant-based formula has allowed us to prove new asymptotic results of the linking number in Corollaries 2–7 of Sect. 6. Since the periodic linking number is a real-valued invariant modulo isometries, it can be used to continuously quantify similarities between periodic crystalline networks [5]. One next possible step is to use formula (3) to prove asymptotic convergence of the periodic linking number for arbitrary lattices, so that we can show that the limit of the infinite sum is a general invariant that can be used to develop descriptors of crystal structures.

The Milnor invariants generalise the linking number to invariants of links with more than two components. An integral for the three component Milnor invariant [6] may be possible to compute in a closed form similarly to Theorem 2. The interesting open problem is to extend the isometry-based approach to finer invariants of knots.

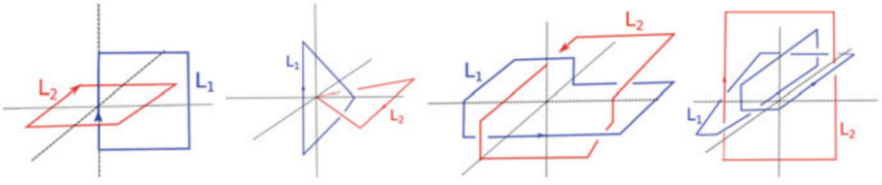


Fig. 4 **1st:** The Hopf link as two square cycles has $\mathrm{lk} = -1$ and vertices with coordinates $L_1 = (-2, 0, 2), (2, 0, -2), (2, 0, 2), (-2, 0, 2)$ and $L_2 = (-1, -2, 0), (-1, 2, 0), (1, 2, 0), (1, -2, 0)$. **2nd:** The Hopf link as two triangular cycles has $\mathrm{lk} = +1$, $L_1 = (-1, 0, -1)$, $(-1, 0, 1)$, $(1, 0, 0)$ and $L_2 = (0, 0, 0)$, $(2, 1, 0)$, $(2, -1, 0)$. **3rd:** Solomon's link has $\mathrm{lk} = +2$, $L_1 = (-1, 1, 1)$, $(-1, -1, 1)$, $(3, -1, 1)$, $(3, 1, -1)$, $(1, 1, -1)$, $(1, 1, 1)$ and $L_2 = (-1, -2, 0)$, $(-1, 2, 0)$, $(1, 2, 0)$, $(1, -2, 0)$. **4th:** Whitehead's link has $\mathrm{lk} = 0$, $L_1 = (-3, -2, -1)$, $(0, -2, -1)$, $(0, 2, 1)$, $(0, 0, 1)$, $(0, 0, 0)$, $(3, 0, 0)$, $(3, 1, 0)$, $(-3, 1, 0)$, $(-3, -2, -1)$ and $L_2 = (-1, 0, -3)$, $(-1, 0, 3)$, $(1, 0, 3)$, $(-1, 0, 3)$

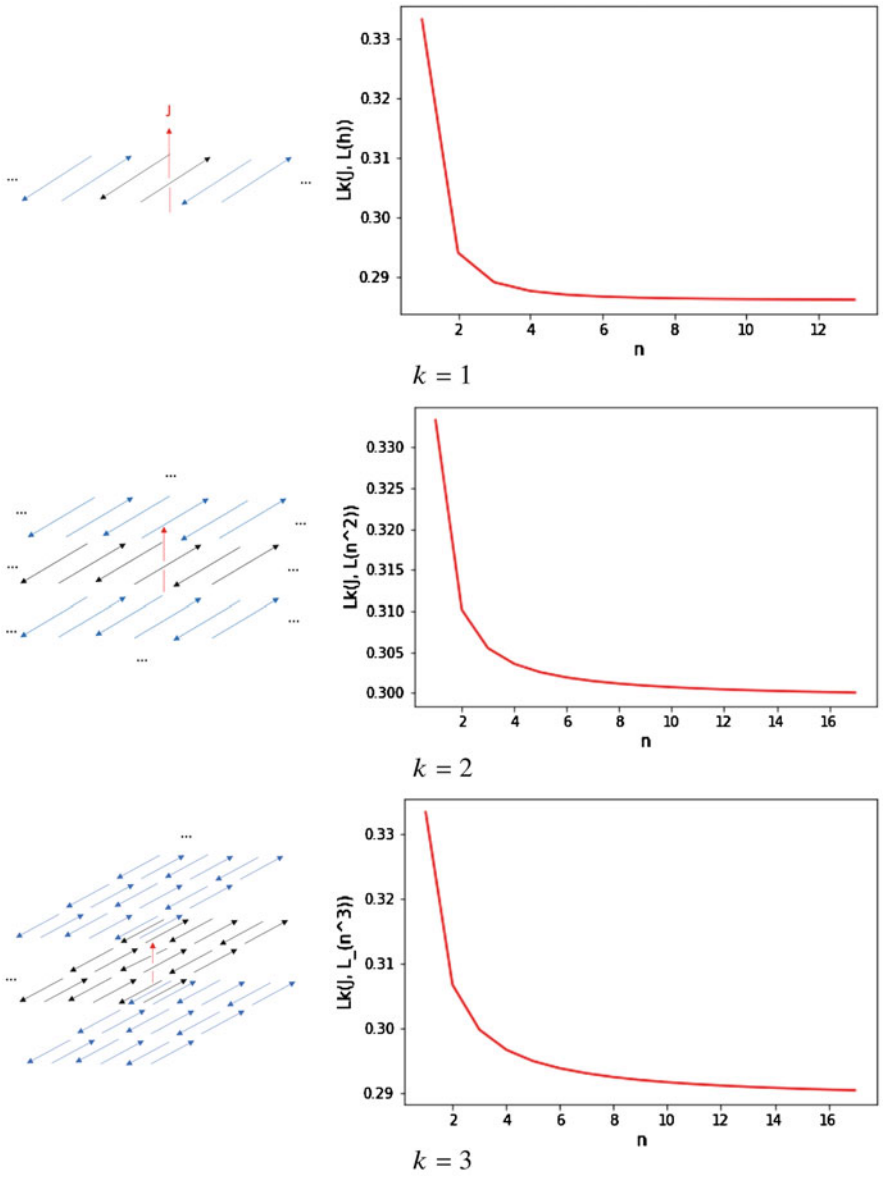


Fig. 5 Left: the line segment $J = (0, 0, -1) + t(0, 0, 2)$ in red and the periodic lattice $L(n^k)$ derived from n copies of the 'unit cell' $L = \{(-1, -1, 0) + t(0, 2, 0), (-1, 1, 0) + s(0, -2, 0)\}$, $t, s \in [0, 1]$, translated in k linearly independent directions for increasing $n \in \mathbb{Z}$. Right: the periodic linking number $\mathrm{lk}(J, L(n^k))$ is converging fast for $n \to +\infty$

The Gauss integral in (1) was extended to the infinite Kontsevich integral containing all finite-type or Vassiliev's invariants of knots [9]. The coefficients of this infinite series were explicitly described [10] as solutions of exponential equations with non-commutative variables x, y in a compressed form modulo commutators of commutators in x, y. The underlying metabelian formula for $\ln(e^x e^y)$ has found an easier proof [11] in the form of a generating series in the variables x, y.

In conclusion, we have proved the analytic formula for the linking number based on 6 isometry invariants that uniquely determine a relative position of two line segments in $\mathbb{R}^3$. Though a similar formula was claimed in [8], no proof was given. Hence this paper fills an important gap in the literature by completing the proof via three non-trivial lemmas in Sect. 5, see detailed computations in the arXiv version of this paper.

Appendix 1: Proofs of Lemmas about Isometry Invariants

Appendices 1, 2, and 3 provide extra details that are not included into the main paper.

Proof (Lemma 1) Any line segments $L_1, L_2 \subset \mathbb{R}^3$ that are not in the same plane are contained in distinct parallel planes. For $i = 1, 2$, the plane Π_i is spanned by L_i and the line parallel to L_{3-i} and passing through an endpoint of L_i. Let L_i' be the orthogonal projection of the line segment L_i to the plane Π_{3-i}. The non-parallel lines through the segments L_i and L_{3-i}' in the plane Π_i intersect at a point, say O_i. Then the line segment O_1O_2 is orthogonal to both planes Π_i, hence to both L_i for $i = 1, 2$.

By Theorem 1, to compute $\mathrm{lk}(L_1, L_2)$, one can apply a rigid motion to move the mid-point of the line segment O_1O_2 to the origin $O = (0, 0, 0) \in \mathbb{R}^2$ and make O_1O_2 vertical, i.e., lying within the z-axis. The signed distance d can be defined as the difference between the coordinates of $O_2 = \Pi_2 \cap (z\text{-axis})$ and $O_1 = \Pi_1 \cap (z\text{-axis})$ along the z-axis. Then L_i lies in the horizontal plane $\Pi_i = \{z = (-1)^i d/2\}, i = 1, 2$.

An extra rotation around the z-axis guarantees that the x-axis in the horizontal plane $\Pi = \{z = 0\}$ is the bisector of the angle $\alpha \in [0, \pi]$ from $\mathrm{pr}_{xy}(\bar{L}_1)$ to $\mathrm{pr}_{xy}(\bar{L}_2)$, where $\mathrm{pr}_{xy}\colon \mathbb{R}^3 \to \Pi$ is the orthogonal projection. Then the infinite lines $\bar{L}_i$ through L_i have the parametric form $(x, y, z) = \big(t\cos(\alpha/2), (-1)^i t\sin(\alpha/2), (-1)^i d/2\big)$ with $s \in \mathbb{R}$.

The point O_i can be considered as the origin of the oriented infinite line $\bar{L}_i$. Let the line segment L_i have a length $l_i > 0$ and its initial point have the coordinate $a_i \in \mathbb{R}$ in the oriented line $\bar{L}_i$. Then the final endpoint of L_i has the coordinate $b_i = a_i + l_i$. To cover only the segment L_i, the parameter t should be replaced by $a_i + l_i t, t \in [0, 1]$. □

Proof (Lemma 2) The vectors along the segments are $\mathbf{L}_i = \mathbf{v}_i - \mathbf{u}_i$, hence the lengths are $l_i = |\mathbf{L}_i| = |\overrightarrow{A_i B_i}|$, $i = 1, 2$. The angle $\alpha \in [0, \pi]$ between $\mathbf{L_1}, \mathbf{L_2}$ can be found from the scalar product $\mathbf{L_1} \cdot \mathbf{L_2} = |\mathbf{L_1}| \cdot |\mathbf{L_2}| \cos\alpha$ as $\alpha = \arccos((\mathbf{L_1} \cdot \mathbf{L_2})/(l_1 l_2))$, because the function $\arccos x \colon [-1, 1] \to [0, \pi]$ is bijective.

Since $\mathbf{L_1}$ and $\mathbf{L_2}$ are not proportional, the normalised vector product $\mathbf{e}_3 = (\mathbf{L_1} \times \mathbf{L_2})/(|\mathbf{L_1} \times \mathbf{L_2}|)$ is well-defined and orthogonal to both $\mathbf{L_1}, \mathbf{L_2}$. Then $\mathbf{e}_1 = \mathbf{L_1}/|\mathbf{L_1}|$, $\mathbf{e}_2 = \mathbf{L_2}/|\mathbf{L_2}|$ and $\mathbf{e}_3$ have lengths 1 and form a linear basis of $\mathbb{R}^3$, where the last vector is orthogonal to the first two.

Let O be any fixed point of $\mathbb{R}^3$, which can be assume to be the origin $(0, 0, 0)$ in the coordinates of Lemma 1, though its position relative to $\overrightarrow{A_i B_i}$ is not yet determined. First we express the points $O_i = (0, 0, (-1)^i d/2) \in \bar{L}_i$ from Fig. 1 in terms of given vectors $\overrightarrow{A_i B_i}$. If the initial endpoint A_i has a coordinate a_i in the line $\bar{L}_i$ through L_i, then $\overrightarrow{O_i A_i} = a_i \mathbf{e}_i$ and

$$\overrightarrow{O_1 O_2} = \overrightarrow{O O_2} - \overrightarrow{O O_1} = (\overrightarrow{O A_2} - \overrightarrow{O_2 A_2}) - (\overrightarrow{O A_1} - \overrightarrow{O_1 A_1}) = \overrightarrow{A_1 A_2} + a_1 \mathbf{e}_1 - a_2 \mathbf{e}_2.$$

By Definition 2, $\overrightarrow{O_1 O_2}$ is orthogonal to the line $\bar{L}_i$ going through the vector $\mathbf{e}_i = \mathbf{L}_i/|\mathbf{L}_i|$ for $i = 1, 2$. Then the product $[\mathbf{e}_1, \mathbf{e}_2, \overrightarrow{O_1 O_2}] = (\mathbf{e}_1 \times \mathbf{e}_2) \cdot \overrightarrow{O_1 O_2}$ equals $|\mathbf{e}_1 \times \mathbf{e}_2| d$, where $\overrightarrow{O_1 O_2}$ is in the z-axis, the signed distance d is the z-coordinate of O_2 minus the z-coordinates O_1.

The product $[\mathbf{e}_1, \mathbf{e}_2, \overrightarrow{O_1 O_2}] = (\mathbf{e}_1 \times \mathbf{e}_2) \cdot (\overrightarrow{A_1 A_2} + a_1 \mathbf{e}_1 - a_2 \mathbf{e}_2) = (\mathbf{e}_1 \times \mathbf{e}_2) \cdot \overrightarrow{A_1 A_2}$ doesn't depend on a_1, a_2, because $\mathbf{e}_1 \times \mathbf{e}_2$ is orthogonal to both $\mathbf{e}_1, \mathbf{e}_2$. Hence the signed distance is

$$d = \frac{[\mathbf{e}_1, \mathbf{e}_2, \overrightarrow{A_1 A_2}]}{|\mathbf{e}_1 \times \mathbf{e}_2|} = \frac{[\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{A_1 A_2}]}{|\mathbf{L_1} \times \mathbf{L_2}|},$$

which can be positive or negative, see Fig. 1.

It remains to find the coordinate a_i of the initial endpoint of L_i relative to the origin $O_i \in \bar{L}_i$, $i = 1, 2$. The vector $\overrightarrow{O_1 O_2} = \overrightarrow{A_1 A_2} + a_1 \mathbf{e}_1 - a_2 \mathbf{e}_2$ is orthogonal to both $\mathbf{e}_i$ if and only if the scalar products vanish: $\overrightarrow{O_1 O_2} \cdot \mathbf{e}_i = 0$. Due to $|\mathbf{e}_1| = 1 = |\mathbf{e}_2|$ and $\mathbf{e}_1 \cdot \mathbf{e}_2 = \cos\alpha$, we get

$$\begin{cases} \mathbf{e}_1 \cdot \overrightarrow{A_1 A_2} + a_1 - a_2 (\mathbf{e}_1 \cdot \mathbf{e}_2) = 0, \\ \mathbf{e}_2 \cdot \overrightarrow{A_1 A_2} + a_1 (\mathbf{e}_1 \cdot \mathbf{e}_2) - a_2 = 0, \end{cases} \leftrightarrow \begin{pmatrix} 1 & -\cos\alpha \\ \cos\alpha & -1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = - \begin{pmatrix} \mathbf{e}_1 \cdot \overrightarrow{A_1 A_2} \\ \mathbf{e}_2 \cdot \overrightarrow{A_1 A_2} \end{pmatrix}.$$

The determinant of the 2×2 matrix is $\cos^2\alpha - 1 = -\sin^2\alpha \neq 0$, because L_1, L_2 are not parallel. Then

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \frac{1}{\sin^2\alpha} \begin{pmatrix} -1 & \cos\alpha \\ -\cos\alpha & 1 \end{pmatrix} \begin{pmatrix} \mathbf{e}_1 \cdot \overrightarrow{A_1 A_2} \\ \mathbf{e}_2 \cdot \overrightarrow{A_1 A_2} \end{pmatrix}$$

and

$$a_1 = \frac{-\mathbf{e}_1 \cdot \overrightarrow{A_1A_2} + \cos\alpha(\mathbf{e_2} \cdot \overrightarrow{A_1A_2})}{\sin^2\alpha} = \frac{(\mathbf{e}_2\cos\alpha - \mathbf{e}_1)\cdot \overrightarrow{A_1A_2}}{\sin^2\alpha}$$
$$= \left(\frac{\mathbf{L_2}}{l_2}\cos\alpha - \frac{\mathbf{L_1}}{l_1}\right)\cdot \frac{\overrightarrow{A_1A_2}}{\sin^2\alpha},$$
$$a_2 = \frac{\cos\alpha(\mathbf{e}_1 \cdot \overrightarrow{A_1A_2}) - \mathbf{e_1} \cdot \overrightarrow{A_1A_2}}{\sin^2\alpha} = \frac{(\mathbf{e}_2 - \mathbf{e}_1\cos\alpha)\cdot \overrightarrow{A_1A_2}}{\sin^2\alpha}$$
$$= \left(\frac{\mathbf{L_2}}{l_2} - \frac{\mathbf{L_1}}{l_1}\cos\alpha\right)\cdot \frac{\overrightarrow{A_1A_2}}{\sin^2\alpha}.$$

The coordinates of the final endpoints are obtained as $b_i = a_i + l_i$, $i = 1, 2$. □

Proof (Lemma 3) Under the central symmetry CS, in the notation of Lemma 2 the vectors $\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{A_1A_2}$ change their signs. Then the formulae for $\alpha, a_1, b_1, a_2, b_2$ gives the same expression, but the triple product $[\mathbf{L_1}, \mathbf{L_2}, \overrightarrow{A_1A_2}]$ and d change their signs.

Since the central symmetry CS is an orthogonal map M with $\det M = -1$, the new linking number changes its sign as follows:

$$\mathrm{lk}(\mathrm{CS}(L_1), \mathrm{CS}(L_2)) = \mathrm{lk}(\mathrm{CS}(L_2), \quad \mathrm{CS}(L_1)) = -\mathrm{lk}(L_1, L_2),$$

where we also make use of the invariance of lk under exchange of the segments from Theorem 1 (f). □

Appendix 2: Proofs of Lemmas for the lk Formula in Theorem 2

Proof (Lemma 4) The following computations assume that $a_1, a_2, l_1, l_2, \alpha$ are given and $t, s \in [0, 1]$.

$$L_1(t) = ((a_1 + l_1 t)\cos\frac{\alpha}{2}, -(a_1 + l_1 t)\sin\alpha, -\frac{d}{2}),$$
$$L_2(s) = ((a_2 + l_2 s)\cos\frac{\alpha}{2}, (a_2 + l_2 s)\sin\alpha, \frac{d}{2}),$$
$$\dot{L}_1(t) = (l_1\cos\frac{\alpha}{2}, -l_1\sin\frac{\alpha}{2}, 0),$$
$$\dot{L}_2(s) = (l_2\cos\frac{\alpha}{2}, l_2\sin\frac{\alpha}{2}, 0),$$

$$\dot{L}_1(t) \times \dot{L}_2(s) = (0, 0, 2l_1l_2 \sin\frac{\alpha}{2}\cos\frac{\alpha}{2}) = (0, 0, l_1l_2 \sin\alpha),$$

$$L_1(t) - L_2(s) = ((a_1 - a_2 + l_1t - l_2s)\cos\alpha, -(a_1 + a_2 + l_1t + l_2s)\sin\alpha, -d),$$

$$\big(\dot{L}_1(t), \dot{L}_2(s), L_1(t) - L_2(s)\big) = -dl_1l_2 \sin\alpha,$$

$$\mathrm{lk}(L_1, L_2) = \frac{1}{4\pi}\int_0^1\int_0^1 \frac{(\dot{L}_1(t), \dot{L}_2(s), L_1(t) - L_2(s))}{|L_1(t) - L_2(s)|^3}\,\mathrm{d}t\,\mathrm{d}s$$

$$= \frac{1}{4\pi}\int_0^1\int_0^1 \frac{-dl_1l_2 \sin\alpha\,\mathrm{d}t\,\mathrm{d}s}{\big(d^2 + (a_1 - a_2 + l_1t - l_2s)^2\cos^2\frac{\alpha}{2} + (a_1 + a_2 + l_1t + l_2s)^2\sin^2\frac{\alpha}{2}\big)^{3/2}}$$

$$= -\frac{dl_1l_2 \sin\alpha}{4\pi}\int_0^1\int_0^1 \frac{\mathrm{d}t\,\mathrm{d}s}{\big(d^2 + (a_1 - a_2 + l_1t - l_2s)^2\cos^2\frac{\alpha}{2} + (a_1 + a_2 + l_1t + l_2s)^2\sin^2\frac{\alpha}{2}\big)^{3/2}}.$$

To simplify the last integral, introduce the variables $p = (a_1 + l_1t)/d$ and $q = (a_2 + l_2s)/d$. In the new variables p, q the expression under the power $3/2$ in the denominator becomes

$$\begin{aligned} d^2 + (pd - qd)^2\cos^2\frac{\alpha}{2} + (pd + qd)^2\sin^2\frac{\alpha}{2} &= d^2\Big(1 + (p^2 - 2pq + q^2)\cos^2\frac{\alpha}{2} + (p^2 - 2pq + q^2)\cos^2\frac{\alpha}{2}\Big) \\ &= d^2\Big(1 + p^2\Big(\cos^2\frac{\alpha}{2} + \sin^2\frac{\alpha}{2}\Big) + q^2 - 2pq\Big(\cos^2\frac{\alpha}{2} - \sin^2\frac{\alpha}{2}\Big)\Big) \\ &= d^2(1 + p^2 + q^2 - 2pq\cos\alpha). \end{aligned}$$

The old variables are expressed as $t = (pd - a_1)/l_1$, $ts = (pd - a_2)/l_2$ and have the differentials $\mathrm{d}t = d/l_1\,\mathrm{d}p$, $\mathrm{d}s = d/l_2\,\mathrm{d}q$. Since $t, s \in [0, 1]$, the new variables p, q have the ranges $[a_1/d, b_1/d]$ and $[a_2/d, b_2/d]$, respectively. Then the linking number has the required expression in the lemma:

$$\mathrm{lk}(L_1, L_2) = -\frac{dl_1l_2 \sin\alpha}{4\pi}\int_{a_1/d}^{b_1/d}\int_{a_2/d}^{b_2/d} \frac{d^2}{l_1l_2}\frac{\mathrm{d}d\,\mathrm{d}q}{d^3(1 + p^2 + q^2 - 2pq\cos\alpha)^{3/2}}$$

$$= -\frac{1}{4\pi}\int_{a_1/d}^{b_1/d}\int_{a_2/d}^{b_2/d} \frac{\sin\alpha\,\mathrm{d}p\,\mathrm{d}q}{(1 + p^2 + q^2 - 2pq\cos\alpha)^{3/2}}.$$

Due to Lemma 3, the above computations assume that the signed distance $d > 0$.

□

***Proof** (**Lemma 5**)* Complete the square in the expression under power 3/2 in Lemma 4:

$$1 + p^2 + q^2 - 2pq\cos\alpha = 1 + p^2\sin^2\alpha + (q - p\cos\alpha)^2.$$

The substitution $(q - p\cos\alpha) = (1 + p^2\sin^2\alpha)\tan^2\psi$ for the new variable ψ simplifies the sum of squares to $1 + \tan^2\psi = 1/\cos^2\psi$. Since q varies within $[a_2/d, b_2/d]$, for any fixed $p \in [a_1/d, b_1 d]$, the range $[\psi_0, \psi_1]$ of ψ satisfies

$$\tan\psi_0 = \frac{\frac{a_2}{d} - p\cos\alpha}{\sqrt{1 + p^2\sin^2\alpha}} \quad \text{and} \quad \tan\psi_1 = \frac{\frac{b_2}{d} - p\cos\alpha}{\sqrt{1 + p^2\sin^2\alpha}}.$$

Since we treat p, ψ as independent variables, the Jacobian of the substitution $(p, q) \mapsto (p, \psi)$ equals

$$\frac{\partial q}{\partial\psi} = \frac{\partial}{\partial\psi}\left(p\cos\alpha + \tan\psi\sqrt{1 + p^2\sin^2\alpha}\right) = \frac{\sqrt{1 + p^2\sin^2\alpha}}{\cos^2\psi}.$$

In the variables p, ψ the expression under the double integral of Lemma 4 becomes

$$\begin{aligned}\frac{\sin\alpha\,\mathrm{d}p\,\mathrm{d}q}{(1 + p^2 + q^2 - 2pq\cos\alpha)^{3/2}} &= \frac{\sin\alpha\,\mathrm{d}q}{((1 + p^2\sin^2\alpha) + (1 + p^2\sin^2\alpha)\tan^2\psi)^{3/2}}\frac{\partial q}{\partial\psi}\,\mathrm{d}\psi \\ &= \frac{\sin\alpha\,\mathrm{d}p}{(1 + p^2\sin^2\alpha)^{3/2}(1 + \tan^2\psi)^{3/2}}\frac{\mathrm{d}\psi\sqrt{1 + p^2\sin^2\alpha}}{\cos^2\psi} \\ &= \frac{\sin\alpha\,\mathrm{d}p\cos\psi\,\mathrm{d}\psi}{1 + p^2\sin^2\alpha}.\end{aligned}$$

$$\begin{aligned}\mathrm{lk}(L_1, L_2) &= -\frac{1}{4\pi}\int_{a_1/d}^{b_1/d}\frac{\sin\alpha\,\mathrm{d}p}{1 + p^2\sin^2\alpha}\int_{\psi_0}^{\psi_1}\cos\psi\,\mathrm{d}\psi \\ &= \frac{1}{4\pi}\int_{a_1/d}^{b_1/d}\frac{\sin\alpha\,\mathrm{d}p}{1 + p^2\sin^2\alpha}(\sin\psi_0 - \sin\psi_1).\end{aligned}$$

Express the sin functions for the bounds ψ_0, ψ_1 in terms of tan as $\sin\psi_0 = \tan\psi_0/\sqrt{1+\tan^2\psi_0}$. Using $\tan\psi_0 = (a_2/d - p\cos\alpha)/(\sqrt{1+p^2\sin^2\alpha})$ obtained above, we get

$$\begin{aligned}\sqrt{1+\tan^2\psi_0} &= \sqrt{\frac{(1+p^2\sin^2\alpha)+(\frac{a_2}{d}-p\cos\alpha)^2}{1+p^2\sin^2\alpha}}\\ &= \sqrt{\frac{1+p^2+(\frac{a_2}{d})^2-2\frac{a_2}{d}p\cos\alpha}{1+p^2\sin^2\alpha}}.\\ \sin\psi_0 &= \frac{\frac{a_2}{d}-p\cos\alpha}{\sqrt{1+p^2\sin^2\alpha}}\sqrt{\frac{1+p^2\sin^2\alpha}{1+p^2+(\frac{a_2}{d})^2-2\frac{a_2}{d}p\cos\alpha}}\\ &= \frac{\frac{a_2}{d}-p\cos\alpha}{\sqrt{1+p^2+(\frac{a_2}{d})^2-2\frac{a_2}{d}p\cos\alpha}}.\end{aligned}$$

Then $\sin\psi_1$ has the same expression with a_2 replaced by b_2. After substituting these expressions in the previous formula for the linking number, we get

$$\begin{aligned}\mathrm{lk}(L_1,L_2) =& \frac{1}{4\pi}\int_{a_1/d}^{b_1/d}\frac{\sin\alpha\,\mathrm{d}p}{1+p^2\sin^2\alpha}\left(\frac{\frac{a_2}{d}-p\cos\alpha}{\sqrt{1+p^2+(\frac{a_2}{d})^2-2\frac{a_2}{d}p\cos\alpha}}\right.\\ &\left.-\frac{\frac{b_2}{d}-p\cos\alpha}{\sqrt{1+p^2+(\frac{b_2}{d})^2-2\frac{b_2}{d}p\cos\alpha}}\right)\\ =& \frac{S(a_2/d)-S(b_2/d)}{4\pi},\end{aligned}$$

where

$$I(r) = \int_{a_1/d}^{b_1/d}\frac{\sin\alpha(r-p\cos\alpha)\,\mathrm{d}p}{(1+p^2\sin^2\alpha)\sqrt{1+p^2+r^2-2pr\cos\alpha}}.$$

□

Proof ***(Lemma 6)*** The easiest way is to differentiate the function $\arctan\omega$ for

$$\omega = \frac{pr\sin^2\alpha+\cos\alpha}{\sin\alpha\sqrt{1+p^2+r^2-2pr\cos\alpha}}$$

with respect to the variable p remembering that r, α are fixed parameters. For notational clarity, we use an auxiliary symbol for the expression under the square root: $R = 1 + p^2 + r^2 - 2pr\cos\alpha$. Then

$$\omega = \frac{pr\sin^2\alpha + \cos\alpha}{\sin\alpha\sqrt{R}}$$

and

$$\begin{aligned}\frac{\mathrm{d}\omega}{\mathrm{d}p} &= \frac{1}{R\sin\alpha}\left(r\sin^2\alpha\sqrt{R} - (rp\sin^2\alpha + \cos\alpha)\frac{2p - 2r\cos\alpha}{2\sqrt{R}}\right)\\ &= \frac{1}{R\sqrt{R}\sin\alpha}\Big(r\sin^2\alpha(1 + p^2 + r^2 - 2pr\cos\alpha) - (rp\sin^2\alpha + \cos\alpha)(p - r\cos\alpha)\Big)\\ &= \frac{rp^2\sin^2\alpha + r^3\sin^2\alpha - 2pr^2\cos\alpha\sin^2\alpha - rp^2\sin^2\alpha + pr^2\cos\alpha\sin^2\alpha - p\cos\alpha + r}{R\sqrt{R}\sin\alpha}\\ &= \frac{r^3\sin^2\alpha - pr^2\cos\alpha\sin^2\alpha - p\cos\alpha + r}{R\sqrt{R}\sin\alpha} = \frac{(r - p\cos\alpha)(1 + r^2\sin^2\alpha)}{R\sqrt{R}\sin\alpha}.\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}p}\arctan\omega &= \frac{1}{1+\omega^2}\times\frac{\mathrm{d}\omega}{\mathrm{d}p} = \frac{(\sin\alpha\sqrt{R})^2}{(\sin\alpha\sqrt{R})^2 + (pr\sin^2\alpha + \cos\alpha)^2}\times\frac{\mathrm{d}\omega}{\mathrm{d}p}\\ &= \frac{R\sin^2\alpha}{R\sin^2\alpha + (p^2r^2\sin^4\alpha + 2pr\sin^2\alpha\cos\alpha + \cos^2\alpha)}\times\frac{(r - p\cos\alpha)(1 + r^2\sin^2\alpha)}{R\sqrt{R}\sin\alpha}\\ &= \frac{\sin\alpha}{\sqrt{R}}\times\frac{(r - p\cos\alpha)(1 + r^2\sin^2\alpha)}{\sin^2\alpha(1 + p^2 + r^2 - 2pr\cos\alpha) + (p^2r^2\sin^4\alpha + 2pr\sin^2\alpha\cos\alpha + \cos^2\alpha)}\\ &= \frac{\sin\alpha(r - p\cos\alpha)(1 + r^2\sin^2\alpha)}{(1 + p^2\sin^2\alpha + r^2\sin^2\alpha + p^2r^2\sin^4\alpha)\sqrt{R}} = \frac{\sin\alpha(r - p\cos\alpha)(1 + r^2\sin^2\alpha)}{(1 + p^2\sin^2\alpha)(1 + r^2\sin^2\alpha)\sqrt{R}}\\ &= \frac{\sin\alpha(r - p\cos\alpha)}{(1 + p^2\sin^2\alpha)\sqrt{R}} = \frac{\sin\alpha(r - p\cos\alpha)}{(1 + p^2\sin^2\alpha)\sqrt{1 + p^2 + q^2 - 2pq\cos\alpha}}.\end{aligned}$$

Since we got the required expression under the integral $I(r)$, Lemma 6 is proved. □

Appendix 3: Proofs of Corollaries of Main Theorem 2

Proof of Corollary 1 By definition any simple orthogonal segments L_1, L_2 have $\alpha = \pi/2$ and initial endpoints $a_1 = 0 = a_2$, hence $b_1 = l_1$, $b_2 = l_2$. Substituting the values above into (3) gives

$$\mathrm{AT}(0, l_2; d, \frac{\pi}{2}) = 0, \quad \mathrm{AT}(l_1, 0; d, \frac{\pi}{2}) = 0, \quad \mathrm{AT}(0, 0; d, \frac{\pi}{2}) = 0.$$

Then

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi}\mathrm{AT}(l_1, l_2; d, \alpha) = -\frac{1}{4\pi}\arctan\left(\frac{l_1 l_2}{d\sqrt{l_1^2 + l_2^2 + d^2}}\right).$$

□

Proof ***(Corollary 2)*** By Theorem 2 $\mathrm{lk}(L_1, L_2)$ is a sum of 4 arctan functions divided by 4π. Since each arctan is strictly between $\pm\pi/2$, the linking number is between $\pm 1/2$. □

Proof ***(Corollary 3)*** If $\alpha = 0$ or $\alpha = \pi$, then $\cot\alpha$ is undefined, so Theorem 2 sets $\mathrm{AT}(a, b; d, \alpha) = \mathrm{sign}(d)\pi/2$. Then $\mathrm{lk}(L_1, L_2) = \mathrm{sign}(d)\pi/2(1 + 1 - 1 - 1) = 0$.

Theorem 2 also specifies that $\mathrm{lk}(L_1, L_2) = 0$ for $d = 0$. If $d \neq 0$ and $\alpha \to 0$ within $[0, \pi]$ while all other parameters remain fixed, then $d^2\cot\alpha \to +\infty$. Hence each of the 4 arctan functions in Theorem 2 approaches $\pi/2$, so $\mathrm{lk}(L_1, L_2) \to 0$. The same conclusion similarly follows in the case $\alpha \to \pi$ when $d^2\cot\alpha \to -\infty$.

If L_1, L_2 are not parallel, the angle α between them belongs to $(0, \pi)$. In $d > 0$, Lemma 4 says that

$$\mathrm{lk}(L_1, L_2) = -\frac{1}{4\pi}\int_{a_1/d}^{b_1/d}\int_{a_2/d}^{b_2/d}\frac{\sin\alpha\,\mathrm{d}p\,\mathrm{d}q}{(1 + p^2 + q^2 - 2pq\cos\alpha)^{3/2}}.$$

Since the function under the integral is strictly positive, $\mathrm{lk}(L_1, L_2) < 0$. By Lemma 3 both $\mathrm{lk}(L_1, L_2)$ simultaneously change their signs under a central symmetry. Hence, the formula $\mathrm{sign}(\mathrm{lk}(L_1, L_2)) = -\mathrm{sign}(d)$ holds for all d including $d = 0$ above. □

Proof ***(Corollary 4)*** Recall that $\lim\limits_{x\to\pm\infty}\arctan x = \pm\pi/2$. By Corollary 3 assume that $\alpha \neq 0, \alpha \neq \pi$, so $\alpha \in (0, \pi)$. Then $\sin\alpha > 0, a^2 + b^2 - 2ab\cos\alpha > (a-b)^2 \geq 0$ and

$$\begin{aligned}\lim_{d\to 0}\mathrm{AT}(a, b; d, \alpha) &= \lim_{d\to 0}\arctan\left(\frac{ab\sin\alpha + d^2\cot\alpha}{d\sqrt{a^2 + b^2 - 2ab\cos\alpha + d^2}}\right)\\ &= \mathrm{sign}(a)\,\mathrm{sign}(b)\,\mathrm{sign}(d)\frac{\pi}{2},\end{aligned}$$

so Theorem 2 gives

$$\lim_{d\to 0}\mathrm{lk}(L_1, L_2) = \frac{\mathrm{sign}(d)}{8}(\mathrm{sign}(a_1) - \mathrm{sign}(b_1))(\mathrm{sign}(b_2) - \mathrm{sign}(a_2)).$$

In the limit case $d = 0$, the line segments $L_1, L_2 \subset \{z = 0\}$ remain disjoint in the same plane if and only if both endpoint coordinates a_i, b_i have the same sign

for at least one of $i = 1, 2$, which is equivalent to $\text{sign}(a_i) - \text{sign}(b_i) = 0$, i.e., $\lim_{d\to 0} \text{lk}(L_1, L_2) = 0$ from the product above. Hence formula (3) is continuous under $d \to 0$ for any non-crossing segments. Any segments that intersect in the plane $\{z = 0\}$ when $d = 0$ have endpoint coordinates $a_i < 0 < b_i$ for both $i = 1, 2$ and have the limit

$$\lim_{d\to 0} \text{lk}(L_1, L_2) = \frac{\text{sign}(d)}{8}(-1-1)(1-(-1)) = -\frac{\text{sign}(d)}{2}$$

as required. □

Proof *(**Corollary 5**)* If $d \to \pm\infty$, while all other parameters of L_1, L_2 remain fixed, then the function

$$\text{AT}(a, b; d, \alpha) = \arctan\left(\frac{ab\sin\alpha + d^2\cot\alpha}{d\sqrt{a^2+b^2-2ab\cos\alpha+d^2}}\right)$$

from Theorem Theorem 2 has the limit $\arctan(\text{sign}(d)\cot\alpha) = \text{sign}(d)\,(\pi/2 - \alpha)$. Since the four AT functions in Theorem 2 include the same d, α, their limits cancel, so $\text{lk}(L_1, L_2) \to 0$. □

Proof *(**Corollary 6**)* If $a_i \to +\infty$, then $a_i \le b_i \to +\infty$, $i = 1, 2$. If $b_i \to -\infty$, then $b_i \ge a_i \to -\infty$, $i = 1, 2$. Consider the former case $a_i \to +\infty$, the latter is similar.

Since d, α are fixed,

$$a^2 + b^2 - 2ab\cos\alpha + d^2 \le (a+b)^2 + d^2 \le 5b^2$$

for large enough b. Since $\arctan(x)$ increases,

$$\text{AT}(a, b; d, \alpha) \ge \arctan\left(\frac{ab\sin\alpha + d^2\cot\alpha}{db\sqrt{5}}\right) \to \text{sign}(d)\frac{\pi}{2}$$

as $b \ge a \to +\infty$. Since the four AT functions in Theorem 2 have the same limit when their first two arguments tend to $+ \pm \infty$, these 4 limits cancel and we get $\text{lk}(L_1, L_2) \to 0$. □

Proof *(**Corollary 7**)* $\text{lk}(L_1, L_2) = 0$ for $d = 0$. It's enough to consider the case $d \ne 0$. Then

$$\text{AT}(a, b; d, \alpha) = \arctan\left(\frac{ab\sin\alpha + d^2\cot\alpha}{d\sqrt{a^2+b^2-2ab\cos\alpha+d^2}}\right)$$

from Theorem 2 is continuous. Let (say for $i = 1$) $a_1 \to b_1$, the case $b_1 \to a_1$ is similar. The continuity of AT implies that $\text{AT}(a_1, b_2; d, \alpha) \to \text{AT}(b_1, b_2; d, \alpha)$ and $\text{AT}(a_1, a_2; d, \alpha) \to \text{AT}(b_1, a_2; d, \alpha)$. In the limit all terms in Theorem 2 cancel, hence $\text{lk}(L_1, L_2) \to 0$. □

References

1. Ahmad, R., Paul, S., Basu, S.: Characterization of entanglements in glassy polymeric ensembles using the gaussian linking number. Phys. Rev. E **101**(2), 022503 (2020)
2. Arai, Z.: A rigorous numerical algorithm for computing the linking number of links. Nonlinear Theory Appl. **4**(1), 104–110 (2013)
3. Banchoff, T.: Self-linking numbers of space polygons. Indiana U. Math. J **25**, 1171–1188 (1976)
4. Bright, M., Kurlin, V.: Encoding and topological computation on textile structures. Comput. Graph. **90**, 51–61 (2020)
5. Cui, P., McMahon, D., Spackman, P., Alston, B., Little, M., Day, G., Cooper, A.: Mining predicted crystal structure landscapes with high throughput crystallisation: old molecules, new insights. Chem. Sci. **10**, 9988–9997 (2019)
6. DeTurck, D., Gluck, H., Komendarczyk, R., Melvin, P., Shonkwiler, C., Vela-Vick, D.: Pontryagin invariants and integral formulas for Milnor's triple linking number (2011). arXiv:1101.3374
7. Gauss, C.F.: Integral formula for linking number. Zur Mathematischen Theorie der Electrodynamischen Wirkungen, Collected Works **5**, 605 (1833)
8. Klenin, K., Langowski, J.: Computation of writhe in modeling of supercoiled DNA. Biopolymers Original Res. Biomol. **54**(5), 307–317 (2000)
9. Kontsevich, M.: Vassiliev's knot invariants. Adv. Soviet Math. **16**, 137–150 (1993)
10. Kurlin, V.: Compressed Drinfeld associators. J. Algebra **292**, 184–242 (2005)
11. Kurlin, V.: The Baker-Campbell-Hausdorff formula in the free metabelian lie algebra. J. Lie Theory **17**(3), 525–538 (2007)
12. Maxwell, J.C.: A Treatise on Electricity and Magnetism. I. Clarenton Press Series, Oxford (1873)
13. Panagiotou, E.: The linking number in systems with periodic boundary conditions. J. Comput. Phys. **300**, 533–573 (2015)
14. Panagiotou, E., Kauffman, L.H.: Knot polynomials of open and closed curves. Proc. Roy. Soc. A. **476**, 20200124 (2020). arxiv:2001.01303
15. Ricca, R.L., Nipoti, B.: Gauss' linking number revisited. J. Knot Theory Ramif. **20**(10), 1325–1343 (2011)
16. Vogel, T.: On the asymptotic linking number. Proc.Amer. Math. Soc. **131**, 2289–2297 (2003)
17. Vologodskii, A., Anshelevich, V.V., Lukashin, A., Frank-Kamenetskii, M.D.: Statistical mechanics of supercoils and the torsional stiffness of the DNA double helix. Nature **280**(5720), 294–298 (1974)

The Singularity Set of Optimal Transportation Maps

Zhongxuan Luo, Wei Chen, Na Lei, Yang Guo, Tong Zhao, and Xianfeng Gu

Abstract Optimal transportation plays an important role in many engineering fields, especially in deep learning. By Brenier theorem, computating optimal transportation maps is reduced to solving Monge–Ampère equations, which in turn is equivalent to construct Alexandrov polytopes. Furthermore, the regularity theory of Monge–Ampère equation explains mode collapsing issue in deep learning. Hence, computing and studying the singularity sets of OT maps become important. In this work, we generalize the concept of medial axis to power medial axis, which describes the singularity sets of optimal transportation maps. Then we propose a computational algorithm based on variational approach using power diagrams. Furthermore, we prove that when the measures are changed homotopically, the corresponding singularity sets of the optimal transportation maps are homotopic equivalent as well.

Z. Luo
Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, China
e-mail: zxluo@dlut.edu.cn

W. Chen
School of Software Technology, Dalian University of Technology, Dalian, China
e-mail: wei.chen@mail.dlut.edu.cn

N. Lei (✉)
DUT-RU ISE, Dalian University of Technolog, Dalian, China
e-mail: nalei@dlut.edu.cn

Y. Guo · X. Gu
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
e-mail: yangguo@cs.stonybrook.edu; tong.zhao@inria.fr

T. Zhao
INRIA Sophia-Antipolis and Telecom Paris, Paris, France
e-mail: gu@cs.stonybrook.edu

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_4

1 Introduction

Recently, optimal transportation theory plays an important role in deep learning, especially for generative models, such as Generative Adverseral Networks (GANs) [10], variational autoencoders (VAEs) [13] and so on. In deep learning, one of the major tasks is to transform a prescribed distribution to the data distribution, another is to measure the Wasserstein distance between two distributions. Both of them require computing optimal transportation maps.

The Brenier theorem reduces the computation of optimal transportation map to solving Monge–Ampère equation, which is equivalent to solve the Alexandrov problem in convex geometry. The construction of the Alexandrov polytope is converted to the computing of power diagram and weighted Delaunay. Therefore, the geometric approach to solve optimal transportation problem becomes valuable for deep learning tasks as in [2, 17].

Furthermore, one of the major challenges in deep learning is mode collapsing, which makes the training process unstable and difficult to converge. The regularity theory of optimal transportation map discovers the intrinsic reason for mode collapsing. Basically, if the support of the target measure is not simply connected or concave, the optimal transportation map may be discontinuous at the singularities. Therefore, computing and studying the singularity sets of optimal transportation maps become crucial.

Contributions In this work, we generalize the concept of medial axis to power medial axis, which describes the singularity sets of optimal transportation maps. Then we propose a computational algorithm based on a variational approach using power diagrams. Furthermore, we prove that when the measures are changed homotopically, the corresponding singularity sets of the optimal transportation maps are homotopic equivalent as well.

Outline The article is organized as follows: Sect. 2 explains how optimal transportation is applied for generative model in deep learning, and why singularity set is crucial to avoid mode collapsing. Section 3 introduces the convex geometric view of optimal transportation, including the Alexandrov and Minkowski theorems, Yau's geometric variational approach and Gelfand's secondary polytope theorem. Section 4 proves the main theorem, which shows the homotopy equivalence relation between the singularities of an optimal transportation map and the medial axis of the support of the target measure. The work is concluded in Sect. 5.

2 Generative Models and Optimal Transportation

GAN Model Generative modeling is an unsupervised learning task in machine learning, which is capable of automatically discovering and learning the pattern in input data, so that the model can be applied to generate new samples that plausibly

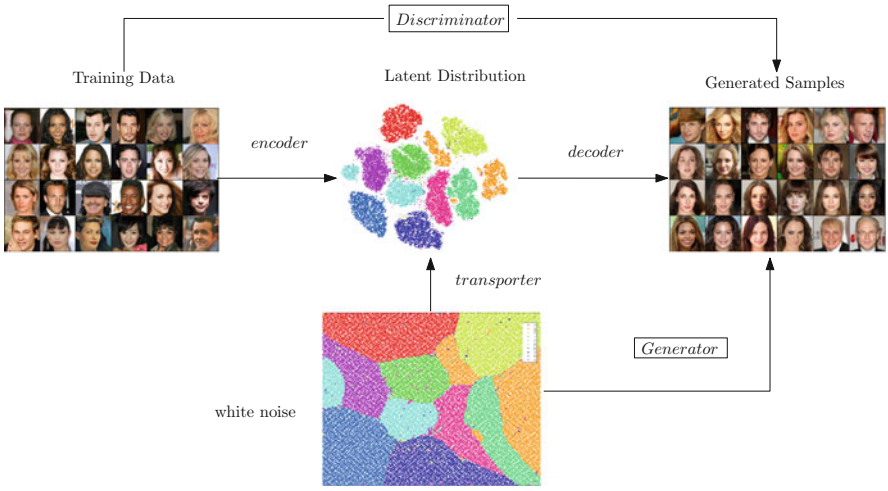


Fig. 1 Generative adversarial networks (GAN) model

could have been drawn from the original data set. Generative adversarial networks (GANs) [10] and variational autoencoders (VAEs) [13] emerge as the dominant approaches for unconditional image generation.

Figure 1 shows the general framework of GAN models. Each training sample, e.g., a human facial image, is treated as a point in the high dimensional image space. The natural class of images, e.g., human facial photos, is treated as a point cloud, which is close to a low dimensional manifold Σ, the so-called data manifold. The data set is modeled as a probability distribution μ on the data manifold Σ. The data manifold is mapped to the parameter domain by an encoding map, the parameter of each image is called the latent code, the parameter domain is called the latent space or the feature space Ω, the mapping is called the encoding map $\varphi\colon \Sigma \to \Omega$. The data distribution μ is push-forwarded by φ to the latent data distribution $\varphi_{\#}\mu$. The decoding map is the inverse of the encoding map, $\varphi^{-1}\colon \Omega \to \Sigma$, which maps the latent code to the image on the data manifold.

In the latent space, there is a known probability distribution, the so-called white noise ν, which is usually uniform distribution or Gaussian distribution. A transporter maps the white noise to the latent data distribution, the transportation map is denoted as $T\colon \nu \to \varphi_{\#}\mu$.

The composition of the transporter and the decoder is called a *generator*, $\varphi^{-1} \circ T$, which maps a white noise sample to a generated image. Equivalently, the generator transports the white noise distribution to the generated data distribution on the data manifold. A *discriminator* computes the distance between the real data distribution μ and the generated data distribution $\left(\varphi^{-1} \circ T\right)_{\#}\nu$. The generator optimizes the transportation map to minimize the distance between the real distribution and the generated distribution; the discriminator maximizes some type of functional to

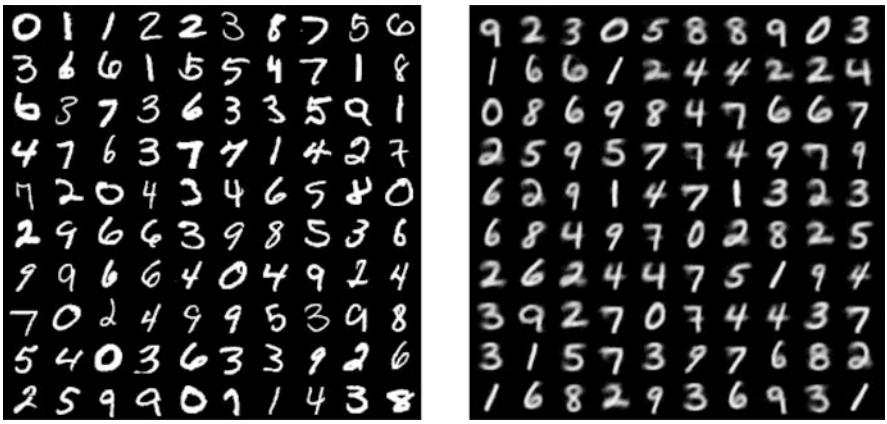


Fig. 2 Left: input MNIST data set. Right: the generated images

differentiate the two distributions. The competition between the generator and the discriminator reaches the Nash equilibrium eventually, at that state human beings can not tell the difference between the real images and the generated ones. Recently, optimal transportation has been broadly applied for generative models. In many models, the generator computes the optimal transportation map between the white noise distribution and the latent code distribution; the discriminator computes Wasserstein distance between the real data distribution and the generated data distribution.

Figure 2 shows one example of a generative model, the left frame shows the MNIST data set of hand-written digit images, the right frame shows the generated images. The MNIST data set is encoded by a UMap model, the latent data distribution is shown in Fig. 3 (left). Each digit corresponds to one cluster. We put 10×10 samples on the latent space, and the decoded images are shown in the right frame in Fig. 3. It is clear that if a generated latent code falls into a cluster, then the generated image is clean and sharp; otherwise, if the generated code falls into the gaps among the clusters, the generated image is obscure and like the mixture of two digits.

Mode Collapsing/Mode Mixture Despite GANs' advantages, they have critical drawbacks. (1) Training of GANs is tricky and sensitive to hyperparameters. (2) GANs suffer from mode collapse, in which the generator only learns to generate few modes of data distribution while missing others, although samples from the missing modes occur throughout the training data (see, e.g., [9]). While for the VAEs, the encoder is used to map the data distribution to a Gaussian latent distribution, which is then mapped back to the data distribution by the decoder. While standard VAEs tend to capture all modes, they often generate ambiguous images on multimodal real data distributions.

In the work of Yau et al. [2, 15], the intrinsic reason for mode collapsing/mode mixture has been explained from the geometric point of view. By Brenier's polar

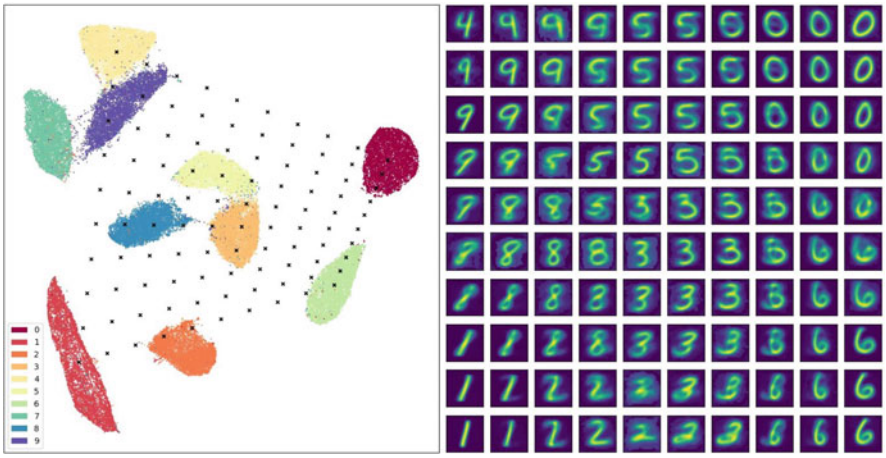


Fig. 3 Left: latent codes of the MNIST data set encoded by a UMap model. Right: the latent codes are decoded and mapped back to the images

factorization theorem [4, 5, 5] and Figalli's regularity theorem [6, 7] (Theorem 5 in Appendix B), if the support of the target distribution is not convex, then there will be *singularity sets* on the support of the source distribution, such that the transportation map is discontinuous on these sets. Generally, the generators/decoders are expressed by deep neural networks, which can only represent continuous mappings, thus they can't approximate the transport maps with high fidelity. This intrinsic conflict induces the mode collapsing and mode mixture in conventional generative models. Figure 4 shows the optimal transportation map from the uniform distribution on the disk to the latent distribution of MNIST data set, i.e., Fig. 3 (left). Since the latent

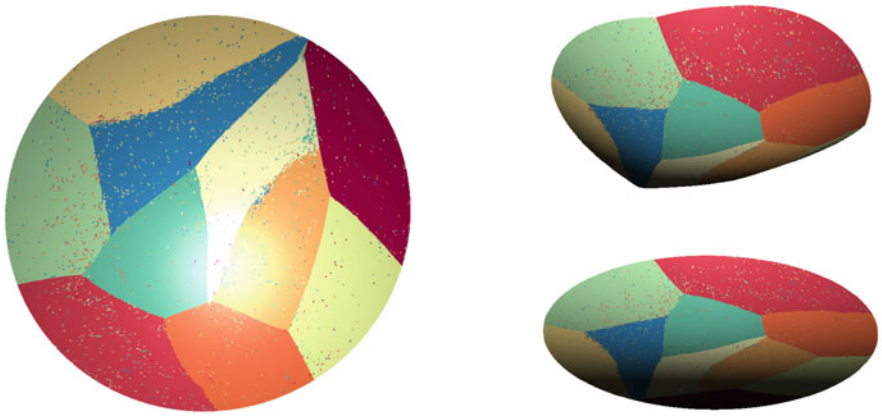


Fig. 4 Optimal transportation map from the uniform distribution on the disk to the latent data distribution of MNIST data set, Fig. 3 (left)

distribution has 10 modes (connected components), the singularity set partitions the disk into ten segments, each segment is mapped onto one cluster. The mapping itself is discontinuous on the singularity set, therefore can't be represented by a neural network. The right frame shows the Brenier potential, which is continuous and differentiable almost everywhere. The projection of the non-differential points is the singularity set. Therefore, the neuron network can present the Brenier potential. The image of the optimal transportation map will cover all the modes, so no mode collapsing will happen. Furthermore, the image of the transportation map won't fall into the gaps among the modes, this will eliminate the mode mixture. Therefore, it is important to study the singularity of optimal transportation maps.

3 Convex Geometric View of Optimal Transportation

There is a close relation between the Alexandrov theorem in convex geometry and the Brenier theorem in optimal transportation. This section will explain this intrinsic relation.

3.1 Optimal Transportation Map

Suppose $\Omega, \Omega^* \subset \mathbb{R}^d$ are domains in Euclidean space, with probability measures μ and ν respectively, satisfying the equal total mass condition

$$\mu(\Omega) = \nu(\Omega^*).$$

The transportation map $T\colon \Omega \to \Omega^*$ is *measure preserving*, if for any Borel set $B \subset \Omega^*$,

$$\int_{T^{-1}(B)} \mathrm{d}\mu(x) = \int_B \mathrm{d}\nu(y)\,.$$

We denote measure preserving condition as $T_{\#}\mu = \nu$.

Monge raised the optimal transportation map problem: given a transportation cost function $c\colon \Omega \times \Omega^* \to \mathbb{R}^+$, find a transportation map $T\colon \Omega \to \Omega^*$ that minimizes the total transportation cost,

$$\text{(MP)} \quad \min \left\{ \int_\Omega c(x, T(x)) : T\colon \Omega \to \Omega^*,\ T_{\#}\mu = \nu \right\}.$$

The minimizer is called the *optimal transportation map*. The transportation cost of the optimal transportation map is called the *Wasserstein distance* between the measures.

Theorem 1 (Brenier[5]) *Given a compact domain $\Omega \subset \mathbb{R}^d$, with absolutely continuous probability measures μ and ν, $\partial\Omega$ has zero Lebesgue-measure. The transportation cost is $c(x, y) = |x - y|^2/2$, then there exists a convex function $u : \Omega \to \mathbb{R}$ unique upto a constant, the so-called Brenier potential, and the optimal transportation map is given by the gradient of u, $T = \nabla u$.*

The Brenier potential satisfies the Monge–Ampère equation

$$\det D^2 u(x) = \frac{\mathrm{d}\mu(x)}{\mathrm{d}\nu \circ \nabla u(x)}, \tag{1}$$

with the boundary condition $\nabla u(\Omega) = \Omega^*$. The unique optimal transportation map is given by $T = \nabla u$.

3.2 Alexandrov Solution

For numerical computation, the Brenier potential is approximated by a piecewise linear convex function, whose graph is a convex polytope. The sub-gradient of a convex function u at x is defined as

$$\partial u(x) := \left\{ p \in \mathbb{R}^d : u(z) \geq \langle p, z - x \rangle + u(x) \ \forall z \in \Omega \right\}.$$

The sub-gradient defines a set-valued map $\partial u : \Omega \to \Omega^*$, $x \mapsto \partial u(x)$. We can use the sub-gradient map to replace the gradient map in the Monge–Ampère equation (1) and rewrite it as

$$(\partial u)_{\#}\mu = \nu, \tag{2}$$

or, equivalently, $\forall$ Borel $B \subset \Omega^*$, $\mu((\partial u)^{-1}(B)) = \nu(B)$. Thus, here, u is called the *Alexandrov solution*. In fact, the Alexandrov solution is equivalent to the Alexandrov polytope, and the Brenier theorem is equivalent to the Alexandrov theorem in convex geometry.

3.3 Minkowski and Alexandrov Theorems

In this subsection, we briefly recall the basic concepts and theorems of Minkowski and Alexandrov theorems in convex geometry, which can be described by Monge–Ampère equation and closely related to power diagrams and weighted Delaunay triangulations in computational geometry. This intrinsic connection gives the theoretical tool to study the Alexandrov polytope space. Details can be found in [1] and [11].

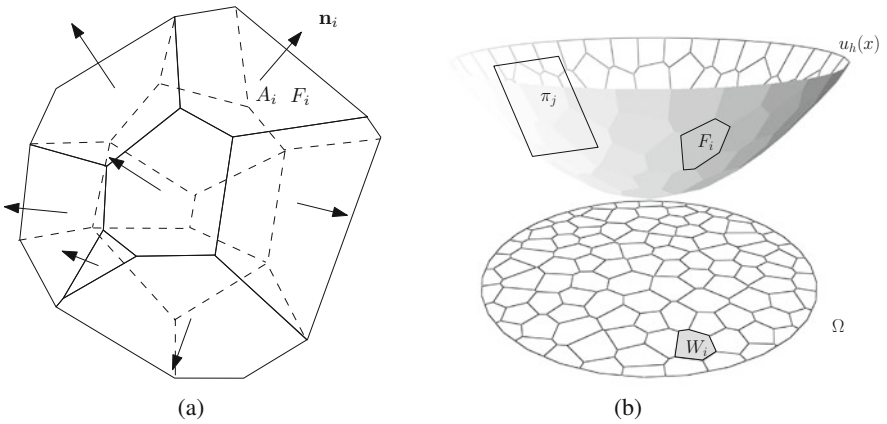


Fig. 5 Minkowski and Alexandrov theorems for convex polytopes with prescribed normals and face areas. (**a**) Minkowski theorem. (**b**) Alexandrov theorem

Minkowski proved the existence and the uniqueness of convex polytope with prescribed face normals and face areas (see Fig. 5).

Theorem 2 (Minkowski) *Suppose $n_1, \ldots, n_k$ are unit vectors which span $\mathbb{R}^n$ and $v_1, \ldots, v_k > 0$ such that $\sum_{i=1}^{k} v_i n_i = 0$. There exists a compact convex polytope $P \subset \mathbb{R}^n$ with exactly k codimension-1 faces $F_1, \ldots, F_k$ so that n_i is the outward normal vector to F_i and v_i is the volume of F_i. Furthermore, such P is unique up to a translation.*

Minkowski's proof is variational and suggests an algorithm to find the polytope. Minkowski theorem for unbounded convex polytopes was considered and solved by A.D. Alexandrov and his student A. Pogorelov. In his book on convex polyhedra, Alexandrov proved the following fundamental theorem [1, Theorem 7.3.2 and Theorem 6.4.2]:

Theorem 3 (Alexandrov [1]) *Suppose Ω is a compact convex polytope with non-empty interior in $\mathbb{R}^n$, $n_1, \ldots, n_k \subset \mathbb{R}^{n+1}$ are distinct k unit vectors, the $(n+1)$-th coordinates are negative, and $v_1, \ldots, v_k > 0$ so that $\sum_{i=1}^{k} v_i = \mathrm{vol}(\Omega)$. Then there exists a convex polytope $P \subset \mathbb{R}^{n+1}$ with exact k codimension-1 faces $F_1, \ldots, F_k$, so that n_i is the normal vector to F_i and the intersection between Ω and the projection of F_i is with volume v_i. Furthermore, such P is unique up to vertical translation.*

Alexandrov's proof is based on the algebraic topology and is non-constructive. Aurenhammer [3] gave a constructive proof using the power diagram. Gu et al. [11] gave another variational proof for the generalized Alexandrov theorem stated in terms of convex functions. The energies in [3] and [11] are Legendre dual to each other.

Definition 1 (Alexandrov Polytope) Given $Y = \{y_1, \ldots, y_k\}$, $y_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, k$, and $h = (h_1, \ldots, h_k) \in \mathbb{R}^k$, the upper envelope of the hyper-planes $\pi_i(x) = \langle x, y_i \rangle - h_i$ is

$$u_h(x) = \max_{i=1}^{k} \{\pi_i(x)\} = \max_{i=1}^{k} \{\langle y_i, x \rangle - h_i\}. \tag{3}$$

The graph of u_h is called the Alexandrov Polytope, denoted as $P(Y, h)$.

The projection of the Alexandrov polytope induces a *power diagram* of $\mathbb{R}^n$, each cell $W_i(h)$ is a closed convex polytope,

$$\mathbb{R}^d = \bigcup W_i(h) = \bigcup_{i=1}^{k} \left\{ x \in \mathbb{R}^d \mid \pi_i(x) \geq \pi_j(x),\ i \neq j \right\}, \tag{4}$$

where $W_i(h)$'s are called power cells. The power diagram can be reformulated using power distance

$$W_i(h) = \left\{ x \in \mathbb{R}^d \mid \mathrm{pow}_h(x, y_i) \leq \mathrm{pow}_h(x, y_j),\ i \neq j \right\},$$

where the *power distance* between x and y_i is defined as

$$\mathrm{pow}_h(y_i, x) \coloneqq \frac{1}{2}|x - y_i|^2 - r_i^2, \tag{5}$$

where r_i^2 is the *power* associated with y_i, defined as

$$r_i^2 \coloneqq \frac{1}{2}|y_i|^2 - h_i. \tag{6}$$

Given a probability measure μ defined on Ω, the volume of $W_i(h)$ is defined as

$$w_i(h) \coloneqq \mu(W_i(h) \cap \Omega) = \int_{W_i(h) \cap \Omega} \mathrm{d}\mu\,.$$

Gu et al. gave a constructive proof for the Alexandrov theorem.

Theorem 4 (Gu et al. [11]) *Let Ω be a compact convex domain in $\mathbb{R}^n$, $Y = \{y_1, \ldots, y_k\}$ be a set of distinct points in $\mathbb{R}^n$ and μ a probability measure on Ω. Then for any $\nu_1, \ldots, \nu_k > 0$ with $\sum_{i=1}^{k} \nu_i = \mu(\Omega)$, there exists $h = (h_1, \ldots, h_k) \in \mathbb{R}^k$, unique up to adding a constant $(c, \ldots, c)$, so that $w_i(h) = \nu_i$, for all i. The vectors h are exactly maximum points of the concave function*

$$E(h) = \int_0^h \sum_{i=1}^{k} w_i(\eta)\, \mathrm{d}\eta_i - \sum_{i=1}^{k} h_i \nu_i, \tag{7}$$

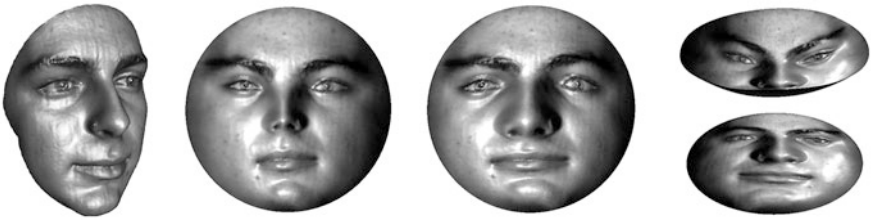


Fig. 6 Experiment one: the 3D face (1st) is mapped to the unit disk by a Riemann mapping (2nd). The optimal transportation map from uniform distribution to the measure induced by the Riemann mapping is computed, the image is in (3rd), the Brenier potential is in (4th)

on the open convex set (admissible height space)

$$\mathcal{H}_\Omega(Y) = \left\{ h \in \mathbb{R}^k \mid w_i(h) > 0 \, \forall i \right\}. \tag{8}$$

Furthermore, ∇u_h *minimizes the quadratic cost*

$$\int_\Omega |x - T(x)|^2 \, \mathrm{d}\mu(x)$$

among all transportation maps $T_\# \mu = \nu$ *with the Dirac measure* $\nu = \sum_{i=1}^k \nu_i \delta(y - y_i)$.

Figure 6 demonstrates an optimal transportation map obtained by Alexandrov polytope method. A male facial surface is digitized to a triangle mesh M (first frame). The surface is conformally mapped onto the planar unit disk by a Riemann mapping (second frame), the planar images of the vertices are denoted as $y_1, y_2, \ldots, y_k$. For each vertex v_i, the total area of the triangular faces adjacent to it is denoted as ν_i. By scaling, the total area $\sum_{i=1}^k \nu_i = \pi$. Then we compute the optimal transportation map (third frame) from the unit disk with the uniform distribution to $\nu = \sum_{i=1}^k \nu_i \delta(y - y_i)$. The Brenier potential is shown in the fourth frame.

3.4 Secondary Polytope and Secondary Power Diagram

In this subsection, we briefly recall the basic concepts and theorems of Gelfand's secondary polytope theory, and its dual secondary power diagram, details can be found in [8, 14] and [16]. This shows the admissible solution space $\mathcal{H}_\Omega(Y)$ has a cell decomposition.

Secondary Polytope Let $Y = \{y_1, y_2, \ldots, y_k\}$ be a *point configuration*, a finite set of distinct points in $\mathbb{R}^d$, $Conv(Y)$ is the convex hull of Y. A triangulation T of $(Y, Conv(Y))$ decomposes the interior volume bounded by $Conv(Y)$ into simplices with vertices in Y. Some $y_i \in Y$ may not appear as a vertex of a simplex.

Given a triangulation T, a *piecewise linear* function $g: Conv(Y) \to \mathbb{R}$ is affine-linear on every simplex of T. Furthermore, g is *concave*, if for any $x, y \in Conv(Y)$, $g(tx + (1-t)y) \geq tg(x) + (1-t)g(y)$.

Definition 2 (Coherent Triangulation) A triangulation T of $(Y, Conv(Y))$ is called *coherent* if there exists a concave piecewise linear function whose domains of linearity are precisely (maximal) simplices of T.

Let T be a triangulation of $(Y, Conv(Y))$. The *characteristic function* of T, $\varphi_T: Y \to \mathbb{R}$, is defined as follows:

$$\varphi_T(\omega) = \sum_{\omega \sim \sigma} \text{vol}(\sigma), \tag{9}$$

where the summation is over all (maximal) simplices of T for which ω is a vertex. If ω is not a vertex of any simplex of T, then $\varphi_T(\omega) = 0$.

Definition 3 (Secondary Polytope) The secondary polytope $\Sigma(Y)$ is the convex hull in the space $\mathbb{R}^Y$ of the vectors ϕ_T for all the triangulations T of $(Y, Conv(Y))$.

The secondary polytope has the following properties:

Theorem 5

(a) The secondary polytope $\Sigma(Y)$ has dimension $k - n - 1$ where $k = \#(Y)$.
(b) Vertices of $\Sigma(Y)$ are precisely the characteristic functions got for all coherent triangulations T of $(Y, Conv(Y))$. If T is a coherent triangulation of $(Y, Conv(Y))$ then $\varphi_T \neq \varphi_{T'}$ for any other triangulation T' of (Y, Conv(Y)).

Secondary Power Diagram Primal weighted Delaunay triangulations are dual to power diagrams; secondary polytopes are dual to secondary power diagrams. The secondary power diagram is the cell decomposition of the admissible height space in (8), where each cell corresponds to a weighted Delaunay triangulation (coherent triangulation) of Y. Details can be found in the secondary power diagram theory in [16].

Fixing a triangulation T of $(Y, Conv(Y))$, a simplex $\sigma \in T$ has volume $\text{vol}(\sigma)$. Given a height vector $\mathbf{h}$, a linear function $\pi_T(\mathbf{h})$ is defined by

$$\pi_T(\mathbf{h}) = \frac{1}{n+1} \sum_{y_i \in Y} \sum_{y_i \sim \sigma, \sigma \in T} \text{vol}(\sigma) h_i. \tag{10}$$

Theorem 6 (Secondary Power Diagram) *Given a point configuration* $Y = \{y_1, y_2, \ldots, y_k\} \subset \mathbb{R}^n$*, a convex domain* $\Omega \subset \mathbb{R}^n$ *containing the origin,*

1. *For each non-degenerated weighted Delaunay triangulation* $T \in \mathcal{T}(Y)$*, if the cell* D_T *is non-empty, then it is convex. Furthermore, if* $h \in D_T$*, then* $\lambda h \in D_T$ *for all* $0 < \lambda < 1$.
2. *The cell decomposition of the Alexandrov power diagram space*

$$\mathcal{H}_\Omega(Y) = \bigcup_{T \in \mathcal{T}(Y)} D_T \tag{11}$$

is a power diagram, produced by the projection of the upper envelope

$$U(Y) = \max_{T \in \mathcal{T}(Y)} \{-\pi_T(\mathbf{h})\}, \tag{12}$$

where the hyper-planes $\pi_T(\mathbf{h}) \subset \mathbb{R}^{k+1}$ *is the triangulation volume of* T *in (10).*
3. *Suppose* T *is a non-degenerated coherent triangulation* $T \in \mathcal{T}(Y)$*, then* D_T *is non-empty.*

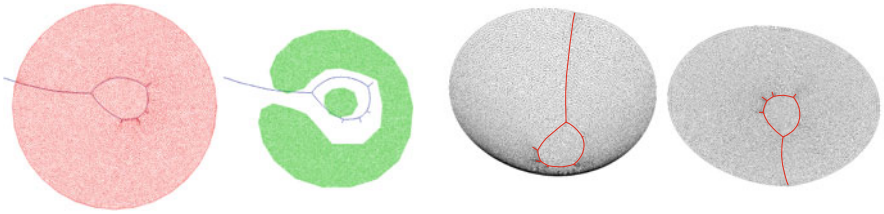
4 Singularity Set of Optimal Transportation Map

The optimal transportation maps may not be globally continuous.The discontinuity singularity set plays an important role in generative models in deep learning.

4.1 Singularity Set

Theorem 7 (Figalli Regularity [7]) *Let* $\Omega, \Lambda \subset \mathbb{R}^d$ *be two bounded open sets, let* $f, g \colon \mathbb{R}^d \to \mathbb{R}^+$ *be two probability densities, that are zero outside* Ω, Λ *and are bounded away from zero and infinity on* Ω, Λ *respectively. Denote* $T = \nabla u \colon (\Omega, f dx) \to (\Lambda, g dy)$ *the optimal transportation map provided by Brenier theorem. Then there exists two relatively closed sets* $\Sigma_\Omega \subset \Omega$ *and* $\Sigma_\Lambda \subset \Lambda$ *with* $|\Sigma_\Omega| = |\Sigma_\Lambda| = 0$ *such that* $T \colon \Omega \setminus \Sigma_\Omega \to \Lambda \setminus \Sigma_\Lambda$ *is a homeomorphism of class* $C^{0,\alpha}_{\text{loc}}$ *for some* $\alpha > 0$.

As shown in Fig. 7, the domain Ω is the unit disk, the range Λ has two connected components. Both the density functions f and g are constants, namely the probability measures are uniform distributions. The optimal transportation map T is the gradient map of the Brenier potential $u \colon \Omega \to \mathbb{R}$. The singularity set Σ_Ω is a loop inside Ω, the image of $\Omega \setminus \Sigma_\Omega$ covers Λ. From the bottom view, it is easy to see that the Brenier potential is C^1 almost everywhere, except at the Σ_Ω, where the potential is only C^0. The singularity set Σ_Ω is a graph, which can be decomposed



(a) (b) (c) (d)

Fig. 7 The singularity set of an optimal transportation map from the uniform distribution on a disk to the island shape. The blue curves show the singularities on the domain, the red curves show the non-differentiable points on the Brenier potential. (**a**) Power diagram. (**b**) Weighted delaunay triangulation. (**c**) Brenier potential (bottom view). (**d**) Brenier potential (top view)

into arcs and branching points. Take a point in the singularity set, $x \in \Sigma_\Omega$, if x is in the interior of an arc, the sub-differential $\partial u(x)$ is a line segment; if x is a branching point, $\partial u(x)$ is a two-dimensional convex set.

The singularity set can be formulated by the generalized medial axis concept.

Definition 4 (Power Medial Axis) Suppose $\Lambda \subset \mathbb{R}^d$ is a domain in the Euclidean space, $h\colon \Lambda \to \mathbb{R}$ is a convex function, which defines the power distance as in (5),

$$\mathrm{pow}_h(x, y) := \frac{1}{2}|x - y|^2 - \left(\frac{1}{2}|y|^2 - h(y)\right).$$

For each point $x \in \mathbb{R}^d$, the *closest point* of x to Λ is defined as

$$\mathrm{Cl}_\Lambda(p, h) := \arg\inf_{y \in \Lambda} \mathrm{pow}_h(p, y),$$

the power medial axis is defined as

$$\mathcal{M}_\Lambda(h) := \left\{ x \in \mathbb{R}^d \mid |\mathrm{Cl}_\lambda(p, h)| > 1 \right\}.$$

Proposition 1 (Singularity Set) *Suppose $\Omega, \Lambda \subset \mathbb{R}^d$ are compact domains, with absolutely continuous measures μ and ν, $\partial\Omega$ has zero Lebesgue-measure. The transportation cost is quadratic Euclidean distance. Suppose $u\colon \Omega \to \mathbb{R}$ is the Brenier potential, $\nabla u\colon \Omega \to \Lambda$ is the optimal transportation map. $u^*\colon \Lambda \to \mathbb{R}$ is the Legendre dual of u, then the singularity set of the optimal transportation map is given by*

$$\Sigma_\Omega = \mathcal{M}_\Lambda(u^*) \cap \Omega. \tag{13}$$

Proof Let the cost function $c(x, y) = \langle x, y\rangle$. Suppose $x_0 \in \omega$, $y_0 \in \Lambda$, and $y_0 \in \partial u$, then

$$u(x_0) + u^*(y_0) = \langle x_0, y_0\rangle,$$
$$u(x_0) + u^*(y) \geq \langle x_0, y\rangle \quad \forall y \in \Lambda.$$

Therefore, for any $y \in \Lambda$,

$$u^*(y_0) - \langle x_0, y_0\rangle \leq u^*(y) - \langle x_0, y\rangle,$$
$$\frac{1}{2}|x_0 - y_0|^2 - \left(\frac{1}{2}|y_0|^2 - u^*(y_0)\right) \leq \frac{1}{2}|x_0 - y|^2 - \left(\frac{1}{2}|y|^2 - u^*(y)\right),$$

this means $\forall y \in \Lambda$, $\text{pow}_{u^*}(x_0, y_0) \leq \text{pow}_{u^*}(x_0, y)$ and therefore $y_0 \in \text{Cl}_\Lambda(x_0, u^*)$. Namely, y_0 is the closest point in Λ (under the power distance) to x_0, the optimal transportation map T maps each point x_0 in Ω to its closest point y_0 in Λ. Inversely, if $y_0 \in \Lambda$ is the closest point to $x_0 \in \Omega$, then $y_0 \in \partial u(x_0)$.

Suppose $x \in \Omega$ and x is in the power medial axis of Λ, $x \in \mathcal{M}_\Lambda(u^*)$, then it has two closest points $y_1, y_2 \in \Lambda$, hence $y_1, y_2 \in \partial u(x)$. u is not differentiable at x, x is a singularity of u, this shows

$$\mathcal{M}_\Lambda(u^*) \cap \Omega \subset \Sigma_\Omega.$$

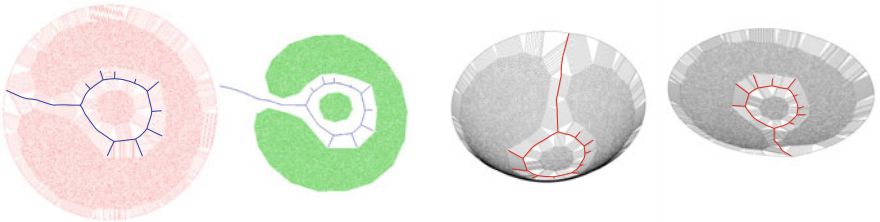
Inversely, suppose $x \in \Sigma_\Omega$, then $\partial u(x)$ has at least two points $y_1, y_2 \in \Lambda$, which are closest to x and with equal power distances. Hence, x is in the power medial axis of Λ, this shows

$$\Sigma_\Omega \subset \mathcal{M}_\Lambda(u^*) \cap \Omega.$$

Thus, (13) holds. □

4.2 Algorithm for Singularity Set

Suppose $\Omega, \Lambda \subset \mathbb{R}^d$ are compact domains, Ω is convex. We densely sample the boundary and the interior of Λ, the samples are denoted as $Y = \{y_1, y_2, \ldots, y_n\}$. The boundary samples are triangulated to form a polyhedral hyper-surface, denoted as ∂Y. Then ∂Y approximates the boundary surface of Λ, $\partial\Lambda$. Given the powers $\{w_1, w_2, \ldots, w_n\}$, or, equivalently, the height $\mathbf{h} = (h_1, h_2, \ldots, h_n)$, the power diagram is denoted as $\mathcal{D}_Y(\mathbf{h})$ as defined in (4). The power medial axis is given by the union of co-dimension 1 faces f_{ij}, which are the intersections of two power Voronoi cells $W_i(\mathbf{h})$ and $W_j(\mathbf{h})$, the corresponding y_i and y_j are in the boundary



(a) (b) (c) (d)

Fig. 8 The blue planar graph shows the medial axis of the island-shaped polygon. The red curve shows the non-differential points on the Brenier potential. (**a**) Voronoi diagram. (**b**) Delaunay triangulation. (**c**) Breiner potential (bottom view). (**d**) Breiner potential (top view)

polyhedron ∂Y, but not adjacent $y_i \not\sim y_j$ in ∂Y,

$$\mathcal{M}_Y(\mathbf{h}) = \bigcup_{y_i, y_j \in \partial Y, y_i \not\sim y_j} \{W_i(\mathbf{h}) \cap W_j(\mathbf{h})\}. \tag{14}$$

The *singularity set* of the discrete optimal transportation map is

$$\Sigma_\Omega(\mathbf{h}) = \mathcal{M}_Y(\mathbf{h}) \cap \Omega. \tag{15}$$

Figure 8 illustrates the power medial axis obtained by this algorithm, where Ω is the unit disk, Λ is an island-shaped planar polygon with two connected components. We densely sample the interior and the boundary of Λ to get a discrete point set $Y = \{y_i\}_{i=1}^n$. The boundary samples are connected consecutively to form ∂Y. We set all the powers to be zeros, $\{r_i = 0\}_{i=1}^n$, equivalently, $\{h_i = |y_i|^2/2\}_{i=1}^n$. Then frame (a) shows the Voronoi diagram of Y, the blue graph is the conventional medial axis. Frames (c) and (d) show the non-differential points on the graph of the Brenier potential as the red curves. If the height is changed, the power medial axis (singularities of the optimal transportation map) can be obtained by the same algorithm.

4.3 Singularity Set Homotopy Equivalence

For the optimal transportation map, $T = \nabla u \colon (\Omega, \mu) \to (\Lambda, \nu)$, if we change the target measure ν, the optimal transportation map T will be changed accordingly. We can show that the singularity sets of the optimal transportation maps are homotopic to each other. We first show this in the discrete setting, then generalize it using the stability of optimal transportation maps.

Definition 5 (Minkowski Sum) Suppose A and B are subsets on $\mathbb{R}^n$, the Minkowski sum of them is defined as

$$A \oplus B \coloneqq \{\, p + q \mid p \in A,\ q \in B \,\}.$$

Theorem 8 (Singular Set Homotopy) *Given a convex domain $\Omega \subset \mathbb{R}^d$ and the discrete point set $Y = \{y_i\}_{i=1}^n$, fix the source measure μ, whose density function is absolutely continuous. The target measure $\nu = \sum_{i=1}^n \nu_i \delta(y - y_i)$, $\nu_i > 0$, $i = 1, 2, \ldots, n$, such that $\mu(\Omega) = \sum_{i=1}^n \nu_i$. Given two target measures ν_0 and ν_1, the corresponding heights are $\mathbf{h}_0$ and $\mathbf{h}_1$ respectively, then their singularity sets are homotopic equivalent to each other:*

$$\Sigma_\Omega(\mathbf{h}_0) \sim \Sigma_\Omega(\mathbf{h}_1).$$

Proof Theorem 4 shows that admissible height space $\mathcal{H}_Y$ in (8) is convex, therefore the line segment connecting $\mathbf{h}_0$ and $\mathbf{h}_1$ is contained in $\mathcal{H}_Y$, $\gamma(t) \coloneqq (1-t)\mathbf{h}_0 + t\mathbf{h}_1$. According to the Secondary Power Diagram Theorem 6, $\mathcal{H}_Y$ has a cell decomposition, $\mathcal{H}_Y = \bigcup_{T \in \mathcal{T}(Y)} \mathcal{D}_T$, each cell corresponds to a weighted Delaunay triangulation (also a combinatorial structure of the power diagram of Ω). Suppose the line $\gamma(t)$ crosses a set of cells corresponding to triangulations $T_0, T_1, \ldots, T_k$. Then the unit interval is divided into segments, $0 = t_0 < t_1 < t_2 \cdots < t_k = 1$, satisfying the following conditions:

1. $\gamma(t) \in \mathcal{D}_{T_i}, \forall t \in [t_i, t_{i+1}]$;
2. $\gamma(t_i) \in \mathcal{D}_{T_i} \cap \mathcal{D}_{T_{i+1}}$. □

Step 1. For any $t \in (t_i, t_{i+1})$, all the weighted Delaunay triangulations $T(\gamma(t))$ have the same combinatorial structure. All the corresponding power diagrams share the same combinatorial structure as well. The Brenier potential $u_{\gamma(t)}(x) = \max_{i=1}^k \left\{ \langle y_i, r \rangle - \gamma(t)_i \right\}$ can be written as the linear combination

$$u_{\gamma(t)} = (1-\lambda) u_{\gamma(t_i)} + \lambda u_{\gamma(t_{i+1})}.$$

where $\lambda = (t_{i+1} - t)/(t_{i+1} - t_i)$. The graph of the Brenier potential can be written as a linear combination using Minkowski sum,

$$G(\gamma(t)) = \lambda G(\gamma(t_i)) \oplus (1-\lambda) G(\gamma(t_{i+1})),$$

where $G(\gamma(t))$ is the graph of the function $u_{\gamma(t)}$. Therefore, as the projection of the graphs of the Brenier potentials, each power cell can be represented as the Minkowski sum

$$W_i(\gamma(t)) = \lambda W_i(\gamma(t_i)) \oplus (1-\lambda) W_i(\gamma(t_{i+1})),$$

This implies the singularity set also satisfies the linear combination relation

$$\Sigma_\Omega(\gamma(t)) = \lambda\Sigma_\Omega(\gamma(t_i)) \oplus (1-\lambda)\Sigma_\Omega(\gamma(t_{i+1})).$$

Step 2. At the critical point t_i, we want to show

$$\lim_{t\to t_i^-} \Sigma_\Omega(\gamma(t)) = \lim_{t\to t_i^+} \Sigma_\Omega(\gamma(t)). \tag{16}$$

As shown in Fig. 9, we prove it for the two-dimensional case first; the proofs for the general dimensional cases are very similar. Fix a weighted Delaunay triangulation T, we choose two adjacent triangles $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$. The power circle associated with vertex v_i is $c(v_i, r_i)$, the power is r_i^2. Then, for each triangle, there is a unique circle $c(o, r)$ (red one) orthogonal to all three vertex circles, the so-called power circle of the face. The power center o and the power radius r satisfy

$$|v_i - o|^2 = r_i^2 + r^2, \quad |v_j - o|^2 = r_j^2 + r^2, \quad |v_k - o|^2 = r_k^2 + r^2.$$

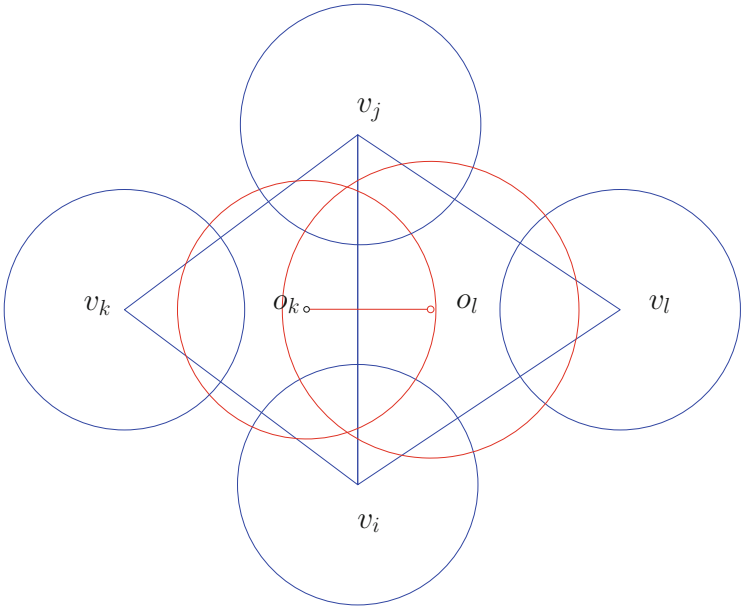


Fig. 9 Power diagram configuration

As shown in Fig. 9, the power centers for the two faces are o_k and o_l respectively. The line segment connecting the two power centers $[o_k, o_l]$ in the power diagram is dual to the edge $[v_i, v_j]$ in the weighted Delaunay triangulation.

Along the curve $\gamma(t)$, the vertex powers change continuously. Suppose when $t < t_i$ and approaches to t_i, the two power centers o_k and o_l get closer and closer. At the critical point t_i, o_k coincides with o_l, and the triangulation is changed by swapping edge $[v_i, v_j]$ to $[v_k, v_l]$. When t increases further, $t > t_i$, the power center of $[v_k, v_i, v_l]$ and that of $[v_l, v_j, v_k]$ diverge. This shows at the critical time t_i, the power diagram edge $[v_i, v_j]$ shrinks to a point, and grows to a new edge $[v_k, v_l]$. Therefore, the power diagram changes continuously. Hence, the singularity set also changes continuously, namely (16) holds. For higher dimensional cases, the edge swap is replaced by a bistellar transformation [12], and the proof is exactly the same.

Combining Steps 1 and 2, we obtain for $0 \leq i < k$,

$$\Sigma_\Omega(\gamma(t_i^+)) \sim \Sigma_\Omega(\gamma(t_{i+1}^-)) = \Sigma_\Omega(\gamma(t_{i+1}^+)).$$

Hence,

$$\Sigma_\Omega(\gamma(t_0)) \sim \Sigma_\Omega(\gamma(t_1^-)) = \Sigma_\Omega(\gamma(t_1^+)) \sim \cdots \sim \Sigma_\Omega(\gamma(t_k)).$$

Figures 7 and 8 show the homotopy relation between the singularity sets of optimal transportation maps for different target measures. We see that minor branches may disappear, but the loops and major branches are well preserved. Figure 10 shows the homotopy deformation between the singularity sets of the optimal transportation maps from the unit disk to a sea-horse shaped domain with different target measures. In the beginning, we compute the conventional medial axis, and the target measure on each y_i equals to the corresponding cell area. The final target measure is uniform distribution.

A more complicated example is illustrated in Fig. 11, the optimal transportation maps from a planar spiral shape to the unit disk are computed, and the corresponding singularity sets are extracted. We can see that the singularity sets (power medial axes) are homotopic equivalent to each other.

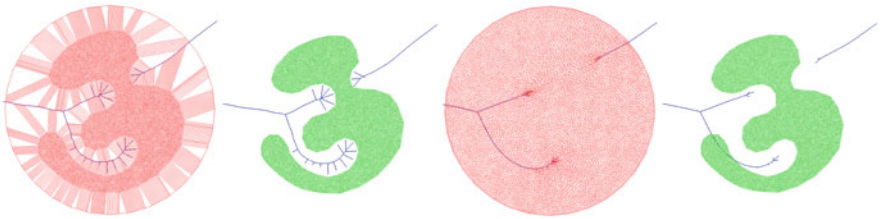


Fig. 10 Singularity sets of the optimal transportation maps between the unit disk and the sea-horse shape with different measures

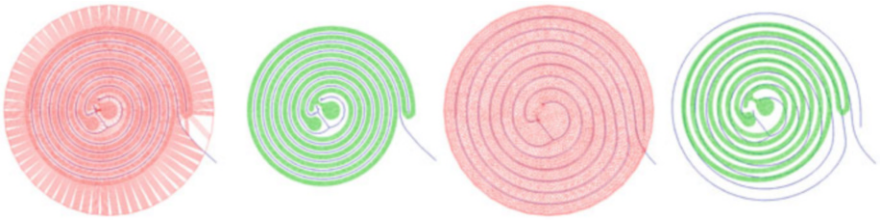


Fig. 11 Power medial axes are homotopic for a complicated planar domain

In deep learning applications, the training data set is encoded to the latent space, as shown in Fig. 3, the latent data distribution is the sum or Dirac measures $\nu = 1/n \sum_{i=1}^{n} \delta(y - \varphi(x_i))$, where n is the number of samples, x_i the training sample. The optimal transportation map can be computed using the proposed algorithm as shown in Fig. 4. The singularity on the support of the white noise distribution, and the non-differential points on the Brenier potential can be easily detected. The generated results can be seen in Fig. 2, the mode collapse and mode mixture have been eliminated by carefully handling the singularities of the optimal transportation map.

5 Conclusion

In this work, we focus on studying the singularity sets of Brenier optimal transportation maps. First, we generalize the concept of medial axis to power medial axis, which describes the singularity sets of semi-discrete optimal transportation maps. Second, we propose an algorithm based on a geometric variational principle using power diagrams to compute the power medial axis. Third, we prove that when the measures are changed continuously, the corresponding singularity sets of the optimal transportation maps are deformed homotopically.

In the future, we will study the sufficient and necessary conditions for the existence of singularity sets, both topologically and geometrically, and generalize the cost function from quadratic Euclidean distance to strictly convex functions.

Acknowledgments This research was supported by the National Natural Science Foundation of China under Grant Nos. 61720106005, 61772105, 61936002, and 61907005.

References

1. Alexandrov, A.D.: Convex Polyhedra. Translated from the 1950 Russian edition by N. S. Dairbekov, S. S. Kutateladze and A. B. Sossinsky. Springer Monographs in Mathematics. Springer, Berlin (2005)
2. An, D., Guo, Y., Lei, N., Luo, Z., Yau, S.T., Gu, X.: Ae-ot: a new generative model based on extended semi-discrete optimal transport. In: International Conference on Learning Representations (2020)
3. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. **16**(1), 78–96 (1987)
4. Brenier, Y.: Polar decomposition and increasing rearrangement of vector fields. C. R. Acad. Sci. Paris Sr. I Math. **305**(19), 805–808 (1987)
5. Brenier, Y.: Polar factorization and monotone rearrangement of vector-valued functions. Commun. Pure Appl. Math. **44**(4), 375–417 (1991)
6. Chen, S., Figalli, A.: Partial w2,p regularity for optimal transport maps. J. Funct. Anal. **272**, 4588–4605 (2017)
7. Figalli, A.: Regularity properties of optimal maps between nonconvex domains in the plane. Commun. Partial Differ. Equ. **35**(3), 465–479 (2010)
8. Gel'fand, I., Kapranov, M., Zelevinsky, A.: Discriminants, Resultants, and Multidimensional Determinants. Birkhäuser, Boston (1994)
9. Goodfellow, I.: NIPS 2016 tutorial: generative adversarial networks. arXiv:1701.00160 (2016)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS 2014), pp. 2672–2680 (2014)
11. Gu, X., Luo, F., Sun, J., Yau, S.T.: Variational principles for minkowski type problems, discrete optimal transport, and discrete monge-ampere equations. Asian J. Math. **20**(2), 383–398 (2016)
12. Joe, B.: Construction of three-dimensional delaunay triangulations using local transformations. Comput. Aided Geom. Des. **8**(2), 123–142 (1991)
13. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv:1312.6114 (2013)
14. Leal, F.S., De Loera, J.A., Rambau, J.: Triangulations, Structures for Algorithms and Applications. Springer, Berlin (2010)
15. Lei, N., Su, K., Cui, L., Yau, S.T., Gu, D.X.: A geometric view of optimal transportation and generative model. Comput. Aided Geom. Des. **68**, 1–2 (2019)
16. Lei, N., Chen, W., Luo, Z., Si, H., Gu, X.: Secondary power diagram, dual of secondary polytope. Comput. Math. Math. Phys. **59**(12), 1965–1981 (2019)
17. Lei, N., An, D., Guo, Y., Su, K., Liu, S., Luo, Z., Yau, S.T., Gu, X.: A geometric understanding of deep learning. Engineering **6**(3), 361–374 (2020)

Polygonal and Polyhedral Delaunay Meshing

Vladimir Garanzha and Liudmila Kudryavtseva

Abstract We consider construction of a polyhedral Delaunay partition as a limit of the sequence of radical partitions (power diagrams), while the dual Voronoi diagram is obtained as a limit of sequence of weighted Delaunay partitions. Using a lifting analogy, this problem is reduced to the construction of a pair of dual convex polyhedra, inscribed and superscribed around circular paraboloid, as a limit of the sequence of pairs of general dual convex polyhedra. The sequence of primal polyhedra should converge to the superscribed polyhedron, while the sequence of dual polyhedra converges to the inscribed polyhedron. Different rules can be used to build sequences of dual polyhedra. We are mostly interested in the case when the vertices of primal polyhedra can move or glue together, meaning that no new faces are allowed for dual polyhedra. These rules essentially define the transformation of the set of initial spheres defining power diagram into the set of Delaunay spheres using sphere movement, radius variation, and sphere elimination as admissible operations. Even though rigorous foundations (existence theorems) for this problem are still unavailable, we suggest a functional which measures deviation of the convex polyhedron from the one inscribed into paraboloid. This functional is the discrete Dirichlet functional for the power function which is a linear interpolant of the vertical distance of the dual vertices from paraboloid. The absolute minimizer of this functional is attained on the constant power field, meaning that the inscribed polyhedron can be obtained by means of a simple translation. This formulation of the Dirichlet functional for the dual surface is not quadratic since the unknowns are the vertices of the primal polyhedron, hence, the transformation of the set of spheres into Delaunay spheres is not unique. Numerical

V. Garanzha (✉)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia
e-mail: garan@ccas.ru
http://www.ccas.ru/gridgen/lab

L. Kudryavtseva
Moscow Institute of Physics and Technology, Moscow, Russia
e-mail: liukudr@yandex.ru

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_5

examples illustrate polygonal Delaunay meshing in planar domains. In this work we concentrate on the experimental confirmation of the viability of suggested approach putting aside mesh quality problems. The gradient of the above functional defines a manifold describing evolution of Delaunay spheres hence one can optimize Delaunay-Voronoi meshes using this manifold as a constraint.

1 Introduction

In his famous talk "Sur la sphere vide" at Toronto geometry congress in 1924 [5] and his paper [6], Boris Nikolaevich Delaunay (Delone) introduced a spatial partition of a prescribed discrete vertex set consisting of d-dimensional simplices with empty circumspheres, empty in the sense that they do not contain any vertices inside. His famous "Delaunay Lemma" states that if the empty circumsphere condition holds locally for any adjacent pair of simplices with a common $d-1$ dimensional face then all circumspheres are empty.

In 1937, Delaunay introduced the so-called "L-partition" [7] which can be obtained by moving, contracting, and inflating an empty sphere among the vertices of discrete set in $\mathbb{R}^d$ (Fig. 1a). In such a way, all empty spheres with a d-dimensional set of vertices on the surface are identified. The convex envelope of the set of vertices lying on such an L-sphere is a convex polyhedron called by Delaunay an L-polyhedron (Fig. 1b). The set of these polyhedra defines a normal partition of the space called the L-partition. The Delaunay lemma is naturally generalized for this polyhedral partition. These days, the L-partition is called Delaunay partition, while the spheres with d-dimensional set of vertices on their surfaces are called Delaunay spheres. The convex envelope of the spheres passing through a vertex defines a Voronoi polyhedron for this vertex. Note that both Delaunay and Voronoi partitions for a prescribed vertex set are unique. Duality relations between Delaunay and Voronoi partitions are simple and elegant, which is not the case for Delaunay triangulations.

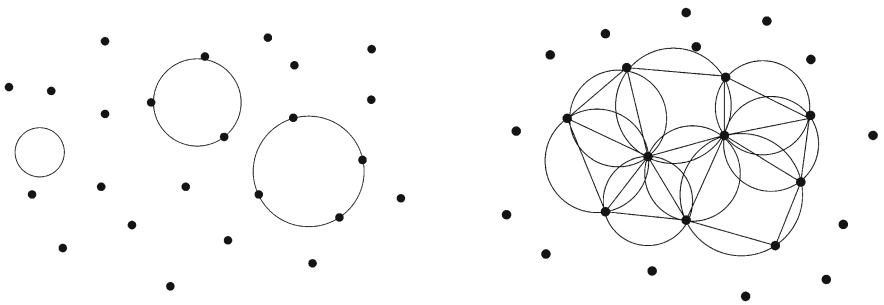


Fig. 1 (**a**) Empty sphere moving through the point set, (**b**) spheres with d-dimensional point sets

Each Delaunay polyhedron can be split into simplices, creating a simplicial partition which is generally called a Delaunay triangulation. Note that this construction excludes flat simplices (slivers). Strictly speaking, a sliver is a degenerate simplex with a $d - m$-dimensional vertex set, $m > 0$, meaning that it is just an incorrect triangulation of the $d - m$-dimensional face of the Delaunay partition. This problem is aggravated if approximate computations are used. In such a case, the difference between a correct but an almost flat simplex and an incorrect triangulation of a Delaunay polyhedron face has to be clearly identified. A natural identification is provided by the power diagram or a radical partition for the set of spheres. The radical partition goes back to René Descartes' work. It is the set of convex polyhedra constructed as intersections of half-spaces defined by $d - 1$-dimensional planes orthogonal to segments connecting centers for all pairs of spheres (see Sect. 2 for details). For a pair of intersecting spheres, the radical plane always passes through the common set. Hence, given a set of Delaunay spheres as an input, the radical partition would coincide with the Delaunay partition.

The equivalence of the radical and the Delaunay partitions provides a natural way to introduce a polyhedral approximation of a Delaunay partition for a disturbed point set. Let us add a small perturbation to the vertex positions and consider an arbitrary simplicial Delaunay partition for this vertex set, allowing for slivers. In this case, the numbers, centers, and radii of the Delaunay spheres evidently would change. However, since the volume of each Delaunay polyhedron was strictly positive initially, the centers and radii of the Delaunay spheres are stable with respect to perturbations. Hence, what we get is the cluster of spheres approximating each Delaunay sphere of the initial partition. A new sphere is the circumsphere of a certain new Delaunay simplex. We can find the best fit for the center of each cluster of spheres using averaging of circumcenters with volume of the corresponding Delaunay simplices as weights. In this case, a sliver would evidently provide a zero contribution to the averaged sphere parameters. The simplest way to compute the average radius for a cluster of spheres is to use least squares for distances between the new center and the disturbed vertices. A sliver may produce an isolated circumsphere which does not fit into the cluster of spheres. Such a sliver should be ignored when constructing the radical partition. The elimination criterion is simple: if the circumsphere is unstable, then the almost flat simplex is too badly conditioned to contribute to the set of spheres. As soon as a radical partition is computed, a stable triangulation of the resulting polyhedral partition can be found which may serve as an approximate Delaunay triangulation. Note that the radical partition for the disturbed point set may produce another point set as a set of partition vertices. It may produce a cluster of close vertices instead of each input vertex potentially creating needle-type simplices. These clusters also should also be glued together.

It is well known that an annoying feature of modern advanced tools for Delaunay triangulation such as TetGen [13] is the creation of artificial slivers. A simplest test set when the vertices of a cubic lattice are triangulated generally produces a huge number of flat tetrahedra. The logic of a Delaunay triangulation via a radical partition is more complicated but it allows to get rid of artificial slivers. Note, however, that the same notation "sliver" is sometimes used for a badly shaped

simplex which is not far from being flat but the computation of its circumcenter and circumradius is stable. Then it is a legitimate badly shaped Delaunay simplex and it should not be eliminated from the triangulation.

Delaunay triangulations have numerous applications in quite different fields [4]. One of the properties which is very important for numerical simulations is the maximum principle for the discrete Laplacian on Delaunay meshes [11].

They key ingredient of algorithms for constructing truly Voronoi computational meshes is the ability to control the location of Delaunay spheres in the key regions of the computational domain, in particular on and near boundaries [1, 12]. We propose a computational framework which, potentially, can serve as a tool of controlling placement of Delaunay spheres and provides a top-to-bottom Delaunay-Voronoi construction when the set of spheres generates the set of seeds for Delaunay meshing.

2 Power Diagram and Lifting Procedure

The idea of lifting goes back to the works of G.F. Voronoi [14] who has shown that a Delaunay triangulation in $\mathbb{R}^d$ is the projection of faces of a convex polyhedron $P \in R^{d+1}$ inscribed into a circular paraboloid Π. The convex body P^*, constructed as the intersection of the upper half-spaces for tangent planes to Π at the vertices of P, is called the Voronoi generatrice. Projection of its faces onto $\mathbb{R}^d$ defines a Voronoi diagram. A more general lifting concept [8, 9] is based on the construction of a pair P, P^* of convex polyhedra which satisfy the polarity relation [3] with respect to the paraboloid Π.

Consider the system of balls $\mathcal{B} = \{B_1, \ldots, B_n\}$ defined by centers and radii c_i, R_i, $c_i \in \mathbb{R}^d$, $R_i \geq 0$. According to the lifting operation, one can consider the lifted point system $\mathcal{E}_l = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^{d+1}$, where

$$p_i^T = \left(c_i^T \ \frac{1}{2}(|c_i|^2 - R_i^2) \right) = (c_i^T h_i).$$

Consider the lower convex envelope of $\mathcal{E}_l$ defined by the convex function $x_{d+1} = v(x_1, \ldots, x_d)$. The Legendre-Young-Fenchel dual [10] of v is denoted by v^*. To be mathematically precise, the function $v(x)$ is equal to $+\infty$ outside the convex envelope $\mathrm{conv}(c_1, \ldots, c_n)$ and its epigraph is a closed set. The dual function $v^*(x)$ is defined everywhere and its graph contains unbounded faces. Further, we will consider constrained problems where unbounded faces are excluded from the problem setting.

Projection of faces of the graph of v defines a weighted Delaunay triangulation $\mathcal{W}$ in $\mathbb{R}^d$ [8]. Vertices of the graph of v are pairs c_i, h_i and vertices of the weighted Delaunay triangulation are c_i. T_k denotes the k-th weighted Delaunay polyhedron.

Projection of faces of the graph of v^* defines a radical partition (power diagram) $\mathcal{R}$ for a system of balls in $\mathbb{R}^d$ [8], as shown in Fig. 2. Projection of the k-th vertex

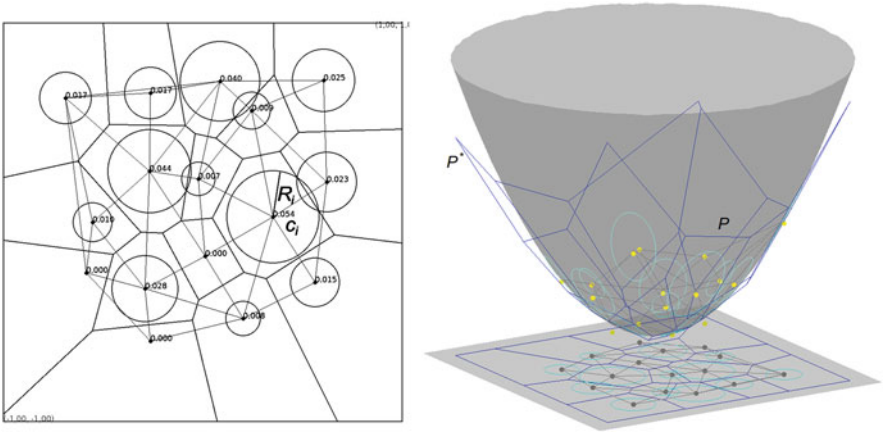


Fig. 2 A weighted Delaunay triangulation, the power diagram, and dual polyhedra from lifting. The figure is constructed with the help of `detri2` by Hang Si

of the graph of v^* onto $x_{d+1} = 0$ is denoted by v_k. This point is dual to $T_k \in \mathcal{W}$, while the vertex c_i is dual to the cell D_i of $\mathcal{R}$.

3 Ball Movement as a Transformation of Dual Polyhedra

Our objective is to move the balls $\mathcal{B}$ in such a way that all vertices of graph of v^* converge to the surface of paraboloid $x_{d+1} = \Pi(x) = \frac{1}{2}(x_1^2 + \cdots + x_d^2)$. It means that the projection of the graph of v^* will eventually converge to the Delaunay partition. Note that the number of vertices v_k may vary during the ball movement. Further, it is convenient to use notation x^T, x_{d+1} for an arbitrary point in $\mathbb{R}^{d+1}$, where $x^T = \{x_1, \ldots, x_d\}$.

For the set of balls $\mathcal{B}$, we build primal P and dual polyhedra P^* defined by the convex piecewise-linear functions $v(x)$ and $v^*(x)$, respectively. According to the polarity relation [3], the vertex c_i, h_i defines the dual plane of the face of the graph of $v^*(x)$,

$$c_i^T x = h_i + x_{d+1}.$$

The vertex v_k, z_k of the graph of $v^*(x)$ is the intersection of at least $d + 1$ such planes:

$$v_k^T (c_i - c_p) = h_i - h_p, \tag{1}$$

where $i \neq p$ are indices of all planes intersecting in the point v_k, z_k. Hence,

$$z_k = v_k^T c_i - h_i = v_k^T c_i - \frac{1}{2}c_i^2 + \frac{1}{2}R_i^2 = -\frac{1}{2}\Big(|c_i - v_k|^2 - R_i^2\Big) + \frac{1}{2}v_k^2.$$

In another words,

$$z_k - \Pi(v_k) = -\frac{1}{2}\tau_i(v_k). \tag{2}$$

Here,

$$\tau_i(y) = |c_i - y|^2 - R_i^2$$

is the power of the point y with respect to the ball B_i. Hence, the vertical distance of the vertex v_k, z_k from the paraboloid Π is fully defined by the value of power. Another interpretation of the equality (1) is that for a vertex v_k dual to the weighted Delaunay polyhedron T_k the equality

$$\tau_i(v_k) = \tau_p(v_k)$$

is satisfied for all vertices of T_k. It means that for the setting we use it is possible to omit indices and just use notation $\tau(v_k)$ for the value of power.

From (1) it follows that the gradient of $v^*(x)$ at the i-th face of its graph is equal to c_i. The dual statement is true as well: the gradient of $v(x)$ at the k-th face of its graph is equal to v_k. For any convex polyhedron T_k one can find d linear independent vectors $c_i - c_p$. In 2d, when T_k is a triangle, this set is simply $c_2 - c_1$, $c_3 - c_1$, as shown in Fig. 3a. When $\tau(v_k) > 0$, one can associate with v_k the sphere with radius $\sqrt{\tau(v_k)}$ which is called the orthosphere. In the section devoted

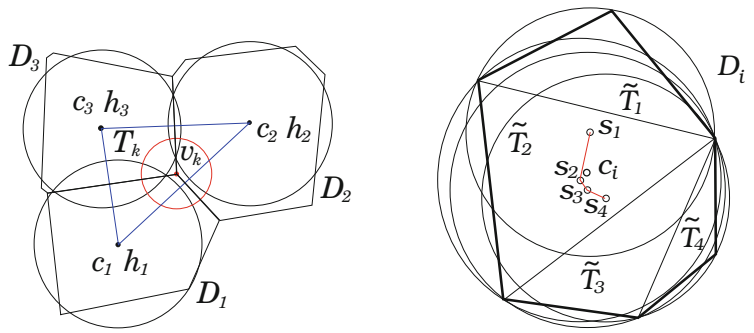


Fig. 3 (**a**) A weighted Delaunay triangle T_k, the dual vertex v_k and orthocircle, (**b**) a radical cell is split into 4 Delaunay triangles. Center c_i is approximated by a cloud of 4 Delaunay circumcenters

to numerical experiments we draw artificial "orthospheres" with radius defined by $\sqrt{|\tau(v_k)|}$. As shown below, these spheres visualize the deviation of radical partition from the Delaunay partition.

Denote by $\tilde{v}^*$ projected version of function v^* which is constructed as follows: consider the set of vertices $\{v_j\ z_j\}$ of the graph of v^* and project them on paraboloid Π just by setting

$$\tilde{z}_j = \frac{1}{2}|v_j|^2, \quad \tilde{v}_j = v_j.$$

As shown above, this projection changes the "vertical component" z_j by $\frac{1}{2}\tau(z_j)$.

Computing the lower convex envelope of the system of points $\tilde{v}_k, \tilde{z}_k, k = 1, n_v$, we get the graph of function $\tilde{v}^*$.

4 Dirichlet Functional for Power Function

In order to measure how far the current radical partition is from the Delaunay partition we consider the Dirichlet functional for the difference of v^* and $\tilde{v}^*$,

$$F(X) = \frac{1}{2}\int_\Omega |\nabla v^* - \nabla \tilde{v}^*|^2 \,\mathrm{d}x\,. \tag{3}$$

Here, X is the vector of unknowns, consisting of c_i, R_i, and Ω is the bounded definition domain of function $\tilde{v}^*$. From (2) it follows that

$$F(X) = \frac{1}{8}\int_\Omega |\nabla \tau(x)|^2 \,\mathrm{d}x\,.$$

We denote by $\tau(x)$ the piecewise linear function which coincides with $\tau(v_k)$ at v_k and is linear on each cell $\tilde{T}_j$ of the auxiliary Delaunay partition. $F(X)$ can be rewritten as

$$F(X) = \frac{1}{2}\sum_i \sum_{\tilde{T}_j \in D_i} |c_i - s_j|^2 \,\mathrm{vol}\,\tilde{T}_j, \tag{4}$$

where s_j is the circumcenter of the Delaunay simplex $\tilde{T}_j$. The above equality is the obvious consequence of the fact that

$$\nabla v^*\,|_{D_i} = c_i,\ \nabla \tilde{v}^*\,|_{\tilde{T}_j \in D_i} = s_j.$$

To clarify this formula, consider a Delaunay simplex $\tilde{T}_j$ with vertices $v_1, \ldots, v_{d+1}$. The gradient g_j of $\tilde{v}^*$ is defined by

$$(v_m - v_l)^T g_j = \frac{1}{2}(v_m^2 - v_l^2), \quad m, l \le d+1,$$

or

$$\frac{1}{2}|g_j - v_m|^2 = \frac{1}{2}|g_j - v_l|^2,$$

which is precisely the set of equations for the circumcenter s_j of $\tilde{T}_j$.

Equality $F(X) = 0$ implies that for each radical polyhedron D_i its dual vertex c_i coincides with all circumcenters of Delaunay triangulation of its set of vertices, meaning that D_i is Delaunay polyhedron.

Consider the following algorithm:

For $n = 0, 1, \ldots$

- Given the set of balls $\mathcal{B}^n$, compute the primal and dual functions v^n and v^{n*} using the lifting operation. These functions define a weighted Delaunay triangulation $\mathcal{W}^n$ and a radical partition $\mathcal{R}^n$, respectively.
- Compute the projected function $\tilde{v}^{n*}$. This function defines a Delaunay triangulation $\tilde{\mathcal{T}}^n$ of the set of vertices of radical partition $\mathcal{R}^n$.
- Do gradient search minimization step for Dirichlet functional $F(x)$, obtaining the set of balls $\mathcal{B}^{n+1}$.

  Repeat until convergence.

In this algorithm, the sequence of weighted Delaunay triangulations $\mathcal{W}^n$ converges to a Voronoi diagram $\mathcal{V}$ while the sequence of radical partitions $\mathcal{R}^n$ to a Delaunay partition $\mathcal{T}$. Note that the limit Delaunay triangulation $\tilde{\mathcal{T}}$ consists of Delaunay simplices, while $\mathcal{T}$ consists of polyhedral Delaunay cells for the same point set.

One can formulate a simplified version of the algorithm when cumbersome gradient computation is avoided. Let us try to find c_i using local minimization of the functional (4) considering it as a quadratic function of c_i,

$$c_i^{new} = \left(\sum_{\tilde{T}_j \in D_i} s_j \operatorname{vol} \tilde{T}_j \right) \Big/ \sum_{\tilde{T}_j \in D_i} \operatorname{vol} \tilde{T}_j,$$

while R_i is computed using simple least squares approximation

$$R_i^{new} = \frac{1}{\sqrt{M}} \left(\sum_{m=1}^{M} \left| c_i^{new} - v_m \right|^2 \right)^{\frac{1}{2}}, \tag{5}$$

where $v_1, \ldots, v_M$ is the set of vertices of D_i. Note that (5) is equivalent to

$$\sum_{m=1}^{M} \tau(v_m) = 0,$$

which, in turn, is the necessary minimum condition with respect to R_i for the local functional

$$\sum_{m=1}^{M} \tau^2(v_m).$$

New positions and radii are computed using a certain damping parameter $0 < \theta < 1$,

$$c_i^{n+1} = c_i^n(1-\theta) + c_i^{new}\theta \quad \text{and} \quad R_i^{n+1} = R_i^n(1-\theta) + R_i^{new}.$$

This heuristic algorithm is quite efficient for initial iterations. Eventually, it slows down and should be replaced by the gradient search technique in order to converge to the exact solution.

One can consider constrained problem where some of the balls are fully or partially fixed: ball centers are allowed to move along a prescribed manifold, radii constraints are added, etc.

Currently, there is no existence theorem for this problem. In principle, an overdetermined constrained problem can be defined such that the set of Delaunay spheres cannot be constructed without allowing introduction of new spheres.

5 Numerical Experiments

We made an attempt to generate polygonal Delaunay meshes for a relatively simple test setting. Consider a square with a circle inside. We cover boundary curves by the protecting circles. For the outer boundary, we fix the circle centers, thus, defining an approximation of the external boundary by Voronoi edges. For the inner boundary (the internal circle) we fix both the position of circle centers and the radii. Additional two lines of circles defining the initial layer of quadrilateral Delaunay cells near the internal boundary are added. The centers of these additional circles are fixed but the radii are allowed to change. We allow for a small radius change on the outer boundary as well. Since the outer boundary is covered by Delaunay cells, we essentially set zero boundary conditions for the power function $\tau(x)$. A lattice set of circles is created inside the domain and random disturbance is added to the centers and the radii. All centers which get inside the protecting circles are eliminated.

Figure 4 shows the general view of the initial radical partition and the final Delaunay partition. It can be observed that the constraints are satisfied by allowing

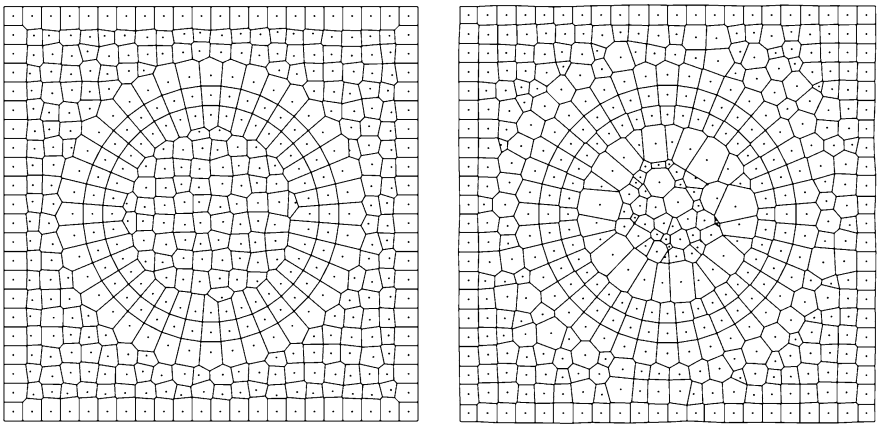


Fig. 4 Initial radical partition $\mathcal{R}_0$ and the final Delaunay partition $\mathcal{T}$

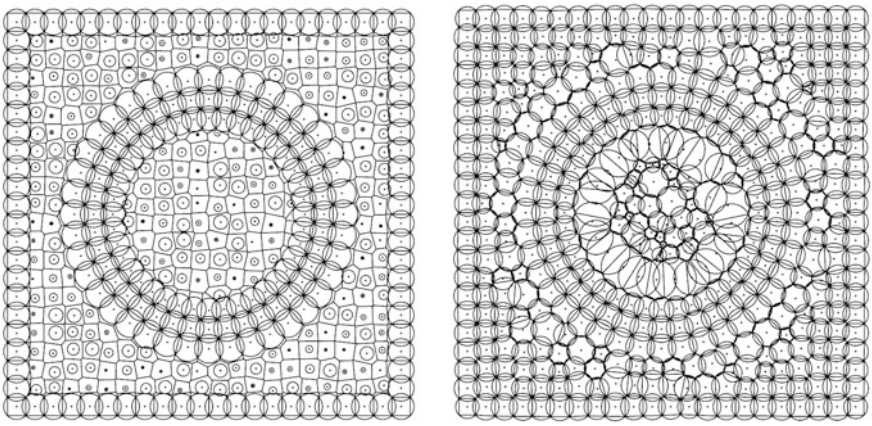


Fig. 5 Initial radical partition $\mathcal{R}_0$ with the initial set of circles and the final Delaunay partition $\mathcal{T}$ with circumcircles

larger cells near the fixed circles which, in turn, compresses some circles almost to zero.

Note that the nice regular layered pattern of Delaunay and Voronoi cells near the internal boundary is disturbed due to introduction of small Delaunay edges. In the current version of the algorithm, there is no mechanism to eliminate these small edges.

Figure 5 adds the set of circles B_i to Fig. 4.

Initial weighted Delaunay triangulation $\mathcal{W}_0$ and the final Voronoi mesh are shown in Fig. 6. Again, we observe quality problems in final Voronoi triangulation. Note that two layers of the Voronoi cells near the internal circle are in fact quadrilateral layers.

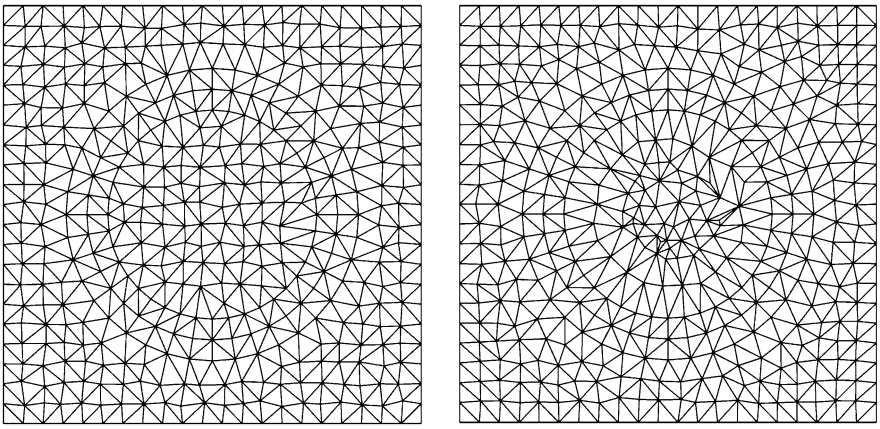


Fig. 6 Initial weighted Delaunay mesh $\mathcal{W}_0$ and the final Voronoi triangulation

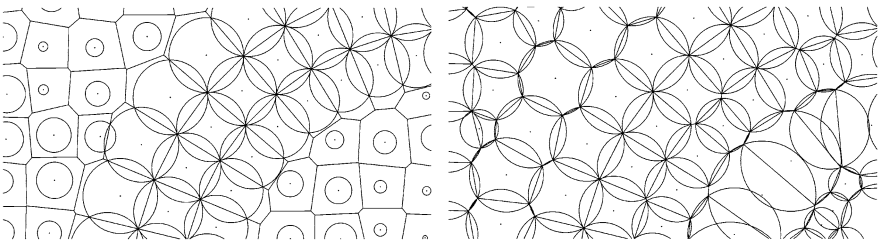


Fig. 7 Fragments of the initial radical and the final Delaunay partitions

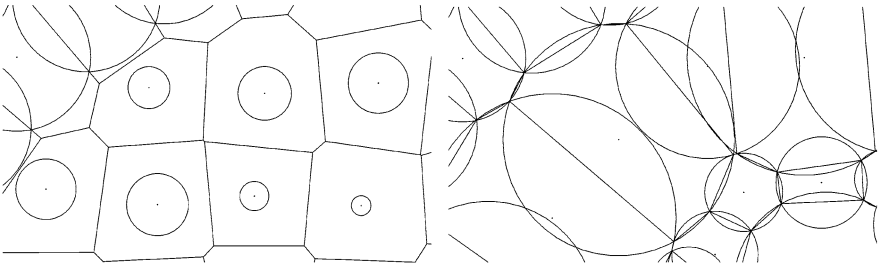


Fig. 8 Fragments of the initial radical and the final Delaunay partitions

Enlarged fragments in Figs. 7 and 8 allow to see clearly how the algorithm works.

Figure 9 show the evolution of the mesh fragment where "orthocircles" are added, meaning the circles centered at the dual vertices v_k with radii equal to $\sqrt{|\tau(v_k)|}$. When $\tau(v_k) < 0$, the red circles are no longer real orthocircles; they are just to evaluate the deviation of the radical cells from the Delaunay cells.

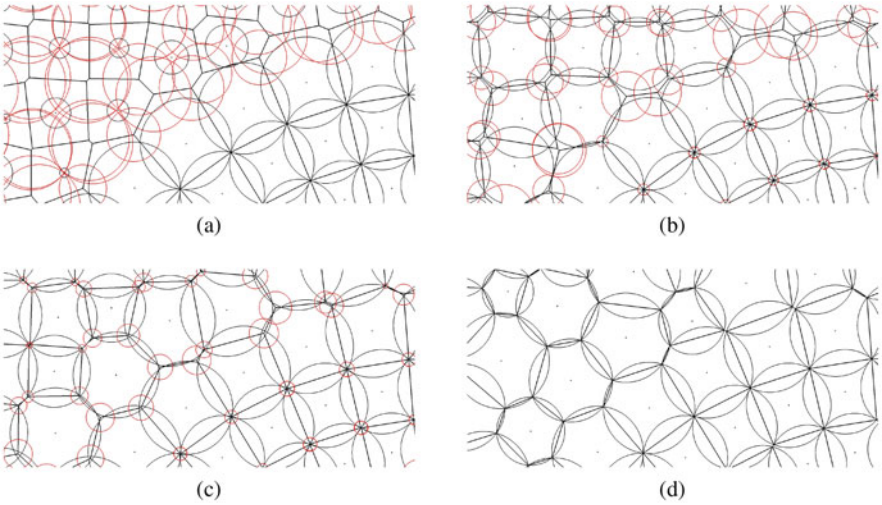


Fig. 9 Converging sequence of radical partitions (**a**)–(**d**) with square roots of absolute values of powers shown as red circles

6 Discussion

We have shown numerically in the 2d case that a radical partition can evolve into the polygonal Delaunay partition via evolution of the set of circles. The problem setting is multi-dimensional, hence, it is expected that the algorithm is applicable in the 3d case as well. While there is no existence theorem for the problem, we do not consider this as a crucial drawback. As soon as the addition of new balls becomes an admissible operation, the existence result become trivial, since the projected function $\tilde{v}^*$ in each iteration already provides a solution. However, the problem of the minimal addition of balls in order to construct a solution is an open one. Numerical experiments suggest that the ability to add new circles/balls locally also can be important in order to attain mesh quality in the presence of constraints.

Note that the concept of lifting allows to create nontrivial computational algorithms. In [2], the balls with unknown radii were assigned to the vertices of surface triangulation in order to solve the recovery problem when the weighted Delaunay tetrahedralization of the point set is constructed which matches prescribed boundary triangles. In [2], a similar existence problem is encountered. Evidently, it is possible to define a surface triangulation which cannot be matched by weighted Delaunay faces. In this case, the surface mesh should be refined: new vertices are added, which alleviates the existence problem.

In practice, the mesh quality functionals should be optimized by using manifold $\nabla F = 0$ as a constraint. Obvious quality requirements are related to elimination of small Delaunay and Voronoi edges and faces. This is subject of ongoing research.

In order to build good Delaunay-Voronoi meshes, one has to follow the sizing function, eliminate small Delaunay edges/faces, eliminate small balls, and eliminate small Voronoi edges/faces, trying to create a polyhedral Voronoi mesh.

Acknowledgments This work is supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2020-799.

References

1. Abdelkader, A., Bajaj, C., Ebeida, M., Mahmoud, A., Mitchell, S., Owens, J., Rushdi, A.: VoroCrust: Voronoi meshing without clipping. ACM Trans. Graph. **39**(3), 23 (2020)
2. Alexa, M.: Conforming weighted Delaunay triangulation. ACM Trans. Graph. **39** (6) (2020). https://doi.org/10.1145/3414685.3417776
3. Alexandrov, A.D.: Convex Polyhedra. Moscow-Leningrad (in Russian) (1950)
4. Aurenhammer, F., Klein, R., Lee, D.-T.: Voronoi Diagrams and Delaunay Triangulations. World Scientific, Singapore (2013)
5. Delone (Delaunay), B.N.: Sur la sphere vide. In: Proc. Internat. Congr. Math. (Toronto 1924), vol. 1, pp. 695–700. Univ. Toronto Press, Toronto (1928)
6. Delaunay, B.N.: Sur la sphere vide. Bull. Acad. Sci. URSS, VII. Ser. 1934 **6**, 793–800 (1934)
7. Delone, B.N.: The geometry of positive quadratic forms. Uspekhi. Mat. Nauk **3**, 16–62 (1937). **4**, 102–164 (1938)
8. Edelsbrunner, H.: Geometry and Topology for Mesh Generation. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge (2001)
9. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. Discrete Comput. Geom. **1**, 25–44 (1986)
10. Fenchel, W.: On conjugate convex functions. Canad. J. Math. **1**, 73–77 (1049)
11. Gärtner K., Kamenski L.: Why do we need Voronoi cells and Delaunay meshes? Essential properties of the Voronoi finite volume method. Comput. Math. Math. Phys. **59**(12), 1930–1944 (2019)
12. Garanzha, V.A., Kudryavtseva, L.N., Tsvetkova, V.O.: Hybrid Voronoi mesh generation: algorithms and unsolved problems. Comput. Math. Math. Phys. **59**(12), 1945–1964 (2019)
13. Si, H.: TetGen, a Delaunay-based tetrahedral mesh generator. ACM Trans. Math. Softw. **41**(2), 11:1–11:36 (2015)
14. Voronoi, G.F.: Nouvelles applications des paramètres continus a la théorie de formes quadratiques. J. Reine Angew. Math. **134**, 198–287 (1908)

On Decomposition of Embedded Prismatoids in $\mathbb{R}^3$ Without Additional Points

Hang Si

Abstract This paper considers embedded three-dimensional *prismatoids*. A subclass of this family is *twisted prisms*, which includes the family of non-triangulable Schönhardt polyhedra (Schönhardt, Math Ann 98:309–312, 1928; Rambau, On a generalization of Schönhardt's polyhedron. In: Goodman, J.E., Pach, J., Welzl, E. (eds.) Combinatorial and Computational Geometry, vol. 52, pp. 501–516. MSRI Publications, Chicago, 2005). We call a prismatoid *decomposable* if it can be cut into two smaller prismatoids (which have smaller volumes) without using additional points. Otherwise, it is *indecomposable*. The indecomposable property implies the non-triangulable property of a prismatoid but not vice versa.
In this paper, we prove two basic facts about the decomposability of embedded prismatoid in $\mathbb{R}^3$ with convex bases. Let P be such a prismatoid, call an edge *interior edge* of P if its both endpoints are vertices of P, and its interior lies inside P. Our first result is a condition to characterize indecomposable twisted prisms. It states that a twisted prism is indecomposable without additional points if and only if it allows no interior edge. Our second result shows that any embedded prismatoid in $\mathbb{R}^3$ with convex base polygons can be decomposed into the union of two sets (one of them may be empty): a set of tetrahedra and a set of indecomposable twisted prisms, such that all elements in these two sets have disjoint interiors.

1 Introduction

Decomposing a geometric object into simpler parts is one of the most fundamental problems in computational geometry.

In 2d, this problem is well solved. Given a polygonal region whose boundary is a planar straight-line graph $G = (V, E)$, there are many efficient algorithms to create a *constrained triangulation* whose vertex set is V and it contains all edges

H. Si (✉)
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
e-mail: si@wias-berlin.de

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_6

of E. Moreover, no additional vertices are needed. Lee and Lin [8] and Chew [4] independently proved that there exists a triangulation of G, called the *constrained Delaunay triangulation*, such that it is as close as to the Delaunay triangulation of V while preserveing all edges of E. Moreover, Chew showed that this triangulation can be constructed in optimal $O(n \log n)$ time [4].

The problem of triangulating polyhedra is complicated even we restrict ourselves to only consider simple polyhedra (without holes). It is known that not all simple polyhedra can be triangulated without adding new vertices, so-called *Steiner points*. The famous example of Schönhardt [12] (known as the Schönhardt polyhedron) shows that a twisted non-convex triangular prism cannot be triangulated without adding new vertices, see Fig. 1 (left). Other examples of non-triangulable polyhedra are presented, e.g., in [1–3, 6, 10, 13].

The existence of non-triangulable polyhedra is a significant difficulty in many 3d problems. Ruppert and Seidel [11] proved that the problem to determine whether a simple non-convex polyhedron can be triangulated without Steiner points is NP-complete. It is necessary to use additional points, so-called *Steiner points*, to triangulate polyhedra. Chazelle [3] constructed a family of polyhedra and proved that they require a large number of Steiner points to be triangulated, see Fig. 1 (right).

There are not many studies about the geometry and topology of such polyhedra. Rambau [10] first showed that any *non-convex twisted prisms* over an n-gon ($n \geq 3$) cannot be triangulated without Steiner points. Furthermore, he showed that the non-triangulability of such polyhedra does not depend on how much it is twisted. His result generalized the Schönhardt polyhedron into a family of polyhedra with such a property. We call polyhedra from this family *Rambau polyhedra*. The Schönhardt polyhedron is the simplest case of a Rambau polyhedron.

Geometrically, a Rambau polyhedron is a special prism such that its top and base polygons are (i) planar, (ii) congruent, and (iii) parallel to each other. In general, a twisted prism is not necessarily a Rambau polyhedron. Indeed, a slightly perturbed Rambau polyhedron whose base polygon has more than three vertices might become triangulable. On the other hand, if a prism (not necessarily a Rambau polyhedron) is

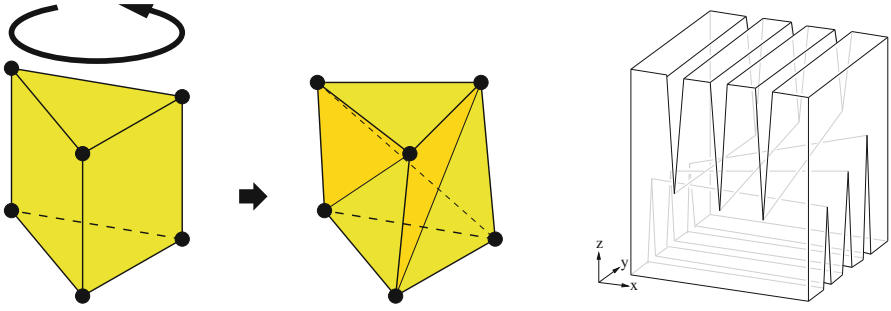


Fig. 1 Left: the Schönhardt polyhedron. Right: a Chazelle polyhedron

twisted sufficiently much, it won't be triangulable without Steiner points. The proof of this fact is rather simple. A basic fact (proved in Sect. 3) is that for a prism having a base polygon with more than five vertices, interior edges have to be decomposed. When a prism is twisted sufficiently large, it will reach a state an interior edge can't be inserted. Hence, it is non-triangulable. Note that a Rambau polyhedron might allow insertion of interior edges. Motivated by this phenomenon, we want to find the critical conditions between the existence and non-existence of a tetrahedralisation for this kind of polyhedra.

This paper considers three-dimensional *prismatoids* which can be embedded in $\mathbb{R}^3$. A subclass of this family are *twisted prisms*, which includes the family of non-triangulable Schönhardt polyhedra [10, 12].

We call a prismatoid *decomposable* if it can be cut into two smaller prismatoids (which have smaller volumes) without using additional points. Otherwise it is *indecomposable*. The indecomposable property implies the non-triangulable property of a prismatoid but not vice versa.

In Sect. 3, we prove two basic facts about the decomposability of embedded prismatoid in $\mathbb{R}^3$ with convex bases. Let P be such a prismatoid, call an edge *interior edge* of P if its both endpoints are vertices of P, and its interior lies inside P. Our first result is a condition to characterize indecomposable twisted prisms. It states that a twisted prism is indecomposable without additional points if and only if it allows no interior edge. Our second result shows that any embedded prismatoid in $\mathbb{R}^3$ with convex base polygons can be decomposed into the union of two sets (one of them may be empty): a set of tetrahedra and a set of indecomposable twisted prisms, such that all elements in these two sets have disjoint interiors.

2 Preliminaries

This section provides the necessary definitions of the family of prismatoids and twisted prismatoids studied in this paper.

2.1 Prisms, Antiprisms, and Prismatoids

In geometry, a *prism* is a solid that has two polygonal faces that are parallel and congruent [7, 14]. In other words, it is a polyhedron comprising an n-sided polygonal base (possibly non convex), a second base which is a translated copy (rigidly moved without rotation) of the first, and n other faces (necessarily all parallelograms) joining corresponding sides of the two bases. All cross-sections parallel to the bases are translations of the bases.

Antiprisms are similar to prisms except the bases are twisted relative to each other, and that the side faces are triangles, rather than quadrilaterals. Formally, an n-sided antiprism is a polyhedron composed of two parallel copies of an

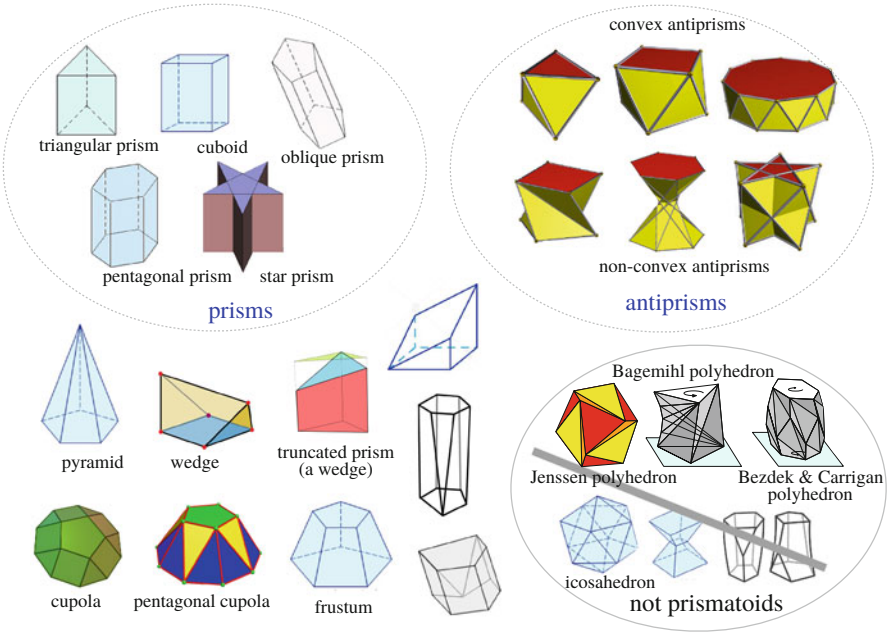


Fig. 2 The family of prismatoids. Right-bottom: some common figures which are not prismatoids[4]

n-sided polygonal base (possibly non-convex), connected by an alternating band of triangles.

Both prisms and antiprisms are subclasses of *prismatoids*. A prismatoid is a polyhedron whose vertices all lie in two parallel planes; its lateral faces can be trapezoids or triangles [7, 14]. The family of prismatoids includes many common geometric shapes, e.g., pyramids, wedges, prisms, antiprisms, and frusta (truncated pyramids). Figure 2 shows various examples as well as some common solids which are not prismatoids.

2.2 $S_{n,m}$-Prismatoids

This section defines a family of prismatoids considered in this paper. In brief, these prismatoids have convex bases which are connected by a band of triangular facets. Additionally, they can be embedded in $\mathbb{R}^3$ without self-intersections.

[4]Here and hereafter: cupola figure (from Wikipedia) used under CC BY-SA 3.0; antiprisms created with Robert Webb's Stella software, https://www.software3d.com/Stella.php; Jessen's polyhedron

Without loss of generality, we will place a prismatoid in such a way such that the two base facets are parallel to the horizontal plane $H_0 \coloneqq \{(x, y, 0) \mid x, y \in \mathbb{R}\}$. Moreover, one of its facets, called *bottom facet*, lies in H_0, and the other facet, called *top facet*, lies in the plane $H_h \coloneqq \{(x, y, h) \mid x, y, h \in \mathbb{R}, h > 0\}$.

Definition 1 Let n, m be two integers satisfying $n, m \geq 1$, $n + m \geq 4$. A $S_{n,m}$-*prismatoid* P (in short $S_{n,m}$) is a three-dimensional solid such that

(i) its top facet is a convex n-gon in H_h, its bottom facet is a convex m-gon in H_0;
(ii) the side facets of P between its bottom and top facets are all triangles; and
(iii) P is topologically a 3-ball embedded in $\mathbb{R}^3$.

Many prismatoids are $S_{n,m}$-prismatoids. For example, $S_{1,m}$ ($m \geq 3$) are pyramids. $S_{2,m}$ ($m \geq 2$) are wedges with triangular facets. In particular, both $S_{1,3}$ and $S_{2,2}$ are tetrahedra. Antiprisms which can be embedded in $\mathbb{R}^3$ with no self-intersected facets are $S_{n,n}$-prismatoids, see Fig. 3 (left). However, all prisms and many of other prismatoids are not $S_{n,m}$-prismatoids, see Fig. 3 (right). If a non $S_{n,m}$-prismatoid satisfies (iii), i.e., it can be embedded in $\mathbb{R}^3$ without self-intersection, it will become an $S_{n,m}$ by a slight perturbation in its vertex set.

Given an $S_{n,m}$-prismatoid P, there are $n + m$ triangles in its band. There is a bijection between the band of triangles and a binary string of $n + m$ 0/1 bits. In [5] a first construction of this transformation is given.

We first construct a *flattened band* D of triangles in the plane. It is done by cutting the band of P along one of its edges and then flatten it into the plane. There are $n + 1$ vertices and n edges on the top of D and $m + 1$ vertices and m edges on the bottom of D. These edges are in one-to-one correspondence to the boundary edges of the top and bottom facets of P. The two vertical boundary edges of D are identified as the same edge, which we cut open. The band's triangles are bijectively mapped into the triangles of D, i.e., the images of the triangles of the band triangulate D. We label each triangle in D as 0 if it has an edge on the top and a vertex on the bottom and as 1 if it has an edge on the bottom and a vertex on

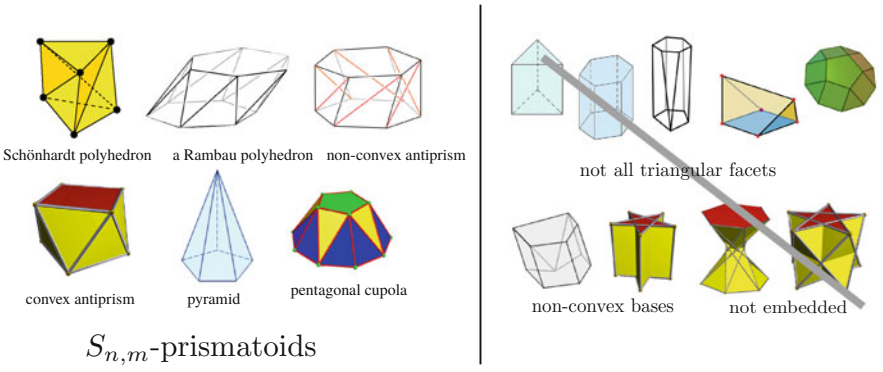


Fig. 3 Prismatoids on the left are $S_{n,m}$-prismatoids, while those on the right are not

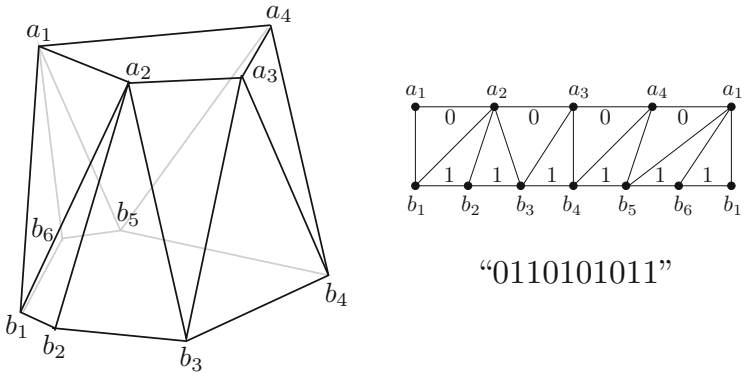


Fig. 4 An $S_{4,6}$-prismatoid is shown in the left and the binary string corresponds to its band of triangles is shown in the right

the top. Now, the set of triangles from left to the right corresponds to a string like 0100101. An example of such transformation is shown in Fig. 4.

With this transformation, the combinatorial structure of an $S_{n,m}$ can be characterized by a binary string. It does not, however, recognize the geometry of the prismatoid. For example, a convex and a non-convex $S_{n,m}$ may have the same binary string.

2.3 Twisted Prisms

We use the above transformation to define a special class of $S_{n,n}$-prismatoids. Recall that an antiprism can be obtained by twist a prism. There are two directions, clockwise or counterclockwise, in the plane. Depending on which direction it is twisted, we obtain two non-convex antiprisms which are similar but with combinatorially different boundary facets (Fig. 5). We call an $S_{n,n}$-prismatoid a *twisted prism* if the band of its triangles corresponds to a binary string which contains no two consecutive 0's or 1's, i.e., a string like 01010101 or 10101010.

The *degree* of a vertex of an $S_{n,m}$-prismatoid is the number of edges shared at this vertex. An equivalent definition of a twisted prism is: a twisted prism is an $S_{n,n}$-prismatoid whose vertices all have a degree of 4.

Note that a twisted prism might be convex or non-convex. We are interested in a special type of non-convex twisted prisms. Let P be a non-convex twisted prism whose base is an n-gon. We call P a *pure* non-convex twisted prism if there are exactly n non-convex edges in its boundary. For examples, the two non-convex prisms in Fig. 5 are pure. In particular, all Rambau's non-convex twisted prisms are pure.

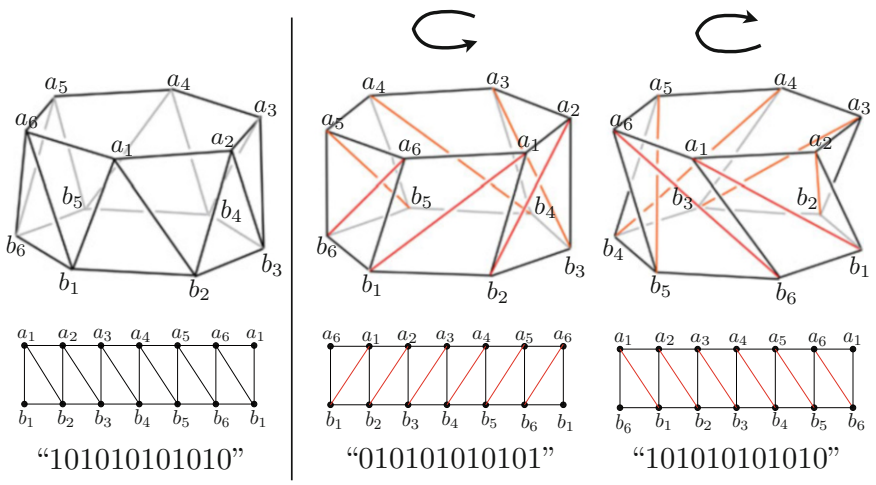


Fig. 5 Twisted prisms (top), bands (middle), and strings (bottom). Left: a convex hexagonal antiprism. Right: two non-convex antiprisms resulted by twisting the top facet of left counter-clockwise or clockwise, respectively

Note that our definition of twisted prisms is slightly more general than Rambau's definition [10] by not requiring the top and the bottom facets to be strictly congruent. They may be two different convex n-gons.

2.4 Decompositions of $S_{n,m}$-Prismatoids

A *triangulation* $\mathcal{T}$ of a prismatoid P is a geometric simplicial complex such that the union of all simplices of $\mathcal{T}$ is P, i.e., the underlying space of $\mathcal{T}$ is P. A triangulation of P may contain additional vertices, which are not vertices of P. These vertices are called *Steiner points* of P. In this paper, we are only interested in those triangulations of P, which have no Steiner points. We say a prismatoid is *triangulable* if it admits a triangulation without Steiner points. Otherwise, it is *non-triangulable*. It is well-known that some prismatoids are non-triangulable, e.g., the Schönhardt polyhedron and Rambau polyhedra.

For two $S_{n,m}$-prismatoids P_1 and P_2 let $\mathrm{Vert}(P_1)$ and $\mathrm{Vert}(P_2)$ be their vertex sets and $\mathrm{Vol}(P_1)$ and $\mathrm{Vol}(P_2)$ their volumes, respectively. We say that P_1 is *smaller than* P_2 if

$$\mathrm{Vert}(P_1) \subseteq \mathrm{Vert}(P_2), \tag{1}$$

$$\mathrm{Vol}(P_1) < \mathrm{Vol}(P_2). \tag{2}$$

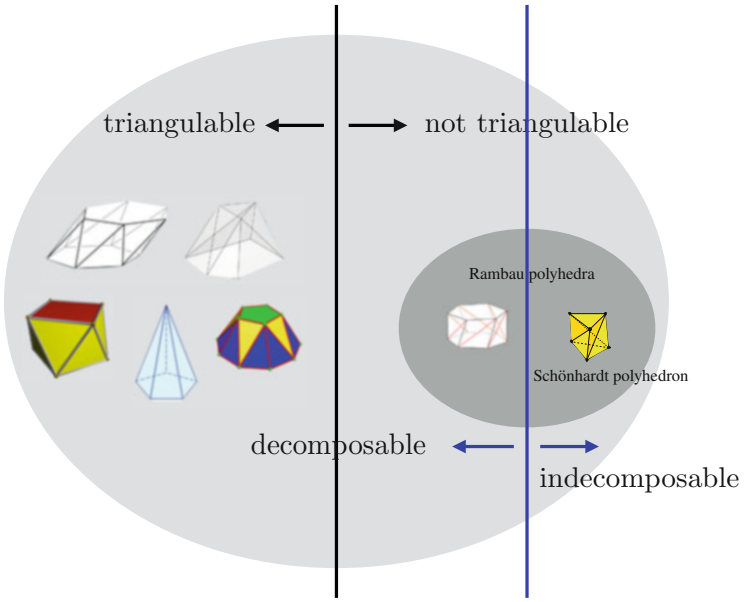


Fig. 6 The difference of being triangulable and being decomposable

Condition (1) means that P_1 and P_2 share the same vertex set of P_2, while (2) means that the volume of P_1 is strictly less than that of P_2. Note that (2) necessarily holds if the number of vertices of P_1 is strictly less than that of P_2, i.e., $\text{Vert}(P_1) \subset \text{Vert}(P_2)$ Note that if P_1 and P_2 have different vertices, than they are not comparable.

We say that an $S_{n,m}$-prismatoid is *decomposable* if it is either a single tetrahedron (i.e., an $S_{1,3}$ or $S_{2,2}$) or there exists a partition of it into two smaller $S_{n,m}$-prismatoids without using Steiner point such that the two prismatoids share no interior points, i.e., they only share at their common boundary facets. Otherwise, it is *indecomposable*.

The difference between being triangulable and being decomposable for a given prismatoid is that a triangulable prismatoid is also decomposable but not vice versa (Fig. 6). A non-triangulable prismatoid might still be decomposable. In contrast, an indecomposable prismatoid must be non-triangulable.

3 New Results on Decomposition of $S_{n,m}$-Prismatoids

We prove the following two theorems about the decomposability of $S_{n,m}$.

Theorem 1 *A twisted prism is indecomposable if and only if it does not contain interior edges.*

Theorem 2 *An $S_{n,m}$-prismatoid P can be decomposed with no Steiner points into the union of two sets $\mathcal{T}$ and $\mathcal{P}$, where $\mathcal{T}$ is a set of tetrahedra and $\mathcal{P}$ is a set of indecomposable twisted prisms. All elements in $\mathcal{T}$ and $\mathcal{P}$ have disjoint interiors.*

3.1 Outline of the Proof

An *ear* of a two-dimensional polygon is a vertex v of this polygon such that the line segment between the two neighbors of v lies entirely in the interior of the polygon. The two-ears-theorem [9] states that every simple polygon with more than three vertices has at least two ears, vertices that can be removed from the polygon without introducing any crossings. This theorem can be used to show that every two-dimensional simple polygon can be triangulated.

The analogue of an ear for a polyhedron is a degree 3 vertex, which has precisely three boundary edges of this polyhedron connecting to it (Fig. 7 left). The following lemma shows that if an $S_{n,m}$ contains a degree 3 vertex, it can be reduced to a smaller prismatoid, which does not contain that vertex. In other words, a degree 3 vertex can be removed from it (Fig. 7 right).

Lemma 1 *If an $S_{n,m}$-prismatoid with more than 5 vertices contains a degree 3 vertex, then it can be dissected into a tetrahedron and a smaller $S_{n,m}$-prismatoid without Steiner points.*

Proof Let P be an $S_{n,m}$. A triangular face is an *interior face* of P if its three vertices are vertices of P, and it is not a boundary facet of P. We prove this lemma in two steps:

1. A degree 3 vertex defines an interior face of P,
2. P can be separated by cutting along this interior face.

Without loss of generality, we assume that P contains a degree 3 vertex a_i in its top facet and the three boundary edges of P containing a_i are $\{a_i, a_{i-1}\}$, $\{a_i, a_{i+1}\}$, and $\{a_i, b_j\}$. Then the face $\{a_{i-1}, a_{i+1}, b_j\}$ is an interior face (Fig. 7 left).

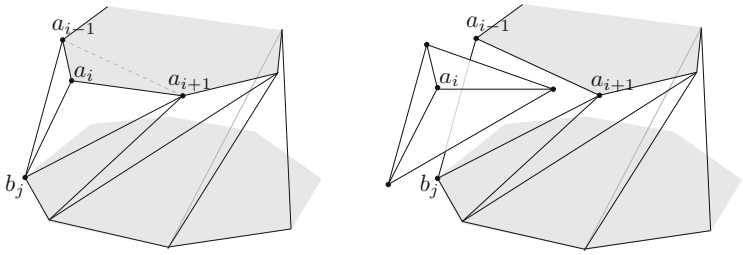


Fig. 7 Left: an $S_{n,m}$-prismatoid contains a degree 3 vertex a_i. Right: this prismatoid is separated by the tetrahedron $\{a_{i-1}, a_i, a_{i+1}, b_j\}$ and a $(n-1, m)$-prismatoid

Our proof of step 2 follows the following observation. Let our eye be at a_i, and we are looking into the interior of P. Our viewing volume is restricted by a cone with apex a_i and three boundary faces $f_1 := \{a_i, a_{i-1}, b_j\}$, $f_2 := \{a_i, a_{i+1}, b_j\}$, and $f_3 := \{a_i, a_{i-1}, a_{i+1}\}$. Note that f_1 and f_2 are original boundary facets of P. Since the edge $\{a_{i-1}, a_{i+1}\}$ lies in the interior of the top facet, the triangle f_3 is an ear in top facet. The property (iii) of P (Definition 1) requires that P contains no self-intersected boundary facets.

The above facts together imply that all interior points of the tetrahedron $\{a_{i-1}, a_i, a_{i+1}, b_j\}$ are interior points of P. Furthermore, the visibility to the four corners from any interior point of $\{a_{i-1}, a_i, a_{i+1}, b_j\}$ is not blocked by a boundary facet of P.

Therefore, the tetrahedron $\{a_{i-1}, a_i, a_{i+1}, b_j\}$ can be separated from P which results in an $S_{n-1,m}$-prismatoid P' with $\{a_{i-1}, a_{i+1}, b_j\}$ as its boundary facet. □

By the above lemma, as long as an $S_{n,m}$-prismatoid contains a degree 3 vertex, it is decomposable. Since a wedge cannot be a twisted prism, the above lemma immediately implies the following fact.

Corollary 1 *All $S_{2,m}$-prismatoids, $m \geq 2$, can be triangulated without Steiner points.*

If a twisted prism is not pure, it must be decomposable by removing a tetrahedron. Therefore we can quickly get the following corollary.

Corollary 2 *All $S_{n,m}$-prismatoids except pure non-convex twisted prisms are decomposable.*

Since not all pure non-convex twisted prisms are indecomposable, we still need to find the condition to characterize whether a pure non-convex twisted prism is indecomposable or not. Consider a twisted prism P. Call an edge $\{a_i, b_j\}$ an *interior edge* of P if both a_i and b_j are vertices of P and $\{a_i, b_j\}$ is not a boundary edge of P, and the interior of $\{a_i, b_j\}$ lies in P (see Fig. 8 left). The following

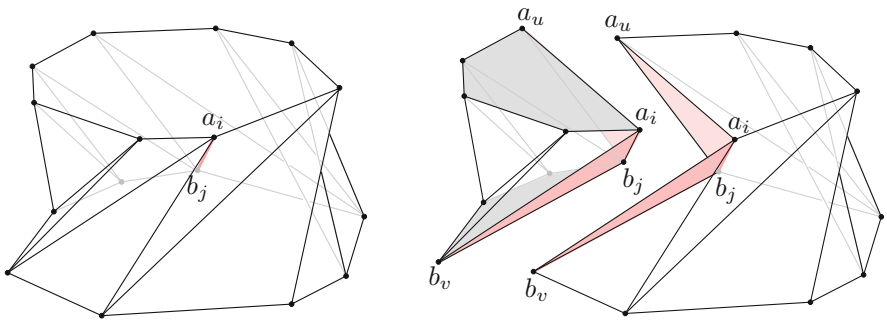


Fig. 8 A twisted prism (left) (an $S_{8,8}$) contains an interior edge $\{a_i, b_j\}$ (shown in pink). It is decomposed into two prismatoids, an $S_{4,5}$ and an $S_{5,6}$ (right) at this interior edge $\{a_i, b_j\}$ and two chosen interior faces $\{a_i, a_u, b_j\}$ and $\{a_i, b_v, b_j\}$ of the prism

Lemma 2 is crucial to reach this condition. It shows that P is decomposable as long as there is an interior edge of P (see Fig. 8 (right) for an example). The proof of this lemma is given in Sect. 3.2.

Lemma 2 *If a pure non-convex twisted prism contains an interior edge, then it can be decomposed into two smaller prismatoids without Steiner point.*

Theorem 1 is then proved by the reserve of Lemma 2. Theorem 2 can be proven by combining Lemmas 1 and 2 as follows.

Proof of of Theorem 2 Given an $S_{n,m}$, as long as it is not a twisted prism, it can be dissected into a set of tetrahedra and a twisted prism. If the twisted prism admits at least one interior edge, it can be dissected into two smaller simplicial prismatoids. The above process can be repeated until either no twisted prism remains or the remaining twisted prisms are indecomposable. □

3.2 Proof of Lemma 2

We will prove this lemma by showing that if an interior edge exists, then there must exist four interior faces which share at this edge, and the original twisted prism can be separated into two smaller prismatoids by these faces on their boundary.

Let P be a pure non-convex twisted prism whose base is a convex n-gon, $n > 3$. Without loss of generality, we assume that the top facet of P is twisted counterclockwise against its bottom facet (Fig. 9). Also, we label the vertices of

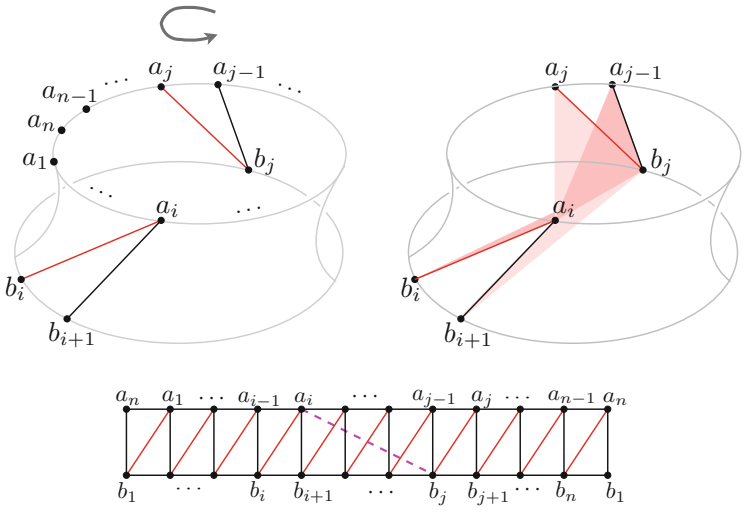


Fig. 9 The labelling of vertices of a pure non-convex twisted prism (top) and its transformed band in the plane (bottom). Red edges are locally non-convex edges of this prism. The edge $\{a_i, b_j\}$ is an interior edge. Left: the four edges in the band of the prism. Right: the four faces at the interior edge $\{a_i, b_j\}$

the top and bottom facets of P in such a way that the edge $\{a_i, b_i\}$ is locally a non-convex edge of P, $i = 1, \ldots, n$ (Fig. 9).

Let $\{a_i, b_j\}$ be an interior edge of P. The indices i and j are within the cyclic sequence $\{1, \ldots, n\}$. By our specific labelling of the vertices, i.e., $\{a_i, b_i\}$ refers to a non-convex edge, i and j must satisfy the following condition (additions and subtractions of indices are all modulo n):

$$j \notin \{i, i+1, i+2\} \quad (\text{equivalently}, i \notin \{j, j-1, j-2\}). \tag{3}$$

Consider the edges connecting at vertices a_i and b_j in the band:

$$\begin{aligned} a_i &: \{a_i, b_i\},\ \{a_i, b_{i+1}\}, \\ b_j &: \{b_j, a_{j-1}\},\ \{b_j, a_j\}. \end{aligned} \tag{4}$$

Each of these boundary edges forms a face that shares at the edge $\{a_i, b_j\}$. There are four faces, which can be sorted into two groups, F_{a_i} which are faces containing two vertices in the top facet, and F_{b_j} which are faces containing two vertices in the bottom facet (Fig. 9), i.e.,

$$\begin{aligned} F_{a_i} &: \{a_i, a_{j-1}, b_j\},\ \{a_i, a_j, b_j\}, \\ F_{b_j} &: \{a_i, b_i, b_j\},\ \{a_i, b_{i+1}, b_j\}. \end{aligned} \tag{5}$$

Given a pair of distinct indices $i, j \in \{1, \ldots, n\}$ satisfying (3), the four faces in (5) exist and they are distinct.

We prove that these four faces in (5) are interior faces of P. It is sufficient to show that all these faces satisfy the following two facts:

(I) they do not intersect any boundary facet of P in its interior, and
(II) all interior vertices of these faces are interior vertices of P.

Our proof of these two facts is based on observation, which suggests an intuitive geometric proof.

We project the prism P along the line containing the edge $a_i b_j$ onto a plane further than b_j, see Fig. 10. The edge vector defines the normal of this plane. Let a_i' be the projection of a_i in this plane and the same for other vertices of P. This projection of P (in the plane) has the following properties:

- P is projected into a (non-convex) region, denoted as R, in this plane, i.e, the shaded area in Fig. 10.
- The projection of the edge $\{a_i, b_j\}$ is coincident at one point in R. The four faces in Eq. (5) are projected into the four edges, shown in blue in Fig. 10.
- Each side facet of P is projected into either a triangle or a line segment in R, an example of the facet $\{a_{j-2}, b_{j-2}, b_{j-1}\}$ and its projection $\{a'_{j-2}, b'_{j-2}, b'_{j-1}\}$ is highlighted in Fig. 10. In particular, a facet is projected into a line segment if it is parallel to the edge $\{a_i, b_j\}$.

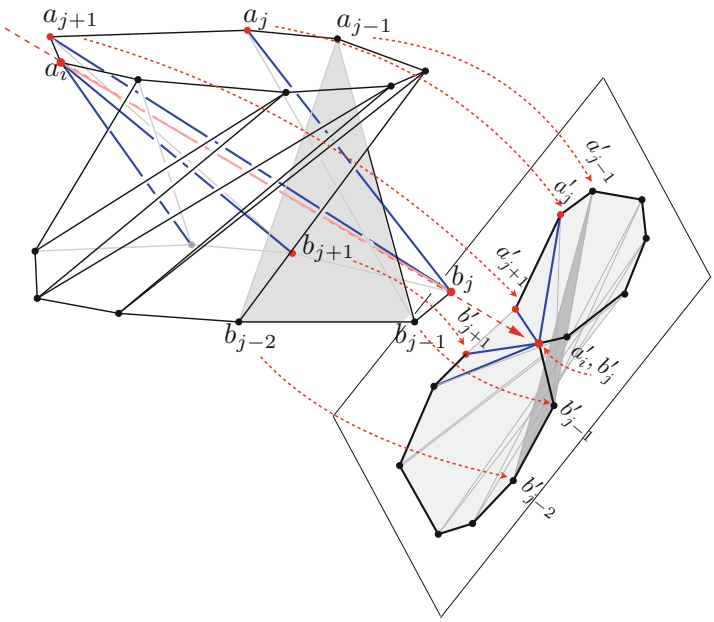


Fig. 10 Projecting the twisted prism P onto a plane orthogonal to the edge $\{a_i, b_j\}$

By this particular projection, we can verify that if the projection of a side facet of P in R does not cross the projection of the four faces, they do not intersect in their interior in $\mathbb{R}^3$. Observe the image of the projection of P (Fig. 10 right): the four edges $\{a'_j, b'_j\}$, $\{a'_{j-1}, b'_j\}$, $\{a'_i, b'_j\}$, and $\{a'_{i+1}, b'_j\}$ are not overlapping any other projected triangles in the plane. This phenomenon implies that the interior of these two faces does not intersect any other boundary facets of P in $\mathbb{R}^3$. From this observation, let us formally prove this fact.

The projections of the top and bottom polygons of P in the plane are two convex polygons that must intersect each other. In general, there are two intersection points, one of them must be the double-point, a'_i and b'_j, which is the projection of the edge $\{a_i, b_j\}$, see Fig. 11. These two polygons may intersect at only one point. In this case, this point must be the double-point; see Fig. 10. Based on this double-point, a'_i and b'_j, we can divide the projected vertices of P into four sets:

$$A_1 := \{a'_j, a'_{j+1}, \ldots, a'_{n-1}, a'_n, \ldots, a'_i\};$$
$$A_2 := \{a'_i, a'_{i+1}, \ldots, a'_{j-1}\};$$
$$B_1 := \{b'_j, b'_{j+1}, \ldots, b'_{n-1}, b'_n, \ldots, b'_i\};$$
$$B_2 := \{b'_{i+1}, b'_{i+2}, \ldots, b'_j\};$$

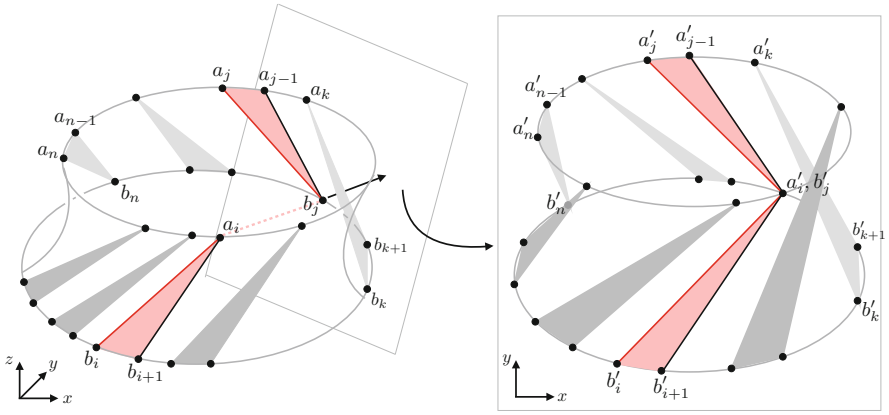


Fig. 11 Proof of Lemma 2. Projecting a pure non-convex twisted prism (left) onto a plane orthogonal to the edge $\{a_i, b_j\}$ (right)

We prove that any projected boundary facet of P does not cross the four edges $\{a'_i, b'_i\}$, $\{a'_i, b'_{i+1}\}$, $\{a'_j, b'_j\}$, and $\{a'_j, b'_{j-1}\}$. Let t be a boundary triangular facet of P, and t' be the projected triangle (may be an edge) in the plane. The vertices of t' must be one of the following cases:

1. The vertices of t' are in $A_1 \cup B_1$. All vertices are on the projected base polygons of P. Due to the convexity of the base polygons of P, t'does not cross any of the four edges.
2. The vertices of t' are in $A_1 \cup B_2$. This case is impossible. Without less of generality, let $t' := \{a'_u, b'_u, b'_{u-1}\}$, see Fig. 12. Since $a'_u \in A_1$, then $j \leq u' \leq i + n$. Since $b'_u \in B_2$, then $i \leq u' \leq j$, a contradiction.

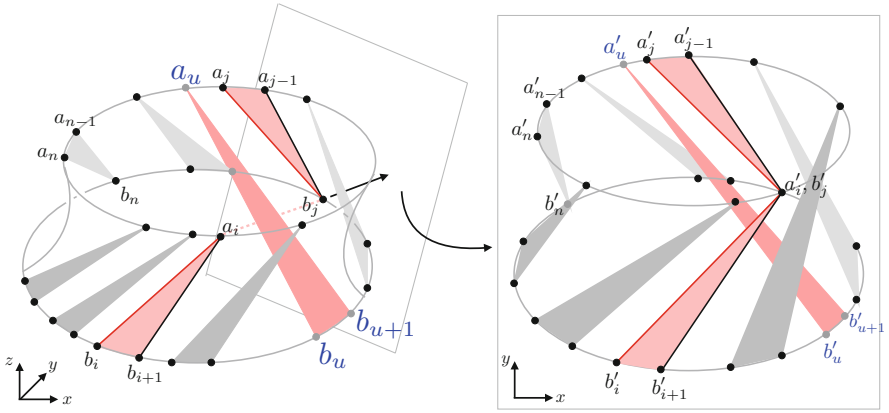


Fig. 12 Proof of Lemma 2. The facet $\{a_u, b_u, b_{u+1}\}$ cannot exist in the boundary of P

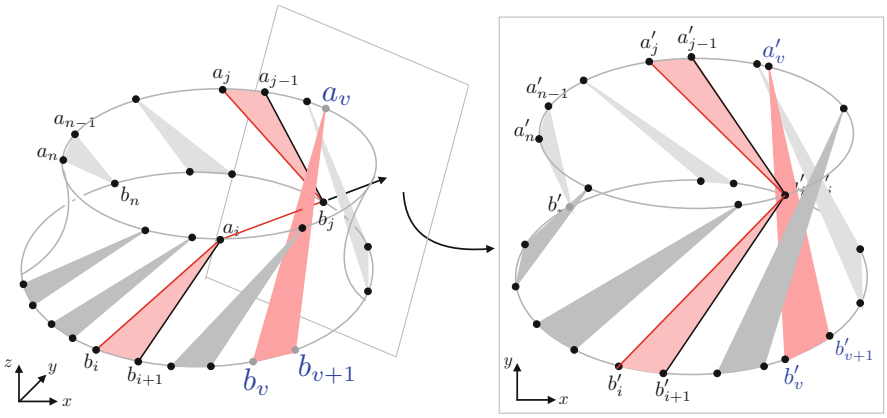


Fig. 13 Proof of Lemma 2. Since $\{a_i, b_j\}$ is an interior edge of P, the facet $\{a_v, b_v, b_{v+1}\}$ whose interior intersects $\{a_i, b_j\}$ cannot exist in the boundary of P

3. The vertices of t' are in $A_2 \cup B_1$. This case is impossible with the same reasoning as in case 2.
4. The vertices of t' are in $A_2 \cup B_2$. Assume t' is a triangle (not an edge). We show that t' does not intersect any of these four edges. Assume the contrary, t' does cross these edges. Without loss of generally, let $t' := \{a'_v, b'_v, b'_{v+1}\}$, where $a'_v \in A_2$ and $b'_v, b'_{v+1} \in B_2$, and assume t' intersect the edges $\{b'_{i+1}, a'_i\}$ and $\{a'_{j-1}, b'_j\}$, see Fig. 13. If this happens, there exist a boundary facet of P, which cuts the edge $\{a_i, b_j\}$, which implies that $\{a_i, b_j\}$ is not an interior edge of P, a contradiction.

Hence the projection of these two faces in the plane must be two edges that are not crossed by any other projected facets of P. By the property (iii) in Definition 1 of an $S_{n,m}$ (P is embedded in $\mathbb{R}^3$) it contains no self-intersected boundary faces. This shows that the two faces $\{a_i, a_j, b_j, \}$ and $\{a_i, a_{j-1}, b_j, \}$ do not intersect other side facets of P in their interiors. This proves (I).

Observe that the edges $\{a'_j, b'_j\}$ and $\{a'_{j-1}, b'_j\}$ lie inside the image of the projection of P. This shows that all interior points of the faces $\{a_i, a_j, b_j\}$ and $\{a_i, a_{j+1}, b_j\}$ must lie inside P. This proves (II).

Therefore the four faces in (5) must be interior faces of P.

Indeed, we also proved that the interiors of the two triangles $\{a'_i, b'_i, b'_{i+1}\}$ and $\{a'_j, a'_{j-1}, b'_j\}$ do not intersect any projected triangles of P. This shows that the interiors of the two tetrahedra

$$\{a_i, a_j, a_{j-1}, b_j\} \quad \text{and} \quad \{a_i, b_i, b_{i+1}, b_j\}$$

do not intersect any boundary facets of P. Hence they can be removed from P. This shows that P is decomposable. Our goal, however, is to separate P into two prismatoids with convex bases.

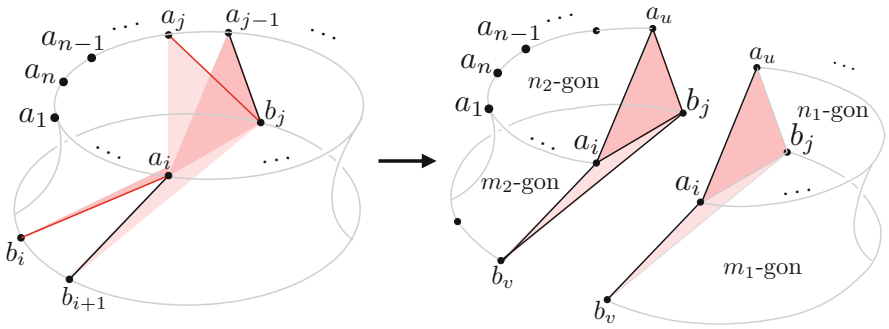


Fig. 14 Decomposition of a non-convex twisted prism $S_{n,n}$ (left) along an interior edge $\{a_i, b_j\}$ results in two prismatoids S_{n_1,m_1} and S_{n_2,m_2}, respectively (right)

It is easy to show that any combination of two faces, one from F_{a_i} and one from F_{b_j}, dissects the prism P into two smaller prismatoids. For example, choose

$$\{a_i, a_u, b_j\} \in F_{a_i} \quad \text{and} \quad \{a_i, b_v, b_j\} \in F_{b_j},$$

where $u = \{j-1, j\}$ and $v = \{i, i+1\}$. The edge $\{a_i, a_u\}$ will divide the top facet (a n-gon) into two convex polygons, an n_1-gon and an n_2-gon. The edge $\{b_v, b_j\}$ will divide the bottom facet (a n-gon) into another two convex polygons, an n_1-gon and an n_2-gon. Therefore, the original prism is cut into two smaller twisted prismatoids S_{n_1,m_1} and S_{n_2,m_2} (see Fig. 14 for an example). Without loss of generality, assume $i < u$ and $v < j$, then $n_1, n_2, m_1, m_2 \leq n$ can be calculated as

$$\begin{aligned} n_1 &\coloneqq u - i + 1, \quad & n_2 &\coloneqq n - n_1 + 2, \\ m_1 &\coloneqq j - v + 1, \quad & m_2 &\coloneqq n - m_1 + 2, \end{aligned} \tag{6}$$

where $u = \{j-1, j\}$ and $v = \{i, i+1\}$.

Since there are four possible combinations of faces from F_{a_i} and F_{b_j}, there are four possible dissections of this prism.

References

1. Bagemihl, F.: On indecomposable polyhedra. Am. Math. Month. **55**(7), 411–413 (1948)
2. Bezdek, A., Carrigan, B.: On nontriangulable polyhedra. Contrib. Algebra Geom. **57**(1), 51–66 (2016). http://dx.doi.org/10.1007/s13366-015-0248-4
3. Chazelle, B.: Convex partition of polyhedra: a lower bound and worst-case optimal algorithm. SIAM J. Comput. **13**(3), 488–507 (1984)
4. Chew, P.L.: Constrained Delaunay triangulation. Algorithmica **4**, 97–108 (1989)
5. Hurtado, F., Noy, M., Urrutia, J.: Flipping edges in triangulations. Discrete Comput. Geom. **22**(3), 333–346 (1999). http://dx.doi.org/10.1007/PL00009464

6. Jessen, B.: Orthogonal icosahedra. Nordisk Mat. Tidskr **15**, 90–96 (1967)
7. Kern, W.F., Bland, J.R.: Solid Mensuration. Wiley, London (1934)
8. Lee, D.T., Lin, A.K.: Generalized delaunay triangulations for planar graphs. Discrete Comput. Geom. **1**, 201–217 (1986)
9. Meisters, G.H.: Polygons have ears. Am. Math. Month. **82**(6), 648–651 (1975). http://www.jstor.org/stable/2319703
10. Rambau, J.: On a generalization of Schönhardt's polyhedron. In: Goodman, J.E., Pach, J., Welzl, E. (eds.) Combinatorial and Computational Geometry, vol. 52, pp. 501–516. MSRI Publications, Chicago. (2005)
11. Ruppert, J., Seidel, R.: On the difficulty of triangulating three-dimensional nonconvex polyhedra. Discrete Comput. Geom. **7**, 227–253 (1992)
12. Schönhardt, E.: Über die zerlegung von dreieckspolyedern in tetraeder. Math. Ann. **98**, 309–312 (1928)
13. Si, H., Goerigk, N.: Generalised Bagemihl polyhedra and a tight bound on the number of interior Steiner points. Comput. Aid. Des. **103**, 92–102 (2018). https://doi.org/10.1016/j.cad.2017.11.009. http://www.sciencedirect.com/science/article/pii/S0010448517302324
14. Wikipedia contributors: prismatoid (2019). https://en.wikipedia.org/wiki/Prismatoid. Accessed 20 May 2019

Out-of-core Constrained Delaunay Tetrahedralizations for Large Scenes

Ziya Erkoç, Aytek Aman, Uğur Güdükbay, and Hang Si

Abstract Tetrahedralization algorithms are used for many applications such as Ray Tracing and Finite Element Methods. For most of the applications, constrained tetrahedralization algorithms are chosen because they can preserve input triangles. The constrained tetrahedralization algorithms developed so far might suffer from a lack of memory. We propose an out-of-core near Delaunay constrained tetrahedralization algorithm using the divide-and-conquer paradigm to decrease memory usage. If the expected memory usage is below the user-defined memory limit, we tetrahedralize using TetGen. Otherwise, we subdivide the set of input points into two halves and recursively apply the same idea to the two halves. When compared with the TetGen, our algorithm tetrahedralizes the point clouds using less amount of memory but takes more time and generates tetrahedralizations that do not satisfy the Delaunay criterion at the boundaries of the merged regions. We quantify the error using the aspect-ratio metric. The difference between the tetrahedralizations that our approach produce and the Delaunay tetrahedralization are small and the results are acceptable for most applications.

1 Introduction

Tetrahedralization has many applications, ranging from finite element simulations to ray tracing accelerations. There are notable tetrahedralization algorithms in the literature. Yet, these algorithms are not appropriate for applications that require the use of very large meshes that do not fit into the memory. Besides, some of these applications require the faces of the input mesh to be preserved after

Z. Erkoç · A. Aman · U. Güdükbay (✉)
Department of Computer Engineering, Bilkent University, Ankara, Turkey
e-mail: ziya.erkoc@bilkent.edu.tr; aytek.aman@cs.bilkent.edu.tr; gudukbay@cs.bilkent.edu.tr

H. Si
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
e-mail: si@wias-berlin.de

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_7

the tetrahedralization is complete. This kind of tetrahedralization algorithm is called *constrained tetrahedralization*. Constrained tetrahedralization algorithms guarantee that the input mesh is contained in the surface of the output tetrahedral mesh. This property is especially important for tetrahedralization-based ray tracing accelerators, not to disturb the surface geometry [4]. Although these algorithms are quite powerful and can complete the tetrahedralization process in a reasonable amount of time, their usage is limited by the available memory. These algorithms might fail when the tetrahedralization of an object requires a large amount of memory. For instance, a bridge model consisting of tens-of-millions of vertices can be analyzed using the Finite Element Method. Besides, a ray-traced scene may contain up to a few hundred million faces [8]. Those examples might require an excessive amount of memory.

We propose an out-of-core divide-and-conquer constrained tetrahedralization algorithm that will take the memory-constraint specified by the user into account and will not exceed it. Our algorithm divides the mesh into two pieces recursively as long as the expected memory usage is above the given constraint. When the given mesh is small enough to satisfy memory requirement it is tetrahedralized. Since our algorithm does not guarantee satisfying Delaunay property for every tetrahedron, it is a near Delaunay tetrahedralization algorithm. Yet, this approximation may be reasonable for applications such as tetrahedralizations as acceleration structures for Ray Tracing [4]. In their paper, Lagae and Dutré show that if tetrahedralization algorithms that are used in ray tracing relax the Delaunay criterion, the construction time decreases but the rendering time increases. Hence, our algorithm offers a trade-off between these two timings.

2 Related Works

There are many triangulation algorithms in the literature. However, these algorithms cannot tetrahedralize large models that do not fit into the memory in a constrained fashion.

Smolik and Skala put forth a divide-and-conquer tetrahedralization algorithm that works both in CPU and GPU [6]. They also develop an out-of-core version of their algorithm and observe a decrease in memory usage, thereby being able to tetrahedralize large objects with the same available memory. Yet, their algorithm is not a constrained tetrahedralization. They divide the input point cloud into a 3D grid and simultaneously tetrahedralize each grid cell and finally merge the cells. Our approach is different because we, before all, have developed a constrained tetrahedralization algorithm. Yet, dividing the object into grids may not be possible for constrained tetrahedralization because triangles should not extend to more than one grid cell, which is not possible with their algorithm. Hence, we divide the object into two at any time instead of dividing it into many small pieces.

Cignoni et al. propose a divide-and-conquer algorithm to triangulate meshes of any dimension [3]. However, they do not describe an out-of-core extension of

their algorithms. Besides, their algorithm is not a constrained tetrahedralization algorithm. Our divide-and-conquer algorithm differs from DeWall in the non-recursive part. Their algorithm applies a merge step before recurring. In this early-merge step, their algorithm uses a dividing plane, and by selecting the closest vertices at either side of this plane, it creates an initial tetrahedralization. Specifically, they choose these vertices so that the generated tetrahedra has the smallest circumsphere radius to satisfy the Delaunay criterion. Therefore, at this step, all the tetrahedra generated intersects that virtual plane. Then, it applies the same recursively for the other two sides. Our method is different from theirs in the sense that we do not select an area to be initially tetrahedralized; we just cut the mesh into two and tetrahedralize the parts. DeWall performs three tetrahedralization operations at each recursive step, one for around the plane, one for the left, and one for the right side. However, we only tetrahedralize left and right without allocation a middle region to be tetrahedralized. This approach comes with a cost for us because we do not have a middle-region like DeWall. Hence, we cannot guarantee that the Delaunay criterion is satisfied for the tetrahedra around the cutting plane.

Blelloch et al. [1] present a parallel Delaunay triangulation algorithm. Their algorithm uses the divide-and-conquer paradigm like ours. They utilize parallelism at the pre-recursive step to reduce the overall run-time cost. They experimented on various point distributions and observed significant improvements. Our algorithm is different because ours is a constrained triangulation algorithm that takes into account not only the input points but also the input triangles. Besides, while we aim to reduce the memory usage by compromising run-time, Blelloch et al. want to improve the run-time performance. Since they introduce parallelism, they need to include new data structures, which require an extra set of data stored in the main memory. Therefore, developing a parallel algorithm might defy our purpose of creating a memory-efficient algorithm.

Si developed a constrained Delaunay tetrahedralization algorithm [5]. This algorithm is both fast and robust. However, it requires a significant amount of memory for the tetrahedralization process because it is not an out-of-core algorithm. We compare our algorithm to TetGen because our algorithm depends, at its core, on it. We aim to create a memory-efficient version of TetGen to tetrahedralize large meshes that do not fit into the memory by applying an out-of-core approach on top of it.

3 Algorithm

3.1 Overview

Our algorithm is an out-of-core divide-and-conquer algorithm for constrained tetrahedralization. It, at its core, makes use of the TetGen software [5]. Our algorithm divides the input mesh into two as long as the memory is not enough

Algorithm 1 Our algorithm

```
1: procedure TETRA(vertices, faces)
2:     if CALCULATE_EXPECTED_MEMORY(vertex_count) ≤ memory_limit then
3:         TETGEN(vertices, faces)
4:     else
5:         left_vertices, left_triangles, right_vertices, right_triangles
6:             = CLIP(vertices, faces)
7:         left_mesh_file = TETRA(left_vertices, left_triangles)
8:         right_mesh_file = TETRA(right_vertices, right_triangles)
9:         output_mesh_file = MERGE(left_mesh_file, right_mesh_file)
10:        return output_mesh_file
11:    end if
12: end procedure
```

for it. We calculate the expected memory usage using linear regression. If the mesh fits into the memory, then we use TetGen with the mesh as input to generate the tetrahedral mesh. Otherwise, we divide the mesh into two pieces by a plane passing through the mean of the most variant axis. We then recursively apply the same procedure to the two parts. When the tetrahedralizations of the parts are complete, we merge these tetrahedral meshes into one tetrahedral mesh as the last step. Algorithm 1 provides thee pseudo-code of the algorithm.

Further, our algorithm can generate a bounding box for the input object so that the space around the object can be tetrahedralized, which is especially convenient for Ray Tracing accelerators.

3.2 Expected Memory Calculation

We observe the memory consumption of TetGen with several models and generated a linear regression model to predict the memory consumption of an input object. The first two columns of Table 1 show the vertex count of each object and the real

Table 1 Memory requirement observations for TetGen in Megabytes (MB). We provide the actual memory requirements and the estimated values for various models of different vertex counts using our linear regression model

Vertex count	Actual memory requirement	Expected memory requirement
1440	7.03	9.70
2880	13.97	16.28
34,560	167.67	161.04
112,220	514.80	515.91
172,971	792.97	793.51

memory usage when that object is tetrahedralized. We set up a linear regression model using this data and we ended up with the following equation:

$$y = 4.57 \times 10^{-3} X + 3.12,$$

where y is the expected memory requirement in Megabytes (MB), and X is the number of vertices of the input mesh. In the table, the last column corresponds to the expected memory requirement calculated using the above equation.

3.3 Subdivision Stage

To decrease the problem size, the input mesh must be divide into two pieces at each recursion level. We are dividing the input using the plane that passes through the mean of the most variant axis. Dividing the object requires a significant effort because after the division the cut surfaces of each object must match so that the resulting tetrahedral mesh of each side matches. Matching triangles will guarantee a match in the resulting tetrahedral mesh because we are applying constrained tetrahedralization.

To this end, we use the clip function of CGAL [7]. It takes input mesh and a plane as input, and slices the mesh using the plane and returns the positive side. It also allows us to triangulate the open-surface. We use it the following way: we first clip the object and get the surface triangulation. Then, for the second half, we again clip the object but not triangulate the surface. Instead, we just copy the triangulation of the first part and paste it to the second part. In that way, we guarantee that the triangulation at both sides will match. However, copy-pasting may lead to duplicate vertices and open borders. Therefore, we propose a repairing algorithm to eliminate these defects.

The way we clip the object prevents the tetrahedra near the clipping plane to breach the Delaunay criterion. This is because after each part is tetrahedralized we cannot guarantee that the circumsphere of the tetrahedra around the plane will contain points from the other side. Since while one half is tetrahedralized the other half is not considered, the Delaunay criterion might be violated. Yet, the magnitude of the violation depends on the number of tetrahedra around the plane and the number of times the object is divided. While choosing the plane, we make sure that the most variant axis is chosen, to also decrease the number of tetrahedra around the plane.

3.4 Repairing Stage

We introduce steps that we apply to eliminate defects in the meshes produced during the subdivision stage. Two defects that may arise during the clipping stage are

overlapping vertices and *overlapping edges*. Specifically, both problems occur because of copying the triangulation of the left side and pasting it to the right side. Faces cannot overlap because we copy the triangles generated for the left side to the open-boundary of the right side.

Overlapping vertices occur because when we copy the triangulation, the vertices of the left surface triangulation might coincide with the border vertices of the right mesh. We can repair overlapping vertices by iterating through all of the triangles around the dividing plane and checking if any of its vertices have any duplicates. If this is the case, we keep only one of the vertex and ignore the other one.

Edges may overlap after copying the triangulation from one side to the other. When a vertex from the left side does not have the corresponding vertex at the right side; then, this vertex would coincide with one of the edges of the right side. The clip function does not insert the same vertices to both sides, which causes this problem.

Figure 1a shows an example of overlapping edges. There are three triangles: ABC, BED, and DEC. Here ABC is an existing triangle of the mesh, but during the clipping operation, we can add the other two faces so that two halves match as described above. The problem here is the edge BC overlaps both the edges BD and DC. To fix this, we form triangles ABD and ADC and remove ABC (see Fig. 1b). We repair only the right side because we copy the triangles of the left side over the right side. Hence, only the topology of the right side is disturbed.

We repair overlapping edges as follows. We iterate over all edges in the right mesh that are close to the dividing plane. For each face in this region, we iterate over all of the vertices of the mesh on the right. If the edge contains a vertex between its terminal points, it means that this vertex is leading to an overlapping edge. We make use of the point-line segment distance to find the overlapping between a vertex and edge. Afterward, the triangle containing this edge is fixed, as shown in Fig. 1.

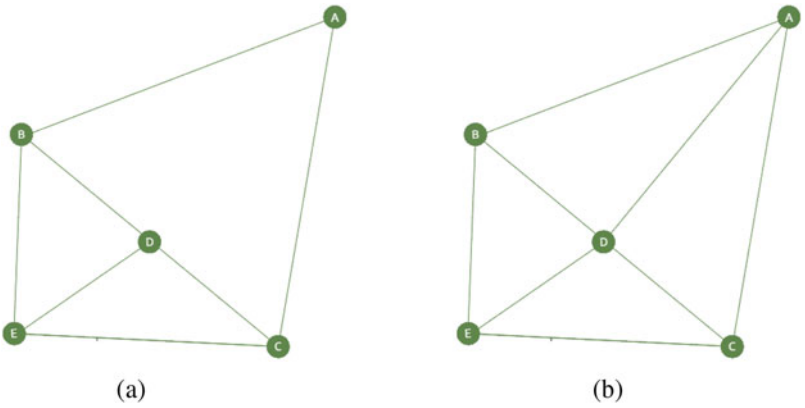


Fig. 1 Handling overlapping edges. (**a**) Overlapping edges. (**b**) After repairing overlapping edges

Algorithm 2 Merge procedure

```
procedure MERGE(left_mesh_file, right_mesh_file)
    # Concatenate vertices and tets
    CONCATENATE_VERTICES(left_mesh_file, right_mesh_file, output_mesh_file)
    CONCATENATE_TETS(left_mesh_file, right_mesh_file,output_mesh_file)
    # Find missing neighbors
    left_centroids = GET_CENTROIDS(left_mesh_file)
    left_centroids_grid = GENERATE_GRID(left_centroids)
    right_centroids = GET_CENTROIDS(right_mesh_file)
    for each right_centroid ∈ right_centroids do
        if right_centroid ∈ left_centroids_grid then
            ADD_NEIGHBOUR(right_tet, left_tet, output_mesh_file)
        end if
    end for
    return output_mesh_file
end procedure
```

3.5 Merging Stage

The merge step is another non-trivial step for our algorithm because it is the place where we finally produce the resulting tetrahedral mesh. In the merge step, we first merge two tetrahedral meshes into one mesh as depicted in Algorithm 2. That step simply involves concatenating two text files. Secondly, we also extract neighbor relations between tetrahedra. In the end, we generate a final *output_mesh_file* consisting of vertex locations, tetrahedra, and neighborhood information between tetrahedra. We put a non-trivial effort to find neighborhood information across the two pieces of the object after subdivision. After we cut the object into two pieces and tetrahedralize each piece, the neighbor relations around the cut faces are missing and we find those as well.

3.5.1 Spatial Hashing

We use three-dimensional *Spatial Hashing*. Specifically, the dimensions of the hash grid we have used are $50 \times 50 \times 50$ corresponding to 125,000 cells. We begin by putting the centroid of faces of the left piece that coincide with the cut plane, into the 3D grid. Then, we iterate over the faces of the right piece and find if their centroids match any of the centroids of the left piece by finding the corresponding cell and iterating through the centroids inside of it. If there is a match, it means that two tetrahedra share a common triangle and they must be neighbors. Then, we save this new neighborhood information.

3.5.2 Merging Time Complexity

Let V_L, V_R be the number of vertices, F_L, F_R be the number of faces and T_L, T_R be the number of tetrahedra of left and right pieces. Merging files will take $\theta(V_L + V_R + T_L + T_R)$ time. The time complexity of finding missing neighbour relations through *Spatial Hashing* will take $\theta(F_L + F_R)$ time on average. Specifically, we, first, iterate through all faces of left piece to putting the centroids in the grid taking $\theta(F_L)$ time. Then, we iterate through all faces in the right piece, $\theta(F_R)$ time, and for each face we search the centroid inside the grid which takes $\theta(1)$ time thanks to hash structure. Hence, overall, it takes $\theta(F_R) * \theta(1) = \theta(F_R)$ time. As a result, overall time complexity of, merge step is $\theta(V_L + V_R + T_L + T_R + F_L + F_R)$. The time complexity function can be further simplified using the fact that $V <= 3 * F$ and $V <= 4 * T$ to end up with $\theta(V_L + V_R)$.

4 Experimental Results

4.1 Runtime and Memory Results

In this section, we present the results of constrained tetrahedralization of several objects using both our algorithm and TetGen to provide the statistics of memory consumption and execution times. Our algorithm performs differently based on the intersection of the cutting plane with any object in the input mesh. If the plane touches an object, then our algorithm will run a repairing procedure. Otherwise, it skips the repairing procedure. We conduct the experiments on a high-end computer with an Intel Xeon E5-2620 2.10 GHz processor and 64 GB of RAM. In the experiments, for simplicity, we divide the input mesh into two parts but not further. For each experiment, we monitor the Windows Task Manager and record the peak memory usage. Hence, we report the physical memory usage.

Table 2 shows the statistics of computation time and memory consumption where each row contains the results of the experiment specified in the first column. We provide details about each experiment in the sequel.

Experiment 1: In that scene, because the plane that divides the scene into two halves intersects the armadillo, we apply the time-costly repairing step. This scene can be tetrahedralized with both methods. Figure 2a shows the resulting mesh.

Experiment 2: In this experiment, we encapsulate the armadillo with a bounding box so that the space around the armadillo can be tetrahedralized. This experiment also requires a repair step because the plane intersects the bounding box. Figure 2b shows the resulting mesh where the armadillo is at the center of the bounding box.

Experiments 3, 4, and 5: In these three experiments, the plane does not intersect any object, and hence the repairing stage is not applied. Consequently, both methods can tetrahedralize this scene. Figure 2c–e show the resulting mesh for Experiments 3, 4, and 5, respectively.

Table 2 Experimental results on the computer with Intel Xeon E5-2620 2.10 GHz processor and 64 GB of RAM. Execution time is in seconds (s) and the memory usage is in Megabytes (MB)

Experiment no.	No. vertices	No. faces	TetGen		Ours	
			Time	Memory	Time	Memory
1	172,969	345,938	37.26	850	429.44	483
2	172,969	345,938	66.54	947	456.67	564
3	345,938	691,876	43.96	1 700	138.20	922
4	1,346,688	2,693,376	294.93	6605	727.89	3584
5	1,704,146	3,408,292	380.74	8359	642.10	4537
6	17,682,248	35,364,496	91,101.46	53,160	7047.26	46,614
7	27,164,160	54,328,320	N/A	N/A	22,221.27	58,964

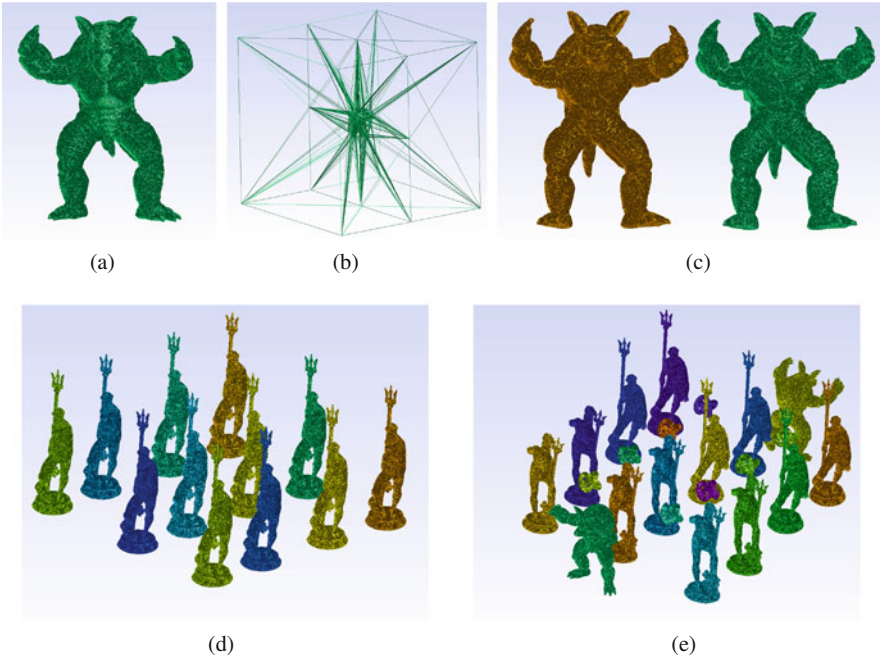


Fig. 2 Generated tetrahedral meshes. (**a**) Experiment 1. (**b**) Experiment 2. (**c**) Experiment 3. (**d**) Experiment 4. (**e**) Experiment 5

Experiment 6: This experiment takes around 117 min and consumes 47 GB of RAM with our algorithm. On the other hand, TetGen tetrahedralizes in 25 h using around 53 GB of memory. TetGen frequently make use of virtual memory to complete its task. Memory footprint shows that its virtual memory usage goes up to 90 GB (out of 130 GB). Yet, this requires an abundance of disk accesses that slows down the process. We observe that TetGen made around 200 page-faults per second, which is the main reason for the slowdown.

Experiment 7: This experiment takes around 6 h and consumes around 59 GB of RAM with our algorithm. Yet, TetGen cannot successfully tetrahedralize after 4 days of execution and force the computer to restart. TetGen uses almost all of the physical memory and consumes all of the 130 GB of virtual memory, which still is not sufficient for its execution. Hence, it cannot achieve to complete the task.

In our experiments, we observe that our method can tetrahedralize using less memory than TetGen. In some experiments, TetGen either takes more time to complete than ours or cannot complete its execution at all. TetGen continuously allocates memory as much as it needs without considering the availability. Hence, the operating system consistently provides it with memory as long as it fits into physical and virtual memory. There are two memory thresholds for TetGen. These are available physical and virtual memories. When these memories are sufficient, TetGen runs fast, as expected. If the physical memory is exhausted, the operating system allocates from virtual memory. In this case, TetGen starts making page faults because its working set cannot fit into physical memory. It directs a large portion of memory accesses to disk, which slows down TetGen. Finally, if the memory requirement of TetGen exceeds even the virtual memory, then the operating system can no longer provide memory to the TetGen, and the computer freezes and restarts itself. This phenomenon happens because TetGen lacks a mechanism to track the expected memory usage and readjust itself to stay below the memory threshold, whereas our algorithm has this mechanism.

4.2 Quality Results

In this section, we present results on the quality of the tetrahedra generated by our algorithm by comparing it with the results of TetGen. We use the aspect ratio metric used to measure the quality of tetrahedron in TetGen [5]. We calculate this metric by dividing the longest edge by the smallest height. A low aspect ratio implies high-quality tetrahedralization. As seen in Table 3, the average tetrahedron quality of TetGen is higher than our method in all cases. In torus knot, the result of TetGen is around three times better than ours, while in Armadillo, it is 1.76 times, and in Neptune, 1.48 times better on the average. Although our algorithm could generate better quality tetrahedra for Armadillo and Neptune according to the

Table 3 Experimental results on the quality of the tetrahedral mesh based on the aspect ratio metric

Model name	No. vertices	No. faces	TetGen			Ours		
			Min.	Max.	Ave.	Min.	Max.	Ave.
Torus knot	1440	2880	1.90	125.74	7.52	2.01	10,365.34	22.20
Neptune	112,224	224,448	1.30	536.49	8.72	1.28	250,088.04	12.92
Armadillo	172,969	345,938	1.30	262,232.06	7.31	1.27	260,203.69	12.84

minimum aspect ratios, overall, our tetrahedral meshes seem to be of worse quality. We expect this quality degradation because of the increase in the surface area of the object and the number of tetrahedra on the surface as we divide the point cloud into two parts. The circumspheres of these tetrahedra might extend outside the object boundary because it will not contain any point, thereby satisfying the Delaunay criterion. Hence, the quality of these tetrahedra might be reduced without breaching the constrained Delaunay criterion (see [2] for constrained Delaunay criterion).

5 Discussion and Future Work

We propose an out-of-core constrained tetrahedralization algorithm for tetrahedralizing large three-dimensional scenes. We have shown that our algorithm uses the memory more efficiently than TetGen and can tetrahedralize meshes that TetGen is unable to do because of insufficient memory. In essence, TetGen does not aim to use memory efficiently. Its main goal is computational efficiency. Therefore, TetGen tetrahedralizes meshes faster than our method if the main memory is sufficient. Our algorithm uses a divide-and-conquer approach and TetGen. In this way, we could create a memory-optimized version of TetGen by compromising the execution time.

Our algorithm divides the scene into two halves at each step, tetrahedralizes them, and finally merges them into a single tetrahedral mesh. Yet, our algorithm does not guarantee that the tetrahedra around the cut region satisfy the Delaunay criterion. In other words, the circumspheres of some tetrahedra around the cut region might contain vertices of other tetrahedra. In fact, we observed that the overall quality of the tetrahedral meshes generated by our algorithm is lower than TetGen. Hence, We are looking forward to adding a refinement process to satisfy the Delaunay criterion around the division region. When we add this step, tetrahedral mesh construction time will be longer, but the quality of the tetrahedra around the cut region will increase.

Although we use TetGen to apply constrained tetrahedralization, the clip procedure may introduce new vertices and faces on the objects in the scene where they intersect the plane. Nevertheless, it only divides a face into smaller parts, and hence the faces on the final tetrahedral mesh cover the input triangle soup.

Our algorithm works faster if the dividing plane does not intersect with any of the objects. If the dividing plane intersects the objects in the input mesh, the tetrahedralization requires costly repairing operation. If we choose the dividing plane carefully, we can tetrahedralize the mesh faster. Hence, a better dividing plane finding algorithm would be employed to avoid the intersection of the dividing plane with the objects in the scene. The dividing plane does not need to be planar; it could be an arbitrary polynomial surface or a curved surface.

Currently, our algorithm does not take the mesh density into account. Its performance in the case of a mesh with high varying density is dependent on the behavior of TetGen. A possible extension to tetrahedralize meshes of highly varying

density is to take the mesh density function as input and balance the partitions during the subdivision stage in terms of mesh density.

Acknowledgments This research is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant No. 117E881.

References

1. Blelloch, G.E., Miller, G.L., Talmor, D.: Developing a practical projection-based parallel Delaunay algorithm. In: Proceedings of the 12th Annual Symposium on Computational Geometry, SCG '96, pp. 186–195. ACM, New York (1996)
2. Chew, L.P.: Constrained Delaunay triangulations. Algorithmica **4**(1–4), 97–108 (1989)
3. Cignoni, P., Montani, C., Scopigno, R.: DeWall: a fast divide and conquer Delaunay triangulation algorithm in E^d. Comput.-Aided Des. **30**(5), 333–341 (1998)
4. Lagae, A., Dutré, P.: Accelerating ray tracing using constrained tetrahedralizations. Comput. Graph. Forum **27**(4), 1303–1312 (2008)
5. Si, H.: TetGen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. **41**(2), 1–36 (2015)
6. Smolik, M., Skala, V.: Fast parallel triangulation algorithm of large data sets in E^2 and E^3 for in-core and out-core memory processing. In: Proceedings of the International Conference on Computational Science and Its Applications, ICCSA '14, pp. 301–314. Springer, Berlin (2014)
7. The CGAL Project: CGAL User and Reference Manual, 5.0.2 edn. (2020). https://doc.cgal.org/5.0.2/Manual/packages.html
8. Woop, S., Schmittler, J., Slusallek, P.: RPU: a programmable ray processing unit for realtime ray tracing. ACM Trans. Graph. **24**(3), 434–444 (2005)

Part II
Adaptive Meshing

Size Gradation Control for Anisotropic Hybrid Meshes

Lucille-Marie Tenkes and Frédéric Alauzet

Abstract Metric-based generation methods provide high-quality anisotropic meshes. Yet, it is necessary to ensure the smoothness of the metric field in the first place. This is achieved through the so-called "metric gradation" process, that is the correction of the size growth throughout the mesh. The smallest size prescriptions are spread using a metric intersection algorithm. In this paper, we demonstrate the relevance of size gradation control in metric-based hybrid mesh generation using a metric-orthogonal point placement. We also show how to design a gradation process that maximizes the number and quality of structured elements in these hybrid meshes.

1 Introduction

Mesh adaptation has proven to strongly improve numerical simulations, both in terms of accuracy and CPU performance. In particular, metric-based generation is a mathematically well-posed framework, that enables to generate automatically highly anisotropic, adapted, unstructured meshes. However, there is still a strong demand for structured meshes, since many numerical schemes favor quadrilateral or hexahedral elements, for example in boundary layers to simulate viscous and turbulent flows. Since structured mesh adaptation is not as developed and reliable as unstructured meshes, hybrid meshes are considered as a suitable alternative solution. A hybrid mesh is a mesh that shows both structured and unstructured areas. The following work focuses on the formation of quad-dominant meshes from quasi-structured triangular meshes, which are generated through so-called *metric-orthogonal* methods [3, 6]. This process highly depends on the quality of the metric-field, particularly its smoothness. Indeed, it sometimes shows some very strong size variations, which results in flaws in the generated mesh. For

L.-M. Tenkes (✉) · F. Alauzet
INRIA Saclay Ile-de-France, Palaiseau, France
e-mail: lucille-marie.tenkes@inria.fr; frederic.alauzet@inria.fr

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_8

example, an abrupt size variation causes the formation of obtuse angles as illustrated in Fig. 1, which unnecessarily breaks the alignment of the elements. These size discontinuities are likely to hinder the formation of quadrilaterals. More generally speaking, the results of metric-based meshing and re-meshing processes highly depend on the quality of the provided metric. Consequently, it is relevant to control the size variation. Such correction consists in providing a minimal reduction of the metric at each vertex, such that the size growth is bounded.

Section 2 recalls the framework of metric-based mesh generation, then Sect. 3 details the process and algorithms of size gradation control. Numerical results and the effect of the method on hybrid mesh generation are discussed in Sect. 4.

2 Metric and Mesh Generation

This section briefly gathers some useful definitions and properties about Riemannian metric fields. These fields are used in mesh generation to prescribe size over the domain. More complete information about metric fields and the concept of unit mesh is detailed in [2, 4, 5].

A metric tensor in $\mathbb{R}^n$, $n \in \{2, 3\}$, is a symmetric, positive-definite tensor of size n that allows to define a scalar product. The notation $\mathcal{M}$ in this paper most of the time refers to a metric. A vector space $(\mathbb{R}^n, \mathcal{M})$ supplied with such scalar product is called a Euclidian metric space. The distance between two points $\mathbf{p}$ and $\mathbf{q}$ is the distance induced by the scalar product and norm defined by $\mathcal{M}$, determining as well the length of the segment $\mathbf{pq}$ denoted $\ell_{\mathcal{M}}(\mathbf{pq})$. The notions of angles and volumes are extended to an Euclidian metric field. In the case of a varying metric field, a Riemannian metric space can be defined. It consists in a continuous manifold Ω supplied with a continuous metric field $\mathcal{M}(\cdot)$ denoted by $(\mathcal{M}(\mathbf{x}))$. In this case, the field $(\mathcal{M}(\mathbf{x}))$ does not define a scalar product but changes the computation of distances, angles and volumes. The length of an edge $\mathbf{pq}$ is computed using a straight line parametrization $\gamma(t) = \mathbf{p} + t\mathbf{pq}$, $t \in [0, 1]$, therefore $\ell_{\mathcal{M}}(\mathbf{pq}) = \int_0^1 \|\gamma(t)\|_{\mathcal{M}} \, \mathrm{d}t = \int_0^1 \sqrt{\mathbf{pq}^{\mathsf{T}} \mathcal{M}(\mathbf{p} + t\mathbf{pq})\mathbf{pq}} \, \mathrm{d}t$.

This framework gives a continuous equivalent of the mesh adaptation problem. Instead of finding the best discrete mesh for a given solution, we seek its continuous equivalent metric field. A mesh is then generated using this metric field as a size prescription field, however this adapted mesh is very dependent from the quality of the metric field, hence the need of a size gradation correction beforehand.

The metric field provides intrinsic directional information through its eigenvectors. Metric-orthogonal approach[3, 6] aim at exploiting this direction field to generate right-angled elements. The points are iteratively inserted following an advancing front point placement. Metric-orthogonal meshes are quasi-structured in areas where the anisotropy ratio of the metric is high enough. Right-angled triangles are then combined into quadrilaterals to get the final hybrid mesh. To prevent the formation of poor-quality quadrilaterals, the combination step starts from highest

aspect ratio triangles and stops at a given threshold. This method gives satisfying results in areas where the metric is highly anisotropic, however when it is isotropic, the orientation of eigenvectors is not defined, so it doesn't necessarily form right-angled elements.

The process is illustrated with the example of a cross-shaped analytical metric field,

$$\mathcal{M}_{\text{cross}} = \begin{pmatrix} h_x^{-2} & 0 \\ 0 & h_y^{-2} \end{pmatrix}, \quad \text{where} \quad \begin{cases} h_x = \min(2^{\alpha_x} \times h_{\min}, h_{\max}), \\ h_y = \min(2^{\alpha_y} \times h_{\min}, h_{\max}), \\ h_{\min} = 0.005, \ h_{\max} = 0.1, \\ \alpha_x = 20 \times |x - 0.5|, \text{ and } \alpha_y = 20 \times |y - 0.5|. \end{cases}$$

Figure 1 shows generated meshes with standard (right) mesh adaptation and metric-orthogonal point-placement (left). No size gradation control has been applied. Consequently, an alignment break can be observed in the close-up view of the metric-orthogonal mesh (bottom right).

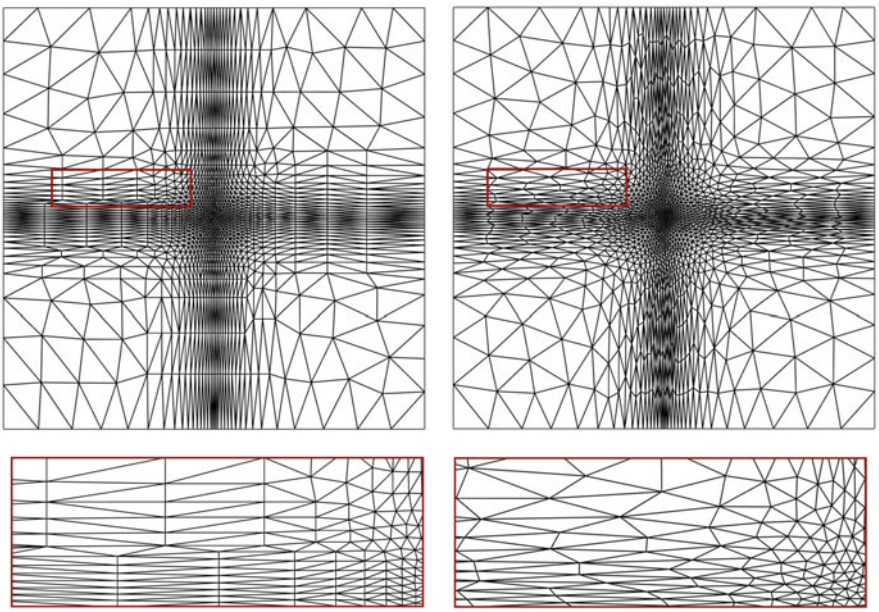


Fig. 1 Adapted meshes with metric-orthogonal (left) and standard (right) processes

3 Size Gradation Control for Anisotropic Meshes

Previous section has introduced the relevance of size gradation control to improve metric-orthogonal meshes quality. This section details the principle of this process. To introduce the definitions and notations, the method is first described in 1D. Then it is extended to a two (or three)-dimensional isotropic metric fields, and finally, to anisotropic metric fields.

Let $\mathcal{M}$ be an *isotropic* metric field defined in the entire domain Ω. The metric field can be rewritten $\mathcal{M} = h(\mathbf{x})^{-2} I$, $\mathbf{x} \in \Omega$ where I is the identity matrix. Let $\mathbf{p}$ and $\mathbf{q}$ be two points of the domain with associated sizes h_p and h_q. To quantify size variation, we define *H-shock* or *size gradation* $c(\mathbf{pq})$ related to the segment $\mathbf{pq}$ of Ω as the value

$$c(\mathbf{pq}) = \max\left(\frac{h_p}{h_q}, \frac{h_q}{h_p}\right)^{1/\ell_{\mathcal{M}}(\mathbf{pq})}.$$

It represents the spatial geometric progression of the size prescription in the metric.

This notion generalizes the classical geometrical element size progression: if it is bounded by a coefficient β, the next element is β times larger than the previous one. Accordingly, if they have a size prescription h and we ask for an increase of β, at a distance $\ell_{\mathcal{M}}$ the size prescription is $h\beta^{\ell_{\mathcal{M}}}$. The isotropic size correction consists in providing a minimal (optimal) reduction $\tilde{\mathcal{M}}$ of $\mathcal{M}$ such that, for all points, the size gradation is bounded by a given threshold β. This correction regularizes the metric field and bounds the variations. In what follows, it is assumed that $h_{\mathbf{p}} < h_{\mathbf{q}}$ so $\mathbf{p}$ is the vertex that corrects the size at $\mathbf{q}$.

The metric field is only known at the vertices of the mesh. To compute the correction numerically, the choice of an interpolation law has to be made. Depending on this choice, the coefficient $r(\mathbf{pq}) = \beta^{\ell_{\mathcal{M}}(\mathbf{pq})}$ can be computed directly or using a Newton algorithm. This aspect is detailed in [1]. For example, using the linear interpolation on h, it becomes $r(\mathbf{pq}) = 1 + \ell_p(\mathbf{pq}) \ln(\beta)$. To rewrite the computations in terms of metrics instead of sizes, the coefficient $\eta^2(\mathbf{pq}) = r(\mathbf{pq})^{-2}$ is introduced. Using the same interpolation, $\eta^2(\mathbf{pq}) = \big(1 + \ell_{\mathbf{p}}(\mathbf{pq})(\beta - 1)\big)^{-2}$. Therefore, the metric of spanned size constraints from $\mathbf{p}$ to $\mathbf{q}$, denoted by $\tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q})$ is obtained from $\mathcal{M}_{\mathbf{p}}$ as

$$\tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q}) = \eta^2(\mathbf{pq})\mathcal{M}_{\mathbf{p}}. \tag{1}$$

Equation (1) also holds to formalize size gradation control in 2D and 3D domains if the metric field is isotropic, i.e., $\mathcal{M} = h(\mathbf{x})^{-2} I$. In the case of a mesh $\mathcal{H}$ supplied with a discrete metric field given at its vertices, each vertex provides a metric for all the other vertices that imposes its size constraints in all directions. The reduced

metric at vertex $\mathbf{q}$ is given by the intersection of the corrected metrics, which is the largest metric taking all the constraints in account.

$$\tilde{\mathcal{M}}(\mathbf{q}) = \left(\bigcap_{\mathbf{p} \in \mathcal{H}} \tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q}) \right) \cap \mathcal{M}(\mathbf{q}).$$

Unfortunately, we face a quadratic complexity algorithm. To avoid this, it is possible to approximate the mesh gradation problem with a linear complexity algorithm, as presented in [1]. It is an iterative process, each iteration being a correction loop on the mesh edges. This algorithm is fast and has given nice results in most cases but it is sensitive to mesh topology.

Metric gradation is quite intuitive when the metric field is isotropic. However anisotropy adds some difficulty, mostly in the metric spanning step. Indeed, with anisotropic metric fields, different size constraints in all directions are involved, so the previously described gradation process cannot be straightforwardly extended. A choice must be made to determine how the size constraints are spanned to bound the size variation correctly. Two options are considered here.

In the first option, the metric is spanned according to a scalar growth factor. That is,

$$\tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q}) = \eta^2(\mathbf{pq}) \mathcal{M}_{\mathbf{p}},$$

where $\eta^2(\mathbf{pq}) = \big(1 + \ell_{\mathbf{p}}(\mathbf{pq})(\beta - 1)\big)^{-2}$ in the h-linear interpolation. The resulting growth is homogeneous in the metric space, since the anisotropic ratio is constant. This method is referred to in the remaining as "metric space growth".

The second option is to apply a growth that depends on the direction, by setting a different growth factor to each eigenvalue. In this case, the constraint metric is

$$\tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q}) = \mathcal{R}^{\mathsf{T}} \mathcal{N}(\mathbf{pq}) \Lambda \mathcal{R},$$

where $\Lambda = \mathrm{diag}((\lambda_i)_{i=1,\dots,n})$ is the diagonal matrix containing the eigenvalues of $\mathcal{M}_{\mathbf{p}}$, $\mathcal{R}$ is the orthogonal matrix of its eigenvectors, $\mathcal{N}(\mathbf{pq}) = \mathrm{diag}((\eta_i^2)_{i=1,\dots,n})$ and $\eta_i^2(\mathbf{pq}) = \big(1 + \sqrt{\lambda_i} \|\mathbf{pq}\|_2 (\beta - 1)\big)^{-2}$. The factor $\sqrt{\lambda_i}$ makes the growth faster in directions of small sizes. As a consequence, $\tilde{\mathcal{M}}_{\mathbf{p}}(\mathbf{q})$ tends to an isotropic tensor when the distance between $\mathbf{p}$ and $\mathbf{q}$ increases. This processed is called in the remaining "physical space growth".

The difference between the two growth process is illustrated in Fig. 2. The choice of either method has a strong effect on the resulting metric-orthogonal mesh, as shown in the next section.

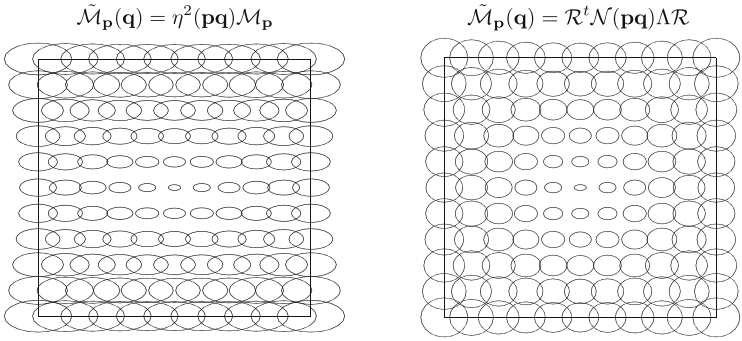


Fig. 2 Metric-space growth (left) and physical space growth (right)

4 Numerical Results

This section compares the two processes of metric growth on two analytical examples and two numerical simulations around a NACA airfoil. The adequacy of the resulting metric-orthogonal meshes is quantified by analyzing the size jumps and the angles repartition. The size jump indicator is evaluated at each vertex as the ratio between the largest and the smallest area of the surrounding triangles. The quality of quadrilaterals is evaluated using the following quality function from [7] based on the quadrilateral's angles

$$\eta(q) = \max\left(1 - \frac{2}{\pi}\max_k\left(\left|\frac{\pi}{2} - \alpha_k\right|\right), 0\right).$$

4.1 Line

This first case is an analytical straight metric field on a square domain $\Omega = [0, 1] \times [0, 1]$. The initial metric is deliberately discontinuous to highlight the difference between the two processes:

$$\mathcal{M}_{\text{line}} = \begin{pmatrix} {h_1}^{-2} & 0 \\ 0 & {h_2}^{-2} \end{pmatrix} \quad \text{if } y = 0, \quad \text{and} \quad \begin{pmatrix} {h_0}^{-2} & 0 \\ 0 & {h_0}^{-2} \end{pmatrix} \quad \text{elsewhere.}$$

The prescribed sizes are $h_1 = 0.1$, $h_2 = 0.001$, and $h_0 = 0.5$. A gradation correction with $\beta = 1.1$ is applied. The resulting metric-orthogonal meshes are displayed in Fig. 3. As expected, this example shows clearly the difference between the two methods. The physical space growth seems to favor the formation of quadrilaterals in this case. Indeed, the histograms show that the proportion of size jumps and obtuse angles is higher in the metric-space mesh. This is corroborated by

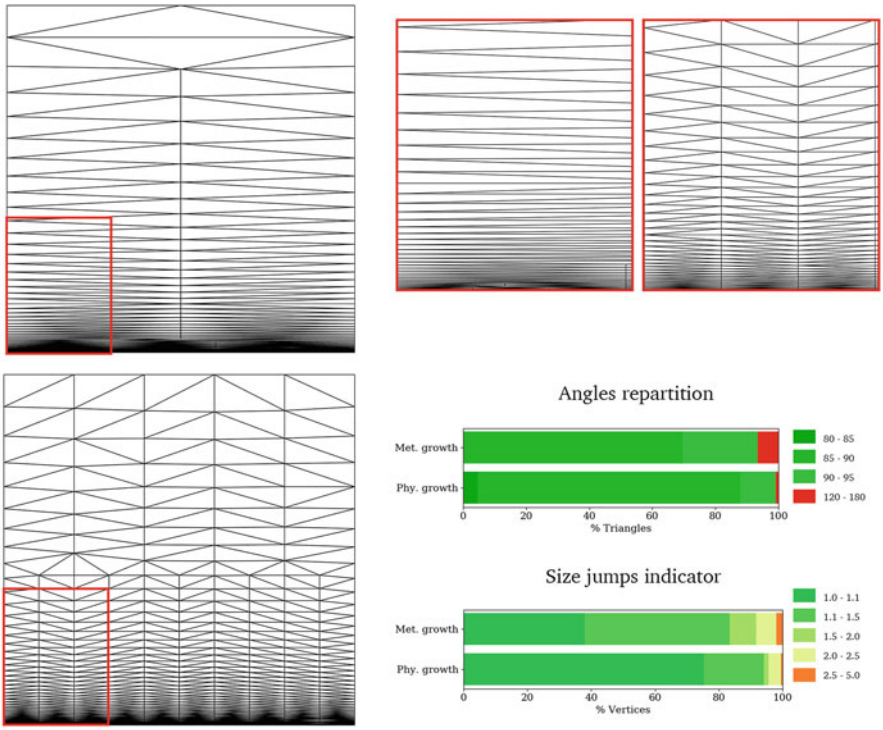


Fig. 3 Metric-orthogonal adapted mesh from the 'line' metric. On the left side, a large scale view of each mesh is displayed, respectively top and bottom result from a metric-space and physical-space growth. On the right side, the top part shows close-up views corresponding to the red rectangles, and the bottom part displays the angles and size jumps histograms

the corresponding hybrid meshes, since the proportion of quadrilaterals is 93.8% in the metric space case and 70.5% in the physical space case.

4.2 Circle

This example is meant to study how metric curvature is handled by the process. Our method is applied to an analytical metric field representing a curved anisotropic feature having the shape of a circle:

$$\mathcal{M}(x,y) = \begin{pmatrix} h_1^{-2}\cos^2\theta + h_2^{-2}\sin^2\theta & \left(h_1^{-2} - h_2^{-2}\right)\cos\theta\sin\theta \\ \left(h_1^{-2} - h_2^{-2}\right)\cos\theta\sin\theta & h_1^{-2}\sin^2\theta + h_2^{-2}\cos^2\theta \end{pmatrix},$$

with $\theta = \arctan(x,y)$, $h_1 = \min(0.002 \times 5^{\alpha}, h_{\max})$, $h_2 = \min(0.05 \times 2^{\alpha}, h_{\max})$, $h_{\max} = 0.1$, and $\alpha = 10 \times |0.75 - \sqrt{x^2+y^2}|$.

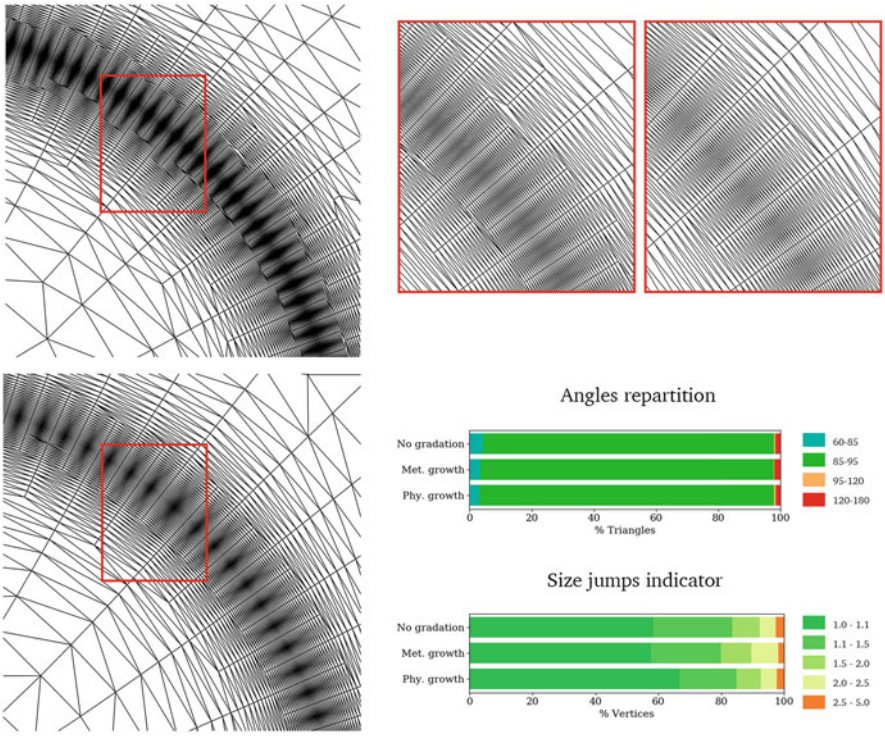


Fig. 4 Metric-orthogonal adapted mesh from the circle metric. On the left side, a large scale view of each mesh is displayed, respectively top and bottom result from a metric-space and physical-space growth. On the right side, the top part shows close-up views corresponding to the red rectangles, and the bottom part displays the angles and size jumps histograms for these two meshes and a no-gradation mesh

Here, $\beta = 1.2$ and a numerical artifact is added to improve the results of metric-space growth: as presented in [1], the dependency on mesh topology creates some 'rays' that are tangent to the curved areas, the metric field is wrongfully modified and so are the resulting meshes. To overcome this issue, the threshold β is multiplied by a coefficient δ (here $\delta = \beta = 1.2$) at each iteration. The results are gathered in Fig. 4 and details of the hybrid meshes are shown on Fig. 5.

Again, both growth processes give satisfying results: metric-space growth gives 90.9% quadrilaterals and physical-space growth gives 92.3%. However, the metric's curvature is handled differently and physical-space growth shows less size jumps in the radial direction. Consequently, quadrilateral-only areas are more easily formed, as observed in Fig. 5, which is more suitable for numerical simulations. The results for metric-space growth are somehow not as good as the mesh obtained without gradation, although the histogram on Fig. 5 shows that the proportion of best quality quadrilaterals is higher for the corrected meshes, which was expected since poor quality elements are formed in low-anisotropy areas, and gradation correction

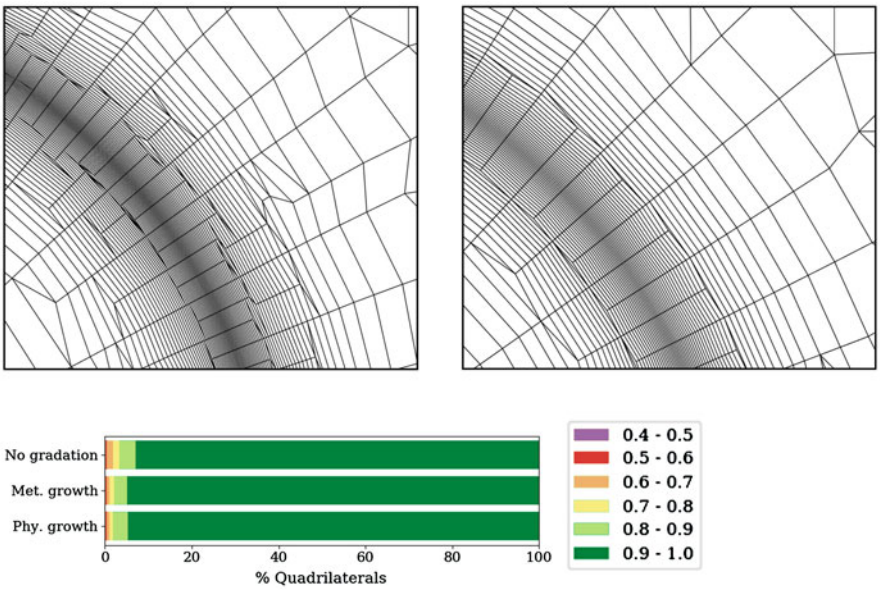


Fig. 5 Hybrid meshes following metric-space growth (left) and physical space growth (right) gradation. The histogram at the bottom shows the repartition of the quadrilateral's quality

induces a propagation of anisotropy. Moreover, the initial metric is quite smooth and bounded, so the improvement due to size gradation correction is limited.

4.3 Inviscid Flow Simulation

This example is an inviscid flow simulation on a NACA 0012, modeled by the incompressible Euler equations. A supersonic flow is considered with a Mach number $M = 1.6$ and an angle of attack $\alpha = 8°$. For both growth process, a gradation correction was performed with $\beta = 1.5$ and a coefficient $\delta = 1.5$ was added for metric-space growth.

Adapted metric-orthogonal meshes for both growth processes are shown in Fig. 6, and the histograms for comparison can be found in Fig. 7. Histograms reveal that physical space growth is clearly the best option for this test case: the proportion of size jumps and wrong angle is considerably smaller. The close-up views of the hybrid meshes also demonstrate that this growth process also favors the formation of larger all-quadrilateral areas. Although metric-space growth captures the shock more precisely [1], it is not the best option for hybrid meshing.

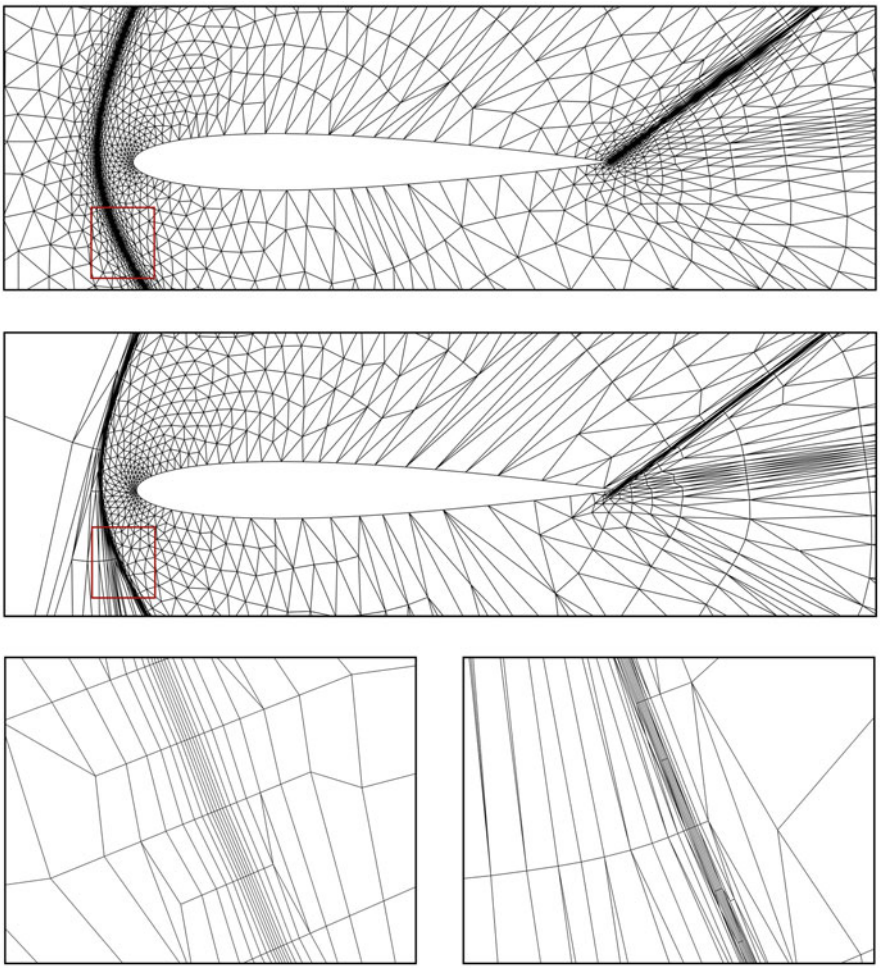


Fig. 6 Supersonic inviscid flow NACA case. Top part shows large scale views of the adapted metric-orthogonal meshes (top is physical-space and bottom is metric-space growth), and the respective close-up views after combination into quadrilaterals

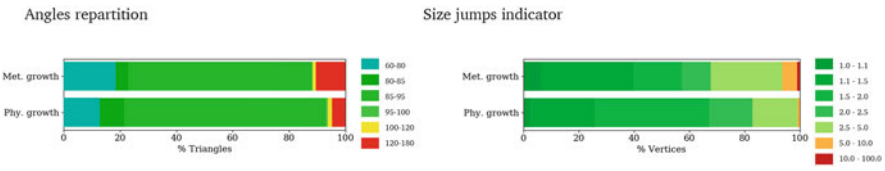


Fig. 7 Supersonic inviscid flow NACA case. Histograms comparing the metric-orthogonal meshes from metric-space and physical-space growth

4.4 Turbulent Flow Simulation

The last example is a turbulent flow simulation on a NACA0012 modeled by the Reynolds-Averaged Navier-Stokes equations, where we consider the Spalart-Allmaras one equation turbulence model. The simulated flow is subsonic with a Mach number $M = 0.5$, a Reynolds number $Re = 10^5$ and an incidence angle $\alpha = 0°$. The adapted meshes are shown in Fig. 8 and the angles and size jumps histograms are presented in Fig. 9. This case is again better handled by the metric gradation with physical-space growth. The effect is particularly observable in the boundary layer region, where the smoothness of the transitions is essential. The small size prescribed at the boundary is propagated throughout almost the entire zone, whereas numerous transitions break the alignment when using the metric-

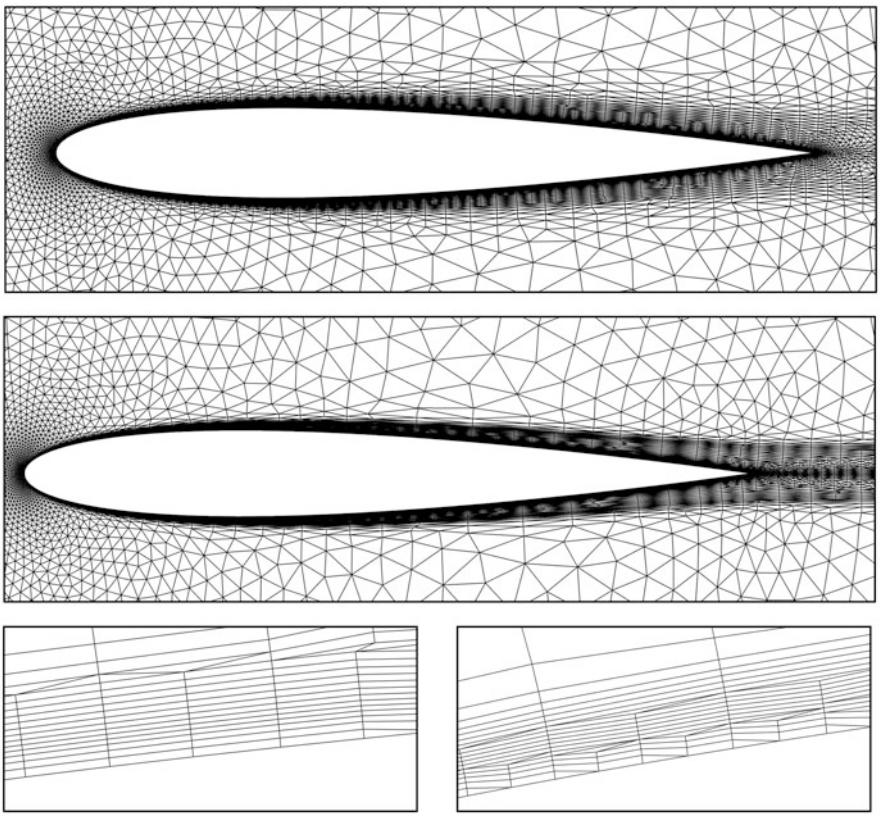


Fig. 8 Subsonic turbulent flow NACA case. Comparison of the two growth processes. Respectively large scale views of the metric-orthogonal mesh and close-up views of the hybrid mesh in the boundary layer region, for physical-space (top and left), and metric-space growth (bottom and right)

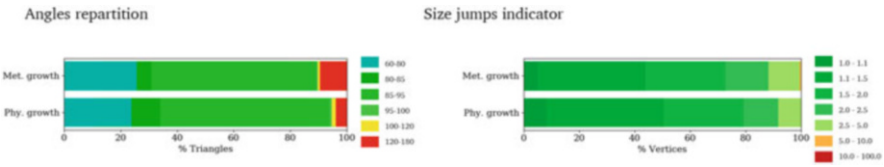


Fig. 9 Subsonic turbulent flow NACA case. Histograms comparing the metric-orthogonal meshes from metric-space and physical-space growth

space growth gradation. This is a promising lead for the purpose of generating hybrid meshes in which the boundary layer is entirely meshed with quadrilaterals.

5 Conclusion

Metric gradation control is an essential step in a metric-based mesh adaptation process. In the standard adaptation, it improves consequently the quality of the mesh and its performances, as shown in previous work. This process is even more important for metric-orthogonal meshes: the metric has to be even smoother since size variations are likely to break the alignment of the mesh and prevent the formation of quadrilaterals in the combination step.

For this purpose, our current metric gradation method has been analyzed to determine how it could favor the formation of good quality orthogonal meshes, and hybrid mesh afterward, i.e., increase the proportion of quadrilaterals and reduce the number of size jumps and obtuse angles. The most crucial step is the metric growth phase, in which the metric at a certain vertex is spanned throughout the domain, such that it mainly affects the closest vertices. Two possibilities have been considered. First, the metric-space homogeneous growth, conserving the anisotropic ratio as it spans. Then, the physical-space homogeneous growth, imposing a faster growth on the smallest prescribed size and tending to become isotropic as it spans. In previous work on the subject, it has been observed that the first growth process leads to better meshes for the considered CFD simulations. However, for the purpose of quad-dominant meshes, the second growth process is more effective since the resulting meshes show smoother size transitions and more potential all-quadrilateral zones. To perfect the quad-dominant meshing procedure, some improvement remains to be made on the re-meshing and combining steps.

References

1. Alauzet, F.: Size gradation control of anisotropic meshes. Finite Elem. Anal. Des. **46**, 181–202 (2010). https://doi.org/10.1016/j.finel.2009.06.028
2. Hecht, F., Mohammadi, B., Hecht, F., Mohammadi, B.: Mesh adaption by metric control for multi-scale phenomena and turbulence. In: 35th Aerospace Sciences Meeting and Exhibit (1997). https://arc.aiaa.org/doi/abs/10.2514/6.1997-859

3. Loseille, A.: Metric-orthogonal anisotropic mesh generation. Procedia Eng. **82**, 403–415 (2014). https://doi.org/10.1016/j.proeng.2014.10.400, https://www.sciencedirect.com/science/article/pii/S1877705814016798. 23rd International Meshing Roundtable (IMR23)
4. Loseille, A., Alauzet, F.: Continuous mesh framework part I: well-posed continuous interpolation error. SIAM J. Numer. Anal. **49**, 38–60 (2011). https://doi.org/10.1137/090754078
5. Loseille, A., Alauzet, F.: Continuous mesh framework part II: validations and applications. SIAM J. Numer. Anal. **49**, 61–86 (2011). https://doi.org/10.2307/23074390
6. Marcum, D., Alauzet, F.: 3D metric-aligned and orthogonal solution adaptive mesh generation. Procedia Eng. **203**, 78–90 (2017). https://doi.org/10.1016/j.proeng.2017.09.790, https://www.sciencedirect.com/science/article/pii/S1877705817343485. 26th International Meshing Roundtable, IMR26, 18–21 September 2017, Barcelona, Spain
7. Remacle, J.F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., Geuzainet, C.: Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. Int. J. Numer. Methods Eng. **89**(9), 1102–1119 (2012). https://doi.org/10.1002/nme.3279

Adjoint Computation on Anisotropic Meshes in High-fidelity RANS Simulations

Francesco Clerici and Frédéric Alauzet

Abstract In the context of high-fidelity RANS simulation, anisotropic mesh adaptation turns out to be an excellent tool to get mesh-independent solutions on complex geometries. In particular, anisotropic mesh adaptation is used to predict accurately dimensionless quantities such as the lift and the drag coefficients, and, in general, functionals depending on the solution field. However, in order to get the optimal adapted mesh with respect to the accuracy of an output functional, it is necessary to solve an adjoint system providing the sensitivity of the goal functional with respect to the residuals of the equations. The linear system associated to the adjoint problem revealed to be stiff for RANS equations with a standard solver such as the GMRES preconditioned with several SGS iterations, and hence an alternative method has been developed, which is based on the transient simulation of the RANS adjoint state.

1 Introduction

Anisotropic mesh adaptation is an efficient tool to deal with CFD simulations. It can automatically handle meshing operations on complex geometries without heavy computational effort, while getting accurate results on inviscid, viscous and turbulent flows [1–5]. In particular, here we will make use of the Navier-Stokes equations in conservative form and the Spalart-Allmaras turbulence model [6]. The resulting RANS model can be written in a vector form of the type

$$W_t + \nabla \cdot \mathcal{F}^{\mathrm{E}} = \nabla \cdot \mathcal{F}^{\mathrm{V}} + \nabla \cdot \mathcal{F}^{\mathrm{S}}, \tag{1}$$

F. Clerici (✉) · F. Alauzet
INRIA Saclay Ile-de-France, Palaiseau, France
e-mail: francesco.clerici@inria.fr; frederic.alauzet@inria.fr

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_9

by defining the primal state

$$W = (\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{\nu}),$$

with ρ the density, $\mathbf{u} = (u, v, w)^T$ the velocity, E the specific energy, $\tilde{\nu}$ the turbulent kinematic viscosity, the Euler fluxes vector being

$$\mathcal{F}^{\mathrm{E}}(W) = (\rho\mathbf{u}, \rho u\mathbf{u} + p\mathbf{e}_1, \rho v\mathbf{u} + p\mathbf{e}_2, \rho w\mathbf{u} + p\mathbf{e}_3, \mathbf{u}(\rho E + p), \rho\tilde{\nu}\mathbf{u})^T, \tag{2}$$

with $\{\,\mathbf{e}_i\,\}_{i=1}^3$ the canonical basis in $\mathbb{R}^3$, the viscous fluxes vector being

$$\mathcal{F}^{\mathrm{V}}(W) = \left(0, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}\cdot\mathbf{u} + \lambda\nabla T, \frac{\rho}{\sigma}(\nu + \tilde{\nu})\nabla\tilde{\nu}\right)^T, \tag{3}$$

with $\{\,\mathcal{T}_i\,\}_{i=1}^3$ the rows of the stress tensor $\mathcal{T}$, λ the laminar conductivity, ν the kinematic viscosity, $\sigma = 2/3$, and the source term vector $\mathcal{F}^{\mathrm{S}}(W)$ contains the production and the destruction terms of the turbulent kinematic viscosity $\tilde{\nu}$ of the Spalart-Allmaras turbulence model [6].

Having defined the primal problem at hand, in the next section we will briefly describe its implementation.

2 Flow Solver

The discretization of Eqs. (1) is made by using a mixed finite element—finite volume discretization on triangular (2D) or tetrahedral (3D) meshes [7, 8]. In particular, the viscous fluxes are discretized by means of finite element, while the convective fluxes by means of finite volume. The convective fluxes at each cell interface are computed using an HLLC approximate Riemann solver [9], and the gradients are constructed using a second order MUSCL extrapolation scheme [10] using the gradients of W on the upwind and the centered elements, ensuring a low dissipation, possibly coupled to a slope limiter. Note that we are interested in the steady state to Eq. (1), and in the sequel this is reached by means of a pseudo-transient algorithm: for this reason the time derivative is kept and discretized in a proper way. The solution W is attached to each vertex of the mesh $\mathcal{H}$, and the convective fluxes are computed by using a dual mesh to $\mathcal{H}$, where its elements are made by the median cells of the mesh $\mathcal{H}$. For instance, in 2D, a median cell C_i associated with a vertex P_i is built using the segments of medians of all triangles sharing P_i. The resulting semi-discretized system reads as

$$|C_i|\frac{\mathrm{d}W_i}{\mathrm{d}t} + \mathbf{F}_i = \mathbf{S}_i + \mathbf{Q}_i + \mathbf{\Gamma}_i \quad \forall i = 1, \ldots, N_V, \tag{4}$$

where $|C_i|$ is the measure of the cell C_i, W_i the mean value of W on C_i, N_V the number of vertices, $\mathbf{F}_i, \mathbf{S}_i, \mathbf{Q}_i, \mathbf{\Gamma}_i$ are respectively the numerical convective, viscous and source flux and boundary terms defined as

$$\mathbf{F}_i = \int_{\partial C_i} F(W_i) \cdot \mathbf{n}_i \, \mathrm{d}\Gamma \,, \qquad \mathbf{S}_i = \int_{\partial C_i} S(W_i) \cdot \mathbf{n}_i \, \mathrm{d}\Gamma \,,$$
$$\mathbf{Q}_i = \int_{C_i} Q(W_i) \, \mathrm{d}\Omega \,, \qquad \mathbf{\Gamma}_i = \int_{\partial C_i \bigcap \partial\Omega} G(W_i) \, \mathrm{d}\Gamma \,,$$

where F, S, Q are defined following the definitions of the fluxes (2), (3), and $\mathcal{F}^{\mathrm{S}}(W)$, and G is a function depending on the chosen boundary conditions. The system (4) is discretized in time by using the backward Euler scheme, getting

$$\frac{|C_i|}{\delta t_i^n} \delta W_i = -\mathbf{F}_i^{n+1} + \mathbf{S}_i^{n+1} + \mathbf{Q}_i^{n+1} + \mathbf{\Gamma}_i^{n+1}, \tag{5}$$

where $\delta W_i = W_i^{n+1} - W_i^n$ and δt_i^n is the local time step at time iteration n, given by

$$\delta t_i^n = CFL_i \frac{h^2}{h(c + \|\mathbf{u}\|) + 2\frac{\lambda + \lambda_t}{\rho}}, \tag{6}$$

where h is the smallest height of the elements sharing P_i and CFL_i is the local CFL number. The system (5) is linearized with respect to W, and an approximating linear system for each time step is found

$$\mathbf{A}^n \delta W^n = \mathbf{R}^n, \tag{7}$$

where

$$\mathbf{A}^n = \frac{|C|}{\delta t^n} \mathbb{1} - \frac{\partial \tilde{\mathbf{R}}^n}{\partial \mathbf{W}}, \tag{8}$$

and $\mathbf{R}^n = -\mathbf{F}_i^n + \mathbf{S}_i^n + \mathbf{Q}_i^n + \mathbf{\Gamma}_i^n$ are the residuals, while $\tilde{\mathbf{R}}^n = -\tilde{\mathbf{F}}_i^n + \mathbf{S}_i^n + \mathbf{Q}_i^n + \mathbf{\Gamma}_i^n$ are the residuals computed by using a first order scheme the convective terms. This is done in order to reduce the memory requirement to store the matrix $\mathbf{A}^n$. The resolution process usually makes use of a starting solution W^0, and the linear system (7) is solved by using several iterations of the symmetric Gauss-Seidel (SGS) algorithm [11, 12] at each time step. Depending on the convergence of the linear system, the local CFL_i number is increased or reduced; notice that for $CFL \to \infty$, the system (7) is equivalent to the Newton method to solve (5). `Wolf` is a CFD solver written in C implementing the procedures described above [13, 14]. For further details about the discretization and implementation see [5].

3 Adjoint System

Once the norm of the residuals $\mathbf{R}$ has reached a proper threshold, we end up with a solution $\mathbf{W}_h$, where the subscript h indicates that this is a discrete finite element function. Such solution can be used to compute the value of a functional, for instance the lift or the drag coefficient. By introducing the pressure and the skin friction coefficients respectively given by

$$C_p(\mathbf{x}) = \frac{p(\mathbf{x}) - p_\infty}{\frac{1}{2}\rho_\infty \|\mathbf{u}_\infty\|^2}\mathbf{n}(\mathbf{x}), \qquad C_f(\mathbf{x}) = \frac{(\mu + \mu_t)\tau(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})}{\frac{1}{2}\rho_\infty \|\mathbf{u}_\infty\|^2}\mathbf{n}(\mathbf{x}) \tag{9}$$

with τ defined as

$$\tau = \left(\nabla \otimes \mathbf{u} + \nabla \otimes \mathbf{u}^T\right) - \frac{2}{3}\nabla \cdot \mathbf{u}\mathbb{1},$$

and the subscript ∞ referring to a far field quantity, the aerodynamic coefficient vector can be computed by integration on the considered body S up to a rotation as

$$\begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} = \frac{1}{S_{\text{ref}}} \int_S \left(C_p(\mathbf{x}) + C_f(\mathbf{x})\right) \mathrm{d}\mathbf{x}\,, \tag{10}$$

with S_{ref} a reference surface area. The aim of this analysis is to find a mesh $\tilde{\mathcal{H}}$ which minimizes the error committed on a functional of the solution, such as (9) or (10). More in detail, given a functional $J(W)$, we want to measure (and bound) the quantity

$$|J(W) - J(W_h)| \approx \left|\left(\frac{\partial J}{\partial W}, W - W_h\right)\right| \tag{11}$$

with W the solution to the continuous problem (1), assuming

$$W \in \mathcal{V} = \left[H^1(\Omega) \cap C^3(\bar{\Omega})\right] \times \left[H_0^1(\Omega) \cap C^3(\bar{\Omega})\right]^3 \times \left[H^1(\Omega) \cap C^3(\bar{\Omega})\right].$$

After some manipulations [5], one gets an upper bound to (11) of the type

$$|J(W) - J(W_h)| \leq \int_\Omega \sum_i |G(W_i, W_i^*)|\ |W_i - \Pi_h W_i|\,, \tag{12}$$

where $G(W_i, W_i^*)$ are weights to the interpolation errors $|W_i - \Pi_h W_i|$ depending on the primal state W and, most of all, on the adjoint state W^*, which is defined by

the following problem

$$\left(\frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}\right)^T W^* = \left(\frac{\partial J}{\partial \mathbf{W}}\right). \tag{13}$$

This kind of analysis can also be applied on Euler or laminar flow [2, 3]. After the actual computation of the quantities involved in (12), we use them to create a new mesh $\tilde{\mathcal{H}}$ by means of metric-based anisotropic mesh adaptation [15], and for this reason it is necessary to compute carefully the adjoint state W^*. We first note several facts on the system (13). The matrix $\left(\frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}\right)^T$ is obtained by transposing the Jacobian of the residual. Hence, the stiffness of the system (13) is equivalent to the one of the system (7) for an infinite time step: one of the main difficulties to solve system (13) is that it is particularly stiff, specially when the mesh $\mathcal{H}$ is highly anisotropic. This difficulty is circumvented in the primal state computation, as its matrix has more diagonal dominance thanks to the presence of the term $\frac{|C|}{\delta t^n}\mathbb{1}$, and this dominance can be properly tuned by setting the CFL, while this cannot be done a priori for the problem (13). In the next section we will discuss how these drawbacks can be bypassed with the application of a pseudo-transient algorithm to the problem (13). For further details on the error estimation see [5], while for further details regarding mesh adaptation see [15].

4 Numerical Implementation

When employing an adaptive procedure with the methodology of Sect. 3, it is important to measure the independency of the flow field from the computational mesh. This is usually reached by studying the convergence of the solution with respect to the number of vertices of a mesh, but the rise of the number of vertices leads in general to a high anisotropy in the mesh. This could be a drawback for the resolution of the primal problem and, most of all, of the adjoint problem, as it has been experienced a high dependency of the problems stiffness and the amount of anisotropy of the mesh. In `Wolf`, a GMRES algorithm [16] with an SGS preconditioner is by default devoted to the solution of the linear system (13). Usually we set a Krylov space size around 200, with 1000 total number of iterations, and with 20–40 SGS iterations in the preconditioning phase. With a high anisotropy of the computational mesh, even Krylov spaces of dimension 1000 are not enough to reach the convergence, and the residual of the system (13) stalls on high values, providing non-consistent solutions and a waste of computational time. In particular, a non-consistent adjoint solution has a negative impact on the mesh adaptation procedure. We underline that a Krylov space size greater than 1000 requires prohibitive computational resources. The idea behind a pseudo-transient adjoint solver is that if the algorithm for the computation of the primal state reaches the convergence, then the same algorithm (characterized by the same time accuracy,

CFL, relaxation strategies, cells measure...) applied to a transient adjoint equation should provide a steady state [17–19]. The transient adjoint equation reads as

$$M\frac{\partial W^*}{\partial t} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T W^* - \frac{\partial J}{\partial \mathbf{W}} = 0, \tag{14}$$

where M is the mass matrix including the area or volume information of the cells and the time step.

Since the mass matrix M is trivially invertible, Eq. (14) can be interpreted as a system of linear ODEs of the form

$$\dot{x} + Ax = b. \tag{15}$$

The solution of (15), when A is invertible and $x(0) \neq \mathbf{0}$, reads as

$$x(t) = x^s + \mathrm{e}^{-At}\big(x(0) - x^s\big), \tag{16}$$

where x^s is the searched stationary solution, $Ax^s = b$, and

$$\mathrm{e}^B := \sum_{m=0}^{\infty} \frac{1}{k!} B^k. \tag{17}$$

Hence, the convergence depends on the eigenvalues of the matrix A. All the eigenvalues must be positive. For an ill-conditioned matrix, we could have at least one small eigenvalue that will slow down the convergence to a steady state.

In the most extreme case, that is when the matrix A is singular, the general solution to (15) reads as (provided $x(0) \neq \mathbf{0}$)

$$x(t) = \mathrm{e}^{-At}x(0) + \int_0^t \mathrm{e}^{-As} b \,\mathrm{d}s\,. \tag{18}$$

Hence, any solution (if existing) satisfying $Ax = b$ will depend on the initial value $x(0)$. This fact is particularly important as we will always start out adjoint computation from an initial guess. If we discretize (14) by means of a backward Euler scheme, we get

$$\frac{M}{\mathrm{d}t}\Big(\big[W^*\big]^{n+1} - \big[W^*\big]^n\Big) + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \big[W^*\big]^{n+1} - \frac{\partial J}{\partial \mathbf{W}} = 0, \tag{19}$$

and, hence,

$$\left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T\right)\big[\delta W^*\big]^n = \frac{\partial J}{\partial \mathbf{W}} - \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \big[W^*\big]^n. \tag{20}$$

We remark that

$$\left[W^*\right]^n = \sum_{k=1}^{n-1} \left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \right)^{-k} \frac{\partial J}{\partial \mathbf{W}}, \tag{21}$$

hence, (21) converges to a steady state if

$$\forall i, \quad \lambda_i \left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \right) > 1. \tag{22}$$

The condition (22) can be reached by setting a proper *CFL*. Hence, this suggests to start from a low *CFL* value, and then increase it until the residual at step n, given by

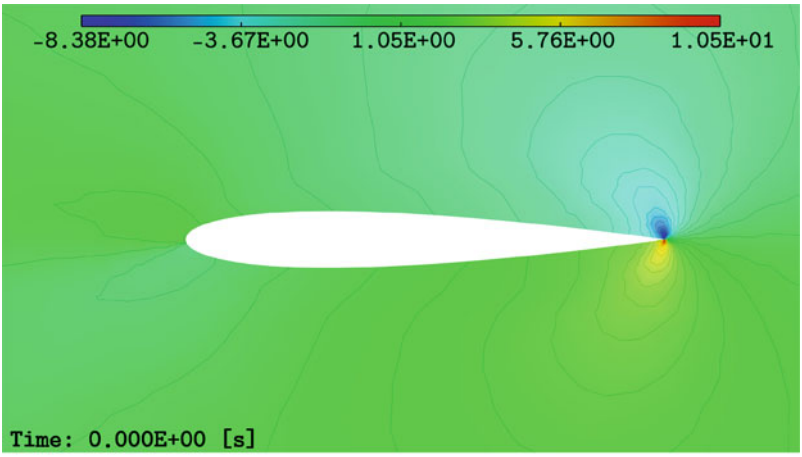
$$R^n_{\text{adj}} = \left\| \frac{\partial J}{\partial \mathbf{W}} - \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \left[W^*\right]^n \right\|, \tag{23}$$

becomes bigger than the residual at step $n-1$ (meaning that the solution is diverging). This is done also on the primal solver. Note, anyway, that no divergence should happen when the Jacobian $\frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}$ is not ill-conditioned. In any case, provided that the system (20) is solved exactly, the convergence to a steady state is guaranteed. If this is the case, the difference between two consecutive solutions at step n and $n-1$ is given by

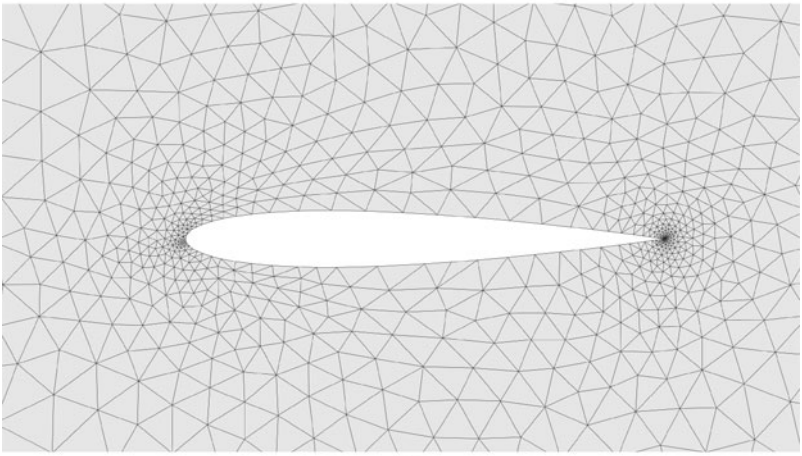
$$\left\| \left[\delta W^*\right]^n \right\| = \left\| \left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \right)^{-n} \frac{\partial J}{\partial \mathbf{W}} \right\|, \tag{24}$$

hence, provided all the hypothesis, we expect $\left\| \left[\delta W^*\right]^k \right\|$ to converge to zero monotonically. The linear system (20) is solved using several SGS iterations, which does not ensure the convergence to a steady state and does not ensure (24) to converge to zero monotonically. In any case, one can choose a rather low *CFL* in order to enforce the diagonal dominance of the matrix $\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T$ and hence reach a high accuracy in the SGS solver by using a fixed number of iterations, but this will slow down the convergence of the quantity (23). Because of these considerations, the *CFL* should be chosen in order to reach a proper balance between the rate of convergence to a steady state (high *CFL*) and an accurate resolution given by the SGS solver (low *CFL*).

A preliminary test of the pseudo-transient adjoint solver has been made on a NACA airfoil by solving an inviscid subsonic flow, and on a turbulent flow on a multi-element airfoil in a high-lift configuration using the Navier Stokes equations with the Spalart-Allmaras turbulence model. The computational mesh, the physical parameters (Mach number *Ma*, angle of attack α, Reynolds number *Re* and number



(a)



(b)

Fig. 1 NACA inviscid subsonic test adapted mesh case. $Ma = 0.3$, $\alpha = 2°$, $N_V = 1.18 \times 10^3$. (**a**) Density adjoint field. (**b**) Adapted mesh

of vertices N_V) and the adjoint density fields are shown in Figs. 1 and 2, respectively. Such results have been obtained by applying at most 10 SGS iterations for each time step, with a tolerance of 0.01 on the residual of the system (20). The stopping criterion on the time increment is based on the adjoint system residual, with a threshold of 10^{-12}. Both the solutions can also be easily obtained with few GMRES iterations. The *CFL* law is geometric, using a multiplication factor of $m_{\text{cfl}} = 1.025$,

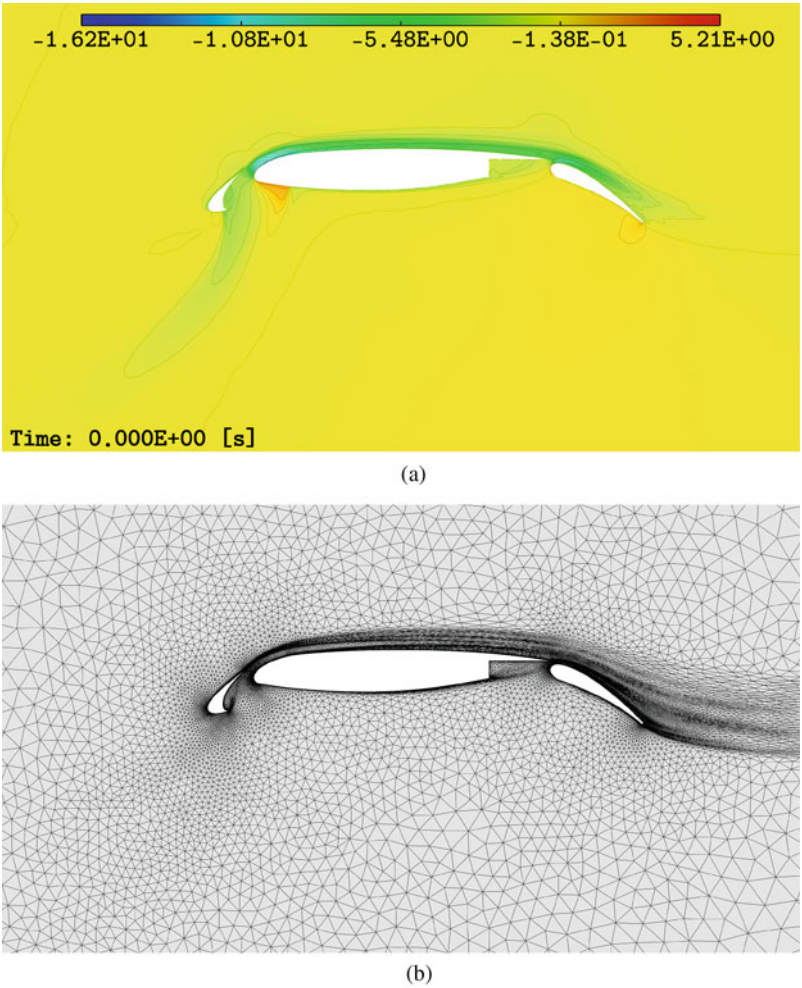


Fig. 2 Multi-element airfoil turbulent flow test case. $Ma = 0.175$, $\alpha = 16.21°$, $Re = 15.1 \times 10^6$, $N_V = 3.85 \times 10^4$. (**a**) Density adjoint field. (**b**) Adapted mesh

and it is modified as follows

$$CFL^n = \begin{cases} m_{\text{cfl}} CFL^{n-1}, & \text{if } R^n_{\text{sgs}} \leq R^{n-1}_{\text{sgs}}, \\ \frac{CFL^{n-1}}{m^4_{\text{cfl}}}, & \text{if } R^n_{\text{sgs}} > R^{n-1}_{\text{sgs}}. \end{cases} \tag{25}$$

That is, we multiply once the *CFL* by m_{cfl} if the SGS solver properly reduces the residual of the time step system, otherwise we divide it by m^4_{cfl}. In order to speed up the computation, exclusively for these two simulations, the initial *CFL* is set to its maximum allowed value. The convergence history is shown in Fig. 3, having

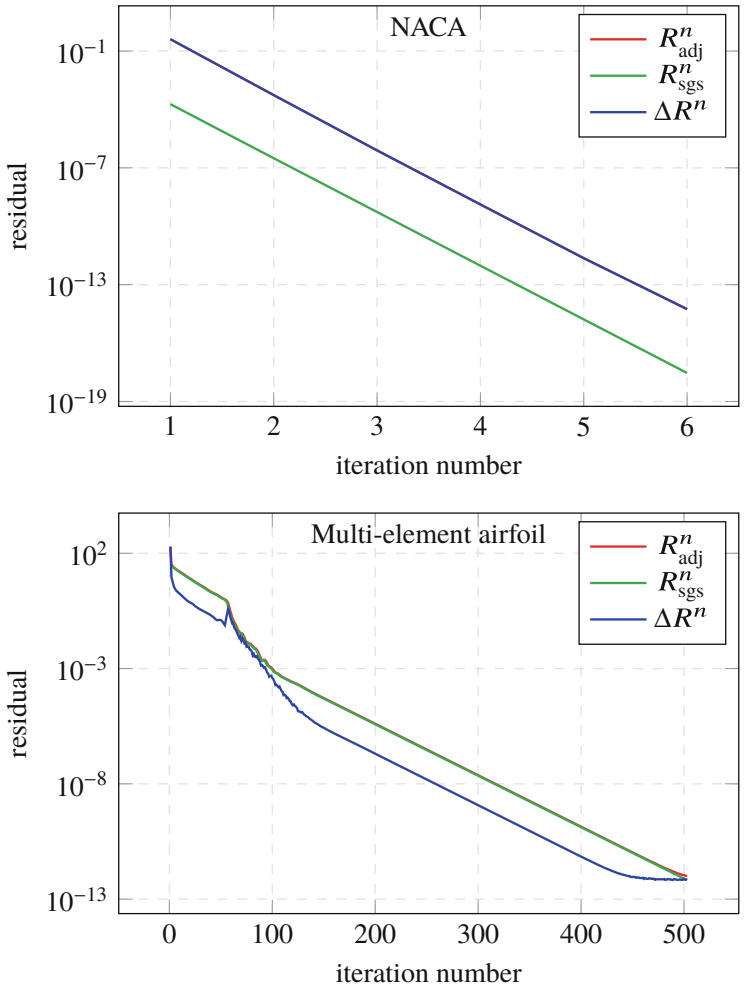


Fig. 3 Convergence of R^n_{adj}, R^n_{sgs}, and ΔR^n

defined the residual of the linear system (20) as

$$R^n_{\text{sgs}} = \left\| \frac{\partial J}{\partial \mathbf{W}} - \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \left[W^*\right]^n - \left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \right) \left[\delta W^*\right]^n \right\|, \tag{26}$$

and the norm of the difference between the residual vector relative to the adjoint system and the residual vector relative to the system (20) given by

$$\Delta R^n = \left\| \left(\frac{M}{\mathrm{d}t}\mathbb{1} + \frac{\partial \tilde{\mathbf{R}}}{\partial \mathbf{W}}^T \right) \left[\delta W^*\right]^n \right\|. \tag{27}$$

The convergence in the NACA case is extremely fast, with a reduction of the residual by two order of magnitude at each iteration, while it takes almost 500 iteration in the case of the multi-element airfoil. This can be related to the mesh size, which is widely different between the two benchmarks. The multi-element airfoil mesh, furthermore, has been adapted using a goal (lift) oriented mesh adaptation procedure. The *CFL* remains fixed to its maximal value, meaning that the SGS solver is able to solve efficiently the system (20) at each time step. Since we are not using an exact solver for the system (20), we remark that in general $\left[\delta W^*\right]^n \neq \left[W^*\right]^{n+1} - \left[W^*\right]^n$. Hence, the difference (27) highly depends on the *CFL*. In the NACA case the value of (27) overlaps on the line of R_{adj}, meaning that the time discretization is responsible of the main difference between R_{sgs} and R_{adj}. In the multi-element airfoil case, the term (27) is smaller than the residuals R_{sgs} and R_{adj}, which are comparable in magnitude. This means that the main source of error is given by the SGS solver: since, in general, we are able to control the SGS residual, for this reason we will keep it as small as possible (usually between 10^{-10} and 10^{-12}). Note that at the end of the simulation, the term (27) is almost stalling, meaning that we are adding the same quantity to the solution W^*. Referring to the multielement airfoil case, proceeding with the time advancement, the residual of the adjoint system stalls on a value around 4×10^{-13}, while with the GMRES, the adjoint system residual stalls on a value of 4×10^{-12}. Eventually, we remark that the results obtained with the GMRES and with the pseudo-transient adjoint are equivalent for both the cases.

5 Numerical Results

In this section, we assess the performance of the pseudo-transient adjoint solver by comparing it with the GMRES linear solver. To this aim, we set up 15 simulations of a multi element airfoil, using always the same parameters with the exception of the angle of attack α, which varies from $-2°$ to $26°$ with an interval length of $2°$. For all the simulations, we have used the compressible Navier-Stokes equations coupled to the Spalart-Allmaras model, the Mach number is set to $Ma = 0.2$ and the Reynolds number $Re = 5.0 \times 10^6$. Each simulation consists in a sequence of meshes of increasing complexity: we first start from a coarse mesh of 4000 vertices, then we adapt it using W_h and W^*, and we recompute the solution W_h and W^* on the new mesh, until the lift coefficient of the airfoil becomes stationary using a relative threshold of 1% among three consecutive values. After that, we double the complexity and we restart all the computations. The number of vertices increase continues until the mesh reaches a complexity of four million vertices: in this way we are able to support the solution independence with respect to the computational mesh. We perform the computations on a new mesh by restarting with the solutions W_h and W^* of the previous mesh, suitably interpolated [20]: this feature is particularly advantageous when employing the pseudo-transient adjoint,

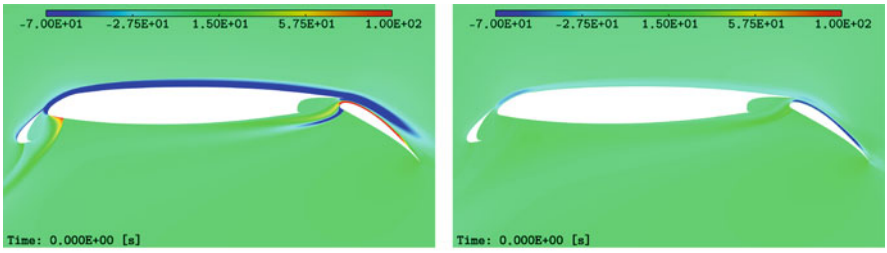
as we are able to stay close to the initial solution even when the problem is stiff. On the other hand, it seems that there is no advantage to restart the adjoint solution for the SGS-GMRES solver, and in this case W^* is set to zero before the resolution. Algorithm 3 depicts the described procedure.

Algorithm 3 Anisotropic mesh adaptation algorithm

Init: W_h^{old}, $W^{*,old}$, $\mathcal{H}$
for $N_V = 4000, 8000, \ldots, 4,096,000$ **do**
  $n = 0$
  while $(C_L^n - C_L^{n-1})/C_L^{n-1} > 0.01$ or $(C_L^{n-1} - C_L^{n-2})/C_L^{n-2} > 0.01$ **do**
    Compute W_h^{new}, $W^{*,new}$ starting from W_h^{old}, $W^{*,old}$
    Compute C_L^n from W_h^{new}
    Adapt $\mathcal{H}$ with N_V vertices using W_h^{new}, $W^{*,new}$
    Interpolate W_h^{new}, $W^{*,new}$ on $\mathcal{H}$
    $W_h^{old} \leftarrow W_h^{new}$, $W^{*,old} \leftarrow W^{*,new}$
    $n \leftarrow n + 1$
  end while
end for

We show the plots of the adjoint density fields for an angle of attack $\alpha = 8°$. We only show the lift-converged solutions at complexities of one, two, and four million vertices in Figs. 4, 5, and 6.

As we can see, the GMRES is able to reach a proper convergence for a complexity of one million vertices (it succeeds also for lower complexities), but it starts degrading the adjoint solution after this point. The pseudo-transient algorithm, on the contrary, is not able to reach a proper convergence (also for lower complexities), but it is able to preserve the solution structure thanks to the restarting from a previous solution. Furthermore, the pseudo-transient algorithm is more efficient in terms of the cpu time and storage, as it does not require to store hundreds of Krylov vectors. The behaviour for the other angles of attack is similar to the one shown for $\alpha = 8°$, meaning that after a certain complexity, usually around one to two million vertices, the GMRES degrades while the pseudo-transient



(a) (b)

Fig. 4 Comparison between the GMRES and the PSTR algorithm: adjoint density field for the multi-element airfoil with one million vertices. (**a**) GMRES, $R_{\text{adj}} = 3.79 \times 10^{-13}$, cpu time = 1 h 18 min 6 s. (**b**) PSTR, $R_{\text{adj}} = 7.18 \times 10^{-3}$, cpu time = 10 min 53 s

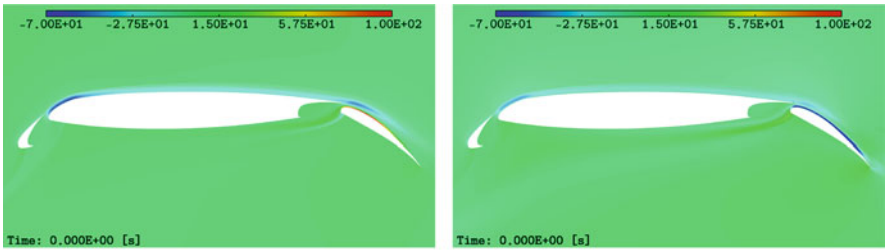


Fig. 5 Comparison between the GMRES and the PSTR algorithm: adjoint density field for the multi-element airfoil with two million vertices. (**a**) GMRES, $R_{\text{adj}} = 4.07 \times 10^{-2}$, cpu time = 1 h 22 min 57 s. (**b**) PSTR, $R_{\text{adj}} = 5.365 \times 10^{-3}$, cpu time = 24 min 49 s

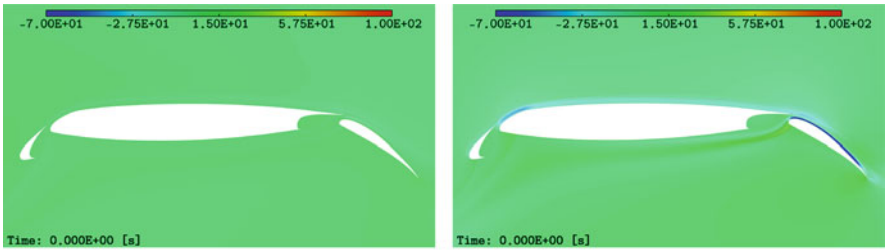


Fig. 6 Comparison between the GMRES and the PSTR algorithm: adjoint density field for the multi-element airfoil with four million vertices. (**a**) GMRES, $R_{\text{adj}} = 2.29 \times 10^{-2}$, cpu time = 2 h 29 min 13 s. (**b**) PSTR, $R_{\text{adj}} = 4.08 \times 10^{-3}$, cpu time = 53 min 21 s

algorithm preserves the solution. Since the adjoint density fields obtained with the GMRES and the pseudo-transient algorithm are structurally different, we perform a crossed simulation by running a pseudo-transient algorithm on a mesh with one million vertices obtained by using the GMRES from the beginning, and using the corresponding converged GMRES solution as its initial value. This check is done in order to verify whether the pseudo-transient algorithm is able to keep the solution in the same basin of attraction of the previous valid solution. The result is presented in Fig. 7, together with the original result obtained with a psuedo-transient adjoint algorithm from the beginning, for a mesh complexity of one million vertices.

We note that the pseudo-transient adjoint restarted on a GMRES converged solution still stalls, even on a lower residual, but it is able to preserve the structure of the converged solution obtained in the previous iteration.

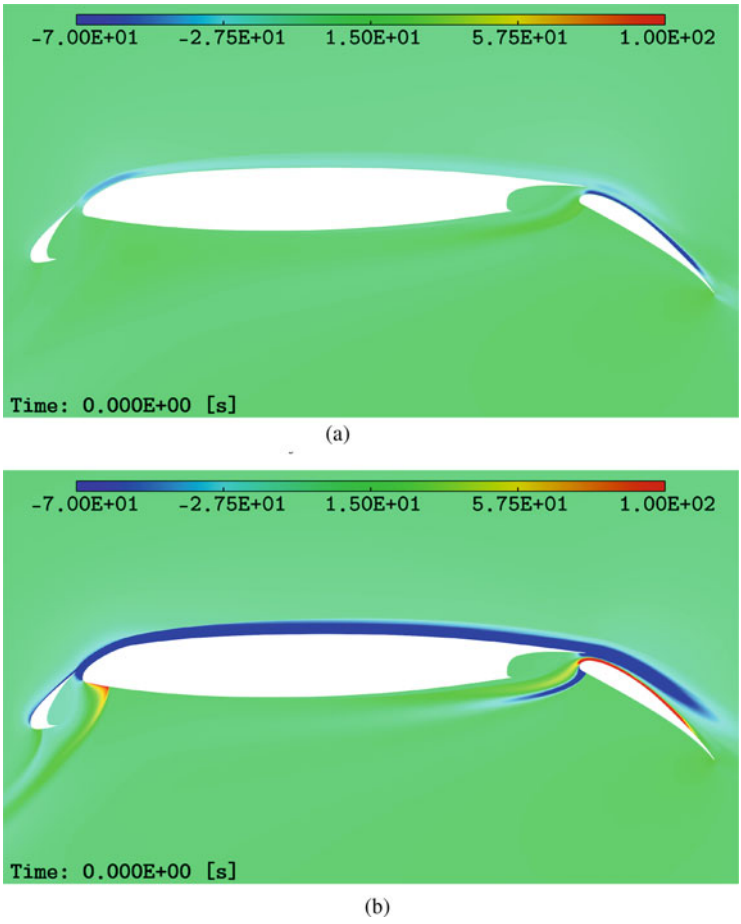


Fig. 7 Comparison between a full pseudo-transient adjoint simulation and a restarted simulation. (**a**) PSTR result, $R_{\text{adj}} = 7.18 \times 10^{-3}$, cpu time = 10 min 53 s. (**b**) PSTR restarted on a converged solution obtained with GMRES, $R_{\text{adj}} = 1.49 \times 10^{-3}$, cpu time = 9 min 45 s

6 Conclusions

A pseudo-transient algorithm for the resolution of the adjoint system in the context of goal-oriented mesh adaptation for RANS has been presented. It turned out to be more efficient in terms of cpu time and storage with respect to the GMRES algorithm, but less robust to converge to the machine zero. Anyway, it revealed to be more accurate on meshes with a high number of vertices, preserving the structure of the adjoint, in contrast to the GMRES, which starts to stall and to degradate the solution after a certain complexity.

References

1. Venditti, D.A., Darmofal, D.L.: Grid adaptation for functional outputs: application to two-dimensional inviscid flows. J. Comput. Phys. **176**, 40–69 (2002)
2. Loseille, A., Dervieux, A., Alauzet, F.: Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. J. Comput. Phys. **229**, 2866–2897 (2010)
3. Menier, V.: Mesh adaptation for the high fidelity prediction of viscous phenomena and their interactions. application to aeronautics. PhD Thesis, Université Pierre et Marie Curie, Paris (2015)
4. Alauzet, F., Loseille, A.: A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. Comput.-Aided Des. **72**, 13–39 (2016)
5. Frazza, L.: 3D anisotropic mesh adaptation for Reynolds averaged Navier-Stokes simulations. PhD Thesis, Sorbonne Université UPMC, Paris (2018)
6. Spalart, P.R., Allmaras, S. R.: A one-equation turbulence model for aerodynamic flows. In: 30th AIAA Aerospace Sciences Meeting and Exhibit, Reno (1992)
7. Cournède, P.H., Koobus, B., Dervieux, A.: Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids. Eur. J. Comput. Mech. **15**, 767–798 (2006)
8. Debiez, C. Dervieux, A.: Mixed Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations. Comput. Fluids **29**, 89–118 (2000)
9. Batten, P., Clarke, N., Lambert, C., Causon, D.M.: On the choice of wavespeeds for the HLLC Riemann solver. SIAM J. Sci. Comput. **18**, 1553–1570 (1997)
10. Van Leer, B.: Towards the ultimate conservative difference scheme I. The quest of monotonicity. In: Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics. Lecture notes in physics, pp. 18–163. Springer, Berlin (1972)
11. Jameson, A., Yoon, S.: Lower-Upper implicit schemes with multiple grids for the Euler equations. AIAA J. **25**, 929–935 (1987)
12. Sharov, D., Luo, H., Baum, J.D., Löhner., R.: Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers. AIAA Paper, vol. 2000-0927 (2000)
13. Alauzet, F., Loseille, A.: High order sonic boom modeling by adaptive methods. J. Comput. Phys. **229**, 561–593 (2010)
14. Menier, V., Loseille, A., Alauzet, F.: CFD validation and adaptivity for viscous flow simulations. 44th AIAA Fluid Dynamics Conference, AIAA Paper, Atlanta (2014)
15. Loseille, A., Löhner, A.: Cavity-based operators for mesh adaptation. In: 51th AIAA Aerospace Sciences Meeting, AIAA Paper, vol. 2013-0152 (2013)
16. Saad, Y., Schultz, H.M.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
17. Pini, M.: Turbomachinery design optimization using adjoint method and accurate equations of state. PhD Thesis, Politecnico di Milano, Milan (2013)
18. Giles, M., Duta, M., Müller, J.D., Pierce, N.: Algorithm developments for discrete adjoint methods. AIAA J. **41**, 198–205 (2003)
19. Nielsen, E., Lu, J., Park, M., Darmofal, D.: An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids. Comput. Fluids **33**, 1131–1155 (2004)
20. Alauzet, F., Mehrenberger, M.: $P1$-conservative solution interpolation on unstructured triangular meshes. Int. J. Numer. Meth. Eng. **84**(13), 1552–1588 (2010)

Moving Deforming Mesh Generation Based on the Quasi-Isometric Functional

Vladimir A. Garanzha and Liudmila Kudryavtseva

Abstract We suggest an algorithm which allows for generation of a moving adaptive mesh with a fixed topology according to the time-dependent control metric in the computational domain using a quasi-isometric mesh quality functional. For each time step, we use the preconditioned gradient search technique for the mesh quality functional in order to compute large displacements of each mesh vertex. Intermediate meshes interpolating between the initial and the displaced states are nonsingular deformations of the initial mesh and can be used for numerical simulations with small time steps, which greatly improves the efficiency of the remeshing algorithm.

1 Introduction

The equidistribution principle [6] is a basic component of moving adapting mesh algorithms [3, 11]. Its idea is to compute the distribution of the mesh size using information from a certain PDE or from the estimate of the interpolation error. In 1d, the equidistribution principle results in the solution of a second order linear elliptic equation with a scalar control function. Note that for time-dependent problems one generally has to use adaptive filters to suppress moving mesh instabilities which were described in the seminal paper by Coyle, Flaherty, and Ludwig, 1986 [5].

V. A. Garanzha (✉)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia
e-mail: garan@ccas.ru
http://www.ccas.ru/gridgen/lab

L. Kudryavtseva
Moscow Institute of Physics and Technology, Moscow, Russia

Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_10

The generalization to multiple dimensions still remains controversial. A widely used generalization is based on the solution of the Laplacian-like equations for the mesh coordinates using a scalar control function [15, 17].

In the beginning of 1990s, S.K. Godunov formulated the following basic principles of optimal variational meshing:

- a mesh should be quasi-isometric deformation by itself and converge to a certain quasi-isometric mapping upon refinement;
- a quasi-isometric mapping should be a unique and stable solution of a variational problem;
- the variational problem should use distortion measures based on the principal invariants of the deformation metric tensor;
- the discretized variational problem should also have a unique and stable solution;

Note that these principles are not reachable in theory and practice of mesh generation.

By definition, the ratio of the length of any simple rectifiable curve to the length of its image under a quasi-isometric mapping is bounded from above by a certain constant $K > 1$ and below by $1/K$.

The implementation of these ideas was first presented in [10] using the conformal mapping technique in a relatively simple 2d setting. These ideas inspired a series of papers including [7, 9], where the variational principle for a construction of multi-dimensional quasi-isometric mappings was suggested and justified theoretically and numerically.

In this paper, we suggest a new algorithm which allows for generation of a moving adaptive mesh with a fixed topology using a time-dependent control metric in the computational domain. We are focused on the methods for constructing the control metric and meshes allowing to improve the accuracy of the immersed boundary conditions (IBC) flow solver [1] for a system of moving bodies defined by the distance-like implicit function.

Each cell of the ideal mesh at a given time after local transformation into uniform coordinates related to the metric should be congruent to the same cell at the initial time level. The quasi-isometric functional [7] is used to measure and control the matching between the real and the ideal mesh. We have found that, when global large deformations of the initial mesh are necessary in order to satisfy the mesh compression criteria inside thin moving layers near boundaries of domains, simple explicit mesh optimization methods fail to follow the metric precisely. Unfortunately, the preconditioned gradient search algorithm for a quasi-isometric functional is quite expensive and, thus, coupling it directly with the flow solver is highly inefficient. Another problem is that, due to the non-linearity of the Euler-Lagrange equations, we cannot assume that the norm of the functional residual is reduced to zero at each time step. Thus, the time continuity of the moving mesh is not guaranteed because the iterative functional minimization with the changed metric may result in a considerable displacements even for infinitely small time steps and the space-time trajectories of the mesh cells may become extremely inclined. In order to solve this problem, we suggest a very simple

and efficient algorithm which is based on the direct mesh interpolation. Using the ideas of the preconditioned gradient search algorithm from [8] we compute the predictor, which defines the minimization direction (displacement field) for each mesh vertex for a large time increment. Every intermediate mesh computed with the help of this displacement is guaranteed to be correct. We have found that the length of the computed displacement field is very large compared to the displacements allowed by the flow solver [1], hence the number of costly implicit minimizations can be greatly reduced. Assuming that the time dependence of the metric is defined by a smooth function, we obtain a mesh deformation/adaptation algorithm which requires onelinear solve per, say, 5–10 time steps, making it a quite efficient component of the moving mesh flow solver. We have found that the linear solver based on the so-called second order incomplete Cholesky factorization (IC2) [13, 14] is a very efficient component of the nonlinear solver, especially for the parallel version of the algorithm. Our experience with simpler algorithms, such as the Conjugate Gradient technique with the standard incomplete Cholesky factorization IC(k) as a preconditioner was quite disappointing.

2 Quasi-Isometric Functional

The quasi-isometric functional was suggested in [7, 9]. Here, we use a simplified version of this functional for a controlled mesh deformation.

Let $\xi_1, \xi_2, \ldots, \xi_d$ denote the Lagrangian coordinates associated with an elastic material and $x_1, x_2, \ldots, x_d$ denote the Eulerian coordinates of a material point. The spatial mapping $x(\xi, t)\colon \mathbb{R}^d \to \mathbb{R}^d$ depending on the parameter t defines a time-dependent elastic deformation. Here, the parameter t denotes the time and ξ_i are associated the domain with the initial mesh. Namely, ξ_i are frozen into cells of the initial mesh, while the Eulerian coordinates are fixed Cartesian coordinates in the computational domain.

The Jacobian matrix of the mapping $x(\xi, \cdot)$ is denoted by C, where $c_{ij} = \partial x_i / \partial \xi_j$.

Let $G_\xi(\xi, t)$ and $G_x(x, t)$ denote the metric tensors defining linear elements and lengths of curves in Lagrangian and Eulerian coordinates in the domains Ω_ξ and Ω_x, respectively. Then, $x(\xi, t)$ is the mapping between the metric manifolds M_ξ and M_x.

Let us define the following polyconvex elastic potential (internal energy) which serves as the distortion measure and is written as a weighted sum of the shape and the volume distortion measures [7]:

$$W(C) = (1-\theta)\frac{\left(\frac{1}{d}\mathrm{tr}(C^T C)\right)}{\det C^{2/d}} + \frac{1}{2}\theta\left(\frac{1}{\det C} + \det C\right). \tag{1}$$

Evidently,

$$W(U) = 1, \quad U^T U = I, \quad \text{and} \quad \det U = 1 \tag{2}$$

for a arbitrary orientation-preserving orthogonal matrix U. As suggested in [7], we set $\theta = 4/5$. This value provides a reasonable balance between the shape and the volume distortions.

We are looking for the mesh deformation $x(\xi, t)$ being the solution of the variational problem

$$F(x(\xi, t)) = \int_{\Omega_\xi} W\Big(Q(x, t)\nabla_\xi x(\xi, t)H(\xi)^{-1}\Big) \det H \, \mathrm{d}\xi \, . \tag{3}$$

In the current version of the mesh deformation algorithm, we do not consider the variation of functional (3) with respect to the time t.

Matrices $H(\xi)$ and $Q(x, t)$ are certain matrix factorizations of the metric tensors G_ξ and G_x, respectively, defined by

$$H^T H = G_\xi, \quad \det H > 0, \quad Q^T Q = G_x, \quad \det Q > 0.$$

We assume that the singular values of the matrices Q and H are uniformly bounded from below and above.

The time-dependent mesh deformation is introduced via a time-dependent metric tensor $G_x(x, t) = Q^T(x, t)Q(x, t)$. It follows from (2) that the absolute minimum of the functional (3) is equal to

$$\mathrm{vol}\, \Omega_\xi = \int_{\Omega_\xi} \det H \, \mathrm{d}\xi$$

and is attained for $Q(x, t)\nabla_\xi x(\xi, t)H(\xi)^{-1} = U$, where U is an orthogonal matrix. It means that, in the uniform coordinate frame $y = Qx$ associated with the local value of the metric $G_x(x, t)$, the mapping $x(x, t)$ is locally isometric to the mapping $x(\xi, 0)$ when $H(\xi) = \nabla_\xi x(\xi, 0)$.

Suppose that the domain Ω_ξ can be partitioned into convex polyhedra D_k. We construct a continuous piecewise-smooth deformation $x_h(\xi, \cdot)$, which is regular on each polyhedron. In practice, we use linear and poly-linear finite elements in order to assemble the global deformation. We call the integral $F(x_h(\xi), t)$ over this deformation a semi-discrete functional.

In order to approximate the integral over a convex cell D_k, certain quadrature rules are required. As a result, the semi-discrete functional is replaced by the discrete functional:

$$F(x_h(\xi,t),t) \approx \sum_k \mathrm{vol}(U_k) \sum_{q=1}^{N_k} \beta_q W(C_q) = F^h(x_h(\xi,t),t).$$

Here, N_k is the number of the quadrature nodes per cell D_k, C_q denotes the Jacobian matrix in the q-th quadrature node of U_k, and β_q are the quadrature weights (see [9] for details). We use only such quadrature rules which guarantee the majorization property

$$F(x_h(\xi,t),t) \le F^h(x_h(\xi,t),t). \tag{4}$$

This property can be used to prove that all intermediate deformations $x_h(\xi,t)$ providing the finite values of the discrete functional are homeomorphisms [9].

In order to derive the discrete mesh functional, special quadrature rules are applied which guarantee the global invertibility of the deformation mapping for finite values of the discrete functional [9].

3 Preconditioned Minimization Algorithm and the Moving Mesh Interpolation Strategy

It is convenient to write our discrete functional as a function $F(Z,Y,t)$ with the spatial argument $Z^T = (z_1^T\ z_2^T\ \dots\ z_{n_v}^T)$ where $z_k \in \mathbb{R}^d$, $k = 1,\dots,n_v$, are the mesh vertex positions. Dependence on the time t is introduced via the time-dependent metric $G_x(y,t)$. The vector Y corresponds to the argument y of the metric G_x.

The Hessian matrix $\tilde{H}$ of the function F with respect to Z is built of $d \times d$ blocks

$$\tilde{H}_{ij} = \frac{\partial^2 F}{\partial z_i \partial z_j^T}.$$

Here, the matrix $\tilde{H}_{ij}$ is placed on the intersection of i-th block row and j-th block column. We filter the Hessian matrix during the finite element assembly procedure to make it positive definite and to reduce number of nonzero elements by a factor of 2 [8]. Below, we use the same notation $\tilde{H}$ for the filtered Hessian matrix. The gradient of F with respect to Z is denoted by R. This vector consists of d-sized subvectors r_i. It is the approximate gradient of the functional since we do not differentiate the metric G_x, hence, the dependence on Y is not taken into account.

Since solving the variational problem on the time level $t^{n+1} = t^n + \Delta t^n$ exactly is too expensive, we are using 1- or 2-step heuristic schemes.

One step of the Newton-type method for finding a function stationary point can be written as

$$\sum_{j=1}^{n_v} \tilde{H}_{ij}(Z^n, Z^n, t^{n+1})\delta z_j + r_i(Z^n, Z^n, t^{n+1}) = 0. \tag{5}$$

This set of equalities can be written in brief as

$$\tilde{H}(Z^n, Z^n, t^{n+1})\delta Z + R(Z^n, Z^n, t^{n+1}) = 0 \tag{6}$$

and

$$\tilde{Z} = Z^n + \tau_n \delta Z. \tag{7}$$

Here, the parameter τ_n is found as a approximate solution of the one-dimensional problem

$$\tau_n = \arg\min_{\tau} F(Z^n + \tau\delta Z, Z^n, t^{n+1}). \tag{8}$$

The standard golden-section search is used to find the approximate minimum.

The preconditioned conjugate gradient technique is used for an approximate solution of (6) with an approximate second order Cholesky factorization [13] as a preconditioner. For a parallel version of a solver we use the additive version of an approximate second order Cholesky factorization based on domain decomposition with overlaps [14]. The resulting nonlinear scheme was found to be very stable and quite efficient.

One can try to simplify the structure of the Hessian matrix by setting $\tilde{H}_{ij} = 0$ when $i \neq j$ [12] or even by considering only the diagonal part of Hessian matrix as a preconditioner. We have found that if global large deformations of the initial mesh are necessary in order to satisfy the mesh compression criteria inside thin moving layers, simple explicit methods of mesh optimization fail to follow the metric precisely. Our observation is that "explicit" solvers are not efficient in the case of global deformations. Our linear solver is based on the extraction of d independent linear systems with symmetric positive definite $n_v \times n_v$ matrices [8] from the linear system (6), which are closely resembling finite element approximations of the scalar Poisson equations with a tensor coefficients.

The simplest 1-step algorithm is formulated as

$$Z^{n+1} = Z^n + \min(1/2,\ \tau_n)\delta Z. \tag{9}$$

The observed paradox of the variational method is that we may assign any reasonable constant K in the range $(0, 1/2)$ to τ_n and, after a short transient period, the algorithm would eventually scale the increment δZ in such a way, that the value $\tau_n|\delta Z|$ is almost invariant since, otherwise, the mesh layer would lag behind the

prescribed position. Setting $K = 1/2$ was found to be a good compromise between the accuracy and the stability. In order to improve the accuracy the following two-step algorithm was tested:

- compute

$$Z^{n+\frac{1}{2}} = Z^n + \frac{1}{2}\tau_n \delta Z, \tag{10}$$

- find the new increment $\delta\tilde{Z}$ such that

$$\tilde{H}(Z^{n+\frac{1}{2}}, Z^{n+\frac{1}{2}}, t^{n+1})\delta Z + R(Z^{n+\frac{1}{2}}, Z^{n+\frac{1}{2}}, t^{n+1}) = 0, \tag{11}$$

- assign

$$Z^{n+1} = Z^n + \tau_n \delta\tilde{Z}. \tag{12}$$

In order to ensure the stability of the algorithm in case when resulting Z^{n+1} is not an admissible solution, one should find the new τ_n via

$$\tau_n = \arg\min_{\tau} F(Z^n + \tau\delta\tilde{Z}, Z^{n+\frac{1}{2}}, t^{n+1}). \tag{13}$$

and repeat the steps (10)–(12) with the new τ_n which by construction should be smaller than the previous one. The resulting algorithm resembles the standard 2-stage Runge–Kutta scheme associated with a single time step along the space-time trajectory. More precisely, Runge–Kutta predictor stage looks like the minimization step of the Newton-type algorithm, while Runge–Kutta corrector stage may serve as an additional iterative step. Resulting scheme would not provide quadratic convergence even in the model cases. However we have found that this 2-step scheme sharply improves the accuracy of the localization of the mesh layers. Moreover, it produces smooth space-time trajectories of cells and allows to run the mesh deformation solver with much larger time steps. From the minimization strategy it follows that the function F has finite values for any vector Z_α defined by

$$Z_\alpha = \alpha Z^n + (1-\alpha)Z^{n+1}, \quad 0 \neq \alpha \neq 1,$$

which, in turn, means that all intermediate mesh deformations are non-degenerate.

4 Mesh Stretching and Size Gradation Control

System of single or multiple moving and deforming bodies which are denoted as a domain $B \in \mathbb{R}^d$ defined by implicit function $d_s(x, t)$ in such a way that the function d_s is negative inside body, positive outside it, and isosurface $d_s(x, t) = 0$ defines the

boundary ∂B at the time t. We assume that $d_s(x,t)$ resembles the signed distance function for the instant domain boundary. In theory, one can assume the existence of the quasi-isometric mapping $x(y)\colon \mathbb{R}^d \to \mathbb{R}^d$ such that the function $d_s(x(y),t)$ is precisely the signed distance function. In practice, we assume that when the vector $\nabla_x d_s$ in the vicinity of the boundary is defined and its norm is bounded from below and above by certain global constants.

The metric tensor $G(x,t)$ is defined as a function of $d_s(x,t)$ such that it attains its largest value on the domain boundary and decreases inside and outside body (using different laws). The Mmesh inside the body is in general quite coarse since the immersed boundary solver solution inside the body does not have a physical meaning.

In order to attain a smooth mesh size gradation, we use a logarithmic auxiliary mapping which allows to attain an almost constant mesh size growth factor. Consider the one-dimensional auxiliary mapping $y(\xi)\colon [0,1] \to [0,1]$. We define a uniform mesh in ξ coordinates with the mesh size $h = 1/N$, where N is the number of cells. Vertices y_i are distributed assuming a constant growth rate

$$y_{i+1} - y_i = (y_i - y_{i-1})(1+\varepsilon).$$

This equality can be considered as the finite difference approximation of the equation

$$h\ddot{y} = \varepsilon \dot{y}.$$

Interchanging dependent and independent variables we obtain

$$-\ddot{\xi} h = \varepsilon \dot{\xi}^2.$$

The solution of this ODE is defined by

$$\xi = \phi(y) = \frac{h}{\varepsilon} \ln\left(1 + \frac{A\varepsilon}{h}(y-\delta)\right) + \delta A.$$

We split the "computational domain" into three subregions: "boundary layer" $0 <= y <= \delta$, transition zone $\delta \le y \le D < 1$, and the outer zone $D \le y \le 1$. Since we apply mesh adaptation for better implementation of immersed boundary method for moving bodies, inside the boundary layer we simply assign the constant maximal mesh compression coefficient A. For the outer zone, we also imply a linear distribution with the constant compression coefficient $\kappa = (1-D)/(1-\phi(D)) \le 2$. The function ϕ controls the transition between the smallest and the largest scales. Figure 1, left, shows the overall result (note that the transposed graph is shown).

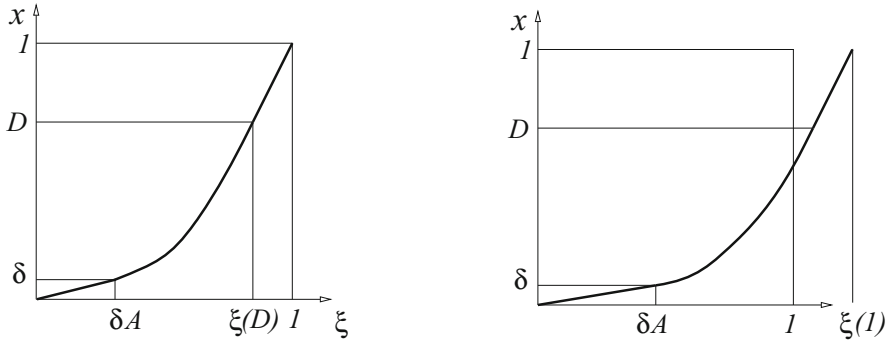


Fig. 1 Left: the auxiliary one-dimensional mapping defining the stretching. Right: the target mesh compression ratio A is too large and should be reduced

The derivatives of this mapping are defined by

$$\gamma(y, \delta, A) = \dot{\phi} = \frac{1}{\frac{1}{A} + \frac{\varepsilon}{h}(y - \delta)}, \qquad \ddot{\phi} = -\frac{\varepsilon}{h}\frac{1}{\left(\frac{1}{A} + \frac{\varepsilon}{h}(y - \delta)\right)^2}.$$

The mesh compression coefficient is defined by the function $\dot{\phi}$. Its graph is a hyperbola.

The set of control parameters δ, A, κ, h can be contradictory since for too large values of the compression factor A one may get $\phi^{-1}(1) > 1$, or even $\phi^{-1}(D) > 1$. In this case, we iteratively reduce the value of A with $\kappa = 2$ until the equality $\phi^{-1(1)} = 1$ is satisfied.

In order to compute the dimensionless control parameters of the function ϕ, we use the radius $R_{\max}$ of the influence zone of the body, the true mesh layer thickness δ_R, and the average mesh size l_R in the direction normal to the body.

Then,

$$h = \frac{3}{2}\frac{l_R}{R_{\max}}, \quad \delta = \max\left(\frac{\delta_R}{R_{\max}}, \frac{h}{A}\right).$$

Suppose that the body is a zero isosurface of the signed distance function $d_s(x, t)$. We compute the normal compression function as follows

$$\sigma_1(x, t) = \gamma(|d_s(x, t)|/R_{\max},\ \delta,\ A_n). \tag{14}$$

where A_n is the compression factor in the normal direction to the boundary.

Denote by $u_1 = \nabla d_s/|\nabla d_s|$ a unit vector of the normal direction. We can obtain a full orthonormal basis $U = (u_1, \ldots, u_d)$ using vectors $u_2, \ldots, u_d$ defining the local tangent basis on the isosurface of d_s. The metric tensor $G_x(x, t)$ can be defined by

$$G_x = U\Sigma U^T, \quad \Sigma = \mathrm{diag}(\sigma_i). \tag{15}$$

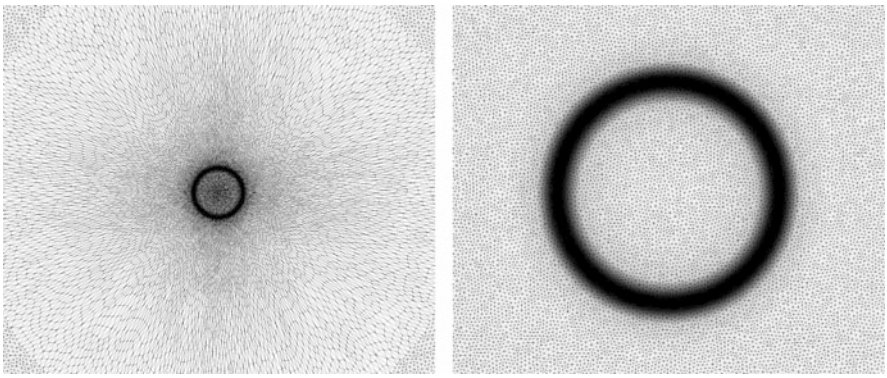


Fig. 2 Isotropic metric: an internal layer in the computational domain and its preimage in the parametric domain

The choice of the "tangential" stretches, namely, the stretches in the directions orthogonal to the gradient of the implicit function, is not trivial and very important in order to obtain a mesh deformation without size jumps and ruptures. Here, "rupture" means appearance of long and skewed mesh cells which resembles an approximation of an elastic material rupture.

The isotropic choice $\sigma_i = \sigma_1$ or $G_x = \sigma_1^2 I$ seems to be the simplest one. Unfortunately, its applicability is very limited, since the preimage of the boundary layer in the initial coordinates ξ is scaled by a factor A_n, as shown in Fig. 2 for $A_n = 6$. An enlarged preimage may cover the whole initial domain or even get outside of the boundary meaning that most of the mesh cells will travel into the boundary layer making the resulting mesh unusable. The mesh in Fig. 2 contains 22 531 vertices and 44 610 triangles, however, the enlargement is almost mesh independent.

If we specify the maximal tangential compression A_t, then the linear size of the preimage would increase by the factor A_t, while the thickness of the preimage layer will increase by the factor A_n, as shown in Fig. 3 for $A_n = 6$, $A_t = 2$, and the same initial mesh.

We assign the maximal anisotropy ratio A_n/A_t in the boundary layer and build σ_i to gradually reduce the anisotropy away from the layer. In some cases, the constant factor A_t does not allow to resolve the boundary features, in particular if sharp or highly curved boundary fragments are present. The general argument is that when the tangential resolution is not enough, one should use smart values of the tangential stretches $A^*_{t\,i}$, $i = 2, \ldots, d$ in different directions in the range $A_t \le A^*_{t\,i} \ge A_n$. Let us denote the metric with constant stretches in the boundary layer by G_1, and consider the metric introduced around surface features by G_2. We use the same representation (14) and (15) for G_2, the main difference being the influence radius

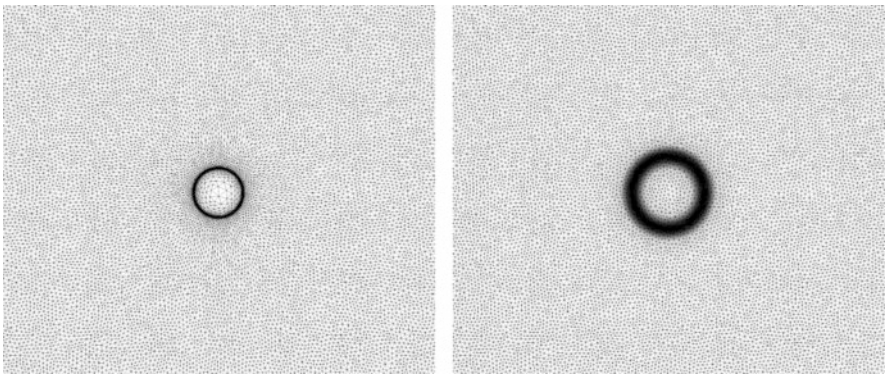


Fig. 3 Anisotropic metric: an internal layer in the computational domain and its preimage in the parametric domain

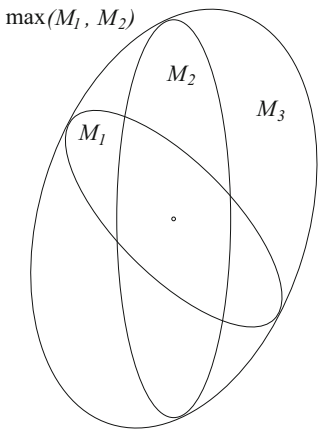


Fig. 4 Illustration of the maximum operation for two metrics

R_2, which should be smaller compared to the global influence radius $R_{\max}$ for G_1. Then final metric is defined by

$$G_x = G_3 = \max(G_1, G_2).$$

The maximum operation is based on the common circum-ellipsoid construction which is close to the one suggested in [4] (see Fig. 4). Consider two concentric ellipsoids M_1 and M_2 defined by the quadratic forms $x^T G_1^{-1} x = 1$ and $x^T G_2^{-1} x = 1$, respectively. The common circum-ellipsoid M_3 defines the matrix G_3. Construction of the ellipsoid M_3 is simple: find an affine map which transforms one of the ellipsoids into a sphere. In transformed coordinates, the circum-ellipsoid is trivially constructed and mapped back into the original coordinates. Since each of the

metrics is defined by its spectral decomposition, this construction is reduced to a number of products of orthogonal and diagonal matrices.

5 MPI-Based Parallel Implementation

We use the spatial mesh decomposition in order to build a parallel algorithm. Since the degrees of freedom defining the mesh deformation are the mesh vertices, we build a consistent partitioning of mesh cells and vertices: the parametric domain $\Omega_x i$ is split into connected subdomains consisting of full mesh cells. Mesh vertices belonging to boundaries between subdomains are distributed between subdomains. We use the parallel ILU2-based iterative solver [13, 14] to compute the minimization direction. The solver input data are the right hand side (partitioned into blocks) and the sparse matrix (partitioned into block rows). Each block corresponds precisely to the partitioning of the mesh vertices. Our implementation of the iterative scheme is based on the extended subdomains defining the two-cell-wide subdomain overlap. At the beginning of each minimization iteration (7), we use data exchange to create the current guess at each of the extended subdomains. Such an extension makes the cell-by-cell assembly of functional, its gradient, the Hessian matrix and its subsequent double scaling completely local operations. Additional data exchanges in order to find the optimal value of τ by an approximate solution of the one-dimensional minimization problem (8) are not necessary, merely the global sums should be computed. All operations of the mesh generation algorithm are fully scalable. Hence, one can expect that the overall scalability is defined by that of the parallel linear solver. Note that the linear solver uses its own overlaps and a data exchange scheme [14].

6 Numerical Experiments

We apply the mesh deformation solver for the geometric adaptation of a computational mesh in order to improve the resolution of the NOISETTE flow solver [16] for numerical modelling of turbulent flow around a quadrocopter propeller. The flow solver is based on the Immersed Boundary Conditions (IBC). We test the numerical technology in two dimensions using the two-dimensional projection of the realistic propeller geometry [2] with the main goal to extend it to the case of a real three-dimensional propeller modeling. At this stage, we run simple three-dimensional tests in order to check the initial geometric adaptation, to compare the computed position of the mesh layer with the target moving position, as well as to check the efficiency and scalability of the three-dimensional algorithm. Figure 5 (left) shows the propeller geometry. Due to the natural geometrical limitations, the mesh is deformed only inside a circle which is 10% larger then the propeller itself, meaning that the mesh adaptation near the blade tip becomes a nontrivial task. The propeller

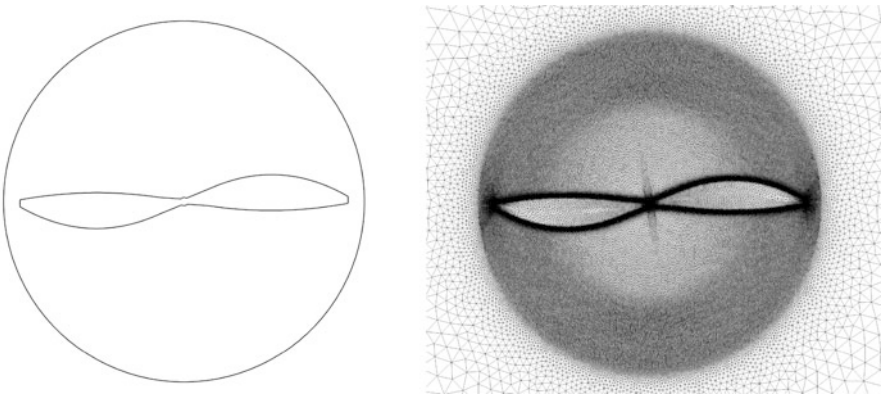


Fig. 5 A two-dimensional propeller model and the initial adapted mesh

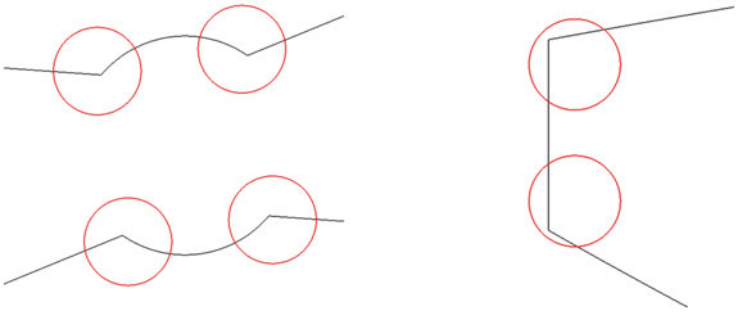


Fig. 6 Circles are the influence zones of the isotropic metric near sharp corners

is defined by an implicit signed distance-like function. We impose compression factor of 30 in the normal direction inside the thin layer near the blades. In order to initiate the time-dependent deformation, we start from an initial mesh adaptation to a fixed shape (Fig. 5, right).

In order to resolve the corners and the highly curved boundary fragment, we introduce locally isotropic metric influence zones, shown as circles in Fig. 6. The isotropic metric is defined by the same law in (14), where the distance to the circle center is considered and $R_{\max}$ is the circle radius. The coefficient A_n is the same as for the global metric and $\sigma_2 = \sigma_1$. The maximum operation for metrics is applied in order to compute G_x.

Figure 7 shows fragments of the initial mesh near the influence zones of the anisotropic metric.

Figure 8 shows the general mesh outline after the first and second propeller rotations.

Figures 9 and 10 show mesh fragments after the first and second rotations.

Computation of the long term mesh deformation demonstrates a behaviour which is close to the periodic one. Figure 11 show trajectories of selected cells in the

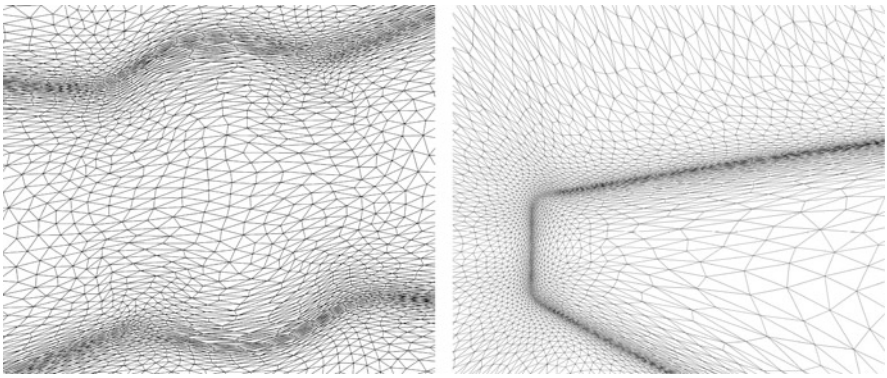


Fig. 7 Fragments of the initial adapted mesh near the sharp corners

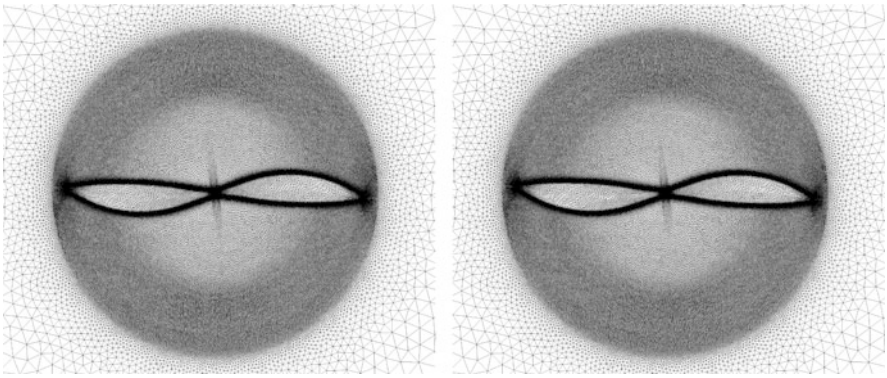


Fig. 8 Mesh after the first and second rotations

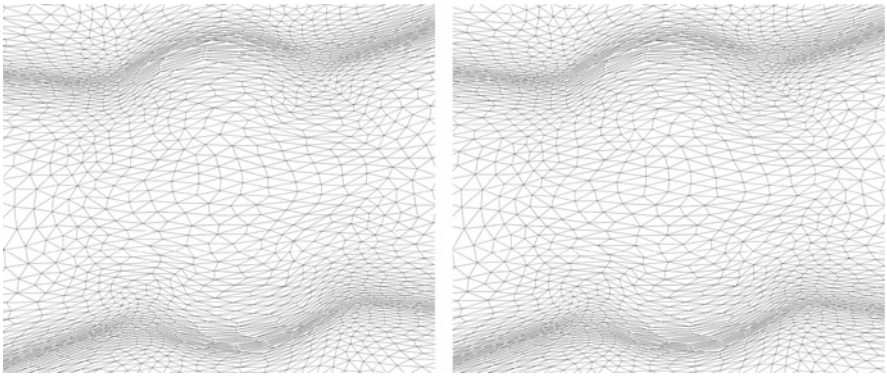


Fig. 9 Mesh fragment after the first and second rotations

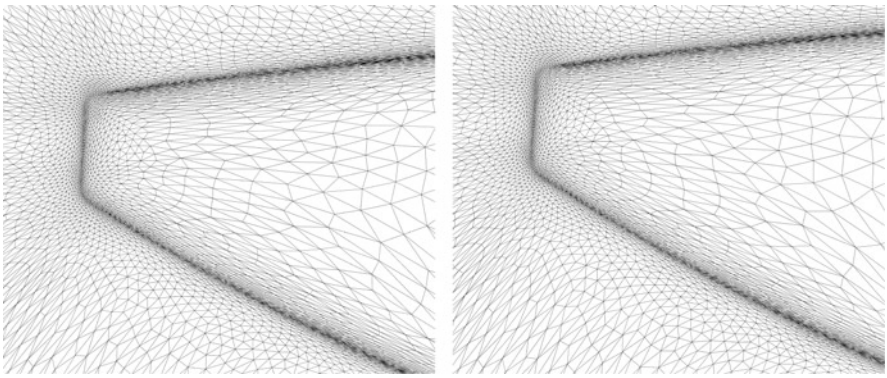


Fig. 10 Mesh fragment after the first and second rotations

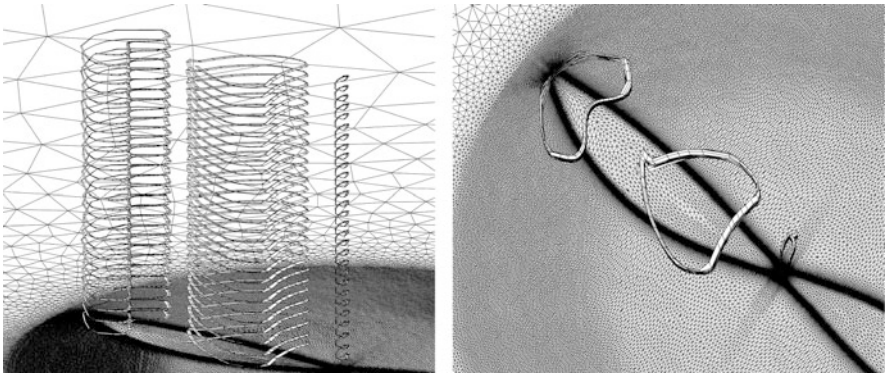


Fig. 11 Space-time trajectories of three selected triangles for 14 propeller rotations shown from different viewpoints

variables x_1, x_2, and t. We select several triangles, save their coordinates every 500 time steps and connect the consecutive positions creating space-time prisms. The resulting triangular beams illustrate the space-time deformation of the mesh.

Figure 12 shows the deformation of mesh subdomains when propeller spans a quarter of the rotation.

Note that we use a vertex-based domain decomposition, so mesh vertices lying on the boundaries of colored regions actually are assigned to the one or another subdomain.

Figure 13 shows the mesh deformation and the movement of the subdomain boundaries in the propeller-related reference frame. The number of subdomains is larger than in Fig. 12. Distinct one-cell-wide layers show the triangles with vertices belonging to different subdomains.

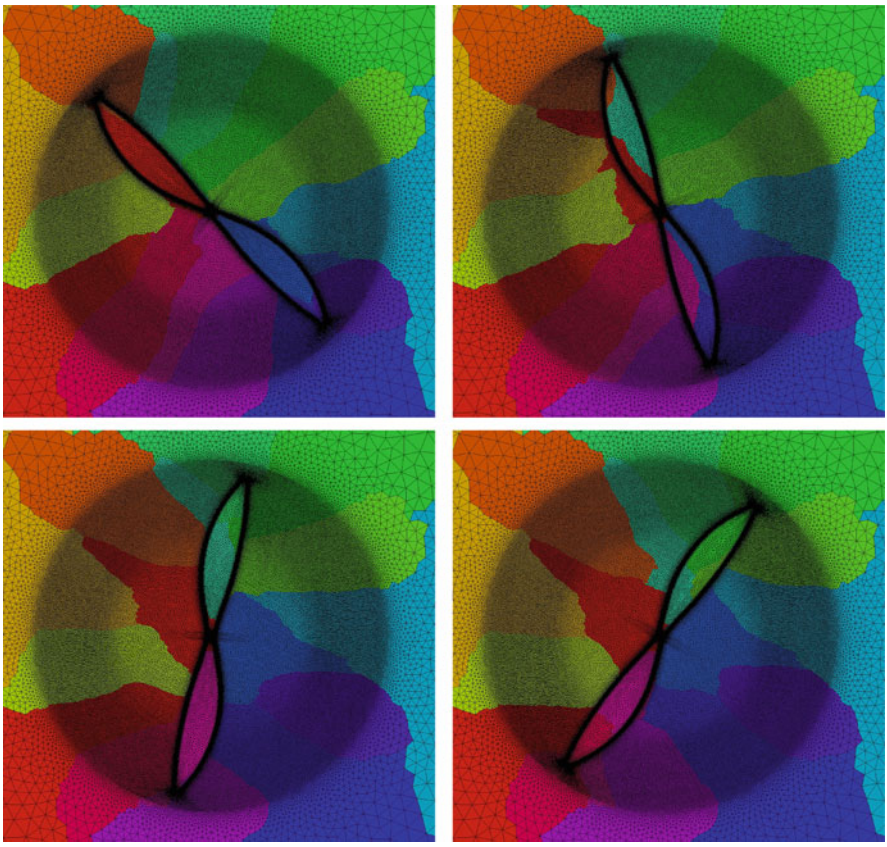


Fig. 12 Deformation of subdomains for a quarter of the rotation, subdomains are marked by distinct colors

Figure 14 shows the target σ_1 distribution in the rotating reference frame for two different time levels. Red color denotes the maximal value. One can observe that the position of the mesh layer follows the controls quite closely. A quite interesting effect is related to the bands of mesh cells which are attracted to the blade, then travel for some time along the boundary layer in the compressed state, and, eventually, leave the propeller. Another group of cells in the middle of the domain is attracted to the blades, cross them, and go out at the other side. Figure 14 (right) is slightly inclined in the coordinates x_1, x_2, and t to make the cell trajectories more visible.

In order to evaluate the scalability of the parallel implementation, we run several two and three-dimensional test cases. In 2d, we consider a "small" problem with 357,781 vertices and 713,760 triangles and a "moderate" problem with 1,429,321 vertices and 2,855,040 triangles. In 3d, we consider a deformation of a tetrahedral mesh with 9,586,347 vertices and 56,715,408 tetrahedra.

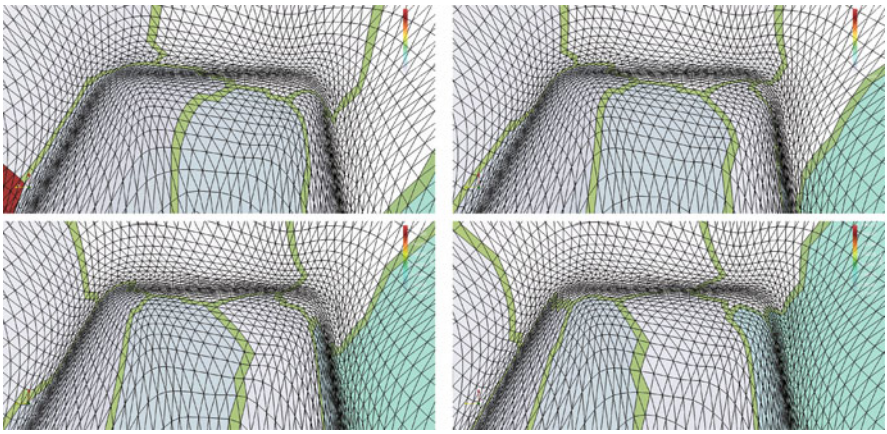


Fig. 13 Mesh deformation and movement of subdomains in the rotating reference frame

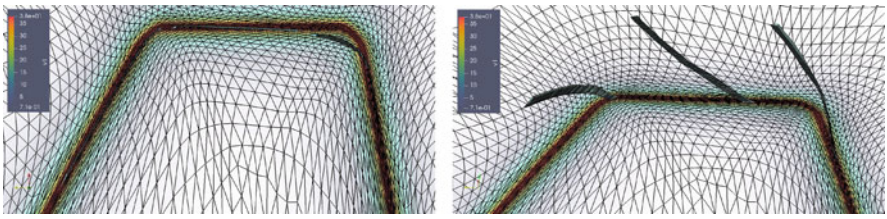


Fig. 14 Mesh layer localization compared to σ_1 distribution and cell trajectories in the rotating reference frame

The scalability experiments were ran on the parallel cluster of the Moscow Institute of Physics and Technology. It is an Intel CPU-based cluster having 24 cores per board and the Infiniband interconnect. Figure 15 illustrates the scalability of the mesh deformation algorithm. A separate graph is devoted to the linear solver. As one can expect, the overall scalability is defined by the linear solver. Note that the scalability with respect to a single core is not impressive, while results scale very well with respect to 24 cores. We were not able to explain this observation. In principle, it may be a drawback of the algorithm or a cluster software/hardware misconfiguration artefact.

Figure 16a shows the speed-up for a small 2d problem where the saturation is quite pronounced for more then 120 cores, while for the 3d case in Fig. 16b the scalability is quite reasonable and the saturation is not observed until 600 cores. Note that a superacceleration is observed which can be attributed to the lower performance of the 3d code for a small number of vertices. We plan to run the 3d algorithm on larger configurations.

Figure 17 illustrates subdomains for the 3d problem in the case of 144 cores. We show the trace of the subdomains on the boundary, a cross-section of the initial uniform mesh, and a cross-section of the mesh adapted to a moving sphere.

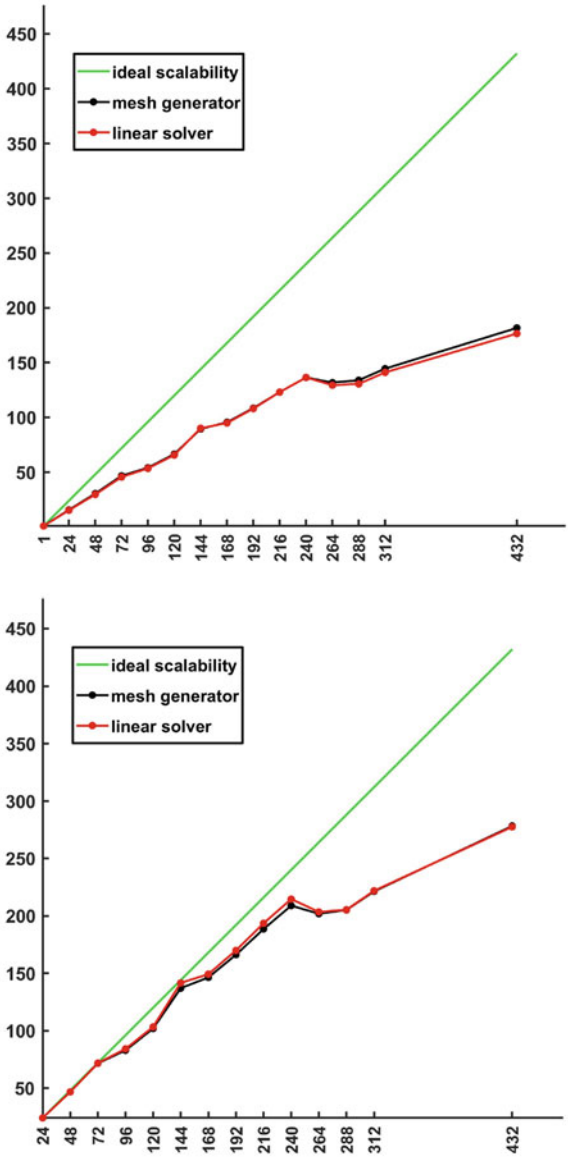


Fig. 15 Speed-up versus number of computational cores for a moderate-sized 2d problem: (top) the speed-up with respect to a single core and (bottom) the speed-up with respect to 24 cores

Figure 18a shows the initial adapted mesh and the target σ_1 distribution. Figure 18b shows the preimage of the compressed boundary layer on the initial uniform 3d mesh. In this example, in the boundary layer, $\sigma_1 = 30$ and $\sigma_{2,3} \approx 3$.

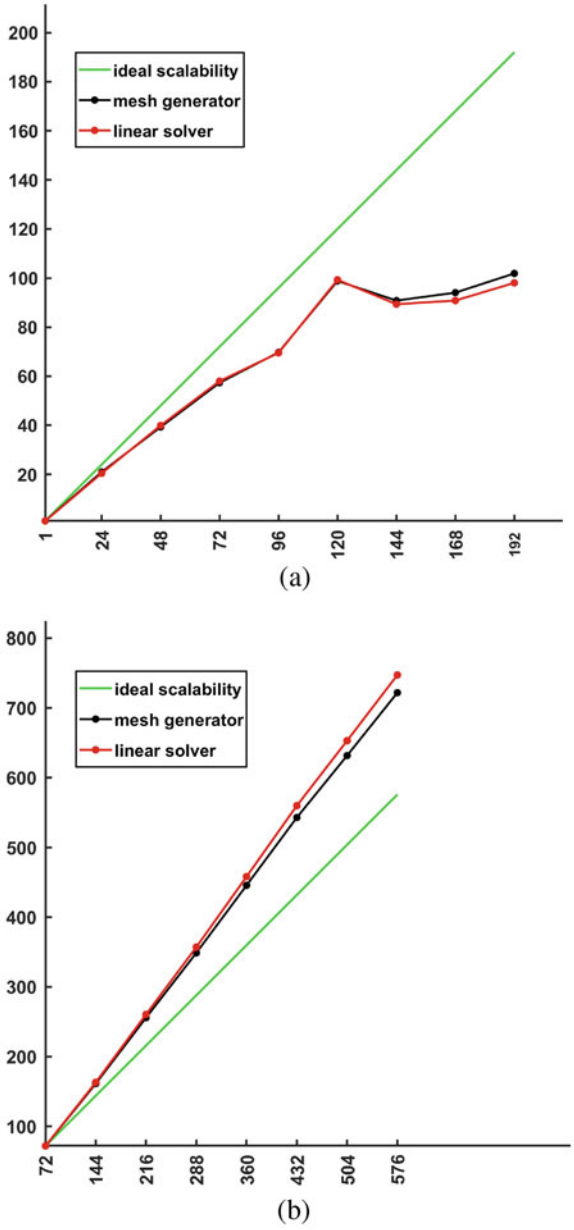


Fig. 16 Speed-up versus number of computational cores: (**a**) The small 2d problem and (**b**) The 3d problem

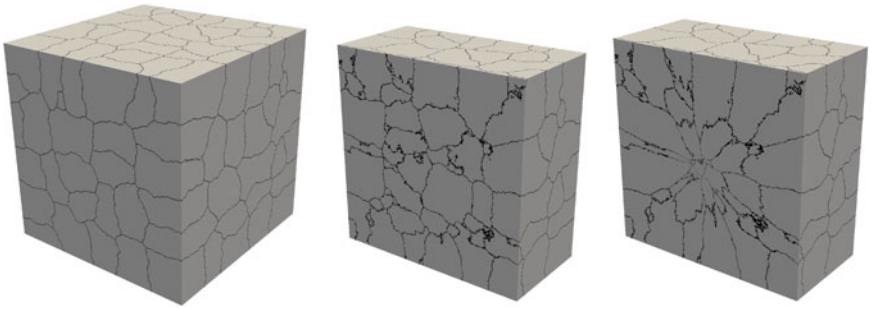


Fig. 17 Subdomains for the initial uniform mesh and for the instant moving deformed 3d mesh

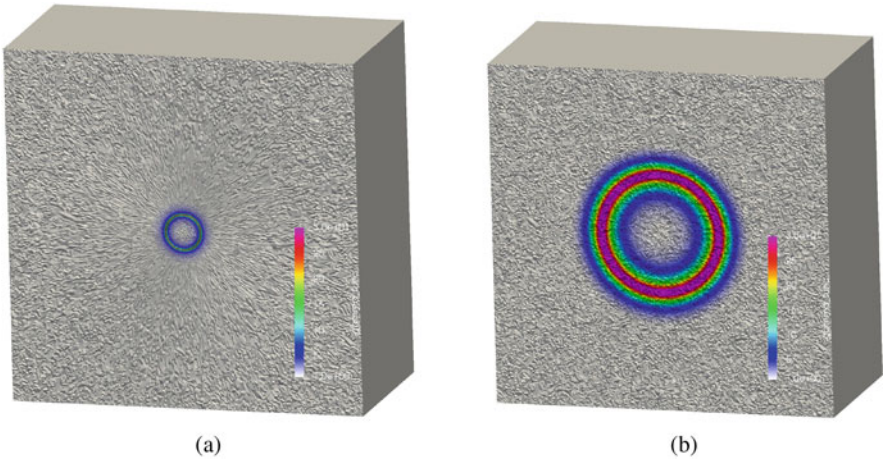


Fig. 18 Distribution of mesh compression factor in normal direction and its preimage on the initial mesh

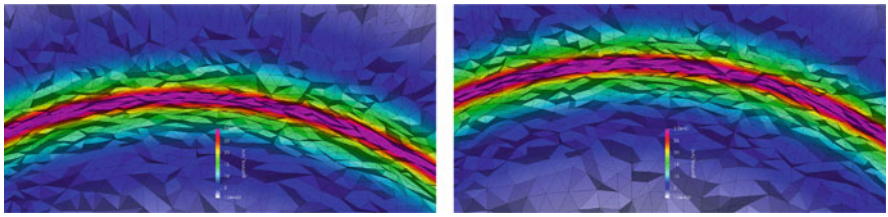


Fig. 19 Two positions of moving layer with imposed distribution of mesh compression factor

Figure 19 shows fragments of the mesh with two positions of the computed mesh layer in the process of the sphere movement.

Figure 20 shows two positions of the mesh layer extracted from the global meshes using a threshold of 20 for the target σ_1 distribution.

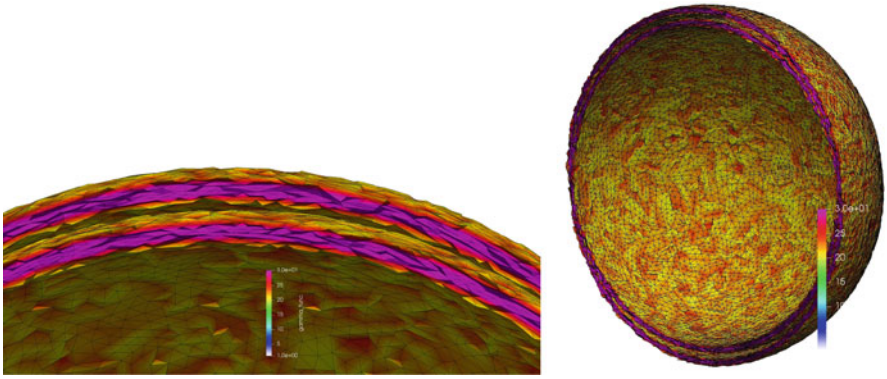


Fig. 20 Two positions of the compressed layer extracted using a threshold of 20 for the normal compression factor

7 Conclusions and Discussion

We describe an algorithm which allows to construct high quality time-dependent mesh deformations via the approximate minimization of a quasi-isometric functional. The presented algorithm is still slower if compared to mesh solvers based on linear elliptic equations, see, e.g., [15]. The difference, however, is no longer crucial since we use d linear solves with linear systems corresponding to FE approximations of scalar Laplace-like equations per several time steps. It is well known that algorithms based on a linear elliptic solver can attain reasonable mesh quality via a careful choice of the metric tensor/weight functions [17], so it may happen that the advantage of presented method in terms of the mesh quality does not overweight computational overhead. However, unlike linear mesh solvers, the presented algorithm does not impose any limitation on the domain shape and the mesh elements type. It can be applied in the case of multiple dimensions and for high-order elements. Due to the advanced parallel linear solver, a reasonable parallel scalability of the numerical algorithm was demonstrated.

Acknowledgments Research of the second author is supported by the Russian Science Foundation, Project 20-41-09018_ANR (acronym NORMA).

References

1. Abalakin, I., Bakhvalov, P., Kozubskaya, T.: Edge-based reconstruction schemes for unstructured tetrahedral meshes. Int. J. Numer. Methods. Fluids **81**(6), 331–356 (2016)
2. Brandt, J.B.: Small-scale propeller performance at low speeds. PhD Thesis. University of Illinois at Urbana-Champaign (2005)

3. Budd, C.J., Huang, W., Russell, R.D.: Adaptivity with moving grids. Acta Numer. **18**, 111–241 (2009)
4. Castro-Díaz, M.J., Hecht, F., Mohammadi, B., Pironneau, O.: Anisotropic unstructured mesh adaption for flow simulations. Int. J. Numer. Methods. Fluids **25**(4), 475–491 (1997)
5. Coyle, J.M., Flaherty, J.E., Ludwig, R.: On the stability of mesh equidistribution strategies for time-dependent partial differential equations. J. Comput. Phys. **62**, 26–39 (1986)
6. de Boor, C: Good approximation by splines with variable knots II. In: Spline Functions and Approximation Theory. Springer Lecture Notes Series, vol. 363. Springer, Berlin (1973)
7. Garanzha, V.A.: The barrier method for constructing quasi-isometric grids. Comput. Math. Math. Phys. **40**, 1617–1637 (2000)
8. Garanzha, V.A., Kudryavtseva, L.N.: Hyperelastic springback technique for construction of prismatic mesh layers. Proc. Eng. **203**, 401–413 (2017)
9. Garanzha, V.A., Kudryavtseva, L.N., Utyzhnikov, S.V.: Untangling and optimization of spatial meshes. J. Comput. Appl. Math. **269**, 24–41 (2014)
10. Godunov, S.K., Gordienko, V.M., Chumakov, G.A.: Quasi-isometric parametrization of a curvilinear quadrangle and a metric of constant curvature. Siberian Adv. Math. **5**(2), 1–20 (1995)
11. Huang, W., Ren, Y., Russell, R.D.: Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. SIAM J. Numer. Anal. **31**(3), 709–730 (1994)
12. Ivanenko, S.A.: Construction of nondegenerate grids. Comput. Math. Math. Phys. **28**, 141–146 (1988)
13. Kaporin, I.E.: High quality preconditioning of a general symmetric positive definite matrix based on its UTU+UTR+RTU-decomposition. Numer. Linear. Algebra. Appl. **5**(6), 483–509 (1998)
14. Kaporin, I.E., Milyukova, O.Yu.: MPI+OpenMP parallel implementation of explicitly preconditioned conjugate gradient method. Keldysh Institute preprints, **008** (Mi ipmp2369) (2018)
15. Tang, H.Z., Tang, T.: Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. SIAM J. Numer. Anal. **41**(2), 487–515 (2003)
16. Tsvetkova, V.O., Abalakin, I.V., Bobkov, V.G., Zhdanova, N.S., Kozubskaya, T.K., Kudryavtseva, L.N.: Simulation of flow near rotating propeller on adaptive unstructured meshes using immersed boundary method. Math. Models Comput. Simul. (accepted for publication, 2021)
17. Van Dam, A., Zegeling, P.A.: Balanced monitoring of flow phenomena in moving mesh methods. Commun. Comput. Phys. **7**(1), 138–170 (2010)

Adaptive Grids for Non-monotone Waves and Instabilities in a Non-equilibrium PDE Model

Paul A. Zegeling

Abstract In this paper, the importance of both the analysis and computation is emphasized, in relation to a bifurcation problem in a non-equilibrium Richard's equation from hydrology. The extension of this PDE model for the water saturation S, to take into account additional dynamic memory effects gives rise to an extra third-order mixed space-time derivative term in the PDE of the form $\tau\ \nabla \cdot [f(S)\nabla(S_t)]$. In one space dimension, travelling wave analysis is able to predict the formation of steep non-monotone waves depending on the parameter τ. In two space dimensions, the parameters τ and the frequency ω of a small perturbation term, predict that the waves may become *unstable*, thereby initiating so-called gravity-driven fingering structures. For the numerical experiments of the time-dependent PDE model, we have used a sophisticated adaptive grid r-refinement technique based on a scaled monitor function.

1 Introduction

Space-time evolution described by nonlinear PDE models involves patterns and qualitative changes induced by parameters. In this report I will emphasize the importance of both the analysis and computation in relation to a bifurcation problem in a non-equilibrium Richard's equation from hydrology. The extension of this PDE model for the water saturation S, to take into account additional dynamic memory effects, was suggested by Hassanizadeh and Gray [7] in the nineties of the last century. This gives rise to an extra third-order mixed space-time derivative term in the PDE of the form $\tau\ \nabla \cdot [f(S)\nabla(S_t)]$. In one space dimension, travelling wave analysis is able to predict the formation of steep non-monotone waves depending on the parameter τ. It is shown that, in this framework, theory from applied analysis, accurate numerical PDE solutions and also the experimental observations from the

P. A. Zegeling (✉)
Utrecht University, Utrecht, The Netherlands
e-mail: P.A.Zegeling@uu.nl

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_11

laboratory [5, 14] can be nicely matched. In two space dimensions, the parameters τ and the frequency ω (appearing in a small perturbation term), predict that the waves may become *unstable*, thereby initiating so-called gravity-driven fingering structures. This phenomenon can be analysed with a linear stability analysis and its effects are supported by the numerical experiments of the 2D time-dependent PDE model. For this purpose, we have used an efficient adaptive grid r-refinement technique based on a scaled monitor function. The numerical experiments in one and two space dimensions confirm the theoretical predictions and show the effectiveness of the adaptive grid solver.

2 The Non-equilibrium PDE Model

The PDE model describing non-equilibrium effects in a two-phase porous medium is given by [4, 7–9, 21, 22]:

$$S_t = \nabla \cdot (\mathcal{D}(S)\nabla S) + [f(S)]_z + \tau \nabla \cdot [f(S)\nabla(S_t)],$$
$$\text{with} \quad (x, z, t) \in [x_L, x_R] \times [z_L, z_R] \times (0, T], \tag{1}$$

where τ is a non-equilibrium parameter, $\mathcal{D}(S)$ is a nonlinear diffusion function and $f(S)$ is a fractional flow function, respectively.

2.1 The One-Dimensional Case

Assuming a constant diffusion and a linearized non-equilibrium term, respectively, in one space dimension, PDE model (1) can be reduced to:

$$S_t = \mathcal{D}\, S_{zz} + [f(S)]_z + \tau\, S_{zzt}, \quad (z, t) \in [z_L, z_R] \times (0, T], \tag{2}$$

with initial condition $S(z, 0) = S_0(z)$. The water saturation is represented by the variable $S(z, t) \in [0, 1]$, $\mathcal{D} > 0$ is a diffusion coefficient and $\tau \geq 0$ the non-equilibrium parameter (see also [7, 21, 22]). The function f satisfies: $f(0) = 0$, $f(1) = 1$, $f'(S) > 0$ and is related to a fractional flow function in the porous media model [22]. In particular, two choices for the function f are considered. The first one is a convex-shaped function, representing a one phase situation (only water), i.e.,

$$f(S) = \frac{S^2}{2},$$

and the second one is a convex-concave function, indicating two phases (both water and air are present):

$$f(S) = \frac{S^2}{S^2 + (1-S)^2}.$$

Dirichlet conditions are imposed at the spatial boundaries: $S(z_L, t) = S_-$ and $S(z_R, t) = S_+$. The initial water saturation $S_0(z)$, the boundaries of the spatial domain, the final time T and the values for $0 \leq S_- < S_+ \leq 1$, $\mathcal{D}$ and τ will be specified in the description of the numerical experiments.

2.2 Travelling Waves: A Bifurcation Diagram

This section discusses special types of solutions in PDE model (2): we are interested in travelling wave (TW) solutions. For simplicity of the analysis, we assume that $f(S) = S^2$. The convex-concave case is treated in [21, 22] which gives rise to an even richer structure of the dynamics. The TW Ansatz, assuming a positive constant speed ν, can be written as:

$$S(z,t) = \varphi(z + \nu\, t) := \varphi(\zeta), \quad \zeta \in (-\infty, +\infty), \quad \nu > 0.$$

Substituting this Ansatz in PDE (2), yields the third-order ODE:

$$\nu\, \varphi' = \mathcal{D}\, \varphi'' + [\varphi^2]' + \nu\, \tau\, \varphi''', \tag{3}$$

where the $'$ stands for taking derivatives with respect to the TW-variable ζ. Integrating (3) between $-\infty$ and ζ and using the fact that $\varphi(-\infty) = S_-, \varphi'(-\infty) = \varphi''(-\infty) = 0$, gives the second-order ODE:

$$\nu\, (\varphi - S_-) = \mathcal{D}\, \varphi' + \varphi^2 - S_-^2 + \nu\, \tau\, \varphi'', \tag{4}$$

which can be re-written as a system of first-order ODEs:

$$\begin{cases} \varphi' = \psi, \\ \psi' = \frac{\nu(\varphi - S_-) + S_-^2 - \varphi^2 - \mathcal{D}\, \psi}{\nu\, \tau}. \end{cases} \tag{5}$$

A TW for (2) in the coordinate system (x, t) is represented by a trajectory in the (φ, ψ)-plane connecting an unstable stationary point (at $\zeta = -\infty$) of (5) with a stable one (at $\zeta = +\infty$). There are only two stationary points in system (5):

$(\varphi, \psi) = (S_-, 0)$, and $(\varphi, \psi) = (S_+, 0)$. The eigenvalues of the linearized system of (5) can easily be calculated:

$$\lambda_{1,2} = \frac{-\mathcal{D}}{2\tau v} \pm \sqrt{\frac{\mathcal{D}^2}{4\tau^2 v^2} + \frac{S_+ - S_-}{2\tau v}}. \tag{6}$$

From (6) it follows that the point $(S_-, 0)$ is an unstable (saddle) point in all cases, since $\lambda_1 \lambda_2 < 0$. Depending on the PDE parameters $\mathcal{D}$ and τ, we distinguish between two cases for the second stationary point $(S_+, 0)$. Non-monotone TWs exist for $\tau > \tau_c = \mathcal{D}^2/(S_+ - S_-)$ since the saddle point is then connected to a spiral point (a focus). This situation is clarified in terms of the PDE solutions (Fig. 1), a bifurcation diagram (left panel in Fig. 2) and a phase plane plot (right panel in Fig. 2). For $\tau = 0$, it is known that only monotone waves satisfy the PDE model [4]. Since we are looking also for non-monotone waves, we need the extra τ-term in PDE (2) to describe such phenomena. Note again that the convex-concave case is treated in [22], for which there may exist *plateau-type* waves as well. In that case three stationary points appear in the dynamical system with a much more complicated behaviour in the phase plane. These plateau-waves will be detected in

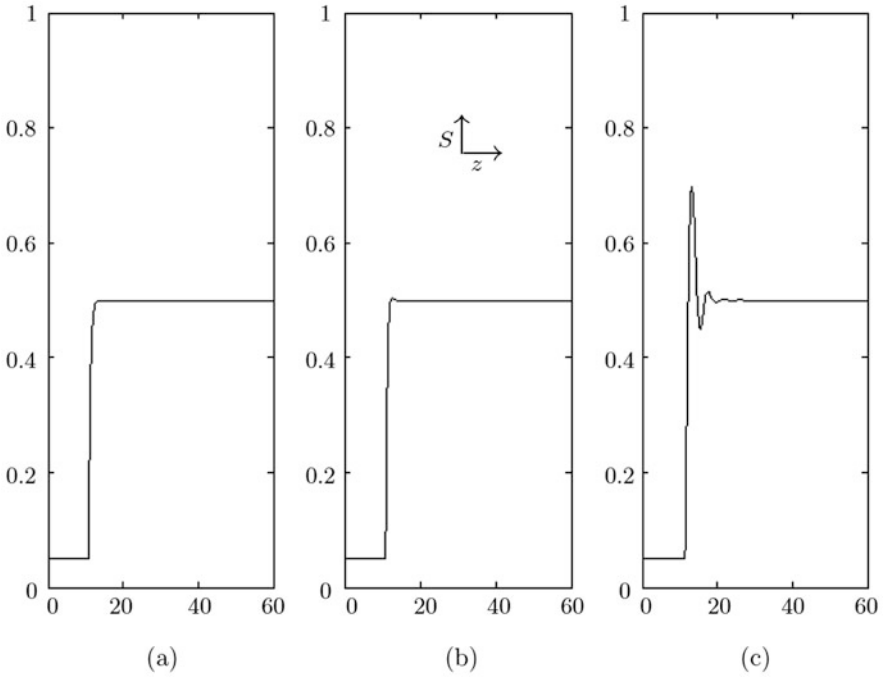


Fig. 1 The one-dimensional saturation profile for various values of the non-equilibrium parameter τ: (**a**) $\tau = 0$ (monotone TW); (**b**) $\tau = 1$ (mildly non-monotone TW); (**c**) $\tau = 5$ (oscillating non-monotone TW)

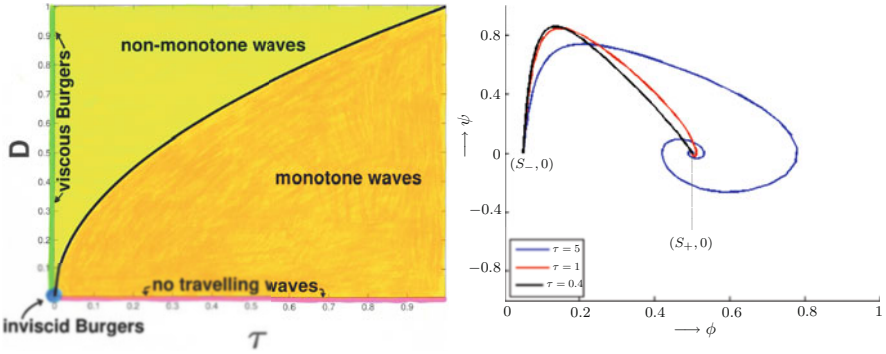


Fig. 2 A bifurcation diagram (left) indicating the existence of monotone waves and non-monotone waves depending on the parameters $\mathcal{D}$ and τ. The black curve is defined by: $\mathcal{D} = \sqrt{\tau(S_+ - S_-)}$. The right panel shows, for three different values of the parameter τ, trajectories in the phase plane (φ, ψ). The red and blue curves correspond to non-monotone waves ($\tau > \tau_c > 0$) and the black curve denotes a monotone wave ($\tau = 0$)

the numerical experiments, where we will use an adaptive moving grid method to solve the PDE model (2).

2.3 The Adaptive Moving Grid in 1D

For the numerical simulations of PDE model (2) we will apply an adaptive moving grid technique that is based on a coordinate transformation (for more details: [2, 8, 10, 11, 19, 24, 25]):

$$\begin{cases} z = z(\xi, \vartheta), \\ t = t(\xi, \vartheta) = \vartheta. \end{cases} \tag{7}$$

Then, PDE model (2) is transformed to the following form:

$$\upsilon_\vartheta - \frac{(\upsilon + z_\vartheta)}{\mathcal{J}}\upsilon_\xi = \frac{\mathcal{D}}{\mathcal{J}}\left(\frac{\upsilon_\xi}{\mathcal{J}}\right)_\xi + \frac{\upsilon_\xi}{\mathcal{J}} f'(\upsilon) + \frac{\tau}{\mathcal{J}}\left[\mathcal{J}\left(\frac{1}{\mathcal{J}}\left(\frac{\upsilon_\xi}{\mathcal{J}}\right)_\xi\right)_\vartheta - z_\vartheta\left(\frac{1}{\mathcal{J}}\left(\frac{\upsilon_\xi}{\mathcal{J}}\right)_\xi\right)_\xi\right], \tag{8}$$

with $\upsilon(\xi,\vartheta) := S(z(\xi,\vartheta),\vartheta)$ and the Jacobian of transformation (7): $\mathcal{J} := z_\xi$. The adaptive grid transformation that defines the time-dependent non-uniform grid has to satisfy the adaptive grid PDE;

$$[(\sigma(\mathcal{J}) + \tau_s\,\mathcal{J}_\vartheta)\,\mathcal{M}]_\xi = 0, \quad \tau_s \geq 0.$$

Here, $\mathcal{M} := \sqrt{1+[S_z]^2}$ is the monitor function, reflecting the dependence of the non-uniform grid on the spatial derivative of the PDE solution.

The operator

$$\sigma := \mathcal{I} + \kappa_s(\kappa_s+1)\frac{\partial^2}{\partial\xi^2}$$

is applied to obtain a smoother grid transformation in space. The first adaptivity constant $\kappa_s > 0$ is a spatial smoothing (or filtering) parameter. Further, the second adaptivity constant τ_s takes care of the smoothness in the time-direction. For $\kappa_s > 0$ and $\tau_s > 0$, after semi-discretization, it can be shown [10] that the spatial grid satisfies the condition

$$\frac{\kappa_s}{\kappa_s+1} \leq \frac{\Delta z_{i+1}}{\Delta z_i} \leq \frac{\kappa_s+1}{\kappa_s}$$

for all gridpoints z_i and all time $t > 0$. Note that, for $\kappa_s = \tau_s = 0$ (no smoothing), we return to the basic equidistribution principle $[z_\xi \mathcal{M}]_\xi = 0$. For more details on the adaptive grid and the smoothing operators we refer to [10, 24, 25]. The transformed PDE and the adaptive grid PDE are simultaneously semi-discretized in the spatial direction following a method-of-lines approach. A central, second-order, uniform approximation for the transformed derivative terms in the ξ-direction is used. The time-integration of the resulting coupled ODE-system is done by a BDF method with variable time-steps in DASSL [16].

2.4 Numerical Results

In this section, we perform some numerical experiments to show the accuracy and effectiveness of the adaptive moving grid. This will also illustrate and confirm the TW analysis in Sect. 2.2. The adaptive grid parameters are chosen as follows: $\kappa_s = 2$ and $\tau_s = 0.001$ and the time-integration a tolerance in DASSL is set to the value 10^{-4}. The initial condition is a steep wave starting at the right boundary of the domain and reads:

$$S(z,0) = S_0(z) = S_- + \frac{1}{2}(S_+ - S_-)(1+\tanh(R(z-z_0))),$$

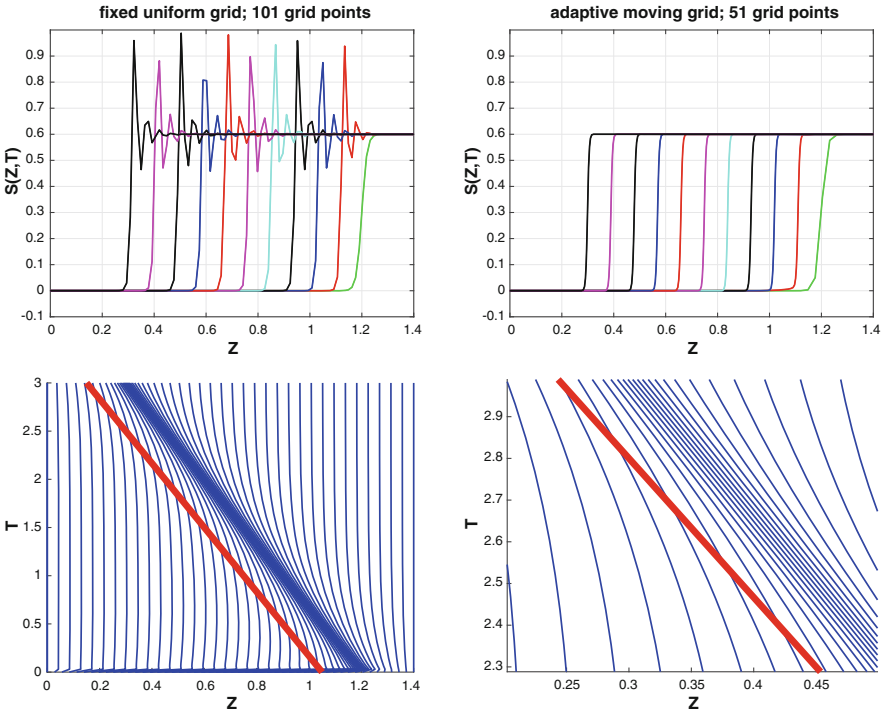


Fig. 3 Upper two panels for the case $\tau = 0$: oscillating uniform ($N = 101$) grid solutions (left) and non-oscillating adaptive grid solutions with $N = 51$ (right). The lower two plots (with a close-up in the right panel) display the time-history of the adaptive grid. In these experiments, we have chosen: $\tau = 3 \times 10^{-4}$, $\mathcal{D} = 2 \times 10^{-3}$. The red lines indicate the exact (asymptotic) wave speed

where $z_L = 0$, $z_R = 1.4$, $S_- = 0$, $S_+ = 0.6$, $R = 50$ and PDE parameters: $\tau = 10^{-4}$ and $\mathcal{D} = 10^{-3}$. In Fig. 3 we show, for $\tau = 0$, in which case we know that only monotone solutions exist, numerical solutions with $N = 101$ fixed uniform grid points and with $N = 51$ adaptive moving grid points. It is clearly observed that the uniform grid produces an unwanted non-monotone wave, whereas the adaptive grid nicely keeps the wave monotone. Also, the plot with the time history of the adaptive grid illustrates the smooth distribution and time-behaviour of the grid with a constant wave velocity. Figure 4 displays the difference between the convex and the convex-concave case: non-monotone TWs and plateau-waves. These waves are predicted by the analysis in Sect. 2.2 and [22].

From ODE (4) it can easily be derived that the asymptotic travelling-wave speed v satisfies:

$$v = \frac{f(S_+) - f(S_-)}{S_+ - S_-}. \tag{9}$$

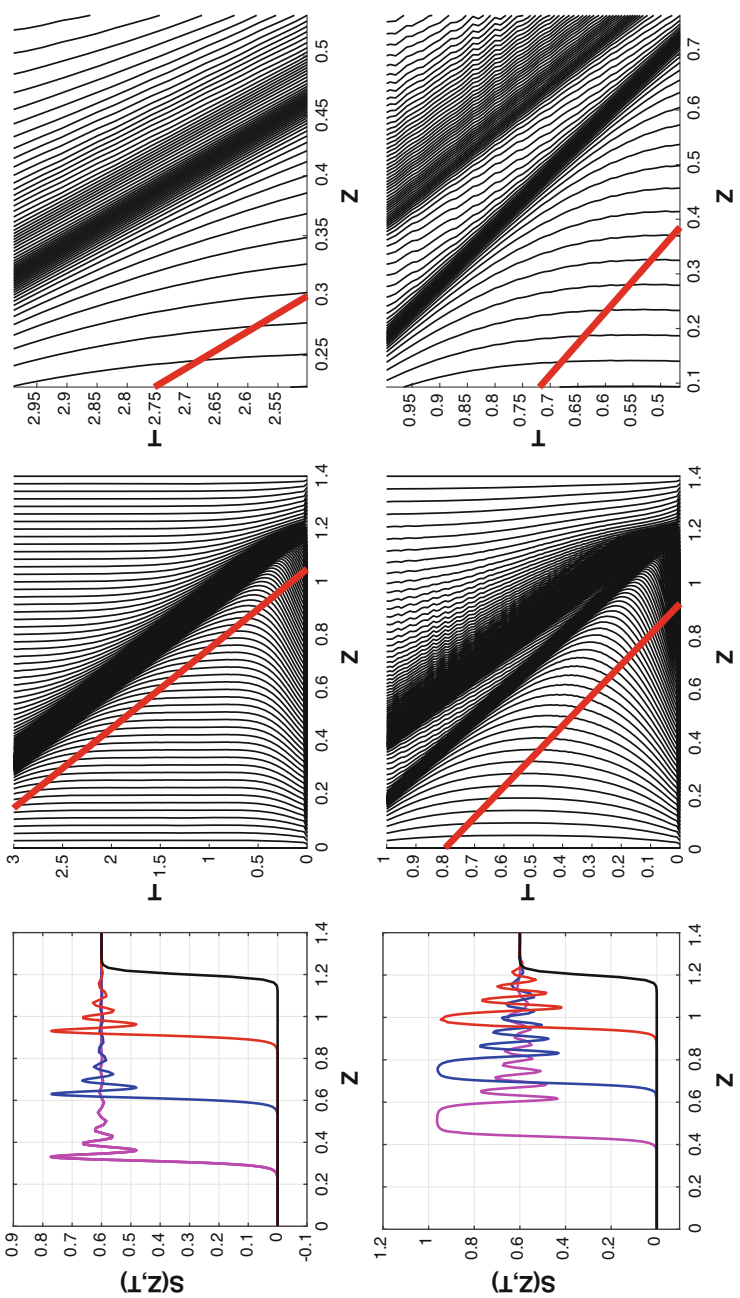


Fig. 4 The time history of the adaptive grid (right, with a close-up around the steep parts of the waves), the solutions at four points of time (left) for two characteristic cases in the porous media model: a convex f (top) and a convex-concave f (bottom). The red straight lines indicate the exact (asymptotic) wave speeds for the two cases, as predicted by formula (9)

This yields, respectively, for the convex case $\nu = 0.3$ and for the convex-concave case $\nu \approx 1.1538$. In Figs. 3 and 4 the red lines indicate these constant TW speeds. We observe that the adaptive moving grid follows the waves very accurately.

3 The Two-Dimensional Case

Next, we consider the two-dimensional version of model (1) for the special choices:

$$f(S) = S^{\alpha}, \quad \mathcal{D}(S) = \beta S^{\alpha-\beta-1}, \quad \alpha > \beta + 1. \tag{10}$$

3.1 Non-monotone Waves and Instabilities

In contrast with the 1D case, for which both the monotone and non-monotone waves are stable under small perturbations, the 2D model may give rise to *in*stabilities ('finger' structures). It can be shown that for specific values of $\tau > 0$, the non-monotone waves may become *un*stable. The analysis is based on the following observations, also mentioned in [6] and [15].

First, the non-equilibrium PDE (1) is re-written as a system of two equations, one for the saturation S and one for the pressure p:

$$\begin{cases} S_t = \nabla \cdot (\mathcal{D}(S)\nabla p) + [f(S)]_z, \\ \tau S_t = p - \mathcal{P}(S), \end{cases} \tag{11}$$

where $\mathcal{P}(S)$ is an equilibrium pressure. Next, the PDEs are written in a travelling wave coordinate, similar as in done in Sect. 2.2. The saturation and pressure waves are then perturbed in the following form:

$$\begin{cases} S = S_0(\zeta) + \varepsilon \mathrm{e}^{i\omega_x + i\omega_z + kt} S_1(\zeta) + O(\varepsilon^2), \\ p = p_0(\zeta) + \varepsilon \mathrm{e}^{i\omega_x + i\omega_z + kt} p_1(\zeta) + O(\varepsilon^2). \end{cases} \tag{12}$$

These perturbed quantities are substituted in the system of two travelling wave equations, higher-order terms are being neglected, and equations for the linear stability analysis are set up. For these, it can be derived, that, for $\tau = 0$, the growth factor k will always be negative, whereas, for $\tau > 0$ and for certain frequencies ω, the growth factor can be positive, thereby initiating unstable waves. These can be related to so-called fingering structures, as we will see in Sect. 3.3.

3.2 The Adaptive Moving Grid in 2D

The adaptive grid method in two dimensions follows similar principles, with some extra features and differences, compared to the 1D situation. More details can be found in, for example, the references [10, 18–20], and [23]. Summarizing the procedure, the 2D grid transformation reads:

$$\begin{cases} x = z(\xi, \eta, \vartheta), \\ z = z(\xi, \eta, \vartheta), \\ t = t(\xi, \eta, \vartheta) = \vartheta. \end{cases} \tag{13}$$

As an example, the first term of the nonlinear diffusion on the righthand side in PDE model (1) transforms to:

$$\begin{aligned} (\mathcal{D}(S)S_x)_x = &\frac{1}{\mathcal{J}}\left[\left(\frac{\mathcal{D}(S)z_\eta^2}{\mathcal{J}}S_\xi\right)_\xi - \left(\frac{\mathcal{D}(S)z_\xi z_\eta}{\mathcal{J}}S_\eta\right)_\xi\right. \\ &\left. - \left(\frac{\mathcal{D}(S)z_\xi z_\eta}{\mathcal{J}}S_\xi\right)_\eta + \left(\frac{\mathcal{D}(S)z_\xi^2}{\mathcal{J}}S_\eta\right)_\eta\right], \end{aligned} \tag{14}$$

where $\mathcal{J} = x_\xi z_\eta - x_\eta z_\xi$ denotes the determinant of the Jacobian of transformation (13). The one-dimensional basic equidistribution principle, $[\mathcal{M}z_\xi]_\xi = 0$, is extended to a system of two coupled nonlinear elliptic PDEs:

$$\begin{cases} \nabla \cdot (\mathcal{M}\nabla x) = 0, \quad \nabla := \left[\frac{\partial}{\partial\xi}, \frac{\partial}{\partial\eta}\right]^T, \\ \nabla \cdot (\mathcal{M}\nabla z) = 0, \end{cases}$$

where the monitor function $\mathcal{M}$ is now defined by

$$\mathcal{M} = \gamma(t) + \sqrt{\nabla S \cdot \nabla S}, \quad \text{with} \quad \gamma(t) = \iint_{\Omega_c} \sqrt{\nabla S \cdot \nabla S}\, \mathrm{d}\xi\, \mathrm{d}\eta\,.$$

It is obvious that more sophisticated monitor functions could be used, but, for the PDE model in this paper, this relatively simple monitor function has shown to be sufficiently effective. Note that we have added a time-dependent adaptivity function $\gamma(t)$ which is automatically calculated during the time-integration process. It provides additional smoothing to the grid distribution and takes care of the scaling in the space and solution directions (see [20] for more information about this choice). It can be shown that the adaptive grid transformation, following this 2D equidistribution principle with the mentioned monitor function $\mathcal{M}$, remains non-singular:

Theorem 1 (For Details of the Proof: [3]) *Let* $\mathcal{M} > 0$, $\mathcal{M} \in C^1(\Omega_c)$ *and* $\mathcal{M}_\xi, \mathcal{M}_\eta \in C^\gamma(\bar{\Omega}_c)$ *for* $\gamma \in (0,1)$. *Then there exists a unique solution* $(x, z) \in C^2(\bar{\Omega}_c)$, *which is a bijection from* $\bar{\Omega}_c$ *into itself. Moreover, the determinant of the Jacobian* $\mathcal{J}$ *satisfies*

$$\mathcal{J} = x_\xi z_\eta - x_\eta z_\xi > 0.$$

Some important ingredients of their proof include the Jordan curve theorem, the Carleman-Hartman-Wintner theorem and the maximum principle for elliptic PDEs.

In reference [1] a deep analysis of the invertibility of more general, so-called σ-harmonic, mappings is given. The transformed PDE model is spatially discretized uniformly in the ξ and η coordinates. For the numerical time-integration of the transformed 2D non-equilibrium PDE and the adaptive grid equations, we have used an IMplicitEXplicit-approach [12, 17]. As an example, the diffusion term (14) is numerically approximated as follows:

$$\begin{aligned}
&(\mathcal{D}(S)S_x)_x|^n_{i,j} \\
&\quad\approx \frac{1}{\mathcal{J}^n_{i,j}}\left[\frac{C_1|^n_{i+1,j}+C_1|^n_{i,j}}{2}\frac{S^{n+1}_{i+1,j}-S^{n+1}_{i,j}}{(\Delta\xi)^2} - \frac{C_1|^n_{i,j}+C_1|^n_{i-1,j}}{2}\frac{S^{n+1}_{i,j}-S^{n+1}_{i-1,j}}{(\Delta\xi)^2}\right. \\
&\qquad - C_2|^n_{i+1,j}\frac{S^{n+1}_{i+1,j+1}-S^{n+1}_{i+1,j-1}}{4\Delta\xi\,\Delta\eta} + C_2|^n_{i-1,j}\frac{S^{n+1}_{i-1,j+1}-S^{n+1}_{i-1,j-1}}{4\Delta\xi\,\Delta\eta} \\
&\qquad - C_2|^n_{i,j+1}\frac{S^{n+1}_{i+1,j+1}-S^{n+1}_{i-1,j+1}}{4\Delta\xi\,\Delta\eta} + C_2|^n_{i,j-1}\frac{S^{n+1}_{i+1,j-1}-S^{n+1}_{i-1,j-1}}{4\Delta\xi\,\Delta\eta} \\
&\qquad \left.+\frac{C_3|^n_{i,j+1}+C_3|^n_{i,j}}{2}\frac{S^{n+1}_{i,j+1}-S^{n+1}_{i,j}}{(\Delta\eta)^2} - \frac{C_3|^n_{i,j}+C_3|^n_{i,j-1}}{2}\frac{S^{n+1}_{i,j}-S^{n+1}_{i,j-1}}{(\Delta\eta)^2}\right],
\end{aligned} \tag{15}$$

where $C_1 := \frac{1}{\mathcal{J}}\mathcal{D}(S)z_\eta^2$, $C_2 := \frac{1}{\mathcal{J}}\mathcal{D}(S)z_\xi z_\eta$, and $C_3 := \frac{1}{\mathcal{J}}\mathcal{D}(S)z_\xi^2$, respectively. Instead of the 'smart' smoothing operators in space and time, as being used in 1D, here, a filter, as in [18, 23], on the monitor function is applied several times in each time step in the following way:

$$\begin{aligned}
\tilde{\mathcal{M}}_{i,j} = \frac{1}{4}\mathcal{M}_{i,j} &+ \frac{1}{8}\big[\mathcal{M}_{i-1,j}+\mathcal{M}_{i+1,j}+\mathcal{M}_{i,j-1}+\mathcal{M}_{i,j+1}\big] \\
&+ \frac{1}{16}\big[\mathcal{M}_{i-1,j-1}+\mathcal{M}_{i+1,j-1}+\mathcal{M}_{i-1,j+1}+\mathcal{M}_{i+1,j+1}\big].
\end{aligned} \tag{16}$$

This modification yields even smoother grid distributions and enhances the time-integration process as well.

3.3 Numerical Results

To support the main results of the analysis in Sect. 3.1, we perform some numerical experiments for the 2D model. The spatial domain is defined by the rectangle $[0, 10] \times [0, 60]$ and the initial solution is a 'tanh'-type function as in 1D, but now situated around the value $z = 55$. We add a small periodic perturbation with frequency ω to test the stability of the two-dimensional waves. The numerical experiments, unless specified differently, make use of a spatial grid with 41×121 grid points.

Figure 5 indeed confirms and illustrates the stability analysis by Egorov et al. [6] and Nieber et al. [15], also briefly described in Sect. 3.1. The left panel shows the numerically calculated growth factor k of the perturbation as a function of the initial frequency ω for several values of the non-equilibrium parameter τ, using the adaptive grid method from Sect. 3.2. The right panel is taken from [6] and depicts a very similar dependence of $k(\omega)$.

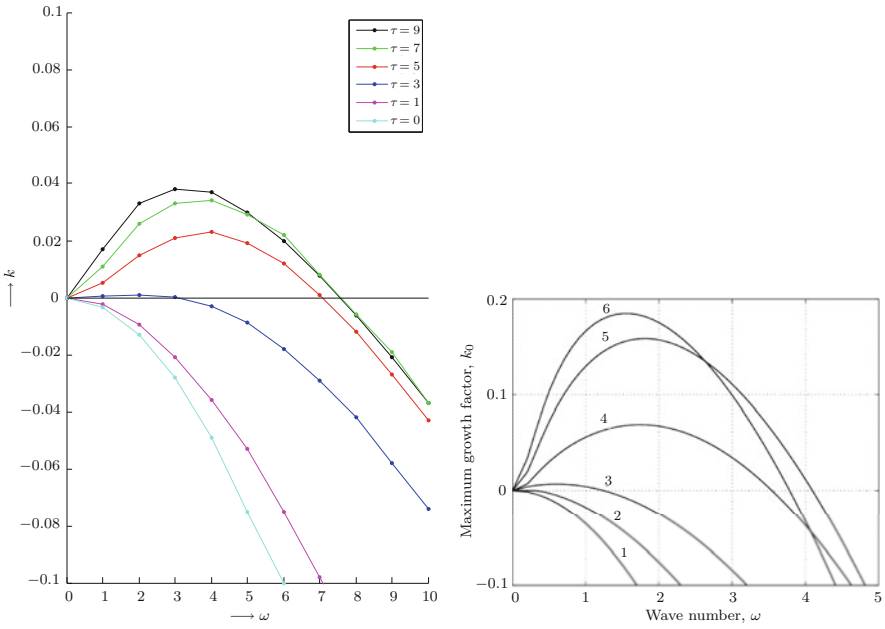


Fig. 5 Left: the growth factor k, numerically determined, as a function of the wave number ω of the perturbation for various values of τ. Right: the theoretical prediction, taken from [6]. Note that the scales on both axes in the two figures are different. The ω on the left is a numerical frequency added to the initial condition, whereas the ω on the right comes from a theoretical analysis. The global behaviour is the same, but the exact values are different. A similar remark holds for the growth factor k

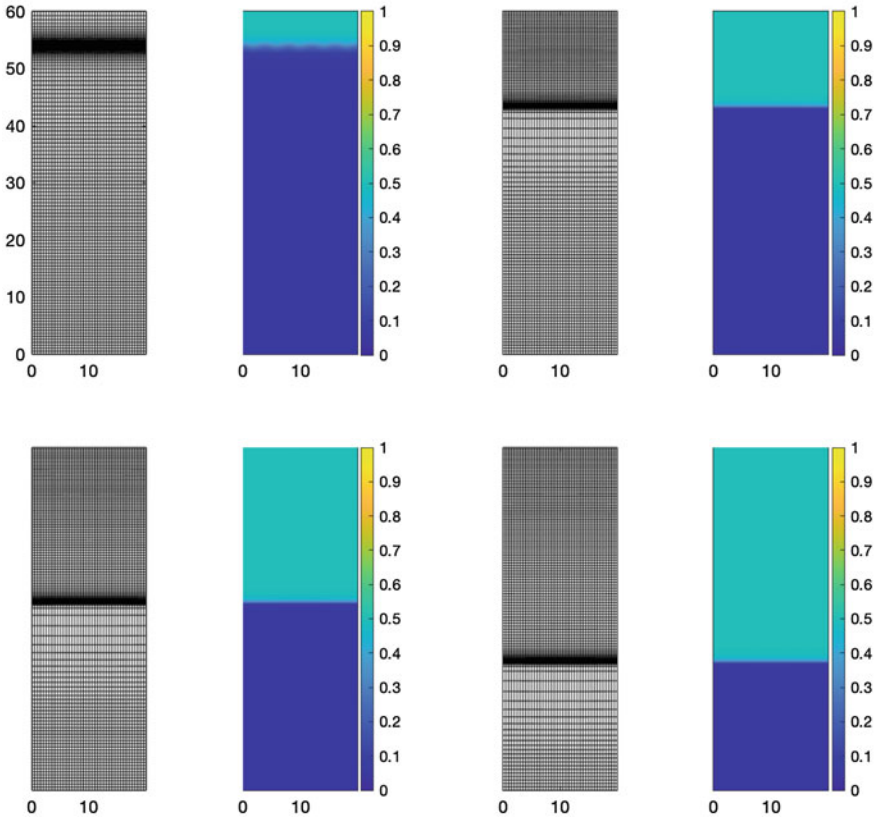


Fig. 6 The water saturation profile and corresponding grid at different times for $\tau = 0$ and $\beta = 0.5$ on a 41×121-grid

In Fig. 6 we show a *stable* travelling wave in two dimensions for $\tau = 0$, $\alpha = 3$ and $\beta = 0.5$: the initial perturbation disappears quickly and the adaptive moving grid follows the steep parts of the wave efficiently and accurately. Figure 7 displays close-ups of the adaptive grid for the solutions in Fig. 6.

Next, in Fig. 8 a similar situation is displayed, but now for $\tau = 10$, $\alpha = 3$ and $\beta = 0.5$: the initial perturbation grows in time and the wave becomes *unstable* creating the shape of a 'finger'. Again, the 2D adaptive grid is situated nicely around the steep parts of the irregularly shaped travelling wave. Figure 9 shows close-ups of the adaptive grids for the solutions in Fig. 8.

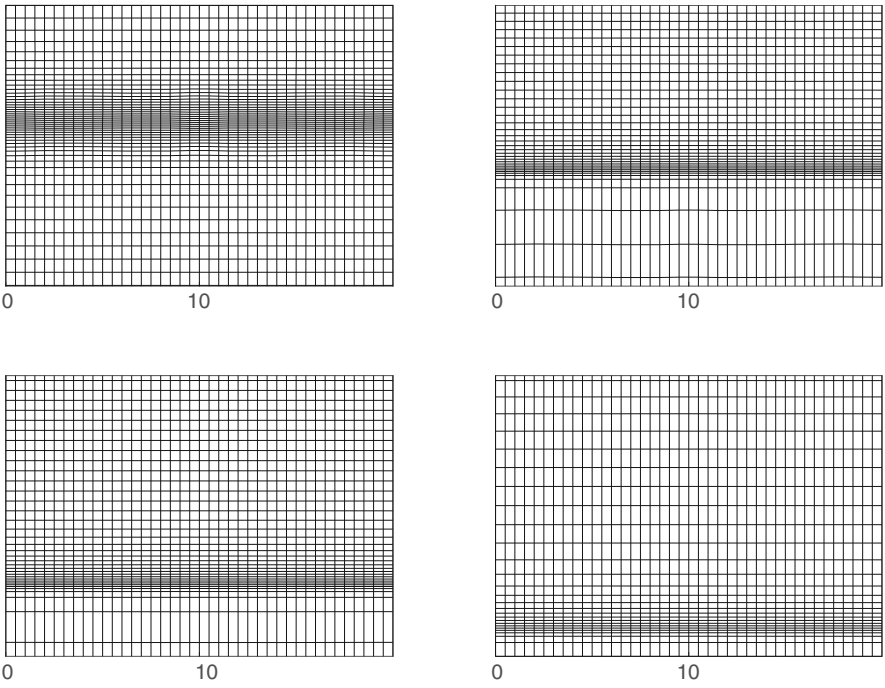


Fig. 7 Close-ups of the adaptive grid near the steep parts of the travelling wave in Fig. 6 at four different points of time for $\tau = 0$ and $\beta = 0.5$

Figure 10 demonstrates the convergence of the adaptive grid solution to an unstable finger with steep transitions. Here, the number of grid points is increased from 21×41 up to 81×241. Note the extra smoothing of the grid distribution. This is due to the time-dependent function $\gamma(t)$ in the monitor function, which becomes more visible for denser grids. Also, it must be mentioned that the non-uniform grid has an important effect of the accuracy of the fingering structure. This phenomenon has also been observed and described in reference [13].

In Fig. 11, the values $\tau = 10$, $\alpha = 3$ and $\beta = 0.2$ are chosen. The smaller value of the parameter β has clearly the effect of producing a more profound fingering structure. Figure 12 shows close-ups of the adaptive grids for these numerical solutions in Fig. 11.

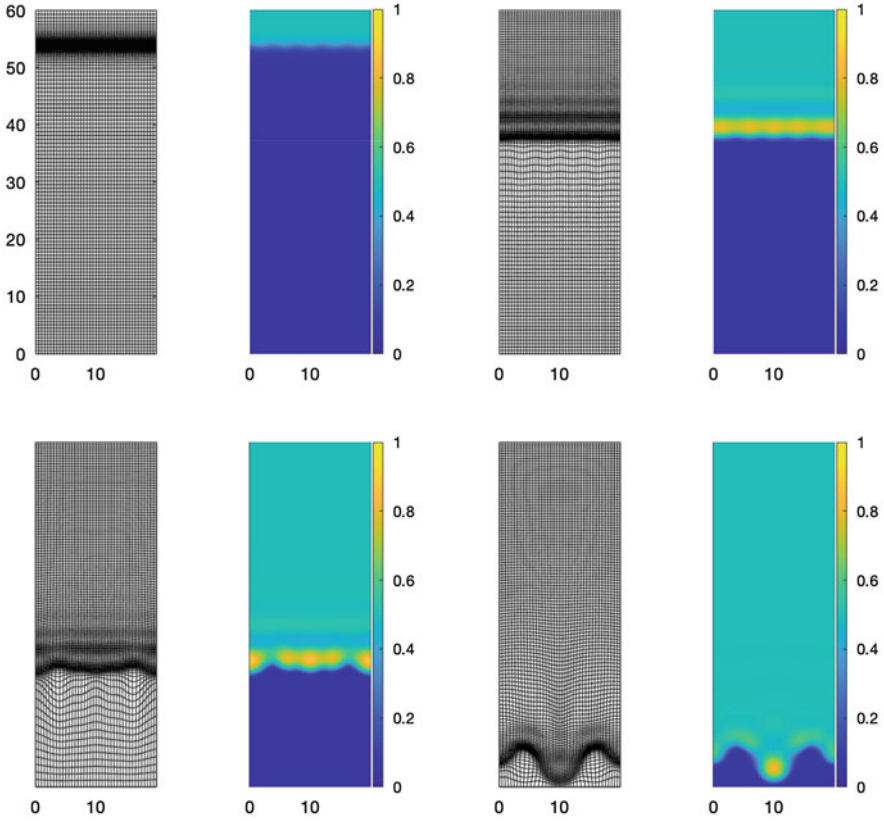


Fig. 8 The water saturation profile and corresponding grid at different times for $\tau = 10$ and $\beta = 0.5$ on a 41×121-grid

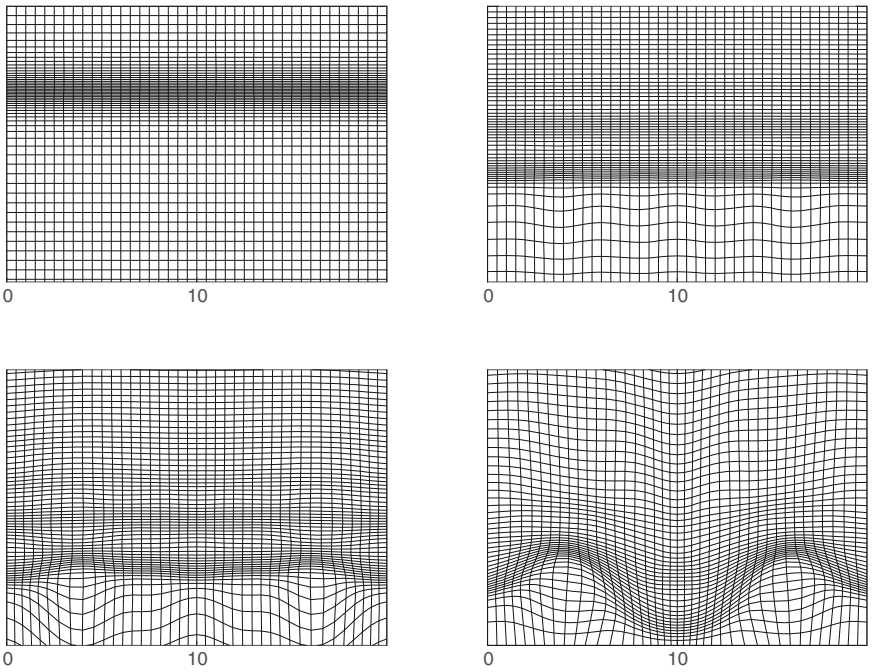


Fig. 9 Close-ups of the adaptive grid near the steep parts of the fingering solution in Fig. 8 for $\tau = 10$ and $\beta = 0.5$

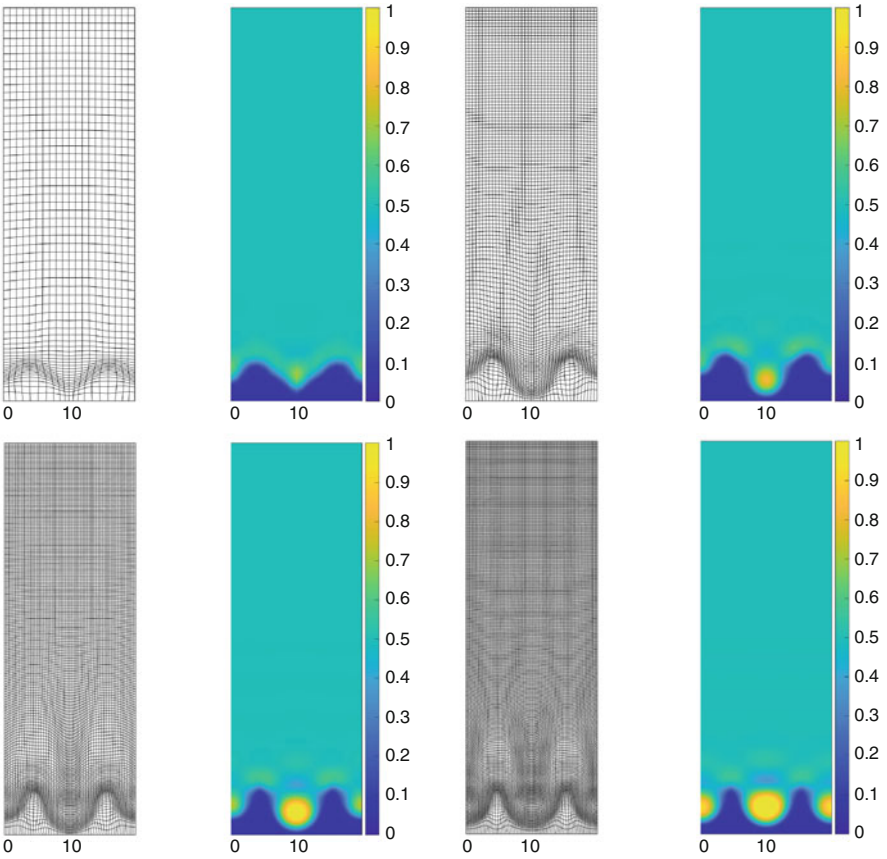


Fig. 10 Adaptive grids and solutions for the case $\tau = 10$ and $\beta = 0.5$ for an increasing number of spatial grid points: 21×61 (top left), 41×121 (top right), 61×181 (bottom left) and 81×241 (bottom right). Note the additional smoothing effect for denser grids of the function $\gamma(t)$ in the monitor function. See also [13] for similar conclusions on the effects of the grid distribution on the fingering structure

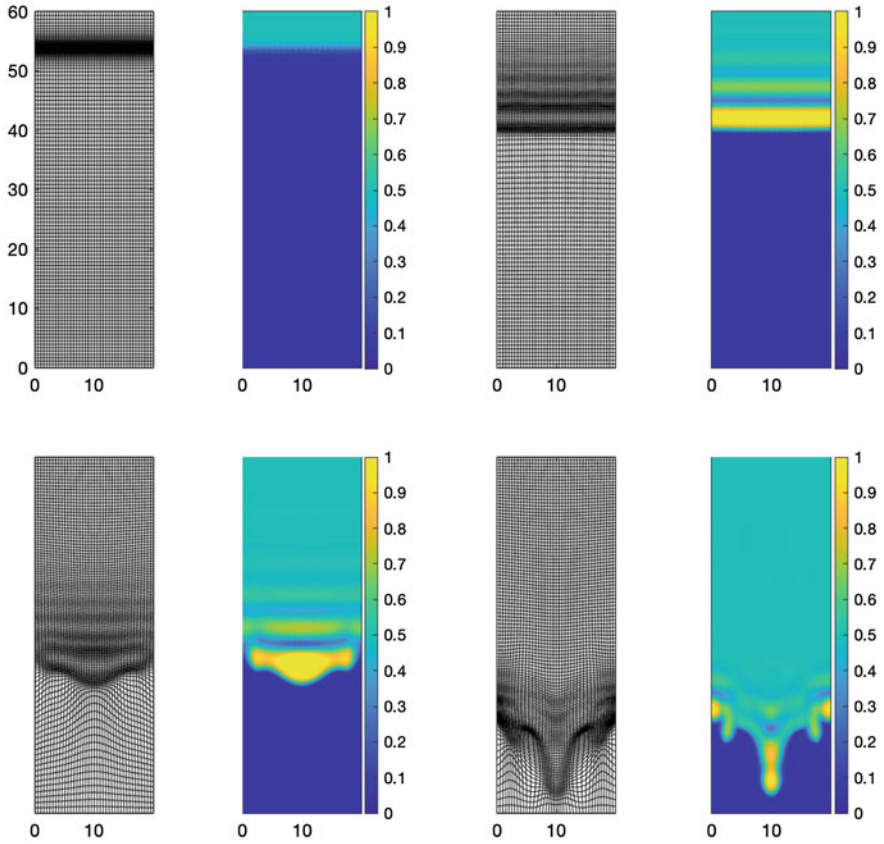


Fig. 11 The water saturation profile and corresponding grid at different times for $\tau = 10$ and $\beta = 0.2$. The instability of the solution, creating a fingering structure, is now more profound than in the previous case. This is due to the smaller value of the parameter β

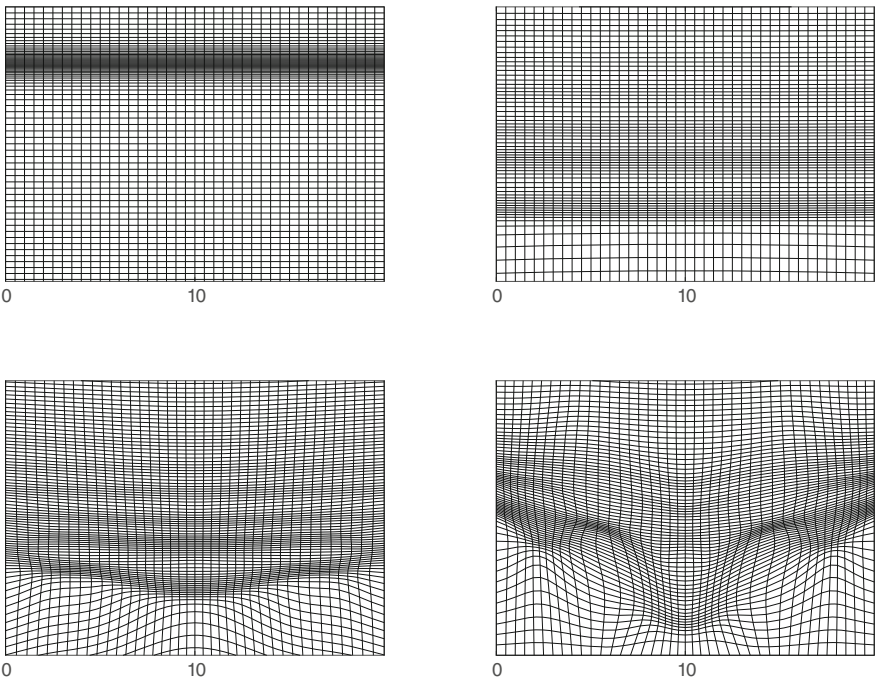


Fig. 12 Close-ups of the adaptive grid near the steep parts of the fingering solution in Fig. 11 for $\tau = 10$ and $\beta = 0.2$

References

1. Alessandrini, G., Nesi, V.: Univalent σ-harmonic mappings. Arch. Rational Mech. Anal. **158**, 155–171 (2001)
2. Budd, C.J., Huang W., Russell R.D.: Adaptivity with moving grids. Acta Numer. **18**, 111–241 (2009)
3. Clement, Ph., Hagmeijer, R., Sweers, G.: On the invertibility of mappings arising in 2D grid generation problems. Numer. Math. **73**(1), 37–52 (1996)
4. Cuesta, C., van Duijn, C.J., Hulshof, J.: Infiltration in porous media with dynamic capillary pressure: travelling waves. Eur. J. Appl. Math **11**, 397 (2000)
5. DiCarlo, D.: Experimental measurements of saturation overshoot on infiltration. Water Resour. Res. **40**, W04215 (2004)
6. Egorov, A.G., Dautov, R.Z., Nieber, J.L., Sheshukov, A.Y.: Stability analysis of gravity-driven infiltrating flow. Water Resour. Res. **39**, 1266 (2003)
7. Hassanizadeh, S.M., Gray, W.G.: Thermodynamic basis of capillary pressure on porous media. Water Resour. Res. **29**, 3389–3405 (1993)
8. Hilfer, R., Doster, F., Zegeling, P.A. Nonmonotone saturation profiles for hydrostatic equilibrium in homogeneous porous media. Vadose Zone J. **11**(3), 201 (2012)
9. Hu, G., Zegeling, P.A.: Simulating finger phenomena in porous media with a moving finite element method. J. Comput. Phys. **230**(8), 3249–3263 (2011)
10. Huang, W., Russell, R.D.: Analysis of moving mesh partial differential equations with spatial smoothing. SIAM J. Numer. Anal. **34**, 1106–1126 (1997)

11. Huang, W., Russell, R.D.: Adaptive Moving Mesh Methods. Springer, New York (2011)
12. Hundsdorfer, W., Verwer, J.: Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer, Berlin (1993)
13. Kampitsis, A.E., Adam, A., Salinas, P., Pain, C.C., Muggeridge, A.H., Jackson, M.D.: Dynamic adaptive mesh optimisation for immiscible viscous fingering. Comput. Geosci. **24**, 1221–1237 (2020)
14. Nicholl, M.J., Glass, R.J.: Infiltration into an analog fracture: experimental observations of gravity-driven fingering. Vadose Zone J. **4**, 1123–1151 (2005)
15. Nieber, J.L., Dautov, R.Z., Egorov, A.G., Sheshukov, A.Y.: Dynamic capillary pressure mechanism for instability in gravity-driven flows; review and extension to very dry conditions. Transp. Porous Media **58**, 147–172 (2005)
16. Petzold, A.G.: A description of DASSL: a differential/algebraic system solver. In: Stepleman, R.S., et al. (eds.) IMACS Trans. Sci. Comput., pp. 65–68. North-Holland, Amsterdam (1983)
17. Ruuth, S.J.: Implicit-explicit methods for reaction-diffusion problems in pattern formation. J. Math. Biol. **34**, 148–176 (1995)
18. Tang, T., Tang, H.: Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. SIAM J. Numer. Anal. **41**(2), 487–515 (2003)
19. van Dam, A., Zegeling, P.A.: A robust moving mesh finite volume method applied to 1d hyperbolic conservation laws from magnetohydrodynamics. J. Comput. Phys. **216**, 526–546 (2006)
20. van Dam, A., Zegeling, P.A.: Balanced monitoring of flow phenomena in moving mesh methods. Commun. Comput. Phys. **7**, 138–170 (2010)
21. van Duijn, C.J., Hassanizadeh, S.M., Pop, I.S., Zegeling, P.A.: Non-equilibrium models for two-phase flow in porous media: the occurrence of saturation overshoot. In: Proc. of the 5th Int. Conf. on Appl. of Porous Media, Cluj-Napoca (2013)
22. van Duijn, C.J., Fan, Y., Peletier, L.A., Pop, I.S.: Travelling wave solutions for a degenerate pseudo-parabolic equation modelling two-phase flow in porous media. Nonlinear Anal. Real World Appl. **14**, 1361–1383 (2013)
23. Zegeling, P.A.: On resistive MHD models with adaptive moving meshes. J. Sci. Comput. **24**(2), 263–284 (2005)
24. Zegeling, P.A.: Theory and application of adaptive moving grid methods. In: Adaptive Computations: Theory and Algorithms, pp. 279–332. Science Press, Beijing (2007)
25. Zegeling, P.A., Lagzi, I., Izsak, F.: Transition of Liesegang precipitation systems: simulations with an adaptive grid PDE method. Commun. Comput. Phys. **10**(4), 867–881 (2011)

RBF-VerBSS Hybrid Method for Mesh Deformation

Jihai Chang, Fei Yu, Jie Cao, and Zhenqun Guan

Abstract The mesh deformation method has been widely applied to numerical simulation of time-variant problems. In the present study, we proposed a hybrid mesh deformation method based on radial basis functions (RBF) method and vertex-ball-spring-smoothing (VerBSS) method. Firstly, a coarse background mesh which is consistent with the boundary of the computational mesh was generated and deformed by RBF. The internal nodal displacements were duplicated to the corresponding nodes of computational mesh. The perturbated nodes and the boundary nodes were then utilized together to calculate the deformation of the computational mesh by employing VerBSS. By such means, better convergence performance was achieved. Results of numerical examples indicate that the proposed method has higher efficiency and better robustness than conventional RBF method or background mesh method for large scale problem.

1 Introduction

Mesh deformation is a widely used method to solve unsteady fluid problems with moving boundary. Mesh deformation methods preserve the topology of the mesh, and avoid rebuilding the flow field data frequently. These methods can be broadly classified into two categories: physical analogy methods and interpolation analogy methods [13].

Physical analogy methods include spring analogy and elastic solid analogy. Batina et al. [1] proposed spring method first. In this method, each edge of the mesh is replaced by a tension spring with the spring stiffness inversely proportional to the edge length. Farhat et al. [5] and Bottasso et al. [2] improved the spring method by introducing the torsional spring and ball-vertex spring, respectively. For the elastic method [15], the whole computational domain is considered as an elastic solid and

J. Chang · F. Yu · J. Cao, · Z. Guan (✉)
Dalian University of Technology, Dalian, China
e-mail: fei.yu@mail.dlut.edu.cn; caojie@mail.dlut.edu.cn; guanzhq@dlut.edu.cn

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_12

the mesh deformation is governed by the classical laws of elastic theory. The elastic method improves the deformation capability comparing with the spring analogy but also introduces considerable computational cost. The main draw-back of physical analogy methods is that they preserve connectivity and large systems of equations, implying a higher computational cost. Lin et al. [8] proposed the VerBSS method by decomposing the linear equations into sub-spring systems. It can improve the computational efficiency and the required memory is greatly reduced. This method will be discussed in the following sections.

Compared with the physical method, the interpolation analogy methods do not require connectivity information and can be applicable to arbitrary mesh types. The displacements of the internal nodes of the mesh are regarded as the interpolation problem of the boundary nodes. The interpolation methods include RBF interpolation [4] and Delaunay graph interpolation [9] mainly.

The RBF can be used as an interpolation function to transfer the displacements known at the boundaries of the mesh to the interior nodes. It produces high-quality meshes with reasonable orthogonality preservation near deforming boundaries. However, for large-scale problems, RBF method is also expensive since a system of linear equations which dimension is equal to the number of the mesh boundary points. Rendall et al. [11] proposed an efficient data reduction algorithm along with interpolation. Sheng and Allen [14] investigated two greedy algorithms to reduce the number of the control points. Selim et al. [12] introduced the concept of solving the RBF system incrementally within the greedy algorithm. However, the reduction of the control points caused that the boundary geometry precision declines and interpolation error is brought in.

Delaunay graph interpolation is a very efficient method. The coarse Delaunay graph is used as an intermediate map. Each interior node is assigned to the Delaunay element which belongs to, and its interpolation based on surface or volume ratios is utilized to map from the original position to its new position. Nevertheless, intersections occur occasionally for complex geometry with large relative movements.

The moving submesh algorithm (MSA) was proposed by Lefrancois et al. [7] who uses better quality background mesh to control the deformation. More auxiliary nodes are added to improve the quality of the background mesh by Zhou et al. [17]. Although these methods improve the deformation quality of the background mesh, it is still difficult to avoid the problem of collapsed elements.

Simple physical methods or interpolation methods have their own advantages and disadvantages, so some hybrid mesh deformation methods can achieve better deformation combine the advantages of different methods [16]. Liu et al. [10] proposed a hybrid method combining MSA with RBF, which makes a deformed mesh more uniform, but the domain of each submesh is discontinuous and some elements may collapse.

In recent years, the multigrid method has been widely concerned [3, 6]. This method can reduce the iteration times effectively. Inspired by the multigrid method, this paper proposes a hybrid deformation method combining the VerBSS method and RBF. Firstly, the coarse mesh is deformed by RBF. The deformed nodal

displacements of the coarse mesh are duplicated to the approximate nodes of the computational mesh. Then the VerBSS method is used to smooth the computational mesh, so as to accelerate the convergence. On the one hand, this method uses the idea of the multigrid method to improve the efficiency of VerBSS method; on the other hand, it improves the robustness of the background mesh method.

The rest of the paper is organized as follows: Sect. 2 briefly recalls the methodology of the RBF-based mesh deformation and the VerBSS method. And the robustness of background mesh is also introduced. The hybrid RBF-VerBSS method is presented in Sect. 3. Numerical examples are presented in Sect. 4 to demonstrate the efficacy of the proposed methods. At last, the concluding remarks are drawn in Sect. 5.

2 Related Work

2.1 RBF Interpolation Method

The RBF method interpolates the displacement of the surface mesh to all the nodes of the flow mesh. The displacement of any of these interior nodes could be calculated by:

$$\mathbf{s}(\mathbf{r}) = \sum_{i=1}^{N_b} \omega_i \phi(\|\mathbf{r} - \mathbf{r}_i\|), \tag{1}$$

where $\mathbf{r}_i$ is the coordinate of the ith boundary node, N_b is the number of the boundary nodes, ω_i is the weight coefficient of the ith boundary node. ϕ is the RBF as function of the Euclidean distance $\|\mathbf{r}\|$, We choose the Wendland's C^2 function, which is suitable for the mesh deformation interpolation:

$$\phi(\eta) = \begin{cases} (1-\eta)^4(4\eta+1), & \text{when } \eta \le 1, \\ 0, & \text{when } \eta > 1, \end{cases} \tag{2}$$

where $\eta = \|\mathbf{r} - \mathbf{r}_i\|/R$, and R is the support radius. The displacements of the boundary nodes are predefined as known values:

$$\mathbf{s}(\mathbf{r}_b) = \Delta\mathbf{r}_b, \tag{3}$$

where $\mathbf{r}_b$ is the coordinate of the j-th boundary node and $\Delta\mathbf{r}_b$ is the prescribed displacement of it. The function coefficients could be calculated as follows:

$$\boldsymbol{\omega} = \Phi_{b,b}^{-1}\Delta\mathbf{r}_b, \tag{4}$$

where $\Phi_{b,b}$ is an $N_b \times N_b$ matrix which carries the RBF evaluations:

$$\Phi_{b,b}(i, j) = \phi\big(\|\mathbf{r}_{b_i} - \mathbf{r}_{b_j}\|\big). \tag{5}$$

2.2 VerBSS Method

The edges of the mesh are considered as spring with stiffness inversely proportional to its length. The force on vertex i exerted by vertex j can be written as

$$\mathbf{f}_{ij}^{\text{Edge}} = k_{ij}(\mathbf{u}_j - \mathbf{u}_i) \cdot \mathbf{n}_{ij}\mathbf{n}_{ij}, \tag{6}$$

where k_{ij} is the stiffness of edge e_{ij}, $\mathbf{n}_{ij}$ is the unit vector from i to j. The displacement of vertices i and j are denoted by $\mathbf{u}_i$ and $\mathbf{u}_j$, respectively. To prevent elements inversion, additional perpendicular linear springs are added in the ball-vertex method. Perpendicular spring S_{ip} is constructed with the stiffness inversely proportional to its length. Similarly to the previous spring equilibrium equations, the resulting force of spring S_{ip} can be expressed as

$$\mathbf{f}_{ip}^{\text{ball-vertex}} = k_{ip}(\mathbf{u}_p - \mathbf{u}_i) \cdot \mathbf{n}_{ip}\mathbf{n}_{ip}, \tag{7}$$

where k_{ip} is the stiffness of the new spring, $\mathbf{n}_{ip}$ is the unit vector from i to p. The displacement of vertex i and p are denoted respectively by $\mathbf{u}_i$ and $\mathbf{u}_p$. Its edge springs and the ball-vertex spring can form a new spring system as:

$$\sum_{j=1}^{n} \mathbf{f}_{ij}^{\text{Edge}} + \sum_{p=1}^{m} \mathbf{f}_{ip}^{\text{ball-vertex}} = 0. \tag{8}$$

Following the vertex-ball spring system, a sub-system for a vertex i is formed by edge springs together with the ball-vertex springs as shown in Fig. 1. The global system of equations $K\mathbf{u} = \mathbf{b}$ is reduced to a series of $d \times d$ sub-systems, where d is the spatial dimension. The right-hand vector $\mathbf{b}_i$ is written as:

$$\mathbf{b}_i = \sum_{j=1}^{n} k_{ij}\mathbf{u}_j \cdot \mathbf{n}_{ij}\mathbf{n}_{ij} + \sum_{p=1}^{m} k_{ip}\mathbf{u}_p \cdot \mathbf{n}_{ip}\mathbf{n}_{ip} = \mathbf{B}_i\mathbf{u}_i^{'}, \tag{9}$$

where $\mathbf{u}_i^{'}$ is the nodal displacement vector of the ball, n and m are the number of nodes and elements of the ball. B_i is a matrix of constant coefficients. Introduction of a relaxation factor ξ can further speed up convergence as follow:

$$\mathbf{u}_i^{\text{new}} = \xi\mathbf{u}_i^{\text{new}} + (1 - \xi)\mathbf{u}_i^{\text{old}}, \tag{10}$$

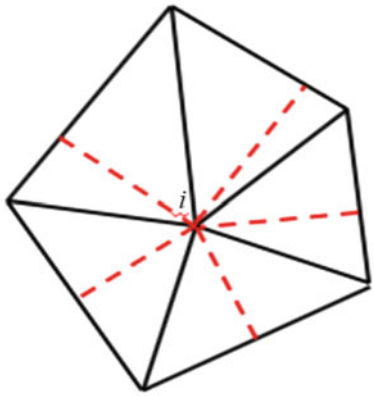


Fig. 1 A sub-system of the VerBSS algorithm

Numerical example show that ξ can be set between 1.5 and 1.7 (over-relaxation). The termination criterion is:

$$\sum_{i=1}^{d\cdot N} \left|\mathbf{u}_i - \mathbf{u}_i^{\text{prev}}\right| / (d \cdot N) < \varepsilon_q, \tag{11}$$

where $\mathbf{u}_i^{\text{prev}}$ is the result from the previous iteration step, N is the number of interior nodes. For a given tolerance $\varepsilon_q \in \mathbb{R}^+$, when the mean value of the difference between two adjacent iteration results falls below a prescribed value ε_q, the mesh smoothing will terminate.

The procedure of the VerBSS is given as follows [8]:

Step 1: Import the initial mesh and store the topological data of all nodes;
Step 2: Construct the vertex-ball spring system of interior node i so as to form stiffness matrix K and coefficient matrix B, and decompose matrix K into L and D using LDL^T solver, and store the matrix LD and B for subsequent steps;
Step 3: Set displacements of boundary nodes to the specified values;
Step 4: Solve the equations by LDL^T matrix decomposition and update the displacements;
Step 5: Repeat Step 4 until the result of displacements meets the computational accuracy requirements.

2.3 Robustness of the Delaunay Graph and the MSA Method

The fundamental principle of Delaunay graph interpolation method is to divide and interpolate. A Delaunay graph is generated by using all or parts of the boundary nodes as the background mesh for the original mesh. Every node in the whole

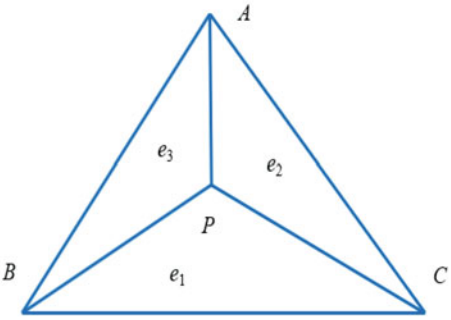


Fig. 2 The barycentric coordinates of node P

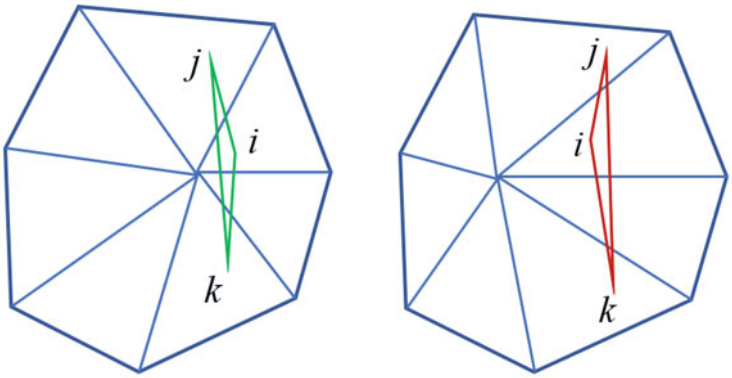


Fig. 3 Deformation of the background mesh. Left original triangle. Right: collapsed triangle

computational mesh can be split into the Delaunay triangles (2D) or tetrahedrons (3D). The new position can be achieved by the surface or volume ratios. For 2D case, the node P is inside the triangle $\triangle ABC$ as shown in Fig. 2, and its barycentric coordinates e_1, e_2, e_3 with respect to A, B, C are calculated by:

$$e_i = \frac{S_i}{S}. \tag{12}$$

However, for large deformation, especially for deformation with large rotation, the mesh quality could degrade suddenly since the lack of controlling mechanisms. The quality of deformed mesh depend on the quality of Delaunay mesh.

The MSA method can be considered as an extension of the Delaunay graph interpolation method. The main difference between the two methods is that in the MSA the background mesh is not a Delaunay graph anymore and have more interior nodes.

In the MSA, the submesh can only control its interior nodes. If the three nodes of a triangle are in different submesh as shown in Fig. 3, then the background mesh

elements do not collapsed after deformation, but the computational mesh elements collapsed.

3 RBF-VerBSS Hybrid Mesh Deformation Method

In this paper, in order to overcome the weakness of background mesh and keep the benefits of RBF interpolation and VerBSS, a hybrid method of RBF and VerBSS mesh deformation is proposed.

Firstly, a coarsen background mesh whose boundary is consistent with the computational mesh is given as shown as Fig. 4a. The nodes of the computational mesh are partitioned according to their coordinates. The triangle element in which the node of the coarse background mesh is located, can be found quickly by the partition blocks. The whole computational domain is considered as an elastic solid for the VerBSS method. Therefore, the displacements of node C nearest to node P could supposed to be equal to the node P approximately as shown as Fig. 4b.

$$\mathbf{u}_c = \mathbf{u}_p \tag{13}$$

Then, the background mesh is deformed by RBF interpolation and the specific process is the steps in Sect. 2.1. The displacements of deformed nodes of the background mesh are duplicated to their corresponding computational nodes.

Finally, according to the VerBSS method in Sect. 2.2, all nodes of the computational mesh are smoothed. This method includes the following steps:

Step 1: Generate the background mesh;
Step 2: Locate the background mesh nodes on the computational mesh element and find the nearest element;

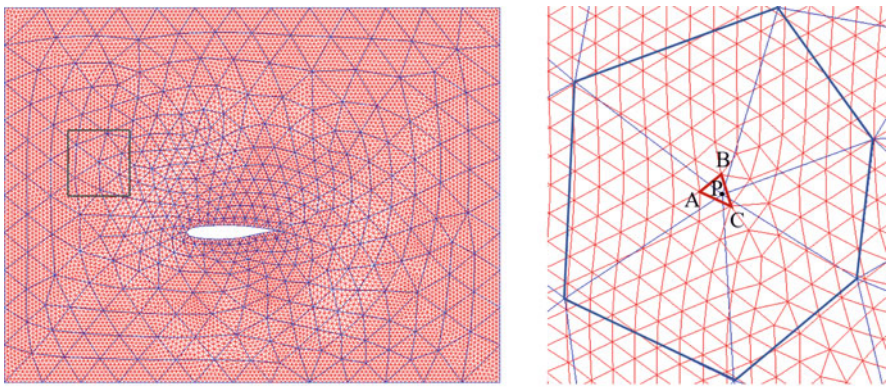


Fig. 4 RBF-VerBSS hybrid method. (**a**) Locating the nodes. (**b**) Enlarged view of a part of mesh

Step 3: Deform the background mesh by RBF and update the coordinates of the approximate node of the computational mesh;
Step 4: Take the pre-deformation displacements of the approximate nodes and the boundary nodes as boundary conditions to solve the computational mesh by VerBSS;
Step 5: Update the coordinates of computational mesh nodes and repeat Step 3 to Step 4 until the end of computation.

4 Results and Discussions

4.1 NACA 0012 Airfoil Rotation

The NACA 0012 airfoil rotated in 2D flow field. The computational mesh and the background mesh are shown in Fig. 5. The computational mesh has 66,527 nodes and 131,786 elements. The background mesh has 1659 nodes and 3175 elements. The performances is measured on a single core Intel i7 3.4 GHz processor.

As can be seen from the figures, the RBF-VerBSS method have more shape-shifting. The maximum number of deformation steps of the two methods are 62 and 92, respectively. Figure 6 illustrates the minimum angle of the mesh changed with time steps in the deformation process of the two methods when the mesh deformed to the 60th step, the quality of the mesh using the VerBSS and the RBF-VerBSS is shown in Table 1. The RBF-VerBSS method can keep a better quality than the VerBSS method.

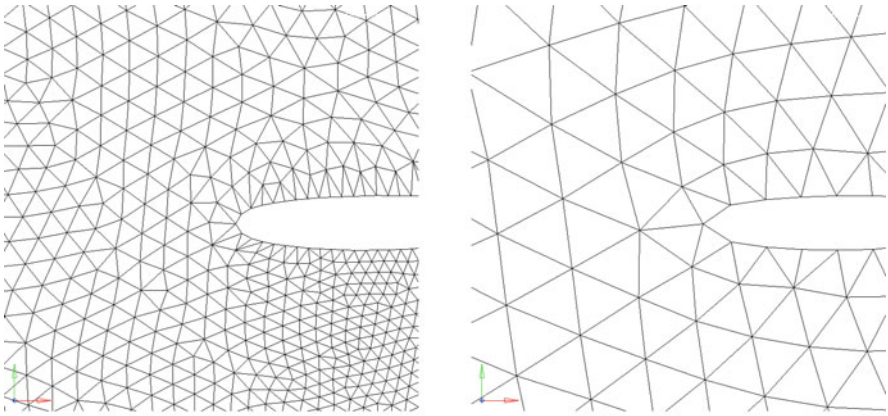


Fig. 5 Airfoil rotation: computational and background meshes. (**a**) Computational mesh. (**b**) Background mesh

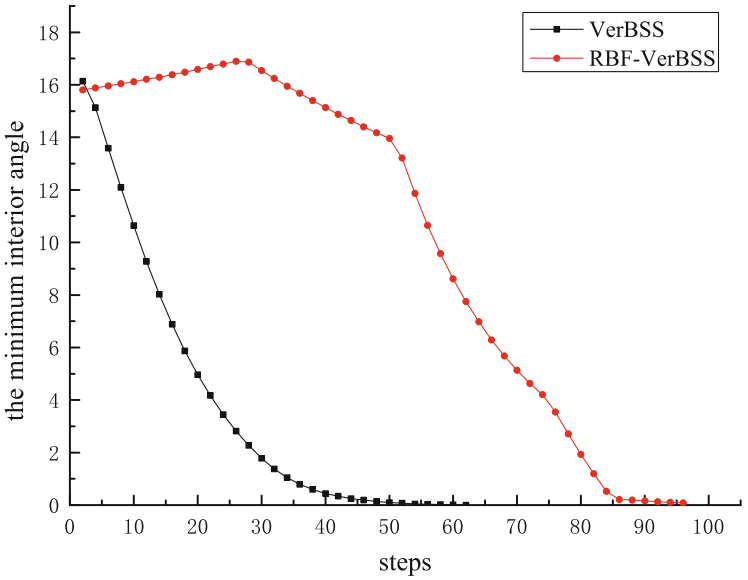


Fig. 6 Airfoil rotation: the minimum interior angle by VerBSS and RBF-VerBSS

Table 1 Airfoil rotation: comparison of mesh quality for the rotation of 1.2 rad

Quality	VerBSS	RBF-VerBSS
0.00–0.02	13	0
0.02–0.10	27	1
0.10–0.40	235	547
0.40–0.70	646	12,578
0.70–1.00	130,865	118,660

The wing rotates clockwise with a rotation step of 0.02 rad by using the VerBSS and the RBF-VerBSS methods. The number of iterations of the VerBSS and RBF-VerBSS methods is 558 and 121, respectively. The meshes are shown in Fig. 7 when the airfoil rotates to the 62nd step. CPU time for a deformation step for the two methods is shown in Table 2. As expected, preprocessing of the background mesh optimizes the condition number of the VerBSS method's matrix and clearly reduces the corresponding computational time for the deformation (Table 2).

4.2 3D Moving Aircraft

For a fluid mesh with a large number of elements, Fig. 8 illustrates two cross-sectional views of an aircraft in a 3D fluid mesh. One is the computational mesh and the other is the background mesh. The number of the nodes is 594,980 and the

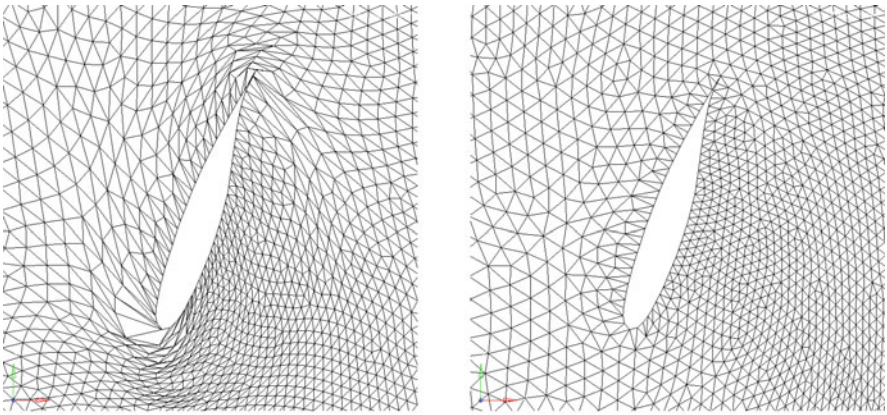


Fig. 7 Airfoil rotation: deformed meshes after the 62nd step. (**a**) VerBSS. (**b**) RBF-VerBSS

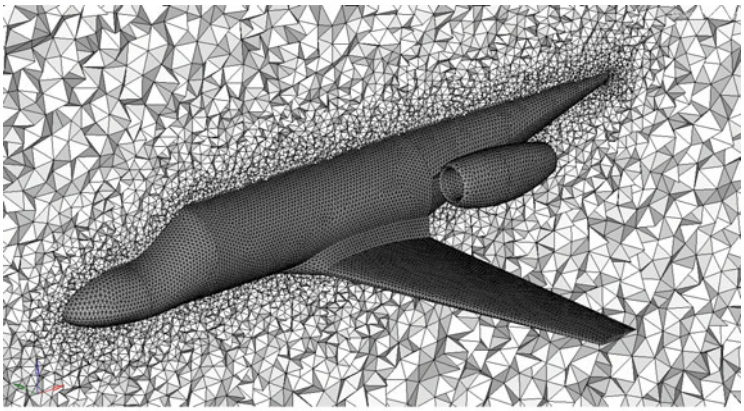
Table 2 Airfoil rotation: CPU time (s) comparison for a single mesh update step

Method	Pre-processing of the background mesh	Construction of the VerBSS matrix	Solve	Total
VerBSS		0.34	5.32	5.66
RBF-VerBSS	0.04	0.33	1.26	1.63

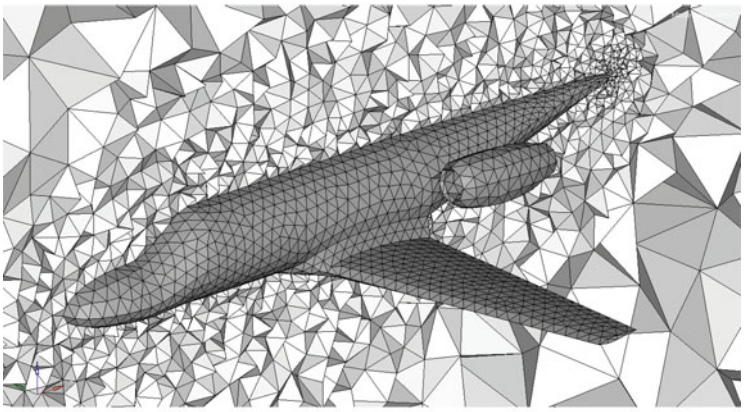
number of the elements is 3,416,966 in the computational mesh. The number of the nodes is 25,885 and the number of the elements is 143,919 in the background mesh.

The fluid mesh is deformed by RerBSS, RBF-VerBSS, and RBFs-MSA, respectively. CPU time for a deformation step for the three methods is shown in Table 3. The number of iterations of the VerBSS method is 379, while the number of iterations of the RBF-VerBSS method is 76. The preprocessing time of RBF-VerBSS is 913 s while that of RBFs-MSA is 9543 s, and the computational cost is too expensive.

As shown in Fig. 9, the maximum number of deformation steps of the three methods are 43, 84, and 73, respectively. Figure 10 illustrates the minimum angle of the mesh changed with time steps in the deformation process of the three methods. When the mesh deformed to the 43th step, the quality is shown in Table 4. The RBF-VerBSS method can keep a better quality than the VerBSS method. Although the quality of the RBFs-MSA method is better than that of the other two, it would suddenly encounter the collapse of an element.



(a)

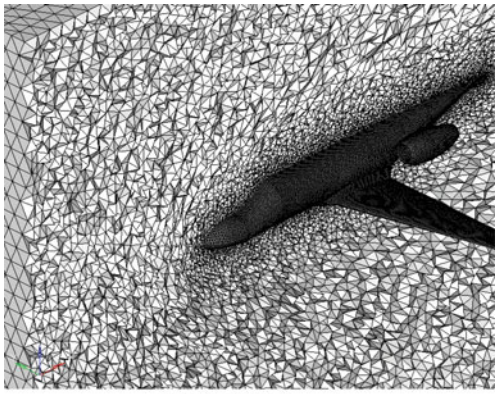


(b)

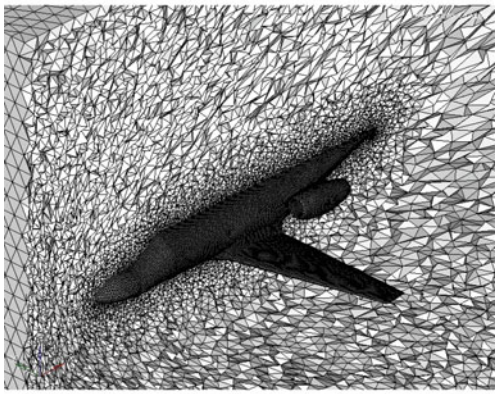
Fig. 8 3D aircraft: cross-sectional views of the computational and background meshes. (**a**) Computational mesh. (**b**) Background mesh

Table 3 3D aircraft: CPU time(s) comparison between different methods in one step

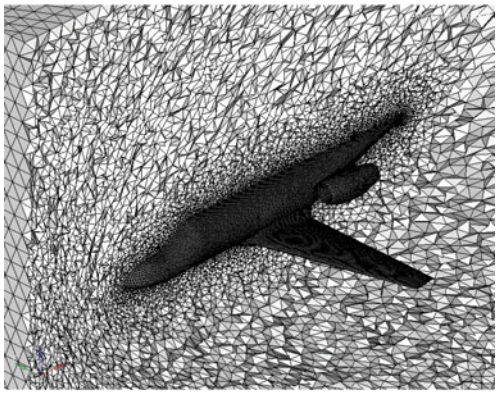
Method	Pre-processing of the background mesh	Construction of the VerBSS matrix	Solve	Total
VerBSS		56.64	166.94	223.58
RBF-VerBSS	10.29	53.41	32.13	95.99
RBFs-MSA	10.19		1.94	12.28



(a) VerBSS (43 steps)



(b) RBF-VerBSS (84 steps)



(c) RBFs-MSA (73 steps)

Fig. 9 3D aircraft: deformed meshes using different methods. (**a**) VerBSS (43 steps). (**b**) RBF-VerBSS (84 steps). (**c**) RBFs-MSA (73 steps)

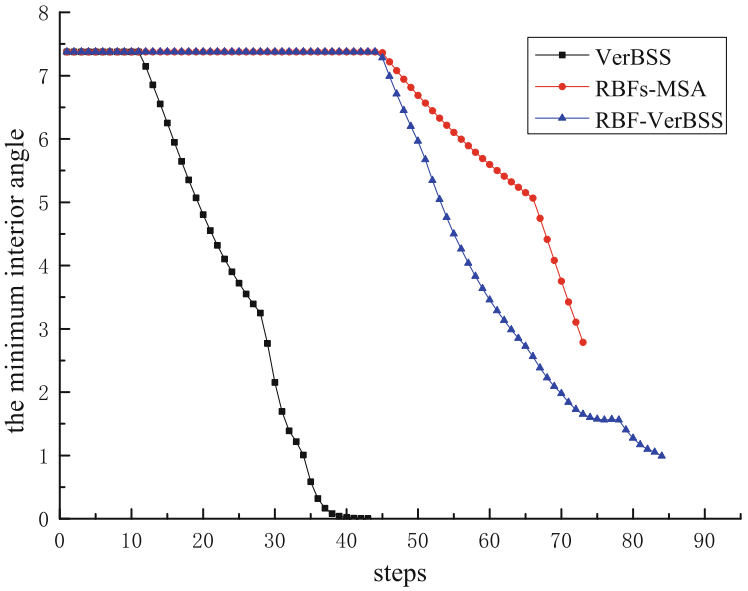


Fig. 10 3D aircraft: the minimum interior angle for different methods

Table 4 3D aircraft: quality comparison between different methods after deformation to the 43th step

Quality	VerBSS	RBFs-MSA	RBF-VerBSS
0.000–0.001	6	0	0
0.001–0.050	135	7	1
0.050–0.100	248	40	39
0.100–0.400	17,330	1926	39,937
0.400–0.700	319,510	364,133	682,063
0.700–1.000	3,079,737	3,040,860	2,694,926

5 Conclusions

The proposed method is based on a background mesh, which is a significant improvement to the original VerBSS method. The pre-deformation of the background mesh improves the initial motion condition of the computational mesh and accelerates the convergence speed. Compared with the RBFs-MSA method, this method retains the connectivity of the mesh. Therefore, it is more robust than the MSA method using ratio of surface or volume to interpolate. The background mesh was calculated by the RBF method because of its less number of elements. In a word, the method in this paper is not limited to interpolation or physical methods, but to exploit the advantages of various methods as much as possible. The various

methods can be combined to achieve a better balance between the efficiency and the robustness as much as possible.

References

1. Batina, J.T.: Unsteady Euler airfoil solutions using unstructured dynamic meshes. AIAA J. **28**(8), 1381–1388 (1990)
2. Bottasso, C.L., Detomi, D., Serra, R.: The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes. Comput. Methods Appl. Mech. Eng. **194**(39–41), 4244–4264 (2005)
3. Chen, J., Bao, H., Wang, T., Desbrun, M., Huang, J.: Numerical coarsening using discontinuous shape functions. ACM Trans. Graph. **37**(4CD), 1–12 (2018)
4. De Boer, A., Van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Comput. Struct. **85**(11–14), 784–795 (2007)
5. Farhat, C., Degand, C., Koobus, B., Lesoinne, M.: Torsional springs for two-dimensional dynamic unstructured fluid meshes. Comput. Methods Appl. Mech. Eng. **163**(1–4), 231–245 (1998)
6. Krishnan, D., Fattal, R., Szeliski, R.: Efficient preconditioning of Laplacian matrices for computer graphics. ACM Trans. Graph. **32**(4), 142:1–142:15 (2013)
7. Lefrançois, E.: A simple mesh deformation technique for fluid–structure interaction based on a submesh approach. Int. J. Numer. Methods Eng. **75**(9), 1085–1101 (2008)
8. Lin, T., Guan, Z., Chang, J., Lo, S.: Vertex-ball spring smoothing: an efficient method for unstructured dynamic hybrid meshes. Comput. Struct. **136**, 24–33 (2014)
9. Liu, X., Qin, N., Xia, H.: Fast dynamic grid deformation based on Delaunay graph mapping. J. Comput. Phys. **211**(2), 405–423 (2006)
10. Liu, Y., Guo, Z., Liu, J.: RBFs-MSA hybrid method for mesh deformation. Chinese J. Aeronaut. **25**(4), 500–507 (2012)
11. Rendall, T.C., Allen, C.B.: Efficient mesh motion using radial basis functions with data reduction algorithms. J. Comput. Phys. **228**(17), 6231–6249 (2009)
12. Selim, M.M., Koomullil, R.: Incremental matrix inversion approach for radial basis function mesh deformation. In: Proceedings of the Fifteenth Annual Early Career Technical Conference, The University of Alabama at Birmingham (2015)
13. Selim, M., Koomullil, R., et al.: Mesh deformation approaches–a survey. J. Phys. Math. **7**(2), 181 (2016)
14. Sheng, C., Allen, C.B.: Efficient mesh deformation using radial basis functions on unstructured meshes. AIAA J. **51**(3), 707–720 (2013)
15. Stein, K., Tezduyar, T., Benney, R.: Mesh moving techniques for fluid-structure interactions with large displacements. J. Appl. Mech. **70**(1), 58–63 (2003)
16. Wang, Y., Qin, N., Zhao, N.: Delaunay graph and radial basis function for fast quality mesh deformation. J. Comput. Phys. **294**, 149–172 (2015)
17. Zhou, X., Li, S., Chen, B.: Spring-interpolation approach for generating unstructured dynamic meshes. Acta Aeronaut. Astronaut. Sin. **31**(7), 1389–1395 (2010)

A Uniform Convergence Analysis for a Bakhvalov-Type Mesh with an Explicitly Defined Transition Point

Thái Anh Nhan

Abstract For singularly perturbed convection-diffusion problems, the truncation error and barrier-function technique for proving parameter-uniform convergence is well-known for finite-difference methods on Shishkin-type meshes (Roos and Linß in Computing, 63 (1999), 27–45). In this paper, we show that it is also possible to generalize this technique to a modification of the Bakhvalov mesh, such that the transition point between the fine and crude parts of the mesh only depends on the perturbation parameter and is defined explicitly. We provide a complete analysis for 1D problems for the simplicity of the present paper, but the analysis can be easily extended to 2D problems. With numerical results for 2D problems we show that the finite-difference discretization on the Bakhvalov-type mesh performs better than the Bakhvalov-Shishkin mesh.

1 Introduction

We consider numerical methods for solving the linear singularly perturbed convection-diffusion problem,

$$\mathcal{L}u := -\varepsilon u'' - b(x)u' + c(x)u = f(x), \quad x \in (0,1), \quad u(0) = u(1) = 0, \qquad (1)$$

where ε is a small positive perturbation parameter, $0 < \varepsilon \ll 1$. We assume that the functions b, c, and f are sufficiently smooth, and that b and c satisfy

$$b(x) \geq \beta > 0, \quad c(x) \geq 0 \quad \text{for } x \in I := [0,1].$$

Then, the boundary value problem (1) has a unique solution, $u \in C^2(I)$, which, generally speaking, exhibits an exponential layer at $x = 0$.

T. A. Nhan (✉)
Department of Mathematics and Science, Holy Names University, Oakland, CA, USA
e-mail: nhan@hnu.edu

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_13

It is well known that in order to achieve ε-uniform convergence with the standard upwind discretization of the problem (1), one should use layer-adapted meshes. Fifty years ago, in 1969, Bakhvalov [4] was the first researcher to introduce the idea of layer-adapted meshes. We refer the reader to a recent comprehensive survey [19], dedicated the 50th anniversary of the original Bakhvalov mesh, in which Roos reviews the important concepts and the development of layer-adapted meshes over last five decades. The monographs [7, 21] are recommended to those who want to explore these meshes and the field of numerical methods for singularly perturbed problems in general.

There are several modifications of the original Bakhvalov mesh, see [5, 7, 19, 20, 25] for instance. When these Bakhvalov-type meshes are used with the upwind finite-difference scheme for the problem (1), one can prove an optimal convergence result,

$$\|u - W^N\| \leq CN^{-1}, \tag{2}$$

where W^N is the upwinding approximation obtained using a mesh with N subintervals, C is a generic positive constant independent of both ε and N, and $\|\cdot\|$ denotes the discrete maximum norm. On the other hand, using a simpler piecewise uniform Shishkin mesh [23], we can prove

$$\|u - W^N\| \leq CN^{-1} \ln N. \tag{3}$$

Among special techniques to prove ε-uniform convergence results like (3) for finite-difference methods on the Shishkin mesh, the barrier-function technique is the most frequently used one. In 1997, Stynes and Roos [24] used this method to prove an almost-second-order convergence for a hybrid central-midpoint scheme. Two years later, the concept of Shishkin-type meshes was introduced in [20] and the barrier-function analysis was extended to these meshes. However, the extension of this technique to Bakhvalov-type meshes has been elusive until recently. This made Linß state the following in his 2003 survey [6] of layer-adapted meshes:

> We are not aware of any results for Bakhvalov-type meshes that make use of this truncation error and barrier function technique.

At that time, it was also known that the estimate (2) for Bakhvalov-type meshes could be proved by another method, the hybrid-stability approach [1, 2, 8], which made this theoretical gap even more puzzling (another way to prove the same result is to use the relatively new preconditioning method [12, 13, 17, 26]).

However, we have recently found in the research literature the first uniform convergence proof by Liseikin, appeared in [9, Section 7.2], that makes use of the barrier function for the semilinear problems of type (1) in the context of layer-resolving grids for which the result can be deduced to the Bakhvalov mesh. A general theory of grid generation and its analysis can be found in the monograph [10]; whereas a comprehensive characteristic comparison between

various well-known and novel layer-resolving grids is discussed by Liseikin and Karasuljić in [11].

The original Bakhvalov mesh and its modifications by Vulanović [25] are generated by $C^1[0, 1]$-functions which redistribute uniformly spaced points so that a mesh dense in the boundary layer is created. Due to the smooth mesh-generating function, the mesh gradually transitions from a fine part in the layer to the coarse part outside the layer. The corresponding transition point is defined implicitly, although in the case of the simplest Vulanović's mesh-generating function, the transition point can be found by solving a quadratic equation. As opposed to this, the Bakhvalov-type meshes of [5, 7, 20] have a pre-defined transition point, which is similar, or even identical, to the transition point of the Shishkin mesh. The mesh-step size of such meshes changes abruptly around the transition point, but depending on how the fine part of the mesh is defined, the optimal ε-uniform accuracy (2) of the upwind scheme may be preserved.

The purpose of this paper is to show that the truncation error and barrier-function technique used for Shishkin-type meshes from [20] can be generalized to the Bakhvalov-type meshes with an explicitly defined transition point. We demonstrate this by considering the mesh from [5], which was also used in [3, 18]. Because of some characteristics, this mesh is closer to the original Bakhvalov mesh than the Bakvalov-Shishkin mesh of [7, 20]. It is worth noting that the proof technique and the barrier function we use here defer from that of Liseikin (cf. [9, Section 7.2]). The key ingredient of our approach is the novel analysis, recently presented in [15, 16], which is specially designed for the Bakhvalov meshes with smooth mesh-generating functions.

Furthermore, the most significant advantage of our new barrier-function approach is that its analysis can be extended to 2D problems:

$$\begin{aligned} -\varepsilon\Delta u - b_1(x, y)u_x - b_2(x, y)u_y + c(x, y)u &= f(x, y) \quad \text{on } \Omega = (0, 1)^2, \\ u &= 0 \qquad\quad \text{on } \Gamma = \partial\Omega. \end{aligned} \tag{4}$$

More importantly, to our best knowledge, this is the only technique which can be used to analyze a finite-difference scheme on a Bakhvalov-type mesh for a higher-dimensional problems like (4) (see [16]). For the simplicity and the purpose of this proceeding, we present only a detailed analysis for 1D problems because this can be easily extended to 2D by following the approach proposed recently in [16]. We then report numerical performance for a 2D test problem for which the computed errors by the upwind discretization on the Bakhvalov-type mesh is convergent uniformly for all values of ε.

In the next section, we list some preliminary facts about the solution u of the problem (1) and introduce the upwind scheme for discretizing the problem. The discretization mesh is described in Sect. 3. Then, in Sect. 4, we highlight the distinguishable improvements of our new approach by comparing our assumptions to those used in the existing barrier-function proof by Roos and Linß [20] for Shishkin-type meshes. In Sect. 5, we provide the truncation error analysis and

the barrier-function choice to obtain the main result (2) for 1D problems. Finally, numerical results for a 2D test problem are presented in Sect. 6 together with a brief concluding remark in the last section.

2 Preliminaries

The following decomposition of u, [7, Theorem 3.48], is often used in the error analysis of numerical methods for (1):

$$u(x) = s(x) + y(x),$$

$$|s^{(k)}(x)| \le C\Big(1 + \varepsilon^{2-k}\Big), \quad |y^{(k)}(x)| \le C\varepsilon^{-k}\mathrm{e}^{-\beta x/\varepsilon}, \tag{5}$$

$$x \in I, \quad k = 0, 1, 2, 3.$$

Moreover, the layer component, y, satisfies a homogeneous differential equation,

$$\mathcal{L}y(x) = 0, \quad x \in (0, 1). \tag{6}$$

Let I^N denote an arbitrary mesh with mesh points x_i, $i = 0, 1, \ldots, N$, such that $0 = x_0 < x_1 < \ldots < x_N = 1$. Let $h_i = x_i - x_{i-1}$, $i = 1, 2, \ldots, N$, be the mesh-step sizes and let $\hbar_i = (h_i + h_{i+1})/2$, $i = 1, 2, \ldots, N-1$. Mesh functions on I^N are denoted by $W^N = \big(W_i^N\big)$, $U^N = \big(U_i^N\big)$, etc. If g is a function defined on I, we write g_i instead of $g(x_i)$ and g^N for the corresponding mesh function.

The upwind finite-difference scheme is used to discretize the problem (1) on I^N,

$$\begin{aligned} U_0^N &= 0, \\ \mathcal{L}^N U_i^N &:= -\varepsilon D''U_i^N - b_i D^+ U_i^N + c_i U_i^N = f_i\,, \quad i = 1, 2, \ldots, N-1, \\ U_N^N &= 0, \end{aligned} \tag{7}$$

where

$$D''W_i^N = \frac{1}{\hbar_i}\Big(D^+W_i^N - D^-W_i^N\Big),$$

and

$$D^+W_i^N = \frac{W_{i+1}^N - W_i^N}{h_{i+1}}, \quad D^-W_i^N = \frac{W_i^N - W_{i-1}^N}{h_i}.$$

It is easy to see that the operator $\mathcal{L}^N$ satisfies the discrete maximum principle. Therefore, the discrete problem (7) has a unique solution U^N.

Let

$$\tau_i[g] = \mathcal{L}^N g_i - (\mathcal{L}g)_i, \quad i = 1, 2, \ldots, N-1,$$

for any $C^2(I)$-function g. In particular, $\tau_i[u]$ is the truncation error of the finite-difference operator $\mathcal{L}^N$ and

$$\tau_i[u] = \mathcal{L}^N u_i - \mathcal{L}^N W_i^N = \mathcal{L}^N \left(u - W^N\right)_i.$$

By Taylor's expansion we get that

$$|\tau_i[u]| \le C h_{i+1} \left(\varepsilon \left\|u'''\right\|_i + \left\|u''\right\|_i\right), \tag{8}$$

where $\|g\|_i := \max_{x_{i-1} \le x \le x_{i+1}} |g(x)|$ for any $g \in C(I)$. Recall from the introduction that C is a positive generic constant independent of ε and N. Some specific constants of this kind will be subscripted below.

3 A Bakhvalov-Type Mesh with an Explicitly Defined Transition Point

The specific Bakhvalov-type mesh we are interested in is defined by

$$x_i = \begin{cases} a\varepsilon\phi(t_i), & i = 0, 1, \ldots, J, \\ \sigma + 2(1-\sigma)(t_i - 1/2), & i = J+1, J+2, \ldots, N, \end{cases} \tag{9}$$

with a positive constant a, $\phi(t) = \phi_B(t) := -\ln[1 - 2(1-\varepsilon)t]$, $t_i = i/N$, $J = N/2$, and

$$\sigma = \sigma_B := a\varepsilon \ln(1/\varepsilon) = x_J, \tag{10}$$

cf. [3, 5, 18]. This mesh shares the following characteristics with the original Bakhvalov mesh: a logarithmic function to generate the points in the layer, and a transition point (defined in (10)) of the order $O(-\varepsilon \ln \varepsilon)$. Furthermore, the last mesh step, h_J, in the layer region grows logarithmically as $\varepsilon \to 0$, and not like $O(\varepsilon)$, which is the case with the Bakhvalov-Shishkin mesh (or other Shishkin-type meshes in [7, 20]). The Bakhvalov-Shishkin mesh is like in (9), where

$$\phi(t) = \phi_{BS}(t) := -\ln[1 - 2(1 - 1/N)t] \quad \text{and} \quad \sigma = \sigma_S := a\varepsilon \ln N = x_J,$$

σ_S being the standard Shishkin transition point.

The mesh sizes in the layer region of the mesh (9)–(10) are estimated in the following lemma. This is what makes this Bakhvalov-type mesh different from Shishkin-type meshes.

Lemma 1 *The mesh widths in the layer regions of the Bakhvalov-type mesh defined by* (9) *and* (10) *satisfy*

$$h_{i-1} \le h_i \le CN^{-1}, \quad i = 2, 3, \ldots, J, \tag{11}$$

and in particular

$$h_i \le a\varepsilon, \quad i = 1, 2, \ldots, J-1. \tag{12}$$

Furthermore,

$$\mathrm{e}^{-\beta x_{J-1}/\varepsilon} = \left(\varepsilon + 2(1-\varepsilon)N^{-1}\right)^{a\beta} \le \left(\varepsilon + 2N^{-1}\right)^{a\beta} \tag{13}$$

and

$$\mathrm{e}^{-\beta x_J/\varepsilon} = (\varepsilon)^{a\beta}. \tag{14}$$

Proof By the definition of $\phi_B(t)$, we have $\phi_B'(t) = \dfrac{2(1-\varepsilon)}{1-2(1-\varepsilon)t}$, thus, $\phi_B(t)$ is monotonically increasing for $t \in [0, 1/2]$. Therefore, $h_{i-1} \le h_i$, $i = 2, 3, \ldots, J-1$, and

$$h_J = a\varepsilon \int_{t_{J-1}}^{t_J} \phi'(s)\,\mathrm{d}s \le \frac{a\varepsilon}{N}\phi'(t_J) = \frac{a\varepsilon}{N} \times \frac{2(1-\varepsilon)}{1-2(1-\varepsilon)t_J} \le \frac{a\varepsilon}{N} \times \frac{2}{\varepsilon} = 2aN^{-1},$$

which gives (11).

For the estimate in (12), we have

$$\begin{aligned} h_i &= a\varepsilon \int_{t_{i-1}}^{t_i} \phi'(s)\,\mathrm{d}s \le \frac{a\varepsilon}{N} \max_{t\in[t_{i-1},t_i]} \phi'(t) = \frac{a\varepsilon}{N} \times \frac{2(1-\varepsilon)}{1-2(1-\varepsilon)t_i} \\ &\le \frac{a\varepsilon}{N} \times \frac{2}{1/(1-\varepsilon) - 2t_{J-1}/(1-\varepsilon)} \le \frac{a\varepsilon}{N} \times N = a\varepsilon. \end{aligned}$$

It is an easy calculation to get the inequalities (13) and (14). □

4 The Barrier-Function Analysis of Roos and Linß for Shishkin-Type Meshes

Here, we only list the assumptions used in [20] to prove ε-uniform convergence on Shishkin-type meshes. The proof is based on the barrier-function technique. In [20], Shishkin-type meshes are defined as generated in the layer by a function ϕ which satisfies the following conditions:

$$\phi' > 0, \quad \max \phi' \leq CN \tag{15}$$

and

$$\int_0^{1/2} \left[\phi'(s)\right]^2 \mathrm{d}s \leq CN. \tag{16}$$

The transition point of Shishkin-type meshes is σ_S. An additional assumption is that

$$\varepsilon \leq CN^{-1}, \tag{17}$$

which, together with (15), is crucial for obtaining

$$h_i \leq a\varepsilon[\phi(t_i) - \phi(t_{i-1})] \leq a\varepsilon N^{-1} \max \phi' \leq C\varepsilon \leq CN^{-1}, \quad i = 1, 2, \ldots, J.$$

The property

$$h_i \leq C\varepsilon, \quad i = 1, 2, \ldots, J, \tag{18}$$

is also important for Shishkin-type meshes and is used in later stages of the error analysis in [20].

Strictly speaking, the Shishkin-type meshes do not provide ε-uniform convergence because (17) is assumed. More precisely, in the case of the Bakhvalov-Shishkin mesh, for instance, one can only prove that

$$\|u - u^N\| \leq C\left(\varepsilon + N^{-1}\right), \tag{19}$$

cf. [7]. As opposed to this, all mesh steps in the layer region of the Bakhvalov-type mesh defined in Sect. 3 are of order $O(N^{-1})$, without the assumption (17), as shown in (11). Therefore, the mesh with a Bakhvalov-type transition point is a theoretically favored choice.

However, the Bakhvalov-type mesh in Sect. 3 does not fulfill (16) and (18). This is because

$$\int_0^{1/2} \left[\phi_B'(s)\,\mathrm{d}s\right]^2 = \frac{2}{\varepsilon} - 4 + \varepsilon = O(\varepsilon^{-1}), \quad \text{and} \quad h_J = a\varepsilon \ln\left(1 + 2\frac{1-\varepsilon}{\varepsilon N}\right).$$

As a result, a new barrier function approach is needed for the Bakhvalov-type mesh. We present this in the next sections.

5 The Error Analysis

We first estimate the truncation error $\tau_i[u]$ for $i = 1, 2, \ldots, N-1$. This is done in Lemma 2. Then, the barrier function is introduced in Lemma 3 and the discrete maximum principle is applied on the entire interval $[0, 1]$.

Let

$$\bar{y}_i^N := \prod_{j=1}^{i} \left(1 + \frac{\beta h_j}{2\varepsilon}\right)^{-1}.$$

It holds that

$$\bar{y}_i^N \geq \prod_{j=1}^{i} \mathrm{e}^{-\beta h_j/(2\varepsilon)} = \mathrm{e}^{-\beta x_i/(2\varepsilon)}, \quad i = 1, 2, \ldots, N.$$

Lemma 2 *Let $a\beta \geq 2$. The truncation error of the upwind discretization of the problem (1) on the Bakhvalov-type mesh defined in (9) and (10) satisfies the following.*

- *For $i \geq J$,*

$$|\tau_i[u]| \leq CN^{-1}. \tag{20}$$

- *For $i \leq J-2$,*

$$|\tau_i[u]| \leq C\left(N^{-1} + \varepsilon^{-1}\bar{y}_i^N N^{-1}\right). \tag{21}$$

- *For $i = J-1$,*

$$|\tau_i[u]| \leq \begin{cases} C\left(N^{-1} + \varepsilon^{-1}\bar{y}_i^N N^{-1}\right), & \text{when } h_{i+1} \leq \varepsilon, \\ C\left(N^{-1} + h_{i+1}^{-1}\bar{y}_i^N N^{-1}\right), & \text{when } h_{i+1} > \varepsilon. \end{cases} \tag{22}$$

Proof We only provide the proof for the singular component y of u, since $|\tau_i[s]|$ is easy to estimate. To prove (20) for $i \geq J+1$, we apply (8) to y. Then we have

$$|\tau_i[y]| \leq Ch_{i+1}\left(\varepsilon \left\|y'''\right\|_i + \left\|y''\right\|_i\right) \leq CN^{-1}\varepsilon^{-2}\mathrm{e}^{-\beta x_J/\varepsilon} \leq CN^{-1},$$

where we have used (14) and $a\beta \geq 2$ in the last inequality.

To prove (20) for $i = J$, we consider two cases, $\varepsilon \le N^{-1}$ and $\varepsilon > N^{-1}$. For $i = J$ and $\varepsilon \le N^{-1}$, we use the truncation error estimate in the form of $\tau_i[y] = \mathcal{L}^N y$, which is valid because of (6). Thus, we have

$$|\tau_i[y]| \le P_i + Q_i + R_i,$$

where

$$P_i = \varepsilon \left| D^- y_i \right|, \quad Q_i = b_i \left| D' y_i \right|, \quad \text{and} \quad R_i = c_i |y_i|.$$

We can bound P_J from above as follows. Since $\hbar_J \ge h_{J+1}/2 \ge CN^{-1}$, we get $\hbar_J^{-1} \le CN$ and, invoking (13),

$$P_J \le C\hbar_J^{-1} \mathrm{e}^{-\beta x_{J-1}/\varepsilon} \le CN\left(\varepsilon + 2N^{-1}\right)^{a\beta} \le CN^{-1}. \tag{23}$$

Analogous arguments apply to Q_J and R_J, implying that $|\tau_J[y]| \le CN^{-1}$.

For $i = J$ and $\varepsilon > N^{-1}$, we get that $h_J \le C\varepsilon$ because of (11). Therefore, we continue as follows:

$$\begin{aligned} |\tau_J[y]| &\le Ch_{J+1}\left(\varepsilon \left\| y''' \right\|_J + \left\| y'' \right\|_J\right) \le CN^{-1}\varepsilon^{-2}\mathrm{e}^{-\beta x_{J-1}/\varepsilon} \\ &\le CN^{-1}\varepsilon^{-2}\mathrm{e}^{-\beta x_J/\varepsilon} \le CN^{-1}, \end{aligned}$$

where (14) is used in the last step.

We can combine the estimates in (21) and the first case of (22) as follows. For $i \le J-1$, we have $h_i \le a\varepsilon$ because of (12) and $h_J \le C\varepsilon$ by the first case of (22). Hence,

$$\begin{aligned} |\tau_i[y]| &\le Ch_{i+1}\left(\varepsilon \left\| y''' \right\|_i + \left\| y'' \right\|_i\right) \le N^{-1}\varepsilon\phi'(t_{i+1})\left(\varepsilon^{-2}\mathrm{e}^{-\beta x_{i-1}/\varepsilon}\right) \\ &\le C\varepsilon^{-1}N^{-1}\left[\phi'(t_{i+1})\mathrm{e}^{-\beta x_{i+1}/2\varepsilon}\right]\mathrm{e}^{-\beta x_i/2\varepsilon} \\ &\le C\varepsilon^{-1}N^{-1}[1-2(1-\varepsilon)t_{i+1}]^{a\beta-1}\mathrm{e}^{-\beta x_i/2\varepsilon} \\ &\le C\varepsilon^{-1}N^{-1}\mathrm{e}^{-\beta x_i/2\varepsilon} \le C\varepsilon^{-1}N^{-1}\bar{y}_i^N. \end{aligned}$$

Lastly, when $i = J-1$ and $h_J > \varepsilon$, this means that $\max\{\varepsilon, h_J\} = h_J$. Then, $\varepsilon \le CN^{-1}$, again because of (11), and we can modify the approach in (23) and use (13) to get

$$\begin{aligned} P_{J-1} &\le C\hbar_{J-1}^{-1}\mathrm{e}^{-\beta x_{J-2}/\varepsilon} \le Ch_J^{-1}\mathrm{e}^{-\beta x_{J-1}/(2\varepsilon)}\mathrm{e}^{-\beta x_{J-1}/(2\varepsilon)} \\ &\le Ch_J^{-1}\bar{y}_{J-1}^N\left(\varepsilon + 2N^{-1}\right)^{a\beta/2} \le Ch_J^{-1}\bar{y}_{J-1}^N N^{-1}. \end{aligned}$$

The same argument applied to R_{J-1} and Q_{J-1} gives

$$|\tau_{J-1}[y]| \le Ch_J^{-1}\bar{y}_{J-1}^N N^{-1},$$

which completes the proof. □

We next form the barrier function γ_i (see also in [14, 15]),

$$\gamma_i = \gamma_i^{(1)} + \gamma_i^{(2)}, \quad i = 0, 1, \ldots, N,$$

with

$$\gamma_i^{(1)} = C_1 N^{-1}(1 - x_i) \quad \text{and} \quad \gamma_i^{(2)} = C_2 \bar{y}_i^N N^{-1}.$$

Lemma 3 *There exist sufficiently large constants C_1 and C_2 such that*

$$\mathcal{L}^N \gamma_i \ge \kappa_i \ge |\tau_i[u]|, \quad i = 1, 2, \ldots, N-1, \tag{24}$$

where

$$\kappa_i = C_1 N^{-1} + C_2[\max\{\varepsilon, h_{i+1}\}]^{-1} \bar{y}_i^N N^{-1}.$$

Proof It is an easy calculation (see, e.g., [14]) to verify the first inequality in (24),

$$\mathcal{L}^N \gamma_i \ge \kappa_i.$$

Then, from Lemma 2, it is clear that

$$|\tau_i[u]| \le CN^{-1} \le C_1 N^{-1} \le \kappa_i \le \mathcal{L}^N \gamma_i, \quad i = J, J+1, \ldots, N-1.$$

For $i \le J-2$, we have from (12) that $h_{i+1} \le C\varepsilon$ and

$$|\tau_i[u]| \le C\left(N^{-1} + \varepsilon^{-1}\bar{y}_i^N N^{-1}\right) \le C_1 N^{-1} + C_2 \varepsilon^{-1} \bar{y}_i^N N^{-1} \le \kappa_i \le \mathcal{L}^N \gamma_i. \tag{25}$$

It remains to consider $i = J-1$. If $h_{i+1} \le \varepsilon$, we have the same situation as above and the estimate (25) is achieved for $i = J-1$. On the other hand, when $h_{i+1} > \varepsilon$, it is clear from (22) that

$$|\tau_{J-1}[u]| \le C\left(N^{-1} + h_J^{-1}\bar{y}_i^N N^{-1}\right).$$

Therefore,

$$|\tau_{J-1}[u]| \le C\left(N^{-1} + h_J^{-1}\bar{y}_{J-1}^N N^{-1}\right) \le C_1 N^{-1} + C_2 h_J^{-1}\bar{y}_{J-1}^N N^{-1} \le \kappa_{J-1} \le \mathcal{L}^N \gamma_{J-1},$$

which completes the proof. □

Combining Lemmas 2 and 3 together with the discrete maximum principle, we arrive at the main result.

Theorem 1 *Let u be the solution of the continuous problem (1) and let U^N be the solution of the discrete problem (7) on the Bakhvalov-type mesh (9)–(10). Then the following error estimate is satisfied:*

$$|u_i - U_i^N| \le CN^{-1}, \quad i = 0, 1, \ldots, N.$$

6 Numerical Results for 2D Problems

For the two-dimensional problem (4), we employ the natural extension of the method (7): an upwind finite-difference scheme on a tensor-product grid of the one-dimensional Bakhvalov-type meshes defined in Sect. 3. Although we do not provide an analysis for the two-dimensional problem (4), it is an easy extension from the above 1D analysis to its 2D analogue in light of the newly tailored barrier-function technique for the original Bakhvalov mesh given in [16]. Therefore, we only present, in this section, the numerical results obtained by the upwind discretization on the Bakhvalov-type mesh to the following test problem taken from [16, Example 1],

$$\begin{aligned} -\varepsilon\Delta u - (x+2)u_x - (y^3+3)u_y + u &= f(x,y) \quad \text{on } \Omega = (0,1)^2, \\ u &= 0 \qquad\quad \text{on } \Gamma = \partial\Omega, \end{aligned} \tag{26}$$

where $f(x, y)$ is chosen so that

$$u(x,y) = \cos\left(\frac{\pi x}{2}\right)\left(1 - \mathrm{e}^{-2x/\varepsilon}\right)(1-y)^3\left(1 - \mathrm{e}^{-3y/\varepsilon}\right)$$

is the exact solution. The discretization errors, E_N, and the rate of convergence, $\rho \approx (\ln E_N - \ln E_{2N})/\ln 2$, are shown in Table 1. We compare this result to the one generated on the Bakhvalov-Shishkin mesh—shown in Table 2. As we can clearly observe, the uniform convergence can be seen in Table 1 for all values of ε. This agrees with the theoretical justification. By contrast, in Table 2 the rate of convergence deteriorates when N increases for $\varepsilon = 0.1$ and $\varepsilon = 0.01$ because of (19).

Table 1 Problem (26): errors (top row) and convergence rates (below row) for the upwind discretization on the modification of the Bakhvalov mesh with $a = 2$

ε	$N = 2^6$	$N = 2^7$	$N = 2^8$	$N = 2^9$	$N = 2^{10}$	$N = 2^{11}$
10^{-1}	2.079e−02	1.063e−02	5.377e−03	2.704e−03	1.356e−03	6.787e−04
	0.97	0.98	0.99	1.00	1.00	–
10^{-2}	2.834e−02	1.419e−02	7.092e−03	3.543e−03	1.771e−03	8.851e−04
	1.00	1.00	1.00	1.00	1.00	–
10^{-3}	2.981e−02	1.534e−02	7.787e−03	3.922e−03	1.968e−03	9.857e−04
	0.96	0.98	0.99	0.99	1.00	–
10^{-4}	3.038e−02	1.567e−02	7.949e−03	4.003e−03	2.008e−03	1.006e−03
	0.96	0.98	0.99	0.99	1.00	–
10^{-5}	3.049e−02	1.571e−02	7.971e−03	4.014e−03	2.014e−03	1.009e−03
	0.96	0.98	0.99	1.00	1.00	–
10^{-6}	3.068e−02	1.572e−02	7.974e−03	4.015e−03	2.014e−03	1.009e−03
	0.96	0.98	0.99	1.00	1.00	–

Table 2 Problem (26): errors (top row) and convergence rates (below row) for the upwind discretization on the Bakhvalov mesh with a Shishkin-type transition point with $a = 2$

ε	$N = 2^6$	$N = 2^7$	$N = 2^8$	$N = 2^9$	$N = 2^{10}$	$N = 2^{11}$
10^{-1}	1.984e−02	1.856e−02	1.729e−02	1.572e−02	1.385e−02	1.178e−02
	0.10	0.10	0.14	0.18	0.23	–
10^{-2}	2.797e−02	1.427e−02	7.175e−03	3.553e−03	1.930e−03	1.154e−03
	0.97	0.99	1.01	0.88	0.74	–
10^{-3}	2.957e−02	1.537e−02	7.803e−03	3.926e−03	1.968e−03	9.856e−04
	0.94	0.98	0.99	1.00	1.00	–
10^{-4}	3.001e−02	1.563e−02	7.949e−03	4.005e−03	2.009e−03	1.006e−03
	0.94	0.98	0.99	1.00	1.00	–
10^{-5}	3.005 02	1.565e−02	7.964e−03	4.013e−03	2.014e−03	1.009e−03
	0.94	0.97	0.99	0.99	1.00	–
10^{-6}	3.005e−02	1.566e−02	7.966e−03	4.014e−03	2.014e−03	1.009e−03
	0.94	0.97	0.99	0.99	1.00	–

7 Conclusion

We presented an alternative barrier-function approach for uniform convergence analysis of the finite-difference discretization on a Bakhvalov-type mesh. In particular, the transfer of our analysis to 2D problems answered an open question recently posed in [22, Question 6]. Numerical experiments show that, from the theoretical and computational view, a Bakhvalov-type transition point is the favoured choice because the computed errors by the upwind scheme on this mesh are convergent, uniformly for all values of ε as seen in Sect. 6.

Acknowledgments This research was supported by the Faculty Development Program, Holy Names University. The author would like to thank an anonymous referee for his suggestions that brought my attention to the barrier-function uniform convergence proof of Liseikin.

References

1. Andreev, V.B., Kopteva, N.V.: On the convergence, uniform with respect to a small parameter, of monotone three-point finite difference approximations. Differ. Equ. **34**(7), 921–929 (1998)
2. Andreev, V.B., Savin, I.A.: The uniform convergence with respect to a small parameter of A. A. Samarskii's monotone scheme and its modification. Comput. Math. Phys. **35**, 581–591 (1995)
3. Apel, Th., Lube, G.: Anisotropic mesh refinement for a singularly perturbed reaction diffusion model problem. Appl. Numer. Math. **26**, 415–433 (1998)
4. Bakhvalov, N.S.: The optimization of methods of solving boundary value problems with a boundary layer. USSR Comput. Math. Math. Phys. **9**, 139–166 (1969)
5. Boglaev, I.P.: The numerical solution of a nonlinear boundary value problem with a small parameter effecting the highest derivative (in Russian). Zh. Vychisl. Mat. Mat. Fiz. **24**, 1649–1656 (1984)
6. Linß, T.: Layer-adapted meshes for convection-diffusion problems. Comput. Methods Appl. Mech. Eng. **192**(9–10), 1061–1105 (2003)
7. Linß, T.: Layer-Adapted Meshes for Reaction-Convection-Diffusion Problems. Lecture Notes in Mathematics, vol. 1985. Springer, Berlin (2010)
8. Linß, T., Roos, H.-G., Vulanović, R.: Uniform pointwise convergence on Shishkin-type meshes for quasilinear convection-diffusion problems. SIAM J. Numer. Anal. **38**, 897–912 (2001)
9. Liseikin, V.D.: Layer Resolving Grids and Transformations for Singular Perturbation Problems, p. 284. VSP, Utrecht (2001)
10. Liseikin, V.D.: Grid Generation Methods, 3rd edn, p. 530. Springer, Berlin (2017)
11. Liseikin, V.D., Karasuljić, S.: Numerical analysis of grid-clustering rules for problems with power of the first type boundary layers. Comput. Technol. **25**(1), 49–65 (2020)
12. Nhan, T.A., Vulanović, R.: Uniform convergence on a Bakhvalov-type mesh using preconditioning approach. Technical report (2015). arXiv: 1504.04283 math.NA
13. Nhan, T.A., Vulanović, R.: Preconditioning and uniform convergence for convection-diffusion problems discretized on Shishkin-type meshes. Adv. Numer. Anal. **2016**, Article ID 2161279 (2016)
14. Nhan, T.A., Vulanović, R.: A note on a generalized Shishkin-type mesh. Novi Sad J. Math. **48**(2), 141–150 (2018). https://doi.org/10.30755/NSJOM.07880
15. Nhan, T.A., Vulanović, R.: Analysis of the truncation error and barrier-function technique for a Bakhvalov-type mesh. Electron. Trans. Numer. Anal. **51**, 315–330 (2019)
16. Nhan, T.A., Vulanović, R.: The Bakhvalov mesh: a complete finite-difference analysis of two-dimensional singularly perturbed convection-diffusion problems. Numer. Algor. (2020). https://doi.org/10.1007/s11075-020-00964-z
17. Nhan, T.A., Stynes, M., Vulanović, R.: Optimal uniform-convergence results for convection-diffusion problems in one dimension using preconditioning. J. Comput. Appl. Math. **338**, 227–238 (2018). https://doi.org/10.1016/j.cam.2018.02.012
18. Roos, H.-G.: Error estimates for linear finite elements on Bakhvalov-type meshes. Appl. Math. **51**, 63–72 (2006)
19. Roos, H.-G.: Layer-adapted meshes: Milestones in 50 years of history, Sept 2019. arXiv: 1909.08273v1 math.NA
20. Roos, H.-G., Linß, T.: Sufficient conditions for uniform convergence on layer-adapted grids. Computing **63**, 27–45 (1999)

21. Roos, H.-G., Stynes, M., Tobiska, L.: Numerical Methods for Singularly Perturbed Differential Equations. Springer Series in Computational Mathematics, vol. 24, 2nd edn. Springer, Berlin (2008)
22. Roos, H.-G., Stynes, M.: Some open questions in the numerical analysis of singularly perturbed differential equations. Comput. Methods Appl. Math. **15**, 531–550 (2015)
23. Shishkin, G.I.: Grid approximation of singularly perturbed elliptic and parabolic equations (in Russian). Second Doctoral Thesis, Keldysh Institute, Moscow (1990)
24. Stynes, M., Roos, H.-G.: The midpoint upwind scheme. Appl. Numer. Math. **23**, 361–374 (1997)
25. Vulanović, R.: On a numerical solution of a type of singularly perturbed boundary value problem by using a special discretization mesh. Univ. u Novom Sadu Zb. Rad. Prir. Mat. Fak. Ser. Mat. **13**, 187–201 (1983)
26. Vulanović, R., Nhan, T.A.: Uniform convergence via preconditioning. Int. J. Numer. Anal. Model. Ser. B **5**, 347–356 (2014)

On a Comprehensive Grid for Solving Problems Having Exponential or Power-of-First-Type Layers

V. D. Liseikin, S. Karasuljic, A. V. Mukhortov, and V. I. Paasonen

Abstract This paper describes an explicit approach for generating layer-resolving grids in problems having exponential or power-of-first-type layers. The grids are generated on the basis of qualitative estimates of solution derivatives in the layers of one-dimensional singularly perturbed problems. The paper presents results of numerical experiments, using appropriate grids and high-order schemes, for two-point boundary-value problems and two-dimensional Navier–Stokes equations.

1 Introduction

The paper aims at developing high-order ε-uniform adaptive algorithms suitable for solving a singularly perturbed problem of the type

$$-[\varepsilon + d(x)]^{\alpha} u'' + a(x,u)u' + f(x,u) = 0 \qquad \alpha > 0,\ d(x) \geq 0\ ,\ 0 < x < 1\ ,$$
$$u(0) = A_0\ , \qquad u(1) = A_1\ , \tag{1}$$

and Navier–Stokes equations modelling a viscous gas flow over a flat plate. Problem (1) for $d(x) = 0$ is analyzed analytically in [1–3], where it is shown that its solutions for the specific functions $a(x,u)$ and $f(x,u)$ may have exponential boundary and power-of-type-2 interior layers; while in [4] it is demonstrated that its solutions for the more arbitrary functions $d(x)$, $a(x,u)$, and $f(x,u)$ may also have other layers, in particular, power-of-type-1, logarithmic, and mixed boundary and interior layers.

V. D. Liseikin (✉) · V. I. Paasonen
Institute of Computational Technologies SB RAS, Novosibirsk, Russia

S. Karasuljic
Faculty of Sciences and Mathematics, University of Tuzla, Tuzla, Bosnia and Herzegovina

A. V. Mukhortov
Novosibirsk State University, Novosibirsk, Russia

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_14

The algorithms advocated in this paper for solving (1) are based on layer-damping coordinate transformations $x(\xi, \varepsilon)$: $[0, 1] \rightarrow [0, 1]$ in compliance with a basic principle: they are to eliminate singularities of high order of solutions $u(x, \varepsilon)$; i.e., the high-order derivatives of any concrete solution with respect to the new coordinate ξ are to have the following bounds:

$$\left| \frac{\mathrm{d}^j}{\mathrm{d}\xi^j} u[x(\xi, \varepsilon), \varepsilon] \right| \leq M \ , \quad j \leq n \ , \quad 0 \leq \xi \leq 1 \ , \tag{2}$$

where the constant M is independent of the parameter ε, and the number n is dependent on the order of the approximation of the problem: the higher the order, the larger the number n will be. The layer-damping coordinate transformations $x(\xi, \varepsilon)$: $[0, 1] \rightarrow [0, 1]$, which eliminate singularities, by formula $x_i = x(i/N, \varepsilon)$ generate the layer-resolving grids x_i, $i = 0, \ldots, N$, $x_0 = 0$, $x_N = 1$, whose nodes cluster in the layers. This principle for $n \leq 3$ is implemented in [4] for solving problem (1) with the functions $d(x)$, $a(x, u)$, and $f(x, u)$ by using a simple upwind scheme of first order. It is proven in this publication that the ε-uniform first order convergence of numerical solutions to some problems having diverse types of layers is achieved using an upwind scheme of first order and corresponding diverse layer-resolving grids. However, for guaranteeing higher-order ε-uniform convergence of solutions by high-order algorithms, the transformations developed for the upwind scheme are not suitable; first, they do not provide grid clustering on the entire layers for the higher derivatives of solutions presented in errors of discretizations by schemes of higher order, as these layers are wider than the layers for lower derivatives of solutions; and, second, in contrast to the upwind scheme, many schemes of high order are not subject to the principle of inverse monotonicity, which allows one to establish a connection between the solution and truncation errors, and so to prove high-order ε-uniform convergence. Moreover, convergence for non-monotone problems, in particular quasi-linear problems, can, in general, be demonstrated by numerical experiments only, although it is quite likely that, for these cases, coordinate transformations for generating grids providing high-order ε-uniform convergence can be found among those satisfying (2) for large n.

Estimates (2) may guarantee high-order ε-uniform convergence in the logical domain ξ, as the truncation errors will be ε-uniformly bounded when $n \geq p + 2$, where p is the order of the scheme. It is proposed that by using the layer-resolving grids obtained by the transformations satisfying (2), ε-uniform high-order convergence will be demonstrated for schemes of high-order in the physical interval x. Moreover, the numerical solution can be interpolated ε-uniformly with high-order accuracy on the entire interval $[0, 1]$. One such transformation, complying with (2) for an arbitrarily specified n, is presented in the paper for exponential and power-of-first-type layers by updating some mapping demonstrated in [4] for the upwind scheme.

The layer-damping coordinate mapping described in the paper for both exponential and power-of-first-type layers produces other forms of layer-resolving grids—grids above and beyond those already well known and broadly accepted,

namely, those developed by Bakhvalov [1] and Shishkin [5]. The grids developed by Bakhvalov and Shishkin have been efficiently applied to diverse singularly perturbed problems, but to problems having only exponential-type layers, typically represented by functions $\exp\left(-bx/\varepsilon^k\right)$. These grids are not suitable for tackling important problems, the solutions of which have wider layers: power, logarithmic, and mixed-type boundary and interior layers (see [6]), and also require knowledge of the constant b affecting the width of the exponential layer. Such knowledge is not always available, for example, for the boundary layers in viscous-gas flows or the interior layers in solutions to the quasi–linear problems discussed in the current paper. In addition, the very popularity of the grids contrived by Bakhvalov and Shishkin seems to hinder researchers from considering other problems with non-exponential layers, which are not handled by these grids. As a result, this very important area of problems with non-exponential layers remains at present nearly an unexplored area to the SPP community (here and below, SPP stands for "singular–perturbation problems"). This paper demonstrates an attempt to break out of this narrow circle. It is hoped that the updated layer-resolving grid described in this paper should strengthen and encourage researchers of the SPP community to solve broader and more important classes of problems having not only exponential, but other boundary and interior layers.

Section 2 formulates a coordinate transformation and theoretically substantiates that it eliminates both exponential and power-of-first-type singularities. Section 3 presents numerical experiments with the grid generated on linear problems approximated by high-order schemes. It also presents numerical experiments on non-linear problems approximated by a first-order scheme. The section further discusses application of the grid to two-dimensional Navier–Stokes equations modelling a viscous-gas flow over a flat plate.

2 Transformation for Eliminating High-Order Singularities

This section describes a basic coordinate transformation eliminating exponential and power-of-first-type singularities of arbitrary order in the vicinity of a boundary layer near the point $x_0 = 0$. It can be applied to specify layer-damping transformations and corresponding layer-resolving grids on the entire interval [0, 1] with randomly located exponential or power-of-first-type layers, by creating local mappings via procedures of scaling, shifting, inverting the basic transformation, and then matching them with themselves and polynomials. When using high-order approximations of equations, such transformations may be suitable for generating layer-resolving grids which provide both high-order ε-uniform convergence and high-order ε-uniform interpolations.

Let us first consider an exponential boundary layer function $u(x, \varepsilon)$ whose jth derivative is estimated by an exponential function and M, i.e.,

$$\left|u^{(j)}(x, \varepsilon)\right| \le M\left[\varepsilon^{-kj} \exp\left(-bx/\varepsilon^k\right) + 1\right], \ b > 0, \ 1 \le j \le n, \ 0 \le x \le 1. \tag{3}$$

This singularity can be eliminated by a coordinate transformation $x(\xi, \varepsilon, a, k) \in C^l[0, 1], n \geq l \geq 0$ in the following form:

$$x(\xi,\varepsilon,a,k)=\begin{cases} c\varepsilon^k\left((1-d\xi)^{-1/a}-1\right), & 0\leq\xi\leq\xi_0,\\ c\Bigg[\varepsilon^{k(1-\beta/a)}-\varepsilon^k \\ \quad+\left(\dfrac{\varepsilon^k}{(1-d\xi)^{1/a}}\right)'(\xi_0)(\xi-\xi_0) \\ \quad+\dfrac{1}{2}\left(\dfrac{\varepsilon^k}{(1-d\xi)^{1/a}}\right)''(\xi_0)(\xi-\xi_0)^2+\ldots & \xi_0\leq\xi\leq 1,\\ \quad+\dfrac{1}{l!}\left(\dfrac{\varepsilon^k}{(1-d\xi)^{1/a}}\right)^{(l)}(\xi_0)(\xi-\xi_0)^l \\ \quad+c_0(\xi-\xi_0)^{l+1}\Bigg], \end{cases} \tag{4}$$

where $d=\left(1-\varepsilon^{k\beta}\right)/\xi_0$; $0<m_2\leq\beta\leq a/(1+na)$; a is an arbitrary positive constant; $1>\xi_0\geq m_3>0$, (for example $\xi_0=1/2$); and $c>0$ is such that the necessary boundary condition $x(1,\varepsilon,a,k)=1$ is satisfied. This transformation is a generalization of the mapping proposed in [7] and [8]. A specific transformation of this kind for $a=1$ was introduced by Vulanovic in [9].

Theorem 1 *The transformation* (4) *eliminates singularity* (3) *up to order n, i.e., estimate* (2) *is valid.*

This theorem was proved in [10] (Sect. 6.3.2).

In the same manner it was proved in [10] that, in order to eliminate the exponential singularity (3) of the function $u(x, \varepsilon)$ up to order n in a new coordinate ξ, we can rely on the basic local logarithmic contraction function $x_1(\xi, \varepsilon, a, k)$ on the corresponding interval $[0,\ \xi_0]$:

$$x_1(\xi,\varepsilon,a,k)=-\left(\varepsilon^k\ln(1-d\xi)\right)/a\ ,\quad 0\leq\xi\leq\xi_0\ ,\quad d=\left(1-\varepsilon^{k/n}\right)/\xi_0\ , \tag{5}$$

but with the restriction $b/n^2\geq a>0$, by prolongating it smoothly on the interval $[0, 1]$. The transformation of this type was introduced by Bakhvalov [1].

The transformation (4) is more convenient for eliminating exponential singularities than the transformation based on (5), since the constant a in (4) is not dependent on b in (3), so that, with an arbitrary fixed constant $a>0$, this transformation alone

is valid for all constants $b \in (0, \infty)$ in (3) for eliminating singularities of $u(x, \varepsilon)$ up to order n. Another popular piecewise uniform transformation

$$x(\xi, \varepsilon, b) = \begin{cases} 2\sigma\xi \ , & 0 \le \xi \le 1/2 \ , \\ \sigma + 2(1-\sigma)\xi \ , & 1/2 \le \xi \le 1 \ , \end{cases} \tag{6}$$

where $\sigma = \min\{0.5, (n/b)\varepsilon \ln N\}$, proposed by Shishkin [5] for generating grids for solving problems with exponential layers, also being dependent on fixed constant b in (3), will not be suitable for all $b \in (0, \infty)$ in (3).

The power transformation (4) with a proper choice of constant $a > 0$ is also suitable for eliminating power singularities of type 1 near $x_0 = 0$, i.e., when solution derivatives are estimated by the following function:

$$\left|u^{(j)}(x, \varepsilon)\right| \le M\left[\varepsilon^{kb}/\left(\varepsilon^k + x\right)^{b+j} + 1\right], \quad 1 \le j \le n \ , \quad 0 \le x \le 1 \ . \tag{7}$$

Here, the boundary layer interval, where all the derivatives up to n of $u(x, \varepsilon)$ are not uniformly bounded over ε, is $[0, \ x_2^n]$, $x_2^n = m_2\varepsilon^{kb/(b+n)} \gg x_1^n = (kn/b)\varepsilon^k \ln\left(\varepsilon^{-1}\right)$ for sufficiently small ε, so that the transformations (5) and (6) are not suitable for generating layer-resolving grids when such singularities have incomparably wider layers than any exponential layer.

Theorem 2 *The transformation* (4)*, with the restrictions* $0 < a \le b/n^2$ *and* $\beta = a/(1 + na)$*, eliminates singularity* (7) *up to order n.*

The proof of this theorem can easily be restored from [10] (Sect. 6.3.2), namely, by adding in that proof the following restriction: $\beta = a/(1 + an)$, without which the theorem is not valid.

3 Numerical Experiments

This section presents results of numerical experiments with problems having exponential or power-of-type-1 boundary and interior layers.

3.1 Numerical Experiments with High-Order Schemes

This subsection compares results of calculations for problem (1) with $d(x) = x$ through the compact schemes of high-order approximations $O(h^p)$, $(p = 1; 2; 3)$, formulations of which were described in [11] for $d(x) = 0$. The compact schemes for the case $d(x) = x$ are easily obtained from those formulations by substituting $\varepsilon + x$ for ε. Problems of this type with $\alpha = 1$ were discussed in the monograph [12]

for modeling a flow of liquid in the vicinity of a circular orifice of a small radius $r = \varepsilon$; those with $\alpha = 2$, for simulating the motions of charges viewed as classical particles in [13]. Solutions to this problem may have layers of exponential, power, and logarithmic types [4]. Numerical solutions of problem (1) with $d(x) = 0$, using high-order approximations and layer-resolving grids, were discussed in [11].

The exact solution was unknown for the test problems being solved. In the figures given below, the exact solution means the numerical solution obtained on a very fine grid ($N = 1600$); in all the tables, the estimates of the error δ and order p have subscripts 1, 2, and 3, corresponding to the orders of accuracy of the schemes used to obtain the results. We calculate δ_k and p_k in this subsection and the next as in [11], i.e., $\delta_k = \max_{i=0}^{N} \left|u_{2i}^{2N} - u_i^N\right|$, $p_k = \log_2\left(\delta_k^{2N}/\delta_k^N\right)$, $k = 1, 2, 3$.

For generating layer-resolving grids for both exponential and power-of-type-1 layers near the boundary point $x_0 = 0$, we use a comprehensive transformation of the class $C^l[0,\ 1]$ in the form (4), assuming $\xi_0 = 1/2$, $l = 2$, $n = p + 1$, where p is the order of the scheme.

Exponential Boundary Layer Letting in (1) $d(x) = x$, $\alpha \geq 2$, $a(x, u) = a(x)$, $b = -a(0) > 0$, we have from [4]

$$\left|u^{(j)}(x, \varepsilon)\right| \leq M\left[1 + \varepsilon^{-j\alpha} \exp\left(-mbx/\varepsilon^{\alpha}\right)\right], \quad j \leq n + 1\,, \quad 0 < m < 1\,, \quad 0 \leq x \leq 1\,, \tag{8}$$

where m is an arbitrary constant from interval $(0, 1)$ independent of ε. With the solution having a single exponential boundary layer of scale α near $x = 0$, an explicit transformation for generating a layer-resolving grid can be defined by the mapping $x(\xi, \varepsilon, a, k)$ from (4) for $k = \alpha$ and an arbitrary $a \geq m > 0$.

Example 1 As an example, we shall assume in (1) $d(x) = x$, $\alpha = 2$, $a(x, u) = -1$, $f(x, u) = u - 2\sin(4\pi x)$, $u(0) = 0$, and $u(1) = 1$. Figure 1 and Table 1 demonstrate the numerical solution and characteristics for the layer-resolving grid $x_i = x(ih, \varepsilon, 2, 2)$, $i = 1, \ldots, N$, $h = 1/N$.

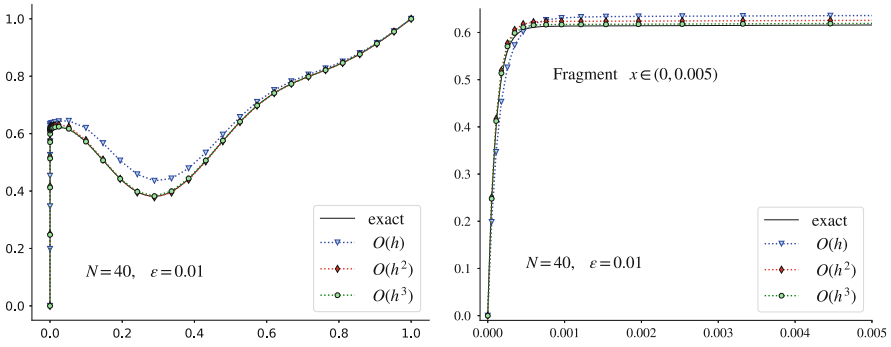


Fig. 1 Exponential boundary layer. Numerical solutions (left); fragment of the solution (right)

Table 1 Data for exponential layer for $\varepsilon = 0.01$

N	δ_1	p_1	δ_2	p_2	δ_3	p_3
40	6.47×10^{-2}	3.95	6.03×10^{-2}	4.05	2.30×10^{-1}	2.12
80	3.67×10^{-2}	0.82	9.84×10^{-3}	2.61	1.02×10^{-2}	4.50
160	1.98×10^{-2}	0.89	1.99×10^{-3}	2.31	6.18×10^{-4}	4.04
320	1.03×10^{-2}	0.94	4.75×10^{-4}	2.07	3.83×10^{-5}	4.01
640	5.25×10^{-3}	0.97	1.17×10^{-4}	2.02	2.39×10^{-6}	4.00
1280	2.65×10^{-3}	0.99	2.91×10^{-5}	2.01	1.49×10^{-7}	4.00

Table 1 presents a comparison of calculations using three schemes on a sequence of grids for $\varepsilon = 0.01$: in the first column, the number of grid points; in the next columns, a posteriori estimates in C-norms of the error δ_k and the order of accuracy p_k obtained in two successive calculations by using the scheme of the k-order approximation. Although $\varepsilon = 0.01$ in accordance with (8), the first derivative of the solution of (1) is estimated in the center of the boundary layer by $10^4 \times M$.

In the calculations which follow, the order of accuracy p_k for $k = 1$, $k = 2$, and $k = 3$ being similar to the order of accuracy in Example 1, this characteristic is omitted in the tables presented.

Power Boundary Layer of Type 1 Letting in (1) $d(x) = x$, $\alpha = 1$, $b = -a(0) > 1$, we have from [4] for $b > 1$

$$\left|u^{(i)}(x,\varepsilon)\right| \le M\left[1 + \varepsilon^{b-1}(\varepsilon + x)^{1-b-i}\right], \quad i \le n+1\,, \quad 0 \le x \le 1\,, \tag{9}$$

i.e., the solution has a single power boundary layer of type 1 and scale 1 near $x = 0$. Thus, an explicit transformation for generating a layer-resolving grid can be defined by the mapping $x(\xi, \varepsilon, a, k)$ from (4) for $k = 1$ and $a = (b-1)/n^2$.

Example 2 As an example, we shall assume $\alpha = 1$, $a(x) = -4$, $F(x,u) = u - 8\sin(4\pi x)$, $u(0) = 0$, and $u(1) = 0.5$, in which case $a = 3/n^2$. The results are presented in Table 2 and Fig. 2.

3.2 Experiments with Autonomous Equations

This subsection presents results of numerical experiments with an idealized problem simulating some shock-wave structures of a steady heat-conducting gas flow [4] (pages 9–10). To calculate the numerical solutions, we use approximations of the autonomous singularly perturbed boundary value problems by the standard iterative upwind finite difference scheme [10, pp. 257] and the scheme of the second order of accuracy [11] on the appropriate modifications [10, pp. 247, pp. 263] of the adaptive layer-resolving grid (4). The characteristics δ_k and p_k, $k = 1, 2$, are calculated in

Table 2 Data for power layer of type 1 for various ε

	δ_1	δ_2	δ_3	δ_1	δ_2	δ_3	δ_1	δ_2	δ_3
N/ε	10^{-2}	10^{-2}	10^{-2}	10^{-4}	10^{-4}	10^{-4}	10^{-6}	10^{-6}	10^{-6}
40	3.5×10^{-2}	4.5×10^{-2}	5.7×10^{-2}	4.8×10^{-2}	7.3×10^{-2}	2.4×10^{-1}	6.1×10^{-2}	8.3×10^{-2}	1.3
80	1.8×10^{-2}	7.9×10^{-3}	2.4×10^{-3}	2.6×10^{-2}	1.2×10^{-2}	6.4×10^{-3}	3.8×10^{-2}	1.4×10^{-2}	9.1×10^{-3}
160	9.4×10^{-3}	1.9×10^{-3}	1.5×10^{-4}	1.4×10^{-2}	2.9×10^{-3}	3.6×10^{-4}	2.1×10^{-2}	3.2×10^{-3}	4.7×10^{-4}
320	4.8×10^{-3}	4.9×10^{-4}	9.7×10^{-6}	7.5×10^{-3}	7.1×10^{-4}	2.4×10^{-5}	1.2×10^{-2}	7.9×10^{-4}	3.0×10^{-5}
640	2.4×10^{-3}	1.2×10^{-4}	6.1×10^{-7}	3.8×10^{-3}	1.8×10^{-4}	1.5×10^{-6}	6.1×10^{-3}	2.0×10^{-4}	1.9×10^{-6}
1280	1.2×10^{-3}	3.0×10^{-5}	3.8×10^{-8}	1.9×10^{-3}	4.5×10^{-5}	9.6×10^{-8}	3.1×10^{-3}	4.9×10^{-5}	1.2×10^{-7}

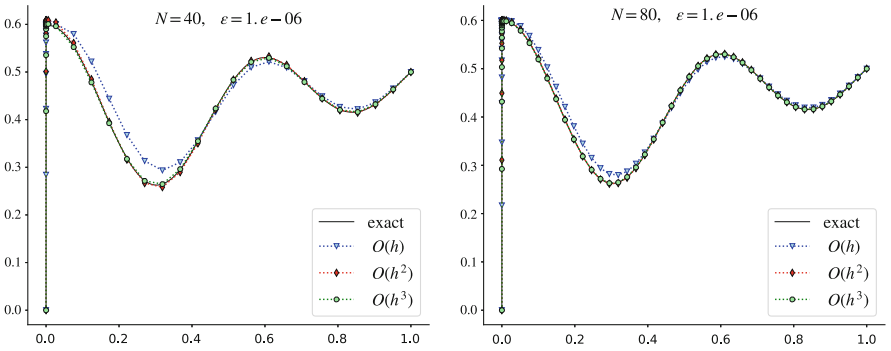


Fig. 2 Power boundary layer of type 1. Numerical solutions for $N = 40$ (left) and $N = 80$ (right)

the same way as in the previous subsection. The subscript k corresponds to the orders of accuracy of the schemes used to obtain the results.

Interior Power Layer Letting in (1) $d(x) = 0$, $f(x, u) = 0$, $\alpha = 1$, $a(x, u) = a(u) \in C[A_0, A_1]$, $b(u) = \int_{A_0}^{u} a(\mu)\,\mathrm{d}\mu$, $b(A_0) = b(A_1)$, $b(u) > b(A_0)$, for $A_0 < u < A_1$ and, besides this, $b'(A_0) = b'(A_1) = 0$, $b''(A_0) \neq 0$, $b''(A_1) \neq 0$, in accordance with [4] for $j \leqslant n - 1$, the estimate holds

$$\left|u^{(j)}(x, \varepsilon)\right| \leqslant M\left[1 + \varepsilon/(\varepsilon + x - x_0)^{1+j}\right], \quad 0 \leqslant x \leqslant 1\,,$$

where $u(x, \varepsilon)$ is the solution to (1). The center x_0 of the layer is calculated as

$$x_0 = \frac{1}{\sqrt{c_0 + c_1}}\left[\sqrt{c_1} - \varepsilon \ln \varepsilon^{-1}\left(\frac{d_0}{c_0^{3/2}} + \frac{d_1}{c_1^{3/2}}\right)\right],$$

with $c_i = a'(A_i)/2$, $d_i = a''(A_i)/6$, $i = 0, 1$. Thus, the solution has a single interior power layer of first type.

Example 3 As an example, we assume $a(u) = 2(u + 1)(u - 2)(2u - 1)$, $A_0 = -1$, $A_1 = 2$, We have $b(u) = (u - 2)^2(u + 1)^2$. It holds $b(A_0) = b(A_1)$, $b(u) = (u - 2)^2(u + 1)^2 > b(A_0) = 0$ for $A_0 < u < A_1$, $b'(u) = a(u)$, $b'(A_0) = b'(A_1) = 0$, and $b''(u) = 2(6u^2 - 6u - 3)$, $b''(A_0) = 18 \neq 0$, $b''(a_1) = 18 \neq 0$, and $c_0 = a'(A_0)/2 = 9$, $c_1 = a'(A_1)/2 = 9$, $d_0 = a''(A_0)/6 = -6$, $d_1 = a''(A_1)/6 = 6$, and $x_0 = \sqrt{2}/2$. The constant a appearing in (4) is subject to $0 < m < a \leqslant 1/n^2$. The values of characteristics δ_k and p_k, $k = 1, 2$, are displayed in Table 3; the numerical solutions and their parts are given in Fig. 3.

Interior Mixed Layer Letting in (1): $d(x) = 0$, $f(x, u) = 0$, $\alpha = 1$, $a(u)$ be such that $a(A_0) \neq 0$, $a^{(p)}(A_1) = 0$, $0 \leqslant p < r$, $a^{(r)}(A_1) \neq 0$, and the function $b(u)$

Table 3 Data for power layer of type 1 for various ε

	δ_1	p_1	δ_2	p_2	δ_1	p_1	δ_2	p_2
N	$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-4}$	
160	1.95×10^{-2}	0.96	1.03×10^{-3}	1.97	2.16×10^{-2}	0.96	1.09×10^{-3}	1.98
320	1.00×10^{-2}	0.98	2.65×10^{-4}	1.98	1.10×10^{-2}	0.98	2.75×10^{-4}	1.98
640	5.06×10^{-3}	0.98	6.72×10^{-5}	1.98	5.60×10^{-3}	0.99	6.97×10^{-5}	1.96
1280	2.65×10^{-3}	–	1.70×10^{-5}	–	2.81×10^{-3}	–	1.79×10^{-5}	–
N	$\varepsilon = 10^{-5}$		$\varepsilon = 10^{-5}$		$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-6}$	
160	2.28×10^{-2}	0.96	1.20×10^{-3}	2.00	2.36×10^{-2}	0.96	1.28×10^{-2}	5.33
320	1.11×10^{-2}	0.98	3.01×10^{-4}	1.97	1.21×10^{-2}	0.98	3.19×10^{-4}	2.00
640	5.94×10^{-3}	0.99	7.70×10^{-5}	1.98	6.14×10^{-3}	0.99	7.99×10^{-5}	2.00
1280	2.98×10^{-3}	–	1.94×10^{-5}	–	3.09×10^{-3}	–	2.00×10^{-5}	–

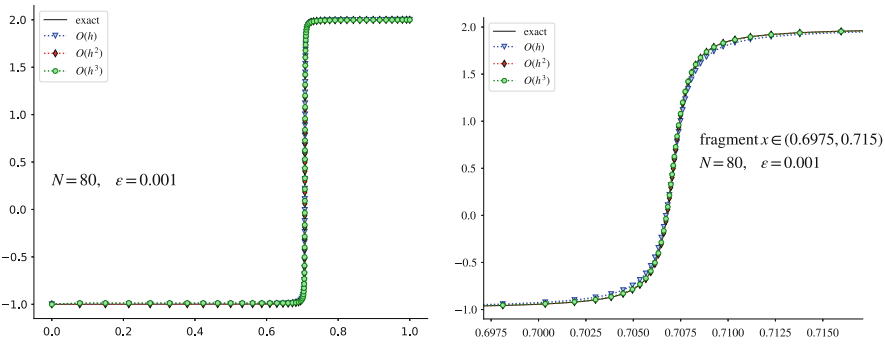


Fig. 3 Interior power layer of kind 1. Numerical solutions (left), fragments (right)

satisfies the conditions $b(A_0) = b(A_1)$ and $b(u) > b(A_0)$, if $A_0 < u < A_1$, then, in accordance with [4], the following estimates hold

$$\left|u^{(j)}(x,\varepsilon)\right| \leqslant M \begin{cases} 1+\varepsilon^{-j}\exp[a_0(x-x_0)/\varepsilon]\,, & j \leqslant n\,, \quad 0 \leqslant x \leqslant x_0\,, \\ 1+\varepsilon^{1/r}(\varepsilon+x-x_0)^{-j-1/r}\,, & j \leqslant n\,, \quad x_0 \leqslant x \leqslant 1\,, \end{cases} \tag{10}$$

where $a_0 = a(A_0)$, $x_0 = \varepsilon(r+1)/(a_0 r)\ln\left(\varepsilon^{-1}\right)$. Hence, the solution to the problem (1) has a mixed interior layer of scale 1 and the central point x_0 of the layer tends to 0, when $\varepsilon \to 0$,

Example 4 As an example, we consider $a(u) = 3u^2 - 6u + 2$, $A_0 = 0$, $A_1 = 1$. We have $b(0) = b(1) = 0$, $b(u) > b(0)$, $0 < u < 1$, and $a(0) = 2 \neq 0$, $a'(u) = -6 + 6u$, $a'(1) = 0$, $a''(1) = 6 \neq 0$, and $r = 2$, $x_0 = \varepsilon(r+1)\ln\varepsilon^{-1}/(a(0)r) = 3\varepsilon\ln\left(\varepsilon^{-1}\right)/4$. The constant a appearing in (4) has an arbitrary value $a \geqslant m > 0$; the selected value is $a = 1/4$. The values of δ_k and p_k, $k = 1, 2$, are displayed in Table 4; the graphics are given in Fig. 4.

Table 4 Data for mixed interior layer for various ε

	δ_1	p_1	δ_2	p_2	δ_1	p_1	δ_2	p_2
N	$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-4}$	
160	1.20×10^{-2}	0.87	5.28×10^{-3}	1.97	2.09×10^{-2}	0.91	4.24×10^{-3}	2.02
320	6.58×10^{-3}	0.97	1.34×10^{-3}	1.98	1.11×10^{-2}	0.96	1.04×10^{-3}	1.99
640	3.36×10^{-3}	0.98	3.41×10^{-4}	1.99	5.71×10^{-3}	0.97	2.63×10^{-4}	2.00
1280	1.70×10^{-3}	–	8.60×10^{-5}	–	2.90×10^{-3}	–	6.58×10^{-5}	–
N	$\varepsilon = 10^{-5}$		$\varepsilon = 10^{-5}$		$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-6}$	
160	3.05×10^{-2}	0.89	9.76×10^{-3}	2.03	4.13×10^{-2}	0.81	8.98×10^{-2}	4.11
320	1.65×10^{-2}	0.95	2.38×10^{-3}	1.98	2.36×10^{-2}	0.93	5.17×10^{-3}	1.99
640	8.52×10^{-3}	0.98	6.04×10^{-4}	1.99	1.24×10^{-2}	0.97	1.29×10^{-3}	2.00
1280	4.32×10^{-3}	–	1.52×10^{-4}	–	6.34×10^{-3}	–	3.25×10^{-5}	–

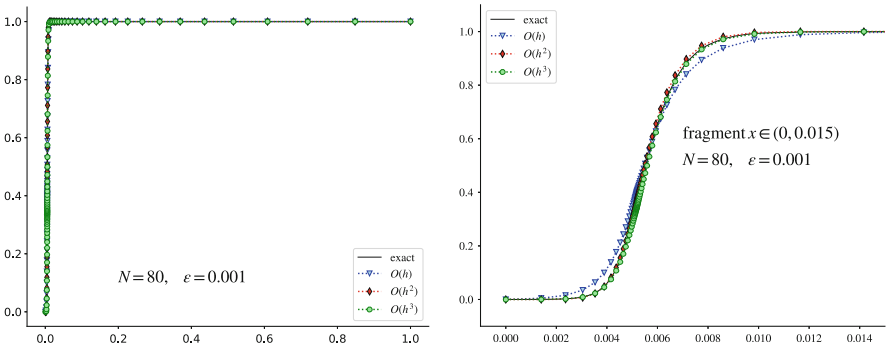


Fig. 4 Interior mixed layer. Numerical solutions (left), fragments (right)

3.3 Numerical Experiments with a Two-Dimensional Problem of a Viscous-Gas Flow over a Flat Plate

This subsection presents results of numerical experiments with a viscous heat-conducting gas flow over a flat plate at a zero angle of attack, modelled by two-dimensional Navier–Stokes equations:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \frac{M_\infty}{Re_\infty}\left(\frac{\partial F^v}{\partial x} + \frac{\partial G^v}{\partial y}\right). \tag{11}$$

Here, Q is a vector of conservative variables, F, G vectors of inviscid flows, F^v, G^v vectors of viscous flows, and Re, M Reynolds and Mach numbers calculated from the parameters of the incoming flow. The system of equations (11) is closed by the ideal gas law: $p = \rho T/\gamma$. On the surface of the plate are imposed next boundary and initial conditions: adhesion condition ($u|_\Gamma = 0$) and adiabatic temperature condition $\left(\frac{\partial T}{\partial n}\big|_\Gamma = 0\right)$. The domain for calculations is a rectangle, the bottom side of which coincides with the axis $y = 0$.

The solution to the problem has a boundary layer, the type of which is not known. However, some information can be obtained from problem (1) for $d(x) = 0$, $a(x, u) = a(x)$, $a(0) = 0$ analyzed analytically in [4], where it was shown that its solutions have exponential boundary layers when $a'(0) = 0$, or power-of-type-1 boundary layers when $a'(0) > 0$. The equation in (11) with respect to a component u of the velocity vector (u, v) can be considered as (1) for $d(x) = 0, a(x, u) = a(x)$, $a(0) = 0$, if $\varepsilon = M/Re$, x in (11) is replaced by y, x in (11) is a parameter, and $\rho v = a()$. Thus, the solution to (11) may have an exponential or power-of-type-1 boundary layer.

Numerical solutions of (11) are based on the code in Fortran described in [14]. The equations are integrated in time by using the second-order Runge–Kutta scheme. The calculations are carried out until a steady solution is achieved. The step of integration over time is chosen from the Courant–Friedrichs–Lewy condition [14]. The grid nodes in the x direction are uniform, while in the y direction they are defined through the transformation (4) with $l = 2$, $k = 1/2$, $\xi_0 = 0.8$, $a = 2$. For estimating the accuracy of numerical solutions, the characteristic $\delta_t = \max_{i=1}^{N_t} |u_{3i}^{N_{t+1}} - u_i^{N_t}|$, $t = 1, 2$, was used. To estimate the order of accuracy of the numerical solution, the characteristic $p_1 = \log_3(\delta_t/\delta_{t+1})$ for two consecutive N_t was applied. For estimating the jump of the numerical solution at nearby nodes and the jump order, the corresponding characteristics $du_t = \max_{i=1}^{N_t} |u_{i+1}^{N_t} - u_i^{N_t}|$, $t = 1, 2, 3$, and $p_2^t = \log_3(du_t/du_{t+1})$, $t = 1, 2$ were applied.

Hypersonic gas flow was investigated for Mach number $M = 6$. Calculations of problem (11) were conducted for various values of $Re = 100{,}000, 1{,}000{,}000$. For each of these values, sequences of grids with tripled numbers of grid nodes in the y coordinate are used: $N_y^t = 3^t N_0$, $t = 0, 1, 2$, $N_0 = 51$. The number of grid nodes along the x coordinate is 192.

The values of characteristics δ_t, p_1, du_t, p_2^t are shown in Table 5, the velocity profiles of the numerical solution in Fig. 5.

Table 5 Data for power layer of type 1 and scale 1/2, one layer near $x = 0$, for various Re

	δ_t	p_1	du_t	p_2^t	δ_t	p_1	du_t	p_2^t
N_t	$M = 7, Re = 100{,}000$				$M = 6, Re = 1{,}000{,}000$			
51	–	–	0.213	–	–	–	0.283	–
153	0.017	–	0.074	0.962	0.064	–	0.099	0.956
459	0.003	1.58	0.025	0.988	0.005	2.32	0.034	0.973

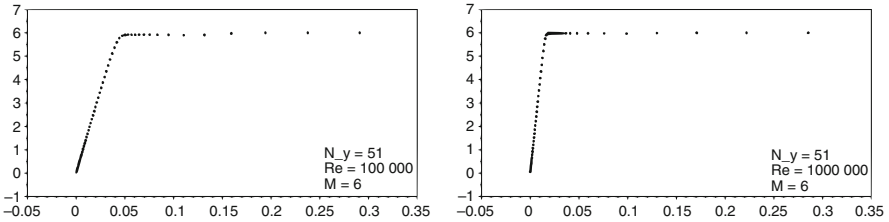


Fig. 5 Longitudinal velocity profiles and values at the grid points in the cross section $x = 0.5$

4 Conclusions

The paper substantiates theoretically that the coordinate transformation formulated eliminates both exponential and power-of-first-type singularities. The numerical calculations performed in this paper show that, from the computational view, the presented grid explicitly produced by the coordinate transformation, is efficient for solving linear or quasilinear problems with such types of layers using first order and high-order algorithms. The values of the characteristics introduced are within expected bounds and stabilize very quickly when the number of grid nodes increases.

Acknowledgment The reported study was funded by RFBR, project No. 20-01-00231.

References

1. Bakhvalov, N.S.: On optimization of the methods of the numerical solution of boundary value problems with boundary layers. USSR Comput. Math. and Math. Phys. **9**(4), 842–859 (1969)
2. Berger, A.E., Han, H., Kellog, R.B.: A priori estimates of a numerical method for a turning point problem. Math. Comput. **42** 166, 465–492 (1984)
3. Farrell, P.A.: A uniformly convergent difference scheme for turning point problems. In: Miller, J.J.H. (ed.), Boundary and Interior Layers, Computational and Asymptotic Methods, pp. 270–274. Boole Press, Dublin (1980)
4. Liseikin, V.D.: Layer Resolving Grids and Transformations for Singular Perturbation Problems. VSP, Utrecht (2001)
5. Shishkin, G.I.: A difference scheme for a singularly perturbed equation of parabolic type with a discontinuous initial condition. Soviet. Math. Dokl. **37**, 792–796 (1988) (in Russian)
6. Liseikin, V. D., Karasuljic, S.: Numerical analysis of grid-clustering rules for problems with power of the first type boundary layers. Comput. Technol. **25**(1), 49–65 (2020)
7. Liseikin, V.D., Yanenko N.N.: On a uniform algorithm for solving of equations of second order with a small parameter affecting the higher derivatives. Chisl. Metody. Mech. Sploshn. Sredy **12**(2), 45–56 (1981) (in Russian)
8. Liseikin, V.D.: On the numerical solution of an equation with a power boundary layer on a nonuniform grid. Chisl. Metody Mech. Sploshn. Sredy **15**(2), 90–97 (1984) (in Russian)

9. Vulanovic, R.: Mesh construction for numerical solution of a type of singular perturbation problems. Numer. Meth. Approx. Theory, Conference Paper, Lis, September 26–28, 137–142 (1984)
10. Liseikin, V.D.: Grid Generation for Problems with Boundary and Interior Layers. NSU, Novosibirsk (2018)
11. Liseikin, V.D., Paasonen, V.I.: Compact difference schemes and layer-resolving grids for numerical modeling of problems with boundary and interior Layers. Numer. Anal. Appl. **12**(1), 1–17 (2019)
12. Polubarinova–Kochina, P. Ya: Theory of Ground Water Movement. Princeton University Press, Princeton, NJ (1962) (translated from Russian)
13. Zamaraev, K.I., Khairutdinov, R.F., Zhdanov, V.P.: Electron Tunneling in Chemistry. Chemical Reactions at Large Distances. Elsevier, Amsterdam (1989) (translated from Russian)
14. Kudryavtsev, A. N., Mironov, S. G., Poplavskaya, T.V., Tsyryul'nikov I.S.: Experimental investigation and direct numerical simulation of the development of disturbances in a viscous shock layer on a flat plate. J. Appl. Mech. Tech. Phys. **47**, 617–627 (2006) (translated from Russian)

Preserved Structure Constants for Red Refinements of Product Elements

Sergey Korotov and Jon Eivind Vatne

Abstract In this paper we discuss some strategy for red refinements of product elements and show that there are certain structure characteristics (d-sines of angles formed by certain edges in the initial partition) which remain constant during refinement processes. Such a property immediately implies the validity of the so-called maximum angle condition, which is a strongly desired property in interpolation theory and finite element analysis. Our construction also gives a clear refinement scheme preserving shape regularity.

1 Introduction and Basic Definitions

Red refinement is one of the most popular meshing techniques used in various branches of numerical mathematics. However, it is usually applied to simplicial partitions only, see, e.g., [1, 14, 15, 17, 18], and various aspects of regularity of the generated meshes are then analysed.

In practice, depending on the shape of the domain over which we construct the meshes, one may prefer to construct some initial mesh consisting of so-called product elements, and only after that refine it into simplices. The simplest illustration in this direction would be the case of cylindric-type domains first naturally split into right prisms and then (conformally) into tetrahedra, see, e.g., [13].

In this work, we consider a more general case of product elements of any dimensions and red refinement techniques used independently for each factor of the product. Some regularity properties of resulting simplicial meshes are discussed.

S. Korotov
Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway
e-mail: sergey.korotov@hvl.no

J. E. Vatne (✉)
Department of Economics, BI Norwegian Business School, Bergen, Norway
e-mail: jon.e.vatne@bi.no

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_15

Products of simplices, and their triangulations, have been studied in many contexts, in particular for two factors. See, e.g., [2, 4, 6, 7, 9].

In 1978, F. Eriksson proposed the following concept of the d-dimensional sine of angles in $\mathbb{R}^d$. In terms of the simplex S, for any of its vertices A_i, $i = 0, \ldots, d$, the d-dimensional sine of the angle of S at A_i, denoted by $\hat{A}_i$, is defined as follows (see (3) in [5, p. 72]):

$$\sin_d(\hat{A}_i | A_0 A_1 \ldots A_d) = \frac{d^{d-1} (\text{vol } S)^{d-1}}{(d-1)! \prod_{j=0, j\neq i}^{d} \text{vol } F_j}, \tag{1}$$

where vol denotes the measure of (simplex or its facets) of relevant dimension. We will apply $\sin_d$ to a set of d vectors, which then will mean that we choose A_i at the common point of origin of the vectors and the remaining vertices $A_0, \ldots, A_{i-1}, A_{i+1}, \ldots, A_d$ as the corresponding endpoints. Write vol(a set of vectors) for the hypervolume of the generalized parallelotope spanned by the vectors. We then have the convenient formula

$$\sin_d(v_i, \ldots, v_d) = \frac{\text{vol}(v_1, \ldots, v_d)^{d-1}}{\prod_{i=1}^{d} \text{vol}(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_d)}.$$

See also [12].

Definition 1 ([10]) A family $\mathcal{F} = \{\mathcal{T}_h\}_{h\to 0}$ of partitions of some polytope into simplices is said to satisfy the *maximum angle condition* if there exists $C_0 > 0$ such that for any $\mathcal{T}_h \in \mathcal{F}$ and any $S = \text{conv}\{A_0, \ldots, A_d\} \in \mathcal{T}_h$ we can always find d edges of S, which when considered as vectors, constitute a (higher-dimensional) angle whose d-sine is bounded from below by the constant C_0. Here, we let h denote the maximal diameter of the simplices in a partition.

Simplicial partitions satisfying the maximum angle condition are highly desired in numerical analysis for various interpolation and finite element convergence proofs, see, e.g., [10, 17]. There is another (equivalent) definition of the maximum angle condition in [12]. However, we prefer the one from Definition 1 in this paper, since it is more suitable for our geometric considerations in what follows.

The maximum angle condition is weaker than the shape regularity property (also known as the minimum angle condition in two dimensions) described, e.g., in [3]. Even though our constructions are suited to the maximum angle property, the shape regularity property can also be preserved if desired, as explained below in Theorem 3(b) and Remark 2. The refinement scheme presented in Theorem 3(c) will in general violate the shape regularity property.

2 Red Refinement Strategy and Its Properties

Recall a standard triangulation of the hypercube (see, e.g., Freudenthal [8]).

Theorem 1 *Consider the unit hypercube in* $\mathbb{R}^d$ *with* 2^d *vertices* $(0, 0, \ldots, 0)$ *to* $(1, 1, \ldots, 1)$ *(all possible combinations of* 0 *and* 1*). For any path from* $(0, 0, \ldots, 0)$ *to* $(1, 1, \ldots, 1)$ *consisting of the standard unit vectors* e_i, $i = 1, \ldots, d$, *in some order, there is a corresponding* d*-simplex (known as a path simplex) given as the convex hull of the vertices of the hypercube along the path.*

a) By varying the path over all possible orderings of the vectors e_i*, this gives a triangulation of the hypercube consisting of* $d!$ *simplices, each sharing the common edge from* $(0, 0, \ldots, 0)$ *to* $(1, 1, \ldots, 1)$.
b) We can subdivide the unit cube into 2^d *smaller cubes, with coordinates given by* 0, $1/2$ *or* 1, *and apply the same construction to each of them.*

The resulting triangulation from Theorem 1 (b) refines the original triangulation, thus defining a *red refinement scheme* for the hypercube by iteration. By making consistent choices of diagonals, we automatically get conformity when we start refining a single simplex as above.

By embedding a simplex into the hypercube we can produce a conforming red refinement scheme by intersecting with the subdivision of the hypercube as presented in Theorem 1, see Fig. 1.

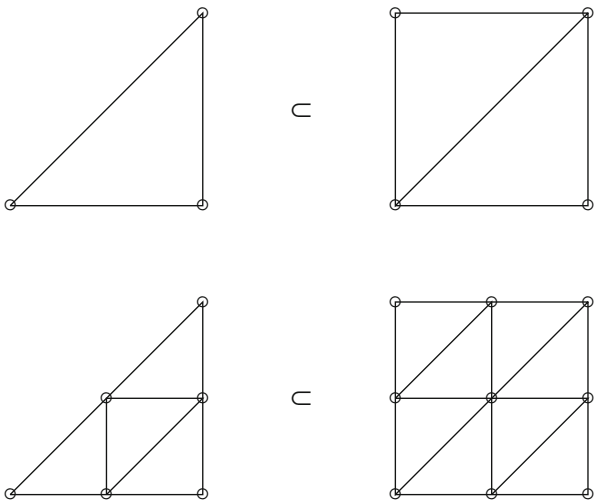


Fig. 1 By embedding a simplex in a hypercube, we construct a red refinement of the simplex by restriction of the red refinement of the hypercube

More formally, given any d-dimensional simplex S, we can choose a path (or an ordering of the vertices) $A_0A_1 \dots A_d$. There is then a unique affine transformation T that maps A_0 to $(0, 0, \dots, 0)$, and A_i to $(1, 1, \dots, 1, 0, 0, \dots, 0)$ (1 in the first i positions), $i = 1, \dots, d$. Then $T(S)$ is one of the simplices in the triangulation of the hypercube described in the theorem, corresponding to the path $e_1, e_2, \dots, e_d$, which we will use as a reference simplex. To describe the set by inequalities, it is the set $1 \geq x_1 \geq x_2 \geq \cdots \geq x_d \geq 0$. Since we will usually only consider these inequalities in the unit hypercube, we will suppress 1 and 0 at the ends of the string of inequalities. Using the refinement from Theorems 1 (b), and then translating back with T^{-1}, gives a red refinement of S.

Theorem 2 *Consider a simplex S and its red refinement as defined above. Then*

a) each sub-simplex resulting from the red refinement step has a path consisting of the vectors $\{\frac{1}{2} f_i\}$ in some order, where $\{f_i\}$ are the vectors along the chosen path in S,
b) repeated red refinements using the paths parallel to the chosen path in S will produce only finitely many similarity types of simplices (up to scaling and rigid transformations),
c) each sub-simplex produced by a repeated red refinement has a path such that $\sin_d$ applied to the vectors along the path is equal to $\sin_d(f_1, \dots, f_d)$.

Proof Part (a) immediately follows from Theorem 1 and standard properties of affine transformations. Part (b) follows since each simplex produced is congruent to a simplex whose vertices are connected by a path consisting of $\{\frac{1}{2} f_i\}$ in some order. Part (c) follows since $\sin_d$ remains unchanged if one of the vectors is multiplied by a non-zero constant, and also after permutation of the inputs (see [5, 12]). □

Remark 1 The statements (a) and (b) in the above theorem are well known (see, e.g., [1]), and are included only for completeness. Statement (c) of Theorem 2 has been considered for tetrahedra in [15].

3 Main Results

In what follows we will need a result about the higher-dimensional sines when the arguments are orthogonal to each other.

Lemma 1 *Suppose that $\{v_1, \dots, v_{d_1}\}$ and $\{u_1, \dots, u_{d_2}\}$ are two sets of vectors that are orthogonal to each other. Then*

$$\sin_{d_1+d_2}(v_1, \dots, v_{d_1}, u_1, \dots, u_{d_2}) = \sin_{d_1}(v_1, \dots, v_{d_1}) \sin_{d_2}(u_1, \dots, u_{d_2}).$$

In case d_1 and/or d_2 is equal to one, we use the natural convention $\sin_1(u_1) = 1$ (sine of a single nonzero vector is one).

Proof We assume that each of the $n = d_1 + d_2$ vectors is a unit vector. Then

$$\begin{aligned}&\sin_n(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})\\ &\quad = \frac{\operatorname{vol}(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})^{n-1}}{\prod_{i=1}^{d_1} \operatorname{vol}(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})}\\ &\qquad \times \frac{1}{\prod_{i=1}^{d_2}(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{d_2})}.\end{aligned}$$

By orthogonality of the two sets of vectors, each hypervolume factors as a product of two lower-dimensional hypervolumes and we get:

$$\begin{aligned}&\sin_n(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})\\ &\quad = \frac{\operatorname{vol}(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})^{n-1}}{\prod_{i=1}^{d_1} \operatorname{vol}(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{d_1}, u_1, \ldots, u_{d_2})}\\ &\qquad \times \frac{1}{\prod_{i=1}^{d_2}(v_1, \ldots, v_{d_1}, u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{d_2})}\\ &\quad = \frac{\operatorname{vol}(v_1, \ldots, v_{d_1})^{n-1} \operatorname{vol}(u_1, \ldots, u_{d_2})^{n-1}}{\prod_{i=1}^{d_1} \operatorname{vol}(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{d_1}) \operatorname{vol}(u_1, \ldots, u_{d_2})^{d_1} \operatorname{vol}(v_1, \ldots, v_{d_1})^{d_2}}\\ &\qquad \times \frac{1}{\prod_{i=1}^{d_2}(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{d_2})}\\ &\quad = \frac{\operatorname{vol}(v_1, \ldots, v_{d_1})^{d_1-1}}{\prod_{i=1}^{d_1} \operatorname{vol}(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{d_1})} \frac{\operatorname{vol}(u_1, \ldots, u_{d_2})^{d_2-1}}{\prod_{i=1}^{d_2}(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{d_2})}\\ &\quad = \sin_{d_1}(v_1, \ldots, v_{d_1}) \sin_{d_2}(u_1, \ldots, u_{d_2}).\end{aligned}$$

□

Let $S_1, \ldots, S_k$ be simplices of dimensions $d_1, \ldots, d_k$, respectively and $n = \sum_{i=1}^{k} d_i$. Now we can independently choose a path (or equivalently a vertex ordering) in each S_i, and embed it in the unit d_i-cube by an affine transformation T_i as discussed before Theorem 2. By this choice, $T_i(S_i)$ is the reference d_i-simplex. The product polytope $\Delta = \prod S_i$ of the simplices is embedded into $\mathbb{R}^n$ by the product map T of the maps T_i:

$$T\colon \Delta = \prod_{i=1}^{k} S_i \to \prod_{i=1}^{k} T_i(S_i) \subset \prod_{i=1}^{k} \mathbb{R}^{d_i} = \mathbb{R}^n.$$

Theorem 3 *Consider the product of simplices* $\Delta = \prod S_i$ *as defined above. Then:*

a) The product Δ *can be triangulated by simplices, each of which contains a path with the property that* $\sin_n$ *of the path vectors is the product of the* $\sin_{d_i}$ *of the chosen paths of the factors.*
b) There is a red refinement scheme so that all simplices occuring by repeated refinement has a path with $\sin_n$ *equal to the* $\sin_n$ *in part (a), and that only contains a finite number of similarity types of product elements.*
c) There are also red refinement schemes where the factors are refined at different rates, so that each simplex occuring by repeated refinement has a path with $\sin_n$ *equal to the* $\sin_n$ *in part (a). In this case, the family may contain an infinite number of similarity types of product elements.*

Proof We apply the affine transformation T defined above, solve the problem in the unit n-cube, and transform using T^{-1} to get back to the original polytope Δ. Look at the coordinates of $T(\Delta)$. For each factor,

$$T_i(S_i) = \left\{ \left(x_1^{(i)}, \ldots, x_{d_i}^{(i)}\right) \mid x_1^{(i)} \geq x_2^{(i)} \geq \cdots \geq x_{d_i}^{(i)} \right\} \cap [0,1]^{d_i} \subset \mathbb{R}^{d_i}.$$

The product $T(\Delta) = \prod T_i(S_i)$ can then be described as a product of these sets with the given coordinates. More precisely, if we use

$$x_1^{(1)}, \ldots, x_{d_1}^{(1)}, x_1^{(2)}, \ldots, x_{d_2}^{(2)}, \ldots, x_1^{(k)}, \ldots, x_{d_k}^{(k)}$$

as coordinates for $\mathbb{R}^n$, we can use the k sets of the same inequalities to describe $T(\Delta)$. We then consider the triangulation of the hypercube in $\mathbb{R}^n$ from Theorem 1. A triangulation of $T(\Delta)$ then uses only those simplices in the triangulation of the hypercube that always satisfy these inequalities. The index set of these simplices can be given by *shuffles*, i.e., permutations of $\{1, 2, \ldots, n\}$ that preserve the order of $\{1, \ldots, d_1\}, \{d_1 + 1, \ldots, d_1 + d_2\}, \ldots, \{d_1 + \cdots + d_k + 1, d_1 + d_2 + \cdots + d_k\}$ separately.

The results in parts (a) and (b) now follow immediately as in Theorem 2, using only that if a given set of n vectors splits as a union of sets of cardinality $d_1, d_2, \ldots, d_k$ such that all the subsets are orthogonal to each other, the corresponding $\sin_n$ is the product of the corresponding $\sin_{d_i}$ (by Lemma 1 and an obvious induction).

For part (c), introduce the notation $f_j^{(i)}$ for the preimage under T of the standard basis vectors $e_j^{(i)}$ on $\mathbb{R}^n$, so that, e.g., $f_1^{(1)}, f_2^{(1)}, \ldots, f_{d_1}^{(1)}$ are the vectors along the chosen path of S_1. We can now perform the refinement on a single factor, keeping the remaining factors unchanged. Without loss of generality, we can assume that we only refine the first factor S_1. Then any simplex in the refined triangulation has a path with vectors $\frac{1}{2}f_1^{(1)}, \frac{1}{2}f_2^{(1)}, \ldots, \frac{1}{2}f_{d_1}^{(1)}$ in some order, and keeping the vectors $f_j^{(k)}$ for all j and for $k \geq 2$. Since $\sin_n$ is unchanged when arguments are

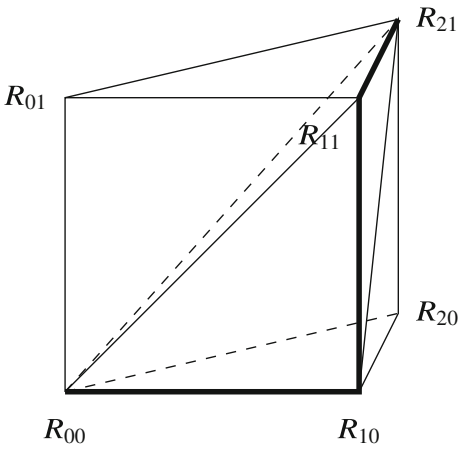


Fig. 2 The prism is split into three tetrahedra, each with a path containing the same three edge vectors in some order. This path is marked in bold for the middle tetrahedron

scaled, this does not change its value. We can then refine another factor, and keep doing this in any order, with any number of repetitions of factors. This process can in general produce infinitely many shapes of the product elements and of their simplicial subdivisions. □

Example 1 We illustrate the main theorem by the simplest nontrivial example, namely a triangular prism, i.e., a cartesian product of a triangle and an interval. It can be split into three tetrahedra in various ways. Let the chosen path in the triangle be R_{00}, R_{10}, R_{20} (see Fig. 2). Each of the three tetrahedra in the figure contains a path with one edge for each of the two edges in the chosen path and a third vertical edge. When we apply $\sin_3$ to these three edges, we get the same value as $\sin_2$ of the two edges in the triangle, which is the same as the ordinary sine of the angle at R_{10}. This value of $\sin_3$ will be preserved by the red refinement scheme described in part (c) of Theorem 3, i.e., when subdividing the height and the triangular base at independent rates.

Remark 2 The maximum angle condition from Definition 1 will be satisfied for all refinements of a product of simplices produced by Theorem 3. Since we have used the formulation with $\sin_n$ this is obvious. As the restricted process in part (b) of the theorem only produces a finite number of similarity types, that process will even preserve the standard regularity property, see [3]. In the general process in part (c), the factors can be subdivided at different rates, and therefore the regularity condition can be violated. We refer to [11] for more information about the case of prisms in this context.

Remark 3 We are currently working on extending the ideas from [13] to guarantee overall conformity of a mesh of product elements, not only for a single product element as considered in the present paper. Since this is more interesting in practice, we aim at numerical experiments in that setting rather than in the present case.

References

1. Bey, J.: Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes. Numer. Math. **85**, 1–29 (2000)
2. Billera, L.J., Cushman, R., Sanders, J.A.: The Stanley decomposition of the harmonic oscillator. Nederl. Akad. Wetensch. Indag. Math. **50**(4), 375–393 (1988)
3. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. North-Holland, Amsterdam (1978)
4. Eilenberg, S., Steenrod, S.: Foundations of Algebraic Topology. Princeton University Press, Princeton, NJ (1952)
5. Eriksson, F.: The law of sines for tetrahedra and n-simplices. Geom. Dedicata **7**, 71–80 (1978)
6. Fomenko, A., Fuchs, D.: Homotopical Topology, 2nd edn. Graduate Texts in Mathematics, vol.'273. Springer, Cham (2016)
7. Freudenthal, H.: Eine Simplizialzerlegung des Cartesischen Produktes zweier Simplexe. Fund. Math. **29**, 138–144 (1937)
8. Freudenthal, H.: Simplizialzerlegungen von beschränkter Flachheit. Ann. Math. **43**, 580–582 (1942)
9. Galashin, P., Nenashev, Postnikov, A.: Trianguloids and triangulations of root polytopes. Preprint. Available as https://arxiv.org/abs/1803.06239
10. Hannukainen, A., Korotov, S., Křížek, M.: Generalizations of the Synge-type condition in the finite element method. Appl. Math. **62**, 1–13 (2017)
11. Khademi, A., Korotov, S., Vatne, J.E.: On interpolation error on degenerating prismatic elements. Appl. Math **63**, 237–257 (2018)
12. Khademi, A., Korotov, S., Vatne, J.E.: On the generalization of the Synge-Křížek maximum angle condition for d-simplices. J. Comput. Appl. Math. **358**, 29–33 (2019)
13. Korotov, S., Křížek, M.: On conforming tetrahedralizations of prismatic partitions. In: Pinelas, S., et al. (eds.) Differential and Difference Equations with Applications. Springer Proceedings in Mathematics & Statistics, vol. 47, pp. 63–68. Springer Science+Business Media, New York (2013)
14. Korotov, S., Křížek, M.: Red refinements of simplices into congruent subsimplices. Comput. Math. Appl. **67**, 2199–2204 (2014)
15. Korotov, S., Vatne, J.E.: On regularity of tetrahedral meshes produced by some red-type refinements. In: Pinelas, S., et al. (eds.) Differential and Difference Equations with Applications. ICDDEA 2019. Springer Proceedings in Mathematics & Statistics, vol. 333. Springer, Cham. https://doi.org/10.1007/978-3-030-56323-3_49
16. Křížek, M.: An equilibrium finite element method in three-dimensional elasticity. Apl. Mat. **27**, 46–75 (1982)
17. Křížek, M.: On the maximum angle condition for linear tetrahedral elements. SIAM J. Numer. Anal. **29**, 513–520 (1992)
18. Zhang, S.: Successive subdivisions of tetrahedra and multigrid methods on tetrahedral meshes. Houston J. Math. **21**, 541–556 (1995)

Part III
Meshing and CAD

Global Parametrization Based on Ginzburg-Landau Functional

Victor Blanchi, Étienne Corman, Nicolas Ray, and Dmitry Sokolov

Abstract Quad meshing is a fundamental preprocessing task for many applications (subdivision surfaces, boundary layer simulation). State-of-the-art quad mesh generators proceed in three steps: first a guiding cross field is computed, then a parametrization representing the quads is generated, and finally a mesh is extracted from the parameterization. In this paper we show that in the case of a periodic global parameterization two first steps answer to the same equation and inherently face the same challenges. This new insight allows us to use recent cross field generation algorithms based on Ginzburg-Landau equations to accurately solve the parametrization step. We provide practical evidence that this formulation enables us to overcome common shortcomings in parametrization computation (inaccuracy away from the boundary, singular dipole placement).

1 Introduction and Related Work

Meshing is so central in geometric modeling because it provides a way to represent functions on the objects studied (texture coordinates, temperature, pressure, speed, etc.). There are numerous ways to represent functions, but if we suppose that the functions are piecewise smooth, the most versatile way is to discretize the domain of interest. Ways to discretize a domain range from point clouds to block-structured meshes; while the first are really easy to produce, the latter are extremely challenging due to the inherent structure that is very difficult to discover automatically.

V. Blanchi
École Normale Supérieure, Paris, France
e-mail: vblanchi@clipper.ens.fr

É. Corman · N. Ray · D. Sokolov (✉)
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
e-mail: Etienne.Corman@loria.fr; Nicolas.Ray@loria.fr; Dmitry.Sokolov@loria.fr

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_16

In this paper we are interested in the problem of quad mesh generation; although over the past years mesh generation have seen great advances and is now used in production, many challenges remained to be solved for a more practical and efficient use.

Common approaches to quad meshing like paving [3] or Q-morph [9] rely on advancing-front algorithms starting from a constraint, e.g., the boundary, and expanding it to fill the interior. When these advancing fronts meet, they have to be merged; merging two fronts of quads is a challenging task in itself prone to create many local singular vertices. Moreover, due to the propagation process, the mesh topology is fixed near the boundary in the early stage of the algorithm and it is often very difficult to go back on these early choices even if they create terrible quad configuration in the interior of the mesh. All these approaches are based on local decisions that do not consider the global structure of quad/hex meshes, and this is the key reason why quad/hex meshing is so hard. The atomic operation in a triangle/tetrahedral mesh is a simple addition of points; whereas in a quad/hexahedral mesh, it would be the addition quad strips/layers of hexahedra.

While front propagation methods are usually fast and robust (for the 2D case), their output is often not as structured as we would like it to be. Kälberer et al. [5] proposed another type of approach, giving extremely nice looking meshes; unfortunately, it is not guaranteed to be able to mesh all possible domains. The main motivation comes from the observation that good quality quad meshes look like a deformed grid almost everywhere. The idea is to define a deformation of the object such that if the final quad mesh (the result) undergoes this deformation, it matches a unit, axis aligned grid. The direct application of this idea computes this deformation, applies its inverse to the unit grid, and obtains a quad mesh. In practice, it is better to introduce more degrees of freedom by considering a global parameterization instead of a deformation. In this case, parameterizations have some discontinuities that make it possible to represent a much larger family of quad meshes: the deformed grid can be cut and glued to itself in a non-trivial way.

All global parameterization approaches are decomposed into three steps (refer to Fig. 1):

1. *Frame field generation:* this step defines the orientation of the grid at each point of the domain.
2. *Field integration:* this step computes the position and the size of the grid cells aligned with the input orientation field.
3. *Final mesh extraction:* at this step we map the grid onto the original object, thus creating the final quad/hex mesh.

In this paper we focus on the second step, namely the parameterization. Since our approach (periodic global parameterization [12]) is strongly tied to the frame field generation problem, let us review it first.

Frame Field Generation A frame is a set of 4 unit vectors $\{z_k\}_{k=0}^3 \subset \mathbb{C}$ which is invariant by a $\pi/2$ rotation. Note that due to its symmetry, a frame can be represented by a single unit vector z such that

$$\{z_k\}_{k=0}^3 = \sqrt[4]{z}\left\{1, \mathrm{e}^{\mathrm{i}\pi/2}, -1, \mathrm{e}^{-\mathrm{i}\pi/2}\right\}.$$

Following this definition, we represent frame fields by complex valued fields. We note as $\Re(z)$ and $\Im(z)$ the real and imaginary parts, respectively.

Frame field generation is a simple problem to state: we are looking for the smoothest *unit* field under boundary constraints. Here, smoothest is equivalent to minimizing the Dirichlet energy while the boundary conditions prescribe one of the 4 frame vectors to be aligned with the normal. More formally, we are looking to minimize the following energy:

$$\arg\min_{|z|=1} \int_\Omega |\nabla z|^2 \quad \text{subject to boundary constraints.} \tag{1}$$

The main challenge is dealing with the unit norm constraint; the very first methods that introduced frame field design for orienting strokes in non-photorealistic rendering [4], represented frame fields by an angle $\mathrm{Arg}\, z$ per vertex, so the unit norm was naturally respected. The optimization was performed by a non-linear solver (BFGS). There is a catch though: Dirichlet energy of a unit frame field is not a reliable measure of quality. While its evaluation on a mesh is finite, in the continuous case the integral diverges, leading to numerically challenging computations. Due to this problem this kind of approaches struggled from bad singularity placement.

Later methods [10, 12, 13] choose to optimize for vector fields directly without the unit norm constraint; the field is normalized in post-processing. This kind of approaches improved greatly the quality of results (and the running times, since only a linear system needs to be solved), but still the geometry of the fields suffered from the unit norm constraint being relaxed. The energy at singular points is well-defined but away from the boundary constraint the frame field is close to zero, making the computation unreliable. Later on, Knöppel et al. [6] proposed to constrain overall norm of the field to be unit (as opposed to the real per-point unit constraint) by solving an eigenvector problem. Fixing the norm of the entire solution instead of the norm of each vector allows to find optimal frame field on closed surfaces, but in presence of boundary constraint the fields are still suboptimal. The latest advances in frame fields [1] proposed to use the Ginzburg-Landau functional to enforce the unity constraint while keeping a well-defined energy. The idea is to redefine the problem (1) as follows:

$$\arg\min_{\text{arbitrary } |z|} \int_\Omega |\nabla z|^2 + P_\varepsilon(z) \quad \text{subject to boundary constraints.} \tag{2}$$

Here $P(z)$ is a penalty term that enforces the field to have a point-wise unit norm. Beaufort et al. [1] propose to choose $P_\varepsilon(z) = \frac{1}{4\varepsilon^2}(|z|^2 - 1)^2$; this problem is well-defined, and can be solved by Newton iterations. As ε tends to zero, the field converges to a unit-norm field, while minimizing the field variation. Viertel et al. [17] suggested another optimization scheme for (2) based on heat diffusion. Both methods yield similar results in term frame field quality. It is to be noted that, while these approaches improve the geometry of frame fields in some cases, the problem is still non convex, and does not provide guarantees of optimality.

Parametrization The parameterization step consists in integrating the frame field, while imposing integer constraints at the boundary and singular points. More precisely, one needs to find two scalar functions [5] whose gradients are as close as possible to the input frame field under the integer constraints. When succeeding, these approaches provide impressive results. There are, however, many limitations. The frame field may not be integrable in the sense that is does not locally correspond to the gradient of a scalar function. This problem can be mitigated by asking the frame field to be curl-free [12, 15, 16]. However this does not prevent a major problem: some frame fields are not consistent with any quadrangulation. Top left image of Fig. 1 provides an example: the radial frame field (without any singularities!) cannot be integrated directly; the only way to compute a parameterization is to insert a dipole (index $1/4$ and $-1/4$) in the scalar fields.

Getting a *bijective* parametrization for quad-meshing is an open problem. The most advanced methods follow the path of [8] by computing a *motorcycle graph* which is a partition of the surface into quads with possible T-junctions. This intermediate state, in between the full quad mesh and the triangle mesh, allows the authors to formulate the targeted grid aligned parametrization into a mixed-integer problem whose variables are subdivisions or collapses of the T-mesh. Even though this is currently the most robust method available, each step has

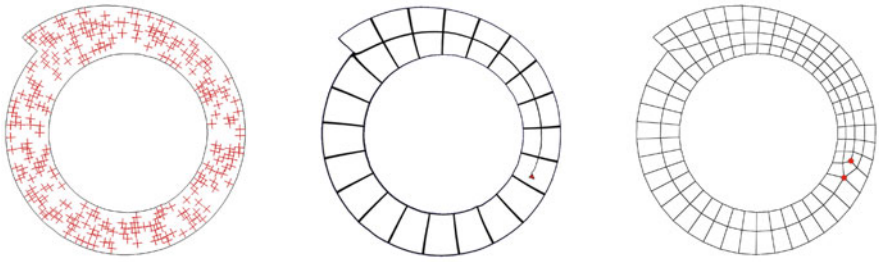


Fig. 1 Quad meshing *via* global parameterization. Left: a cross field prescribing the orientation of the elements. Middle: a parameterization prescribing the size and the position of the elements to place. Note that it is *not* a quad mesh, it is a unit grid texture image mapped to the triangle mesh. In this case it is a periodic global parameterization, the red triangle shows a singularity of the parameterization. Right: a quad mesh extracted from the parameterization. The singularity of the parameterization corresponds to a dipole (a pair of vertices of valence 3 and 5, shown in red)

its practical shortcoming. For instance, extracting motorcycle graphs from a frame field is a notoriously challenging task as it requires to trace non-intersecting streamlines [8, 11]. The intermediate T-mesh has many T-junctions which must be resolved either by collapsing the edge into a quad or by placing a dipole of singularities. In either case it requires to robustly solve a mixed-integer problem which is often NP-complete.

In this landscape algorithms in the wake of Periodic Global Parametrization (PGP) [12, 16] stand aside. The main idea is to consider quad mesh edges as zeros of two oscillators integrating orthogonal branches of the frame. Thus, all integer constraints are built in the algorithm by design and extracting a mesh does not require to compute streamlines.

The optimization problem involved in PGP is nearly identical to the one for computing frame field: a modified Dirichlet energy with Dirichlet boundary conditions. Thus, similar challenges regarding unit norm constraints and singularity placement arise. The original paper [12] solves the optimization problem by relaxing the unit norm constraint, leading to a nearly zero solution away from the boundary.

Based on the frame field generation experience, we propose an optimization strategy based on the Ginzburg-Landau functional [2]. It correctly specifies the oscillators away from constraints, creates more regular quads and improves the quality of the mesh near singularities.

N.B. For the sake of clarity, we focus on integrating *singularity-free* frame fields. PGP can deal with singular fields through a quad-covering of the surface [12], but we found it unnecessary to show the potential of the Ginzburg-Landau functional in the context of field integration.

2 Periodic Global Parametrization

In this paper we mesh a planar domain $\Omega \subset \mathbb{R}^2$. In order to simplify the presentation, our input frame fields are without singularities: it allows us to extract two continuous orthogonal vector fields from the input frame field; we integrate these vector fields independently one from another. In this section the vector field to be integrated is denoted as V and $\partial\Omega^v$ denotes the subset where V is orthogonal to the boundary.

2.1 PGP Basics

In the standard quad meshing pipeline the edges of the resulting mesh are orthogonal to one of the frame field directions. The goal of the integration step is to compute a global parametrization whose integer level sets respect this constraint. The most

basic scheme [5], which laid ground for many quad meshing methods, is to optimize for parameters (u, v) such that their gradients are equal to one of the directions of the frame. At the same time integer constraints are imposed in order to conform the boundary of the domain with the boundary of the quad mesh.

Another possibility is to look for a periodic function. Indeed, the periodic function $z(p) = e^{2i\pi\theta(p)}$ has clearly identified integer level sets which can be easily aligned with the boundary. A level set of z is orthogonal to the vector field V whenever V is equal to the direction of oscillation $\nabla\theta$. Thus taking the gradient of z, the periodic function must satisfy [7, 12]:

$$\nabla z = 2i\pi V z.$$

By specifying the norm of V, we determine the speed of oscillation and thus the size of the quads.

As the mesh must conform to the boundary of the domain, we have to enforce the Dirichlet boundary constraint $z = 1$. Moreover z has pointwise unit norm over the domain. So integrating the vector field V with a global periodic function amounts for solving the following optimization problem:

$$\begin{aligned} \min_{z\in\mathbb{C}} \quad & \frac{1}{2}\int_\Omega |\nabla z - 2i\pi V z|^2 \\ \text{s.t.:} \quad & z = 1 \text{ on } \partial\Omega^v, \\ & |z| = 1 \text{ on } \partial\Omega. \end{aligned} \tag{3}$$

The optimization problem in (3) is non-convex due to the unit norm constraint. The authors of original paper [12] simply removed this constraint altogether to recover a quadratic optimization similar to a Laplacian smoothing operation. Note that if not for the boundary constraints, the solution of (3) would be equal to zero everywhere. So a common issue with this approach is that far away from the constraint the norm of z is numerically close to zero making the integration unreliable (see Fig. 2 top row). The Ginzburg-Landau functional is a sound way of accurately solving (3).

2.2 *Ginzburg-Landau to the Rescue*

As mentioned earlier, (3) is a non-convex optimization problem because of the unit norm constraint. Recent work on frame field generation used the Ginzburg-Landau functional to deal with such a constraint. In practice this strategy allows for a better placement of singularities [1]. Note that our problem (3) is very similar to the frame

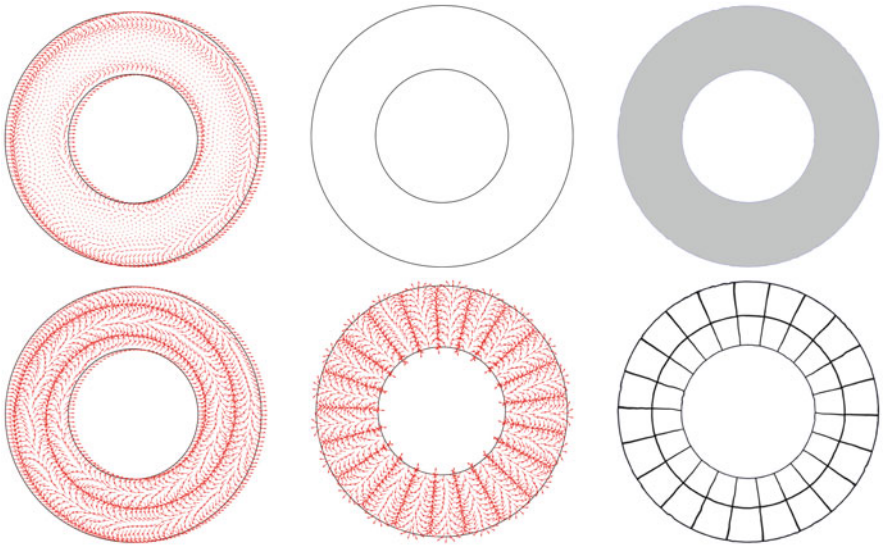


Fig. 2 Quad meshing *via* Periodic Global Parametrization. Left and middle: periodic function z respectively integrating the radial and the tangent direction field. Right: quad mesh extracted from the parameterization. Top row: the integration with PGP causes a drastic norm reduction in the integration of the radial field and the tangent field integration simply outputs an everywhere vanishing field. Bottom row: in comparison, our Ginzburg-Landau based optimization yield a unit norm vector field on both directions and a valid quad mesh is extracted

field generation problem (2); following this idea we propose to use a Ginzburg-Landau functional for periodic global parametrization:

$$z_\varepsilon = \arg\min_{z\in\mathbb{C}} E(z) + P_\varepsilon(z) = \frac{1}{2}\int_\Omega |\nabla z - 2\mathrm{i}\pi V z|^2 + \frac{1}{4\varepsilon^2}\int_\Omega \left(|z|^2 - 1\right)^2 \tag{4}$$

$$\text{s.t.:} \quad z = 1 \text{ on } \partial\Omega^v.$$

Theoretical results [14] demonstrated that as ε goes to zero, the complex function z tends to be a unit complex number everywhere except at isolated singular points. Asymptotically the singular points have integer indexes and are placed in a way that minimizes the overall field curvature.

In the rest of the paper we show how to discretize our problem with standard finite elements and solve the optimization problem with a Newton method for decreasing value of ε.

2.3 Newton Method

In order to solve the optimization problem of (4), we use a modified Newton method. More precisely, a local extremum of the energy is reached whenever z_ε is a stationary point of the energy i.e. $\nabla E(z_\varepsilon) + \nabla P_\varepsilon(z_\varepsilon) = 0$. Thus, after explicitly differentiating E and P_ε, the oscillator z_ε must satisfy the non-linear PDE:

$$\begin{cases} \nabla E(z_\varepsilon) + \nabla P_\varepsilon(z_\varepsilon) = 0 & \text{on } \Omega, \\ z_\varepsilon = 1 & \text{on } \partial\Omega^v, \end{cases} \tag{5}$$

where the gradients are explicitly given by:

$$\begin{cases} \nabla E(z_\varepsilon) = -\Delta z + 4\pi^2 |V|^2 z + \mathrm{i}\pi \left(\langle \nabla z, V \rangle + \operatorname{div}(Vz) \right) \\ \nabla P_\varepsilon(z) = \frac{1}{\varepsilon^2} z \left(|z|^2 - 1 \right) \end{cases} \tag{6}$$

For each value of ε, we approximate the solution of (5) with Newton iterations. To realize this scheme, we need an expression of the Hessian matrix as a function of point z. The closed-form expression of the Hessian is easier to read as a matrix applied to the vector $\begin{pmatrix} \Re(z) & \Im(z) \end{pmatrix}^\top$ containing the real and imaginary part of the complex:

$$\begin{cases} H_E(z)h = -\Delta h + 4\pi^2 |V|^2 h + \mathrm{i}\pi \left(\langle \nabla h, V \rangle + \operatorname{div}(Vh) \right) \\ H_{P_\varepsilon}(z)h = \frac{1}{\varepsilon^2} \begin{pmatrix} 3\Re(z)^2 + \Im(z)^2 - 1 & 2\Im(z)\Re(z) \\ 2\Im(z)\Re(z) & 3\Im(z)^2 + \Re(z)^2 - 1 \end{pmatrix} \begin{pmatrix} \Re(h) \\ \Im(h) \end{pmatrix} \end{cases} . \tag{7}$$

Newton iterations are well-defined only when the Hessian is positive definite. The first term $E(z)$ of the energy in (4) is convex thus its Hessian is positive. The second term P_ε constraining the unit norm is non-convex and its Hessian is negative near 0. In this case it is a common practice to approximate the Hessian with a positive definite matrix by removing all negative terms from the Hessian (7):

$$\tilde{H}_{P_\varepsilon}(z)h = \frac{1}{\varepsilon^2} \begin{pmatrix} 3\Re(z)^2 & 2\Im(z)\Re(z) \\ 2\Im(z)\Re(z) & 3\Im(z)^2 \end{pmatrix} \begin{pmatrix} \Re(h) \\ \Im(h) \end{pmatrix} \tag{8}$$

Therefore, for a fixed ε we compute the sequence of oscillator z_ε^n satisfying:

$$\begin{cases} \left(H_E\left(z_\varepsilon^n\right) + \tilde{H}_{P_\varepsilon}\left(z_\varepsilon^n\right) \right) h_\varepsilon^{n+1} = -\nabla E\left(z_\varepsilon^n\right) - \nabla P_\varepsilon\left(z_\varepsilon^n\right) \\ z_\varepsilon^{n+1} = z_\varepsilon^n + h_\varepsilon^{n+1} \end{cases} , \tag{9}$$

until convergence to a stationary point.

2.4 Discretization

We discretize (9) with standard first order finite elements. The oscillator z is a complex per vertex and the vector field V is encoded as a two dimensional vector per vertex. Both are linearly interpolated on each triangle with "hat" functions φ (Fig. 3). On the triangle (ijk) the interpolation, denoted with superscript ^{ijk}, reads: $z^{ijk} = z_i\varphi_i + z_j\varphi_j + z_k\varphi_k$.

The Hessian (8) are square complex matrices of size the number of vertices. They are assembled by accumulated the 6×6 elementary matrices computed on an element Ω_{ijk}, given by:

$$\begin{cases} H_E^{ijk} & = \int_{\Omega_{ijk}} -\langle \nabla\varphi_n, \nabla\varphi_m\rangle + 4\pi^2 |V^{ijk}|^2 \varphi_n\varphi_m \\ & \quad + \mathrm{i}\pi \left(\langle \nabla\varphi_n, V^{ijk}\rangle + \mathrm{div}\left(V^{ijk}\varphi_n\right)\right)\varphi_m, \\ H_{P_\varepsilon}^{ijk} & = \frac{1}{\varepsilon^2} \begin{pmatrix} 3\int_{\Omega_{ijk}} \Re(z^{ijk})^2 \varphi_m\varphi_n & 2\int_{\Omega_{ijk}} \Im(z^{ijk})\Re(z^{ijk})\varphi_m\varphi_n \\ 2\int_{\Omega_{ijk}} \Im(z^{ijk})\Re(z^{ijk})\varphi_m\varphi_n & 3\int_{\Omega_{ijk}} \Im(z^{ijk})^2 \varphi_m\varphi_n \end{pmatrix}, \end{cases}$$

for $m, n = 1, 2, 3$. Similarly, (5) is turned into a vector of complex of the size number of vertex. The elementary vector of size 3 on an element Ω_{ijk} are given by:

$$\begin{cases} \nabla E^{ijk} & = \int_{\Omega_{ijk}} -\langle \nabla z^{ijk}, \nabla\varphi_m\rangle + 4\pi^2 |V^{ijk}|^2 z^{ijk}\varphi_m \\ & \quad + \mathrm{i}\pi \left(\langle \nabla z^{ijk}, V^{ijk}\rangle + \mathrm{div}\left(V^{ijk} z^{ijk}\right)\right)\varphi_m, \, . \\ \nabla P_\varepsilon^{ijk} & = \frac{1}{\varepsilon^2}\int_{\Omega_{ijk}} z^{ijk}\left(\left|z^{ijk}\right|^2 - 1\right)\varphi_m, \end{cases}$$

with $m = 1, 2, 3$.

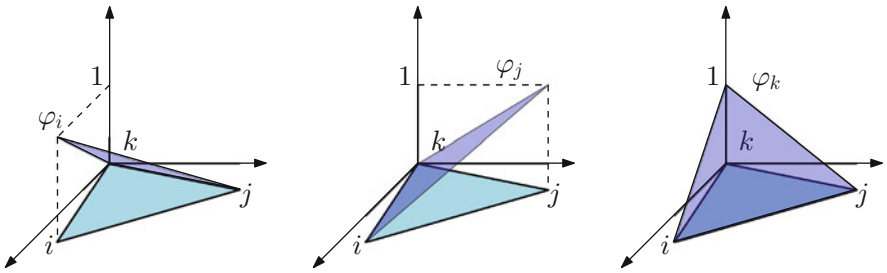


Fig. 3 The three local FEM basis functions on the element ijk

3 Results

This article focuses on the parametrization step of the quad meshing pipeline, thus the results shown in Fig. 4 show a unit grid texture image applied to the domain, we do not extract actual quad meshes.

For the sake of clarity, we presented an algorithm for integrating *non-singular* frame fields over a *flat* 2D domain. While it is perfectly possible to compute a periodic global parameterization of a 3D triangulated surface with Ginzburg-Landau functional, as well as to incorporate the frame field singularities in the computation, the notations become cumbersome and we prefer to avoid it.

For Fig. 1, the vector fields are obtained by solving (2) as specified in [1]. For the other examples (Figs. 4 and 5), the vector fields are computed as two independent orthogonal unit vector fields U and V. First, we partition the boundary in two complementary sets based on the normal orientation. If the normal is close to the x-axis, the point belongs to $\partial\Omega^u$, but if it is closer to the y-axis, it belongs the set $\partial\Omega^v$. Second, we find the smoothest non-singular unit vector field U normal to the boundary on $\partial\Omega^u$ and tangent to the boundary on $\partial\Omega^v$ by smoothly interpolating the angle of the vector. The vector field V is obtained by 90° rotation of U.

The vector fields U, V are then integrated separately by doing the Newton iterations of (9) for decreasing values of ε. The boundary conditions naturally arise from the construction of the vector fields, i.e., $U = 1$ on $\partial\Omega^u$ and $V = 1$ on $\partial\Omega^v$.

The frame field on Fig. 1 exhibits a limit cycle making it non-integrable. Thus standard integration algorithms often fail to output a valid quad mesh. Our method is able to correctly place a singularity (middle image) which can be turn it a quad mesh (right image). Frame fields used for our results are singularity-free, yet our algorithm is able compute smooth parametrizations with a minimal amount of

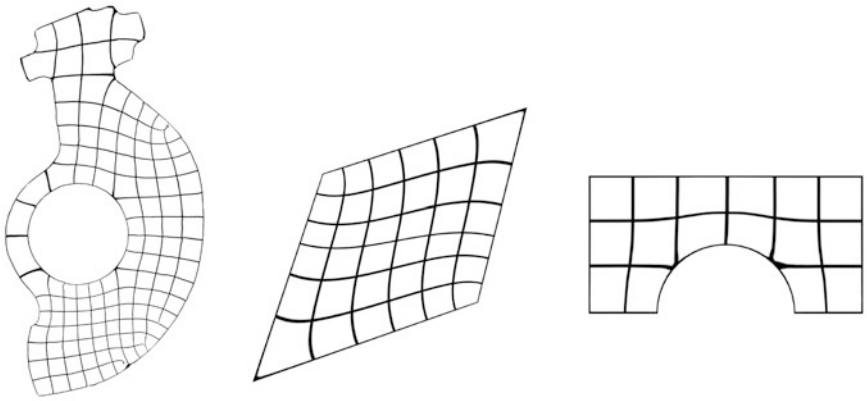


Fig. 4 Examples of parametrization obtained with our resolution of periodic global parametrization with Ginzburg-Landau functional. Note that the algorithm tries to limit the number of singularities and optimally places unavoidable dipoles

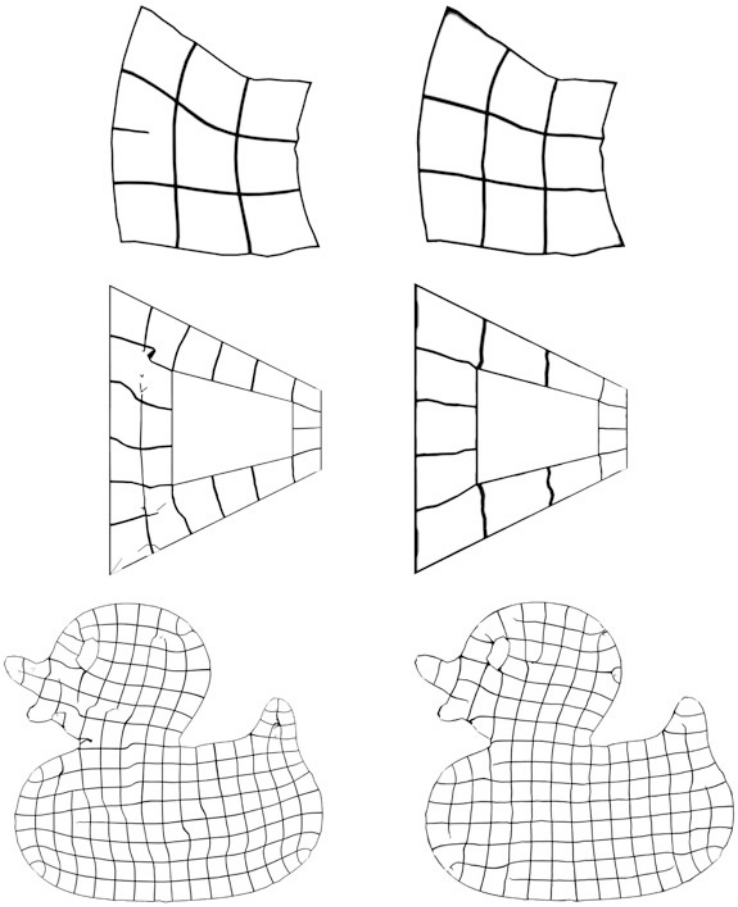


Fig. 5 Comparison between one step of periodic global parametrization (left) and our periodic global parametrization with Ginzburg-Landau functional (right). Note that our algorithm removes unnecessary singularities and produces more regular quads

parameterization singularities as shown in Fig. 4. In Fig. 5, we compare our results with a naive implementation of PGP, simply solving (3) as a quadratic problem by removing the unit norm constraint. It is easy to see that the parameterization obtained with the Ginzburg-Landau functional is more regular and presents less singularities. Moreover, in the some dramatic cases, omitting the unit norm constraint makes PGP unable to output valid parametrization (see Fig. 2)

4 Conclusion

The main goal of this article is to underline the similar nature between frame field generation and periodic global parameterizations. Both problems can be solved by (almost) the same set of equations using the Ginzburg-Landau functional. This article brings a solid theoretical basis that allows to solve accurately the periodic global parameterization problem, making the problem well-posed and leading to an automatic and optimal singularity placement in the parametrization.

References

1. Beaufort, P.A., Lambrechts, J., Henrotte, F., Geuzaine, C., Remacle, J.F.: Computing cross fields a pde approach based on the ginzburg-landau theory. Procedia Eng. **203**, 219–231 (2017)
2. Bethuel, F., Brezis, H., Hélein, F., et al.: Ginzburg-landau Vortices, vol. 13. Springer, New York (1994)
3. Blacker, T.D., Stephenson, M.B.: Paving: A new approach to automated quadrilateral mesh generation. Int. J. Numer. Methods Eng. **32**(4), 811–847 (1991). https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620320410
4. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: Proceedings of SIGGRAPH 2000, pp. 517–526 (2000)
5. Kälberer, F., Nieser, M., Polthier, K.: Quadcover-surface parameterization using branched coverings. In: Computer Graphics Forum, vol. 26.3, pp. 375–384. Wiley Online Library (2007)
6. Knöppel, F., Crane, K., Pinkall, U., Schröder, P.: Globally optimal direction fields. ACM Trans. Graph. **32**(4) (2013)
7. Knöppel, F., Crane, K., Pinkall, U., Schröder, P.: Stripe patterns on surfaces. ACM Trans. Graph. **34**(4), 1–11 (2015)
8. Myles, A., Pietroni, N., Zorin, D.: Robust field-aligned global parametrization. ACM Trans. Graph. **33**(4), 1–14 (2014)
9. Owen, S.J., Staten, M.L., Canann, S.A., Saigal, S.: Q-morph: an indirect approach to advancing front quad meshing. Int. J. Numer. Methods Eng. **44**(9), 1317–1340 (1999). http://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0207%2819990330%2944%3A9%3C1317%3A%3AAID-NME532%3E3.0.CO%3B2-N
10. Palacios, J., Zhang, E.: Rotational symmetry field design on surfaces. ACM Trans. Graph. **26**(3) (2007). http://doi.acm.org/10.1145/1276377.1276446
11. Ray, N., Sokolov, D.: Robust polylines tracing for n-symmetry direction field on triangulated surfaces. ACM Trans. Graph. **33**(3), 1–11 (2014)
12. Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P.: Periodic global parameterization. ACM Trans. Graph. **25**(4), 1460–1485 (2006)
13. Ray, N., Vallet, B., Alonso, L., Levy, B.: Geometry-aware direction field processing. ACM Trans. Graph. **29**(1), 1:1–1:11 (2009). http://doi.acm.org/10.1145/1640443.1640444
14. Rivière, T., Given, M.c., Harpes, P.: Asymptotic analysis for the ginzburg-landau equations (1997)
15. Sageman-Furnas, A.O., Chern, A., Ben-Chen, M., Vaxman, A.: Chebyshev nets from commuting polyvector fields. ACM Trans. Graph. (TOG) **38**(6), 1–16 (2019)
16. Sokolov, D., Ray, N., Untereiner, L., Lévy, B.: Hexahedral-dominant meshing. ACM Trans. Graph. **35**(5) (2016). https://doi.org/10.1145/2930662
17. Viertel, R., Osting, B.: An approach to quad meshing based on harmonic cross-valued maps and the ginzburg–landau theory. SIAM J. Sci. Comput. **41**(1), A452–A479 (2019)

Parametrization of Plane Irregular Regions: A Semi-automatic Approach I

Pablo Barrera and Iván Méndez

Abstract In some problems, the solutions of partial differential equations use parametrizations of plane regions. However, it is difficult to get suitable parametrizations of irregular regions. In this paper we introduce a method for finding a parametrization of a polygonal region Ω. Our method decomposes Ω into a finite collection of admissible subregions. We use compatible parametrizations of these subregions to construct the parametrization of Ω as a block structured mesh.

1 Introduction

The parametrizations of irregular regions have many applications in Computer Aided Design, Engineering Modeling, and Shape Recognition [15]. The Finite Element Analysis and the Isogeometric Analysis use parametrizations of plane regions to solve partial differential equations [10].

Our UNAMALLA workgroup has generated bilinear and biquadratic B-splines parametrizations of simply connected regions using structured mesh generation [1, 2]. Nevertheless, some cells are elongated or skewed on highly irregular regions. This limitation can be overcome by splitting the region into subregions suitable for parametrization.

The problem we address is to decompose polygonal regions into admissible subregions and generate compatible parametrizations of these subregions. We want a method to solve this problem.

Researchers in Computer Aided Design have developed some methods for constructing compatible parametrizations of 2D regions. In that regard, Xu G. et al. [18, 19] have constructed high quality parametrizations by solving constrained optimization problems. Even so the overall process is computationally expensive for irregular regions.

P. Barrera · I. Méndez (✉)
Faculty of Sciences, National Autonomous University of Mexico, Mexico City, Mexico
e-mail: pablobarrera@ciencias.unam.mx; vanmc@ciencias.unam.mx

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_17

On the other hand, admissible subregions and their compatible parametrizations can be efficiently constructed by automatic methods of block structured mesh generation. Zint et al. [21] use triangulations for region decomposition, so the block connectivity depends on the triangulation. Bommes et al. [6] use the singularities of a cross-field to construct a family of parametrizations between coarse quad layouts of surfaces. Nevertheless, as the number of singularities increases, the number of layouts also increases. Recently, Xiao et al. [17] use the singular points of a cross-field to generate high quality structured meshes with smooth lines between the subregions. However, they provide examples in which the geometry is not irregular. On the other hand, Zhang et al. [20] decompose a polygonal region into a main block with multiple subregions organized in a hierarchy structure. Mesh generation is automatically carried out block by block from the main root block to the highest level. This method is not easy to extend to multiply-connected regions.

In this paper we propose a method to find both admissible subregions and their compatible parametrizations. Our method consists of two stages: region decomposition and parametrization construction.

Motivated by Liu et al.[13], we propose in Sect. 2 an interactive region decomposition method to obtain admissible subregions. In Sect. 3 the parametrization construction is carried out from the boundary to the interior of the subregions. First, the subregion boundaries are approximated by compatible B-splines curves in Sect. 3.1. Then, these curves are extended into the whole region by structured mesh generation in Sect. 3.2. Finally, we summarize the main steps of our method and show some examples to illustrate its robustness in Sect. 4.

2 Region Decomposition

Region decomposition is a fundamental step in our method. Let Ω be a a counterclockwise oriented polygonal region. We decompose Ω into non-overlapping admissible polygons. A polygonal region is *admissible* if it has a parametrization such that its cells are rectangle-like quadrilaterals.

The key point of our region decomposition is the concavity. Lien [12] decomposes recursively a polygon into approximate convex polygons. He introduces concavity criteria to decompose polygons. Later, Liu et al. [13] proposed the *Dual-space Decomposition* (DUDE) to split polygons using their convex complements.

2.1 Concavity Measures and Admissible Regions

We use some concepts introduced in Lien [12] and Liu et al. [13] to measure the concavity of Ω. Let $H(\Omega)$ be the convex hull of Ω. The convex complement of Ω is $H(\Omega) \setminus \Omega$. The *bridges* of Ω are line segments contained in the convex complement of Ω which join two points in $\partial\Omega$. Each bridge β has a *pocket* ρ, that

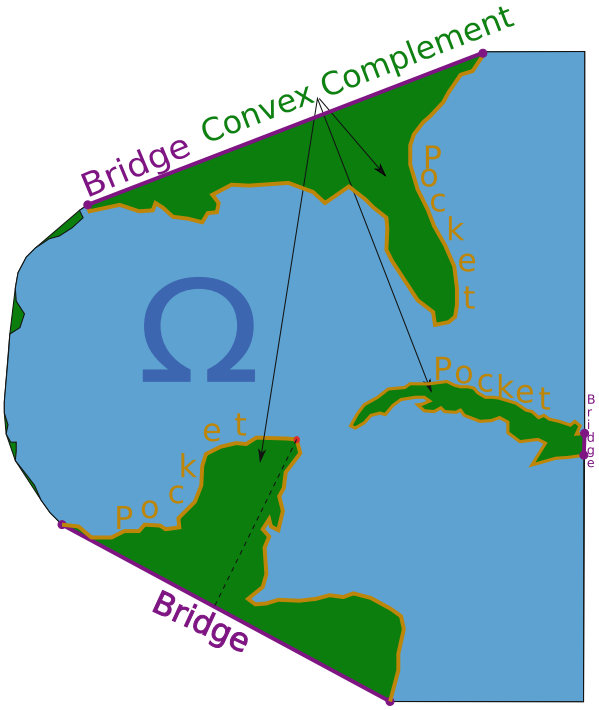


Fig. 1 Convex complement of the region `Gulf` and some of its bridges and pockets. The red point has the largest concavity in the corresponding pocket

is, the polygonal curve of $\partial\Omega$ with the smallest length which joins the ending points of β (Fig. 1).

We measure the concavity of pocket points and bridges. Let β be a bridge of Ω with pocket ρ. The concavity of a point $\boldsymbol{x}$ of ρ is the distance from $\boldsymbol{x}$ to β. This distance is the straight line distance to β or the arc length of the polygonal curve in ρ which joins $\boldsymbol{x}$ with the nearest ending point of β.

The concavity of a bridge is the largest concavity of its pockets points. We measure the concavity of the bridges using the straight line distance to the bridges. The largest concavity of the bridges of Ω is denoted by $c_1(\Omega)$. The region Ω is scaled inside a circle of radius one centered at the centroid of Ω so that $c_1(\Omega)$ be scale independent.

In addition to the concavity measures in [12, 13], we introduce the concavity measure

$$c_2(\Omega) := \frac{\text{Area}(H(\Omega)) - \text{Area}(\Omega)}{\text{Area}(H(\Omega))}. \tag{1}$$

This is the relative size of region with respect to its convex hull.

Regions with small concavity measures are suitable for parametrization, so we want subregions of Ω that satisfy the following concavity criteria:

- 1° *concavity criterion:* The concavity of the bridges of Ω is small.

$$c_1(\Omega) \leq \tau_1, \quad \tau_1 \in (0, 1). \tag{2}$$

- 2° *concavity criterion:* The area difference between the region and its convex hull is small.

$$c_2(\Omega) \leq \tau_2, \quad \tau_2 \in (0, 1). \tag{3}$$

Regions which satisfy both concavity criteria are admissible regions. So convex regions are admissible. On the other hand, non-convex regions which do not satisfy the previous criteria are decomposed into admissible subregions. We want an *admissible decomposition* of Ω, that is, a collection $\{\Omega_k\}_{k=1}^n$ of polygonal subregions of Ω such that

1. $\Omega = \cup_{k=1}^n \Omega_k$.
2. $\Omega_i \cap \Omega_j \neq \emptyset \Longrightarrow \Omega_i \cap \Omega_j \subset \partial\Omega_i \cap \partial\Omega_j \quad \forall i, j$.
3. Each Ω_k satisfies both concavity criteria (2)–(3).

2.2 Decomposition Method

Our decomposition method is interactive. It uses some ideas of DUDE [13]. The region Ω is recursively split into two subregions Ω_1 and Ω_2 by a cut when Ω does not satisfy the concavity criteria (2)–(3) (Fig. 2).

A *cut* of Ω is a line segment in the interior of Ω which joins two points of $\partial\Omega$. We make a cut in each step of our method. However, not just any cut separates Ω into admissible subregions. We propose an interactive cut choice method based on concavity measures. It consist of the following steps:

1. Compute the convex hull $H(\Omega)$
2. Find the pockets of Ω with ending points in the boundary of $H(\Omega)$. Select the pockets such that their union with the corresponding bridges have large area.
3. In each pocket compute the concavity of their points using either the straight line distance or the arc length. Select a point $\boldsymbol{x}_1$ with large concavity in a pocket ρ_1 as an ending point of the cut.
4. Select a point $\boldsymbol{x}_2$ with large concavity in a pocket different from ρ_1 such that the segment $\overline{\boldsymbol{x}_1\boldsymbol{x}_2}$ is a cut, else choose $\boldsymbol{x}_2$ as an intersection point of $\partial\Omega$ with the line P perpendicular to the bridge of ρ_1 which passes at $\boldsymbol{x}_1$. Otherwise choose $\boldsymbol{x}_2 \in \partial\Omega \setminus \rho_1$ with the smallest straight line distance to $\boldsymbol{x}_1$ such that the segment $\overline{\boldsymbol{x}_1\boldsymbol{x}_2}$ does not cross previous cuts of Ω (Fig. 3).

The segment $\overline{\boldsymbol{x}_1\boldsymbol{x}_2}$ is the chosen cut of Ω.

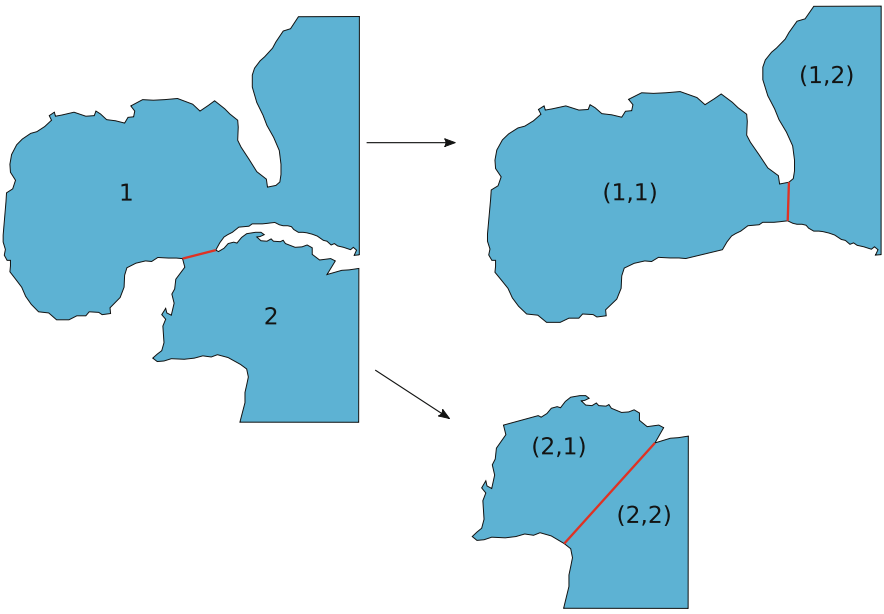
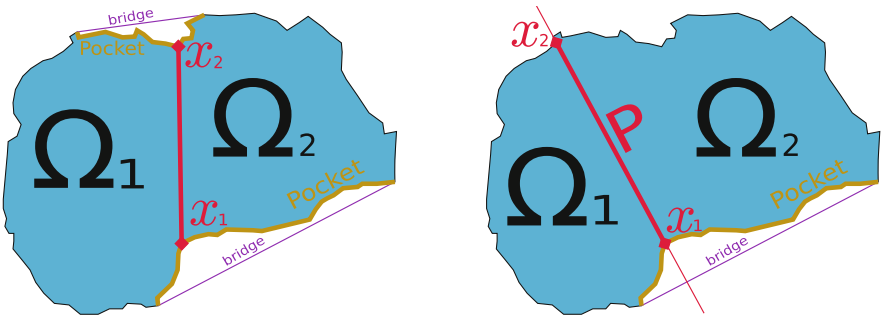


Fig. 2 The decomposition process of the region `Gulf`. The cuts are colored in each step and the subregions are labeled by 1–2 tuples to indicate a binary tree structure



(a) (b)

Fig. 3 Choices for a cut $\overline{x_1 x_2}$ in our region decomposition method. (**a**) Pocket points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with large concavity in different pockets. (**b**) A pocket point $\boldsymbol{x}_1$ with large concavity and the intersection $\boldsymbol{x}_2$ of $\partial\Omega$ and the line P

Our region decomposition method is shown in Algorithm 1. We interactively choose suitable cuts in each step. These cuts does not necessarily connect two pockets. On the other hand, the DUDE method joins points with large concavity measure and automatically selects the cuts using a triangulation.

Algorithm 1 Region decomposition method

```
procedure DECOMPOSITION(Ω)
    if Ω satisfies both concavity criteria (2)–(3) then
        return Ω
    else
        x̄1x̄2 ← Cut Choice Method(Ω)
        split Ω into Ω1 and Ω2 using the cut x̄1x̄2
        Decomposition(Ω1)
        Decomposition(Ω2)
    end if
end procedure
```

3 Parametrization Construction

Our next task is to find compatible parametrizations of admissible subregions. Any two parametrizations of different subregions are *compatible* if they have the same points on the intersection. First, we generate compatible parametrizations of the boundaries. Then, we extend these parametrizations into the interior of the subregions.

3.1 Compatible Parametrizations of the Boundaries

Let $\{\Omega_k\}_{k=1}^n$ be an admissible decomposition of Ω. We want to parametrize $\partial\Omega_k$ on the boundary of $R = [0, 1] \times [0, 1]$. To that end, we split $\partial\Omega_k$ into four consecutive polygonal curves $\Omega_{k,\text{bottom}}$, $\Omega_{k,\text{right}}$, $\Omega_{k,\text{top}}$ and $\Omega_{k,\text{left}}$ delimited by four points of $\partial\Omega_k$. The polygonal curves $\Omega_{k,\text{bottom}}$ and $\Omega_{k,\text{top}}$ are opposite boundaries of Ω_k. The same applies for $\Omega_{k,\text{right}}$ and $\Omega_{k,\text{left}}$.

Since the subregions Ω_k are admissible, we can manually choose four points of $\partial\Omega_k$ such that their interior angles are less than 180° and the opposite boundaries have approximately the same length. Zhang et al. [20] propose another choice for the four points.

We identify the decomposition cuts of Ω in the four boundary curves of each suregion, then we split each one of polygonal curves $\Omega_{k,\text{bottom}}$, $\Omega_{k,\text{right}}$, $\Omega_{k,\text{top}}$ and $\Omega_{k,\text{left}}$ into consecutive polygonal sections which are either cuts of Ω or maximal polygonal curves contained in $\partial\Omega$. We enumerate these polygonal sections starting with $\Omega_{k,\text{bottom}}$, then the sections in $\Omega_{k,\text{right}}$ and $\Omega_{k,\text{top}}$, and finally those in $\Omega_{k,\text{left}}$ for $k = 1, \ldots, n$. These sections form a polygonal decomposition $\{Q_p\}_{p=1}^s$ of $\cup_{k=1}^n \partial\Omega_k$ (Fig. 4).

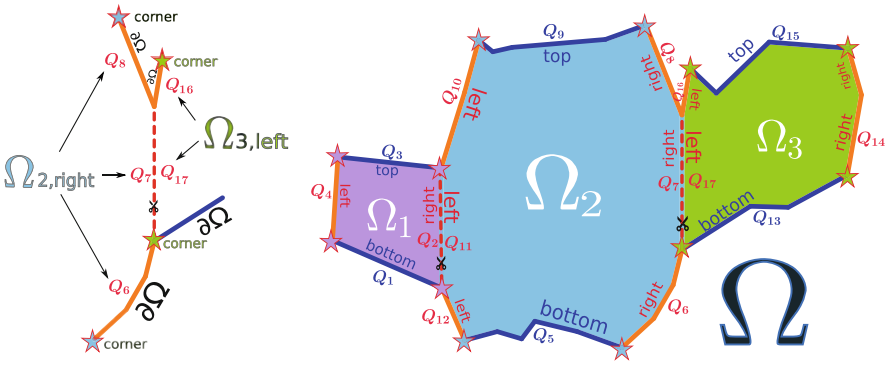


Fig. 4 The region Ω is split into Ω_1, Ω_2 and Ω_3. We split $\partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3$ into 17 polygonal sections Q_p. The sections Q_2, Q_7, Q_{11} and Q_{17} are cuts of Ω while the other sections are contained in $\partial\Omega$. The boundary curve $\Omega_{2,\text{right}}$ is split into three sections while $\Omega_{3,\text{left}}$ is split into two sections

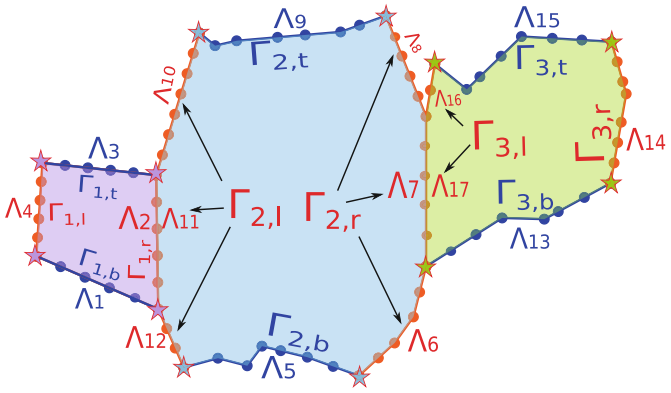


Fig. 5 The sections of the boundaries in Fig. 4 are reparametrized as polygonal curves Λ_p and the four boundary curves of each subregion are reparametrized as $\Gamma_{k,b}$, $\Gamma_{k,r}$, $\Gamma_{k,t}$ and $\Gamma_{k,l}$

Each polygonal section Q_p is reparametrized with respect to arc-length as a polygonal curve Λ_p with equidistant points. We join the sections Λ_p to obtain reparametrizations of $\Omega_{k,\text{bottom}}$, $\Omega_{k,\text{right}}$, $\Omega_{k,\text{top}}$ and $\Omega_{k,\text{left}}$ denoted by $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$, respectively (Fig. 5).

Afterwards, we generate four uniform linear B-spline curves:

$$\psi_{k,\text{bottom}}\colon [0,1] \to \Gamma_{k,\text{b}},$$
$$\psi_{k,\text{right}}\colon [0,1] \to \Gamma_{k,\text{r}},$$
$$\psi_{k,\text{top}}\colon [0,1] \to \Gamma_{k,\text{t}},$$
$$\psi_{k,\text{left}}\colon [0,1] \to \Gamma_{k,\text{l}}.$$

The control points of these B-spline curves are given by their polygonal curves. We combine these curves to get parametrizations $\psi_k \colon \partial R \to \partial \Omega_k$.

We want to extend ψ_k to the interior of Ω_k as explained by the UNAMALLA workgroup [1]. To that end, $\psi_{k,\text{bottom}}$ and $\psi_{k,\text{top}}$ must have the same number of points and the same condition on $\psi_{k,\text{right}}$ and $\psi_{k,\text{left}}$, that is,

$$\text{number of points of } \Gamma_{k,\text{b}} = \text{number of points of } \Gamma_{k,\text{t}}, \tag{4}$$

$$\text{number of points of } \Gamma_{k,\text{r}} = \text{number of points of } \Gamma_{k,\text{l}}. \tag{5}$$

Let m_p be the number of points of Λ_p. In order to formulate Eqs. (4) and (5) in terms of m_p we identify the sets of indexes of the polygonal sections Λ_p in each subregion by introducing some notation.

Let $s_{k,b}$, $s_{k,r}$, $s_{k,t}$ and $s_{k,l}$ be the number of polygonal sections Λ_p in $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$, respectively. Denote by s_k the number of polygonal sections Λ_p in $\Gamma_{k,\text{b}} \cup \Gamma_{k,\text{r}} \cup \Gamma_{k,\text{t}} \cup \Gamma_{k,\text{l}}$. Let

$$\sigma_k = \begin{cases} 0, & \text{if } k = 1; \\ \sum_{\ell=1}^{k-1} s_\ell, & \text{if } k > 1; \end{cases} \qquad k = 1, \ldots, n.$$

We define the set of indexes

$$J_{k,b} = \sigma_k + \{1, \ldots, s_{k,b}\},$$

where the sum means that σ_k is added to each element of the other set. Similarly, we define

$$\begin{aligned} J_{k,r} &= \sigma_k + \{s_{k,b} + 1, \ldots, s_{k,b} + s_{k,r}\}, \\ J_{k,t} &= \sigma_k + \{s_{k,b} + s_{k,r} + 1, \ldots, s_{k,b} + s_{k,r} + s_{k,t}\}, \\ J_{k,l} &= \sigma_k + \{s_{k,b} + s_{k,r} + s_{k,t} + 1, \ldots, s_k\}. \end{aligned}$$

Then, Eqs. (4)–(5) are formulated as

$$\sum_{j \in J_{k,b}} m_j - s_{k,b} = \sum_{j \in J_{k,t}} m_j - s_{k,t}, \tag{6}$$

$$\sum_{j \in J_{k,r}} m_j - s_{k,r} = \sum_{j \in J_{k,l}} m_j - s_{k,l}. \tag{7}$$

We want compatible parametrizations of the boundaries. So they must have the same points in the intersections. By construction, the intersections of the subregions are cuts of Ω. Since we have n subregions, there are $n - 1$ cuts. Let $\{c_i\}_{i=1}^{n-1}$ be the set of the decomposition cuts of Ω. For each cut c_i we have exactly two polygonal sections of $\cup_{k=1}^{n} \partial \Omega_k$ which coincide with c_i.

Let us remember that s is the number of polygonal sections Λ_p in $\cup_{k=1}^{n} \partial\Omega_k$. Let $\gamma_i, \delta_i \in \{1, \ldots, s\}$ be the indexes of two sections which coincide with c_i. Then $\psi_1, \ldots, \psi_n$ are compatible if

$$m_{\gamma_i} = m_{\delta_i}, \quad i = 1, \ldots, n-1. \tag{8}$$

Let $q = 3n - 1$. We put together Eqs. (6)–(8) as the system of linear equations

$$A\boldsymbol{m} = \boldsymbol{b}, \tag{9}$$

where $A \in \mathbb{Z}^{q \times s}$ with entries given by

$$a_{2k-1,j} = \begin{cases} 1, \text{ if } j \in J_{k,b}; \\ -1, \text{ if } j \in J_{k,t}; \\ 0, \text{ otherwise;} \end{cases} \qquad \begin{matrix} k = 1, \ldots, n, \\ j = 1, \ldots, s. \end{matrix}$$

$$a_{2k,j} = \begin{cases} 1, \text{ if } j \in J_{k,r}; \\ -1, \text{ if } j \in J_{k,l}; \\ 0, \text{ otherwise} \end{cases} \qquad \begin{matrix} k = 1, \ldots, n, \\ j = 1, \ldots, s. \end{matrix}$$

$$a_{2n+i,j} = \begin{cases} 1, \text{ if } j = \gamma_i; \\ -1, \text{ if } j = \delta_i; \\ 0, \text{ otherwise} \end{cases} \qquad \begin{matrix} i = 1, \ldots, n-1, \\ j = 1, \ldots, s, \end{matrix}$$

$\boldsymbol{b}$ is a vector in $\mathbb{Z}^q$ with entries given by

$$\begin{aligned} b_{2j-1} &= s_{j,b} - s_{j,t}, && j = 1, \ldots, n; \\ b_{2j} &= s_{j,r} - s_{j,l}, && j = 1, \ldots, n; \\ b_j &= 0, && j = 2n+1, \ldots, 3n-1, \end{aligned}$$

and

$$\boldsymbol{m} = \begin{bmatrix} m_1 & \cdots & m_s \end{bmatrix}^T.$$

The system of linear equations (9) is underdetermined since the matrix A has $3n - 1$ rows and for each subregion there are at least four boundary curves, i.e., there are at least $4n$ variables.

The vector $\boldsymbol{m}$ can be chosen as the solution of a linear integer programming problem:

$$\min \left\{ \mathbf{1}^T \mathbf{m} : \mathbf{m} \in \mathbb{Z}^s, A\mathbf{m} = \boldsymbol{b} \right\}, \tag{10}$$

where $\mathbf{1}$ is the vector of s ones. However, some entries of the optimal solution of the problem (10) could be negative or zero.

Let ℓ_p be the length of the polygonal section Λ_p. We choose compatible number of points so that their sum is minimized and the boundary point distribution depends on ℓ_p. Let $\ell_{\min}$ be the length of the smallest Λ_p. Given $K \in \mathbb{N}$, we measure the proportion of ℓ_p in comparison to $\ell_{\min}$ by

$$L_p = K \left\lceil \frac{\ell_p}{\ell_{\min}} \right\rceil. \tag{11}$$

Since the number of boundary points in Λ_p is m_p, we distribute at least L_p points in Λ_p by adding the constraint $m_p > L_p$ to the problem (10). So we solve the following integer linear programming problem:

$$\min \left\{ \mathbf{1}^T \mathbf{m} : \mathbf{m} \in \mathbb{Z}^s, A\mathbf{m} = \boldsymbol{b}, m_p \geq L_p \ p = 1, \ldots, s \right\}. \tag{12}$$

By construction, $\boldsymbol{b}$ is an integer vector and A is a totally unimodular matrix, that its, all its square submatrices have determinant 0, 1 or -1. Therefore, the integer programming problem (12) is feasible by the Hoffmann-Kruskal theorem [14].

We get compatible mesh sizes by solving the problem (12). Other authors use a tree structure of the region decomposition [20]. The parametrizations $\psi_{k,\text{bottom}}$, $\psi_{k,\text{right}}$, $\psi_{k,\text{top}}$ and $\psi_{k,\text{left}}$ with these mesh sizes are compatible.

3.2 Parametrizations of the Subregions

Now, we extend the parametrization ψ_k into the interior of Ω_k. Let M_k be the number of points of $\Gamma_{k,\text{b}}$, and let N_k be the number of points of $\Gamma_{k,\text{r}}$ We generate convex structured quadrilateral meshes

$$G_k = \left\{ P_{i,j}^{(k)} \in \overline{\Omega_k} : i = 1, \ldots, M_k, \ j = 1, \ldots, N_k \right\}$$

such that

$$\text{points of } \Gamma_{k,\text{b}} = \left\{ P_{i,1}^{(k)} : i = 1, \ldots, M_K \right\},$$

$$\text{points of } \Gamma_{k,\text{r}} = \left\{ P_{M_k,j}^{(k)} : j = 1, \ldots, N_K \right\},$$

$$\text{points of } \Gamma_{k,\text{t}} = \left\{ P_{i,N_k}^{(k)} : i = 1, \ldots, M_K \right\},$$

$$\text{points of } \Gamma_{k,\text{l}} = \left\{ P_{1,j}^{(k)} : j = 1, \ldots, N_K \right\}.$$

The meshes G_k are automatically generated using the discrete variational approach of our UNAMALLA workgroup [3, 4, 16]. Garanzha [8] and Ivanenko [11] generate quadrilateral meshes with boundary adaptability.

By construction, the boundary points of G_k are the control points of the linear B-spline curves $\psi_{k,\text{bottom}}$, $\psi_{k,\text{right}}$, $\psi_{k,\text{top}}$ and $\psi_{k,\text{left}}$. Since the polygonal curves $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$ satisfy Eqs. (4) and (5), the meshes $G_1, \ldots, G_n$ are compatible and their union is a block structured mesh G on Ω.

We use G_k to extend the boundary parametrization into the interior of Ω_k following the approach of the UNAMALLA workgroup [1]. Let B^2_{i,M_k} be the i-th linear B-spline with knot sequence given by a uniform partition of $[0, 1]$ with M_k elements for $i = 1, \ldots, M_k$, and let B^2_{j,N_k} be the j-th linear B-spline with knot sequence given by a uniform partition of $[0, 1]$ with N_k elements for $j = 1, \ldots, N_k$. We use the parametrizations $\varphi_k : R \to \Omega_k$ given by the bilinear tensor product B-spline

$$\varphi_k(\xi, \eta) = \sum_{i=1}^{M_k} \sum_{j=1}^{N_k} P^{(k)}_{i,j} B^2_{i,M_k}(\xi) B^2_{j,N_k}(\eta), \qquad \xi, \eta \in [0, 1]. \tag{13}$$

Let us make a few observations of these parametrizations:

- The control points of φ_k are the points of G_k.
- The map φ_k is 1-1 since all the cells of G_k are convex [1].
- By construction,

$$\varphi_k|_{\partial\Omega_k} \equiv \psi_k. \tag{14}$$

Moreover, since $\psi_1, \ldots, \psi_n$ are compatible, then $\varphi_1, \ldots, \varphi_n$ are compatible.

Therefore, we have decomposed the region Ω into a set of admissible regions $\Omega_1, \ldots, \Omega_n$, and we have constructed a family of compatible and admissible parametrizations $\varphi_1, \ldots, \varphi_n$ for these subregions given by (13).

4 Summary and Examples

We summarize the key points of our methodology:

1. Get an admissible decomposition $\{\Omega_k\}_{k=1}^n$ of Ω by Algorithm 1.
2. In each Ω_k choose four points as explained in Sect. 3.1.
3. Identify the cuts of Ω in each $\partial\Omega_k$ and split $\cup_{k=1}^n \partial\Omega_k$ into polygonal sections. Reparametrize these sections and join them to obtain reparametrizations $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$ of the four boundary curves.
4. Solve the integer linear programming problem (12) to get compatible number of points for $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$.
5. Generate convex structured quadrilateral meshes G_k on Ω_k such that their boundary points are the points of $\Gamma_{k,\text{b}}$, $\Gamma_{k,\text{r}}$, $\Gamma_{k,\text{t}}$ and $\Gamma_{k,\text{l}}$.
6. Construct parametrizations φ_k using the bilinear tensor product B-spline (13).

We generate parametrizations of four irregular polygonal regions using our method. First, the region decomposition is interactively carried out using our subroutines in JULIA [5] with concavity criteria tolerances $\tau_1 = 0.1$ and $\tau_2 = 0.45$. We do not use the DUDE code. Then, a Julia interface of the COIN-OR Branch and Cut solver [7] is used to solve the integer programming problem (12). We choose $K = 2$ for the lower bound L_p given by (11). Finally, automatic mesh generation is carried out by our UNAMALLA software [16] using a convex combination of the weighted discrete functionals H_ω and Area-Orthogonality so that mesh cells are accumulated in the boundary of the subregions [4, 9].

The region decomposition of the four polygonal regions and their corresponding block structured meshes are shown in Figs. 6, 7, 8, and 9. Table 1 shows the number of points, subregions and polygonal sections Λ_p of each region.

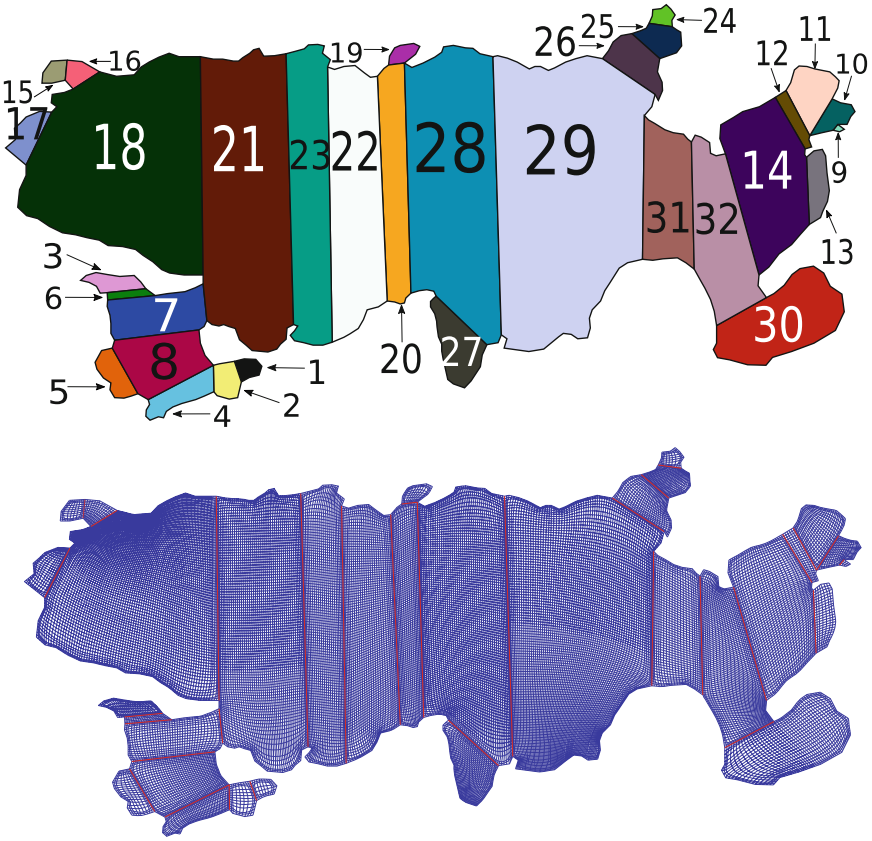


Fig. 6 Admissible decomposition of `Titicaca` and its block structured mesh

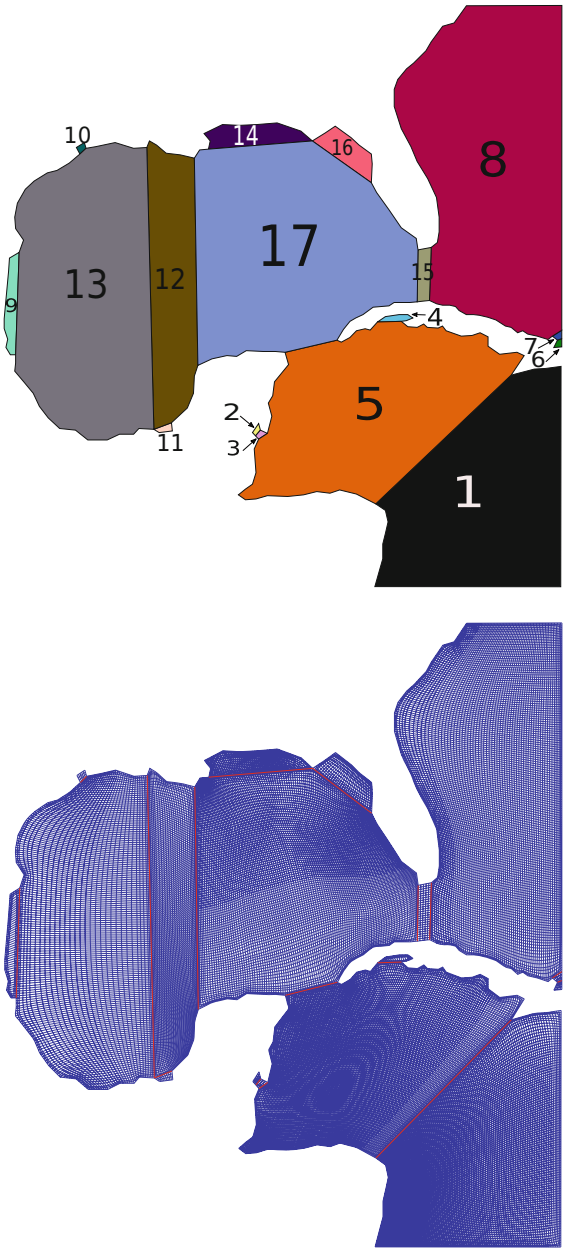


Fig. 7 Admissible decomposition of the region `Gulf` and its block structured mesh

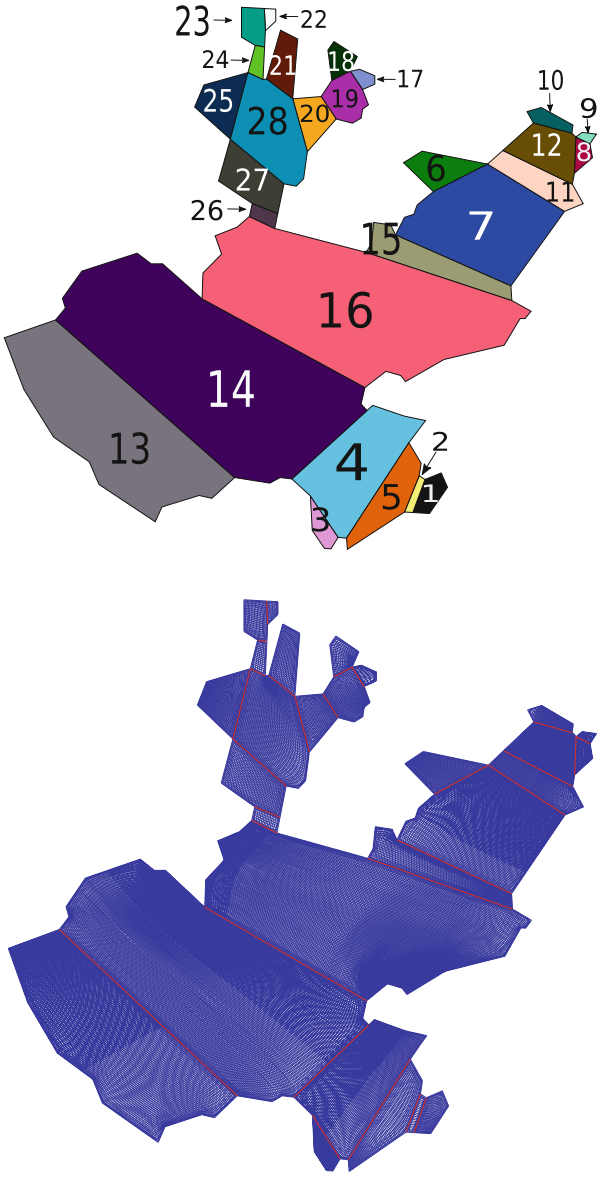


Fig. 8 Admissible decomposition of the region `Jalisco` and its block structured mesh

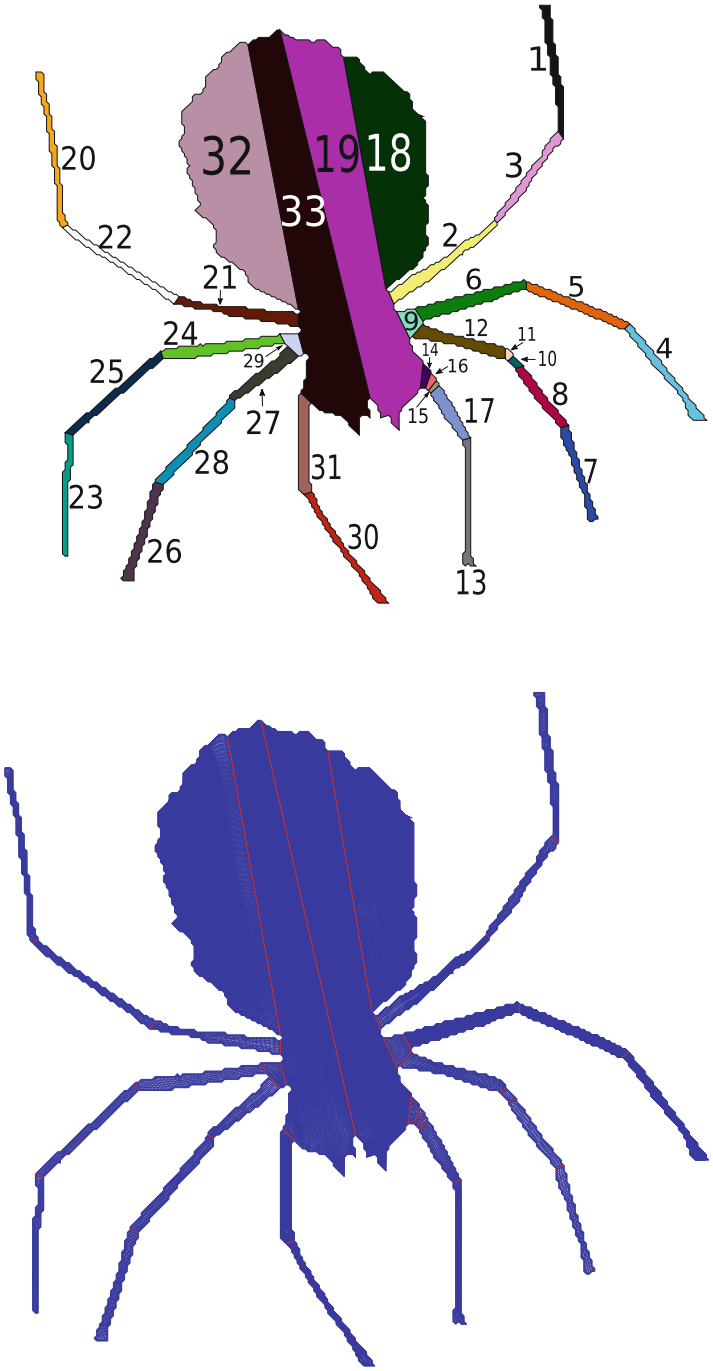


Fig. 9 Admissible decomposition of the region `Spider` [13] and its block structured mesh

Table 1 Number of points, subregions and polygonal sections in our example regions

Region	Number of points	Number of subregions	Number of polygonal sections Λ_p
`Gulf` Gulf of Mexico	199	17	88
`Titicaca` Lake Titicaca in Peru	365	32	161
`Jalisco` State of Jalisco in Mexico	120	28	139
`Spider` [13]	1388	33	150

5 Conclusions and Future Work

We have proposed a methodology to find a decomposition of an irregular polygonal region into admissible subregions and a family of compatible bilinear B-spline parametrizations of these subregions.

Irregular regions are interactively decomposed into admissible subregions using concavity measures. Then, the subregion boundaries are parametrized by compatible linear B-spline curves. Afterwards, these parametrizations are extended into the interior of the subregions as compatible bilinear B-splines by automatic structured mesh generation.

We plan to measure the quality of our meshes using the quality measures reported by UNAMALLA [9]. Our parametrizations are compatible, but they are not necessarily smooth between the subregions. We address this issue later.

We want to extend our method to multiply-connected plane regions. Our results would be submitted in the part II of this paper. We thank the anonymous referees that help us to make significant changes to improve our paper.

References

1. Abello, I.A., Hernández, V., Barrera, P., González, G.F.: Parametrización B-spline de regiones planas con frontera irregular. Ci. Mat. **31**:2, 95–107. Cuba (2017)
2. Abello, I.A., Hernández, V., Barrera, P., González, G. F.: Injectivity of B-spline biquadratic maps. Comput. Methods Appl. Mech. Eng. **341**, 586–608. Elsevier (2018)
3. Barrera, P., González, G.F., Domínguez, F.J.: Robust discrete grid generation on plane irregular regions. USSR Comput. Math. Math. Phys. **43**(6), 884–892 (2003)
4. Barrera, P., Cortés, J.J., González, G.F., Domínguez, F.J., Tinoco, J.G.: Smoothness and convex area functionals revisited. SIAM J. Sci. Comput. **32**(4), 1913–1928 (2010)
5. Bezanson, J., Edelman, A., Karpinski, S., Shah V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
6. Bommes, D., Campen, M., Ebke, H.C., Alliez, P., Kobbelt, L.: Integer-grid maps for reliable quad meshing. ACM Trans. Graph. **32**(4) (2013). https://doi.org/10.1145/2461912.2462014
7. Forrest, J., Ralph, T., Vigerske, S.: coin-or/Cbc: Version 2.9.9. https://projects.coin-or.org/Cbc. Cited 19 July 2018

8. Garanzha, V.A.: Barrier method for quasi-isometric grid generation. Comput. Math. Math. Phys. **40**(11), 1617–1637 (2000)
9. González G. F.: Generación Numérica de Mallas Estructuradas de Calidad y Adaptativas en Regiones Planas Irregulares. PhD. Thesis. Universidad Nacional Autónoma de México. México (2018)
10. Gravesen, J., Evgrafov, A., Nguyen, M., Nørtoft, P.: Planar parametrization in isogeometric analysis. In: Floater, M., et al. (eds) Mathematical Methods for Curves and Surfaces. MMCS 2012. Lecture Notes in Computer Science, vol. 8177. Springer, New York (2014). https://doi.org/10.1007/978-3-642-54382-1_11
11. Ivanenko, S.A.: Control of cells shapes in the course of grid generation. Comput. Math. Math. Phys. **40**(11), 1596–1616 (2000)
12. Lien, J.: Approximate Convex Decomposition and its Applications. PhD. Thesis. Texas A&M University, USA (2006)
13. Liu, G., Xi, Z., Lien, J.: Dual-Space Decomposition of 2D Complex Shapes. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4154–4161 (2014)
14. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1998)
15. Sheffer. A., Praun, E., Rose, K.: Mesh Parameterization Methods and Their Applications. Found. Trends Comput. Graphics Vision **2**(2), 105–171 (2006)
16. UNAMalla Workgroup: UNAMalla: Version 5.0. http://www.mathmoo.unam.mx/unamalla. Cited 23 May 2013
17. Xiao, Z., He, S., Xu, G., Chen, J., Wu, Q.: A boundary element-based automatic domain partitioning approach for semi-structured quad mesh generation. Engrg. Anal. Boundary Elem. **11**, 133–144. Elsevier (2020)
18. Xu, G., Mourrain, B., Duvigneau, R., Galligo, A.: Parametrization of computational domain in isogeometric analysis: methods and comparison. Comput. Methods Appl. Mech. Eng. **200**, 23–24, 2021–2031. Elsevier (2011). https://doi.org/10.1016/j.cma.2011.03.005
19. Xu, G., Li, M., Mourrain, B., Rabczuk, T., Xu, J., Bordas, S.P.A.: Constructing IGA-suitable planar parametrization from complex CAD boundary by domain partition and global/local optimization. Comput. Methods Appl. Mech. Eng. **328**, 175–200 (2018)
20. Zhang, Y., Jia, Y.: 2D automatic body-fitted structured mesh generation using advancing extraction method. J. Comput. Phys. **353**, 316–335. Elsevier (2018). https://doi.org/10.1016/j.jcp.2017.10.018
21. Zint, D., Grosso, R., Aizinger, V., Köstler, H.: Generation of block structured grids on complex domains for high performance simulation. In: Garanzha, V.A., et al. (eds.) Numerical Geometry, Grid Generation and Scientific Computing. Lecture Notes in Computational Science and Engineering, vol. 131, pp. 87–99. Springer, New York (2019). https://doi.org/10.1007/978-3-030-23436-2_6

A Hybrid Approach to Fast Indirect Quadrilateral Mesh Generation

Daniel Zint and Roberto Grosso

Abstract Indirect quadrilateral mesh generation methods are commonly used particularly for numerical simulations due to their adaptiveness to different element sizes across the domain. The well-known Blossom-Quad algorithm generates high quality meshes but has a worst case complexity of $O\left(N^2 + N \log N\right)$. A method which merges triangles using topological operations is less complex, indeed we show that it has a complexity of $O(N \log N)$, but resulting meshes might have low quality especially near boundaries. We propose a combination of these two methods. Boundary regions are processed by Blossom-Quad and the interior by triangle merging. Post-processing solves quality issues caused by triangle merging. The results are comparable to pure Blossom-Quad but this approach is much faster. Therefore, it is favorable especially on large meshes where the runtime of Blossom-Quad might become troublesome. The efficiency of this hybrid approach is shown on ocean meshes with up to several million triangles.

1 Introduction

Quadrilaterals are often favored over triangles in numerical simulations for various reasons. Besides requiring less memory, quadrilateral (quad) meshes are known to have several positive side effects on simulations [2]. A well-known quad meshing algorithm is Blossom-Quad [28]. It belongs to the group of indirect quad meshing methods, i.e., it requires a triangle mesh as input. Triangles are merged by solving the best matching problem of graph theory with Edmonds' Blossom algorithm [14–16]. Remaining triangles only exist on boundaries and appear pairwise. They can be eliminated by duplicating vertices, swapping, or collapsing edges. A major drawback of Blossom-Quad is its worst case complexity of $O\left(N^2 + N \log N\right)$, where N is the number of triangles [19]. Even though modern implementations like

D. Zint (✉) · R. Grosso
Visual Computing, Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany
e-mail: daniel.zint@fau.de; roberto.grosso@fau.de

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_18

the one of Kolmogorov [24] terminate early in many cases, runtime is unpredictable. It might happen that a mesh with several million triangles is processed within less than a minute while another mesh with only a fraction of elements takes several minutes or longer.

Another indirect way of generating quad meshes is merging as many triangles as possible in a greedy fashion and moving remaining triangles across the mesh to eventually merge them. This approach is very simple and appears in different forms in literature [6, 33]. We show that triangle merging is of complexity $O(N \log N)$. Post-processing solves quality issues on the interior. On boundaries, especially those with complex shapes, achieving good quality with post-processing is difficult.

We present a hybrid approach which uses Blossom-Quad in boundary regions and triangle merging on the interior, Fig. 1. Blossom-Quad ensures high quality on boundaries whereas merging the majority of triangles in a more efficient way improves runtime. This approach outperforms Blossom-Quad on all our test meshes while yielding similar quality. A positive side effect of our method is that we adjust our post-processing such that the number of quads is half the number of triangles. This control over complexity is useful for high performance simulations. For example, block-structured grids with a precise number of blocks can be generated this way.

In Sect. 2 we give a brief introduction to quad mesh generation. In Sect. 3 we explain our implementation of triangle merging and show how we combine it with Blossom-Quad. The post-processing, consisting of smoothing and topological improvements, is explained in Sect. 4. A comparison of Blossom-Quad, triangle merging, and the hybrid approach are given in Sect. 5. Finally, conclusions are drawn in Sect. 6. A C++/CUDA implementation of our method is provided at https://github.com/DanielZint/RatRace .

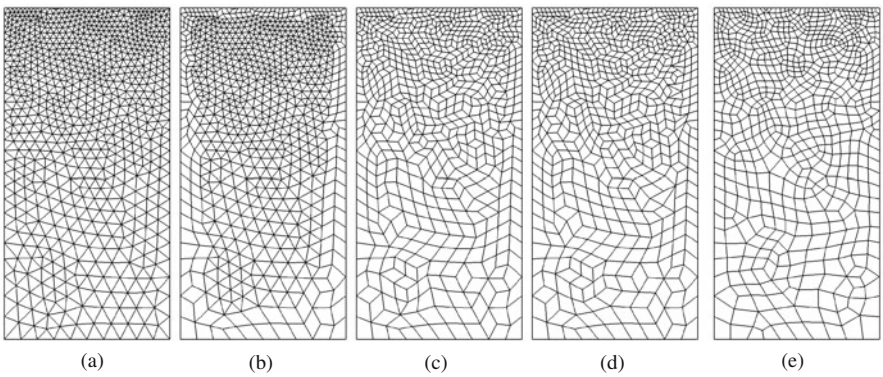


Fig. 1 Hybrid quad mesh generation which performs Blossom-Quad near boundaries and triangle merging on the interior, followed by smoothing and topological optimization. (**a**) Triangle mesh. (**b**) Blossom-Quad near boundaries. (**c**) Greedy triangle merging. (**d**) Merge remaining triangles. (**e**) Post-processing

2 Related Work

Several papers compare the vast range of quad meshing methods. Bommes et al. discuss in [7] methods focusing on applications in computer graphics. Armstrong et al. in [2] and Campen in [8] discuss multi-block structured quad mesh generation. A survey about unstructured mesh generation is given in [25]. Usually, quad meshing algorithms are divided into two categories, direct and indirect methods, depending on weather they construct quads from a triangle mesh or directly.

Advancing front algorithms, mostly considered as indirect methods, work well on boundaries but might result in poorly shaped elements in the center of a domain [4, 26, 34]. Other methods use domain decompositions given by the medial axis [32], a quadtree [3, 30], the Morse-Smale complex [13], or cross fields [5, 22]. They generate mostly structured meshes with overall good quality. The challenging task here is generating a reasonable domain decomposition which can become tedious for meshes with complex boundaries and strongly varying element sizes.

Given a triangle mesh, the easiest way to obtain a quad mesh is applying one Catmull-Clark subdivision step [10]. However, besides increasing the number of elements by a factor of three, the resulting quad mesh also contains many irregular vertices. Rank et al. first generate a hybrid mesh by merging neighboring triangles and perform one subdivision step afterwards [27]. By applying topological changes Docampo-Sánchez and Haimes generate quad meshes with good quality directly from a subdivided triangle mesh [12]. However, they have no control over the element size. Tarini et al. remove triangles of a hybrid mesh by moving them through the mesh and merging them [33]. The resulting mesh is improved by post-processing but especially on boundaries quality might not be sufficient.

Remacle et al. formulate triangle merging as the best matching problem of graph theory and solve it with the Blossom-Quad algorithm [28]. Several methods were developed to improve the triangle mesh before applying Blossom-Quad [17, 20, 29]. Often it is used to generate hybrid meshes which are then subdivided using Catmull-Clark. This is for example the default behavior in Gmsh [21]. The method generates overall good quality but has performance issues because of its complexity of $O\left(N^2 + N \log N\right)$.

3 Hybrid Quad Mesh Generation

Our hybrid approach processes boundary regions with Blossom-Quad. Those are determined by a segmentation step presented in Sect. 3.1. Triangles on the interior are greedily merged by a method which is based on the one of Rank et al. [27], Sect. 3.2. In Sect. 3.3 we explain our implementation of triangle merging which uses the dual graph for finding paths between two triangles. Throughout this work we explicitly distinguish between *nodes* and *vertices*. Nodes belong to the dual graph, vertices are part of the mesh.

3.1 Segmentation of Boundary Regions

Each boundary vertex gets an index assigned which is unique for the boundary. Next, all neighboring vertices and also their neighboring vertices inherit the same index. A triangle is considered as part of the boundary region if all of its vertices have the same boundary index. Thus, two layers of triangles belong to each boundary, Fig. 2a. Sharp corners should be avoided in the segmentation. Therefore, vertices with more than half of their neighbors having a boundary index are also added to the boundary region. Several boundaries that are close to each other might have intersecting regions. We solve this by merging those boundary regions into a single one, Fig. 2b. Furthermore, a boundary region might cut off some interior triangles from the rest, Fig. 2c. There must exist only one interior region. Otherwise, triangle merging might have to iterate through boundary regions and cause unsolvable quality problems. Only the largest of interior regions is not considered as island. We check for such islands of interior triangles and add them to the surrounding boundary region, Fig. 2d. As a last step we have to ensure that all boundary regions consist of an even number of triangles. If this condition is not satisfied, we just add a random adjacent triangle from the interior.

Blossom-Quad is applied independently to each boundary region. Remaining triangles are not processed as described by Remacle et al. in [28]. Instead, triangle merging and post-processing will take care of those.

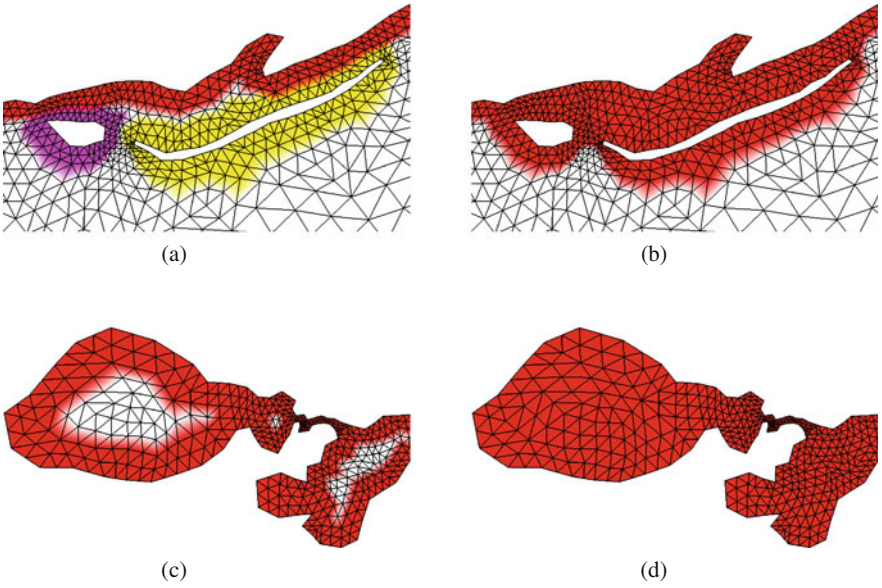


Fig. 2 Segmentation of boundary regions. (**a**) Several boundaries with unique indices. (**b**) Touching boundaries are merged and interior vertices in between are added. (**c**) Islands of interior vertices enclosed by a boundary. (**d**) Islands are added to the boundary

Table 1 Number of remaining triangles for the method of Rank et al. and our extended version

Mesh	# triangles	# remaining triangles	
		Rank et al.	Ours
Chile	6,517,764	852,488	142,466
Chile_low	3,396,752	445,780	78,916
Okushiri	3,098,880	142,228	13,510
Ike	2,678,402	229,788	17,296

3.2 Greedy Triangle Merging

A simple and fast way of generating a hybrid mesh from triangles is the method of Rank et al. [27]. For each edge of the triangle mesh, we compute the quality of the quad that would be generated by deleting this edge. Edges are deleted greedily starting with the best quality. With this method a rather high number of triangles remains. In our tests, approximately 10% could not be merged.

We reduce the number of remaining triangles by extending the method of Rank et al. We also start with the best edge, i.e., the edge that generates the best quadrilateral, and remove it but instead of removing the second best edge, we continue triangle merging in the neighborhood of the best edge. All edges that would generate a quad incident to the one we just created are stored in a queue, starting with the best edge. Edges with low quality are omitted. We then merge the edges from this queue, recursively adding edges in the neighborhood like in the previous step. If the queue is empty, we add the best remaining edge to the queue and continue until no more merging is possible.

This extended version of the method of Rank et al. has much less remaining triangles while also creating quads with good quality. In our tests we had less than 3% of triangles remaining for large meshes. A detailed comparison is given in Table 1.

3.3 Merging of Remaining Triangles

Triangle merging is performed on the dual graph and is based on concepts introduced by Tarini et al. [33]. The path between two triangles is determined with breadth first search (BFS). Two triangles can be merged if there exists a connecting path purely consisting of quads. Thus, we can always find a quad topology for any manifold mesh with an even number of triangles as long as the mesh does not contain multiple independent patches. Having a triangle node n_i on the dual graph we use BFS to find the nearest triangle node n_j. Backtracking the path from n_j to n_i we modify the dual graph such that n_j "moves" towards n_i. This graph modification is done in two steps, *merge* and *split*.

Our data structure must be capable of dealing with multiple edges, which appear when two quads share two edges. We use a modification of the halfedge data

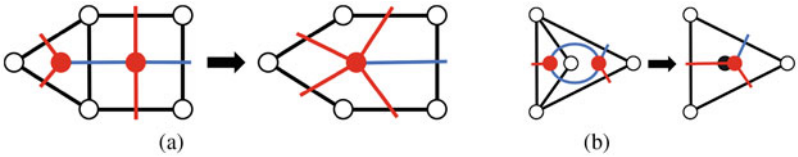


Fig. 3 All possible triangle/quad merges. Blue edges represent the merging path. (**a**) Single edge. (**b**) Multiple edge

structure [23]. Instead of a face, halfedges store their neighboring vertex. Thus, we can always reconstruct the hybrid mesh from the dual graph. Each element of the mesh is represented by a node. When merging two nodes that are connected by a multiple edge we need to store the vertex that is enclosed by the multiple edge. Therefore, each node contains a vector of vertices.

3.3.1 Merging Nodes

Merging two nodes is an edge collapse where multiple edges have to be considered. By merging a triangle with a quad we get a pentagon, Fig. 3a, but by merging along a multiple edge we get a triangle with an unreferenced vertex, Fig. 3b. Multiple edges appear when two graph edges connect the same two nodes, i.e., there exists a vertex with two incident edges in the mesh. The unreferenced vertex is stored in the remaining node.

3.3.2 Splitting Nodes

In the previous step we have merged a triangle with a quad. This produces either a node with five outgoing halfedges, Fig. 4a, or a node with three outgoing halfedges and one vertex stored, Fig. 4b. There is only a limited set of possible splits if the outcome should be again a triangle and a quad. There are at most four possible split cases when we consider the moving-direction (which might be also a multiple edge). We choose the case which produces the best quad according to a quality measure, e.g., the shape metric [31]. Note that this decision won't stop the mesh from getting tangled. It merely helps to keep mesh quality somewhat reasonable. Tangling will be handled by post-processing.

After moving a triangle towards another, the last step is just combining the two triangles to one quad, i.e., merging two nodes in the dual graph. After merging all triangles the result is a valid quad topology. However, the chances for having a tangled result are high. There might also exist valence two vertices in the mesh which is usually not acceptable. Later we show how to recover mesh quality with post-processing.

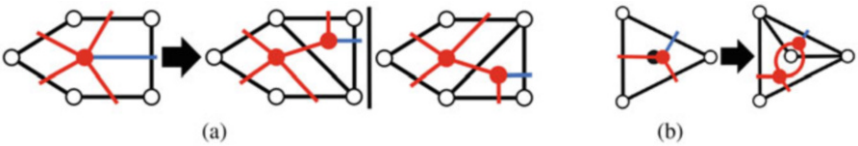


Fig. 4 All possible node splits when the triangle is moved towards the blue edge. (**a**) Single edge. (**b**) Multiple edge

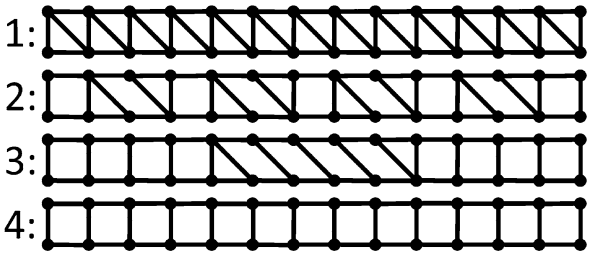


Fig. 5 Merging on a triangle strip with the maximum amount of operations possible

3.3.3 Complexity of Triangle Merging

The BFS algorithm on triangle meshes has a worst case complexity of $O(N)$, where N is the number of triangles. If BFS has to be applied to all triangles the worst case complexity is $O(N^2)$. We show that the complexity is actually lower.

Assume a pure triangle mesh on which triangle merging is performed in the most inefficient way, i.e., with as many operations as possible. In the first step, triangles will be merged with their direct neighbors. In the worst case, every third triangle would be left over. We illustrate this for a triangle strip in Fig. 5. Thus, in the first step we need to perform $N/3$ operations, one search operation for every third triangle. When considering more complex examples than a triangle strip, the number of operations might be actually higher but search is still local and therefore of $O(1)$ for each triangle. In the second step, BFS needs 3 search operations (distance on the initial triangle mesh) to find the next triangle. Again, this is only done for every third triangle, $N/3^2 \cdot 3$. In general, BFS requires 3^{i-1} operations on 3^i triangles in the i-th step, $N/3^1 \cdot 3^{i-1} = N/3$. The number of remaining triangles is reduced by a factor of 3 in every iteration, leading to the following number of operations:

$$\sum_{i=1}^{3^i \leq N} \frac{1}{3} N = \sum_{i=1}^{i \leq \log_3 N} \frac{1}{3} N \leq \frac{1}{3} N \log_3 N. \tag{1}$$

This estimation only considers the complexity of BFS but all other steps like moving the triangle are only local operations with O(1) and therefore not relevant. Thus, the complexity of triangle merging is $O(N \log N)$.

4 Post-Processing

Post-processing consists of smoothing and a set of topological operations. For smoothing we use a combination of Laplacian and optimization-based smoothing. With the topological operations the number of irregular vertices is reduced. An interior vertex is considered as irregular if it has more or less than four incident edges, i.e., its valence is unequal four. Topology is only changed locally to preserve element size.

Topological Improvements

We use three topological operations: edge swap, diagonal collapse, and vertex split. Edge swapping is a key feature for topology optimization not just on triangle but also on quad meshes [9, 12, 33, 34]. There exist three ways to orient an edge between two quads, Fig. 6a. Thus, there are two possible configurations for an edge swap. An edge is swapped whenever this reduces the number of irregular vertices.

Vertex split is the inverse operation of diagonal collapse, Fig. 6b. A vertex split reduces the valence of the vertex and increases the valence of two of its neighbors. Again, both operations are applied if they reduce the number of irregular vertices.

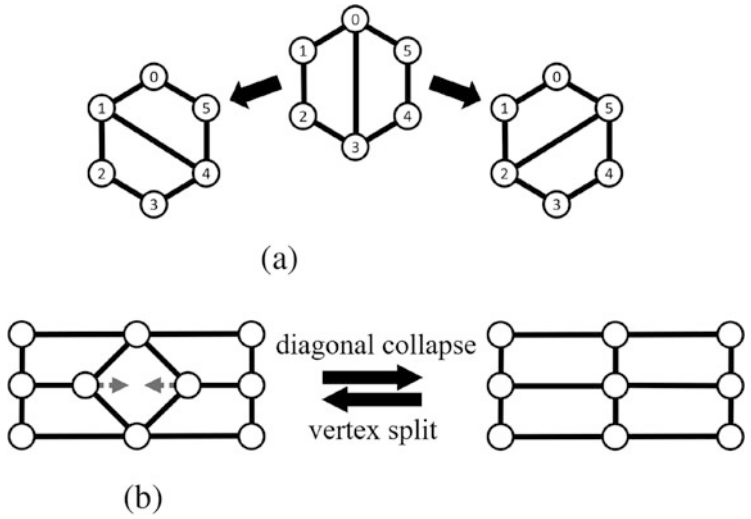


Fig. 6 Topological operations on quad meshes. (**a**) Edge swap. (**b**) Diagonal collapse and vertex split

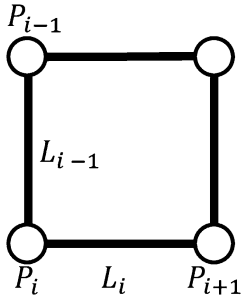
The first step of post-processing is eliminating valence two vertices with diagonal collapses. Also valence two vertices on boundaries are eliminated if they have low shape quality (3). For this we set a threshold for the quality to 0.1. Next, we perform edge swaps, diagonal collapses, and vertex splits.

Diagonal collapses are only performed if the resulting elements do not become too long. In practice, if the diagonal of the collapsing quad is longer than 1.5 times the size of the initial edge length in this position, the quad is not collapsed. This condition is evaluated by using a size function.

Vertex splits are only performed if elements do not become too short. We only perform vertex splits if the edge that is shortened due to the split is at maximum 0.9 times the size of the initial edge length in this position. Additionally, vertex splits are only performed if the number of elements is smaller or equal than the expected number of elements, i.e., half the number of initial triangles. In our experience there are usually more vertex splits possible than diagonal collapses. Therefore, we do not consider the number of elements in the diagonal collapse. If the number of quads is smaller than expected it can be easily increased by performing vertex splits. This way we can precisely control the number of elements.

Smoothing

For smoothing we use the method *discrete mesh optimization* (DMO) [35, 36]. DMO evaluates a quality metric on a uniform candidate grid and chooses the optimal position solving an *argmaxmin*-problem. DMO does not require derivatives and therefore the metric can be chosen arbitrarily. We measure element quality with the shape metric of Stimpson et al. [31],



$$c = \min_{i \in [1,4]} c_i \tag{2}$$

$$c_i = 2\frac{\alpha_i}{\|L_i\|^2 + \|L_{i-1}\|^2} \tag{3}$$

$$\alpha_i = (L_i \times L_{i-1})^\intercal \cdot n \tag{4}$$

$$L_i = P_{i+1} - P_i \ , \tag{5}$$

where c_i is the inverse condition of a vertex v_i within this quad, P_i is the position, and $n = (0, 0, 1)^\top$. For optimization and especially untangling we need to modify the metric as it has a negative minimum at -1. A common approach for untangling is using the signed area [18]. We combine shape quality and signed area,

$$c_i' = \begin{cases} 2\frac{\alpha_i}{\|L_i\|^2 + \|L_{i-1}\|^2}, & \text{if } \alpha_i > 0, \\ \alpha_i, & \text{otherwise.} \end{cases} \tag{6}$$

For vertex repositioning we must not consider the quality of the opposite quad vertex,

$$c'(v_i) = \min(c'_i, c'_{i+1}, c'_{i-1}). \tag{7}$$

A similar approach which builds upon the Winslow method was presented by Charakhch'yan and Ivanenko in [11]. The optimization is here formulated as a global minimization problem and the objective function contains the inverse shape quality metric.

Considering only minimal element quality often generates jagged lines within a mesh. We avoid this by only requiring a shape quality of at least 0.5. If the quality is higher, we use standard Laplace smoothing.

5 Results

We compare the hybrid approach to Blossom-Quad and pure triangle merging on several meshes designed for ocean simulation. The meshes *chile*, *chile_low*, and *okushiri* were generated by Sven Harig at the Alfred Wegener Institute for tsunami simulations using TsunAWI [1]. All meshes except for *diam10/diam11* have varying element sizes. Besides *okushiri* and *diam10/diam11*, they have multiple and complex boundaries. None of the meshes were pre-conditioned for quad meshing. In Fig. 7 we show the result of the hybrid approach on mesh *chile_low*. Quantitative results are presented in Table 2. Runtimes exclude post-processing as this is applied to all methods equally. Quality is measured in terms of element shape as defined in [31]. Irregular vertices are only counted on the interior and displayed relatively to all interior vertices.

For Blossom-Quad there is no direct link between runtime and mesh size. The complexity of $O\big(N^2 + N \log N\big)$ is merely an upper bound and often not reached. We use Blossom V, an implementation of Kolmogorov [24] which is up to an order of magnitude faster than prior implementations of Edmonds' Algorithm. For pure triangle merging and the hybrid approach runtime scales linearly with mesh size. The only exceptions are *okushiri* and *diam10/diam11* which have simple boundaries and are mostly structured on the interior. In that case, triangle merging is very efficient. The hybrid approach outperforms Blossom-Quad on all test meshes that have more than 20,000 triangles. On smaller meshes the segmentation has a major effect on performance. Pure triangle merging performs best independently of the mesh size.

Mesh quality of Blossom-Quad and the hybrid approach is comparable. Both have the same minimal quality which is reasonable considering that low quality elements appear on boundaries. If better minimal quality is required, the threshold

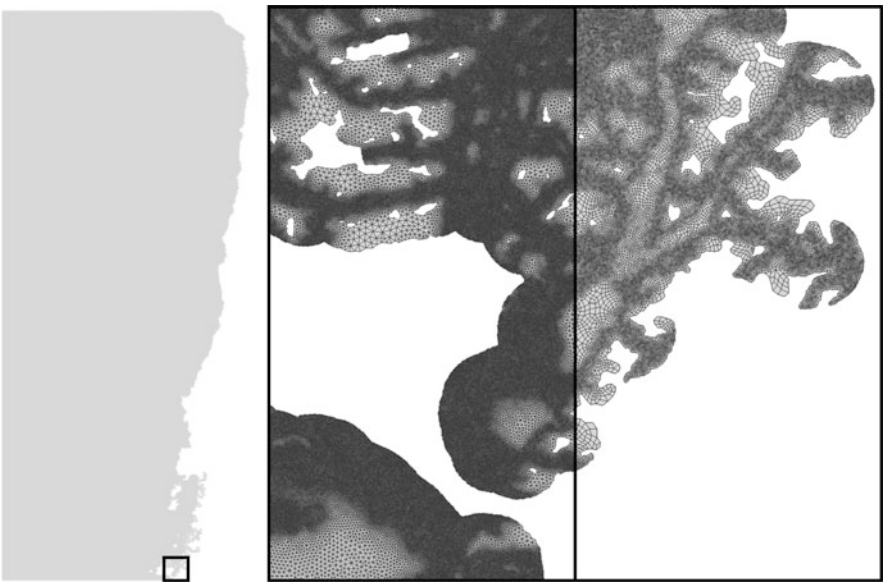


Fig. 7 The mesh *chile_low* in the initial form with triangles and as quad mesh generated with the hybrid approach. The initial mesh contains 3,396,752 triangles

for valence two vertex removal on boundaries can be increased. The average quality is a little bit better for Blossom-Quad almost everywhere. Furthermore, Blossom-Quad produces the lowest amount of irregular vertices but the difference to the hybrid approach is only up to 2%. Triangle merging reaches good quality in many cases when post-processing is applied but on some complex boundaries post-processing cannot solve all quality related issues, i.e. degenerated quads remain. Therefore, meshes generated purely with triangle merging might not be appropriated for numerical simulations.

The meshes *diam10/diam11* are artificial tests and show the major difference between Blossom-Quad and triangle merging. The meshes consist of two equilateral triangles which were 10/11 times uniformly subdivided. Blossom-Quad reaches optimal quality by generating a fully structured grid whereas triangle merging and the hybrid approach generate irregularities. The optimality of Blossom-Quad comes with the price of a very slow performance. For *diam11* Blossom-Quad took approximately 30 min, the hybrid approach only 36 s. The reason for the bad performance of Blossom-Quad is that all triangles have the same quality and therefore finding the global optimum is challenging.

Table 2 Quality and performance of hybrid approach (HA) in comparison to Blossom-Quad (BQ) and pure triangle merging (TM)

		Runtime [s]			Min. quality			Avg. quality			Irreg. vert. [%]		
Mesh	# triangles	BQ	TM	HA	BQ	TM	HA	BQ	TM	HA	BQ	TM	HA
Chile	6,517,764	102	74	78	0.11	0.10	0.11	0.89	0.87	0.87	16.6	18.2	18.2
Chile_low	3,396,752	203	21	27	0.14	0.10	0.14	0.89	0.87	0.87	16.9	18.8	18.7
Okushiri	3,098,880	155	11	13	0.62	0.53	0.54	0.92	0.89	0.89	3.9	2.4	2.5
Ike	2,678,402	70	10	22	0.10	< 0.01	0.10	0.90	0.89	0.89	9.7	11.7	11.7
Gom	57,076	0.2	0.2	0.2	0.11	0.11	0.11	0.88	0.85	0.86	10.5	14.1	12.7
East	18,578	< 0.1	< 0.1	< 0.1	0.42	0.17	0.42	0.86	0.84	0.85	19.1	21.4	21.4
East_low	2000	< 0.1	< 0.1	< 0.1	0.20	0.21	0.20	0.69	0.67	0.68	30.3	31.7	30.8
Diam11	8,388,608	1766	29	36	0.87	< 0.01	0.63	0.87	0.88	0.88	0	5.2	7.0
Diam10	2,097,152	210	7	8	0.87	0.43	0.63	0.87	0.89	0.89	0	1.2	1.4

6 Conclusion

We presented a hybrid method combining Blossom-Quad and triangle merging. Boundary regions are processed by Blossom-Quad and the interior by triangle merging. This hybrid approach is almost as fast as triangle merging and much faster than Blossom-Quad due to its low complexity of $O(N \log N)$. In terms of quality, the hybrid approach is comparable to Blossom-Quad and better than triangle merging making it a fast alternative especially on large meshes with millions of triangles.

References

1. Androsov, A., Behrens, J., Danilov, S.: Tsunami modelling with unstructured grids. interaction between tides and tsunami waves. In: Krause, E., Shokin, Y., Resch, M., Kröner, D., Shokina, N. (eds.) Computational Science and High Performance Computing IV, pp. 191–206. Springer, Berlin (2011)
2. Armstrong, C.G., Fogg, H.J., Tierney, C.M., Robinson, T.T.: Common themes in multi-block structured quad/hex mesh generation. Procedia Eng. **124**, 70 – 82 (2015). 24th International Meshing Roundtable
3. Baehmann, P.L., Wittchen, S.L., Shephard, M.S., Grice, K.R., Yerry, M.A.: Robust, geometrically based, automatic two-dimensional mesh generation. Int. J. Numer. Methods Eng. **24**(6), 1043–1078 (1987)
4. Blacker, T.D., Stephenson, M.B.: Paving: a new approach to automated quadrilateral mesh generation. Int. J. Numer. Methods Eng. **32**(4), 811–847 (1991)
5. Bommes, D., Zimmer, H., Kobbelt, L.: Mixed-integer quadrangulation. ACM Trans. Graph. **28**(3) (2009)
6. Bommes, D., Lempfer, T., Kobbelt, L.: Global structure optimization of quadrilateral meshes. In: Computer Graphics Forum, vol. 30.2, pp. 375–384. Wiley Online Library, Hoboken (2011)
7. Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., Zorin, D.: Quad-mesh generation and processing: A survey. Comput. Graph. Forum **32**(6), 51–76 (2013)
8. Campen, M.: Partitioning surfaces into quadrilateral patches: A survey. Comput. Graph. Forum **36**(8), 567–588 (2017)
9. Canann, S.A., Muthukrishnan, S.N., Phillips, R.K.: Topological improvement procedures for quadrilateral finite element meshes. Eng. Comput. **14**(2), 168–177 (1998)
10. Catmull, E., Clark, J.: Recursively generated b-spline surfaces on arbitrary topological meshes. Comput. Aided Des. **10**(6), 350–355 (1978)
11. Charakhch'yan, A., Ivanenko, S.: A variational form of the Winslow grid generator. J. Comput. Phys. **136**(2), 385–398 (1997)
12. Docampo-Sanchez, J., Haimes, R.: A regularization approach for automatic quad mesh generation. In: 28th International Meshing Roundtable. Zenodo (2020)
13. Dong, S., Bremer, P.T., Garland, M., Pascucci, V., Hart, J.C.: Spectral surface quadrangulation. In: ACM SIGGRAPH 2006 Papers, SIGGRAPH '06, p. 1057–1066. Association for Computing Machinery, New York (2006)
14. Edmonds, J.: Maximum matching and a polyhedron with 0, 1-vertices. J. Res. Natl. Bur. Stand. B **69**(125–130), 55–56 (1965)
15. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17**, 449–467 (1965)
16. Edmonds, J., Johnson, E.L., Lockhart, S.C.: Blossom I: A computer code for the matching problem. In: IBM TJ Watson Research Center, Yorktown Heights, New York (1969)

17. Ekelschot, D., Ceze, M., Garai, A., Murman, S.M.: Robust metric aligned quad-dominant meshing using Lp centroidal voronoi tessellation. In: 2018 AIAA Aerospace Sciences Meeting, p. 1501 (2018)
18. Freitag, L.A., Plassmann, P.: Local optimization-based simplicial mesh untangling and improvement. Int. J. Numer. Methods Eng. **49**(1–2), 109–125 (2000)
19. Gabow, H.N.: Data structures for weighted matching and nearest common ancestors with linking. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 434–443 (1990)
20. Georgiadis, C., Beaufort, P., Lambrechts, J., Remacle, J.: High quality mesh generation using cross and asterisk fields: application on coastal domains. CoRR **abs/1706.02236** (2017)
21. Geuzaine, C., Remacle, J.F.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Methods Eng. **79**(11), 1309–1331 (2009)
22. Kälberer, F., Nieser, M., Polthier, K.: Quadcover - surface parameterization using branched coverings. Comput. Graph. Forum **26**(3), 375–384 (2007)
23. Kettner, L.: Using generic programming for designing a data structure for polyhedral surfaces. Comput. Geom. **13**(1), 65 – 90 (1999)
24. Kolmogorov, V.: Blossom V: a new implementation of a minimum cost perfect matching algorithm. Math. Program. Comput. **1**(1), 43–67 (2009)
25. Owen, S.J.: A survey of unstructured mesh generation technology. In: Freitag, L.A. (ed.) Proceedings of the 7th International Meshing Roundtable, pp. 239–267 (1998)
26. Owen, S.J., Staten, M.L., Canann, S.A., Saigal, S.: Q-morph: an indirect approach to advancing front quad meshing. Int. J. Numer. Methods Eng. **44**(9), 1317–1340 (1999)
27. Rank, E., Schweingruber, M., Sommer, M.: Adaptive mesh generation and transformation of triangular to quadrilateral meshes. Commun. Numer. Methods Eng. **9**(2), 121–129 (1993)
28. Remacle, J.F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., Geuzainet, C.: Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. Int. J. Numer. Methods Eng. **89**(9), 1102–1119 (2012)
29. Remacle, J.F., Henrotte, F., Baudouin, T.C., Geuzaine, C., Béchet, E., Mouton, T., Marchandise, E.: A frontal delaunay quad mesh generator using the L ∞ norm. Int. J. Numer. Methods Eng. **94**(5), 494–512 (2013)
30. Rushdi, A.A., Mitchell, S.A., Mahmoud, A.H., Bajaj, C.C., Ebeida, M.S.: All-quad meshing without cleanup. Comput. Aided Des. **85**, 83 – 98 (2017). 24th International Meshing Roundtable Special Issue: Advances in Mesh Generation
31. Stimpson, C., Ernst, C., Knupp, P., Pébay, P., Thompson, D.: The verdict library reference manual. Sandia Nat. Laborat. Techn. Rep. **9**(6), 32–57 (2007)
32. Tam, T., Armstrong, C.: 2d finite element mesh generation by medial axis subdivision. Adv. Eng. Softw. Workstat. **13**(5), 313 – 324 (1991)
33. Tarini, M., Pietroni, N., Cignoni, P., Panozzo, D., Puppo, E.: Practical quad mesh simplification. Comput. Graph. Forum **29**(2), 407–418 (2010)
34. Zhu, J.Z., Zienkiewicz, O.C., Hinton, E., Wu, J.: A new approach to the development of automatic quadrilateral mesh generation. Int. J. Numer. Methods Eng. **32**(4), 849–866 (1991)
35. Zint, D., Grosso, R.: Discrete mesh optimization on GPU. In: Roca, X., Loseille, A. (eds.) 27th International Meshing Roundtable, pp. 445–460. Springer International Publishing, Cham (2019)
36. Zint, D., Grosso, R., Lunz, F.: Discrete mesh optimization on surface and volume meshes. In: 28th International Meshing Roundtable. Zenodo (2020)

Hexahedral Mesh Generation Using Voxel Field Recovery

Alexander Sergeevich Karavaev and Sergey Petrovich Kopysov

Abstract We consider a modification of the previously developed voxel-based mesh algorithm to generate models given by a triangular surface mesh format (STL). Proposed hexahedral mesh generator belongs to the family of grid methods, and its capable to use as source data both volume and surface types of model geometry representation. To define the initial position of mesh nodes, a «signed distance field» volume data file, obtained from the STL geometry, is used. A special projection technique was developed to adapt constructed orthogonal mesh on the models boundary. It provides an approximation of sharp edges and corners and performs before running any other operations with the mesh. Finally, to improve the mesh quality, additional procedures were implemented, including boundary layers insertion, bad quality cells splitting, and optimization-based smoothing technique.

1 Introduction

The first step in the process of element mesh generation belongs to the description of the initial model geometry. Surface representation is a set of faces describing a model's boundary. This description has become the main technique applied in engineering calculations and it is widely used by the existing CAD software.

However, for some applications, such as medicine and biomodeling, the only possible way is to use voxel data obtained by a CT, MRI, or MicroCT scanner [11]. Preserving geometric corners and edges is not so critical in the case of biological tissues modeling. Usually, mesh algorithms for such problems are high performance and robust. They are based on combining pairs of Cartesian voxels grid into hexahedra followed by boundary nodes position adaptation.

At the same time, generation of hexahedral meshes of a given quality with an accurate description of an arbitrary model geometry is not a completely solved

A. S. Karavaev (✉) · S. P. Kopysov
Department of Computational Mechanics, Udmurt State University, Izhevsk, Russia
e-mail: karavaev-alexander@yandex.ru

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_19

task [1, 8]. In comparison with tetrahedral meshes, there is no well-developed theory and robust general-purpose software to solve the problem under discussion.

Specially for this, a sufficient number of algorithms have been developed with their advantages and disadvantages.

For example, block decomposition approaches break the model into pieces that are easier to mesh. They provide good quality cells but are semi-automatic, and therefore time-consuming to construct complex geometries. It should be mentioned that a lot of strategies have been developed to automate the block decomposition process, such as medial surface transformation, midpoint subdivision, and singularity-restricted frame fields [1].

The advancing fronts methods (sweeping, paving, plastering, etc.) generate a hex-mesh from the boundary of the surface mesh inward [9]. Such techniques have issues with robustness, especially when two different fronts are merged during processing. The same problems of reliability have the methods based on spatial twist continuum concept which is a dual representation of an all hex-mesh that defines the global connectivity constraints [6]. In recent years, many investigations have been devoted to the indirect approaches. They convert a given tetrahedral mesh to the hexahedral one by using a different type of combining techniques [3, 8]. Unfortunately, in many cases, the result meshes are hex-dominant while containing a small number of irregular polyhedra.

The grid-based or octree family of methods are robust but tends to generate poor quality elements at the boundary of the volume and to create hanging nodes arising from size transitions [1, 10]. Also, these methods incline to produce a large number of hexahedra and are highly dependent upon the orientation of the interior orthogonal grid of hex elements. The recent attempts to quality hexahedral meshing are either robust but lack the ability to align to the surface curvature and sharp features [4, 7], or produce high quality meshes but are extremely fragile.

In this article, we propose an algorithm that provides good compromise between robustness and quality meshing. We consider a modification of previously developed voxel mesh generator which was originally intended to work with volume (tomographic) data representation [5]. Figure 1 shows the result of the algorithm for construction a multicomponent head model. Here, the boundary is reconstructed with the «Dual Contouring» algorithm [5, 11]. This approach is widely used to render tomographic images with a surface quadrangular mesh. It allows to restore the sharp features of the model contour to a certain degree. Unfortunately, in the case of STL geometry, the accuracy of the «Dual Contouring» algorithm is not sufficient. Therefore, additional options have to be considered to build hexahedral meshes of models given in STL format.

Our approach belongs to the family of grid methods. It utilizes initial surface representations and «signed distance fields» voxel data format obtained from STL file.

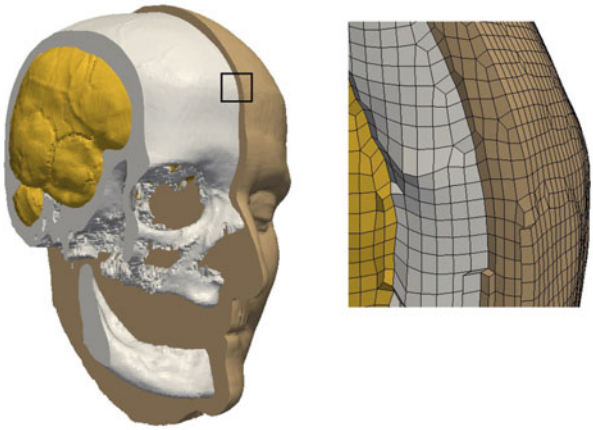


Fig. 1 Human head model with three materials. Hexahedral mesh constructed from tomographic data by the authors' algorithm [5]

2 Initial Data Representation

As mentioned before, the proposed method can generate meshes from both surface and volumetric types of model representation.

In literature, volumetric (voxel) data $\mathcal{V}$ is defined as scalar values of an implicit function $\mathcal{F}$ on a Cartesian coordinate grid $\mathcal{V} = \mathcal{F}(i, j, k)$, where i, j, k are indexes in the form of x, y, z coordinates of Cartesian grid. The isosurface (surface of equal values) corresponding to the value α represents points of a constant value within a volume $I_{\mathcal{F}}(\alpha) = \{ (x, y, z) \mid \mathcal{F}(x, y, z) = \alpha \}$.

Volumetric data may be viewed by extracting isosurfaces from the volume and rendering them as polygonal meshes or by rendering the volume directly as a block of data.

For now, the algorithm takes a standard DICOM volumetric data file obtained from tomography scans to construct meshes of biological tissues. In the case of STL-models, the «signed distance fields» (SDF) data structure is used.

Each voxel in a SDF file contains a positive or a negative shortest distance to the model surface according to its relative position. Therefore, a border of the model is set by the isosurface of value $\alpha = 0$.

Let the model Ω be bounded by a surface triangular mesh T. Then, for an arbitrary voxel $\boldsymbol{v}$, the function $\mathcal{F}$ could be written as

$$\mathcal{F}(\boldsymbol{v}) = S_{\mathcal{T}}(\boldsymbol{v}) \times \min_{t \in T} d(t, \boldsymbol{v}), \quad S_T(\boldsymbol{v}) = \begin{cases} +1, & \text{if } \boldsymbol{v} \subseteq \Omega, \\ -1, & \text{if } \boldsymbol{v} \not\subset \Omega, \end{cases}$$

where $d(t, \boldsymbol{v})$ denotes the minimal distance between a triangle $t \in T$ and a voxel $\boldsymbol{v} \in \mathcal{V}$.

The initial SDF voxel array is constructed from a given STL geometry by a straightforward («brute force») conversion technique. For every voxel $\boldsymbol{v} \in \mathcal{V}$ it calculates the distance to all triangles of T and takes the minimal one. Because of this, the execution time is a little bit high and it is efficient to use parallel computations. On the other hand, this method calculates the accurate Euclidean distance field without errors and has a simple implementation.

The obtained volume data file implicitly contains all information about an orthogonal hexahedral grid covering the model. Thus, a result of the mesh generation directly depends on the accuracy of voxels values in $\mathcal{V}$.

3 Hexahedral Mesh Generation

According to Schneider [10], the mesh of the model should fulfill the following requirements: each point of the model is covered by the mesh point; each edge of the model is covered by a sequence of mesh edges.

To overcome this problem, we construct the set Λ of «characteristic» edges and nodes. The set Λ is considered to be a «frame» of the model which preserves all its sharp features. The resulting hexahedral mesh $\mathcal{T}$ has to cover the elements of Λ with a continuous chain of its edges and nodes.

To construct Λ we follow Schneider's idea and don't take into account edges of triangular mesh T whose adjacent faces enclose a dihedral angle of about 180°.

In Fig. 2a and b we can see the STL geometry of a half part of the metal sleeve and its set Λ with 10 «characteristic» nodes and 15 «characteristic» edges.

The second step of mesh generation is the construction of the orthogonal structured grid with size h (Fig. 2c). The grid size h is chosen in advance, and it defines an interval (span) between voxels that will be taken to form mesh nodes. This size determines the number of cells in the resulting mesh and thus a final accuracy of the model description. The minimal value of h is 1, and it means that all voxels of $\mathcal{V}$ taking into account during mesh generation.

For every 8 neighboring voxels with $\gamma = 4$ or more positive values a new cell is constructed.

Let $\dim_z \times \dim_y \times \dim_x$ be the dimension of SDF voxel array $\mathcal{V}$ and $sgn_{\mathcal{F}}^{(x,y,z)}$ is a signum function of voxel value with indexes (x, y, z), $sgn_{\mathcal{F}}^{(x,y,z)} = sgn\left(\mathcal{F}\left(\boldsymbol{v}^{(x,y,z)}\right)\right)$, then the above process can be written as following: "Algorithm 1".

In addition to number $\gamma = 4$ of positive voxels, we also require that they have to belong to one quadrangle of a hexahedron. Such choice provides a situation when every face of a cell has at least 2 points inside the model. From tests, we established that such value $\gamma = 4$ is optimal because of a less number of additional hexahedra (in comparison with the case $\gamma < 4$, see Fig. 3a and b) and uniform size of boundary and inner cells (that is not provided in the case of $\gamma > 4$, see Fig. 3c and d).

It should be mentioned that the number of positive voxels is not strict and can be lower than 4. Also, there is an option to add any other conditions, such as acceptable

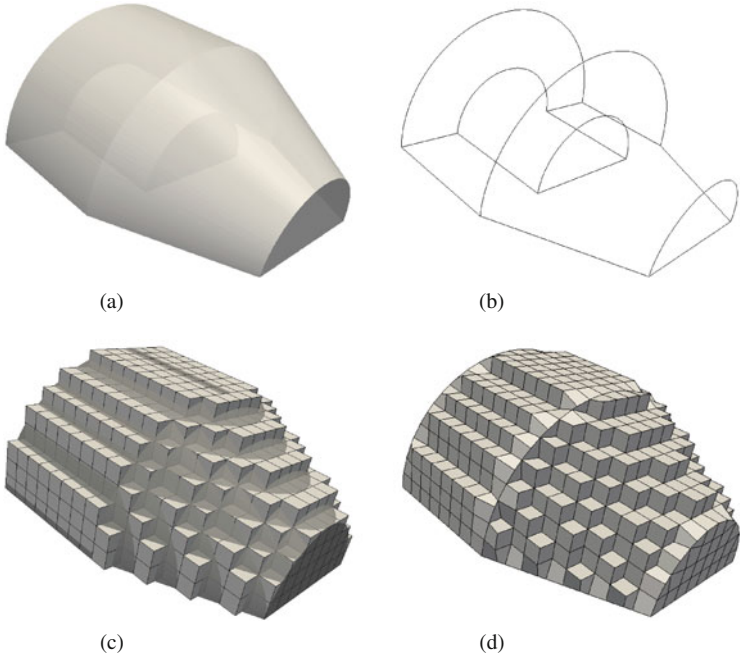


Fig. 2 Hexahedral mesh construction. (**a**) STL-geometry. (**b**) Model's «frame». (**c**) Structured hex-grid (with background STL surface). (**d**) Grid projected to the model's «frame»

Algorithm 1 Orthogonal structured grid generation

for $z = 1; z < \dim_z / h; z++$ **do**
  for $y = 1; y < \dim_y / h; y++$ **do**
    for $x = 1; x < \dim_x / h; x++$ **do**
      if $sgn_{\mathcal{F}}^{(x,y,z)} + sgn_{\mathcal{F}}^{(x+h,y,z)} + sgn_{\mathcal{F}}^{(x+h,y+h,z)} + sgn_{\mathcal{F}}^{(x,y+h,z)} + sgn_{\mathcal{F}}^{(x,y,z+h)} + sgn_{\mathcal{F}}^{(x+h,y,z+h)} + sgn_{\mathcal{F}}^{(x+h,y+h,z+h)} + sgn_{\mathcal{F}}^{(x,y+h,z+h)} \geqslant \gamma$ **then**
      end if
      Create hexagon c from the voxels $\left\{\boldsymbol{v}^{(x+i,y+j,z+k)}\right\}_{i,j,k=0,h}$
    end for
  end for
end for

distance of voxels to the surface. For example, the cell c is not constructed if $\exists \boldsymbol{v}' \in \left\{\boldsymbol{v}^{(x+i,y+j,z+k)}\right\}$, $i, j, k = 0, h$ such that $\mathcal{F}\left(\boldsymbol{v}'\right) < \varepsilon_0 < 0$, where ε_0 is some threshold value defined by the user.

The third step is the projection of the orthogonal grid onto the set Λ to provide an approximation of «characteristic» edges and nodes of the model. This procedure is slightly different from the isomorphism technique proposed by Schneiders [10]. In our case, the adaptation is performed before running any smoothing or topological operations with the mesh, and it can be divided into 3 main steps:

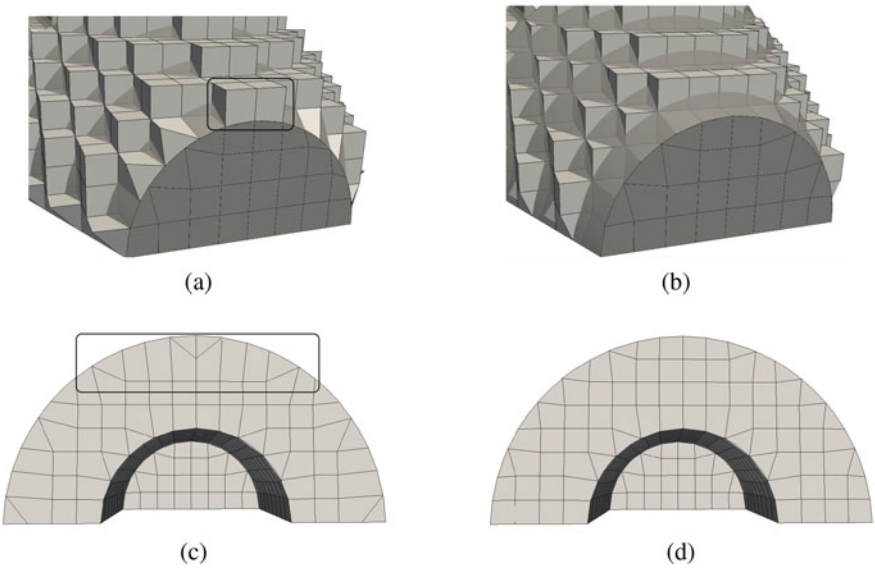


Fig. 3 The choice of the number γ of positive voxels in a cell during structured grid generation: (**a**, **b**) additional hexahedra (in frame) in the case of $\gamma < 4$ in comparison with $\gamma = 4$; (**c**, **d**) lack of uniform cell size (large ones in frame) in the case of $\gamma > 4$ in comparison with $\gamma = 4$

(1) At first on each «characteristic» node η an appropriate grid node $\mathbf{x}$ is projected. Usually, $\mathbf{x}$ is taken as the nearest one to η, but sometimes other rules can be adopted. For example, if two or more nodes have the closest distance to η, the node which is connected to all of them should be chosen.
(2) The next step is the covering «characteristic» edges of the model. At first, all boundary nodes with distance to the «characteristic» edges less than some threshold $\varepsilon_1 \approx 0.1 \times h$ are projected onto them.
(3) The remaining uncovered sections of Λ is handling with sequential nodes projection procedure. It starts from every node $\mathbf{x}$ which is already projected onto Λ. The next candidate $\mathbf{x}'$ is selected from boundary neighbors of $\mathbf{x}$ according to proposed priority criteria which work in the following way:

   – The neighboring nodes, which have two or more boundary inverted quads after projection, are not considered.
   – The nodes with 3 adjacent boundary quads are more preferable than others.
   – Among the nodes with the previous two conditions are fulfilled, the one with the shortest distance to the «characteristic» edge should be taken.

Figure 2d demonstrates the result of the described operation. It should be mentioned that every selected node $\mathbf{x}' \in \mathcal{T}$ is projected to the nearest edge $e \in E_\Lambda$, and this position typically doesn't coincide with vertices of the triangular mesh T.

After the set Λ has been constructed, the remaining boundary nodes are projected to the surface of the model. Here we use two different techniques: the well-known

ray-tracing method and a special iterative movement through the volume of SDF data field described in [11]. The first procedure is more accurate, but the latter one has better performance in the case of STL geometries with a high number of triangles. Such STL meshes are often required to describe models with curvilinear surfaces. Each iterative movement consists of 2 steps:

– Check an approximation of node $\mathbf{x}$ to the iso-surface $I_{\mathcal{F}}(\alpha)$. If the value $\mathcal{F}(\mathbf{x})$ is close enough to α, e.g. $|\mathcal{F}(\mathbf{x}) - \alpha| \approx 10^{-4} \times h$, then it can be assumed that $\mathbf{x}$ locates on the surface and further iterations are not needed.
– Define new coordinates of node by moving it in a small distance $\mathbf{x}^{+} = \mathbf{x} + \delta\mathbf{n}$, where $\mathbf{n}$ is the normal of vertex $\mathbf{x}$ and δ the size of iteration step.

In the case of SDF data format, the boundary of the model is set with zero iso-surface of value $\alpha = 0$.

According to our tests, in most cases, the implementation of these procedures allows to fully describe the model so that each «characteristic» node is assigned to one corresponding mesh node, and each «characteristic» edge is completely covered by mesh edges.

As one can see from Fig. 4a, the projection of remaining nodes to the surface allows to fully described initial STL model with sufficient accuracy.

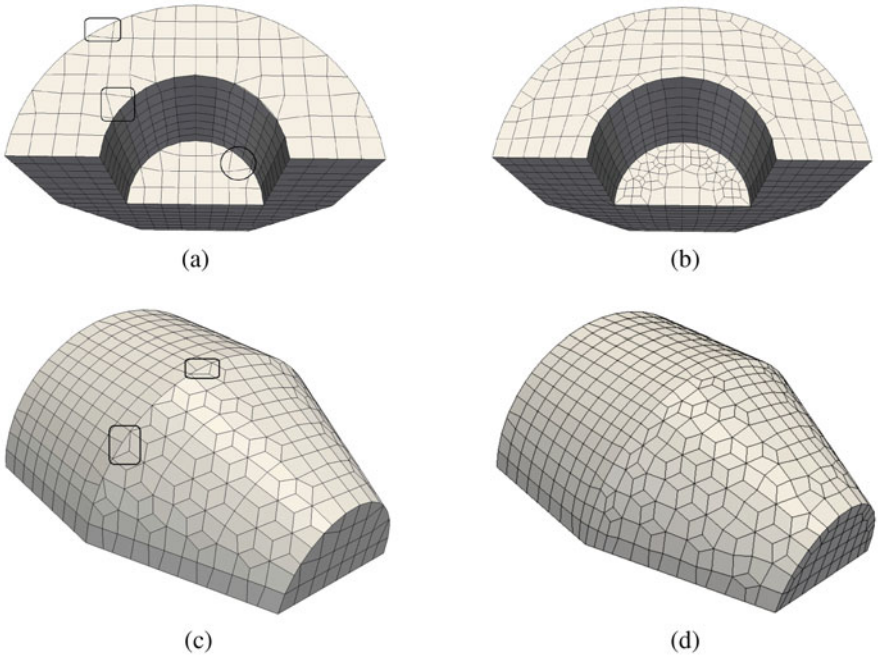


Fig. 4 Removing inverted boundary hexahedra and quads (in frames) with: (**a**, **b**) volume layer insertion and splitting procedures; (**c**, **d**) surface layer insertion along a «characteristic» edge

4 Mesh Quality Improvement

To improve the mesh quality, a Laplacian smoothing procedure is applied. Unfortunately, this operation is not enough to get rid of invalid mesh elements along the model boundary. This is because of hexahedra may have two or three faces to be projected on the same plane, and surface quads may have two edges to be projected on the same sharp line.

To resolve these issues, three standard procedures for grid type meshes are used.

First, a volume layer of hexahedra is inserted around the constructed mesh so that new boundary elements have only one face to be projected on the real geometry (Fig. 4a and b). The shape of boundary quads with angle $\geqslant 180°$ (two are marked with rectangular frames in Fig. 4a) are significantly improved after the volume layer of hexahedra has been inserted (Fig. 4b).

It should be mentioned, that sometimes for better mesh quality it is useful to skip the insertion of volume layer along surfaces which are orthogonal to the Cartesian axes. For instance, for the final mesh displayed in Fig. 4d, bottom, front, and back boundary surfaces stay unchanged, and only for the top curvilinear surface an additional layer of cells was added.

Likewise, a second layer of hexahedra is inserted along the set of «characteristic» edges, if the solid angle alongside them is sufficiently smaller than 180°. Also, another restriction exists, the operation is not allowed if the number of «characteristic» edges adjacent to a «characteristic» point does not equal 2 or 3.

An application of this optimization step is demonstrated in Fig. 4c and d. All hexahedra with flat angles (two are marked in frames) have been removed.

The remaining degenerate elements are removed by a splitting procedure (one is marked with circular frame in Fig. 4c) proposed in [10]. All hexahedra with an inverted face are split into four new ones by specific template (Fig. 4d), thus, the degenerate boundary quad along «characteristic» edge has vanished. To maintain the conformity of the mesh, the neighbor elements are also split up (Fig. 4d).

Finally, to improve the mesh quality, a smoothing procedure is applied. We use different types of Laplacian smoothing weighted by the edges length, quads areas, and volume of hexahedra. It should be mentioned that often a simple Laplacian technique is not enough to get properly shaped cells.

One of the options is to use the optimization-based smoothing method proposed in [2]. In our algorithm, we apply a modification of this method for the case of hexahedral cells. This approach is an iterative one and could improve the value of any hexahedra quality measure q which varies in the interval $[-1; 1]$.

The algorithm works with three quality criteria for hexahedra cells, among which are scaled Jacobian, inverse aspect ratio, and skew metric. Also, we propose a normalized measure for a warping angle Θ of a quadrilateral face $q_\Theta = (\pi - \Theta)/\pi$.

Since the optimization-based technique is time-consuming, it is often applied to the nodes with poor quality cells. In other cases, various types of Laplacian smoothing are used [2].

5 Test Examples

In this section, the results of the algorithm on three types of STL geometry and one volumetric data representation are shown.

All demonstrated hexahedral meshes do not contain inverted cells. The measure of the minimum scaled Jacobian has a value of $Q_J \geqslant 0.2$, which is acceptable for finite element algorithms. This quality metric was chosen as an objective function for the optimization-based smoothing technique. Meshes quality characteristics can be viewed in Table 1.

Figure 5a demonstrates a cut view of the metal sleeve model with a cylindrical inclusion, the result hexahedral mesh contains $N = 15{,}930$ nodes and $C = 13{,}842$ cells. A relatively large value of maximal aspect ratio $Q_a = 33.8$ emerges in the boundary layer cells around inclusion.

Maximal skew metric of cells in the star model is $Q_s = 0.84$. Though, all boundaries of the model are flat, the maximal warp angle of the inner quads of the mesh equals $\Theta_w = 61.1°$. This is because the only scaled Jacobian value was optimized, and other metrics were not taken into account.

Due to the small number of «characterictic» features (only edges and no nodes), the mesh of the spring model has better values of scaled jacobian $Q_J = 0.4$ and aspect ratio $Q_a = 4.2$. The most optimal skew value $Q_s = 0.63$ was obtained for a cube model containing 13 elliptic inclusions.

The human head model (Fig. 1) was generated from a DICOM volume data file. Hounsfield scale values (or the values of extracted isosurfaces) for each material were the following: soft tissue $\alpha = -720$, brain $\alpha = 1$, skull $\alpha = 172$. In addition to scaled Jacobian, a warping angle of quadrangles was also improved. Quality measures of the resulting mesh can also be seen in Table 1.

Table 1 Meshes characteristics

Characteristic	Sleeve	Star	Spring	Cube	Head
Dim	$70 \times 129 \times 129$	$129 \times 129 \times 28$	$86 \times 151 \times 86$	65^3	1029^3
h	4	4	1	1	2
N	15,930	3894	239,757	288,152	1,363,369
C	13,842	2860	195,904	276,003	1,308,341
Θ_w	73.6°	61.1°	54.3°	57.1°	37°
Q_J	0.22	0.29	0.4	0.4	0.2
Q_a	33.8	20.7	4.2	4.9	15.9
Q_s	0.84	0.84	0.8	0.63	0.85

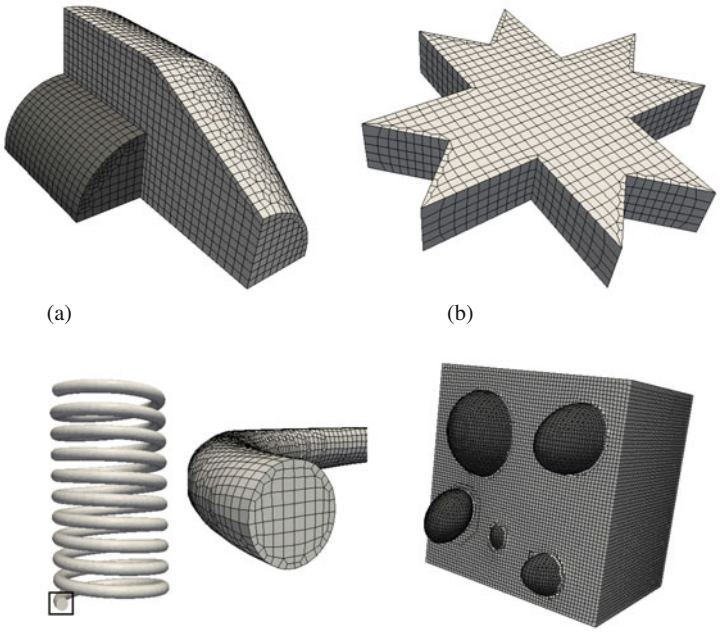


Fig. 5 Unstructured hexahedral mesh examples. (**a**) Sleeve with inclusion. (**b**) Star. (**c**) Spring. (**d**) Cube with inclusions

6 Conclusions

This paper presented a further modification of early developed volume hexahedral mesh generator as applied to the models described by STL geometry.

The main developments of new functionality are the following:

- Obtaining from the original STL geometry a set Λ of «characteristic» edges and nodes, which positions should be saved in the resulting hexahedral mesh $\mathcal{T}$. The set Λ is considered to be a «frame» of the model, which preserves all its sharp angles and faces.
- Construction and mapping the part of boundary nodes of $\mathcal{T}$ onto the set Λ to provide an approximation of boundary contour. The procedure is based on a sequential node projection at every element of Λ. Here is the most suitable nodes are selected according to proposed priority criteria, which evaluate adjacent hexahedra number, quality measures of cells and faces, and the distance to the elements of Λ.
- Removing degenerated cells of $\mathcal{T}$ emerging along «characteristic» edges. The operation includes insertion of an additional surface layer of hexahedra along the edges and splitting inverted cells into four new ones according to the template described in [10].

Implementation of the new procedures allows to generate a uniform mesh accurately describing the original geometry of an arbitrary STL-model.

Generation of multicomponent models for the case of voxel data is carried out by a sequential mesh construction for each isosurface value in descending order.

In the case of a surface representation, the application of such an approach seems difficult, since the SDF file obtained from the STL-geometry contains only one isosurface value $\alpha = 0$. Thus, the SDF file stores information about only one material of the model.

Using one file in the SDF format allows to solve a specific case when the model contains areas with inclusions located entirely inside, or on one level with the orthogonal plane (Fig. 5a and d).

To implement the general case it is necessary to use a set of SDF files, generated for every specific material. Then all orthogonal meshes, obtained from each SDF file, should be assembled in one.

At the moment, an integration of the described technique to the algorithm just like an implementation of adaptive hexahedral mesh generation are questions of further research.

References

1. Awad, M., Rushdi, A., Misarah, A., Mitchell, S., Mahmoud, A., Chandrajit, B., Ebeida, M.: All-hex meshing of multiple-region domains without cleanup. Procedia Eng. **163**, 251–261 (2016). https://doi.org/10.1016/j.proeng.2016.11.055
2. Cannan, S., Tristano, J., Staten, M.: An approach to combined Laplacian and optimization-based Smoothing for triangular, quadrilateral, and quad-dominant meshes. In: 7th International Meshing Roundtable. Dearborn. Michigan, pp. 479–494 (1998)
3. Gao, X., Jakob, W., Tarini, M., Panozzo, D. Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. ACM Trans. Graph. **36**(4), 114 (2017)
4. Gao, X., Shen, H., Panozzo, D.: Feature preserving octree-based hexahedral meshing. Comput. Graph. Forum **38**, 135–149 (2019). https://doi.org/10.1111/cgf.13795
5. Karavaev, A., Kopysov, S.: The method of unstructured hexahedral mesh generation from volumetric data. Comput. Res. Model. **5**, 11–24 (2013). https://doi.org/10.20537/2076-7633-2013-5-1-11-24
6. Ledoux, F., Weill, J.C.: An extension of the reliable Whisker Weaving algorithm. In: Proceedings of the 16th International Meshing Roundtable. Springer, Berlin (2008)
7. Marechal, L.: Advances in octree-based all-hexahedral mesh generation: Handling sharp features. In: Proceedings of the 18th International Meshing Roundtable. Springer, Berlin (2009)
8. Pellerin, J., Johnen, A., Remacle, J.-F.: Identifying combinations of tetrahedra into hexahedra: a vertex based strategy. Procedia Eng. **203**, 2–13 (2017). https://doi.org/10.1016/j.proeng.2017.09.779
9. Ruiz-Girones, E., Roca, X., Sarrate, J.: The receding front method applied to hexahedral mesh generation of exterior domains. Eng. Comput. **28**, 391—408 (2012). https://doi.org/10.1007/s00366-011-0233-y
10. Schneiders, R.: Automatic generation of hexahedral finite element meshes. Comput. Aided Geom. Des. **12**, 693–707 (1995). https://doi.org/10.1016/0167-8396(95)00013-V
11. Zhang, Y., Bajaj, C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Comput. Methods Appl. Mech. Engrg. **195**(9), 942–960 (2006). https://doi.org/10.1016/j.cma.2005.02.016

Generation of Boundary Layer Meshes by the Enhanced Jump-and-Walk Method with a Fast Collision Detecting Algorithm

Jie Cao, Fei Yu, Zhonghai Gao, S. H. Lo, and Zhenqun Guan

Abstract Boundary layer meshes as an important part of hybrid mesh are often constructed via advancing layer method for CFD fluid simulations. One major difficulty in implementing a robust boundary layer meshing tool is to handle mesh collisions. In this work, the authors propose to enhance the Jump-and-Walk method by constructing a medial surface mesh in a constrained Delaunay triangulation to detect the safe marching space of front points. Medial surface can build a separation wall between adjacent boundary surfaces to prevent the intersection of boundary layer elements, and the computation of the distance between front points and the medial surface can be accelerated by walking through tetrahedra in sequence without additional data structures. A range of tests were performed for several complex configurations to demonstrate the capability of the method.

1 Introduction

Considering the ease of use and calculation accuracy, the hybrid grid that takes full advantage of semi-structured prismatic grid and unstructured tetrahedral grid and considers the boundary layer effect is considered to be the best grid form for CFD calculations [1–3]. Since unstructured grid meshing techniques are very mature now [4–6], so the focus is commonly on the prismatic mesh generation. Advancing layer method is one of most popular algorithms for boundary layer meshing. Early attempts of this method were proposed by Lohner [7] and Pirzadeh [8]. There is a problem that self or global-intersecting meshes may appear especially at narrow

J. Cao (✉) · F. Yu · Z. Gao · Z. Guan
Dalian University of Technology, Dalian, China
e-mail: caojie@mail.dlut.edu.cn; fei.yu@mail.dlut.edu.cn; gaozhonghai@mail.dlut.edu.cn; guanzhq@dlut.edu.cn

S. H. Lo
The University of Hong Kong, Hong Kong SAR, China
e-mail: hreclsh@hku.hk

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_20

valley corners or adjacent surfaces. How to robustly and efficiently detect the grid collisions is one of main issues in related studies.

Collision detection is a point location problem in nature. A data structure is usually needed to accelerate the search for objects, such as point, edge, cell, and so on. A k-d tree can be employed with a computational time between $O(n \log n)$ and $O(n^2)$ for preprocessing the tree and $O(\log n)$ for each query. In addition, a closest point search algorithm based on uniform cubes can achieve a time efficiency of $O(1)$ with small amount of memory for evenly distributed point set. One of the most common algorithms to locate the position of target point in a triangulation is Walk-through method [9–12]. A Delaunay triangulation can be constructed in $O(n^{5/4})$ expected time in 3D and the computational complexity of the algorithm is generally $O(n^{1/4})$ per point inherently bound by the connectivity of the triangular mesh [11]. In general, collision detection algorithms can be divided into two categories according to the means used: (1) Intersection test; (2) Space detection.

If an intersection test is used, benefiting from the determined locations of elements, it is possible to locate all intersecting or overlapping prisms precisely [7, 13, 14]. However, an additional data structure, like a k-d tree, Octree, alternating digital tree (ADT) [15] and so on, is needed to reduce the time complexity of locating operations.

Grid intersections also can be removed by computing the marching space of front points [8, 9, 16, 17]. If a constrained Delaunay triangulation (CDT) is taken as the background grid, additional data structure is not necessary as we can call an existing tetrahedral mesh generator. One way is to fill the domain with a CDT and advance layers while always keeping the volume of elements positive [17]. This method is simple and robust but generally needs a large number of iterations to adjust the CDT, which may result in a low-efficiency problem. Jump-and-Walk method as a variant of Walk-through method was first proposed by Mücke et al. [11] and was used to compute the maximal advance distances of points on the boundary to reduce the total number of layers in narrow areas by Wang and Mare [9]. The time of only handling the boundary points is much less than other methods. However, the robustness of Wang and Mare's method is poor for complex configurations.

Medial surface as a skeleton abstraction of solid object provides a representation that simply captures the geometric proximity of boundary elements, which is described by the locus of the center of the maximal sphere as it rolls along the boundary. A continuous expression by geometric surfaces or a triangular mesh is usually necessary in many applications, such as path planning, size control [18, 19], structured mesh generation [20, 21] and so on. Medial surface also can be a natural wall to separate the opposite frontiers of boundary layer meshes at narrow regions and avoid the global and local grid collisions.

In this work, we improve the robustness of the Jump-and-Walk method by constructing a medial surface mesh in the background grid to generate valid and high-quality boundary layer meshes. We aim to make the boundary layer meshing procedure easy to build and quick to execute with a simple data structure.

2 Overview

The goal of this paper is to develop a fast collision detection algorithm to avoid the intersections of boundary layer meshes. The core of the enhanced Jump-and-Walk method (EJW) is a Ray Tracing accelerator based on a Walk-through algorithm. A CDT of boundary points is constructed first as the background mesh with a general tetrahedral mesh generator. We then test all front points on the boundary surface and make a rough adjustment to their total marching distances by walking through tetrahedra along the marching directions. After that, the marching pathes still have a risk in crossing each others. We introduce a medial surface mesh into the background mesh to build a wall between adjacent front points When front points are advanced layer by layer, all self-intersecting prisms are cleaned compulsively by computing the element Jacbian. In addition, we will test all risky points at each layer based on the new background mesh to remove the remaining local and global-intersecting prisms. Isotropic meshes are filled in the remaining region by Tetgen finally after boundary layer meshes are generated. A brief of our algorithm is provided in Algorithm 1.

Algorithm 1 An overview of hybrid grid generation

Require: Closed boundary surface mesh.
Ensure: Hybrid mesh consisting of boundary layer meshes and isotropic tetrahedral meshes.
1: Initialize marching frontiers; compute and smooth marching directions and distances.
2: Construct a CDT background mesh and adjust the total marching distances of front points by EJW.
3: Introduce a medial surface mesh in the CDT as the new background mesh of EJW.
4: **while** $i < n$ and $N_{frontiers} > 0$ **do**
5: Initialize the i^{th}-layer marching frontiers; compute and smooth the marching directions and distances.
6: Front point will stop marhcing if it satisfies one of the following conditions:

- Any adjacent prism generated produces a negative Jacbian.
- A multilevel difference in the number of neighboring layers will form.
- Current total marhcing distance will exceed its allowed value or current march is tested to be risky by EJW.

7: Advance and update the frontiers.
8: $i = i + 1$.
9: **end while**
10: Fill isotropic meshes in the remaining region bounded by boundary layer meshes.

3 Computation and Smoothing of Marching Directions and Distances

The orthogonality of prism is an important property for the accuracy of fluid calculation. The initial marching direction of front point is determined by the normals of facets connected to the point. An angle-weighted average of these normals is usually taken to produce a reasonable surface normal at the patch as shown in Fig. 1a. However, it may result in an invalid visibility condition for the surface of discontinuous slopes, such as wing-fuselage junctions, sharp edges, troughs and peaks.

This situation can be avoided with a visibility polygonal cone constructed by extending the facets around the front point. As a simplification of the region, a maximal inscribed circular cone with a half-cone angle β_{visi} can be constructed as shown in Fig. 1b [1], and its central axis can produce a valid normal vector most orthogonal to the sharp patch. Consider that most of the boundary surface is smooth, it is inefficient to compute visibility cones for all front points. To balance the reliability and efficiency of the algorithm, an alternative strategy relative to the maximal angle $\theta_{\max}$ between adjacent facet normals is proposed:

- If $\theta_{\max} < 30°$, the weighted average of surface normals is adopted;
- If $\theta_{\max} \geq 30°$, the visibility cone is computed.

The initial marching distance of front point could be a function value of characteristic angle of the patch that produces a relatively large marching distance in concave regions and a small value in convex regions to increase grid orthogonality

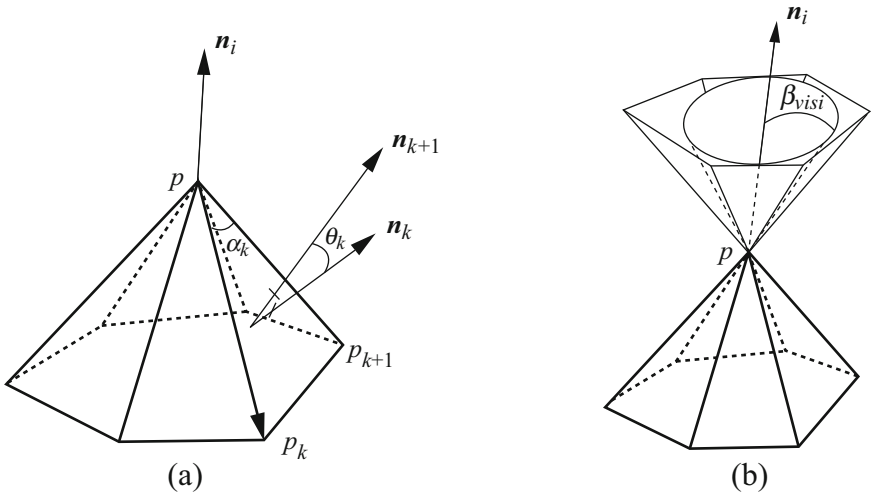


Fig. 1 Computation of the normal at a point p: (**a**) the angle-weighted average of surface normals and (**b**) the axis vector of visibility cone

as the layers grow. Wang and Mare [9] defined the ith-layer marching distance with the customized layer thickness d_i^l as

$$d_i = (1 \pm |\cos(\beta_{\text{visi}})|)d_i^l \ . \tag{1}$$

The reason to select the cos function is that its value is between -1 and 1 and insensitive to a small variation of the β_{visi}. After the initial marching directions and distances are computed, an inverse-distance weighted Laplacian smooth is used to prevent grid overlap especially for closely spaced points.

4 Validity Check for Elements

Several tests about the validity of prisms are carried out before the frontiers are advanced:

(1) Element quality. The orthogonality of side edges and the aspect ratio of two triangles are two important quality indexes for prism. A scaled-aspect-ratio quality measure [16] is adopted in our algorithm. When the quality factor $\varrho \leq 0$, prism is a collapsed element which is not allowed to generate.
(2) Topological connection. If the marching status of a point is turned off, all frontiers around it stop marching and pyramid transitional elements are generated between prisms and tetrahedra. The front point on the border of the marching and non-marching frontiers will become a cliff point in the next layer. The marching status of cliff points is always closed, then the number difference of adjacent layers will stay below 2 and the transitional elements can achieve a good quality.
(3) Collision detection. In this work, EJW is proposed to make a quick adjustment to the total marching distances of all front points on the boundary and then estimate part of front points at each layer whether they have enough space to march to the next layer.

5 Construction of the Background Grid with Medial Surface Mesh (MSM)

A CDT of boundary points is constructed and then is split to produce a decomposed CDT (DCDT) as shown in Fig. 2. To quickly locate the medial surface when walking through the tetrahedra, a MSM is approximately represented by a triangular mesh extracted from the DCDT. The details are depicted in Algorithm 2.

As shown in Fig. 3, there are seven kinds of tetrahedra totally according to the number of boundary edges contained. This algorithm can build a tight wall mesh in a linear complexity by a two-step marking. The topology of MSM is closely

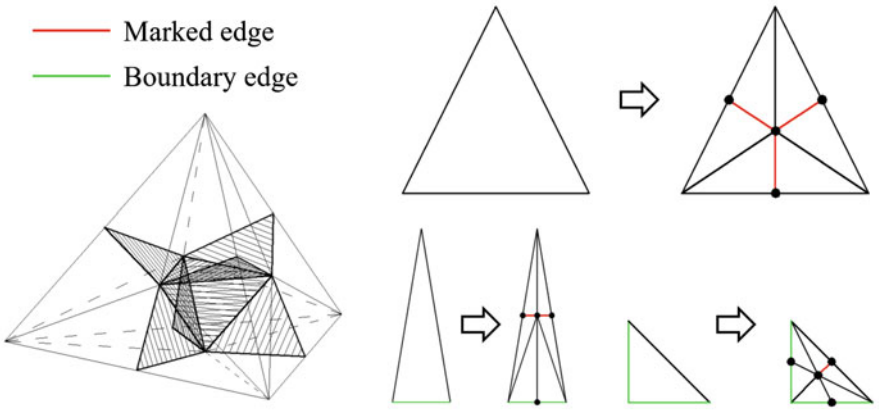


Fig. 2 Left: a tetrahedron is decomposed into 17 small tetrahedrons. Right: The face containing 0, 1, or 2 boundary edges is decomposed by inserting extra points

Algorithm 2 Construction of MSM

Require: A CDT
Ensure: A MSM located in the DCDT
1: **for each** tetrahedron ∈ CDT **do**
2: Split it into 17 small tetrahedra by inserting points on the edges or inside the faces.

- 6 points are inserted at the midpoints of edges.
- 4 points are placed inside facets. For the facet containing 1 boundary edge, a point is inserted at the midpoint of the edge joining the midpoints of two non-boundary edges; otherwise, the point is inserted at the barycenter of the facet.

3: Mark small facets inside the tetrahedron.

- Mark all small facets consisting of three new points.
- If the tetrahedron contains boundary edges, unmark the small facets connecting to the midpoint of boundary edge; if the number of boundary edges $\neq 2$, unmark the small facets opposite to the endpoint of boundary edge.

4: **end for**
5: Collect all marked facets as the MSM.

related to the tetrahedralization, which is similar to the dual graph of the CDT, i.e., Voronoi diagram. A further filtering for the MSM is not allowed to avoid small holes produced. Although the MSM is noisy near curved surfaces, as shown in Fig. 4, the intersection test will not be performed as the advance of front points is safe after the smoothing of marching vectors. The facets of MSM are alternately parallel to the boundary surfaces on their two sides with a size half of the boundary elements. If the angle between the normal vectors of boundary surfaces is near 180°, the MSM tends to be a planar surface with a minimal approximate error.

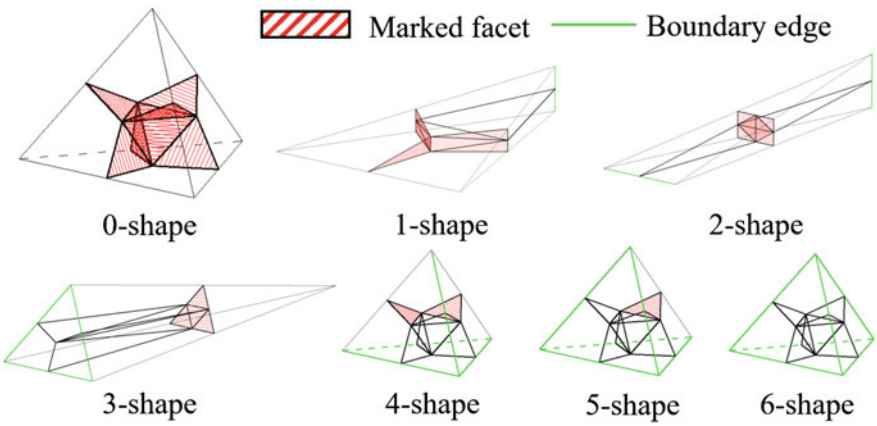


Fig. 3 Tetrahedrons containing 0–6 boundary edges with facets marked inside

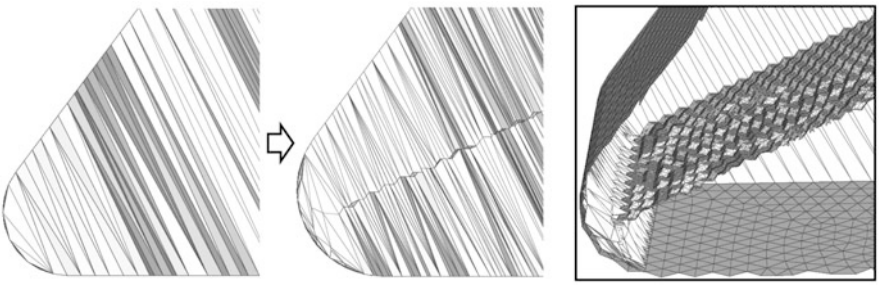


Fig. 4 A MSM is built in a DCDT

6 The Enhanced Jump-and-Walk Method (EJW)

By walking through the tetrahedra of CDT in sequence along the marching direction of a front point on the boundary, each iteration of Jump-and-Walk method (JWM) will reach a new tetrahedron. This process is repeated until the tetrahedron containing target point is found or the walk hits a boundary surface. Most grid intersections could be resolved by compressing the thickness of boundary layer meshes with the maximal walking distance computed by JWM. The algorithm is shown in Algorithm 3. To improve its robustness, EJW is proposed with two parts. EJW-A is the same as JWM while EJW-B is an extra collision test based on JWM for the front points still with a certain risk of intersecting with other grid after the adjustment of EJW-A.

Algorithm 3 JWM

Require: A DT with the topology data structure of $O(1)$ query complexity; Starting point p; Target point q which is far from p along the marching direction

Ensure: The maximal marching distance $d_{\max}$ of p

1: Initialize a random tetrahedron $\sigma_0 \in$ DT with p as one of vertexes.
2: $\sigma \leftarrow \sigma_0$.
3: **while** $n < n_{\max}$ **do** ▷ $n_{\max}$ is the maximum number of iterations limited
4: Determine a face τ of σ along the walking direction:

- no τ is selected, then q is inside σ, $d_{\max} = \|\mathbf{q} - \mathbf{p}\|$.
  return $d_{\max}$.
- τ is selected and lies on boundary, then q is outside the DT, $d_{\max} = \|\mathbf{p_{inter}} - \mathbf{p}\|$.
  return $d_{\max}$. ▷ $\mathbf{p_{inter}}$ is the intersection point between the walking ray and the τ
- τ is selected and σ_{neigh} can be obtained.

5: Jump over the τ and walk along the σ_{neigh}.
6: $\sigma \leftarrow \sigma_{\text{neigh}}$.
7: $n = n + 1$.
8: **end while**

6.1 Risky Cliff Points

Collision detection of the meshes relying on JWM is not always reliable. Figure 5 shows two typical examples that the test may fail. At the corner, the space is gradually increasing. The prisms generated upon the frontiers propelled from inflection points will be twisted first when the walking rays overlap each other. These frontiers then stop marching, from which steplike boundary layer meshes will be generated as shown in Fig. 6 because the numbers of neighboring layers cannot differ by more than one in our algorithm. As the layers increases, grid may intersects locally. At the exit of narrow channel, when the space increases sharply, global intersections may appear in a similar form even though the JWM has compressed the total thickness of layers in the channel properly. Each layer of

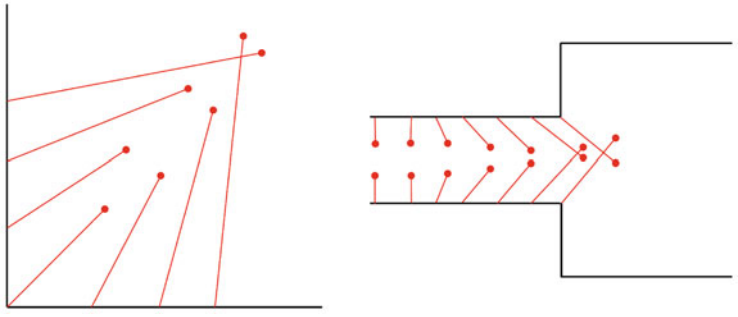


Fig. 5 Two typical examples that collision detection may fail just relying on Jump-and-Walk method. Left: at a corner. Right: at the exit of a narrow channel

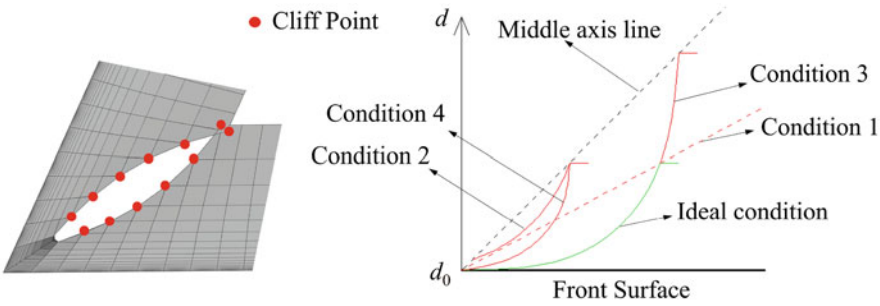


Fig. 6 Under one of the four conditions, mesh collisions may happen at cliff points. Condition 1: small corner angle. Condition 2: large minimum number of layers. Condition 3: large total number of layers. Condition 4: big growth rate of layer thickness or small surface mesh size

the steplike meshes will produces some cliff points on the edges. The cliff point is a point on which an incomplete front patch is centered. As the number of layers increases, grid intersections first appear at the cliff points, which is related to corner angle, minimal number of layers where frontiers stop marching at first, growth rate of layer thickness or surface mesh size, total number of layers, and so on. Although sufficient iterations on smoothing marching vectors can alleviate or even avoid this problem, the time cost is expensive and it is not always effective.

6.2 Algorithm

MSM can approximately split the solid space into several subdomains bounded by a continuous convex boundary surface and part of the MSM. The walking rays from a subdomain are hard to intersect with those from the neighboring subdomains as long as they does not cross the MSM. EJW-B performs a space detection in a DCDT with the MSM as the walking boundary. However, if the frontiers tested by EJW-B are not filtered, the number of layers produced is less near curved surfaces as the frontiers will hit the MSM quickly due to its noisy representation in sliver elements. One way to balance robustness and quality is to take an extra test just for cliff points, which as the starting points for Ray Tracing are far away from curved surfaces after the smoothing of marhcing vectors.

In the preprocessing, front point p is a 0th-layer point and q is the target point far away from p. The maximal number of marching layers of p can be estimated by EJW-A with a safe factor r_{gap}, which takes 0.3 to ensure clear distance between boundary layer meshes as shown in Fig. 7a. At the ith layer, p_0 is the 0th-layer point and q is the target point $(d_i + d_{\text{ave}})$ far away from p, where d_i is the marching distance and d_{ave} is the average surface mesh size at p_0 as a safe threshold. If the corresponding $(i + 1)$th-layer point of p is a cliff point, EJW-B is carried out to check whether the cliff point is a risky point with a two-walk scheme. The first walk

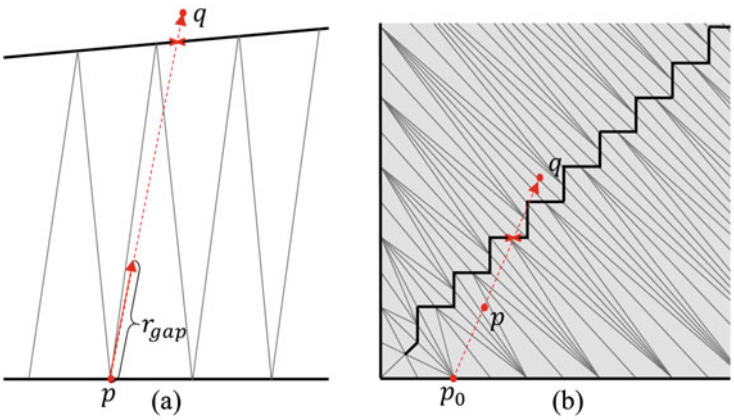


Fig. 7 Schematic of the enhanced Jump-and-Walk method: (**a**) EJW-A and (**b**) EJW-B

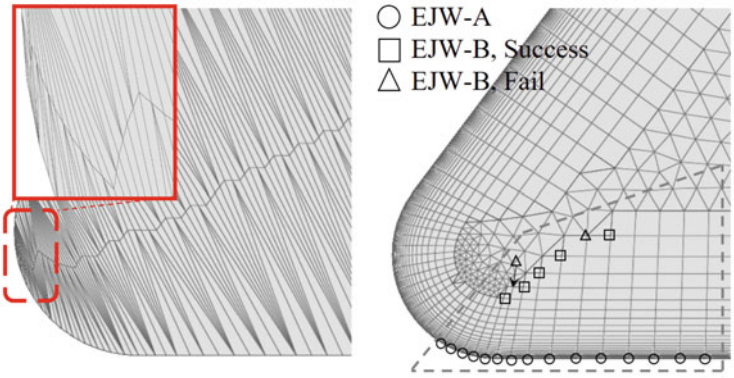


Fig. 8 DCDT and boundary layer meshes generated based on it

from p_0 to p is to determine the tetrahedron where p is located and the second one from p to q is to evaluate whether the advance of p is safe as shown in Fig. 7b. p can march to the next layer only if i is less than the maximal number of marching layers and the ray **pq** does not intersect with the MSM.

If p stops marching, the $(i + 1)$th-layer points of its neighbors will become cliff points. Therefore, its neighbors will also be tested until all points pass this test. If p_0 is a feature point, then p will not be checked because the MSM will extends to the feature edge. As shown in Fig. 8, a reasonable number of boundary layers are generated near the curved boundary.

6.3 Time Complexity

Let n be the number of front points on boundary, EJW-A will handle n points while EJW-B just handle εn points. ε is smaller than 1 as only the top-layer front points in narrow area may be tested by EJW-B like the ones marked in the dotted box of Fig. 8. Totally the collision detection operation will be called $(1 + \varepsilon)n$ times. As JWM requires the expected time close to $O\left(n^{1/4}\right)$ per point, the time complexity of our algorithm is $O\left(n^{5/4}\right)$.

7 Generation of Isotropic Meshes

With the closed inflated boundary as input, isotropic tetrahedral meshes can be generated in the remaining flow field. There are many efficient, stable and open-source tetrahedral mesh generators, such as Gmsh, NetGen, Tetgen, etc. In this work, the generation of CDT and isotropic tetrahedral meshes is completed via TetGen program.

8 Work Examples and Industrial Applications

In this section, we present three representative cases to discuss the performance of our algorithm. 3D-cavity is a model constructed by simplifying and extruding the 2D-cavity below the combustion chamber of a gas turbine engine without any curved feature for a basic performance test; Impingement cooling channel (ICC) is a common internal flow model containing periodic surfaces and smooth corners while F6 is a complex external flow aircraft. All cases were run in a single core of an Intel Core i7-8850H CPU with 16 GB of RAM.

8.1 Hybrid Mesh

Table 1 shows the statistics of surface mesh inputted and hybrid mesh generated by our algorithm for the models. In this work, the input is required to be a closed trianglar mesh and the boundary layer meshes generated consist of prisms, but quadral and hexahedral elements also can be supported simply by a splitting of quads before the input and a merging of prisms after the output.

For the ease of examination, the surface meshes in narrow areas were marked red while the rest were green. As shown in Fig. 9 (left), the locus of MSM is consistent with actual medial surface on the whole and geometric features of 3D-cavity have been identified. From the cut-up view of hybrid mesh and the local

Table 1 Data statistics of hybrid mesh

Model	Algorithm	# nodes	# tri.	# layers	# tet.	# pyramids	# prisms
Cav	Ours	381 k	762 k	26	3.4 M	62 k	13.0 M
	Pointwise	381 k	762 k	26	9.4 M	76 k	13.8 M
ICC	Ours	152 k	304 k	26	3.7 M	37 k	6.0 M
	Pointwise	152 k	304 k	26	2.8 M	42 k	6.3 M
F6	Ours	150 k	299 k	35	2.4 M	53 k	8.9 M
	Pointwise	150 k	299 k	35	1.9 M	49 k	9.3 M

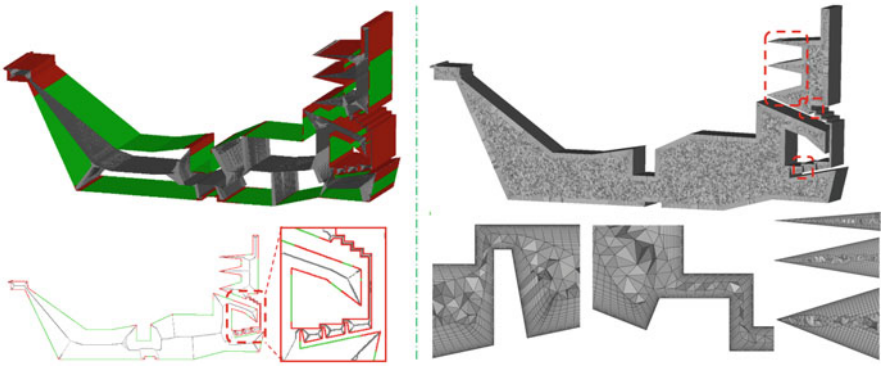


Fig. 9 MSM and hybrid mesh of 3D-cavity

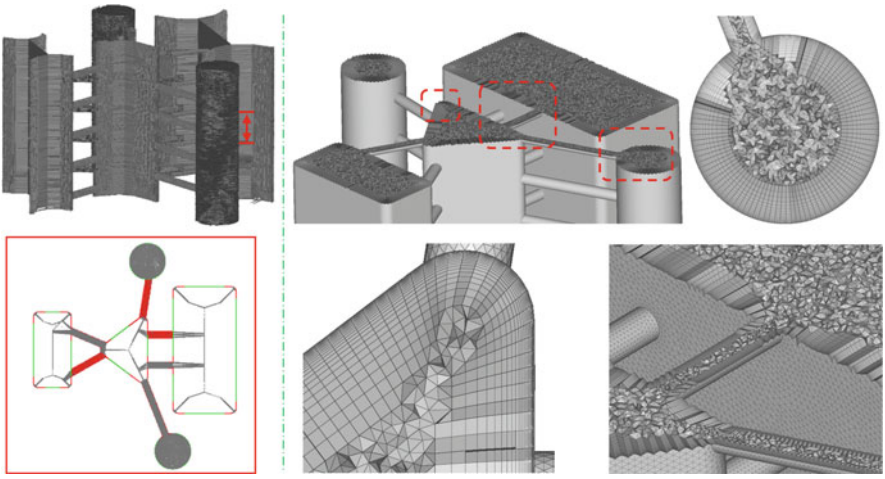


Fig. 10 MSM and hybrid mesh of ICC

enlarged pictures in Fig. 9 (right), both global and local grid intersections at narrow channels and corners are effectively avoided with our algorithm. In the same way, Fig. 10 shows the MSM and hybrid mesh of ICC. Although part of the MSM is chaotic around periodic surfaces and curved corners, reasonable boundary layer

meshes have been generated with enough layers. It indicates the drawback of MSM can be overcome. Moreover, the MSM is continuous and almost leak-free, which improves the robustness of our algorithm. Figure 11 (left) gives a cut-up view of the DCDT outside F6. The locus of MSM is visible from the enlarged pictures with the branches in the big tetrahedra of CDT. Figure 11 (right) shows the front cut-up views of hybrid mesh outside F6. The thickness of boundary layers increases gradually from the corners to the outside and a clear spacing between prismatic meshes is always maintained.

To demonstrate the quality of the boundary layer meshes generated by the proposed method, a comparison was made with a popular commercial software Pointwise, which contains a T-Rex technique that anisotropic tetrahedral elements are generated first around the model, and then boundary layer meshes are formed by combining three tetrahedral elements into a prism [22]. The determination of self-intersection in the front surface mesh during continuous local mesh modification is based on an intersection test accelerated by an axis-aligned binary space partitioned tree. With the same surface mesh of the models as the input, the hybrid mesh generated by Pointwise are shown in Table 1, and the quality distributions of prismatic elements are plotted in Fig. 12 with the scaled aspect ratio, where the ratio in the range of 0.9–1.0 has been cut by 70 for the ease of examination. As our algorithm avoids grid collisions based on space detection by maintaining a safe

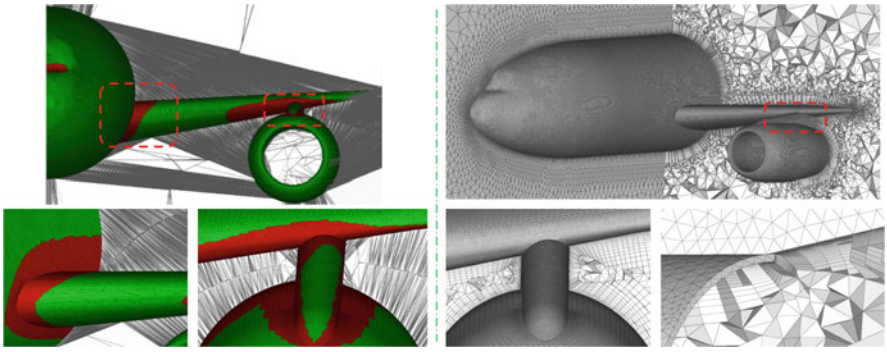


Fig. 11 MSM and hybrid mesh of F6

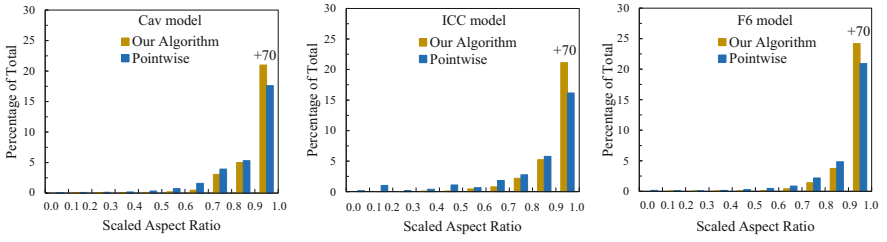


Fig. 12 Quality distributions of the prismatic elements

space between adjacent meshes, the total number of prisms is less than Pointwise. However, a more desirable distribution can be observed in the result produced by our algorithm.

8.2 Comparison of JWM and EJW

Table 2 shows the time of background grid generation and Ray Tracing (RT) and the results of intersection tests by JWM and EJW. JWM or EJW-A performs the RT in a CDT while EJW-B is based on a DCDT. The constructions of the CDT and DCDT consumed about the same amount of time according to the statistics. The RT in EJW-B took less than 10 s to detect the collisions of boundary layers meshes in the millions while that in EJW-A took about 5 times as long. For each of front points tested we count the number of tetrahedra visited in a RT and take the mean M_n. Figure 13 plots the ratio $M_n/n^{1/4}$ with a front point set of size m tested by EJW-A and EJW-B in a triangulation of n points for the 3D-cavity model. Both of the ratio values are less than 1.5 close to the result produced by Mücke et al. [11]. As EJW-A tested all front points on the boundary, totally about 1.2 layers of front points were handled by the proposed method.

When the value of n reaches 381 k, JWM fails the intersection test for the 3D-cavity model. The grid also intersects for the ICC and F6 if JWM is adopted. Figure 14 shows the intersected or almost intersected meshes generated by JWM, while these problems are effectively alleviated by EJW at the same positions. It indicates that our algorithm can produce a valid result under different conditions. Moreover, the construction of DCDT and the ray tracing operations can run in parallel without large modification to the program.

Table 2 Time statistics of JWM and EJW

				JWM			EJW				
Model	# nodes	# tri.	# layers	CDT(s)	RT(s)	T/F	CDT(s)	RTA(s)	DCDT(s)	RTB(s)	T/F
Cav	24 k	47 k	26	0.977	0.612	T	0.952	0.593	1.212	0.153	T
	47 k	94 k	26	1.736	1.424	T	1.762	1.417	2.263	0.479	T
	91 k	183 k	26	3.444	3.872	T	3.402	3.767	4.528	0.753	T
	175 k	351 k	26	6.463	9.180	T	6.56	9.538	8.283	1.545	T
	381 k	762 k	26	14.959	29.904	F	16.498	30.214	19.219	5.936	T
	562 k	1125 k	26	26.838	55.051	F	26.909	55.855	30.099	9.092	T
ICC	152 k	304 k	26	11.203	22.970	F	10.935	21.911	12.949	2.93	T
F6	150 k	299 k	35	13.467	58.959	F	13.574	58.208	13.344	6.953	T

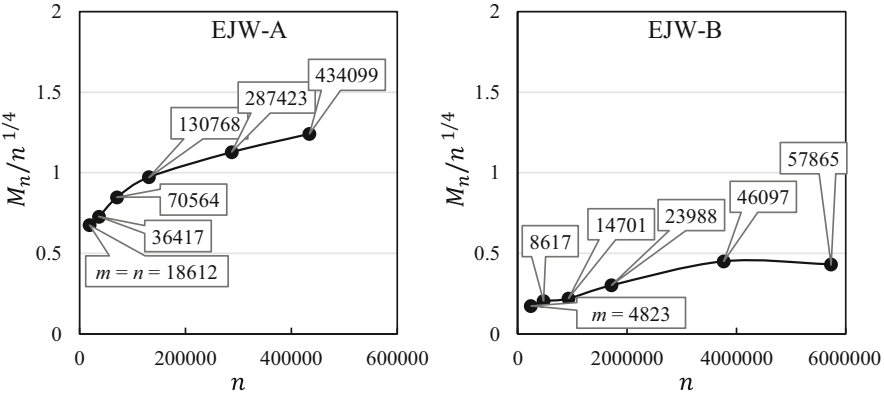


Fig. 13 The ratio $M_n/n^{1/4}$ by EJW-A and EJW-B for Cav. M_n is the mean of the number of tetrahedra visited for a front point set of size m tested in a triangulation of n points

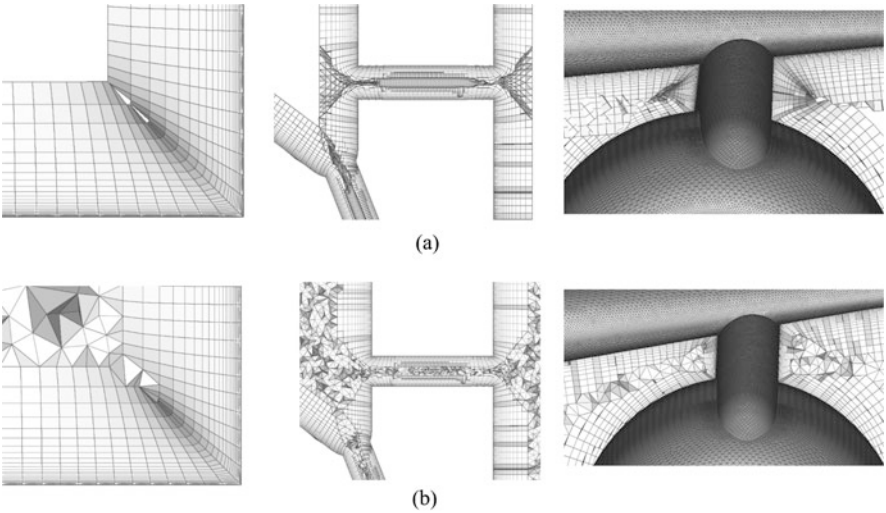


Fig. 14 Comparison of boundary layer meshes: (**a**) JWM and (**b**) EJW

9 Conclusions and Discussions

In this work, the JWM has been developed, which relies only on CDT without other extra data structures. By introducing a wall of medial surface into the background grid, the intersections of boundary layer meshes around complex geometrical features can be much alleviated. Several complex geometries with a lot of special features were tackled to test the effectiveness of the proposed method.

This paper also has proposed a novel method to quickly extract a medial surface from the CDT. As the construction of medial surface is a significant topic, further researches can be conducted to improve the approximate accuracy.

References

1. Kallinderis, Y., Ward, S.: Prismatic grid generation for three-dimensional complex geometries. AIAA J. **31**, 1850–1856 (1993)
2. Kallinderis, Y., Khawaja, A., McMorris, H.: Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. AIAA J. **34**, 291–298 (1996)
3. Zheng, Y., Xiao, Z., Chen, J., et al.: Novel methodology for viscous-layer meshing by the boundary element method. AIAA J. **56**, 209–221 (2018)
4. Si, H.: TetGen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. **41**, 11 (2015)
5. Lo, S.H.: 3D Delaunay triangulation of non-uniform point distributions. Finite Elem. Anal. Des. **90**, 113–130 (2014)
6. Yu, F., Zeng, Y., Guan, Z.Q., et al.: A robust Delaunay-AFT based parallel method for the generation of large-scale fully constrained meshes. Comput. Struct. **228**, 106170 (2020)
7. Loehner, R.: Matching semi-structured and unstructured grids for Navier-Stokes calculations. In: AIAA 93-3348-CP, pp. 555–564 (1993)
8. Pirzadeh, S.: Unstructured viscous grid generation by the advancing-layers method. AIAA J. **32**, 1735–1737 (1994)
9. Wang, F., Mare, L.D.: Hybrid meshing using constrained Delaunay triangulation for viscous flow simulations. Int. J. Numer. Methods Eng. **108**, 1667–1685 (2016)
10. Borouchaki, H., Lo, S.H.: Fast Delaunay triangulation in three dimensions. Comput. Methods Appl. Mech. Eng. **128**, 153–167 (1995)
11. Mücke, E.P., Saias, I., Zhu, B.: Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations, Comput. Geom. **12**, 63–83 (1999)
12. Shan, J.L., Li, Y.M., Guo, Y.Q., et al.: A robust backward search method based on walk-through for point location on a 3D surface mesh. Int. J. Numer. Methods Eng. **73**, 1061–1076 (2008)
13. Tomac, M., Eller, D.: Towards automated hybrid-prismatic mesh generation. Procedia Eng. **82**, 377–389 (2014)
14. Wang, Z., Quintanal, J., Corral, R.: Accelerating advancing layer viscous mesh generation for 3D complex configurations. Procedia Eng. **112**, 35–46 (2019)
15. Bonet, J., Peraire, J.: An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. Int. J. Numer. Methods Eng. **31**, 1–17 (1991)
16. Dyedov, V., Einstein, D.R., Jiao, X., et al.: Variational generation of prismatic boundary-Layer meshes for biomedical computing. Int. J. Numer. Methods Eng. **79**, 907–945 (2009)
17. Alauzet, F., Marcum, D.: A closed advancing-layer method with changing topology mesh movement for viscous mesh generation. In: Proceedings of the 22th International Meshing Roundtable. Springer, Cham (2014)
18. Hubbard, P.M.: Approximating polyhedra with spheres for time-critical collision detection. ACM Trans. Graph. **15**, 179–210 (1996)
19. Quadros, W.R., Shimada, K., Owen, S.J.: Skeleton-based computational method for the generation of a 3D finite element mesh sizing function. Eng. Comput. **20**, 249–264 (2004)
20. Fogg, H.J., Armstrong, C.G., Robinson, T.T.: New techniques for enhanced medial axis based decompositions in 2-D. Procedia Eng. **82**, 162–174 (2014)

21. Price, M.A., Armstrong, C.G., Sabin, M.A.: Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges. Int. J. Numer. Methods Eng. **38**, 3335–3359 (1995)
22. Steinbrenner, J.P., Abelanet, J.P.: Anisotropic tetrahedral meshing based on surface deformation techniques. In: 45th AIAA Aerospace Sciences Meeting and Exhibit, pp. 554. AIAA, Reno (2007)

Part IV
Numerical Geometry and Applications

An Improved Algorithm for Scattered Data Interpolation Using Quartic Triangular Bézier Surfaces

Krassimira Vlachkova

Abstract We revisit the problem of interpolation of scattered data in $\mathbb{R}^3$ and propose a solution based on Nielson's minimum norm network and triangular Bézier patches. We aimed at solving the problem using the least number of polynomial patches of the smallest possible degree. We propose an alternative to the previously known algorithms, see Clough and Tocher (Finite element stiffness matrices for analysis of plate bending. In: Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics, vol. 66–80, pp. 515–545. Wright-Patterson A. F. B., Ohio, 1965) and Shirman and Séquin (Comput Aided Geom Des 4:279–295, 1987; 8:217–221, 1991). Although conceptually similar, our algorithm differs from the previous in all its steps. As a result the complexity of the resulting surface is reduced and its smoothness is improved. We present results of our numerical experiments.

1 Introduction

Scattered data interpolation is an important problem in Computer Aided Geometric Design and finds applications in various areas including automotive, aircraft and ship design, architecture, archeology, computer graphics, biomedical informatics, scientific visualization, and others. In general the problem can be formulated as follows: Given a set of points $\mathbf{d}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i = 1, \ldots, N$, find a bivariate function $F(x, y)$ defined in a certain domain D containing points $\mathbf{v}_i = (x_i, y_i)$, such that F possesses continuous partial derivatives up to a given order and $F(x_i, y_i) = z_i$.

K. Vlachkova (✉)
Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
e-mail: krassivl@fmi.uni-sofia.bg

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_21

Various methods for solving this problem were proposed and applied, see [10, 11, 13, 14]. Further information and treatment more recently have been presented in [1–3, 7]. A standard approach to solve the problem consists of two steps:

Step 1. Construct a triangulation $T = T(\mathbf{v}_1, \ldots, \mathbf{v}_N)$;
Step 2. For every triangle in T construct a surface (patch) which interpolates the data at the three vertices of T.

The interpolation surface constructed in Step 2 is usually polynomial or piecewise polynomial. Typically, the patches are computed with a priori prescribed normal vectors at the data points. G^1 or G^2-smoothness of the resulting surface is achieved either by increasing the degree of the patches, or by the so called *splitting* in which for each triangle in T a macro-patch consisting of a fixed number of Bézier sub-patches is constructed. Splitting was originally proposed by Clough and Tocher [6] and further developed by Percell [16] and Farin [8] for solving different problems. Known splitting algorithms apply splitting to all macro-patches defined in D. We point out that splitting decreases the smoothness of a polynomial macro-patch to G^1.

Shirman and Séquin [18, 20] construct a G^1-smooth surface consisting of quartic triangular Bézier surfaces. Their method assumes that the normal vectors at points $\mathbf{d}_i$, $i = 1, \ldots, N$, are given as part of the input. Shirman and Séquin construct a smooth cubic curve network defined on the edges of T, first, and then degree elevate it to quartic. This increases the degrees of freedom and allows them to connect smoothly the adjacent Bézier patches. Next, they apply splitting where for each triangle in T a macro-patch consisting of three quartic Bézier sub-patches is constructed. To compute the inner Bézier control points closest to the boundary of the macro-patch, Shirman and Séquin use a method proposed by Chiyokura and Kimura [4, 5]. The interpolation surfaces constructed by Shirman and Séquin's algorithm often suffer from unwanted bulges, tilts, and shears as pointed out by the authors in [19] and more recently by Hettinga and Kosinka in [12].

Nielson [15] proposes a method which computes a smooth interpolation curve network defined on the edges of T so as to have common tangent planes at the data points and to satisfy an extremal property. This curve network is called *minimum norm network* (MNN) and is cubic. Nielson extends the MNN to a smooth interpolation surface using a *blending* method based on convex combination schemes. The interpolant obtained is a rational function on every triangle in T.

In this paper we aim at constructing an interpolation surface that is piecewise polynomial and consists of the least number of patches of the smallest possible degree. Such interpolants are computationally tractable and desired in practice. We propose an algorithm that constructs interpolants consisting of quartic triangular Bézier patches and improves on Shirman and Séquin's method in various ways. More precisely, our contributions are as follows.

(i) We use the MNN and then degree elevate it to quartic. An important advantage of this approach is that the MNN is obtained through a global optimization of the curve networks defined on the edges of T which improves the shape of the

patches. Another advantage is that the normal vectors at the data points are obtained through the computation of the MNN.

(ii) Our algorithm does not necessarily apply splitting in all triangles of T. We identify a subset of triangles $T_s \subset T$ where splitting needs to be applied. Finding the minimum size set T_s is computationally hard, and instead, we adopt an efficient incremental heuristic, where at each step the best candidate triangle is added to T_s. Decreasing the size of T_s reduces the complexity of the resulting surfaces and improves their quality.

(iii) We compute the control points of the Bézier patches so that the oscillations of the resulting interpolation surface near edges and vertices are minimized and distortions and twists are avoided.

(iv) Shirman and Séquin impose an additional condition that the three quartic curves defined on the common edges of the three sub-patches must be degree elevated cubic curves. This condition is not necessary to obtain G^1-continuity across the common edges of the sub-patches. We use different condition for the inner points of the sub-patches and believe that such a choice would facilitate the construction of convex macro-patches.

The remainder of this paper is organized as follows. In Sect. 2 we present some preliminary results. Our algorithms are presented in Sect. 3. In the final Sect. 4 examples of some of our numerical experiments are presented and compared with surfaces generated by Shirman and Séquin's method.

2 Preliminaries

2.1 Nielson's MNN

Let $N \geq 3$ be an integer and $\mathbf{d}_i := (x_i, y_i, z_i)$, $i = 1, \ldots, N$, be different points in $\mathbb{R}^3$. We call this set of points *data*. The data are *scattered*[1] if the projections $\mathbf{v}_i := (x_i, y_i)$ onto the plane Oxy are different and non-collinear. Hereafter we assume that a triangulation T of the points $\mathbf{v}_i$, $i = 1, \ldots, N$, is given and fixed. Furthermore, for the sake of simplicity, we assume that the domain D formed by the union of the triangles in T is simply connected. The set of the edges in T is denoted by E. If there is an edge between $\mathbf{v}_i$ and $\mathbf{v}_j$ in E, it will be referred to by e_{ij} or simply by e if no ambiguity arises. A *curve network* is a collection of real-valued univariate functions $\{f_e\}_{e \in E}$ defined on the edges in E. With any real-valued

[1] Note that this definition of scattered data slightly misuses the commonly accepted meaning of the term. It allows data with some structure among points $\mathbf{v}_i$. We have opted to do this in order to cover all cases where our presentation and results are valid.

bivariate function F defined on D we naturally associate the curve network defined as the restriction of F on the edges in E, i.e., for $e = e_{ij} \in E$,

$$f_e(t) := F\left(\left(1 - \frac{t}{\|e\|}\right)x_i + \frac{t}{\|e\|}\,x_j,\ \left(1 - \frac{t}{\|e\|}\right)y_i + \frac{t}{\|e\|}\,y_j\right), \quad \text{where} \quad 0 \le t \le \|e\| \quad \text{and} \quad \|e\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{1}$$

Furthermore, according to the context F will denote either a real-valued bivariate function or a curve network defined by (1). We introduce the following class of *smooth interpolants* defined on D

$$\mathcal{F} := \left\{ F(x, y) \in C^1(D) \mid F(x_i, y_i) = z_i, i = 1, \ldots, N,\ f'_e \in AC,\ f''_e \in L^2,\ e \in E \right\},$$

where $C^1(D)$ is the class of bivariate functions defined in D which possess continuous first order partial derivatives, AC is the class of univariate absolutely continuous functions defined in $[0, \|e\|]$, and L^2 is the class of univariate functions defined in $[0, \|e\|]$ whose second power is Lebesgue integrable.

The restrictions on E of the functions in $\mathcal{F}_p$ form the corresponding class of so-called *smooth interpolation curve networks*

$$C(E) := \left\{ F_{|E} = \{f_e\}_{e \in E} \mid F(x, y) \in \mathcal{F} \right\}.$$

The smoothness of the interpolation curve network $F \in C(E)$ geometrically means that at each point $\mathbf{d}_i$ there is a *tangent plane* to F, where a plane is *tangent* to the curve network at the point $\mathbf{d}_i$ if it contains the tangent vectors at $\mathbf{d}_i$ of the curves incident to $\mathbf{d}_i$.

For $F \in C(E)$ we denote the curve network of second derivatives of F by $F'' := \{f''_e\}_{e \in E}$. The L^2-norm of F'' is defined by

$$\left\| F'' \right\|_{L^2(T)} := \left\| F'' \right\| = \left(\sum_{e \in E} \int_0^{\|e\|} \left| f''_e(t) \right|^2 \mathrm{d}t \right)^{1/2}.$$

Nielson [15] considered and solved the following extremal problem

$$(\mathbf{P}) \quad \textit{Find } F^* \in C(E) \textit{ such that } \left\| F^{*\prime\prime} \right\| = \inf_{F \in C(E)} \left\| F'' \right\|.$$

The unique solution (MNN) to (**P**) is a cubic curve network and is obtained by solving a linear system of equations.

2.2 The G^1-Continuity Conditions

Let C_1 and C_2 be cubic triangular Bézier patches whose common boundary is the cubic curve $q(t)$. Let $q(t) = \sum_{i=0}^{3} \mathbf{q}_i B_i^3(t)$, where $\mathbf{q}_i$, $i = 0, \ldots, 3$, be the control points of $q(t)$, and $B_i^m(t)$ be the Bernstein polynomials of degree m, $m \in \mathbb{N}$, defined for $0 \le t \le 1$ by

$$B_i^m(t) := \binom{m}{i} t^i (1-t)^{m-i}, \qquad \binom{m}{i} = \begin{cases} \frac{m!}{i!(m-i)!}, & \text{for } i = 0, \ldots, m, \\ 0, & \text{otherwise.} \end{cases}$$

Let us degree elevate C_1 and C_2 to quartic Bézier patches and denote the control points of the degree elevated $q(t)$ by $\hat{\mathbf{q}}_i$, $i = 0, \ldots, 4$, where $\hat{\mathbf{q}}_0 \equiv \mathbf{q}_0$ and $\hat{\mathbf{q}}_4 \equiv \mathbf{q}_3$. Then $q(t) = \sum_{i=0}^{4} \hat{\mathbf{q}}_i B_i^4(t)$ where $\hat{\mathbf{q}}_i = \frac{i}{4}\mathbf{q}_{i-1} + \left(1 - \frac{i}{4}\right)\mathbf{q}_i$, $i = 0, \ldots, 4$.

Let $\mathbf{p}_i$ and $\mathbf{r}_i$, $i = 0, \ldots, 3$, be the nearest to $q(t)$ control points of C_1 and C_2, respectively, see Fig. 1 (left). Farin [9, pp. 368–371] proposed the following sufficient conditions for G^1-continuity between C_1 and C_2.

$$\frac{i}{4}\mathbf{b}_{i,4} + \left(1 - \frac{i}{4}\right)\mathbf{b}_{i,0} = 0, \quad i = 0, \ldots, 4, \tag{2}$$

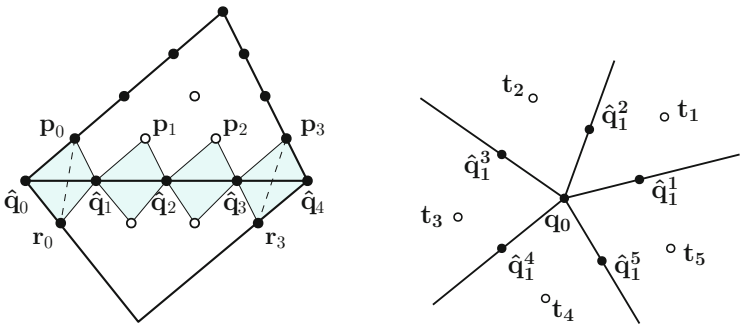


Fig. 1 Left: the G^1-continuity conditions: the four shaded quadrilaterals are planar. Right: the vertex enclosure problem for an odd degree vertex always has a solution

where

$$\mathbf{b}_{i,0} = \alpha_0 \mathbf{p}_i + (1-\alpha_0)\mathbf{r}_i - \big(\beta_0 \hat{\mathbf{q}}_i + (1-\beta_0)\hat{\mathbf{q}}_{i+1}\big),$$
$$\mathbf{b}_{i,4} = \alpha_1 \mathbf{p}_{i-1} + (1-\alpha_1)\mathbf{r}_{i-1} - \big(\beta_1 \hat{\mathbf{q}}_{i-1} + (1-\beta_1)\hat{\mathbf{q}}_i\big),$$

and $0 < \alpha_i < 1$, $i = 1, 2$. From (2) for $i = 0$ and $i = 4$ we obtain

$$\mathbf{b}_{0,0} = 0 \Rightarrow \alpha_0 \mathbf{p}_0 + (1-\alpha_0)\mathbf{r}_0 = \beta_0 \hat{\mathbf{q}}_0 + (1-\beta_0)\hat{\mathbf{q}}_1, \tag{3}$$
$$\mathbf{b}_{4,4} = 0 \Rightarrow \alpha_1 \mathbf{p}_3 + (1-\alpha_1)\mathbf{r}_3 = \beta_1 \hat{\mathbf{q}}_3 + (1-\beta_1)\hat{\mathbf{q}}_4. \tag{4}$$

The geometric meaning of (2) is that the four shaded quadrilaterals in Fig. 1 (left) are planar. Moreover, α_i, β_i, $i = 1, 2$, are uniquely determined by the intersection point of the diagonals of the first and the last quadrilaterals, respectively.

2.3 The Vertex Enclosure Problem

Let $\mathbf{v}$ be an inner vertex in T, i.e., $\mathbf{v} \equiv \mathbf{v}_i$ for some i, $1 \le i \le N$, and $\mathbf{q}_0$ be the corresponding point $\mathbf{d}_i$. Let n be the *degree* of $\mathbf{v}$, i.e., the number of the edges incident to $\mathbf{v}$. Let τ_k, $k = 1, \ldots, n$, be the triangles in T with common vertex $\mathbf{v}$ listed in counterclockwise order around $\mathbf{v}$, where τ_1 is arbitrarily chosen. For $k = 1, \ldots, n$, let $\mathcal{T}_k$ be the quartic Bézier patch defined in τ_k, and $q^k(t) = \sum_{i=0}^{4} \hat{\mathbf{q}}_i^k B_i^4(t)$ be the degree elevated cubic curves of the MNN with starting point $\mathbf{q}_0$ defined on the edges of τ_k. We denote by α_i^k, β_i^k, $i = 0, 1$, the corresponding coefficients for $q^k(t)$ defined by (3) and (4). Let $\mathbf{t}_k$ be the nearest to $\mathbf{q}_0$ inner control point of $\mathcal{T}_k$, $k = 1, \ldots, n$, see Fig. 1 (right) for $n = 5$.

By applying (3) to $\mathcal{T}_k$, $k = 1, \ldots, n$, we obtain the following linear system for the unknowns $\mathbf{t}_k$,

$$\begin{pmatrix} 1-\alpha_0^2 & \alpha_0^2 & 0 & \cdots & 0 & 0 \\ 0 & 1-\alpha_0^3 & \alpha_0^3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1-\alpha_0^n & \alpha_0^n \\ \alpha_0^1 & 0 & 0 & \cdots & 0 & 1-\alpha_0^1 \end{pmatrix} \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_{n-1} \\ \mathbf{t}_n \end{pmatrix} = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_{n-1} \\ \mathbf{s}_n \end{pmatrix}, \tag{5}$$

where the points $\mathbf{s}_k$ depend on α_i^k, β_i^k, $\hat{\mathbf{q}}_i^k$, $k = 1, \ldots, n$; $i = 1, 2$.

The existence of a solution to system (5) is known as the *vertex enclosure problem*. Peters [17] proved that the rank of (5) in n for odd n, and $n - 1$ for even n. Therefore, if n is odd then system (5) always has a unique solution. If n is

even then by Gauss elimination we obtain that (5) has a solution if and only if

$$\mathbf{s}_n\left(1-\alpha_0^2\right)\ldots\left(1-\alpha_0^n\right)-\mathbf{s}_{n-1}\alpha_0^1\ldots\alpha_0^{n-1} + \sum_{k=1}^{n-2}(-1)^k\mathbf{s}_k\alpha_0^1\ldots\alpha_0^k\left(1-\alpha_0^{k+2}\right)\ldots\left(1-\alpha_0^n\right)=0. \tag{6}$$

3 Outline of Our Algorithms

In this section we propose algorithms to construct G^1-continuous surface consisting of Bézier patches in the presence of even degree vertices. In [17] it is shown that although splitting does not eliminate even degree vertices, the vertex enclosure problem can be solved successfully since splitting adds more degrees of freedom. However, it is not necessary to split all triangles in T. In case the vertex is of even degree and (6) doesn't hold, it suffices to split just one triangle incident to the vertex. Our goal is to reduce the number of the splitted triangles since splitting decrease the smoothness of the macro-patch to G^1.

3.1 Determining the Set T_s

First, using degree elevation we represent all cubic curves comprising the MNN as quartic Bézier curves. Next, we identify an optimized set T_s of triangles where splitting needs to be applied. For the remaining triangles, those in $T \setminus T_s$, we simply construct the standard quartic Bézier patch.

Let V_1 be the set of all inner vertices in T. First, for each edge e incident to V_1 for which the corresponding coefficients $\alpha_0(e)$ and $\alpha_1(e)$ defined by (3) and (4) are different, we insert the triangles incident to e in T_s and mark the endpoint vertices of e. Next, we identify a set $V_2 \subset V_1$ consisting of vertices that:

(i) are not marked;
(ii) have even degree;
(iii) Equation (6) does not hold.

Let T_2 be the set of triangles incident to vertices in V_2. We assign weights w, $w = 1, 2$, or 3, to triangles in T_2 equal to the number of vertices in T_2 incident to the corresponding triangle. We organize T_2 in a priority queue with respect to the weights. While T_2 is nonempty we insert a triangle with maximum weight in T_s, remove its vertices from V_2, and update the weights and the queue.

Algorithm 1 below takes the MNN as an input and construct a G^1-continuous interpolating surface $F(x, y)$ defined on D which consists of triangular quartic Bézier patches.

Algorithm 1

Step 1. Compute the control points of the curves in the MNN
Step 2. Degree elevate all curves to quartic curves.
Step 3. Compute an optimized set T_s as described above.
Step 4. For each $\tau \in T \setminus T_s$ compute 3 inner control points and construct a macro-patch.
Step 5. For each $\tau \in T_s$ compute 19 inner control points and construct three sub-patches using Algorithm 2 for splitting as described in Sect. 3.2.

3.1.1 Solving the Vertex Enclosure Problem

Let $\mathbf{v}$ be an inner vertex of T of degree n and $\mathbf{q}_0$ be the corresponding data point. First, we compute the control points $\mathbf{t}_1, \ldots, \mathbf{t}_n$ which are the nearest to $\mathbf{q}_0$, see Fig. 2. If no triangle with vertex $\mathbf{v}$ belongs to T_s then we compute $\mathbf{t}_1, \ldots, \mathbf{t}_n$ by solving the corresponding system (5), see Fig. 1 (right). Let there be a triangle in T_s with vertex $\mathbf{v}$. In Fig. 2 an inner vertex $\mathbf{v}$ of degree 7 incident to four triangles in T_s is shown. Let $\tau_1, \ldots, \tau_r$ be the triangles with vertex $\mathbf{v}$ listed in counterclockwise order around $\mathbf{v}$, where τ_1 is arbitrarily chosen. Let $\mathbf{t}_k$ be the nearest to $\mathbf{q}_0$ point for $\tau_k, k = 1, \ldots, r$. We divide $\tau_1, \ldots, \tau_r$ into groups of consecutive triangles such that the first and the last triangles in each group are sub-patches, and the intermediate triangles are macro-patches, see Fig. 2 where the groups are four: $\tau_1, \tau_2, \tau_3, \tau_4$; τ_5, τ_6; τ_7, τ_8, τ_9; τ_{10}, τ_{11}. The groups are uniquely defined and their number is equal to the number of the triangles in T_s incident to $\mathbf{v}$. Let $\tau_1, \ldots, \tau_m$ be any of the groups. We compute $\mathbf{t}_1, \ldots, \mathbf{t}_m$ by solving the corresponding system (5). Since τ_1 and τ_m do not have a common edge then the last equation of (5) is excluded. Then we can represent $\mathbf{t}_1, \ldots, \mathbf{t}_m$ as linear functions of $\mathbf{t}_m$ using (5). We aim at minimizing the unwanted oscillations between patches $Q_1, \ldots, Q_m$ using $\mathbf{t}_m$ as a

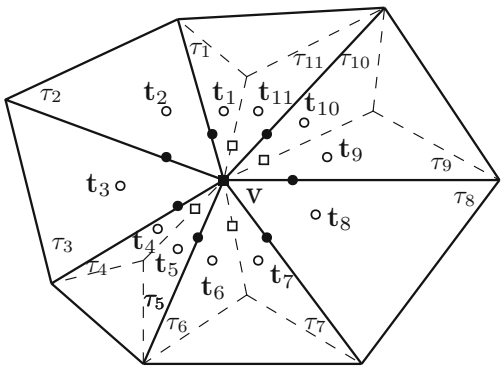


Fig. 2 An inner vertex $\mathbf{v}$ of degree 7 with four incident triangles in T_s. The points $\mathbf{t}_i$ are the nearest control points to the corresponding data point $\mathbf{q}_0, i = 1, \ldots, 11$

shape parameter. We compute its first two coordinates as shown in Sect. 3.2. Let z_i denote the third coordinate of $\mathbf{t}_i$, $i = 1, \ldots, m$. We compute z_m by minimizing the function $g(z_m) = \sum_{i=1}^{m-1} (z_{i+1} - z_i)^2$. The minimum is found readily since $g'(z_m) = 0$ is a linear equation.

In the case where $\mathbf{v}$ is a boundary vertex we compute the corresponding points $\mathbf{t}_1, \ldots, \mathbf{t}_r$ analogously to the case of an inner vertex. In this case τ_1 is uniquely determined and it is possible that either the first group of triangles starts, or the last group ends with a macro-patch instead of a sub-patch.

We note that if a triangle doesn't belong to T_s then by solving the vertex enclosure problem for its three vertices we completely determine the corresponding macro-patch since we obtain its three inner control points.

3.2 *Computing the Control Points of the Sub-patches*

Let τ be a triangle in T_s and $\mathcal{T}$ be the corresponding macro-patch defined in τ. We compute the control points of the three Bézier sub-patches consecutively in four layers as shown in Fig. 3(left). The first layer consists of inner control points that are the nearest to the boundary of τ. The last fourth layer contains a single point $\mathbf{z}$. This point $\mathbf{z}$ is the splitting point and will be computed as a center of the triangle with vertices in the previous third layer. We use the following notation.

- ■ vertices of the sub-patches;
- ● inner control points on the boundary of the macro-patch;
- □ inner control points of the three inner boundary curves of the sub-patches;
- ○ inner control points of the sub-patches.

Next we explain in detail our choice of the control points in the first level. There are three points of type □ and six points of type ○ in this layer, see Fig. 3(left). First, we compute points of type □ as centers of the three small triangles with vertices

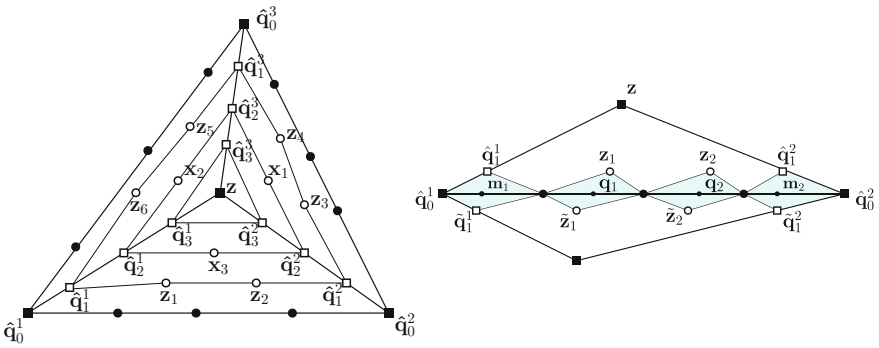


Fig. 3 (left) Construction of a G^1-continuous Bézier macro-patch by splitting to three sub-patches. (right) The control points in the first layer are computed to avoid unwanted twisting, tilting, and oscillations between the adjacent patches

●■● on the boundary of the macro-patch. Then we compute the points $\mathbf{z}_i$, $i = 1, 2$, of type ○ as follows.

Let the corresponding edge of τ be inner for T and $\tilde{\mathbf{q}}_1^i$, $\tilde{\mathbf{z}}_i$, $i = 1, 2$, be the corresponding control points of the neighbouring patch, see Fig. 3(right). We compute consecutively the following points.

$$\mathbf{z}_i' := \hat{\mathbf{q}}_1^1 + \mathbf{q}_i - \mathbf{m}_1, \qquad \tilde{\mathbf{z}}_i' := \tilde{\mathbf{q}}_1^1 + \mathbf{q}_i - \mathbf{m}_1,$$
$$\mathbf{z}_i'' := \hat{\mathbf{q}}_1^2 + \mathbf{q}_i - \mathbf{m}_2, \qquad \tilde{\mathbf{z}}_i'' := \tilde{\mathbf{q}}_1^2 + \mathbf{q}_i - \mathbf{m}_2,$$
$$\mathbf{a}_i := \left(1 - \frac{i}{3}\right)\mathbf{z}_i' + \frac{i}{3}\mathbf{z}_i'', \qquad \tilde{\mathbf{a}}_i := \left(1 - \frac{i}{3}\right)\tilde{\mathbf{z}}_i' + \frac{i}{3}\tilde{\mathbf{z}}_i'', \quad i = 1, 2,$$

where $\mathbf{q}_i$, $i = 1, 2$, are control points of the cubic curve defined on the common edge of τ and its neighbouring triangle, and $\mathbf{m}_i$, $i = 1, 2$, are the intersection points of the diagonals of the quadrilaterals $\hat{\mathbf{q}}_0^1\hat{\mathbf{q}}_1^1\bullet\tilde{\mathbf{q}}_1^1$ and $\bullet\hat{\mathbf{q}}_1^2\hat{\mathbf{q}}_0^2\tilde{\mathbf{q}}_1^2$, respectively, as shown in Fig. 3(right).

Let $\mathbf{z}_i := (\xi_i, \eta_i, \zeta_i)$ and $\tilde{\mathbf{z}}_i := (\tilde{\xi}_i, \tilde{\eta}_i, \tilde{\zeta}_i)$. We choose ξ_i, η_i and $\tilde{\xi}_i$, $\tilde{\eta}_i$ to be equal to the corresponding coordinates of $\mathbf{a}_i$ and $\tilde{\mathbf{a}}_i$, respectively, $i = 1, 2$. In this way the projections of $\mathbf{z}_i$, $i = 1, 2$, onto Oxy lie inside τ. Hence, we avoid unwanted twisting and tilting of the patch. We compute the third coordinates ζ_i, $\tilde{\zeta}_i$ so that $\zeta_i = \tilde{\zeta}_i$ and $\mathbf{z}_i$, $\tilde{\mathbf{z}}_i$, $\mathbf{q}_i$ are collinear, $i = 1, 2$. In this way we avoid unwanted oscillations between the adjacent patches.

In the case where the corresponding edge of τ is boundary for T, i.e., there is no neighbouring patch of $\mathcal{T}$, we compute $\mathbf{z}_i$, $i = 1, 2$ as follows.

$$\mathbf{r}_i' := \hat{\mathbf{q}}_1^1 - \hat{\mathbf{q}}_0^1 + \mathbf{q}_i, \quad \mathbf{r}_i'' := \hat{\mathbf{q}}_1^2 - \hat{\mathbf{q}}_0^2 + \mathbf{q}_i, \quad \mathbf{z}_i := \left(1 - \frac{i}{3}\right)\mathbf{r}_i' + \frac{i}{3}\mathbf{r}_i'', \quad i = 1, 2.$$

The rest of the points $\mathbf{z}_i$, $i = 3, \ldots, 6$, are computed analogously.

Algorithm 2 below takes a triangle τ in T_s and the degree-elevated quartic boundary control points of the corresponding patch and computes 19 control points of the three G^1-continuous quartic Bézier sub-patches.

4 Examples

To demonstrate the results of our work we present here two examples and compare the surfaces generated by our algorithms and Shirman and Séquin's algorithm. For the latter we also use the MNN as input.

Example 1 We consider data obtained from a regular triangular pyramid. We have $N = 4$, $\mathbf{v}_1 = (-1/2, -\sqrt{3}/6)$, $\mathbf{v}_2 = (1/2, -\sqrt{3}/6)$, $\mathbf{v}_3 = (0, \sqrt{3}/3)$, $\mathbf{v}_4 = (0, 0)$, and $z_i = 0$, $i = 1, 2, 3$, $z_4 = -1$. The corresponding MNN is shown in Fig. 4 (left). In this case T_s is empty and no triangle has been splitted. The corresponding

Algorithm 2

Step 1. Compute the control points in the first layer:

1.1 Points of type □ are centers of the three small triangles with vertices ●■●.
1.2 Then points of type ○ are computed as described in Sect. 3.2.

Step 2. Compute the control points in the second layer:

2.1 Points of type □ are centers of the three small triangles with vertices ○□○ in the first layer.
2.2 Then points of type ○ are mid-points of the segments with vertices of type □ in the second layer.

Step 3. Compute the control points in the third layer: the three points of type □ are centers of the small triangles with vertices ○□○ in the second layer.

Step 4. Compute the splitting point of type ■ as a center of the triangle with vertices □ in the third layer.

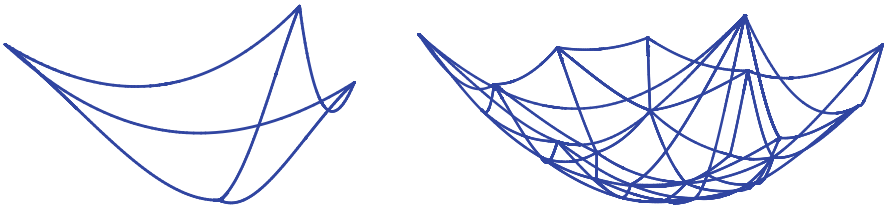


Fig. 4 Left: the MNN for Example 1. Right: the MNN for Example 2

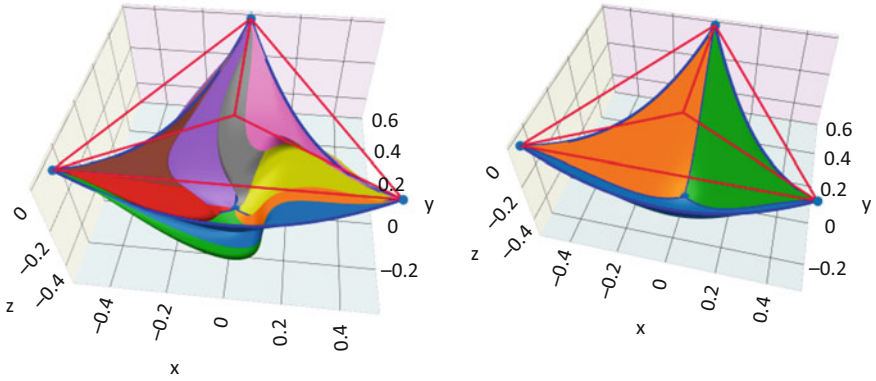


Fig. 5 Comparison of the two surfaces for the data in Example 1. Left: the surface generated using Shirman and Séquin's algorithm. Right: the surface generated using our Algorithms 1 and 2

Shirman and Séquin's surface is shown in Fig. 5 (left). The surface generated by our Algorithms 1 and 2 is shown in Fig. 5 (right).

Example 2 We have $N = 25$ and data sampled from the function $F(x, y) = 5 \times \exp\big((x - 0.5)^2 + (y - 0.5)^2\big)$.

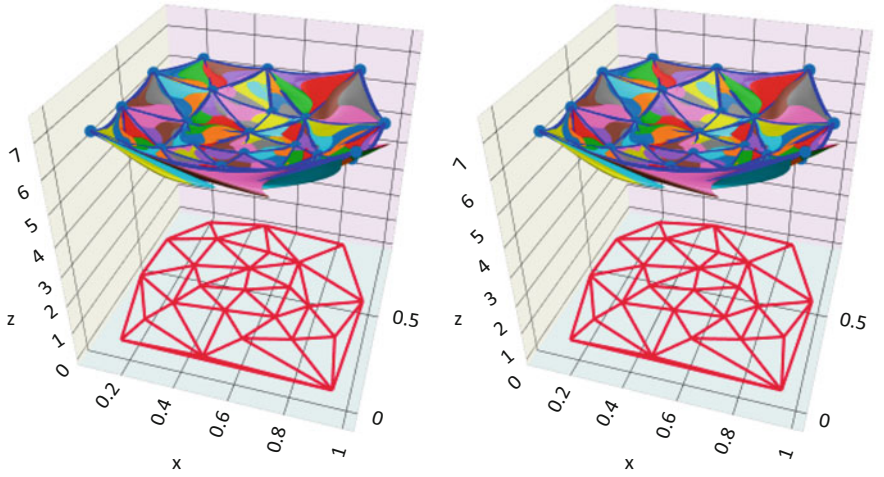


Fig. 6 Comparison of the two surfaces for the data in Example 2. Left: the surface generated using Shirman and Séquin's algorithm. Right: the surface generated using our Algorithms 1 and 2. The set T_s is marked in black and consists of 6 triangles

The triangulation is the Delaunay triangulation, it consists of 41 triangles and is shown in Fig. 6 (left). The corresponding MNN is shown in Fig. 4 (right). The set T_s consists of 6 triangles, see Fig. 6 (right). The corresponding Shirman and Séquin's surface is shown in Fig. 6 (left). The surface generated by our Algorithms 1 and 2 is shown in Fig. 6 (right).

Acknowledgments This research was supported by Sofia University Science Fund Grant No. 80-10-171/2020, and by European Regional Development Fund and the Operational Program "Science and Education for Smart Growth" under Contract № BG05M2OP001-1.001-0004 (2018–2023). The author acknowledges Krum Radev's work and support in implementation and testing of the algorithms.

References

1. Amidror, I.: Scattered data interpolation methods for electronic imaging systems: a survey. J. Electron. Imaging **11**, 157–176 (2002). https://doi.org/10.1117/1.1455013
2. Anjyo, K., Lewis, J., Pighin, F.: Scattered Data Interpolation for Computer Graphics, SIGGRAPH 2014 Course Notes. http://olm.co.jp/rd/research_event/scattered-data-interpolation-for-computer-graphics (2014). Accessed 22 Aug 2020
3. Cazals, F., Giesen, J.: Delaunay triangulation based surface reconstruction. In: Boissonat, J.D., Teillaud, M. (eds.) Effective Computational Geometry for Curves and Surfaces, pp. 231–276. Springer, Berlin (2006). https://doi.org/10.1007/978-3-540-33259-6_6
4. Chiyokura, H.: Localized surface interpolation method for irregular meshes. In: Kunii, T. (ed.) Advanced Computer Graphics, Proceedings of Computer Graphics Tokyo'86, vol. 66–80, pp. 3–19. Springer, Tokyo (1986). https://doi.org/10.1007/978-4-431-68036-9_1

5. Chiyokura, H., Kimura, F.: Design of solids with free-form surfaces. In: Tanner, P.P. (ed.) SIGGRAPH '83 Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques, vol. 17, pp. 289–298. ACM, New York (1983). https://doi.org/10.1145/964967.801160
6. Clough, R., Tocher, J.: Finite element stiffness matrices for analysis of plate bending. In: Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics, vol. 66–80, pp. 515–545. Wright-Patterson A. F. B., Ohio (1965) URL http://contrails.iit.edu/reports/8574
7. Dell'Accio, F., Tommaso, F.D.: Scattered data interpolation by Shepard's like methods: classical results and recent advances. Dolomites Res. Notes Approx. **9**, 32 – 44 (2016). https://doi.org/10.14658/pupj-drna-2016-Special_Issue-5
8. Farin, G.: A modified Clough-Tocher interpolant. Comput. Aided Geom. Des. **2**(1–3), 19–27 (1985). https://doi.org/10.1016/0167-8396(85)90003-2
9. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide, 5th edn. Morgan-Kaufmann, San Francisco (2002). https://doi.org/10.1017/CBO9780511546860
10. Foley, T., Hagen, H.: Advances in scattered data interpolation. Surv. Math. Ind. **4**, 71–84 (1994)
11. Franke, R., Nielson, G.: Scattered data interpolation and applications: a tutorial and survey. In: Hagen, H., Roller, D. (eds.) Geometric Modeling, pp. 131–160. Springer, Berlin (1991). https://doi.org/10.1007/978-3-642-76404-2_6
12. Hettinga, G., Kosinka, J.: Multisided generalisations of Gregory patches. Comput. Aided Geom. Des. **62**, 166–180 (2018). https://doi.org/10.1016/j.cagd.2018.03.005
13. Lodha, S., Franke, K.: Scattered data techniques for surfaces. In: Proceedings of Dagstuhl Conference on Scientific Visualization, pp. 182–222. IEEE Computer Society Press, Washington (1997). https://ieeexplore.ieee.org/document/1423115
14. Mann, S., Loop, C., Lounsbery, M., Meyers, D., Painter, J., DeRose, T., Sloan, K.: A survey of parametric scattered data fitting using triangular interpolants. In: Hagen, H. (ed.) Curve and Surface Design, pp. 145–172. SIAM, Philadelphia (1992). https://doi.org/10.1137/1.9781611971651.ch8
15. Nielson, G.: A method for interpolating scattered data based upon a minimum norm network. Math. Comput. **40**, 253–271 (1983). https://doi.org/10.2307/2007373
16. Percell, P.: On cubic and quartic Clough-Tocher finite elements. SIAM J. Numer. Anal. **13**(1), 100–103 (1976). https://doi.org/10.1137/0713011
17. Peters, J.: Smooth interpolation of a mesh of curves. Constr. Approx. **7**(1), 221–246 (1991). https://doi.org/10.1007/BF01888155
18. Shirman, L., Séquin, C.: Local surface interpolation with Bézier patches. Comput. Aided Geom. Des. **4**(4), 279–295 (1987). https://doi.org/10.1016/0167-8396(87)90003-3
19. Shirman, L., Séquin, C.: Local surface interpolation with shape parameters between adjoining gregory patches. Comput. Aided Geom. Des. **7**(5), 375–388 (1990). https://doi.org/10.1016/0167-8396(90)90001-8
20. Shirman, L., Séquin, C.: Local surface interpolation with Bézier patches: errata and improvements. Comput. Aided Geom. Des. **8**(3), 217–221 (1991). https://doi.org/10.1016/0167-8396(91)90005-V

On Integral-Based (Transfinite) Laplace Coordinates

Alexander G. Belyaev and Pierre-Alain Fayolle

Abstract In this theoretical work, we analyze general constructions for integral-based (transfinite, continuous) barycentric coordinates and consider a simple variational principle to arrive at a continuous version of the Laplace barycentric coordinates. We demonstrate how our approach leads to a general description of the integral-based barycentric coordinates and establish links with Dirichlet energy minimization problems for conical surfaces. Both the 2D and 3D cases are studied. An application to a surface generation problem is briefly considered.

1 Introduction

The present active research on generalized barycentric coordinates was initiated by works of Wachspress [26], Warren [27], and Floater [10] and is currently fuelled by numerous applications of generalized barycentric interpolation schemes in computational mechanics, computer graphics, and geometric modelling. See, for example, [5, 11, 16, 29] and references therein. This paper focuses on the analysis of integral-based Laplace coordinates, a continuous version of a highly popular generalized barycentric interpolation scheme known under the names of Laplace coordinates [3], non-Sibsonian coordinates [1], cotangent weights [22], discrete harmonic coordinates [15], and Voronoi coordinates [18] (see also earlier works [7, 9, 20, 25]). The high popularity of the Laplace coordinates arises from the fact that they approximate the Laplace operator (hence the name) on polygonal meshes [21, 22].

A. G. Belyaev (✉)
Institute of Sensors, Signals and Systems, School of Engineering & Physical Sciences, Heriot-Watt University, Edinburgh, UK
e-mail: a.belyaev@hw.ac.uk

P.-A. Fayolle
Computer Graphics Laboratory, University of Aizu, Aizu-Wakamatsu, Japan
e-mail: fayolle@u-aizu.ac.jp

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_22

Integral-based (transfinite, continuous) barycentric coordinates were originally proposed by Warren et al. [28]. Although at present integral-based barycentric interpolation constitutes an active area of research [2, 4, 6, 8, 12, 13, 17, 19, 23], very little is known about integral-based versions of the Laplace coordinates. In this paper, we consider a simple variational principle for integral-based Laplace barycentric coordinates and show how it leads to a general description of integral-based barycentric coordinates.

One of our main results can be formulated as follows. Let Ω be a strictly convex bounded domain in $\mathbb{R}^N$ with $N = 2$ or $N = 3$. Consider a function $u(\mathbf{y})$ defined for each $\mathbf{y} \in \partial\Omega$. Given $\mathbf{x} \in \Omega$, let us assume that $u(\mathbf{x})$ is known. We fix $\mathbf{x}$ and consider a conical surface $U_{\mathbf{x}}(\mathbf{z})$ generated by straight segments connecting the inner point $(\mathbf{x}, u(\mathbf{x}))$ with the boundary points $(\mathbf{y}, u(\mathbf{y}))$, $\mathbf{y} \in \partial\Omega$. See the left image of Fig. 1 for a visual explanation of how the conical surface $\left\{(\mathbf{z}, U_{\mathbf{x}}(\mathbf{z})) \in \mathbb{R}^{N+1}, \mathbf{z} \in \Omega\right\}$, is constructed.

Let the value $u(\mathbf{x})$ be defined by minimizing Dirichlet's energy

$$\iint_{\Omega} |\nabla U_{\mathbf{x}}|^2 \, \mathrm{d}\mathbf{z} \to \min . \tag{1}$$

Then the above Dirichlet energy minimization interpolation performed for each $\mathbf{x} \in \Omega$ is obtained by integral-based Laplace interpolation

$$u(\mathbf{x}) = \int_{S_{\mathbf{x}}} \frac{u(\mathbf{y}) w(\mathbf{x}, \mathbf{e}_\theta)}{|\mathbf{x} - \mathbf{y}|} \, \mathrm{d}\boldsymbol{\theta} \Bigg/ \int_{S_{\mathbf{x}}} \frac{w(\mathbf{x}, \mathbf{e}_\theta)}{|\mathbf{x} - \mathbf{y}|} \, \mathrm{d}\boldsymbol{\theta} \tag{2}$$

with the weighting function given by

$$w(\mathbf{x}, \boldsymbol{\theta}) = \Delta_S f + (N - 1) f, \quad \text{where} \quad f = |\mathbf{x} - \mathbf{y}|^{N-1}, \tag{3}$$

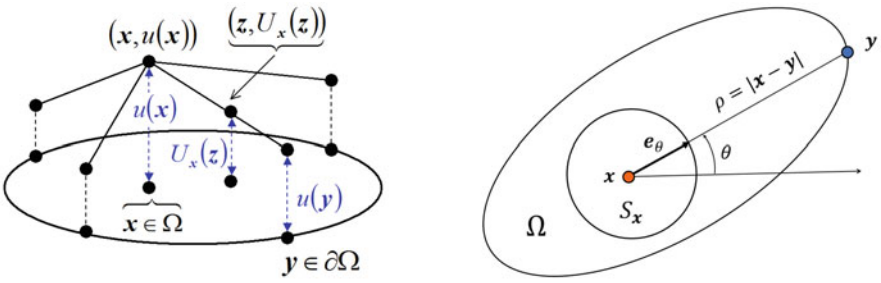


Fig. 1 Left: Point $\mathbf{x} \in \Omega$ parameterizes family of conical surfaces $\left\{(\mathbf{z}, U_{\mathbf{x}}(\mathbf{z})) \in \mathbb{R}^{N+1}, \mathbf{z} \in \Omega \subset \mathbb{R}^N\right\}$, each of which spans $(\mathbf{y}, u(\mathbf{y}))$, $\mathbf{y} \in \partial\Omega$, and has its apex at $(\mathbf{x}, u(\mathbf{x}))$. Right: Notations used to define integral-based barycentric coordinates

and defined up to a multiplicative constant. Here $S_{\mathbf{x}}$ is the unit sphere centered at $\mathbf{x}$, integration is taken with respect to spherical coordinates $\boldsymbol{\theta}$ (point $\boldsymbol{\theta} \in S_{\mathbf{x}}$ corresponds to the unit vector $\mathbf{e}_\theta$ and is obtained as the radial projection of $\mathbf{y} \in \partial\Omega$ onto $S_{\mathbf{x}}$), and Δ_S is the sphere Laplacian. The fact that $(N-1)$ is the smallest non-zero eigenvalue of $-\Delta_S$ implies the linear precision of (2) and (3) (see the last paragraph of Sect. 2 for an explanation).

2 Introduction to Integral-Based Barycentric Coordinates

Let Ω be a bounded convex domain in $\mathbb{R}^N$ and $\mathbf{x}$ be a point inside Ω. Assume that we know the values of function $u(\cdot)$ on $\partial\Omega$. Integral-based barycentric interpolation T interpolates $u(\cdot)$ inside Ω

$$T : u|_{\partial\Omega} \to u(\mathbf{x})$$

while preserving the linear functions.

Consider the unit sphere $S_{\mathbf{x}}$ centered at $\mathbf{x}$. We assume that $S_{\mathbf{x}}$ is parameterized by its outer unit normal $\mathbf{e}_\theta$, where $\boldsymbol{\theta}$ stands for spherical coordinates. See the right image of Fig. 1 for a visual explanation of some notations used. A general form of interpolation T is given by

$$u(\mathbf{x}) = \int\limits_{S_{\mathbf{x}}} \frac{u(\mathbf{y})w(\mathbf{x},\mathbf{e}_\theta)}{|\mathbf{x}-\mathbf{y}|}\,\mathrm{d}\boldsymbol{\theta} \Bigg/ \int\limits_{S_{\mathbf{x}}} \frac{w(\mathbf{x},\mathbf{e}_\theta)}{|\mathbf{x}-\mathbf{y}|}\,\mathrm{d}\boldsymbol{\theta}\,, \quad \mathbf{x}\in\Omega, \quad \mathbf{y}\in\partial\Omega, \tag{4}$$

where $w(\mathbf{x},\mathbf{e}_\theta)$ is a weighting function satisfying orthogonality conditions

$$\mathbf{0} = \int\limits_{S_{\mathbf{x}}} \mathbf{e}_\theta\, w(\mathbf{x},\mathbf{e}_\theta)\,\mathrm{d}\boldsymbol{\theta} \quad \text{for each} \quad \mathbf{x}\in\Omega. \tag{5}$$

Note that (5) is necessary and sufficient for linear precision. Indeed setting $u(\mathbf{x}) \equiv \mathbf{x}$ yields

$$u(\mathbf{y}) \equiv \mathbf{y} = \mathbf{x} + \rho\,\mathbf{e}_\theta, \quad \rho = |\mathbf{x}-\mathbf{y}|,$$

which, after a substitution into (4), gives (5).

If $w(\mathbf{x},\mathbf{e}_\theta) \equiv 1$, we arrive at the integral-based mean value interpolation scheme (or transfinite mean value coordinates)

$$u(\mathbf{x}) = \int\limits_{S_{\mathbf{x}}} \frac{u(\mathbf{y})\,\mathrm{d}\boldsymbol{\theta}}{|\mathbf{x}-\mathbf{y}|} \Bigg/ \int\limits_{S_{\mathbf{x}}} \frac{\mathrm{d}\boldsymbol{\theta}}{|\mathbf{x}-\mathbf{y}|}, \tag{6}$$

which was originally proposed in [10] for 2D polygons and then in [14, 17] for simplicial polyhedra and the continuous case.

2D Case
In the 2D case, $\mathbf{e}_\theta = (\cos\theta, \sin\theta)$ and (5) simplifies to a system of two equations

$$\int_0^{2\pi} w(\mathbf{x}, \theta)\cos\theta \, \mathrm{d}\theta = 0 = \int_0^{2\pi} w(\mathbf{x}, \theta)\sin\theta \, \mathrm{d}\theta \tag{7}$$

for each $\mathbf{x} \in \Omega$.

Let us expand $w(\mathbf{x}, \theta)$ into the Fourier series

$$w(\mathbf{x}, \theta) = \sum c_n(\mathbf{x})\mathrm{e}^{jn\theta}, \quad j = \sqrt{-1}.$$

Note that (7) is equivalent to $c_{-1} = 0 = c_1$ and, therefore, $w(\mathbf{x}, \theta)$ can be represented as

$$h''_{\theta\theta}(\mathbf{x}, \theta) + h(\mathbf{x}, \theta) = w(\mathbf{x}, \theta) \tag{8}$$

for some function $h(\mathbf{x}, \theta)$ periodic in θ. Indeed, expanding $h(\mathbf{x}, \theta)$ into the Fourier series

$$h(\mathbf{x}, \theta) = \sum h_n(\mathbf{x})\mathrm{e}^{jn\theta}$$

and substituting the expansion into (8) leads to

$$-n^2 h_n(\mathbf{x}) + h_n(\mathbf{x}) = c_n(\mathbf{x}), \quad h_n(\mathbf{x}) = c_n(\mathbf{x})/(1 - n^2), \quad \text{where} \quad n \neq \pm 1,$$

and defines $h(\mathbf{x}, \theta)$ uniquely if, in addition, we set $h_{-1}(\mathbf{x}) = 0 = h_1(\mathbf{x})$.

We can interpret (8) geometrically. For each point $\mathbf{x} \in \Omega$, let us consider a closed curve $\Sigma_\mathbf{x}$ whose support function is given by $h(\mathbf{x}, \theta)$ defined by (8). The radius of curvature of $\Sigma_\mathbf{x}$ is given by the left-hand side of (8) and orthogonality conditions (7) can be written as

$$\int_0^{2\pi} \mathbf{n}\frac{\mathrm{d}\theta}{k} \equiv \int_{\Sigma_\mathbf{x}} \mathbf{n} \, \mathrm{d}l = 0, \tag{9}$$

where $\mathbf{n} = \mathbf{e}_\theta = (\cos\theta, \sin\theta)$ is the outer unit normal of $\Sigma_\mathbf{x}$, k is the curvature of $\Sigma_\mathbf{x}$, and l stands for the arc-length parametrization of $\Sigma_\mathbf{x}$. For example, if for each $\mathbf{x}$ the curve $\Sigma_\mathbf{x}$ is a unit circle centered at $\mathbf{x}$, then $w(\mathbf{x}, \theta) \equiv 1$, and we arrive at the 2D version of the transfinite mean value coordinates (6).

N-Dimensional Case

The N components of the unit normal $\mathbf{e}_\theta$ are the eigenfunctions corresponding to the minimum non-zero eigenvalue $\lambda_{\min} = N - 1$ of the sphere Laplacian $-\Delta_S$. Thus if the weighting function $w(\mathbf{x}, \boldsymbol{\theta})$ in (4) is given by

$$w(\mathbf{x}, \boldsymbol{\theta}) = \Delta_S h + (N-1)h \tag{10}$$

for some function $h(\mathbf{x}, \boldsymbol{\theta})$, then the orthogonality conditions (5) are satisfied. Indeed, simple integration by parts on the unit sphere $S_\mathbf{x}$ yields

$$\begin{aligned}\int_{S_\mathbf{x}} \mathbf{e}_\theta w(\mathbf{x}, \mathbf{e}_\theta)\, \mathrm{d}\boldsymbol{\theta} &= \int_{S_\mathbf{x}} \mathbf{e}_\theta [\Delta_S h + (N-1)h]\, \mathrm{d}\boldsymbol{\theta} \\ &= \int_{S_\mathbf{x}} [\Delta_S \mathbf{e}_\theta + (N-1)\mathbf{e}_\theta] h(\mathbf{x}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} = \mathbf{0}.\end{aligned}$$

3 2D Integral-Based Barycentric Coordinates as a Limit Case of Their Discrete Counterparts

Let us recall a general construction for generalized barycentric coordinates introduced in [15]. Let $A(\mathbf{x}, \mathbf{y}, \mathbf{z})$ denote the signed area of the triangle formed by points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$. Given a convex polygon with vertices $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, and a point $\mathbf{x}$ inside the polygon, consider signed triangle areas $A_i(\mathbf{x}) = A(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_{i+1})$ and $B_i(\mathbf{x}) = A(\mathbf{x}, \mathbf{v}_{i-1}, \mathbf{v}_{i+1})$. Then, as shown in [15], the weights

$$w_i = \frac{c_{i+1} A_{i-1} - c_i B_i + c_{i-1} A_i}{A_{i-1} A_i}, \tag{11}$$

where $c_i(\mathbf{x})$ are real functions, define a system of generalized barycentric coordinates. Moreover, any system of generalized barycentric coordinates can be represented by (11) for some functions $c_i(\mathbf{x})$, $i = 1, 2, \ldots, n$.

Following [15], we can rewrite (11) as

$$w_i = \frac{2}{\rho_i} \left(\frac{h_{i+1} - h_i \cos\theta_i}{\sin\theta_i} + \frac{h_{i-1} - h_i \cos\theta_{i-1}}{\sin\theta_{i-1}} \right), \tag{12}$$

where $h_i(\mathbf{x}) = c_i(\mathbf{x})/\rho_i(\mathbf{x})$, $\rho_i(\mathbf{x}) = |\mathbf{x} - \mathbf{v}_i|$, and θ_i is the angle between the rays $[\mathbf{x}\,\mathbf{v}_i)$ and $[\mathbf{x}\,\mathbf{v}_{i+1})$.

Assume now that functions $h_i(\mathbf{x})$, $i = 1, 2, \ldots, n$, are sufficiently smooth, the number of vertices of the polygon tends to infinity, and all θ_i uniformly tend to zero: $\theta_i \approx d\theta \to 0$. Passing to the limit we arrive at a smooth function $h(\mathbf{x}, \theta)$ satisfying

$$\frac{h_{i+1} - h_i \cos\theta_i}{\sin\theta_i} \approx \frac{h_{i+1} - h_i + h_i\,\theta_i^2/2}{\theta_i} \approx \left[h(\mathbf{x},\theta)'_\theta + h(\mathbf{x},\theta)\frac{\theta_i}{2}\right]_{\theta=\sum_{k=1}^{i+1}\theta_k},$$

$$\frac{h_{i-1} - h_i \cos\theta_{i-1}}{\sin\theta_{i-1}} \approx \frac{h_{i-1} - h_i + h_i\,\theta_{i-1}^2/2}{\theta_{i-1}} \approx \left[-h(\mathbf{x},\theta)'_\theta + h(\mathbf{x},\theta)\frac{\theta_{i-1}}{2}\right]_{\theta=\sum_{k=1}^{i}\theta_k}.$$

Adding together the right-hand sides yields

$$\left[\frac{h(\mathbf{x},\theta+\mathrm{d}\theta)'_\theta - h(\mathbf{x},\theta)'_\theta}{\mathrm{d}\theta} + h(\mathbf{x},\theta)\right]_{\theta=\sum_{k=1}^{i}\theta_k}\mathrm{d}\theta + \text{higher order terms}.$$

Therefore (12) is approximately equal to

$$\frac{2}{\rho_i}\left[h(\mathbf{x},\theta)''_{\theta\theta} + h(\mathbf{x},\theta)\right]\Big|_{\theta=\sum_{k=1}^{i}\theta_k}\mathrm{d}\theta$$

and we arrive at (8). We can now formulate our result as a proposition which generalizes the result of [16, Section 3.2.2].

Proposition 1 *In the 2D case, integral-based barycentric coordinates* (4) *and* (8) *are obtained as the limit case of the Floater-Hormann-Kós construction* (11)*.*

In view of the work of Kosinka and Barton [19] where the quadratic rate convergence of generalized barycentric coordinates to their continuous counterparts, barycentric kernels, was proved and numerically verified, the importance of (1) consists of revealing a link between the functions $c_i(\mathbf{x})$ in (11) and the support function $h(\mathbf{x})$ defined by (8).

4 2D Integral-Based Coordinates and Dirichlet Energy Minimization

Integral-Based Laplace Coordinates in 2D

By 2D integral-based Laplace coordinates we denote the continuous version of the discrete harmonic coordinates. As shown in [15], setting $c_i = \rho_i^2$ in (11) yields the discrete harmonic coordinates. Thus, in view of Proposition 1, the integral-based Laplace coordinates are described by

$$w(\mathbf{x},\theta) = \rho''_{\theta\theta} + \rho. \tag{13}$$

If the weight $w(\mathbf{x}, \theta)$ in (4) is positive, the interpolation property of (4) follows from [12, Theorem 1]. However (13) is not necessary positive and the interpolation property of (4) and (13) requires a special attention. Let us start from the simplest case: assume that Ω is a unit circle centered at the origin of coordinates and x has coordinates $(-a, 0)$, where $0 < a < 1$. Simple analytic calculations shows that the kernel

$$k(\mathbf{x}, \theta) = \big(\rho''_{\theta\theta} + \rho\big)/\rho \tag{14}$$

has two equal maxima achieved at θ_1 and θ_2 which approach $\pi/2$ and $3\pi/2$, respectively, as $a \to 1$. See the left image of (2) for an illustration. The maxima are equal to $32/\big[27(1-a^2)\big]$ and become sharper and sharper, as $\mathbf{x}$ approaches the boundary, when $a \to 1$, as demonstrated by the middle and right images of Fig. 2. The boundary points $\mathbf{y}_1$ and $\mathbf{y}_2$ corresponding to θ_1 and θ_2, respectively, become closer and closer to each other, as $a \to 1$, and, in the limit, merge with $\mathbf{x}$ on the boundary. Thus, similar to the classical Shepard's interpolation [24],

$$u(\mathbf{x}) = \int\limits_{S_\mathbf{x}} k(\mathbf{x}, \theta) u(\mathbf{y})\, \mathrm{d}\theta \Big/ \int\limits_{S_\mathbf{x}} k(\mathbf{x}, \theta)\, \mathrm{d}\theta \tag{15}$$

does the boundary interpolation. The general case of strictly convex Ω can be easily reduced to the above case of a circle, if for a given point $\mathbf{y} \in \partial\Omega$ we consider the osculating (best-fitted) circle touching $\partial\Omega$ at $\mathbf{y}$.

It turns out that (13) can be also derived by mimicking the Dirichlet's energy minimization property of the harmonic functions [2, 23]. Given $u(\mathbf{y})$ defined for each $\mathbf{y} \in \partial\Omega$, let us choose $\mathbf{x} \in \Omega$ and assume that $u(\mathbf{x})$ is known. Consider a conical surface patch generated by straight segments connecting the inner point $(\mathbf{x}, u(\mathbf{x}))$ with the boundary points $(\mathbf{y}, u(\mathbf{y}))$. In polar coordinates (r, θ) centered at $\mathbf{x}$, $\partial\Omega$ is described by $r = \rho(\theta)$, $\rho = |\mathbf{x} - \mathbf{y}|$, and the conical surface associated with point $\mathbf{x}$ is given by

$$U_\mathbf{x}(\mathbf{z}) = \frac{(\rho - r)u(\mathbf{x}) + r u(\mathbf{y})}{\rho}, \tag{16}$$

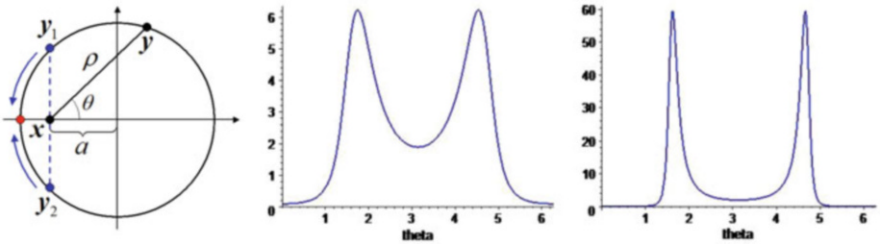


Fig. 2 Left: An illustration of notations used to establish interpolation properties of (14) and (15) for a circle. Middle and right: the graphs of kernel $k(\mathbf{x}, \theta)$ defined by (14) for $\mathbf{x} = (-0.9, 0)$ and $\mathbf{x} = (-0.99, 0)$, respectively

where $r = |\mathbf{x} - \mathbf{z}|$ and $\mathbf{y} \in \partial\Omega$ denotes the intersection point between $\partial\Omega$ and the ray from $\mathbf{x}$ through $\mathbf{z}$. See Fig. 1 for a rough sketch of the conical surface $(\mathbf{z}, U_{\mathbf{x}}(\mathbf{z}))$, $\mathbf{z} \in \Omega$ and notations used.

Now the value $u(\mathbf{x})$ is defined such that the Dirichlet's energy of the constructed conical surface attains its minimal value

$$\iint_{\Omega} |\nabla U_{\mathbf{x}}|^2 \, \mathrm{d}\mathbf{z} \to \min .$$

Proposition 2 *In the 2D case, the minimum of the Dirichlet energy for a conical surface consisting of straight segments connecting the inner point* $(\mathbf{x}, u(\mathbf{x}))$, $\mathbf{x} \in \Omega$, *with the boundary points* $(\mathbf{y}, u(\mathbf{y}))$, $\mathbf{y} \in \partial\Omega$, *is attained on the 2D integral-based barycentric interpolation* (4) *and* (13).

A brief derivation of this result is given in [2]. For the sake of completeness, we present here a detail proof of the proposition.

Proof For a function $f(r, \theta)$, its gradient (and its magnitude) in polar coordinates is given by

$$\nabla f(r,\theta) = \frac{\partial f}{\partial r}\mathbf{e}_r + \frac{1}{r}\frac{\partial f}{\partial \theta}\mathbf{e}_\theta, \quad |\nabla f|^2 = \left|\frac{\partial f}{\partial r}\right|^2 + \frac{1}{r^2}\left|\frac{\partial f}{\partial \theta}\right|^2.$$

Therefore

$$\frac{\partial U_{\mathbf{x}}}{\partial r} = \frac{u(\mathbf{y}) - u(\mathbf{x})}{\rho} \quad \text{and} \quad \frac{\partial U_{\mathbf{x}}}{\partial \theta} = r\left[(u(\mathbf{y}) - u(\mathbf{x}))\left(\frac{1}{\rho}\right)'_{\theta} + \frac{1}{\rho}u'_{\theta}(\mathbf{y})\right].$$

We arrive at the following minimization problem

$$\begin{aligned}
\min \leftarrow & \iint_{\Omega} |\nabla U_{\mathbf{x}}|^2 \, \mathrm{d}\mathbf{z} \\
&= \int_0^{2\pi} \mathrm{d}\theta \int_0^{\rho} r \, \mathrm{d}r \left\{ \left[\frac{u(\mathbf{x}) - u(\mathbf{y})}{\rho}\right]^2 + \left[(u(\mathbf{y}) - u(\mathbf{x}))\left(\frac{1}{\rho}\right)'_{\theta} + \frac{1}{\rho}u'_{\theta}(\mathbf{y})\right]^2 \right\} \\
&= \frac{1}{2}\int_0^{2\pi} d\theta \left\{ [u(\mathbf{x}) - u(\mathbf{y})]^2 + \left[u'_{\theta}(\mathbf{y}) + (u(\mathbf{x}) - u(\mathbf{y}))\frac{\rho'_{\theta}}{\rho}\right]^2 \right\},
\end{aligned}$$

where the last integral is a quadratic function w.r.t. $u(\mathbf{x})$. Thus for the optimal value of $u(\mathbf{x})$ we have

$$\int_0^{2\pi} u(\mathbf{x})\left\{1+\left(\frac{\rho'_\theta}{\rho}\right)^2\right\}\mathrm{d}\theta = \int_0^{2\pi}\left\{u(\mathbf{y})-u'_\theta(\mathbf{y})\frac{\rho'_\theta}{\rho}+u(\mathbf{y})\left(\frac{\rho'_\theta}{\rho}\right)^2\right\}\mathrm{d}\theta$$

and, therefore, $u(\mathbf{x})$ is given by

$$u(\mathbf{x}) = \int_0^{2\pi}\left\{u(\mathbf{y})-u'_\theta(\mathbf{y})\left[\rho'_\theta/\rho\right]+u(\mathbf{y})\left[\rho'_\theta/\rho\right]^2\right\}\mathrm{d}\theta \Big/ \int_0^{2\pi}\left\{1+\left[\rho'_\theta/\rho\right]^2\right\}\mathrm{d}\theta\,. \tag{17}$$

Integration by parts yields

$$-\int_0^{2\pi} u'_\theta(\mathbf{y})\left[\rho'_\theta/\rho\right]\mathrm{d}\theta = \int_0^{2\pi} u(\mathbf{y})\left[\rho'_\theta/\rho\right]'_\theta\,\mathrm{d}\theta = \int_0^{2\pi} u(\mathbf{y})\left[\rho''_{\theta\theta}/\rho-\left(\rho'_\theta/\rho\right)^2\right]\mathrm{d}\theta\,,$$

$$\int_0^{2\pi}\left[\rho'_\theta/\rho\right]^2\mathrm{d}\theta = \int_0^{2\pi}\left(\rho''_{\theta\theta}/\rho-\left[\rho'_\theta/\rho\right]'_\theta\right)\mathrm{d}\theta = \int_0^{2\pi}\left(\rho''_{\theta\theta}/\rho\right)\mathrm{d}\theta$$

and thus (17) becomes

$$u(\mathbf{x}) = \int_0^{2\pi} u(\mathbf{y})\frac{\rho''_{\theta\theta}+\rho}{\rho}\,\mathrm{d}\theta \Big/ \int_0^{2\pi}\frac{\rho''_{\theta\theta}+\rho}{\rho}\,\mathrm{d}\theta\,. \tag{18}$$

One can see now that (18) corresponds to (4) with (13). □

General Construction in 2D

In this section, we show that, similar to the integral-based Laplace coordinates, the general construction of 2D integral-based barycentric coordinates (4) and (8) can also be obtained as the solution to a Dirichlet energy minimization problem.

Again, we consider a point $\mathbf{x}$ inside a convex domain Ω and assume that $\partial\Omega$ is given by $r = \rho(\theta)$ in polar coordinates centered at $\mathbf{x}$. Thus Ω can be described by $\{(r\cos\theta, r\sin\theta)\}$ with $0 \le r < \rho(\theta)$. Let us now consider another domain G defined by $0 \le r < g(\mathbf{x},\theta)$, where $g(\mathbf{x},\theta)$ is some function. Then ∂G is given by $r = g(\mathbf{x},\theta)$.

Assume that we know $u(\mathbf{x})$ and use linear interpolation between $u(\mathbf{x})$ and $u(\mathbf{y})$, where $\mathbf{y} \in \partial\Omega$. Then the values of $u(\cdot)$ on ∂G are given by

$$v = \frac{u(\mathbf{y})g(\mathbf{x},\theta)+u(\mathbf{x})(\rho-g(\mathbf{x},\theta))}{\rho}. \tag{19}$$

Now let us apply the integral-based Laplace interpolation to the domain G. We have

$$\int_0^{2\pi} \frac{g''_{\theta\theta} + g}{g} v \, \mathrm{d}\theta = u(\mathbf{x}) \int_0^{2\pi} \frac{g''_{\theta\theta} + g}{g} \, \mathrm{d}\theta$$

and substituting $v(\cdot)$ defined by (19) yields

$$\int_0^{2\pi} \frac{g''_{\theta\theta} + g}{\rho} u(\mathbf{y}) \, \mathrm{d}\theta + u(\mathbf{x}) \int_0^{2\pi} \left(1 - \frac{g}{\rho}\right) \frac{g''_{\theta\theta} + g}{g} \, \mathrm{d}\theta = u(\mathbf{x}) \int_0^{2\pi} \frac{g''_{\theta\theta} + g}{g} \, \mathrm{d}\theta \,,$$

which immediately leads to the following general representation of 2D integral-based coordinates

$$u(\mathbf{x}) = \int_0^{2\pi} \frac{g''_{\theta\theta} + g}{\rho} u(\mathbf{y}) \, \mathrm{d}\theta \Big/ \int_0^{2\pi} \frac{g''_{\theta\theta} + g}{\rho} \, \mathrm{d}\theta \tag{20}$$

and yields the general 2D integral-based barycentric interpolation (4) and (8).

5 Potential Application to Surface Generation

In this section we briefly discuss a potential application of the variational approach of Sect. 4 to surface generation. Consider a simple closed curve Γ in $\mathbb{R}^3$. The task we consider here consists of constructing a smooth surface patch S spanning contour Γ. For example, S can be constructed as a minimal surface patch bounded by Γ.

Our approach can be considered as a simplified version of the surface area minimization. For each point $\mathbf{x}$, we consider a conical surface patch generated by straight segments connecting $\mathbf{x}$ with the points of Γ. Let $A(\mathbf{x})$ denote the area of the conical surface patch with apex at $\mathbf{x}$. Then we move $\mathbf{x}$ by the anti-gradient flow $-\nabla_{\mathbf{x}} A(\mathbf{x})$. Numerical experiments show that the flow has a single stationary point, see the left image of Fig. 3. Now let us apply the flow to the points of contour Γ. The trajectories which converge to the stationary point defined by Γ form a surface patch bounded by Γ, see the middle image of Fig. 3. As seen in the right image of Fig. 3, the shape of the surface patch is similar to the minimal surface spanning Γ.

For practical computations, we approximate Γ by a polygon, connect a given point $\mathbf{x}$ with the vertices of Γ, and calculate $A(\mathbf{x})$ as the side area of a triangular pyramid with non-flat base Γ and apex $\mathbf{x}$. Our straightforward implementation of the anti-gradient flow $-\nabla_{\mathbf{x}} A(\mathbf{x})$ by the forward Euler method is not computationally efficient and additional work is needed to make our approach a practical one.

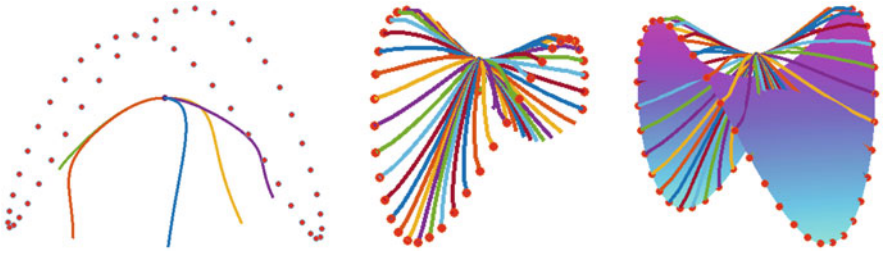


Fig. 3 Surface generation with integral-based Laplace coordinates. Left: several trajectories of proposed anti-gradient flow $-\nabla_{\mathbf{x}} A(\mathbf{x})$ converge to a single stationary point. Middle: A surface patch is generated by the trajectories of $-\nabla_{\mathbf{x}} A(\mathbf{x})$ emitted from the boundary vertices. Right: the minimal surface spanned the boundary curve is added; its shape is similar to that of the surface generated by the trajectories

6 3D Integral-Based Coordinates and Dirichlet Energy Minimization

Integral-Based Laplace Coordinates in 3D
We define the 3D integral-based Laplace coordinates as the barycentric interpolation (4) which, for each $\mathbf{x} \in \Omega$, minimizes the Dirichlet energy

$$\iiint_{\Omega} |\nabla U_{\mathbf{x}}|^2 \, \mathrm{d}\mathbf{z} \to \min$$

of the conical surface (16) associated with $\mathbf{x}$.

Proposition 3 *In the 3D case, the minimum of the Dirichlet energy for a conical surface consisting of straight segments connecting the inner point* $(\mathbf{x}, u(\mathbf{x}))$, $\mathbf{x} \in \Omega$, *with the boundary points* $(\mathbf{y}, u(\mathbf{y}))$, $\mathbf{y} \in \partial\Omega$, *is attained on the integral-based barycentric interpolation* (4) *with the weighting function given by*

$$w(\mathbf{x}, \mathbf{e}_\theta) = \Delta_S(\rho^2/2) + 2(\rho^2/2), \tag{21}$$

where Δ_S *is the sphere Laplacian and* $\rho(\boldsymbol{\theta})$ *represents* $\partial\Omega$ *in the spherical coordinates centered at* $\mathbf{x}$ *(that is, for each* $\mathbf{y} \in \partial\Omega$, $\rho = |\mathbf{x} - \mathbf{y}|$ *and* $\mathbf{e}_\theta = (\mathbf{y} - \mathbf{x})/\rho$*).*

Proof Note that (5) is satisfied. The components of the sphere normal $\mathbf{e}_\theta$ are the eigenvectors of the sphere Laplacian $-\Delta_S$ corresponding to the minimum positive eigenvalue $\lambda = 2$. Integration by parts yields

$$\int\limits_{S_\mathbf{x}} \mathbf{e}_\theta \, w(\mathbf{x}, \mathbf{e}_\theta) \, d\boldsymbol{\theta} = \int\limits_{S_\mathbf{x}} \mathbf{e}_\theta \Big[\Delta_S(\rho^2/2) + 2(\rho^2/2) \Big] \, \mathrm{d}\boldsymbol{\theta}$$

$$= \int\limits_{S_\mathbf{x}} [\Delta_S \mathbf{e}_\theta + 2\mathbf{e}_\theta](\rho^2/2) \, \mathrm{d}\boldsymbol{\theta} = 0,$$

where $\boldsymbol{\theta}$ stand for spherical coordinates.

Given a point $\mathbf{x} \in \Omega$, we consider standard spherical coordinates centered at $\mathbf{x}$: radial distance r, polar angle θ, and azimuthal angle φ. Thus $\boldsymbol{\theta} = (\varphi, \theta)$ and $\mathrm{d}\boldsymbol{\theta} = \sin\theta \, \mathrm{d}\varphi \, \mathrm{d}\theta$. Given a function $f(r, \theta, \varphi)$, its gradient and the magnitude of the gradient in spherical coordinates are given by

$$\nabla f(r, \theta, \varphi) = \frac{\partial f}{\partial r}\mathbf{e}_r + \frac{1}{r}\frac{\partial f}{\partial \theta}\mathbf{e}_\theta + \frac{1}{r \sin\theta}\frac{\partial f}{\partial \varphi}\mathbf{e}_\varphi,$$

$$|\nabla f|^2 = \left|\frac{\partial f}{\partial r}\right|^2 + \frac{1}{r^2}\left|\frac{\partial f}{\partial \theta}\right|^2 + \frac{1}{r^2 \sin^2\theta}\left|\frac{\partial f}{\partial \varphi}\right|^2.$$

Similar to the planar case we have

$$U_\mathbf{x}(\mathbf{z}) = \frac{(\rho - r)u(\mathbf{x}) + ru(\mathbf{y})}{\rho}$$

and, therefore,

$$\frac{\partial U_\mathbf{x}}{\partial r} = \frac{u(\mathbf{y}) - u(\mathbf{x})}{\rho},$$

$$\frac{\partial U_\mathbf{x}}{\partial \theta} = r\left[(u(\mathbf{y}) - u(\mathbf{x}))\left(\frac{1}{\rho}\right)'_\theta + \frac{1}{\rho}u'_\theta(\mathbf{y})\right],$$

$$\frac{\partial U_\mathbf{x}}{\partial \varphi} = r\left[(u(\mathbf{y}) - u(\mathbf{x}))\left(\frac{1}{\rho}\right)'_\varphi + \frac{1}{\rho}u'_\varphi(\mathbf{y})\right].$$

Similar to the 2D case, we will define $u(\mathbf{x})$ such that the Dirichlet's energy of the constructed conical surface $U_{\mathbf{x}}(\mathbf{z})$ attains its minimal value. We consider the following minimization problem

$$\min \leftarrow \iiint_{\Omega} |\nabla U_{\mathbf{x}}|^2 \, \mathrm{d}\mathbf{z} = \int_0^{\pi}\int_0^{2\pi}\int_0^{\rho} r^2 \sin\theta \, \mathrm{d}r \, \mathrm{d}\varphi \, \mathrm{d}\theta \left\{ \left[\frac{u(\mathbf{x}) - u(\mathbf{y})}{\rho} \right]^2 \right.$$

$$\left. + \left[(u(\mathbf{y}) - u(\mathbf{x})) \left(\frac{1}{\rho} \right)'_{\theta} + \frac{1}{\rho} u'_{\theta}(\mathbf{y}) \right]^2 + \frac{1}{\sin^2\theta} \left[(u(\mathbf{y}) - u(\mathbf{x})) \left(\frac{1}{\rho} \right)'_{\varphi} + \frac{1}{\rho} u'_{\varphi}(\mathbf{y}) \right]^2 \right\}$$

$$= \frac{1}{3} \int_0^{\pi}\int_0^{2\pi} \rho(\theta, \varphi) \sin\theta \, \mathrm{d}\varphi \, \mathrm{d}\theta \left\{ [u(\mathbf{x}) - u(\mathbf{y})]^2 \right.$$

$$\left. + \left[u'_{\theta}(\mathbf{y}) + (u(\mathbf{x}) - u(\mathbf{y})) \frac{\rho'_{\theta}}{\rho} \right]^2 + \frac{1}{\sin^2\theta} \left[u'_{\varphi}(\mathbf{y}) + (u(\mathbf{x}) - u(\mathbf{y})) \frac{\rho'_{\varphi}}{\rho} \right]^2 \right\}.$$

Thus, for the optimal value of $u(\mathbf{x})$ we have

$$\int_0^{\pi}\int_0^{2\pi} u(\mathbf{x}) \left\{ 1 + \left(\frac{\rho'_{\theta}}{\rho} \right)^2 + \left(\frac{\rho'_{\varphi}}{\rho \sin\theta} \right)^2 \right\} \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi$$

$$= \int_0^{\pi}\int_0^{2\pi} \left\{ u(\mathbf{y}) - u'_{\theta}(\mathbf{y}) \frac{\rho'_{\theta}}{\rho} - u'_{\varphi}(\mathbf{y}) \frac{\rho'_{\varphi}}{\rho \sin^2\theta} \right.$$

$$\left. + u(\mathbf{y}) \left(\frac{\rho'_{\theta}}{\rho} \right)^2 + u(\mathbf{y}) \left(\frac{\rho'_{\varphi}}{\rho \sin\theta} \right)^2 \right\} \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \,.$$

Let us consider

$$\int_0^{\pi}\int_0^{2\pi} \left\{ 1 + \left(\frac{\rho'_{\theta}}{\rho} \right)^2 + \left(\frac{\rho'_{\varphi}}{\rho \sin\theta} \right)^2 \right\} \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \tag{22}$$

and set $f = \rho^2/2$. We have $(\rho'_{\varphi})^2/\rho = f''_{\varphi\varphi}/\rho - \rho''_{\varphi\varphi}$. Thus the third term in (22) can be rewritten as

$$\int_0^{\pi}\int_0^{2\pi} \left(\frac{\rho'_{\varphi}}{\rho \sin\theta} \right)^2 \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi = \int_0^{\pi}\int_0^{2\pi} \frac{1}{\sin^2\theta} \frac{f''_{\varphi\varphi}}{\rho} \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \,,$$

since

$$\int_0^{2\pi} \rho''_{\varphi\varphi}\, \mathrm{d}\varphi = 0.$$

Obviously for the first term in (22) we have

$$\int_0^{\pi}\int_0^{2\pi} \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi = \int_0^{\pi}\int_0^{2\pi} \frac{2f}{\rho} \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \, .$$

Finally we use $(\rho'_\theta)^2/\rho = f''_{\theta\theta}/\rho - \rho''_{\theta\theta}$ and integration by parts w.r.t. θ for rearranging the second term in (22)

$$\begin{aligned}
\int_0^{\pi}\int_0^{2\pi} \left(\frac{\rho'_\theta}{\rho}\right)^2 \rho \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi &= \int_0^{\pi}\int_0^{2\pi} \left(\frac{f''_{\theta\theta}}{\rho} - \rho''_{\theta\theta}\right) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \\
&= \int_0^{\pi}\int_0^{2\pi} \left(\frac{f''_{\theta\theta}}{\rho} + \rho'_\theta \frac{\cos\theta}{\sin\theta}\right) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \\
&= \int_0^{\pi}\int_0^{2\pi} \left(\frac{f''_{\theta\theta}}{\rho} + \frac{f'_\theta}{\rho} \frac{\cos\theta}{\sin\theta}\right) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \, .
\end{aligned}$$

Thus, using the fact that the sphere Laplacian is given by

$$\Delta_S f = \frac{f''_{\varphi\varphi}}{\sin^2\theta} + \frac{1}{\sin\theta}\left(\sin\theta \, f'_\theta\right)'_\theta \equiv f''_{\theta\theta} + f'_\theta \frac{\cos\theta}{\sin\theta} + \frac{f''_{\varphi\varphi}}{\sin^2\theta},$$

(22) can be rewritten as

$$\int_0^{\pi}\int_0^{2\pi} \frac{\Delta_S f + 2f}{\rho} \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi \, ,$$

where $f = \rho^2/2$ and $\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi$ is the area element of the unit sphere.

In a similar way we have

$$\int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\rho\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi = \int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\frac{2f}{\rho}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,,$$

$$\int_0^{\pi}\int_0^{2\pi} u'_\theta(\mathbf{y})\frac{\rho'_\theta}{\rho}\rho\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi = -\int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\frac{\left(\rho'_\theta\sin\theta\right)'_\theta}{\sin\theta}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,,$$

$$\int_0^{\pi}\int_0^{2\pi} u'_\varphi(\mathbf{y})\frac{\rho'_\varphi}{\rho\sin^2\theta}\rho\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi = -\int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\frac{\rho''_{\varphi\varphi}}{\sin^2\theta}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,,$$

$$\int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\left(\frac{\rho'_\theta}{\rho}\right)^2\rho\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi = \int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\left(\frac{f''_{\theta\theta}}{\rho}-\rho''_{\theta\theta}\right)\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,,$$

$$\int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\left(\frac{\rho'_\varphi}{\rho\sin\theta}\right)^2\rho\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi = \int_0^{\pi}\int_0^{2\pi} u(\mathbf{y})\frac{1}{\sin^2\theta}\left(\frac{f''_{\varphi\varphi}}{\rho}-\rho''_{\varphi\varphi}\right)\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,.$$

Combining these terms together yields

$$u(\mathbf{x}) = \int_0^{\pi}\int_0^{2\pi}\frac{u(\mathbf{y})w(\mathbf{x},\theta,\varphi)}{\rho}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\Big/\int_0^{\pi}\int_0^{2\pi}\frac{w(\mathbf{x},\theta,\varphi)}{\rho}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi\,,$$

with

$$w(\mathbf{x},\theta,\varphi) = \Delta_S f + 2f, \quad \text{where} \quad f = \rho^2/2.$$

□

General Construction in 3D

Similar to the 2D case, let us introduce the domain G defined by the radial function $g(\mathbf{x},\theta,\varphi)$ and apply the integral-based Laplace interpolation to G, where the values of $u(\cdot)$ on ∂G are found by linear interpolation

$$v = \frac{u(\mathbf{y})g + u(\mathbf{x})(\rho - g)}{\rho}.$$

Then we have

$$\int_{S_\mathbf{x}}\frac{\Delta_S(g^2/2)+2(g^2/2)}{g}v\,\mathrm{d}\boldsymbol{\theta} = u(\mathbf{x})\int_{S_\mathbf{x}}\frac{\Delta_S(g^2/2)+2(g^2/2)}{g}\,\mathrm{d}\boldsymbol{\theta}\,,$$

where $d\boldsymbol{\theta} = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi$ is the area element of $S_{\mathbf{x}}$, the unit sphere centered at $\mathbf{x}$. Similar to the 2D case, this leads to

$$\int_{S_{\mathbf{x}}} \frac{\Delta_S(g^2/2) + 2(g^2/2)}{\rho} u(\mathbf{y}) \, \mathrm{d}\boldsymbol{\theta} + u(\mathbf{x}) \int_{S_{\mathbf{x}}} \left(1 - \frac{g}{\rho}\right) \frac{\Delta_S(g^2/2) + 2(g^2/2)}{g} \, \mathrm{d}\boldsymbol{\theta}$$

$$= u(\mathbf{x}) \int_{S_{\mathbf{x}}} \frac{\Delta_S(g^2/2) + 2(g^2/2)}{g} \, \mathrm{d}\boldsymbol{\theta}$$

and we arrive at

$$u(\mathbf{x}) = \int_{S_{\mathbf{x}}} \frac{\Delta_S(g^2/2) + 2(g^2/2)}{\rho} u(\mathbf{y}) \, \mathrm{d}\boldsymbol{\theta} \Big/ \int_{S_{\mathbf{x}}} \frac{\Delta_S(g^2/2) + 2(g^2/2)}{\rho} \, \mathrm{d}\boldsymbol{\theta} \, .$$

Thus we have arrived at the general 3D integral-based barycentric interpolation (4) and (8) with $h(\mathbf{x}, \theta, \varphi) = g^2/2$.

7 Conclusion

We have used a simple variational principle (the minimization of the Dirichlet energy of a conical surface) to obtain an integral-based version of the Laplace barycentric coordinates and extended the approach to arrive at a general description of integral-based barycentric coordinates in 2D and 3D. We have also considered an application of our approach to a surface generation problem.

Acknowledgments We would like to thank the anonymous reviewers of this paper for their valuable and constructive comments.

References

1. Belikov, V.V., V.D. Ivanov, V.D., Kontorovich, V.K., Korytnik, S.A., Semenov, A.Y.: The non-Sibsonian interpolation: a new method of interpolation of the values of a function on an arbitrary set of points. Comput. Math. Math. Phys. **37**(1), 9–15 (1997)
2. Belyaev, A.: On transfinite barycentric coordinates. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing (SGP 2006), pp. 89–99 (2006)
3. Bobach, T., Bertram, M., Umlauf, G.: Issues and implementation of C^1 and C^2 natural neighbor interpolation. In: International Symposium on Visual Computing, pp. 186–195 (2006)
4. Bruvoll, S., Floater, M.S.: Transfinite mean value interpolation in general dimension. J. Comp. Appl. Math. **233**, 1631–1639 (2010)

5. Budninskiy, M., Liu, B., Tong, Y., Desbrun, M.: Power coordinates: A geometric construction of barycentric coordinates on convex polytopes. ACM Trans. Graph. **35**(6), 241:1–11 (2016)
6. Chen, R., Gotsman, C.: Complex transfinite barycentric mappings with similarity kernels. Comput. Graph. Forum **35**(5), 41–53 (2016). SGP 2016 Special Issue
7. Christ, N.H., Friedberg, R., Lee, T.D.: Weights of links and plaquettes in a random lattice. Nucl. Phys. B **210**(3), 337–346 (1982)
8. Dyken, C., Floater, M.S.: Transfinite mean value interpolation. Comput. Aided Geom. Des. **26**, 117–134 (2009)
9. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, vol. 95, pp. 173–182 (1995)
10. Floater, M.S.: Mean value coordinates. Comput. Aided Geom. Des. **20**(1), 19–27 (2003)
11. Floater, M.S.: Generalized barycentric coordinates and applications. Acta Numer. **24**, 161–214 (2015)
12. Floater, M.S., Kosinka, J.: Barycentric interpolation and mappings on smooth convex domains. In: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, pp. 111–116 (2010)
13. Floater, M.S., Patrizi, F.: Transfinite mean value interpolation over polygons (2019). arXiv:1906.08358
14. Floater, M.S., Kós, G., Reimers, M.: Mean value coordinates in 3D. Comput. Aided Geom. Des. **22**(7), 623–631 (2005)
15. Floater, M.S., Hormann, K., Kós, G.: A general construction of barycentric coordinates over convex polygons. Adv. Comput. Math. **24**(1–4), 311–331 (2006)
16. Hormann, K., Sukumar, N. (eds.): Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics. CRC Press, Boca Raton (2017)
17. Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. ACM Trans. Graph. **24**(3), 561–566 (2005). Proceedings of SIGGRAPH 2005
18. Ju, T., Liepa, P., Warren, J.: A general geometric construction of coordinates in a convex simplicial polytope. Comput. Aided Geom. Des. **24**(3), 161–178 (2007)
19. Kosinka, J., Barton, M.: Convergence of barycentric coordinates to barycentric kernels. Comput. Aided Geom. Des. **43**, 200–210 (2016)
20. MacNeal, R.H.: An asymmetrical finite difference network. Q. Appl. Math. **11**(3), 295–310 (1953)
21. Meyer, M., Lee, H., Barr, A., Desbrun, M.: Generalized barycentric coordinates on irregular polygons. J. Graph. Tools **7**(1), 13–22 (2002)
22. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Exp. Math. **2**(1), 15–36 (1993)
23. Schaefer, S., Ju, T., Warren, J.: A unified, integral construction for coordinates over closed curves. Comput. Aided Geom. Des. **24**(8-9), 481–493 (2007)
24. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference, pp. 517–524. ACM Press, New York (1968)
25. Sugihara, K.: Surface interpolation based on new local coordinates. Comput. Aided Des. **31**(1), 51–58 (1999)
26. Wachspress, E.L.: A Rational Finite Element Basis. Academic, New York (1975)
27. Warren, J.: Barycentric coordinates for convex polytopes. Adv. Comput. Math. **6**(2), 97–108 (1996)
28. Warren, J., Schaefer, S., Hirani, A., Desbrun, M.: Barycentric coordinates for convex sets. Adv. Comput. Math. **27**(3), 319–338 (2007)
29. Yan, Z., Schaefer, S.: A family of barycentric coordinates for co-dimension 1 manifolds with simplicial facets. Comput. Graph. Forum **38**(5), 75–83 (2019). SGP 2019 Special Issue

Part V
Numerical Methods

Fully-Implicit Collocated Finite-Volume Method for the Unsteady Incompressible Navier–Stokes Problem

Kirill M. Terekhov

Abstract This article introduces a collocated finite-volume method for the incompressible Navier–Stokes equations. Based on the linearity assumption of the velocity and pressure unknowns, the coupled one-sided flux expression is derived. Analysis and correction of the eigenvalues in the matrix coefficients of the vector flux expression result in the inf-sup stable method. A single continuous flux expression follows from the continuity of the one-sided flux approximations. As a result, the conservation for the momentum and the divergence is discretely exact. The method handles general polyhedral meshes but requires an artificial pressure boundary condition for the pressure gradient reconstruction.

1 Introduction

Previous work [12] introduced a fully-implicit collocated finite-volume method with the piecewise-constant approximation of the pressure field. The primary motivation for the fully-implicit approach is the coupling of a stiff cascade of reactions into a single system with the flow in the blood coagulation model [1, 18]. Current method investigates the improvement of the collocated finite-volume scheme from [12]. We consider piecewise-linear continuous approximation for the pressure field and full symmetric stress tensor in diffusion term.

The typical problem of the collocated methods is the inf-sup instability issue [6]. A usual solution to the problem is to use a staggered velocity arrangement [5, 7, 8]. However, with the staggered scheme, it is hard to retain the conservation properties [9]. Another solution is to use the Rhie-Chow interpolation method [10] with the collocated arrangement of unknowns. It is standard in industrial applications.

K. M. Terekhov (✉)
Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow, Russia

Moscow Institute of Physics and Technology, Moscow, Russia

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_23

This work uses the hypothesis that inf-sup stability is satisfied if all the matrix coefficients in the vector flux expression have positive eigenvalues. Previously we applied this hypothesis to the saddle-point mixed Darcy problem [15], the Navier–Stokes problem [12] and the multi-domain coupling of Darcy and poroelasticity equations [13].

We seek the solution to the system of Navier–Stokes equations:

$$\begin{cases} \dfrac{\partial \rho \mathbf{u}}{\partial t} + \operatorname{div}\left(\rho \mathbf{u}\mathbf{u}^T - \boldsymbol{\tau}(\mathbf{u}) + p\mathbb{I}\right) = \mathbf{f}, & \text{in } \Omega, \\ \operatorname{div}(\mathbf{u}) = 0, & \\ \text{B.C.} & \text{on } \partial\Omega. \end{cases} \tag{1}$$

The system (1) is closed with the appropriate boundary conditions, discussed in Sect. 5. In (1), $\mathbf{u} = [u, v, w]^T$ is the velocity vector, p is the pressure, u, v, w, $p \in H^1(\Omega)$, with appropriate amendment to the space at the boundary. $\rho = \text{const}$ is the fluid density, constant due to incompressibility, $\boldsymbol{\tau}(\mathbf{u})$ is the deviatoric stress tensor:

$$\boldsymbol{\tau}(\mathbf{u}) = 2\mu \mathbf{D}(\mathbf{u}), \quad \mathbf{D}(\mathbf{u}) = \frac{1}{2}\left[\nabla \mathbf{u}^T + \mathbf{u}\nabla^T\right], \tag{2}$$

here $\mu = \text{const}$ is the dynamic fluid viscosity. In the present work, we assume that the components of the velocity vector and the pressure are piecewise-linear.

In the finite volume method we make use of Gauss' formula for the divergence operator on each cell $V \in \mathcal{V}(\Omega)$, which results in:

$$\begin{aligned} \int_V \nabla \cdot \left(\rho \mathbf{u}\mathbf{u}^T - \boldsymbol{\tau}(\mathbf{u}) + p\mathbb{I}\right) \mathrm{d}V &\approx \sum_{\sigma \in \mathcal{F}(V)} |\sigma| \left[\rho \mathbf{u}\mathbf{u}^T - \boldsymbol{\tau}(\mathbf{u}) + p\right]\Big|_{\mathbf{x}_\sigma} \mathbf{n}_\sigma, \\ \int_V \nabla \cdot \mathbf{u} \, \mathrm{d}V &\approx \sum_{\sigma \in \mathcal{F}(V)} |\sigma| \left[\mathbf{u} \cdot \mathbf{n}_\sigma\right]|_{\mathbf{x}_\sigma}. \end{aligned} \tag{3}$$

Here $\mathbf{n}_\sigma$ is an average unit normal of face σ directed outwards of cell V. For briefness we further denote it by $\mathbf{n}$. Integrals of the other terms are evaluated as follows:

$$\int_V \frac{\partial \rho \mathbf{u}}{\partial t} \mathrm{d}V = |V| \left.\frac{\partial \rho \mathbf{u}}{\partial t}\right|_{\mathbf{x}_V} \approx \rho \frac{3\mathbf{u}^{n+1} - 4\mathbf{u}^n + \mathbf{u}^{n-1}}{\Delta t} |V|, \quad \int_V \mathbf{f} \mathrm{d}V = |V| \, \mathbf{f}|_{\mathbf{x}_V}. \tag{4}$$

Further, whenever the upper index for the velocity is omitted, it is implied $\mathbf{u} \equiv \mathbf{u}^{n+1}$.

2 The Interface Flux Approximation

Equations in (3) introduce the momentum $\mathbf{t}$ and continuity q fluxes:

$$\mathbf{t} = \rho \mathbf{u}\mathbf{u} \cdot \mathbf{n} - \boldsymbol{\tau}(\mathbf{u})\mathbf{n} + p\mathbf{n}, \quad q = \mathbf{u} \cdot \mathbf{n}, \tag{5}$$

here $\mathbf{t}$ is composed of the advection part $\mathbf{t}_A = \rho \mathbf{u}\mathbf{u} \cdot \mathbf{n}$, the traction part $\mathbf{t}_T = -\boldsymbol{\tau}(\mathbf{u})\mathbf{n}$ and the pressure part $\mathbf{t}_P = p\mathbf{n}$. The fluxes $\mathbf{t}$ and q are continuous on any surface, i.e., any mesh interface σ. Note, that due to the continuity equation, the velocity $\mathbf{u}$ is continuous throughout the domain.

Let $\mathbf{u}_1$ be the velocity vector collocated at $\mathbf{x}_1$, the barycenter of the cell V_1. Then, according to Taylor decomposition, the second-order approximations of the advection part $\mathbf{t}_A$ of the momentum flux at the interface barycenter $\mathbf{x}_\sigma$ is the following:

$$\begin{aligned} \mathbf{t}_A|_{\mathbf{x}_\sigma} &\approx \rho\, \mathbf{u}\mathbf{u} \cdot \mathbf{n}|_{\mathbf{x}_1} + \left.\frac{\rho \mathbf{u}\mathbf{u} \cdot \mathbf{n}}{\partial \mathbf{u}}\right|_{\mathbf{x}_1} \nabla \mathbf{u}|_{\mathbf{x}_1} (\mathbf{x}_\sigma - \mathbf{x}_1) \\ &\approx \frac{\rho}{2} \left(\mathbf{u}_1 \mathbf{n}^T + \mathbf{u}_1 \cdot \mathbf{n}\mathbb{I}\right) (2\mathbf{u}_\sigma - \mathbf{u}_1)\,. \end{aligned} \tag{6}$$

The approximation of the traction part $\mathbf{t}_T$ is formulated as follows:

$$\begin{aligned} \mathbf{t}_T|_{\mathbf{x}_\sigma} &\approx -\,\boldsymbol{\tau}(\mathbf{u})|_{\mathbf{x}_1}\, \mathbf{n} \approx -\mu \left(\mathbb{I} \otimes \mathbf{n}^T + \mathbf{n}^T \otimes \mathbb{I}\right) \mathbf{u} \otimes \nabla|_{\mathbf{x}_1} \\ &\approx \mu r_1^{-1} \left(\mathbb{I} + \mathbf{n}\mathbf{n}^T\right) (\mathbf{u}_1 - \mathbf{u}_\sigma) \\ &\quad - \mu \left(\mathbb{I} \otimes \mathbf{n}^T + \mathbf{n}^T \otimes \mathbb{I} - r_1^{-1} \left(\mathbb{I} + \mathbf{n}\mathbf{n}^T\right) \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T\right) \mathbf{u}_1 \otimes \nabla, \end{aligned} \tag{7}$$

where $r_1 = \mathbf{n} \cdot (\mathbf{x}_\sigma - \mathbf{x}_1)$ and $\otimes$ denotes the Kronecker product.

The pressure part of the momentum flux $\mathbf{t}_P$ and the continuity flux q form a saddle-point system:

$$\left.\begin{bmatrix} \mathbf{t}_P \\ q \end{bmatrix}\right|_{\mathbf{x}_\sigma} = \begin{bmatrix} p_\sigma \mathbf{n} \\ \mathbf{u}_\sigma \cdot \mathbf{n} \end{bmatrix} = \begin{bmatrix} & \mathbf{n} \\ \mathbf{n}^T & \end{bmatrix} \begin{bmatrix} \mathbf{u}_\sigma \\ p_\sigma \end{bmatrix} = S(\mathbf{n}) \begin{bmatrix} \mathbf{u}_\sigma \\ p_\sigma \end{bmatrix}, \tag{8}$$

here the indefinite 4×4 matrix $S(\mathbf{n})$ has two eigenvalues ± 1. The approximation to the combination of fluxes is:

$$\left.\begin{bmatrix} \mathbf{t}_P \\ q \end{bmatrix}\right|_{\mathbf{x}_\sigma} \approx A \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} - (A - S(\mathbf{n})) \begin{bmatrix} \mathbf{u}_\sigma \\ p_\sigma \end{bmatrix} + A \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla, \tag{9}$$

where 4×4 matrix $A > 0$ is used to stabilize the method. Let us introduce matrix A along with the other matrix coefficients:

$$A(\mathbf{u},\mathbf{v},\mathbf{n}) = \begin{bmatrix} a\left(\mathbb{I} + \mathbf{n}\mathbf{n}^T\right) & c\mathbf{n} \\ c\mathbf{n}^T & b \end{bmatrix}, \qquad W(\mathbf{n}) = \begin{bmatrix} \mu\left(\mathbb{I} \otimes \mathbf{n}^T + \mathbf{n}^T \otimes \mathbb{I}\right) \\ \end{bmatrix},$$
$$T(\mathbf{u},\mathbf{v},\mathbf{n}) = A(\mathbf{u},\mathbf{v},\mathbf{n}) + \begin{bmatrix} \mu r^{-1}\left(\mathbb{I} + \mathbf{n}\mathbf{n}^T\right) \\ \end{bmatrix}, \quad Q(\mathbf{u},\mathbf{n}) = \begin{bmatrix} \frac{\rho}{2}(\mathbf{u}\mathbf{n}^T + \mathbf{u}\cdot\mathbf{n}\mathbb{I}) \\ \end{bmatrix}, \tag{10}$$

where $r = \mathbf{n}\cdot\mathbf{v}$ and the parameters a, b and c depending on $\mathbf{u}$, $\mathbf{v}$, $\mathbf{n}$ are to be defined later in Sect. 3. Defining $T_1 = T(\mathbf{u}_1, \mathbf{x}_\sigma - \mathbf{x}_1, \mathbf{n})$, $Q_1 = Q(\mathbf{u}_1, \mathbf{n})$, $S_1 = S(\mathbf{n})$ and $W_1 = W(\mathbf{n})$ we get the coupled flux expression:

$$\begin{bmatrix} \mathbf{t} \\ q \end{bmatrix}\Big|_{\mathbf{x}_\sigma} \approx (T_1 - Q_1)\begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} - (T_1 - S_1 - 2Q_1)\begin{bmatrix} \mathbf{u}_\sigma \\ p_\sigma \end{bmatrix} + \left(T_1 \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T - W_1\right)\begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla. \tag{11}$$

3 Analysis of Eigenvalues of Matrix Coefficients

According to the hypothesis, the method is inf-sup stable, if all the matrix coefficients in the vector flux expression (11) have positive eigenvalues. The goal is to make eigenvalues positive by tuning parameters a, b and c with the minimal a and b. Let $\nu = \rho\mathbf{n}^T\mathbf{u}/2$. The eigenvalues of the matrix coefficient $T(\mathbf{u},\mathbf{v},\mathbf{n}) - S(\mathbf{n}) - 2Q(\mathbf{u},\mathbf{n})$ at the interface vector unknown are

$$\begin{aligned} \lambda_{1,2} &= a + \mu r^{-1} - 2\nu + b/2 \pm \sqrt{\left(a + \mu r^{-1} - 2\nu - b/2\right)^2 + (c-1)^2}, \\ \lambda_{3,4} &= a + \mu r^{-1} - 2\nu. \end{aligned} \tag{12}$$

The eigenvalues of the matrix coefficient $T(\mathbf{u},\mathbf{v},\mathbf{n}) - Q(\mathbf{u},\mathbf{n})$ at the cell vector unknown are

$$\begin{aligned} \lambda_{1,2} &= a + \mu r^{-1} - \nu + b/2 \pm \sqrt{\left(a + \mu r^{-1} - \nu - b/2\right)^2 + c^2}, \\ \lambda_{3,4} &= a + \mu r^{-1} - \nu. \end{aligned} \tag{13}$$

Requiring non-negativity of eigenvalues, we get a set of inequalities:

$$\begin{aligned} a+\mu r^{-1}-2\nu &\geq 0, & a+\mu r^{-1}-\nu &\geq 0,\\ 2b\left(a+\mu r^{-1}-2\nu\right) &\geq (c-1)^2, & 2b\left(a+\mu r^{-1}-\nu\right) &\geq c^2. \end{aligned} \tag{14}$$

The first two inequalities in (14) are satisfied with:

$$a=\max\left(2\nu-\mu r^{-1},0\right)+\theta>0, \tag{15}$$

here the parameter $\theta \geq 0$ is to be defined later. Let $q = a + \mu r^{-1} - 2\nu \geq 0$, note that $q + \nu \geq 0$. The second two inequalities in (14) are satisfied with the condition on b:

$$b \geq \max\left(\frac{1}{2q}(1-c)^2, \frac{1}{2(q+\nu)}c^2\right). \tag{16}$$

Expressions under max corresponds to a pair of parabolas, oriented upwards with $q > 0$ and $q + \nu > 0$. The minimum for b is realized at the intersection of the parabolas. The two parabolas have roots at $c = 1$ and $c = 0$, respectively. The minimum is reached at the intersection within $c \in [0, 1]$. To minimize b we solve:

$$\nu c^2 - 2(q+\nu)c + (q+\nu) = 0 \tag{17}$$

and select solutions corresponding to $c \in [0, 1]$ and corresponding value of b:

$$c=\begin{cases} 1+t-\sqrt{t+t^2}, & \nu>0,\\ 1/2, & \nu=0,\\ 1+t+\sqrt{t+t^2}, & \nu<0, \end{cases} \qquad b=\frac{1}{q}\begin{cases} t\left(1/2+t-\sqrt{t+t^2}\right), & \nu>0,\\ 1/8, & \nu=0,\\ t\left(1/2+t+\sqrt{t+t^2}\right), & \nu<0, \end{cases} \tag{18}$$

where $t = q/\nu$ and $bq \in [0, 1/2]$. Present work uses $\theta = \rho\sqrt{\mathbf{u}^T\left(\mathbb{I}-\mathbf{n}\mathbf{n}^T\right)\mathbf{u}+\varepsilon^2}$ with small $\varepsilon = 10^{-7}$. This choice helps prevent singular value of b due to zero q.

4 The Flux Continuity on Internal Face

Let us define $p_i, \nabla p_i, \mathbf{u}_i, \nabla\mathbf{u}_i$, pressure, velocity and the gradient of velocity at cell V_i barycenter $\mathbf{x}_i$. Let internal interface $\sigma \in \mathcal{F}(\Omega)$ share two cells V_1 and V_2, i.e., $\sigma = V_1 \cap V_2$, $V_1, V_2 \in \mathcal{V}(\Omega_h)$. We assume the normal $\mathbf{n}$ is oriented outside of V_1 into V_2.

Let $T_2 = T(\mathbf{u}_2, \mathbf{x}_\sigma - \mathbf{x}_2, -\mathbf{n})$, $Q_2 = Q(\mathbf{u}_2, -\mathbf{n})$ $S_2 = S(-\mathbf{n})$, $W_2 = W(-\mathbf{n})$ and $\Theta_2 = \Theta(\mathbf{x}_\sigma - \mathbf{x}_2, -\mathbf{n})$, then the approximation of the flux from cell V_2 reads as:

$$\left.\begin{bmatrix}\mathbf{t}\\ q\end{bmatrix}\right|_{\mathbf{x}_\sigma} \approx (T_2 - S_2 - 2Q_2)\begin{bmatrix}\mathbf{u}_\sigma\\ p_\sigma\end{bmatrix} - (T_2 - Q_2)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} - \left(T_2 \otimes (\mathbf{x}_\sigma - \mathbf{x}_2)^T - W_2\right)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} \otimes \nabla. \tag{19}$$

The reasoning for the eigenvalues and choice of parameters from Sect. 3 holds for matrix coefficients in (19) with the reversed normal direction. Enforcing the equality of flux approximations (11) and (19) we obtain the interface vector unknown:

$$\begin{bmatrix}\mathbf{u}_\sigma\\ p_\sigma\end{bmatrix} = (T_1 + T_2 - S_1 - S_2 - 2Q_1 - 2Q_2)^{-1} \times \begin{pmatrix} (T_1 - Q_1)\begin{bmatrix}\mathbf{u}_1\\ p_1\end{bmatrix} + \left(T_1 \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T - W_1\right)\begin{bmatrix}\mathbf{u}_1\\ p_1\end{bmatrix} \otimes \nabla \\ +(T_2 - Q_2)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} + \left(T_2 \otimes (\mathbf{x}_\sigma - \mathbf{x}_2)^T - W_2\right)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} \otimes \nabla \end{pmatrix}. \tag{20}$$

The continuous flux approximation is obtained by substituting (20) into either (11) or (19):

$$\begin{aligned}\left.\begin{bmatrix}\mathbf{t}\\ q\end{bmatrix}\right|_{\mathbf{x}_\sigma} \approx{}& (T_2 - S_2 - 2Q_2)\,(T_1 + T_2 - S_1 - S_2 - 2Q_1 - 2Q_2)^{-1} \\ &\times \left((T_1 - Q_1)\begin{pmatrix}\mathbf{u}_1\\ p_1\end{pmatrix} + \left(T_1 \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T - W_1\right)\begin{bmatrix}\mathbf{u}_1\\ p_1\end{bmatrix} \otimes \nabla\right) \\ &- (T_1 - S_1 - 2Q_2)\,(T_1 + T_2 - S_1 - S_2 - 2Q_1 - 2Q_2)^{-1} \\ &\times \left((T_2 - Q_2)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} + (T_2 \otimes (\mathbf{x}_\sigma - \mathbf{x}_2)^T - W_2)\begin{bmatrix}\mathbf{u}_2\\ p_2\end{bmatrix} \otimes \nabla\right).\end{aligned} \tag{21}$$

5 Boundary Conditions

The boundary conditions are prescribed by

$$\begin{cases} \mathbf{n}^T \left(\bar{\alpha}\mathbf{u} + \bar{\beta}\left(\boldsymbol{\tau}(\mathbf{u}) - p\mathbb{I}\right)\mathbf{n}\right)\Big|_{\mathbf{x}_\sigma} = \bar{r}, \\ \left(\mathbb{I} - \mathbf{n}\mathbf{n}^T\right)\left(\bar{\bar{\alpha}}\mathbf{u} + \bar{\bar{\beta}}\left(\boldsymbol{\tau}(\mathbf{u}) - p\mathbb{I}\right)\mathbf{n}\right)\Big|_{\mathbf{x}_\sigma} = \bar{\bar{\mathbf{r}}}, \end{cases} \tag{22}$$

The boundary conditions (22) are used to derive interface unknown vector. For this purpose the boundary conditions are augmented with the condition on pressure:

$$(\alpha p + \beta\mu\mathbf{n}\cdot\nabla p)|_{\mathbf{x}_\sigma} = r \tag{23}$$

In (23), $r = \alpha p_b + \beta\mu\xi$, with the prescribed boundary pressure p_b. For simplicity we use the pressure boundary condition $\xi = \mathbf{n}\cdot\nabla p^n$, proposed in [4]. Here p^n is the pressure at previous time step.

Coefficient choices for Dirichlet Γ_{D}, no-slip Γ_{NS}, slip condition Γ_{S}, traction-free Γ_{TF}, prescribed pressure Γ_{P} and Maxwell-Navier Γ_{MN} types of boundary conditions are presented in Table 1.

We combine (22) and (23) into a single block system to get

$$\begin{bmatrix} \mathbf{u}_\sigma \\ p_\sigma \end{bmatrix} = (D + NT_b)^{-1} \times \left(R + NT_b \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} + N\left(T_b \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T - W_b\right)\begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla \right), \tag{24}$$

Table 1 Coefficients for common types of boundary conditions

	Γ_{D}	Γ_{NS}	Γ_{S}	Γ_{TF}	Γ_{P}	Γ_{MN}
$\bar{\alpha}$	1	1	1	0	0	1
$\bar{\bar{\alpha}}$	1	1	0	0	0	λ
$\bar{\beta}$	0	0	0	1	1	0
$\bar{\bar{\beta}}$	0	0	1	1	1	1
α	0	0	0	0	1	0
β	1	1	1	1	0	1
$\bar{r}$	$\mathbf{n}\cdot\mathbf{u}_b$	0	0	0	$-p_b$	0
$\bar{\bar{\mathbf{r}}}$	$\mathbf{u}_b - \mathbf{n}\mathbf{n}\cdot\mathbf{u}_b$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
r	$\mathbf{n}\cdot\nabla p^n$	$\mathbf{n}\cdot\nabla p^n$	$\mathbf{n}\cdot\nabla p^n$	$\mathbf{n}\cdot\nabla p^n$	p_b	$\mathbf{n}\cdot\nabla p^n$

where

$$D = \begin{bmatrix} \bar{\bar{\alpha}}\mathbb{I} + \left(\bar{\alpha} - \bar{\bar{\alpha}}\right)\mathbf{n}\mathbf{n}^T & \bar{\beta}\mathbf{n} \\ & \alpha \end{bmatrix}, \; N = \begin{bmatrix} \bar{\bar{\beta}}\mathbb{I} + \left(\bar{\beta} - \bar{\bar{\beta}}\right)\mathbf{n}\mathbf{n}^T & \\ & \beta \end{bmatrix}, \; R = \begin{bmatrix} \mathbf{n}\bar{r} + \bar{\bar{\mathbf{r}}} \\ r \end{bmatrix}, \tag{25}$$

$T_b = \bar{T}(\mathbf{x}_\sigma - \mathbf{x}_1, \mathbf{n})$ and $W_b = \bar{W}(\mathbf{n})$, where

$$\bar{T}(\mathbf{v}, \mathbf{n}) = \begin{bmatrix} \mu r^{-1}\left(\mathbb{I} + \mathbf{n}\mathbf{n}^T\right) & \\ & \mu r^{-1} \end{bmatrix}, \quad \bar{W}(\mathbf{n}) = \begin{bmatrix} \mu\left(\mathbb{I} \otimes \mathbf{n}^T + \mathbf{n}^T \otimes \mathbb{I}\right) & \\ & \mu\mathbf{n}^T \end{bmatrix}. \tag{26}$$

Expression in (24) is used in (11) to evaluate the boundary flux. The matrix coefficient $(D + NT_b)$ is invertible for $\mu > 0$. A small $\varepsilon > 0$ can be added to μ in T_b to overcome the singularity without loss of the approximation property.

6 Gradient Reconstruction

To compute the fluxes numerically it remains to calculate the gradient at each cell V_i. Let us consider cell V_1 with the set of interfaces $\sigma \in \mathcal{F}(V_1)$. For the internal $\sigma \in \Omega \setminus \partial\Omega, \sigma = V_1 \cap V_2$ we impose the following condition on the gradient:

$$\mathbb{I} \otimes (\mathbf{x}_2 - \mathbf{x}_1)^T \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla = \begin{bmatrix} \mathbf{u}_2 \\ p_2 \end{bmatrix} - \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix}, \tag{27}$$

and on the boundary interface $\sigma \in \partial\Omega, \sigma = \partial\Omega \cap V_1$ we obtain from (24) the condition for the gradient:

$$\left(D \otimes (\mathbf{x}_\sigma - \mathbf{x}_1)^T + NW_b\right) \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla = R - D \begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \tag{28}$$

Assembling the conditions (27) and (28) over all interfaces of V_1 forms the linear system with matrix $A \in \Re^{4|\mathcal{F}(\sigma)| \times 12}$ and right-hand side $B \in \Re^{4|\mathcal{F}(\sigma)| \times 1}$. The non-square system is solved with the least-squares method by

$$\begin{bmatrix} \mathbf{u}_1 \\ p_1 \end{bmatrix} \otimes \nabla = \left(A^T A\right)^{-1} A^T B. \tag{29}$$

7 Problem Solution

On each iteration k of the Newton's method the right-hand side vector $\mathcal{R}^k$ of size $4|\mathcal{V}(\Omega)| \times 1$ and corresponding matrix $\mathcal{J}^k$ of size $4|\mathcal{V}(\Omega)| \times 4|\mathcal{V}(\Omega)|$ are assembled. The present work uses automatic differentiation for the sparse matrix assembly, as a result only the residual assembly is of concern. The assembly proceeds as follows:

1. Precompute the velocity and pressure gradients $[\mathbf{u}_i\ p_i]^T \otimes \nabla$ together with the derivatives on each cell $V_i \in \mathcal{V}(\Omega)$ by assembling and solving the system (29).
2. Assemble the residual of cell $V_i \in \mathcal{V}(\Omega)$ for the divergence terms as:

$$\mathcal{R}^k_{V_i} = \sum_{\sigma \in \mathcal{F}(V_i)} |\sigma| \left[\begin{matrix}\mathbf{t} \\ q\end{matrix}\right]\Bigg|_{\mathbf{x}_\sigma}, \tag{30}$$

   where the flux is evaluated according to (21) and (11)–(24) on internal and boundary σ, respectively.
3. For each cell $V_i \in \mathcal{V}(\Omega)$, subtract the right hand side and add the time evolution term to the residual $\mathcal{R}^k_{V_i}$ according to (4).

The velocity and pressure at the next Newton's iteration $k+1$ are obtained from the system:

$$\mathcal{J}^k \left(\left[\begin{matrix}\mathbf{u}^{k+1} \\ p^{k+1}\end{matrix}\right] - \left[\begin{matrix}\mathbf{u}^{k} \\ p^{k}\end{matrix}\right] \right) = \mathcal{R}^k, \tag{31}$$

and the next linear iteration continues with $\mathbf{u}^{k+1}$ and p^{k+1} until either relative τ_{rel} or absolute τ_{abs} tolerance is satisfied. The linear system is solved iteratively using BiConjugate Stabilized Gradient method with multi-level inverse-based second-order Crout-ILU preconditioner [14, 18]. The complete implementation is based on the INMOST toolkit for distributed mathematical modelling [16–18] which provides mesh handling, linear system assembly via automatic differentiation, linear system solution and visualization.

8 Numerical Tests

We consider the analytical solution of the unsteady Navier–Stokes equations suggested by Ethier and Steinman [3], where one may find the details on the reference solution. We introduce the lagrange multiplier and impose the constraint on pressure integral following [12] for unique pressure solution.

The problem is solved in the unit cube $\Omega \in [0, 1]^3$ in time interval $t \in [0, 1]$ for fluid with $\rho = 1$ and $\mathbf{f} = \mathbf{0}$ and Dirichlet boundary conditions, $\partial\Omega = \Gamma_{\text{D}}$. The

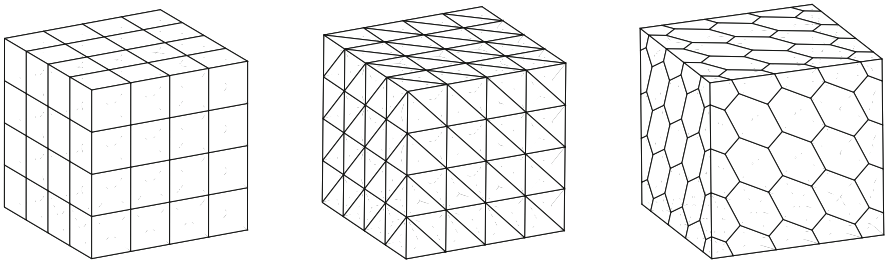


Fig. 1 Grids Ω_1 in the unit cube. Cubic (left), tetrahedral (middle), polyhedral (right) at *level* $= 1$

Table 2 Error norms for the Ethier-Steinman problem

level	$\|\mathbf{u}_h - \mathbf{u}\|_{L_2}$	$\|p_h - p\|_{L_2}$	$\|\mathbf{u}_h - \mathbf{u}\|_{L_2}$	$\|p_h - p\|_{L_2}$	$\|\mathbf{u}_h - \mathbf{u}\|_{L_2}$	$\|p_h - p\|_{L_2}$
	Cubic		Tetrahedral		Polyhedral	
$\mu = 10^{-1}$						
1	2.446×10^{-2}	1.234×10^{-1}	1.761×10^{-2}	5.516×10^{-2}	3.676×10^{-2}	2.471×10^{-1}
2	4.264×10^{-3}	1.383×10^{-2}	4.518×10^{-3}	1.764×10^{-2}	1.227×10^{-2}	6.055×10^{-2}
3	9.680×10^{-4}	3.880×10^{-3}	1.136×10^{-3}	7.124×10^{-3}	3.967×10^{-3}	1.465×10^{-2}
4	4.181×10^{-4}	1.528×10^{-3}	3.645×10^{-4}	3.171×10^{-3}	1.113×10^{-3}	3.704×10^{-3}
rate	1.211	1.344	1.971	1.168	1.916	2.033
$\mu = 10^{-5}$						
1	3.650×10^{-2}	2.460×10^{-1}	3.329×10^{-2}	9.499×10^{-2}	5.663×10^{-2}	4.746×10^{-1}
2	1.031×10^{-2}	3.391×10^{-2}	7.393×10^{-3}	1.799×10^{-2}	2.228×10^{-2}	1.147×10^{-1}
3	4.336×10^{-3}	6.361×10^{-3}	2.185×10^{-3}	3.739×10^{-3}	8.256×10^{-3}	2.555×10^{-2}
4	1.302×10^{-3}	1.724×10^{-3}	5.940×10^{-4}	9.752×10^{-4}	2.643×10^{-3}	6.320×10^{-3}
rate	1.736	1.883	1.879	1.924	1.717	2.106

boundary condition velocity $\mathbf{u}_b$ is defined from the reference solution at centers of faces. Initial velocity $\mathbf{u}^0$ at $t^0 = 0$ and previous velocity $\mathbf{u}^{-1}$ at $t^{-1} = -\Delta t$ are defined from the reference solution at barycenters of cells.

The problem is solved on three grid types depicted on Fig. 1. Four refinement *level*s for each mesh are considered. The time step is taken according to $\Delta t = 1/2^{(level-1)}$. The maximal *CFL* number on hexahedral mesh is $\sim$12. The computed errors for velocity and pressure with different viscosity μ are displayed in Table 2. The convergence *rate* in the tables is reported by the formula $rate = \log(\mathcal{E}_4/\mathcal{E}_3)/\log\big((|\mathcal{V}(\Omega_4)|/|\mathcal{V}(\Omega_3)|)^{1/3}\big)$, where $\mathcal{E}_{\text{level}}$ is the L_2-norm of the error and $|\mathcal{V}(\Omega_{\text{level}})|$ is the number of cells at given mesh refinement *level*. The number of cells for each grid type is reported in Table 3. Nonlinear problem on each time step requires up to three iterations to converge to the nonlinear tolerances $\tau_{\text{abs}} = 10^{-5}$ and $\tau_{\text{rel}} = 10^{-9}$. The observed reduction of the convergence *rate* at high viscosity can be attributed to the artificial boundary condition.

Next, we consider an unsteady flow around the cylinder with circular cross-section at Reynolds number 100. The setup is similar to the one from [11], with

Table 3 Number of cells for the grid refinement *level* in the Eshier-Steinman problem

Level	Cubic	Tetrahedral	Dual
1	64	384	125
2	512	3072	729
3	4096	24,576	4913
4	32,768	196,608	35,937

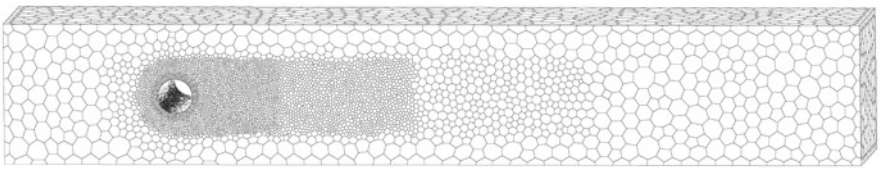


Fig. 2 The polyhedral grid for the experiment of the fluid flow around circular cylinder

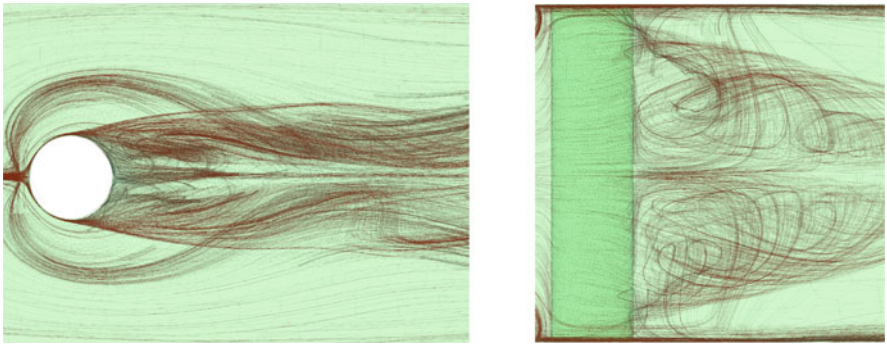


Fig. 3 Streamlines near the cylinder: side (left) and top (right) views

maximal inflow velocity $U = 2.25$. The density is $\rho = 1$ and dynamic viscosity is $\mu = 1/1000$. This experiment aims to reproduce a Karman vortex street on the mesh with a low spatial resolution. The mesh is demonstrated in Fig. 2. It is composed of 115,491 polyhedra and is aggressively coarsened away from the zone of the cylinder. The characteristic size $h = |V|^{1/3}$ of cells is in the range $h \in [0.0015, 0.08]$, time step size is $\Delta t = 0.005$ and $CFL \sim 4.5$.

As seen from Figs. 3 and 5 of the flow at 4 s, the method can reproduce the instability and the complex flow pattern in the fine region. However, the fine-scale effects diffuse away rapidly in the coarse region of the mesh. One may compare the result with the method from [12] that uses piecewise-constant pressure approximation in Fig. 4. In comparison, the new method is less dissipative and does not require a free parameter dependent on the global flow characteristics. In Table 4, the comparison to the results with the finite-volume method from [12] and the reference values of the drag and lift coefficients from [11] are provided. The drag and lift coefficients are computed by the volumetric integrals over the domain Ω, following [2]. The results show that the proposed method with piecewise-linear

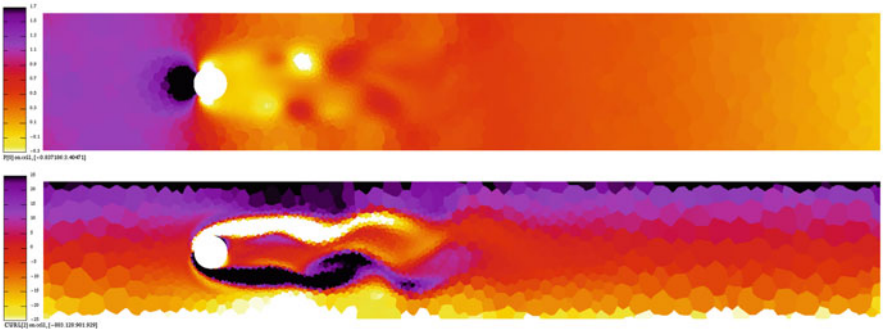


Fig. 4 Pressure field colored in the range $p \in [-0.3 : 1.7]$ (top) and z-component of the vorticity $(\nabla \times \mathbf{u})_z \in [-25 : 25]$ (bottom) in the middle cutaway of the mesh. Obtained with the method in the present article

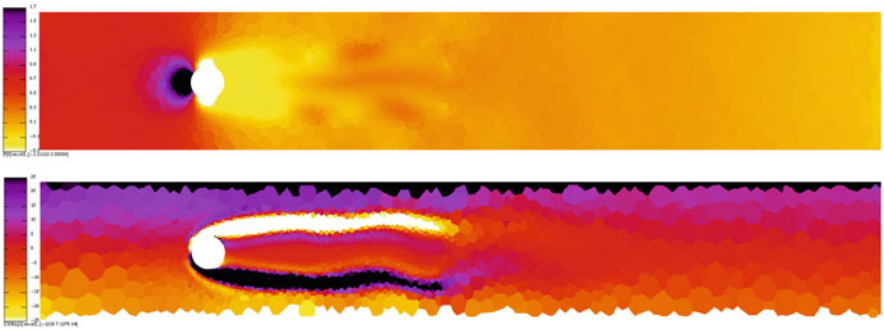


Fig. 5 Pressure field colored in the range $p \in [-0.3 : 1.7]$ (top) and z-component of the vorticity $(\nabla \times \mathbf{u})_z \in [-25 : 25]$ (bottom) in the middle cutaway of the mesh. Obtained with the method from [12] with piecewise-constant pressure approximation

Table 4 Estimated lift and drag coefficients and the Strouhal number for the flow around the cylinder with circular cross-section

Method	$\max(C_D)$	$-\max(C_L)$	St
Piecewise-linear pressure	3.303	0.011	0.29
Piecewise-constant pressure [12]	3.172	0.0219	0.305
Schäfer and Turek [11]	3.29–3.31	0.008–0.011	0.29–0.35

pressure approximation fits within the interval of reference values on the given grid, whereas the method from [12] with piecewise-constant pressure approximation is outside of the range (Fig. 5).

9 Conclusion

This work proposed a new collocated finite-volume method that features a simple flux-based construction. The method does not suffer from the inf-sup stability issue. In future work, we shall consider a comparison of different approaches to artificial pressure boundary condition and study the applicability of the method to non-Newtonian fluids.

Acknowledgement This work was supported by the Russian Science Foundation through the grant 19-71-10094.

References

1. Bouchnita, A., Terekhov, K., Nony, P., Vassilevski, Y., Volpert, V.: A mathematical model to quantify the effects of platelet count, shear rate, and injury size on the initiation of blood coagulation under venous flow conditions. PLOS One **15**(7), e0235392 (2020)
2. Braack, M., Richter, T.: Solutions of 3d Navier–Stokes benchmark problems with adaptive finite elements. Comput. Fluids **35**(4), 372–392 (2006)
3. Ethier, C., Steinman, D.: Exact fully 3D Navier–Stokes solutions for benchmarking. Int. J. Numer. Methods Fluids **19**(5), 369–375 (1994)
4. Gresho, P., Sani, R.: On pressure boundary conditions for the incompressible Navier–Stokes equations. Int. J. Numer. Methods Fluids **7**(10), 1111–1145 (1987)
5. Harlow, F., Welch, J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids **8**(12), 2182–2189 (1965)
6. Ladyzhenskaya, O.: The Mathematical Theory of Viscous Incompressible Flow, vol. 12. Gordon & Breach, New York (1969)
7. Lebedev, V.: Difference analogues of orthogonal decompositions, basic differential operators and some boundary problems of mathematical physics. I. USSR Comput. Math. Math. Phys. **4**(3), 69–92 (1964)
8. Olshanskii, M., Terekhov, K., Vassilevski, Y.: An octree-based solver for the incompressible Navier–Stokes equations with enhanced stability and low dissipation. Comput. Fluids **84**, 231–246 (2013)
9. Perot, B.: Conservation properties of unstructured staggered mesh schemes. J. Comput. Phys. **159**(1), 58–89 (2000)
10. Rhie, C., Chow, W.: Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA J. **21**(11), 1525–1532 (1983)
11. Schäfer, M., Turek, S., Durst, F., Krause, E., Rannacher, R.: Benchmark computations of laminar flow around a cylinder. In: Hirschel E.H. (eds.) Flow Simulation with High-Performance Computers II. Notes on Numerical Fluid Mechanics, vol. 48, pp. 547–566. Vieweg+Teubner Verlag, Berlin (1996)
12. Terekhov, K.: Collocated finite-volume method for the incompressible Navier-Stokes problem. J. Numer. Math. **1**(ahead-of-print) (2020)
13. Terekhov, K.: Multi-physics flux coupling for hydraulic fracturing modelling within INMOST platform. Russ. J. Numer. Anal. Math. Model. **35**(4), 223–237 (2020)
14. Terekhov, K.: Parallel multilevel linear solver within INMOST platform. In: Voevodin V., Sobolev S. (eds.) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science. Springer, Cham (2020)

15. Terekhov, K., Vassilevski, Y.: Finite volume method for coupled subsurface flow problems, I: Darcy problem. J. Comput. Phys. **395**, 298–306 (2019)
16. Terekhov, K., Vassilevski, Y.: INMOST parallel platform for mathematical modeling and applications. In: Voevodin V., Sobolev S. (eds.) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, vol. 965, pp. 230–241. Springer, Cham (2019)
17. Terekhov, K., Vassilevski, Y.: Mesh modification and adaptation within INMOST programming platform. In: Garanzha V., Kamenski L., Si H. (eds.) Numerical Geometry, Grid Generation and Scientific Computing. Lecture Notes in Computational Science and Engineering, vol. 131, pp. 243–255. Springer, Cham (2019)
18. Vassilevski, Y., Terekhov, K., Nikitin, K., Kapyrin, I.: Parallel Finite Volume Computation on General Meshes. Springer International Publishing, New York (2020). https://books.google.ru/books?id=hYvtDwAAQBAJ

Efficient Steady Flow Computations with Exponential Multigrid Methods

Shu-Jie Li

Abstract An exponential multigrid framework is developed and assessed with a modal high-order discontinuous Galerkin method in space. The algorithm based on a global coupling, exponential time integration scheme provides strong damping effects to accelerate the convergence towards the steady-state, while high-frequency, high-order spatial error modes are smoothed out with a s-stage preconditioned Runge-Kutta method. Numerical studies show that the exponential time integration substantially improves the damping and propagative efficiency of Runge-Kutta time-stepping for use with the p-multigrid method, yielding rapid and p-independent convergences to steady flows in both two and three dimensions.

1 Introduction

An important requirement for computational fluid dynamics is the capability to predict steady flows such as the case of flow past a body, so that key performance parameters, e.g., the lift and drag coefficients can be estimated. While the classical second-order methods are still being used extensively, high-order spatial discretizations attract more attention. For steady-state computations, most of the spatial discretizations have rested on the use of limited, traditional time discretizations combining with various acceleration methods. Recently, as an alternative to traditional time-marching methods, an exponential time integration scheme, the predictor-corrector exponential time-integrator scheme (PCEXP) [1–5] has been developed and successfully applied to the time stepping of fluid dynamics equations, exhibiting some advantages in terms of accuracy and efficiency for solving the fluid dynamics equations in both time-dependent and time-independent regimes.

In this paper, the exponential time integration is exploited in a multigrid framework which consists of an exponential time marching method and a s-stage

S.-J. Li (✉)
Beijing Computational Science Research Center, Beijing, China
e-mail: shujie@csrc.ac.cn

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_24

preconditioned Runge-Kutta method as an effective way to increase the feasibility of arbitrarily p-order DG for the high-order simulations of steady-state flows. The remainder of this paper is organized as follows. Section 2 presents the multigrid algorithm which combines two stand-alone methods in a V-cycle p-multigrid framework. Section 3 introduces the spatial discretization with a modal high-order DG method. Section 4 discusses how to evaluate the time steps in the p-multigrid framework. Section 5 presents the numerical results including two inviscid flow problems: (a) flow past a circular cylinder; (b) flow flow over a sphere. The numerical results obtained with the exponential p-multigrid method (eMG) are compared directly with a fully implicit method solved with the Incomplete LU preconditioned GMRES (ILU-GMRES) linear solver. Finally, Sect. 6 concludes this work. The Appendix provides the details of Jacobian matrices for the DG space discretization and time-step evaluations.

2 Exponential Multigrid Frame

In this section, a high-order (p) multigrid frame is detailed which is expected to have comparable performance to implicit methods for steady flow computations. The algorithm combines two stand-alone methods: the *exponential time integration* method and a s-stage *preconditioned Runge-Kutta* method. The two methods are introduced separately first and are finally integrated into a whole V-cycle multigrid frame.

2.1 Exponential Time Integration

We start with the following semi-discrete system of ordinary differential equations which may be obtained from a spatial discretization:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{R}(\mathbf{u}), \tag{1}$$

where $\mathbf{u} = \mathbf{u}(t) \in \mathbb{R}^K$ denotes the vector of the solution variables and $\mathbf{R}(\mathbf{u}) \in \mathbb{R}^K$ the right-hand-side term which may be the spatially discretized residual terms of the discontinuous Galerkin method used in this work. The dimension K is the degrees of freedom which can be very large for 3-D problems. Without loss of generality, we consider $\mathbf{u}(t)$ in the interval of one time step, i.e., $t \in [t_n,\ t_{n+1}]$. Applying the term splitting method [6] to (1) leads to a different exact expression

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{J}_n\mathbf{u} + \mathbf{N}(\mathbf{u}), \tag{2}$$

where the subscript n indicates the value evaluated at $t = t_n$, $\mathbf{J}_n$ denotes the Jacobian matrix $\mathbf{J}_n = \partial \mathbf{R}(\mathbf{u}) \big/ \partial \mathbf{u} \mid_{t=t_n} = \partial \mathbf{R}(\mathbf{u}_n) \big/ \partial \mathbf{u}$ and $\mathbf{N}(\mathbf{u}) = \mathbf{R}(\mathbf{u}) - \mathbf{J}_n \mathbf{u}$ denotes the remainder, which in general is nonlinear. Equation (2) admits the following formal solution:

$$\mathbf{u}_{n+1} = \exp(\Delta t \mathbf{J}_n)\mathbf{u}_n + \int_0^{\Delta t} \exp\left((\Delta t - \tau)\mathbf{J}_n\right)\mathbf{N}(\mathbf{u}(t_n + \tau))\, \mathrm{d}\tau\,, \tag{3}$$

where $\Delta t = t_{n+1} - t_n$ and

$$\exp(-t\mathbf{J}_n) = \sum_{m=0}^{\infty} \frac{(-t\mathbf{J}_n)^m}{m!}. \tag{4}$$

If using (3), the stiff linear term and the nonlinear integral term could be computed separately. The linear term could be computed analytically for some specialized equations but the nonlinear term is usually approximated numerically. In this paper, we use its equivalent form (5) instead, in which the linear and nonlinear terms are collected into a single term so that it can be efficiently approximated at one time.

Recently, a two-stage exponential scheme PCEXP [5] is shown to be effective for computing various flow regimes. However, using only the first stage of PCEXP, namely, the EXP1 scheme [1, 3] is shown to be more efficient for steady flows, although it is incapable of solving unsteady flows. Considering the constant approximation of the nonlinear term leads to the EXP1 scheme, namely

$$\mathbf{u}_{n+1} = \exp(\Delta t \mathbf{J}_n)\mathbf{u}_n + \Delta t \boldsymbol{\Phi}_1(\Delta t \mathbf{J}_n)\mathbf{N}_n = \mathbf{u}_n + \Delta t \boldsymbol{\Phi}_1(\Delta t \mathbf{J}_n)\mathbf{R}_n, \tag{5}$$

where

$$\boldsymbol{\Phi}_1(\Delta t \mathbf{J}) := \frac{\mathbf{J}^{-1}}{\Delta t}\big[\exp(\Delta t \mathbf{J}) - \mathbf{I}\big], \tag{6}$$

and $\mathbf{I}$ denotes the $K \times K$ identity matrix.

The physical nature of such type of exponential schemes relies on the global coupling feature via the global Jacobian matrix $\mathbf{J}$, so that flow transportation information can be broadcasted to the whole computational domain without a *CFL* restriction. That is why the exponential schemes behave like a fully implicit method but only depends on the current solution, i.e., in an explicit way as (5). Therefore, the EXP1 scheme is curiously exploited in the p-multigrid framework for steady flow computations.

2.2 Realization of EXP1 with the Krylov Method

The implementation of exponential time integration schemes requires evaluations of matrix-vector products, and in particular, the product of the exponential functions of the Jacobian and a vector, e.g., $\mathbf{\Phi}_1(\Delta t\mathbf{J}_n)\mathbf{N}$ in (5). They can be approximated efficiently using the Krylov method [7, 8]. Consider a m-dimensional Krylov subspace

$$\mathbb{K}_m(\mathbf{J},\mathbf{N}) = \text{span}\left\{\mathbf{N}, \mathbf{JN}, \mathbf{J}^2\mathbf{N}, \ldots, \mathbf{J}^{m-1}\mathbf{N}\right\}. \tag{7}$$

The orthogonal basis matrix $\mathbf{V}_m \coloneqq (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m) \in \mathbb{R}^{K\times m}$ satisfies the so-called Arnoldi decomposition [8]:

$$\mathbf{JV}_m = \mathbf{V}_{m+1}\widetilde{\mathbf{H}}_m, \tag{8}$$

where $\mathbf{V}_{m+1} \coloneqq (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m, \mathbf{v}_{m+1}) = (\mathbf{V}_m, \mathbf{v}_{m+1}) \in \mathbb{R}^{K\times(m+1)}$. The matrix $\widetilde{\mathbf{H}}_m$ is the $(m+1)\times m$ upper-Hessenberg matrix. Then (8) becomes

$$\mathbf{JV}_m = \mathbf{V}_m\mathbf{H}_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^{\mathrm{T}}. \tag{9}$$

Because $\mathbf{V}_m^{\mathrm{T}}\mathbf{V}_m = \mathbf{I}$, therefore

$$\mathbf{H}_m = \mathbf{V}_m^{\mathrm{T}}\mathbf{JV}_m, \tag{10}$$

$\mathbf{H}_m$ is thus the projection of the linear transformation of $\mathbf{J}$ onto the subspace $\mathbb{K}_m$ with the basis $\mathbb{V}_m$. Since $\mathbf{V}_m\mathbf{V}_m^{\mathrm{T}} \neq \mathbf{I}$, (10) leads to the following approximation:

$$\mathbf{J} \approx \mathbf{V}_m\mathbf{V}_m^{\mathrm{T}}\mathbf{JV}_m\mathbf{V}_m^{\mathrm{T}} = \mathbf{V}_m\mathbf{H}_m\mathbf{V}_m^{\mathrm{T}}, \tag{11}$$

and $\exp(\mathbf{J})$ can be approximated by $\exp\left(\mathbf{V}_m\mathbf{H}_m\mathbf{V}_m^{\mathrm{T}}\right)$ as below

$$\exp(\mathbf{J})\mathbf{N} \approx \exp\left(\mathbf{V}_m\mathbf{H}_m\mathbf{V}_m^{\mathrm{T}}\right)\mathbf{N} = \mathbf{V}_m\exp(\mathbf{H}_m)\mathbf{V}_m^{\mathrm{T}}\mathbf{N}. \tag{12}$$

The first column vector of $\mathbf{V}_m$ is $\mathbf{v}_1 = \mathbf{N}/\|\mathbf{N}\|_2$ and $\mathbf{V}_m^{\mathrm{T}}\mathbf{N} = \|\mathbf{N}\|_2\,\mathbf{e}_1$, thus (12) becomes

$$\exp(\mathbf{J})\mathbf{N} \approx \|\mathbf{N}\|_2\mathbf{V}_m\exp(\mathbf{H}_m)\mathbf{e}_1. \tag{13}$$

Consequently, $\mathbf{\Phi}_1$ can be approximated by

$$\begin{aligned}\mathbf{\Phi}_1(\Delta t\mathbf{J})\mathbf{N} &= \frac{1}{\Delta t}\int_0^{\Delta t}\exp((\Delta t-\tau)\mathbf{J})\mathbf{N}\,\mathrm{d}\tau \\ &\approx \frac{1}{\Delta t}\int_0^{\Delta t}\|\mathbf{N}\|_2\mathbf{V}_m\exp((\Delta t-\tau)\mathbf{H}_m)\mathbf{e}_1\,\mathrm{d}(\tau)\,.\end{aligned} \tag{14}$$

In general, the dimension of the Krylov subspace, m, is chosen to be much smaller than the dimension of $\mathbf{J}$, K, thus, $\mathbf{H}_m \in \mathbb{R}^{m\times m}$ can be inverted easily, so $\mathbf{\Phi}_1$ can be easily computed as the following

$$\begin{aligned}\mathbf{\Phi}_1(\Delta t\mathbf{J})\mathbf{N} &\approx \frac{1}{\Delta t}\|\mathbf{N}\|_2\mathbf{V}_m\int_0^{\Delta t}\exp((\Delta t-\tau)\mathbf{H}_m)\mathbf{e}_1\,\mathrm{d}\tau\\ &= \frac{1}{\Delta t}\|\mathbf{N}\|_2\mathbf{V}_m\mathbf{H}_m^{-1}\big[\exp(\Delta t\mathbf{H}_m)-\mathbf{I}\big]\mathbf{e}_1,\end{aligned} \tag{15}$$

where the matrix-exponential $\exp(\Delta t\mathbf{H}_m)$ can be computed efficiently by the Chebyshev rational approximation (cf., e.g., [8, 9]) due to the small size of $\mathbf{H}_m$.

2.3 Preconditioned Runge-Kutta Method

Consider a s-stage preconditioned Runge-Kutta (PRK) method of the following form

$$\begin{aligned}\mathbf{u}^{(0)} &= \mathbf{u}^n\\ \mathbf{u}^{(k)} &= \mathbf{u}^n + \beta_k\mathbf{P}^{-1}(\mathbf{u}^n)\mathbf{R}\Big(\mathbf{u}^{(k-1)}\Big), \quad k=1,2,\ldots,s,\\ \mathbf{u}^{n+1} &= \mathbf{u}^{(s)}\end{aligned} \tag{16}$$

where $\beta_k = 1/(s-k+1)$. $\mathbf{P}$ is taken as the diagonal part of the global residual Jacobian $\mathbf{J} = \partial\mathbf{R}/\partial\mathbf{u}$, representing the element-wise wave propagation information. $s = 4$ is used for all the test cases of this work.

The physical nature of this type of RK method can be interpreted in two different views which are helpful for us to see how does PRK make sense. First, we consider the first-order spatial discretization of finite volume or discontinuous Galerkin method to the i-th element surrounded by adjoined cells j ($1 \le j \le N$) with the inter-cell surface area S_{ij}, and the spatial residual using a upwinding flux can be written as

$$V_i\frac{\Delta\mathbf{u}_i}{\Delta t} = \mathbf{R}_i = \sum_{j=1}^{N}\frac{1}{2}\big[\mathbf{F}(\mathbf{u}_i)+\mathbf{F}(\mathbf{u}_j)\big]\mathbf{n}_{ij}\mathrm{S}_{ij} + \frac{1}{2}|\mathbf{A}_{ij}^n|\,(\mathbf{u}_i-\mathbf{u}_j)\mathrm{S}_{ij}. \tag{17}$$

So $\mathbf{P}$ can be derived as

$$\mathbf{P}_i = \frac{\partial\mathbf{R}_i}{\partial\mathbf{u}_i} \approx \sum_{j=1}^{N}\frac{1}{2}|\mathbf{A}_{ij}^n|\,\mathrm{S}_{ij}. \tag{18}$$

A matrix $\mathbf{\Delta t}$ can be defined as

$$\mathbf{\Delta t} = V_i \mathbf{P}^{-1} = \frac{V_i}{\sum_{j=1}^{N} \frac{1}{2} |\mathbf{A}_{ij}^n| \, \mathrm{S}_{ij}}. \tag{19}$$

One can uncover the relationship between the matrix $\mathbf{\Delta t}$ and the traditional definition of time step by considering a cell-constant scalar spectral radius approximation $\lambda_{ij}^{\max}$ to $|\mathbf{A}_{ij}^n|$, i.e.,

$$\Delta t = \frac{2V_i}{\sum_{j=1}^{N} \lambda_{ij}^{\max} \mathrm{S}_{ij}} \xrightarrow{1D} \frac{\Delta x_i}{\lambda_i^{\max}} \tag{20}$$

Therefore, $\mathbf{P}^{-1}$ is equivalent to a matrix time step and it is consistent to the usual definition of time step in the scalar case. As demonstrated in [10], this matrix is a kind of preconditioner which can provide effective clustering of convective eigenvalues and substantial improvements to the convergence of RK time-stepping. In this work, different from [10], an exact way of evaluating matrix time steps with exact Jacobian is proposed in [3] so that all the stiffness effects from spatial discretizations and boundary conditions can be exactly taken into account. Note that such a matrix time stepping has no temporal order of accuracy, since the traditional scalar, physical time step dose not show up at all. Therefore, the PRK scheme has no temporal order of accuracy. The formula of PRK scheme can be considered as a simplified fully implicit scheme or implicit-explicit Runge Kutta methods, but has it own physical significance. To increase robustness, we recommend to increase diagonal domination to $\mathbf{P}$, namely, $\mathbf{P} = \partial \mathbf{R} / \partial \mathbf{u} + \mathbf{I}/\delta\tau$, where $\delta\tau$ is a pseudo time step which is computed by (32).

2.4 The V-Cycle p-Multigrid Framework

The use of p-multigrid smoother with explicit RK or preconditioned RK methods is observed inefficient at eliminating low-frequency error modes at lower orders of accuracy. To provide a better smoother with stronger damping effects, the EXP1 scheme that exhibits fast convergence rates for Euler and Navier-Stokes equations is considered. Unlike the explicit RK smoother that only produces weak damping effects in a local, point-wise manner, the exponential scheme is a global method that allows large time steps with strong damping effects to all the frequency modes across the computational domain, as shown in the previous works [3].

In the exponential p-multigrid method (eMG), the EXP1 scheme is utilized on the accuracy level $p = 0$ and the PRK method is used for accuracy levels $p > 0$, contributing both memory deduction and efficiency enhancement. The smoothing employs a V-cycle p-multigrid process, where a two-level algorithm is recursively used. To illustrate the algorithm, let us consider a nonlinear problem $\mathbf{A}(\mathbf{u}^p) = \mathbf{p}^p$,

where $\mathbf{u}^p$ is the solution vector, $\mathbf{A}(\mathbf{u}^p)$ is the nonlinear operator and p denotes the accuracy level of DG. Let $\mathbf{v}^p$ be an approximation to the solution vector $\mathbf{u}^p$ and define the residual $\mathbf{r}(\mathbf{v}^p)$ by

$$\mathbf{r}(\mathbf{v}^p) = \mathbf{f}^p - \mathbf{A}^p(\mathbf{v}^p).$$

In the eMG framework, the solution on the $p-1$ level is used to correct the solution of p level in the following steps:

1. Conduct a time stepping with the PRK scheme on the highest accuracy level $p_{\max}$.
2. Restrict the solution and the residual of p to the $p-1$ level ($1 \le p \le p_{\max}$)

$$\mathbf{v}_0^{p-1} = \mathcal{R}_p^{p-1}\mathbf{v}^p, \quad \mathbf{r}^{p-1} = \mathcal{R}_p^{p-1}\mathbf{r}^p(\mathbf{v}^p), \tag{21}$$

   where $\mathcal{R}_p^{p-1}$ is the restriction operator from the level p to the level $p-1$.
3. Compute the forcing term for the $p-1$ level

$$\mathbf{s}^{p-1} = \mathbf{A}^{p-1}(\mathbf{v}_0^{p-1}) - \mathbf{r}^{p-1}. \tag{22}$$

4. Smooth the solution with the PRK scheme on the $p-1$ level but switch to use the EXP1 scheme on the lowest accuracy level $p = 0$,

$$\mathbf{A}^{p-1}(\mathbf{v}^{p-1}) = \mathcal{R}_p^{p-1}\mathbf{f}^p + \mathbf{s}^{p-1}. \tag{23}$$

5. Evaluate the error of level $p-1$

$$\mathbf{e}^{p-1} = \mathbf{v}^{p-1} - \mathbf{v}_0^{p-1}. \tag{24}$$

6. Prolongate the $p-1$ error and correct the approximation of level p

$$\mathbf{v}^p = \mathbf{v}^p + \mathcal{P}_{p-1}^p\mathbf{e}^{p-1}, \tag{25}$$

   where $\mathcal{P}_{p-1}^p$ is the prolongation operator.

3 Spatial Discretization

In this paper, the eMG method is applied to solve three-dimensional Euler equations discretized by a modal discontinuous Galerkin method. Consider the Euler equations in three-dimensional space

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0, \tag{26}$$

where $\mathbf{U}$ stands for the vector of conservative variables, $\mathbf{F}$ denotes the convective flux

$$\mathbf{U} = \begin{pmatrix} \varrho \\ \varrho \boldsymbol{v} \\ \varrho E \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \varrho \boldsymbol{v}^{\mathrm{T}} \\ \varrho \boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} + p\mathbf{I} \\ \varrho\, H\, \boldsymbol{v}^{\mathrm{T}} \end{pmatrix}, \tag{27}$$

where $\boldsymbol{v} = (u, v, w)^{\mathrm{T}}$ is the absolute velocity, ϱ, p, and e denote the flow density, pressure, and the specific internal energy; $E = e + \frac{1}{2}\|\boldsymbol{v}\|^2$ and $H = E + p/\varrho$ denote the total energy and total enthalpy, respectively; $\mathbf{I}$ denotes the 3×3 unit matrix; and the pressure p is given by the equation of state for a perfect gas

$$p = \varrho(\gamma - 1)e, \tag{28}$$

where $\gamma = 7/5$ is the ratio of specific heats for perfect gas.

3.1 Modal Discontinuous Galerkin Method

Considering a computational domain Ω divided into a set of non-overlapping elements of arbitrary shape, the modal discontinuous Galerkin method [3] seeks an approximation $\mathbf{U}_h$ in each element $E \in \Omega$ with finite-dimensional space of polynomial P^p of order p in the discontinuous finite element space

$$\mathbb{V}_h := \left\{ \psi_i \in L^2(\Omega) : \psi_i|_E \in P^p(\Omega),\ \forall E \in \Omega \right\}. \tag{29}$$

The numerical solution of $\mathbf{U}_h$ can be approximated in the finite element space $\mathbb{V}_h$

$$\mathbf{U}_h(\mathbf{x},\ t) = \sum_{j=1}^{n} \mathbf{u}_j(t)\psi_j(\mathbf{x}). \tag{30}$$

In the weak formulation, the Euler equations (26) in an element E becomes:

$$\int_E \psi_i \psi_j \,\mathrm{d}\mathbf{x} \frac{\mathrm{d}\mathbf{u}_j}{\mathrm{d}t} = -\int_{\partial E} \psi_i \widetilde{\mathbf{F}} \cdot \hat{\mathbf{n}} \,\mathrm{d}\boldsymbol{\sigma} + \int_E \mathbf{F} \cdot \nabla \psi_i \,\mathrm{d}\mathbf{x} := \mathbf{R}_i, \tag{31}$$

where $\hat{\mathbf{n}}$ is the out-normal unit vector of the surface element $\boldsymbol{\sigma}$ with respect to the element E, $\widetilde{\mathbf{F}}$ is the Riemann solver [11], which will be approximated by Roe scheme [12], and the Einstein summation convention is used. For an orthonormal basis $\{\psi_i\}$, the term on the left-hand side of (31) becomes diagonal, so the system is in the standard ODE form of (1), thus avoiding solving a a linear system as required for a non-orthogonal basis. More importantly, using an orthogonal basis would yield more accurate solutions, especially for high-order methods, e.g., $p = 6$.

4 Time-Stepping Strategy

In this section, the time-stepping strategy of the eMG framework is discussed as a time-marching solver to compute the steady solutions of the Euler equations. There are two different time steps needed to be determined. One for the PRK time stepping $\delta\tau$ and the other for EXP1 smoothing which is empirically chosen as large as $(p_{\max}+1)\delta\tau$. As such, only $\delta\tau$ should be determined. $\delta\tau$ is determined by

$$\delta\tau = \frac{CFL\, h_{3\mathrm{D}}}{(2p+1)(\|\boldsymbol{v}\|+c)}, \quad h_{3\mathrm{D}} \coloneqq 2d\frac{|E|}{|\partial E|}, \tag{32}$$

where CFL is the global Courant-Friedrichs-Lewy (CFL) number, p the accuracy level, $\boldsymbol{v}$ the velocity vector at the cell center, c the speed of sound, d the spatial dimension, $|E|$ and $|\partial E|$ are the volume and the surface area of the boundary of E, respectively; and $h_{3\mathrm{D}}$ represents a characteristic size of a cell in 3D defined by the ratio of its volume and surface area. All the methods mentioned in this paper have been implemented in the HA3D flow solver developed by the author, which is for solving three-dimensional problems as its name indicates. So to support 2-D computations, a 2-D mesh is extruded to a 3-D (quasi-2D) mesh by one layer of cells and we use $h_{2\mathrm{D}}$ instead of $h_{3\mathrm{D}}$ to eliminate the effect of the z dimension on obtaining the truly 2-D time step. Given the cell size Δz in the z direction, $h_{2\mathrm{D}}$ is determined by

$$\frac{2}{h_{2\mathrm{D}}} = \frac{3}{h_{3\mathrm{D}}} - \frac{1}{\Delta z}. \tag{33}$$

To enhance the computational efficiency for the steady problems, the CFL number of both schemes are dynamically determined by the following formula

$$CFL_n = \min\left\{CFL_{\max}, \max\left[\|R(\varrho_n)\|_2^{-1},\ 1+\frac{(n-1)}{(2p+1)}\right]\right\}, \tag{34a}$$

$$\|R(\varrho_n)\|_2 \coloneqq \frac{1}{|\Omega|}\left[\int_\Omega R(\varrho_n)^2\,\mathrm{d}\mathbf{x}\right]^{1/2}, \tag{34b}$$

where $R(\varrho_n)$ denotes the residual of density, $CFL_{\max}$ is the user-defined maximal CFL number, n is the number of iterations, and p is the spatial order of accuracy. Such a variable CFL evolution strategy allows a robust code startup and good overall computational efficiency in practices. In all the test cases considered, the upper-bound CFL number of (34a) is taken as follows: $CFL_{\max} = 10^3$ for the implicit BE method; $CFL_{\max} = 10^2$ for the eMG method.

5 Numerical Results

In this section, we focus on the investigations of performance and convergence of the eMG framework. The feasibility is of eMG is demonstrated and compared to a fast fully implicit ILU preconditioned GMRES method. Two external flow cases are considered: flows past a circular cylinder in quasi-2D and a sphere in 3D. Since both eMG and GMRES are based on the Krylov subspace, the same Krylov space parameters are used. The Krylov subspace dimension m is 30 and the tolerance of Krylov subspace approximation error is 10^{-5}, see Saad's classic works [8, 13] for more details about the error estimations. The fastest first-order backward Euler (BE) discretization solved with the ILU preconditioned GMRES linear solver is used as the performance reference solution.

5.1 Flow Over a Circular Cylinder in Quasi-2D

In this case, the results obtained for flow over a circular cylinder at Mach number $Ma = 0.3$ is presented. The cylinder has a radius of 1 and is surrounded in a circular computational domain of radius 15, as shown in Fig. 1. The quasi-2D mesh with $128 \times 32 = 4096$ quadratic curved hexahedral elements is generated by extruding the 2D mesh by one layer of grids. The final 3D mesh is used by the HA3D arbitrarily high-order discontinuous Galerkin flow solver[1–5]. The total degree of freedoms is up to 81,920 with DG at $p = 3$.

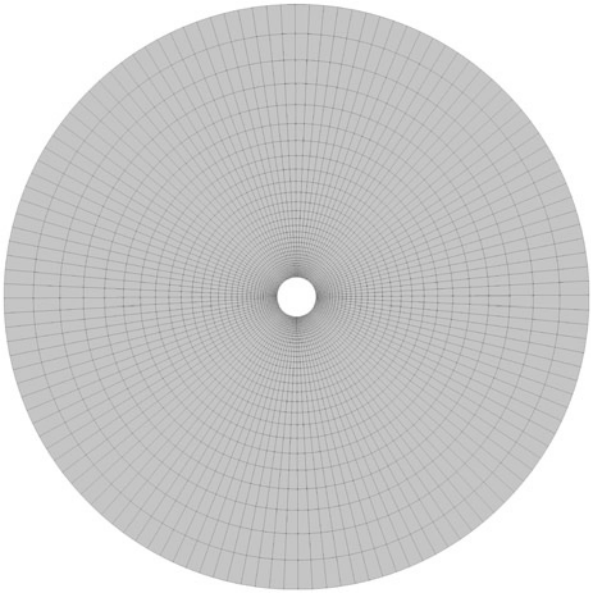


Fig. 1 Flow over a circular cylinder in quasi-2D: $128 \times 32 = 4096$ quadratic curved elements

In Fig. 2, the L_2 norm of density residual $R(\varrho_n)$ is plotted versus the iteration by using the eMG scheme, indicating convergence rates independent of spatial order of accuracy p, or say p-independent. The results obtained with a fast, implicit ILU preconditioned GMRES is computed in Fig. 3, which shows the convergence

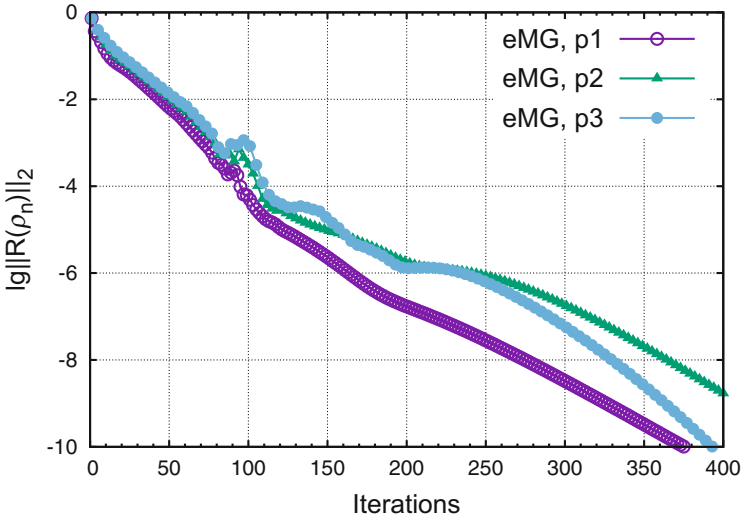


Fig. 2 Flow over a circular cylinder in quasi-2D: p-independent convergences with the eMG method

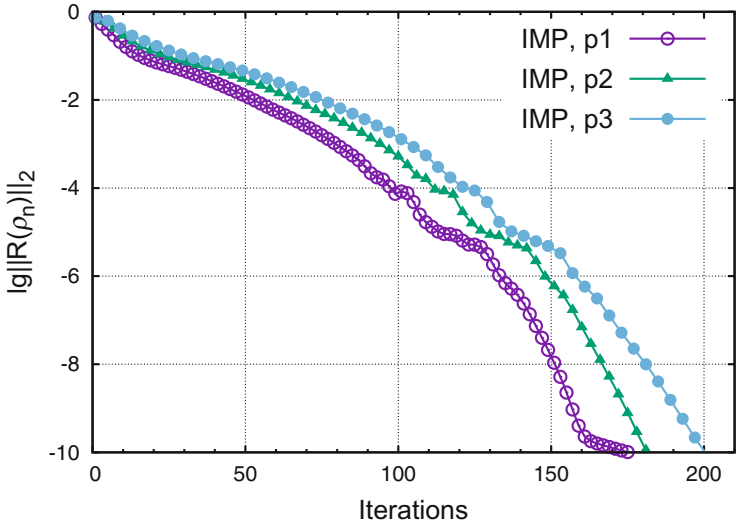


Fig. 3 Flow over a circular cylinder in quasi-2D: Convergence histories of the implicit method with varying spatial accuracy

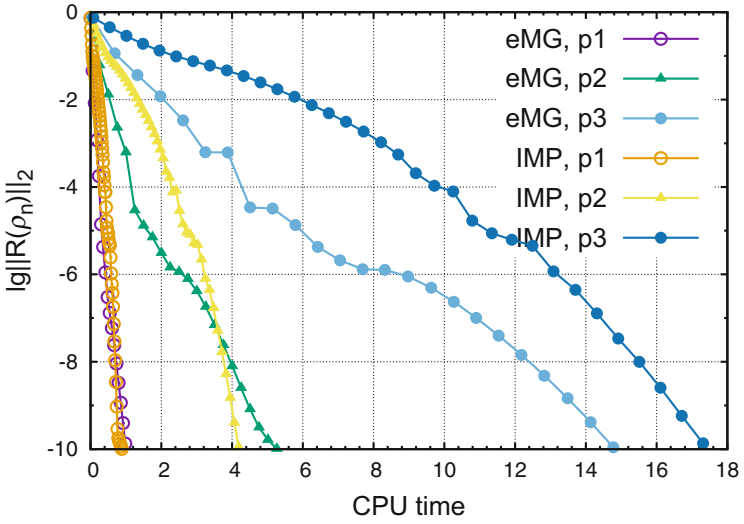


Fig. 4 Flow over a circular cylinder in quasi-2D: Performance comparison between the eMG method and the implicit method at different spatial accuracy

histories of the implicit method with varying spatial accuracy. The results show rapid quadratic Newton convergences which are dependent on the spatial order of accuracy p. To see how promising is the eMG performance comparing with the fully implicit method, the two results are compared in Fig. 4, where the CPU time is normalized by that of the eMG scheme. As we can see, the implicit method (IMP) is faster for $p = 1, 2$ cases but is slower than the eMG scheme for the $p = 3$ case. So for high-order computations, the eMG method is at least comparable to the implicit method in terms of overall performance.

5.2 Flow Over a Sphere in 3D

The computational efficiency of the eMG scheme is investigated for the three-dimensional flow past a sphere with the Mach number $Ma = 0.3$. The radius of the sphere is 1 and the far-field spherical radius is 5. The sphere surface is set as a slip wall boundary condition, and the outer boundary uses a far-field characteristic boundary condition with Riemann invariants. The mesh respects the flow symmetries of the horizontal and vertical planes, on which a symmetry boundary condition is imposed. The generated curved mesh consists of 9778 tetrahedrons and 4248 prisms, 14,026 cells in total. A close-up view of the mesh about the sphere and the velocity contour computed with the eMG scheme at $p = 3$ is illustrated in Fig. 5.

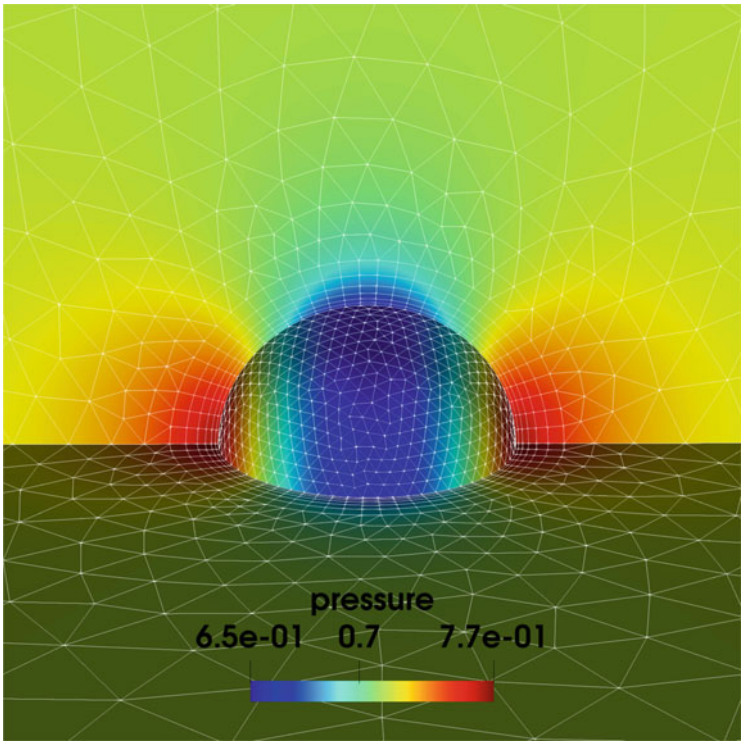


Fig. 5 Flow contour computed for the flow past a sphere at $Ma = 0.3$ with eMG and DG $p = 3$

Figure 6 shows the convergence histories of the eMG method for spatial order of accuracy $p = 1, 2, 3$. Again, p-independent convergences do appear. In Fig. 7, convergence histories of the implicit method (IMP) are shown with iteration counts. Figure 8 compares both methods measured in CPU time. As one can see that although IMP is fast in terms of iteration counts, the computational cost per iteration is relatively high and the resulting CPU time is penalized. When using high-order spatial schemes along with an implicit method, the high-order global Jacobian matrix consumes a large amount of memory. The most significant part of memory usage (M) of the two methods eMG and IMP are compared as follows

$$\begin{aligned} \mathrm{M}_{\mathrm{emg}} &= \mathrm{NE}\left[\frac{5}{3}(p+1)(p+2)(p+3) + 150\right], \\ \mathrm{M}_{\mathrm{imp}} &= 6\mathrm{NE}\left[\frac{5}{6}(p+1)(p+2)(p+3)\right]^2. \end{aligned} \tag{35}$$

For problems sized up to $\mathrm{NE} = 10^5$ elements at $p = 3$, fourth-order spatial accuracy, a fully implicit method requires 45 GB memory only for storing the

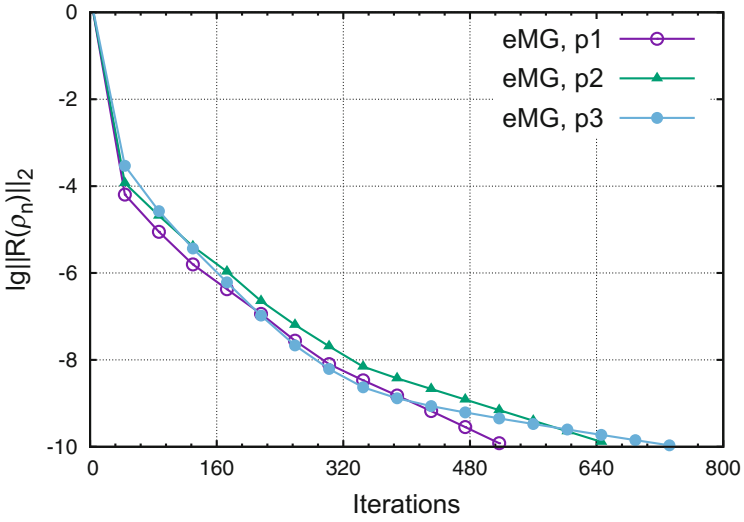


Fig. 6 Flow over a sphere in 3D: p-independent convergences with the eMG method

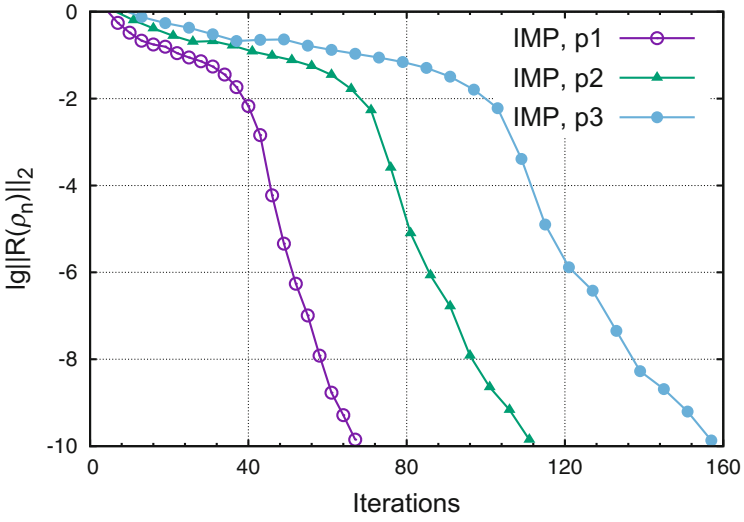


Fig. 7 Flow over a sphere in 3D: convergence histories of the implicit method with varying spatial accuracy

Jacobian matrix, while eMG only requires 0.03 GB memory for storing the solution vectors plus the first-order Jacobian matrix for the same sized problem. Therefore, the eMG method is far more memory friendly compared with a fully implicit method, providing a more practical while efficient strategy for solving steady problems with high-order methods.

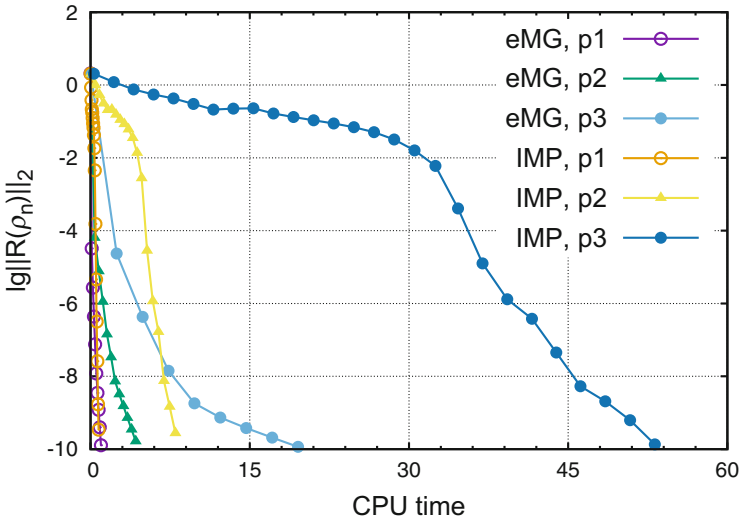


Fig. 8 Flow over a sphere in 3D: performance comparison between the eMG method and the implicit method at different spatial accuracy

6 Conclusions

The first-order exponential time integration scheme, EXP1 has been exploited to increase the feasibility of arbitrarily p-order DG for high-order simulations of steady-state flows. The algorithms and the physical natures of the methods are presented. The performance and memory usage are investigated and compared with a fast backward-Euler ILU preconditioned GMRES fully implicit method for 2-D and 3-D steady flow problems. The results show expected p-independent convergence rates for $p = 1, 2, 3$ order of accuracy and the eMG method is up to 20 times faster than the $p = 3$ ILU-GMRES for the 3-D case. Comparing to the fully implicit method, the eMG framework uses less memory but achieves comparable computational efficiency. Although the eMG method shows promising results, the PRK scheme inside has a time step restriction and thus affects the overall efficiency. Further studies are needed to pursue a better high-frequency smoother and a more efficient exponential scheme as well.

Acknowledgments This work is funded by the National Natural Science Foundation of China (NSFC) under the Grant U1930402. The computational resources are provided by Beijing Computational Science Research Center (CSRC).

References

1. Li, S.-J., Wang, Z.J., Ju, L., Luo, L.-S.: Explicit large time stepping with a second-order exponential time integrator scheme for unsteady and steady flows. In: 55th AIAA Aerospace Sciences Meeting, AIAA-2017-0753 (2017)
2. Li, S.-J., Wang, Z.J., Ju, L., Luo, L.-S.: Fast time integration of Navier-Stokes equations with an exponential-integrator scheme. In: 2018 AIAA Aerospace Sciences Meeting, AIAA-2018-0369 (2018)
3. Li, S.-J., Luo, L.-S., Wang, Z.J., Ju, L.: An exponential time-integrator scheme for steady and unsteady inviscid flows. J. Comput. Phys. **365**, 206–225 (2018)
4. Li, S.-J.: Mesh curving and refinement based on cubic Bézier surface for High-order discontinuous Galerkin methods. Comput. Math. Math. Phys. **59**(12), 2080–2092 (2019)
5. Li, S.-J., Ju, L., Si, H.: Adaptive exponential time integration of the Navier-Stokes equations. In: AIAA-2020-2033 (2020)
6. Caliari, M., Ostermann, A.: Implementation of exponential Rosenbrock-type integrators, Appl. Numer. Math. **59**(3), 568–582 (2009)
7. Tokman, M., Loffeld, J.: Efficient design of exponential-Krylov integrators for large scale computing. Procedia Comput. Sci. **1**(1), 229–237 (2010)
8. Saad, Y.: Analysis of some Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **29**(1), 209–228 (1992)
9. Moler, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM J. Numer. Anal. **45**(1), 3–49 (2003)
10. Pierce, N., Giles, M.: Preconditioning compressible flow calculations on stretched meshes. In: 34th Aerospace Sciences Meeting and Exhibit, AIAA-1996-889 (1996)
11. Toro, E.: Riemann Solvers and Numerical Methods for Fluid Dynamics. Springer, Berlin (1999)
12. Roe, P.: Approximate Riemann solvers, parameter vectors, and difference schemes. J. Comput. Phys. **43**(2), 357–372 (1981)
13. Saad,Y., GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**(3), 856–869 (1986)

Exponential Time Integrators for Unsteady Advection–Diffusion Problems on Refined Meshes

Mikhail A. Botchev

Abstract Time integration of advection dominated advection–diffusion problems on refined meshes can be a challenging task, since local refinement can lead to a severe time step restriction, whereas standard implicit time stepping is usually hardly suitable for treating advection terms. We show that exponential time integrators can be an efficient, yet conceptually simple, option in this case. Our comparison includes three exponential integrators and one conventional scheme, the two-stage Rosenbrock method ROS2 which has been a popular alternative to splitting methods for solving advection–diffusion problems.

1 Introduction

Time integration of unsteady advection–diffusion problems discretized in space on locally refined meshes can be a challenging problem. This is especially the case for advection dominated problems. On the one hand, requirements of accuracy, monotonicity and total variation diminishing (TVD) usually rule out the use of implicit time integration for advection terms [23, Chapter III.1.3] (for a notable exception see [31]). On the other hand, locally refined meshes can impose a severe CFL stability restriction on the time step, thus making explicit schemes very inefficient.

Within the method of lines framework, i.e., when discretization in space is followed by time integration, different approaches exist to cope with this problem. A straightforward and widely used approach is operator splitting [23, Chapter IV],[43, Chapter 3], when advection is usually treated explicitly in time and diffusion implicitly. Though being conceptually simple and easy to apply in practice, splitting

M. A. Botchev (✉)
Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, Russia

Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russia
e-mail: botchev@ya.ru

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_25

methods unavoidably lead to splitting errors, see, e.g., [10, 27]. Moreover, a proper use of boundary conditions within the splitting is sometimes not trivial and may lead to error order reduction [15, 38]. To reduce splitting errors many various approaches have been proposed, with none being fully successful. Here we mention source splitting techniques [42] and Rosenbrock schemes [23, Chapter IV.5].

Other possible approaches to integrate advection–diffusion problems efficiently include implicit-explicit (IMEX) methods [23, Chapter IV.4] and multirate schemes [9, 34, 35].

In this paper we show that in some cases exponential time integration schemes can serve as an efficient yet simple way to integrate advection–diffusion problems in time on locally refined meshes. Similar to implicit schemes, exponential schemes have attractive stability properties. However, exponential schemes have also excellent accuracy properties and in some cases, especially for linear ODE (ordinary differential equation) systems, are able to produce exact solution to initial-value problem (IVP) being solved. This is the property we exploit in this work. An example of an IVP that can be solved exactly by an exponential solver is

$$y'(t) = -Ay(t) + g, \quad y(0) = v, \quad t \in [0, T], \tag{1}$$

where $v, g \in \mathbb{R}^N$ are given and $A \in \mathbb{R}^{N\times N}$ represents the advection–diffusion operator discretized in space. Exponential solvers involve matrix-vector products with the matrix exponential and related, the so-called φ functions. More specifically, the exact solution $y(t)$ of (1) can be written as

$$y(t) = v + t\varphi(-tA)(g - Av), \quad t \geqslant 0, \tag{2}$$

where a matrix-vector product of the matrix function $\varphi(-tA)$ and the vector $g - Av$ has to be computed. Here the function φ is defined as [21]

$$\varphi(z) = \begin{cases} \dfrac{\mathrm{e}^z - 1}{z}, & \text{for } z \neq 0,\ z \in \mathbb{C}, \\ 1, & \text{for } z = 0. \end{cases}$$

In case $g \equiv 0$ expression (2) reduces to a familiar relation $y(t) = \exp(-tA)v$, $t \geqslant 0$. Note that such an "all-at-once" exact solution of systems (1) is also possible if g is a given vector function $g(t)$ of time t, see [1]. Furthermore, to solve general nonlinear IVPs within this approach, across-time iterations of the waveform relaxation type can successfully be employed [25]. Then, at each waveform relaxation iteration, a linear IVP of the type (1) is solved by an exponential scheme. Such an approach is attractive because no time stepping is involved and solution can often be obtained for the whole time interval $t \in [0, T]$.

Exponential time integration is a rapidly developing research area [21]. Two classes of exponential schemes can be distinguished: (1) time-stepping schemes having a certain accuracy order where the matrix exponential or related matrix functions are evaluated *within* each time step and (2) schemes which, formally

speaking, deliver an exact solution to the IVP, where the actions of the matrix exponential or related matrix functions are employed *across* a certain time interval. The former class includes extrapolated second order exponential Euler method (EE2, discussed and tested below), exponential Rosenbrock schemes [22], and predictor-corrector exponential schemes [30]. Examples of the latter class are the scheme (2) applied to IVP (1), exponential waveform relaxation schemes [5] where solution to (3) is sought by replacing A with a preconditioner $M \approx A$ and iterating, and the exponential block Krylov (EBK) method discussed and tested below. Note that exponential schemes of the both classes can handle nonlinear problems, in particular, compressible [28, 29] and incompressible [26] Navier-Stokes equations.

In this work we present comparison results for three different exponential solvers. All these methods are based on the Krylov subspace techniques discussed in [1, 6, 36]. Krylov subspace methods have been successfully used for evaluating matrix exponential and related matrix functions since the eighties, see in chronological order [11, 12, 20, 24, 32, 33, 39]. An attractive property of the Krylov subspace methods, which distinguishes them from the other methods used for large matrix function evaluations $f(A)v$, is their adaptivity with respect to the spectral properties of A and the particular vector v, see [40]. To work efficiently, Krylov subspace methods often need a restarting [14, 18], a mechanism allowing to keep Krylov subspace dimension restricted while preserving convergence of the unrestarted method.

The structure of this paper is as follows. In Sect. 2 the problem and methods used for its solution are presented. Section 3 is devoted to numerical experiments, here the methods are compared and comparison results are discussed. Finally, some conclusions are drawn in Sect. 4.

2 Problem Formulation and Methods

In this paper we assume that a linear PDE of the advection–diffusion type is solved by the method of lines and, after a suitable space discretization, the following IVP has to be solved

$$y'(t) = -Ay(t) + g(t), \quad y(0) = v, \quad t \in [0, T]. \tag{3}$$

Here $A \in \mathbb{R}^{N\times N}$ represents the discretized advection–diffusion operator and the given vector function $g(t)$ accounts for time dependent sources or boundary conditions.

2.1 Exponential Time Integrators

Perhaps the simplest exponential integrator is exponential Euler method which, applied to problem (3), reads

$$y_{n+1} = y_n + \Delta t \varphi(-\Delta t A)(g_n - A y_n), \tag{4}$$

where Δt is the time step size, y_n is numerical solution at time $t = \Delta t n$ and $g_n = g(\Delta t n)$. The method is inspired by relation (2) and for constant source term g is exact. It is not difficult to check that it is first order accurate.

Using extrapolation [3, 44], i.e., by combining solutions obtained with different time steps, higher order methods can be obtained. Globally extrapolated second order exponential Euler method (EE2) is considered in [1],

(1) carry out $T/\Delta t$ steps of (4) with Δt, set result to $y_{\Delta t}(T)$,

(2) carry out $2T/\Delta t$ steps of (4) with $\Delta t/2$, set result to $y_{\Delta t/2}(T)$, (5)

(3) obtain solution by extrapolation: $y_{\mathrm{EE2}}(T) := 2y_{\Delta t/2}(T) - y_{\Delta t}(T)$,

where its combination with EXPOKIT [36], used to evaluate the φ matrix vector products, is argued to be a competitive integrator. The `phiv` function of EXPOKIT is able to efficiently compute actions of the φ matrix functions by a restarted Arnoldi process, where the restarting is done by time stepping based on an error estimation. In experiments presented below we show that EE2/EXPOKIT can be significantly improved by introducing residual-based error control [4, 8, 13] and replacing the restarting procedure by the residual-time (RT) restarting presented in [6, 7].

EE2 is an exponential integrator which evaluates matrix functions *within* a time stepping procedure: at each time step φ is computed by a Krylov subspace method. It is often more efficient [2, 5], if possible, to organize work in such a way that numerical linear algebra work for matrix function evaluations is done *across* time stepping, for a certain time interval. For instance, as shown in [1], if for a certain time range $g(t)$ allows an approximation

$$g(t) \approx U p(t), \quad U \in \mathbb{R}^{N\times m}, \quad p\colon \mathbb{R} \to \mathbb{R}^m, \quad m \ll N, \tag{6}$$

then (3) can be solved for this time range by a single projection on a block Krylov subspace. The matrix U and function p in (6) can be easily constructed by truncated singular value decomposition (SVD) of the vectors $g(t_i)$, at a small additional cost [1]. This procedure simultaneously provides an error estimation in (6), so that a proper value for m can be chosen.

To be more specific, consider problem (1), where for simplicity and without loss of generality assume $v = 0$. A usual Krylov subspace solution of (1) constructs a matrix $V_k \in \mathbb{R}^{N\times k}$, whose orthonormal columns span the Krylov subspace [40]

$$\text{span}(g, Ag, \ldots, A^{k-1}g),$$

and, searching for an approximate solution $y_k(t) = V_k u(t) \approx y(t)$, reduces (1) to its Galerkin projection

$$V_k^T V_k u'(t) = -V_k^T A V_k u(t) + V_k^T g \quad \Leftrightarrow \quad u'(t) = -H_k u(t) + \beta e_1, \tag{7}$$

where $H_k = V_k^T A V_k$, $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^k$ is the first canonical basis vector and $\beta = \|g\|$. We have $V_k^T g = V_k^T V_k(\beta e_1) = \beta e_1$ because, by construction, the first column of V_k is $g/\|g\|$. The small projected IVP (7) can be solved by relation (2), evaluating $\varphi(-tH_k)$ with well developed matrix function techniques for small matrices (see, e.g., [19]).

Now consider, still assuming $v = 0$, problem (3) where $g(t)$ allows (6). Projecting (3) on a block Krylov subspace [40]

$$\text{span}(U, AU, \ldots, A^{k-1}U) = \text{colspan} V_k, \quad V_k \in \mathbb{R}^{N\times km},$$

we can reduce (3) to its projected form

$$u'(t) = -H_k u(t) + E_1 p(t), \tag{8}$$

where we now have $H_k \in \mathbb{R}^{km\times km}$ and $E_1 \in \mathbb{R}^{km\times m}$ is a matrix whose columns are the first m columns of the $km \times km$ identity matrix. These observations lead to the exponential block Krylov (EBK) method described in [1].

The EBK solver exploits a stopping criterion and restarting which are based on the exponential residual concept [4, 8, 13]. In particular, EBK iterations stop as soon as for the computed approximate solution $y_k(t)$ holds

$$\|r_k(t)\| \leqslant \texttt{tol}, \quad r_k(t) \equiv -Ay_k(t) + g(t) - y_k'(t), \quad t \in [0, T].$$

The EBK method can be summarized as follows (see [1] for details):

(1) form the approximation (6),

(2) carry out $k = 1, \ldots, k_{\max}$ block Krylov steps, obtaining V_k, H_k, (9)

(3) solve projected IVP (8), compute solution $y_k(T) = V_k u(T)$.

Comparing the EE2 and EBK schemes, we note that an attractive feature of EE2 is its relative simplicity. On the other hand, a strong point for EBK is its potential efficiency, as a single block Krylov subspace has to be constructed for the whole time interval $t \in [0, T]$.

2.2 ROS2 Method: Beyond Splitting

Rosenbrock schemes [23, Chapter IV.5] have been a popular alternative to splitting methods, as they allow to reduce splitting errors and avoid other negative effects related to splitting, such as order reduction. Let $f(t, y) = -Ay(t) + g(t)$ be the ODE right hand side in (3). The two-stage Rosenbrock method ROS2 reads

$$\begin{aligned} y_{n+1} &= y_n + \frac{3}{2}\Delta t k_1 + \frac{1}{2}\Delta t k_2, \\ (I - \gamma \Delta t \widehat{A}) k_1 &= f(t_n, y_n), \\ (I - \gamma \Delta t \widehat{A}) k_2 &= f(t_{n+1}, y_n + \Delta t k_1) - 2k_1. \end{aligned} \tag{10}$$

The method is second order consistent for any $\widehat{A} \in \mathbb{R}^{N\times N}$ and, to have good stability properties, one usually takes $\widehat{A} \approx A$. Typically, $\widehat{A}$ corresponds to the terms in A which have to be integrated implicitly in time. For instance in [41], for advection-diffusion-reaction problems, $\widehat{A}$ is taken such that

$$I - \gamma \Delta t \widehat{A} = (I - \gamma \Delta t A_{\text{diff}})(I - \gamma \Delta t A_{\text{react}}),$$

where A_{diff} contains diffusion terms and A_{react} is the reaction Jacobian. In this work we take $\widehat{A}$ to be either A or the diffusion part of A. Following suggestion in [23, Chapter IV.5, Remark 5.2] we set $\gamma = 1$.

3 Numerical Experiments

Numerical experiments described here are carried in Matlab on a Linux PC with 6 Intel Core i5-8400 2.80 GHz CPUs with 16 GB RAM.

3.1 Test 1: Time Dependent Source and Boundary Conditions

In this test we solve (3) where A is a finite-element discretization of the two-dimensional advection–diffusion operator:

$$L[u] = -\nu \nabla^2 u + \boldsymbol{v} \cdot \nabla u, \qquad u = u(x, y), \quad (x, y) \in [-1, 1] \times [-1, 1], \tag{11}$$

where ν is the viscosity parameter and the velocity field is $\boldsymbol{v} = [v_1(x, y), v_2(x, y)]$,

$$v_1(x, y) = y(1 - x^2), \qquad v_2(x, y) = x(y^2 - 1).$$

For this test, the function $g(t)$ in (3) takes the form

$$g(t) \equiv y'_{\rm ex}(t) + Ay_{\rm ex}(t),$$

where $y_{\rm ex}(t)$ is exact solution function chosen as

$$\begin{aligned} y_{\rm ex}(t) &= \alpha(t)\big(A^{-1}g_{\rm bc} + T\varphi(-TA)g_{\rm peak}\big), \\ \alpha(t) &= 1 - {\rm e}^{-t/300} + {\rm e}^{-t/100}. \end{aligned} \tag{12}$$

Here $g_{\rm bc} \in \mathbb{R}^N$ is a vector containing Dirichlet boundary values prescribed below and the vector $g_{\rm peak} \in \mathbb{R}^N$ consists of the values of function ${\rm e}^{-10x^2-50y^2}$ on the mesh. The boundary conditions imposing by $g_{\rm bc}$ are

$$u(-1, y) = u(1, y) = u(x, -1) = 5, \qquad u(x, 1) = 5 + 5{\rm e}^{-50x^2}.$$

Note that $A^{-1}g_{\rm bc}$ is the steady state solution of (3) for $g(t) \equiv g_{\rm bc}$ and $T\varphi(-TA)g_{\rm peak}$ is the solution of (3) with $g(t) \equiv g_{\rm peak}$ at time $t = T$. The final time is $T = 1000$ in this test. In Fig. 1 the exact solution $y_{\rm ex}(T)$ is plotted.

In this test the IFISS finite element discretization [16, 37] by bilinear quadrilateral (Q_1) finite elements with the streamline upwind Petrov–Galerkin (SUPG) stabilization is employed. We set viscosity to $\nu = 1/6400$ and use nonuniform Cartesian stretched 256×256 and 512×512 grids with default refinement parameters, which get finer near the domain boundaries, see Table 1.

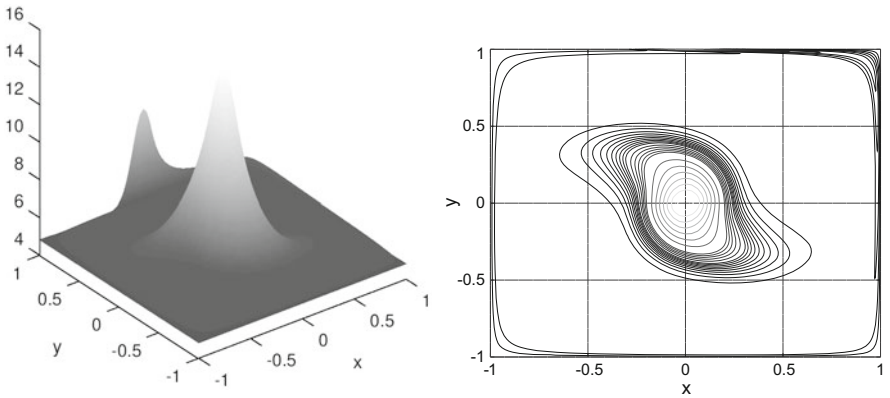


Fig. 1 Solution function (12) on the mesh 256×256 at final time $T = 1000$ as surface (left) and contour (right) plots

Table 1 Parameters of the IFISS stretched meshes

Mesh	$\min h_x = \min h_y$	$\max h_x = \max h_y$	Ratio $\max h_{x,y} / \min h_{x,y}$	max.elem. Peclet for $\nu = 1/6400$
256×256	5.9804×10^{-4}	0.0312	52.17	1.9989×10^2
512×512	2.0102×10^{-4}	0.0176	87.5535	1.1248×10^2

When constructing an advection-diffusion matrix, the IFISS package provides the value the maximum finite element grid Peclet number, evaluated per element as

$$\frac{1}{2\nu} \min \left\{ \frac{h_x}{\cos\alpha}, \frac{h_y}{\sin\alpha} \right\} \|\boldsymbol{v}\|_2, \quad \alpha = \arctan \frac{v_2}{v_1},$$

where $h_{x,y}$ and $v_{1,2}$ are respectively the element sizes and the velocity components. The maximum element Peclet numbers reported for these meshes are given in Table 1. Due to the SUPG stabilization, the resulting matrices for both meshes are weakly nonsymmetric: the ratio $\|A - A^T\|_1 / \|A + A^T\|_1$ amounts approximately to 0.022 (mesh 256×256) and 0.012 (mesh 512×512).

In addition to the requested accuracy tolerance, two input parameters have to provided to EBK: the number of the truncated SVD terms m and the number of time snapshots n_s to construct approximation (6). From the problem description, we see that $y_{\text{ex}}(t)$ is a linear combination of two linearly independent vectors for any t. Hence, $g(t)$ is a linear combination of no more than four vectors and we should take $m \leqslant 4$. The actual situation is displayed by the singular values available from the thin SVD of the time samples: the largest truncated singular value σ_{m+1} is an upper bound for the truncation error $\|g(t) - Up(t)\|_2$, see, e.g., [17]. In this case it turns out that taking $m = 2$ is sufficient. A proper snapshot number n_s can be estimated from given $\alpha(t)$ or by checking, for constructed U and $p(t)$, the actual error $\|g(t) - Up(t)\|$ a posteriori, see (Table 2). Based on this, we set $n_s = 120$ in all EBK runs in this test. This selection procedure for n_s is computationally very cheap and can be done once, before all the test runs.

As the problem is two-dimensional, linear systems with A can be solved efficiently by sparse direct methods. Therefore to solve the linear systems in ROS2, we use the Matlab standard sparse LU factorization (provided by UMFPACK) computing it once and using at each time step.

Table 2 Error of approximation (6), 256×256 mesh. The EBK errors are obtained for $\texttt{tol} = 10^{-6}$

n_s	$\max_{s\in[0,T]} \|g(s) - Up(s)\|$	$\dfrac{\int_0^T \|g(s) - Up(s)\| \, \mathrm{d}s}{\int_0^T \|g(s)\| \, \mathrm{d}s}$	EBK error (13)
30	2.37×10^{-3}	1.82×10^{-5}	2.24×10^{-5}
60	1.27×10^{-4}	9.83×10^{-7}	1.23×10^{-6}
120	7.40×10^{-6}	5.73×10^{-8}	7.90×10^{-8}

The error reported below for the test runs is measured as

$$\text{error} = \frac{\|y(T) - y_{\text{ex}}(T)\|_2}{\|y_{\text{ex}}(T)\|_2}. \tag{13}$$

The results of the test runs are presented in Tables 3 and 4. As we see, EBK turns out to be more efficient than the other solvers. Within the EE2 integrator, the change of the Krylov subspace solver from EXPOKIT's `phiv` to the RT-restarted algorithm leads to a significant increase in efficiency. Note that this gain is not due to the restarting but due a more reliable residual-based error control. Restarting is usually not done because, due to a sufficiently small Δt, just a couple Krylov steps are carried out in EE2 each time step. In both EE2/EXPOKIT and EE2/RT we should be careful with setting a proper tolerance value, which is used at each time step for stopping the Krylov subspace method evaluating the φ matrix function. Taking a large tolerance value may lead to an accuracy loss. For increasingly small tolerance values the same accuracy will be observed (as it is determined by the time step size) at a higher cost: more matrix-vector multiplications per time step will be needed for the φ matrix function evaluations.

From Tables 3 and 4 we also see that the ROS2 solver becomes less efficient than EE2/RT on the finer mesh as the costs for solving linear systems become more pronounced.

Table 3 Test 1. Results for the 256×256 mesh

Method	CPU time, s	Fevals[a]	l.s.s.[b]	Error
EBK, $\texttt{tol} = 10^{-4}$	0.36	20	—	8.01×10^{-8}
EBK, $\texttt{tol} = 10^{-6}$	0.40	24	—	7.90×10^{-8}
EE2/RT, $\Delta t = 20$, $\texttt{tol} = 10^{-4}$	1.84	500	—	1.51×10^{-3}
EE2/RT, $\Delta t = 10$, $\texttt{tol} = 10^{-4}$	3.52	900	—	3.79×10^{-4}
EE2/RT, $\Delta t = 5$, $\texttt{tol} = 10^{-4}$	6.93	1800	—	9.50×10^{-5}
EE2/EXPOKIT, $\Delta t = 20$, $\texttt{tol} = 10^{-4}$	17.99	9408	—	1.51×10^{-3}
EE2/EXPOKIT, $\Delta t = 10$, $\texttt{tol} = 10^{-4}$	24.53	12,608	—	3.79×10^{-4}
EE2/EXPOKIT, $\Delta t = 5$, $\texttt{tol} = 10^{-4}$	37.74	19,200	—	9.50×10^{-5}
ROS2, $\widehat{A} = A$, $\Delta t = 20$	1.95	100	100	3.03×10^{-3}
ROS2, $\widehat{A} = A$, $\Delta t = 10$	3.59	200	200	7.60×10^{-4}
ROS2, $\widehat{A} = A$, $\Delta t = 5$	6.85	400	400	1.91×10^{-4}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 2$	19.55	1000	1000	8.49×10^{-4}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 1$	34.72	2000	2000	7.59×10^{-6}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 0.5$	68.33	4000	4000	1.90×10^{-6}

[a] Number of function evaluations or matvec (matrix-vector) products

[b] Number of linear system solutions

Table 4 Test 1. Results for the 512×512 mesh

Method	CPU time, s	Fevals[a]	l.s.s.[b]	Error
EBK, $\texttt{tol} = 10^{-4}$	1.26	4	—	3.08×10^{-8}
EBK, $\texttt{tol} = 10^{-6}$	1.31	8	—	2.33×10^{-8}
EE2/RT, $\Delta t = 20$, $\texttt{tol} = 10^{-4}$	9.10	450	—	8.91×10^{-4}
EE2/RT, $\Delta t = 10$, $\texttt{tol} = 10^{-4}$	17.97	900	—	2.40×10^{-4}
EE2/RT, $\Delta t = 5$, $\texttt{tol} = 10^{-4}$	35.90	1800	—	8.07×10^{-5}
ROS2, $\widehat{A} = A$, $\Delta t = 20$	11.82	100	100	1.90×10^{-3}
ROS2, $\widehat{A} = A$, $\Delta t = 10$	22.68	200	200	5.46×10^{-4}
ROS2, $\widehat{A} = A$, $\Delta t = 5$	36.91	400	400	1.85×10^{-4}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 2$	86.55	1000	1000	5.73×10^{-3}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 1$	167.95	2000	2000	4.44×10^{-6}
ROS2, $\widehat{A} = A_{\text{diff}}$, $\Delta t = 0.5$	331.38	4000	4000	1.11×10^{-6}

[a] Number of function evaluations or matvec (matrix-vector) products

[b] Number of linear system solutions

3.2 Test 2: Time Dependent Boundary Conditions

In the previous test we see that the EBK solver apparently profits from the specific source function, exhibiting a very quick convergence. Although this is not an unusual situation, we now consider another test problem which appears more difficult for EBK. We take the same matrix A as in the first test and the following initial value vector v and source function $g(t)$:

$$g(t) = \alpha(t) g_{\text{bc}}, \qquad v = -T\varphi(-TA) g_{\text{peak}},$$

where $\alpha(t)$ and g_{bc} are the same as in (12). This test problem does not have a known analytical solution and we compute a reference solution $y_{\text{ref}}(t)$ by running EE2/RT with a tiny time step size. The errors of computed numerical solutions $y(t)$ reported below are

$$\text{error} = \frac{\|y(T) - y_{\text{ref}}(T)\|_2}{\|y_{\text{ref}}(T)\|_2}$$

Note that $y_{\text{ref}}(t)$ is influenced by the same space error as $y(t)$, hence, the error shows solely the time error.

From the problem definition we see that the number of SVD terms m can be at most 2. Therefore, in this test EBK is run with the block size $m = 2$ and $n_s = 80$ time snapshots (the value is determined in the same way as in Test 1). For this test we include in comparisons the two solvers which come out as best in the first test, EBK and EE2/RT. The results presented in Table 5 show that EBK does require more steps for this test but is still significantly more efficient than EE2/RT.

Table 5 Test 2. Results for the 256×256 mesh

Method	CPU time, s	Fevals[a]	l.s.s.[b]	Error
EBK, $\texttt{tol} = 10^{-4}$, $n_s = 80$	0.77	36	—	1.83×10^{-5}
EBK, $\texttt{tol} = 10^{-6}$, $n_s = 80$	1.32	50	—	1.91×10^{-7}
EE2/RT, $\Delta t = 10$, $\texttt{tol} = 10^{-6}$	6.53	1306	—	8.91×10^{-5}
EE2/RT, $\Delta t = 5$, $\texttt{tol} = 10^{-6}$	11.54	2406	—	5.58×10^{-5}

[a] Number of function evaluations or matvec (matrix-vector) products

[b] Number of linear system solutions

4 Conclusions

We show that exponential time integrators can be an attractive option for integrating advection–diffusion problems in time, as they possess good accuracy as well as stability properties. In presented tests, they outperform state-of-the-art implicit-explicit ROS2 solvers. Exponential solvers which are able to exploit their matrix function evaluation machinery for a whole time interval (such as EBK in this paper) appear to be preferable to exponential integrators where matrix functions have to be evaluated at each time step.

Acknowledgement This work is supported by the Russian Science Foundation under grant No. 19-11-00338.

References

1. Botchev, M.A.: A block Krylov subspace time-exact solution method for linear ordinary differential equation systems. Numer. Linear Algebra Appl. **20**(4), 557–574 (2013) https://doi.org/10.1002/nla.1865
2. Botchev, M.A.: Krylov subspace exponential time domain solution of Maxwell's equations in photonic crystal modeling. J. Comput. Appl. Math. **293**, 24–30 (2016) https://doi.org/10.1016/j.cam.2015.04.022
3. Botchev, M.A., Verwer, J.G.: Numerical integration of damped Maxwell equations. SIAM J. Sci. Comput. **31**(2), 1322–1346 (2009) https://doi.org/10.1137/08072108X
4. Botchev, M.A., Grimm, V., Hochbruck, M.: Residual, restarting and Richardson iteration for the matrix exponential. SIAM J. Sci. Comput. **35**(3), A1376–A1397 (2013) https://doi.org/10.1137/110820191
5. Botchev, M.A., Oseledets, I.V., Tyrtyshnikov, E.E.: Iterative across-time solution of linear differential equations: Krylov subspace versus waveform relaxation. Comput. Math. Appl. **67**(12), 2088–2098 (2014) https://doi.org/10.1016/j.camwa.2014.03.002
6. Botchev, M.A., Knizhnerman, L.A.: ART: Adaptive residual-time restarting for Krylov subspace matrix exponential evaluations. J. Comput. Appl. Math. **364**, 112311 (2020) https://doi.org/10.1016/j.cam.2019.06.027
7. Botchev, M.A., Knizhnerman, L.A., Tyrtyshnikov, E.E.: A residual concept for Krylov subspace evaluation of the φ matrix function (2020). Preprint arXiv:2010.08494. https://arxiv.org/abs/2010.08494

8. Celledoni, E., Moret, I.: A Krylov projection method for systems of ODEs. Appl. Numer. Math. **24**(2–3), 365–378 (1997) https://doi.org/10.1016/S0168-9274(97)00033-0
9. Constantinescu, E.M., Sandu, A.: Multirate timestepping methods for hyperbolic conservation laws. J. Sci. Comput. **33**(3), 239–278 (2007)
10. Csomós, P., Faragó, I., Havasi, Á.: Weighted sequential splittings and their analysis. Comput. Math. with Appl. **50**(7), 1017–1031 (2005)
11. Druskin, V.L., Knizhnerman, L.A.: Two polynomial methods of calculating functions of symmetric matrices. U.S.S.R. Comput. Maths. Math. Phys. **29**(6), 112–121 (1989)
12. Druskin, V.L., Knizhnerman, L.A.: Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic. Numer. Lin. Alg. Appl. **2**, 205–217 (1995)
13. Druskin, V.L., Greenbaum, A., Knizhnerman, L.A.: Using nonorthogonal Lanczos vectors in the computation of matrix functions. SIAM J. Sci. Comput. **19**(1), 38–54 (1998) https://doi.org/10.1137/S1064827596303661
14. Eiermann, M., Ernst, O.G.: A restarted Krylov subspace method for the evaluation of matrix functions. SIAM J. Numer. Anal. **44**, 2481–2504 (2006)
15. Einkemmer, L., Ostermann, A.: Overcoming order reduction in diffusion-reaction splitting. Part 1: Dirichlet boundary conditions. SIAM J. Sci. Comput. **37**(3), A1577–A1592 (2015) https://doi.org/10.1137/140994204
16. Elman, H.C., Ramage, A., Silvester, D.J.: IFISS: A computational laboratory for investigating incompressible flow problems. SIAM Rev. **56**(2), 261–273 (2014)
17. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press, Baltimore and London (1996)
18. Güttel, S., Frommer, A., Schweitzer., M.: Efficient and stable Arnoldi restarts for matrix functions based on quadrature. SIAM J. Matrix Anal. Appl **35**(2), 661–683 (2014)
19. Higham, N.J.: Functions of Matrices: Theory and Computation. Society for Industrial and Applied Mathematics, Philadelphia (2008)
20. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **34**(5), 1911–1925 (1997)
21. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010) https://doi.org/10.1017/S0962492910000048
22. Hochbruck, M., Ostermann, A., Schweitzer, J.: Exponential Rosenbrock-type methods. SIAM J. Numer. Anal. **47**(1), 786–803 (2008/2009). https://doi.org/10.1137/080717717
23. Hundsdorfer, W., Verwer, J.G.: Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer, Berlin (2003)
24. Knizhnerman, L.A.: Calculation of functions of unsymmetric matrices using Arnoldi's method. U.S.S.R. Comput. Maths. Math. Phys. **31**(1), 1–9 (1991)
25. Kooij, G.L., Botchev, M.A., Geurts, B.J.: A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations. J. Comput. Appl. Math. **316**(Supplement C), 229–246 (2017) https://doi.org/10.1016/j.cam.2016.09.036
26. Kooij, G., Botchev, M.A., Geurts, B.J.: An exponential time integrator for the incompressible Navier–Stokes equation. SIAM J. Sci. Comput. **40**(3), B684–B705 (2018) https://doi.org/10.1137/17M1121950
27. Lanser, D., Verwer, J.G.: Analysis of operator splitting for advection–diffusion–reaction problems from air pollution modelling. J. Comput. Appl. Math. **111**(1–2), 201–216 (1999)
28. Li, S.J.: Efficient p-multigrid method based on an exponential time discretization for compressible steady flows (2018). arXiv preprint 1807.01151. https://arxiv.org/abs/1807.01151
29. Li, S.J.: Time advancement of the Navier-Stokes equations: p-adaptive exponential methods. J. Flow Control Measurement Vis. **8**(2), 63–76 (2020) https://doi.org/10.4236/jfcmv.2020.82004
30. Li, S.J., Luo, L.S., Wang, Z.J., Ju, L.: An exponential time-integrator scheme for steady and unsteady inviscid flows. J. Comput. Phys. **365**, 206–225 (2018) https://doi.org/10.1016/j.jcp.2018.03.020

31. Lie, K.A., Mykkeltvedt, T.S., Møyner, O.: A fully implicit WENO scheme on stratigraphic and unstructured polyhedral grids. Comput. Geosci. **24**(2), 405–423 (2020)
32. Park, T.J., Light, J.C.: Unitary quantum time evolution by iterative Lanczos reduction. J. Chem. Phys. **85**, 5870–5876 (1986)
33. Saad, Y.: Analysis of some Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **29**(1), 209–228 (1992)
34. Savcenco, V., Hundsdorfer, W., Verwer, J.G.: A multirate time stepping strategy for stiff ordinary differential equations. BIT Numer. Math. **47**(1), 137–155 (2007)
35. Schlegel, M., Knoth, O., Arnold, M., Wolke, R.: Multirate Runge–Kutta schemes for advection equations. J. Comput. Appl. Math. **226**(2), 345–357 (2009). https://doi.org/10.1016/j.cam.2008.08.009
36. Sidje, R.B.: EXPOKIT. A software package for computing matrix exponentials. ACM Trans. Math. Softw. **24**(1), 130–156 (1998) www.maths.uq.edu.au/expokit/
37. Silvester, D.J., Elman, H.C., Ramage, A.: Incompressible flow & iterative solver software (2019). http://www.manchester.ac.uk/ifiss/
38. Sommeijer, B.P., van der Houwen, P.J., Verwer, J.G.: On the treatment of time-dependent boundary conditions in splitting methods for parabolic differential equations. J. Numer. Methods Engrg. **17**(3), 335–346 (1981). https://doi.org/10.1002/nme.1620170304
39. van der Vorst, H.A.: An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A. J. Comput. Appl. Math. **18**, 249–263 (1987)
40. van der Vorst, H.A.: Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, Cambridge (2003)
41. Verwer, J.G., Spee, E.J., Blom, J.G., Hundsdorfer, W.: A second order Rosenbrock method applied to photochemical dispersion problems. SIAM J. Sci. Comput. **20**, 456–480 (1999)
42. Verwer, J.G., Hundsdorfer, W., Blom, J.G.: Numerical time integration for air pollution models. Surv. Math. Ind. **10**, 107–174 (2002) https://ir.cwi.nl/pub/4620
43. Zlatev, Z.: Computer Treatment of Large Air Pollution Models. Kluwer Academic Publishers, New York (1995)
44. Zlatev, Z., Dimov, I., Faragó, I., Havasi, Á.: Richardson Extrapolation: Practical Aspects and Applications. De Gruyter, Berlin (2018)

Author Index

A
Alauzet, F., 127, 141
Aman, A., 113
Anosova, O., 37

B
Barrera, P., 263
Belyaev, A.G., 341
Blanchi, V., 251
Botchev, M.A., 391
Bright, M., 37

C
Cao, J., 199, 307
Chang, J., 199
Chen, W., 61
Clerici, F., 141
Corman, É., 251

D
Dolbilin, N., 3

E
Erkoç, Z., 113

F
Fayolle, P.-A., 341
Fedoseev, D.A., 13

G
Güdükbay, U., 113
Gao, Z., 307
Garanzha, V., 81, 157
Grosso, R., 281
Gu, X., 61
Guan, Z., 199, 307
Guo, Y., 61

K
Karasuljic, S., 227
Karavaev, A.S., 295
Kopysov, S.P., 295
Korotov, S., 241
Kudryavtseva, L., 81, 157
Kurlin, V., 37

L
Lei, N., 61
Li, S.-J., 375
Liseikin, V.D., 227
Lo, S.H., 307
Luo, Z., 61

M
Méndez, I., 263
Manturov, V.O., 13
Mukhortov, A.V., 227

N
Nhan, T.A., 213
Nikonov, I.M., 13

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3

P
Paasonen, V.I., 227

R
Ray, N., 251

S
Si, H., 95
Sokolov, D., 251

T
Tenkes, L.-M., 127
Terekhov, K.M., 361

V
Vatne, J.E., 241
Vlachkova, K., 327

Y
Yu, F., 199, 307

Z
Zegeling, P.A., 179
Zhao, T., 61
Zint, D., 281

Editorial Policy

1. Volumes in the following three categories will be published in LNCSE:

i) Research monographs
ii) Tutorials
iii) Conference proceedings

Those considering a book which might be suitable for the series are strongly advised to contact the publisher or the series editors at an early stage.

2. Categories i) and ii). Tutorials are lecture notes typically arising via summer schools or similar events, which are used to teach graduate students. These categories will be emphasized by Lecture Notes in Computational Science and Engineering. **Submissions by interdisciplinary teams of authors are encouraged.** The goal is to report new developments – quickly, informally, and in a way that will make them accessible to non-specialists. In the evaluation of submissions timeliness of the work is an important criterion. Texts should be well-rounded, well-written and reasonably self-contained. In most cases the work will contain results of others as well as those of the author(s). In each case the author(s) should provide sufficient motivation, examples, and applications. In this respect, Ph.D. theses will usually be deemed unsuitable for the Lecture Notes series. Proposals for volumes in these categories should be submitted either to one of the series editors or to Springer-Verlag, Heidelberg, and will be refereed. A provisional judgement on the acceptability of a project can be based on partial information about the work: a detailed outline describing the contents of each chapter, the estimated length, a bibliography, and one or two sample chapters – or a first draft. A final decision whether to accept will rest on an evaluation of the completed work which should include

– at least 100 pages of text;
– a table of contents;
– an informative introduction perhaps with some historical remarks which should be accessible to readers unfamiliar with the topic treated;
– a subject index.

3. Category iii). Conference proceedings will be considered for publication provided that they are both of exceptional interest and devoted to a single topic. One (or more) expert participants will act as the scientific editor(s) of the volume. They select the papers which are suitable for inclusion and have them individually refereed as for a journal. Papers not closely related to the central topic are to be excluded. Organizers should contact the Editor for CSE at Springer at the planning stage, see *Addresses* below.

In exceptional cases some other multi-author-volumes may be considered in this category.

4. Only works in English will be considered. For evaluation purposes, manuscripts may be submitted in print or electronic form, in the latter case, preferably as pdf- or zipped ps-files. Authors are requested to use the LaTeX style files available from Springer at http://www.springer.com/gp/authors-editors/book-authors-editors/manuscript-preparation/5636 (Click on LaTeX Template → monographs or contributed books).

For categories ii) and iii) we strongly recommend that all contributions in a volume be written in the same LaTeX version, preferably LaTeX2e. Electronic material can be included if appropriate. Please contact the publisher.

Careful preparation of the manuscripts will help keep production time short besides ensuring satisfactory appearance of the finished book in print and online.

5. The following terms and conditions hold. Categories i), ii) and iii):

Authors receive 50 free copies of their book. No royalty is paid.
Volume editors receive a total of 50 free copies of their volume to be shared with authors, but no royalties.

Authors and volume editors are entitled to a discount of 40 % on the price of Springer books purchased for their personal use, if ordering directly from Springer.

6. Springer secures the copyright for each volume.

Addresses:

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
barth@nas.nasa.gov

Michael Griebel
Institut für Numerische Simulation
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
griebel@ins.uni-bonn.de

David E. Keyes
Mathematical and Computer Sciences
and Engineering
King Abdullah University of Science
and Technology
P.O. Box 55455
Jeddah 21534, Saudi Arabia
david.keyes@kaust.edu.sa

and

Department of Applied Physics
and Applied Mathematics
Columbia University
500 W. 120 th Street
New York, NY 10027, USA
kd2112@columbia.edu

Risto M. Nieminen
Department of Applied Physics
Aalto University School of Science
and Technology
00076 Aalto, Finland
risto.nieminen@aalto.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
dirk.roose@cs.kuleuven.be

Tamar Schlick
Department of Chemistry
and Courant Institute
of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012, USA
schlick@nyu.edu

Editor for Computational Science
and Engineering at Springer:

Martin Peters
Springer-Verlag
Mathematics Editorial IV
Tiergartenstrasse 17
69121 Heidelberg, Germany
martin.peters@springer.com

Lecture Notes in Computational Science and Engineering

1. D. Funaro, *Spectral Elements for Transport-Dominated Equations.*
2. H.P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.
3. W. Hackbusch, G. Wittum (eds.), *Multigrid Methods V.*
4. P. Deuflhard, J. Hermans, B. Leimkuhler, A.E. Mark, S. Reich, R.D. Skeel (eds.), *Computational Molecular Dynamics: Challenges, Methods, Ideas.*
5. D. Kröner, M. Ohlberger, C. Rohde (eds.), *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws.*
6. S. Turek, *Efficient Solvers for Incompressible Flow Problems.* An Algorithmic and Computational Approach.
7. R. von Schwerin, ***M**ulti **B**ody **S**ystem **SIM**ulation.* Numerical Methods, Algorithms, and Software.
8. H.-J. Bungartz, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
9. T.J. Barth, H. Deconinck (eds.), *High-Order Methods for Computational Physics.*
10. H.P. Langtangen, A.M. Bruaset, E. Quak (eds.), *Advances in Software Tools for Scientific Computing*.
11. B. Cockburn, G.E. Karniadakis, C.-W. Shu (eds.), *Discontinuous Galerkin Methods.* Theory, Computation and Applications.
12. U. van Rienen, *Numerical Methods in Computational Electrodynamics.* Linear Systems in Practical Applications.
13. B. Engquist, L. Johnsson, M. Hammill, F. Short (eds.), *Simulation and Visualization on the Grid.*
14. E. Dick, K. Riemslagh, J. Vierendeels (eds.), *Multigrid Methods VI.*
15. A. Frommer, T. Lippert, B. Medeke, K. Schilling (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics.*
16. J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems.* Theory, Algorithm, and Applications.
17. B.I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition.*
18. U. van Rienen, M. Günther, D. Hecht (eds.), *Scientific Computing in Electrical Engineering.*
19. I. Babuška, P.G. Ciarlet, T. Miyoshi (eds.), *Mathematical Modeling and Numerical Simulation in Continuum Mechanics.*
20. T.J. Barth, T. Chan, R. Haimes (eds.), *Multiscale and Multiresolution Methods.* Theory and Applications.
21. M. Breuer, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
22. K. Urban, *Wavelets in Numerical Simulation.* Problem Adapted Construction and Applications.
23. L.F. Pavarino, A. Toselli (eds.), *Recent Developments in Domain Decomposition Methods.*

24. T. Schlick, H.H. Gan (eds.), *Computational Methods for Macromolecules: Challenges and Applications.*

25. T.J. Barth, H. Deconinck (eds.), *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*.

26. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations*.

27. S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*.

28. C. Carstensen, S. Funken, W. Hackbusch, R.H.W. Hoppe, P. Monk (eds.), *Computational Electromagnetics*.

29. M.A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*.

30. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders (eds.), *Large-Scale PDE-Constrained Optimization*.

31. M. Ainsworth, P. Davies, D. Duncan, P. Martin, B. Rynne (eds.), *Topics in Computational Wave Propagation*. Direct and Inverse Problems.

32. H. Emmerich, B. Nestler, M. Schreckenberg (eds.), *Interface and Transport Dynamics.* Computational Modelling.

33. H.P. Langtangen, A. Tveito (eds.), *Advanced Topics in Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.

34. V. John, *Large Eddy Simulation of Turbulent Incompressible Flows.* Analytical and Numerical Results for a Class of LES Models.

35. E. Bänsch (ed.), *Challenges in Scientific Computing - CISC 2002.*

36. B.N. Khoromskij, G. Wittum, *Numerical Solution of Elliptic Differential Equations by Reduction to the Interface.*

37. A. Iske, *Multiresolution Methods in Scattered Data Modelling.*

38. S.-I. Niculescu, K. Gu (eds.), *Advances in Time-Delay Systems.*

39. S. Attinger, P. Koumoutsakos (eds.), *Multiscale Modelling and Simulation.*

40. R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Wildlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering.*

41. T. Plewa, T. Linde, V.G. Weirs (eds.), *Adaptive Mesh Refinement – Theory and Applications.*

42. A. Schmidt, K.G. Siebert, *Design of Adaptive Finite Element Software.* The Finite Element Toolbox ALBERTA.

43. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations II.*

44. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Methods in Science and Engineering.*

45. P. Benner, V. Mehrmann, D.C. Sorensen (eds.), *Dimension Reduction of Large-Scale Systems.*

46. D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems.*

47. A. Boriçi, A. Frommer, B. Joó, A. Kennedy, B. Pendleton (eds.), *QCD and Numerical Analysis III*.

48. F. Graziani (ed.), *Computational Methods in Transport.*

49. B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, A. Mark, T. Schlick, C. Schütte, R. Skeel (eds.), *New Algorithms for Macromolecular Simulation.*

50. M. Bücker, G. Corliss, P. Hovland, U. Naumann, B. Norris (eds.), *Automatic Differentiation: Applications, Theory, and Implementations.*

51. A.M. Bruaset, A. Tveito (eds.), *Numerical Solution of Partial Differential Equations on Parallel Computers.*

52. K.H. Hoffmann, A. Meyer (eds.), *Parallel Algorithms and Cluster Computing.*

53. H.-J. Bungartz, M. Schäfer (eds.), *Fluid-Structure Interaction.*

54. J. Behrens, *Adaptive Atmospheric Modeling.*

55. O. Widlund, D. Keyes (eds.), *Domain Decomposition Methods in Science and Engineering XVI.*

56. S. Kassinos, C. Langer, G. Iaccarino, P. Moin (eds.), *Complex Effects in Large Eddy Simulations.*

57. M. Griebel, M.A Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations III.*

58. A.N. Gorban, B. Kégl, D.C. Wunsch, A. Zinovyev (eds.), *Principal Manifolds for Data Visualization and Dimension Reduction.*

59. H. Ammari (ed.), *Modeling and Computations in Electromagnetics: A Volume Dedicated to Jean-Claude Nédélec.*

60. U. Langer, M. Discacciati, D. Keyes, O. Widlund, W. Zulehner (eds.), *Domain Decomposition Methods in Science and Engineering XVII.*

61. T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations.*

62. F. Graziani (ed.), *Computational Methods in Transport: Verification and Validation.*

63. M. Bebendorf, *Hierarchical Matrices.* A Means to Efficiently Solve Elliptic Boundary Value Problems.

64. C.H. Bischof, H.M. Bücker, P. Hovland, U. Naumann, J. Utke (eds.), *Advances in Automatic Differentiation.*

65. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations IV*.

66. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Modeling and Simulation in Science.*

67. I.H. Tuncer, Ü. Gülcat, D.R. Emerson, K. Matsuno (eds.), *Parallel Computational Fluid Dynamics 2007.*

68. S. Yip, T. Diaz de la Rubia (eds.), *Scientific Modeling and Simulations*.

69. A. Hegarty, N. Kopteva, E. O'Riordan, M. Stynes (eds.), *BAIL* 2008 – *Boundary and Interior Layers.*

70. M. Bercovier, M.J. Gander, R. Kornhuber, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XVIII.*

71. B. Koren, C. Vuik (eds.), *Advanced Computational Methods in Science and Engineering.*

72. M. Peters (ed.), *Computational Fluid Dynamics for Sport Simulation.*

73. H.-J. Bungartz, M. Mehl, M. Schäfer (eds.), *Fluid Structure Interaction II - Modelling, Simulation, Optimization.*

74. D. Tromeur-Dervout, G. Brenner, D.R. Emerson, J. Erhel (eds.), *Parallel Computational Fluid Dynamics 2008.*

75. A.N. Gorban, D. Roose (eds.), *Coping with Complexity: Model Reduction and Data Analysis.*

76. J.S. Hesthaven, E.M. Rønquist (eds.), *Spectral and High Order Methods for Partial Differential Equations.*

77. M. Holtz, *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance*.

78. Y. Huang, R. Kornhuber, O.Widlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering XIX*.

79. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations V*.

80. P.H. Lauritzen, C. Jablonowski, M.A. Taylor, R.D. Nair (eds.), *Numerical Techniques for Global Atmospheric Models*.

81. C. Clavero, J.L. Gracia, F.J. Lisbona (eds.), *BAIL 2010 – Boundary and Interior Layers, Computational and Asymptotic Methods*.

82. B. Engquist, O. Runborg, Y.R. Tsai (eds.), *Numerical Analysis and Multiscale Computations*.

83. I.G. Graham, T.Y. Hou, O. Lakkis, R. Scheichl (eds.), *Numerical Analysis of Multiscale Problems*.

84. A. Logg, K.-A. Mardal, G. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method*.

85. J. Blowey, M. Jensen (eds.), *Frontiers in Numerical Analysis - Durham 2010.*

86. O. Kolditz, U.-J. Gorke, H. Shao, W. Wang (eds.), *Thermo-Hydro-Mechanical-Chemical Processes in Fractured Porous Media - Benchmarks and Examples*.

87. S. Forth, P. Hovland, E. Phipps, J. Utke, A. Walther (eds.), *Recent Advances in Algorithmic Differentiation*.

88. J. Garcke, M. Griebel (eds.), *Sparse Grids and Applications.*

89. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations VI.*

90. C. Pechstein, *Finite and Boundary Element Tearing and Interconnecting Solvers for Multiscale Problems*.

91. R. Bank, M. Holst, O. Widlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering XX*.

92. H. Bijl, D. Lucor, S. Mishra, C. Schwab (eds.), *Uncertainty Quantification in Computational Fluid Dynamics*.

93. M. Bader, H.-J. Bungartz, T. Weinzierl (eds.), *Advanced Computing*.

94. M. Ehrhardt, T. Koprucki (eds.), *Advanced Mathematical Models and Numerical Techniques for Multi-Band Effective Mass Approximations*.

95. M. Azaïez, H. El Fekih, J.S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2012*.

96. F. Graziani, M.P. Desjarlais, R. Redmer, S.B. Trickey (eds.), *Frontiers and Challenges in Warm Dense Matter*.

97. J. Garcke, D. Pflüger (eds.), *Sparse Grids and Applications – Munich 2012*.

98. J. Erhel, M. Gander, L. Halpern, G. Pichot, T. Sassi, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XXI*.

99. R. Abgrall, H. Beaugendre, P.M. Congedo, C. Dobrzynski, V. Perrier, M. Ricchiuto (eds.), *High Order Nonlinear Numerical Methods for Evolutionary PDEs - HONOM 2013*.

100. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations VII*.

101. R. Hoppe (ed.), *Optimization with PDE Constraints - OPTPDE 2014*.

102. S. Dahlke, W. Dahmen, M. Griebel, W. Hackbusch, K. Ritter, R. Schneider, C. Schwab, H. Yserentant (eds.), *Extraction of Quantifiable Information from Complex Systems*.

103. A. Abdulle, S. Deparis, D. Kressner, F. Nobile, M. Picasso (eds.), *Numerical Mathematics and Advanced Applications - ENUMATH 2013*.

104. T. Dickopf, M.J. Gander, L. Halpern, R. Krause, L.F. Pavarino (eds.), *Domain Decomposition Methods in Science and Engineering XXII*.

105. M. Mehl, M. Bischoff, M. Schäfer (eds.), *Recent Trends in Computational Engineering - CE2014*. Optimization, Uncertainty, Parallel Algorithms, Coupled and Complex Problems.

106. R.M. Kirby, M. Berzins, J.S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM'14*.

107. B. Jüttler, B. Simeon (eds.), *Isogeometric Analysis and Applications 2014*.

108. P. Knobloch (ed.), *Boundary and Interior Layers, Computational and Asymptotic Methods – BAIL 2014*.

109. J. Garcke, D. Pflüger (eds.), *Sparse Grids and Applications – Stuttgart 2014*.

110. H. P. Langtangen, *Finite Difference Computing with Exponential Decay Models*.

111. A. Tveito, G.T. Lines, *Computing Characterizations of Drugs for Ion Channels and Receptors Using Markov Models*.

112. B. Karazösen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, Ö. Uğur (eds.), *Numerical Mathematics and Advanced Applications - ENUMATH 2015*.

113. H.-J. Bungartz, P. Neumann, W.E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2013-2015*.

114. G.R. Barrenechea, F. Brezzi, A. Cangiani, E.H. Georgoulis (eds.), *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*.

115. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations VIII*.

116. C.-O. Lee, X.-C. Cai, D.E. Keyes, H.H. Kim, A. Klawonn, E.-J. Park, O.B. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XXIII*.

117. T. Sakurai, S.-L. Zhang, T. Imamura, Y. Yamamoto, Y. Kuramashi, T. Hoshi (eds.), *Eigenvalue Problems: Algorithms, Software and Applications in Petascale Computing*. EPASA 2015, Tsukuba, Japan, September 2015.

118. T. Richter (ed.), *Fluid-structure Interactions*. Models, Analysis and Finite Elements.

119. M.L. Bittencourt, N.A. Dumont, J.S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016.* Selected Papers from the ICOSAHOM Conference, June 27-July 1, 2016, Rio de Janeiro, Brazil.

120. Z. Huang, M. Stynes, Z. Zhang (eds.), *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2016.*

121. S.P.A. Bordas, E.N. Burman, M.G. Larson, M.A. Olshanskii (eds.), *Geometrically Unfitted Finite Element Methods and Applications*. Proceedings of the UCL Workshop 2016.

122. A. Gerisch, R. Penta, J. Lang (eds.), *Multiscale Models in Mechano and Tumor Biology*. Modeling, Homogenization, and Applications.

123. J. Garcke, D. Pflüger, C.G. Webster, G. Zhang (eds.), *Sparse Grids and Applications - Miami 2016*.

124. M. Schäfer, M. Behr, M. Mehl, B. Wohlmuth (eds.), *Recent Advances in Computational Engineering*. Proceedings of the 4th International Conference on Computational Engineering (ICCE 2017) in Darmstadt.

125. P.E. Bjørstad, S.C. Brenner, L. Halpern, R. Kornhuber, H.H. Kim, T. Rahman, O.B. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XXIV*. 24th International Conference on Domain Decomposition Methods, Svalbard, Norway, February 6–10, 2017.

126. F.A. Radu, K. Kumar, I. Berre, J.M. Nordbotten, I.S. Pop (eds.), *Numerical Mathematics and Advanced Applications – ENUMATH 2017*.

127. X. Roca, A. Loseille (eds.), *27th International Meshing Roundtable*.

128. Th. Apel, U. Langer, A. Meyer, O. Steinbach (eds.), *Advanced Finite Element Methods with Applications*. Selected Papers from the 30th Chemnitz Finite Element Symposium 2017.

129. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differencial Equations IX*.

130. S. Weißer, BEM-based Finite Element *Approaches on Polytopal Meshes*.

131. V.A. Garanzha, L. Kamenski, H. Si (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*. Proceedings of the 9th International Conference, NUMGRID2018/Voronoi 150, Celebrating the 150th Anniversary of G. F. Voronoi, Moscow, Russia, December 2018.

132. H. van Brummelen, A. Corsini, S. Perotto, G. Rozza (eds.), *Numerical Methods for Flows*.

133. H. van Brummelen, C. Vuik, M. Möller, C. Verhoosel, B. Simeon, B. Jüttler (eds.), *Isogeometric Analysis and Applications 2018*.

134. S.J. Sherwin, D. Moxey, J. Peiro, P.E. Vincent, C. Schwab (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018*.

135. G.R. Barrenechea, J. Mackenzie (eds.), *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2018*.

136. H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, W.E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016–2019*.

137. M. D'Elia, M. Gunzburger, G. Rozza (eds.), *Quantification of Uncertainty: Improving Efficiency and Technology*.

138. R. Haynes, S. MacLachlan, X.-C. Cai, L. Halpern, H.H. Kim, A. Klawonn, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XXV*.

139. F.J. Vermolen, C. Vuik (eds.), *Numerical Mathematics and Advanced Applications ENUMATH 2019*.

140. O. Sander, *DUNE – The Distributed and Unified Numerics Environment*.

141. I.B. Badriev, V. Banderov, S.A. Lapin (eds.), *Mesh Methods for Boundary-Value Problems and Applications*.

142. ——

143. V.A. Garanzha, L. Kamenski, H. Si (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*. Proceedings of the 10th International Conference, NUMGRID 2020 / Delaunay 130, Celebrating the 130th Anniversary of Boris Delaunay, Moscow, Russia, November 2020

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/3527

Monographs in Computational Science and Engineering

1. J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.-A. Mardal, A. Tveito, *Computing the Electrical Activity in the Heart.*

For further information on this book, please have a look at our mathematics catalogue at the following URL: www.springer.com/series/7417

Texts in Computational Science and Engineering

1. H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming. 2nd Edition
2. A. Quarteroni, F. Saleri, P. Gervasio, *Scientific Computing with MATLAB and Octave.* 4th Edition
3. H. P. Langtangen, *Python Scripting for Computational Science*. 3rd Edition
4. H. Gardner, G. Manduchi, *Design Patterns for e-Science*.
5. M. Griebel, S. Knapek, G. Zumbusch, *Numerical Simulation in Molecular Dynamics*.
6. H. P. Langtangen, *A Primer on Scientific Programming with Python.* 5th Edition
7. A. Tveito, H. P. Langtangen, B. F. Nielsen, X. Cai, *Elements of Scientific Computing.*
8. B. Gustafsson, *Fundamentals of Scientific Computing.*
9. M. Bader, *Space-Filling Curves.*
10. M. Larson, F. Bengzon, *The Finite Element Method: Theory, Implementation and Applications.*
11. W. Gander, M. Gander, F. Kwok, *Scientific Computing: An Introduction using Maple and MATLAB.*
12. P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*.
13. M. H. Holmes, *Introduction to Scientific Computing and Data Analysis*.
14. S. Linge, H. P. Langtangen, *Programming for Computations* - A Gentle Introduction to Numerical Simulations with MATLAB/Octave.
15. S. Linge, H. P. Langtangen, *Programming for Computations* - A Gentle Introduction to Numerical Simulations with Python.
16. H.P. Langtangen, S. Linge, *Finite Difference Computing with PDEs* - A Modern Software Approach.
17. B. Gustafsson, *Scientific Computing from a Historical Perspective*.
18. J. A. Trangenstein, *Scientific Computing*. Volume I - Linear and Nonlinear Equations.

19. J. A. Trangenstein, *Scientific Computing*. Volume II - Eigenvalues and Optimization.

20. J. A. Trangenstein, *Scientific Computing*. Volume III - Approximation and Integration.

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/5151