Journal Subline

LNCS 5300

Transactions on Computational Science III

Marina L. Gavrilova · C.J. Kenneth Tan
Editors-in-Chief

Springer

Lecture Notes in Computer Science 5300

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Marina L. Gavrilova C.J. Kenneth Tan (Eds.)

Transactions on Computational Science III

Springer

Volume Editors

Marina L. Gavrilova
University of Calgary
Department of Computer Science
2500 University Drive N.W.
Calgary, AB, T2N 1N4, Canada
E-mail: marina@cpsc.ucalgary.ca

C.J. Kenneth Tan
OptimaNumerics Ltd.
Cathedral House
23-31 Waring Street
Belfast BT1 2DX, UK
E-mail: cjtan@optimanumerics.com

Library of Congress Control Number: 2009920697

CR Subject Classification (1998): H.5, I.3.7, H.2.8, J.2, K.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743 (Lecture Notes in Computer Science)
ISSN 1866-4733 (Transactions on Computational Science)
ISBN-10 3-642-00211-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00211-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12603824 06/3180 5 4 3 2 1 0

LNCS Transactions on Computational Science

Computational science, an emerging and increasingly vital field, is now widely recognized as an integral part of scientific and technical investigations, affecting researchers and practitioners in areas ranging from aerospace and automotive research to biochemistry, electronics, geosciences, mathematics, and physics. Computer systems research and the exploitation of applied research naturally complement each other. The increased complexity of many challenges in computational science demands the use of supercomputing, parallel processing, sophisticated algorithms, and advanced system software and architecture. It is therefore invaluable to have input by systems research experts in applied computational science research.

Transactions on Computational Science focuses on original high-quality research in the realm of computational science in parallel and distributed environments, also encompassing the underlying theoretical foundations and the applications of large-scale computation. The journal offers practitioners and researchers the opportunity to share computational techniques and solutions in this area, to identify new issues, and to shape future directions for research, and it enables industrial users to apply leading-edge, large-scale, high-performance computational methods.

In addition to addressing various research and application issues, the journal aims to present material that is validated – crucial to the application and advancement of the research conducted in academic and industrial settings. In this spirit, the journal focuses on publications that present results and computational techniques that are verifiable.

Scope

The scope of the journal includes, but is not limited to, the following computational methods and applications:

- Aeronautics and Aerospace
- Astrophysics
- Bioinformatics
- Climate and Weather Modeling
- Communication and Data Networks
- Compilers and Operating Systems
- Computer Graphics
- Computational Biology
- Computational Chemistry
- Computational Finance and Econometrics
- Computational Fluid Dynamics
- Computational Geometry

- Computational Number Theory
- Computational Physics
- Data Storage and Information Retrieval
- Data Mining and Data Warehousing
- Grid Computing
- Hardware/Software Co-design
- High-Energy Physics
- High-Performance Computing
- Numerical and Scientific Computing
- Parallel and Distributed Computing
- Reconfigurable Hardware
- Scientific Visualization
- Supercomputing
- System-on-Chip Design and Engineering

Preface

The Transactions on Computational Science journal is part of the Springer series *Lecture Notes in Computer Science*, and is devoted to the gamut of computational science issues, from theoretical aspects to application-dependent studies and the validation of emerging technologies.

The current issue is devoted to computer systems research and the application of such research, which naturally complement each other. The issue is comprised of Part 1: Computational Visualization and Optimization, and Part 2: Computational Methods for Model Design and Analysis.

Part 1 – Computational Visualization and Optimization – is devoted to state-of-the-art research carried out in this area with the use of novel computational methods. It is comprised of five papers, each addressing a specific computational problem in the areas of shared virtual spaces, dynamic visualization, multimodal user interfaces, computational geometry, and parallel simulation, respectively.

Part 2 – Computational Methods for Model Design and Analysis – continues the topic with an in-depth look at selected computational science research in the areas of data representation and analysis. The four papers comprising this part cover such areas as efficient reversible logic design, missing data analysis, stochastic computation and neural network representation for eccentric sphere models. Each paper describes a detailed experiment or a case study of the methodology presented to amplify the impact of the contribution.

In conclusion, we would like to extend our sincere appreciation to all authors for submitting their papers to this issue, and to all associate editors and referees for their meticulous and valuable reviews. We would also like to express our gratitude to the LNCS editorial staff of Springer, in particular Alfred Hofmann, Ursula Barth and Anna Kramer, who supported us at every stage of the project.

It is our hope that the fine collection of papers presented in this issue will be a valuable resource for Transactions on Computational Science readers and will stimulate further research into the vibrant area of computational science applications.

November 2008

Marina L. Gavrilova
C.J. Kenneth Tan

LNCS Transactions on Computational Science – Editorial Board

Table of Contents

Part 1: Computational Visualization and Optimization

Part 2: Computational Methods for Model Design and Analysis

Visual Immersive Haptic Mathematics in Shared Virtual Spaces

Alexei Sourin, Olga Sourina, Lei Wei, and Paul Gagnon

Nanyang Technological University, Singapore
{assourin,eosourina,wei10004,pgaganon}@ntu.edu.sg

Abstract. When teaching subjects richly infused with mathematics, in particular geometry, topology and shape modeling, there is a frequent problem that the learners are not able to "visualize" the attendant theoretical concepts. It is important, therefore, to constantly illustrate the associated theories with practical visual exercises, which are preferably to be done in collaboration with other learners to allow them to discuss possible approaches to the problem and to consult with the instructor, virtually or face-to-face. We have proposed an approach that would allow for solving mathematical problems while being immersed within shared virtual 3D collaborative environments. Only mathematical formulas are used by the learners for immediate interactive definition of geometry, appearance and physical property of the shapes being created in the virtual environment. We target learners and educators who are studying subjects rich in mathematics and geometry, or teaching them at the secondary or tertiary level or doing research on these topics. The process allows the learners to see and feel the geometric meaning of mathematics, thus making it less abstract and more perceptual and tangible. We also see ways of incorporating beneficial uses of immersive virtual environments into traditional courses at Universities which might benefit from 3D visualization.

Keywords: Web Technologies, Networked Learning, Distance Education, Virtual Campus, Virtual Classroom, Virtual Learning, Web-Based Learning, Shape Modeling, Haptic Collaboration, Shared Virtual Spaces.

1 Introduction

Generally, although the right hemisphere of the human brain is totally responsible for geometric reasoning, in our modern world most people do not use their brain to its full extent as everything around us is often simplified to pictures or symbols of 3D objects displayed on screens, billboards, etc. Our current generation of children, often called "Digital Natives", are very comfortable with 2D text-based SMS messages, emails, web-pages etc., which are an essential part of their everyday life beginning in primary school. As a result, the brain is not being sufficiently trained to deal with problems which require 3D perception. A good example of this is illustrated by this simple test which the readers could do. Look around you and then close one eye and look around again. What is the major difference when you use both eyes and only one eye? Most people will answer that there is not much difference or that with one eye the view range

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 1–19, 2009.

is narrower. Less than 50% (according to out studies and other sources like http://www.vision3d.com) will say that when they see with one eye there is no stereo vision: the phenomenon of 3D depth perception which is characteristic of living creatures having both eyes located side-by-side in the front of their heads rather than on the sides of heads. It was developed as a means of survival for carnivores, as it allowed them to estimate the distance of a leap to the prey—while modern people mostly need skills to read the price of the food on the supermarket price tag. Geometry as one of the oldest sciences with its name originating from the Greek words *geo* (earth) and *metria* (measure) is now largely separated from everyday life. We do not measure earth. When commuting, we do not actually care how 3-dimensional the world around us is and how many hills and valleys our transport is crossing—it rather becomes for us just a means of rapid transportation from one point to another, if not to say a type of teleportation neither controlled nor monitored by our brain. General weakening of geometric mentality, 3D perception and binocular vision, in particular, has been de-facto acknowledged and reflected in education systems of different countries by making curriculum structures more geometry oriented starting from the primary or even pre-school level. For example, in Japan teaching Euclidean geometry is introduced at lower secondary school mathematics level [1]. In the Singapore Mathematics Syllabus Primary [2] and the Secondary Mathematics Syllabuses [3], one of the listed competences which learners can gain from mathematics training is "Spatial visualization". The importance and specifics of visualization in the teaching and learning of mathematics was also discussed as a special topic at the recent 10th International Congress of Mathematical Education [4]. However, due to many reasons, strengthening the spatial visualization is still not emphasized as much as other competences. As a result, students often find it difficult to deal with 3D objects, such as forming different figures of concrete 3D models, identifying the nets of solid figures and engaging in vector analysis. Although there are commercial software tools that can help to visualize 3D objects, they are either expensive or not suitable for school students to use. At the university level, there are many subjects requiring advance geometric reasoning such as calculus, computer graphics, computer animation, geometric modeling, computer-aided design, etc. However, for the same reason as discussed above it is usually a challenge for the students to follow the instructor and visualize how mathematical concepts reflect in 3D geometry and colors.

Electronic education is increasingly being adopted and accepted as an important and vital part of university education. It has become common practice to use e-learning systems, like Blackboard or open-source systems like Moodle and Sakai, to upload and distribute course materials on the web. However, in a world where more and more information is being provided electronically, it often results in disorientation and exhaustion for the students. Current educational practice advocates that student learning should reflect an active learning pedagogy and teaching strategies which support different student learning styles. Cyber-education through immersive shared virtual environments expands these pedagogical frontiers and creates new learning opportunities.

Today, there are many 3D virtual educational spaces created with different communication and shared VR platforms, for example, virtual universities in Active Worlds (http://www.activeworlds.com/edu/index.asp) and Second Life (http://secondlife.com) shared communities. However it remain to be understood as to what constitutes the

perfect teaching/learning environments and tools for conducting education in such virtual spaces—quite often these virtual spaces appear to be merely reflections of the existing 'transmission' or 'expository' paradigms of teaching while cyber-education needs to not just duplicate what is available in real life, but go beyond it. Given our own advance experience of creating and using educational cyber-campuses within the project Virtual Campus of NTU, (http://www.ntu.edu.sg/home/assourin/VirCampus.html), we advocate that the biggest advantage of cyber-learning in shared virtual environments is the ability to create unlimited, enhanced active learning collaboration so that the learners from different physical locations can meet and work on problems in environments specially tailored and best suited for the application problem. The phenomenon of explosively growing 3D virtual communities like *Second Life*, *Entropia Universe*, *There* and others, with their millions of virtual residents, is one cogent example of this fact.

2 Background

2.1 Cyberworlds and Their Educational Uses

Canadian novelist William Gibson coined the term "cyberspace" in his novel "Neuromancer" [5]. We are now using it with reference to events taking place on the Internet as 'happenings' in cyberspace where the participants or the network servers are actually located, rather than in their countries. Since the invention of the Internet, cyberspace has been always used as a space for creating information communities, or cyberworlds capable of supporting 3D visual representation. Concomitant with the first known shared community, *Habitat,* which was created in 1985, this has become a very popular trend, especially over the last few years. A few of the most popular examples listed chronologically are shared virtual communities in *Cybertown* (since 1993, http://www.cybertown.com), *ActiveWorlds* (since 1995, http://www.activeworlds.com), *There* (since 2003, http://www.there.com), *Open Croquet* project (since 2003, http://www.opencroquet.com), *Second Life* (since 2003, http://www.secondlife.com), and two massively multiplayer online role-playing games (MMORPG): *World of Warcraft* (since 2004, http://www.worldofwarcraft.com), and *Entropia Universe* (since 2006, http://www.entropiauniverse.com).

The recent explosive phenomenon and popularity of such places may have many roots. It could be growing computer literacy which allows potential content creators to apply their skills in building 3D cyberworlds. It may be related to limitation of resources in the real world which could motivate one to settle in 3D virtual world with no geographical, climatic and political restrictions. Last but not least, it may simply reflect a desire to explore a new life in a new environment which can be perceived through vision, sound, and communication with other habitants now taking up residence there—perhaps suggesting a new frontier mentality is taking hold. However, the reported millions of registered residents of different cyberworlds are still merely a few per cent of the more than a billion people using the web worldwide.

In the past few years, the usage of 3D cyberworlds for educational purposes has increased significantly. There are now many virtual campuses implemented in ActiveWorlds and Second Life shared environments. The metaphors governing the visual design of such cyberworlds are quite diverse, from replication of real universities,

art museums and scientific labs to non-existing fictitious places [6]. They provide a social arena where students and teachers can meet and thereby overcome the barriers of the physical world not to mention learn "on-the-fly" much like we acquire knowledge in real life.

When making a collaborative virtual environment in cyberspace, 3D web visualization and collaboration can be achieved through several ways. For a strong server and weak client configuration *image and event transmissions* can be used. In this case the 3D rendering is performed at the server side and the clients receive only streamed images which they can use for interactive navigation through the scene and communication with other clients. This mode is quite typical of grid-based visualization when either a very powerful server or a cluster of networked computers are used for rendering. Its disadvantage is in a limited resolution of the image and slow update rate due to bandwidth limitations. Another scenario assumes that all the models are preloaded to client computers and only *even transmission* is performed between the clients and an optional server. This mode is quite common for MMORPG games and assumes downloading of an advance rendering engine and a large model database which may have gigabytes of information stored. Though this method provides very photorealistic rendering, the disadvantage is in difficulties which result when a new model is introduced into the scene, i.e. it has to be somehow delivered to all the participating clients to update the model databases. A compromise method which can be called *model and event transmission* progressively downloads the model of the scene on to client computers while it is being rendered. The rendering engines can be light-weighted like plug-ins to web browsers (e.g. VRML and X3D plug-ins to MS Internet Explorer and Mozilla Firefox), as well as stand-alone applications (viewers) with some preloaded textures and models (e.g. SecondLife viewer).

In this article we consider problems associated with creation of educational cyberworlds using VRML and X3D and the model transmission visualization method. Since VRML and X3D do not natively support collaboration, a third party communication platform has to be used or developed. Examples of such platforms are open-source DeepMatrix (*http://www.deepmatrix.org*), blaxxun Communication Platform (*http://www.blaxxun.com*), and Bitmanagement Collaborate (*http://www.bitmanagement.com*).

Each of these platforms has unique features. *DeepMatrix* is a popular open-source multi-user 3D communication platform that features chat, shared objects, and shared events. It is written in Java consisting of one server application and one client applet that works with different web3D plug-ins. DeepMatrix can use either existing VRML browser as the client via the Java EAI, or its own VRML client written in Java. The DeepMatrix's functionality includes text chatting facilities and shared events. However, the shared events in DeepMatrix do not provide any support of locking mechanism.

The *blaxxun Communication Server* is available as a commercial product as well as through a complimentary web-access albeit with limited functions. It consists of two main components: the server and the client. The server acts not only as an HTTP server, which holds the VRML scene and HTML web pages, but also some special communication servers, which serve two purposes: /1/ To hold additional information of the shared virtual scene and /2/ to exchange shared information across different clients. In its SDK, it provides a text chat box and shared events. By using the text

chat box, users can type text messages and communicate with other users within the same session. It is also possible to install on top of it the *Bitmanagement BS Contact VRML/X3D* (*http://www.bitmanagement.com*) VRML/X3D browser plug-in for MS Internet Explorer which then will do VRML/X3D visualization, while communication with the server is performed by the blaxxun Contact. However, in that case X3D files cannot be used as root scene files but only can be called from VRML files.

The *Bitmanagement Software Collaborate System* is a very new networked communication platform which supports both VRML and X3D. It has a platform independent server which supports shared events, text chatting, server side computation and client identification mechanism however, it is still under development and plenty of the technical details are not fixed and implemented.

The Virtual Campus of Nanyang Technological University (NTU) in Singapore [7] is an example of a photo-realistic shared virtual world built using VRML and the blaxxun Communication Platform. It includes models of the land, buildings, interiors, avatars, and texture images resembling the real campus of NTU (Fig. 1). Besides the exteriors of the university buildings, there are also interiors of the main places, tutorial rooms, lecture theaters and student hostels. All these interiors can be used for virtual meetings, classes and other learning activities. There is a large large set of predefined private houses as well as different household items which can be obtained from the virtual shopping mall. Many visitors to the Virtual Campus are computer graphics students, who come to study concepts related to virtual reality and shape modeling. This is a mandatory part of their course and the Virtual Campus is used during lectures, as well as after classes for consultations. The students go to their favorite places, meet with friends in their hostel rooms, and attend collaborative modeling sessions in the virtual shape modeling lab. They also can watch educational videos in the tutorial rooms, attend video lectures in the lecture theaters or watch the university's television channel. Visitors from around the world usually just wander around and chat, adding an international flavor to this place. The Virtual Campus often serves as a guide for foreign students who may consider studying at the university.

Fig. 1. Snapshots of the Virtual Campus of NTU

2.2 Teaching Geometry with a Computer

There are a few commercially available tools which are commonly used for doing research on mathematics and geometry as well as for learning and teaching these subjects, e.g., Mathcad (http://www.ptc.com/appserver/mkt/products), Maple (http://www.maplesoft.com), Mathematica (http://www.wolfram.com/products/mathematica), MATLAB (http://www.mathworks.com/products/matlab), and Geometer's Sketchpad (http://www.dynamicgeometry.com). Among other features, these tools allow learners to perform visualization of geometric shapes (2D/3D curves and surfaces) and some provide further means for making web-enabled interactive applications. However when using these tools, the learners are only able to see images. It could be, however, more interesting to get immersed within the 3D scene and explore the shapes which are being modeled. Moreover, it would be even more beneficial if this immersion could be done collaboratively with other learners and the instructor.

There are a few examples of such collaborative approaches to learning geometry in virtual augmented worlds [8] and in VRML spaces on the web [9, 10], however these projects are restricted to a limited class of geometric objects and only teach geometry rather than provide the learner with the ability to see the geometric shapes behind the mathematical formulas and illustrate how mathematics creates and supports immersive virtual spaces, as we are proposing to do in our project.

3 Function-Based Web Visualization and Haptic Rendering in Shared Virtual Spaces

In our project we seek to further expand the shared collaborative potential of virtual educational environments by developing an efficient way of exchanging geometric, appearance and physical properties across the network. Instead of traditionally used polygon and voxel based models, we are using relatively small mathematical functions, which like individual DNAs define the geometry, appearance and physical properties of the objects. We are also using haptic force-feedback in shared virtual environments on the web so that the learners can also make physical contact with objects in the virtual scene, as well as with other learners. Haptic technology provides an interface to the user via the sense of touch by applying forces, vibrations and/or motions to the user. We are expecting, soon, the arrival on the consumer market of affordable interactive 3D touch devices. For example, Novint Falcon (http://www.novint.com) is currently offered in the US at two hundred dollars, which will soon make haptic communication as common as interactions using mouses and joysticks.

A function-based approach to shape modeling assumes that mathematical functions are used for defining geometric shapes rather than traditional polygons, voxels or points. Hybrid function-based shape modeling in application with VRML and X3D was introduced and further developed in [11-13]. It allows for the defining of time-dependent geometric shapes, their appearances, physical properties and transformations by analytical functions which can be used concurrently as individual formulas or as java-style function scripts. FVRML/FX3D opens VRML and X3D to practically any

type of geometry, 3D colors and geometric textures. It also introduces to VRML and X3D set-theoretical operations (union, intersection, difference), as well as allows for defining any other operations (e.g., morphing) and physical properties of objects. In contrast to the polygon-based models of VRML/X3D, the function-based extension allows for a greater reduction of the model size and provides an unlimited level of detail. The defining functions can be functions of time which in turn allows for easy definition of sophisticated animation applied to geometry and appearance of the resulting shapes. These function-defined shapes can be used together with the standard VRML and X3D shapes and appearances, as well as allow for their use as parts the standard VRML/X3D shape and appearance fields. We implemented a plug-in to two VRML and X3D browsers (blaxxun Contact and Bitmanagement BS Contact VRML/X3D). While staying within the VRML/X3D rendering pipeline designed for objects built from polygons, we are able to perform haptic rendering of shared VRML and X3D scenes by touching surfaces of the objects with different desktop haptic devices. Besides this, VRML/X3D objects can be converted to solid objects by defining their inner densities using mathematical functions. This can be done by typing analytical functions straight into the VRML/X3D code or by defining them in dynamic-link libraries. Geometric solid objects can also be reconstructed from CT and MRI data. Since these function-defined models are much smaller in size, we can efficiently use them in web-based collaborative projects.

For defining geometry, appearance, transformations and physical properties, three types of function definitions can be used concurrently: implicit, explicit and parametric.

Implicit functions are defined as $f(x,y,z,t)=0$, where x, y, z are Cartesian coordinates and t is the time. When used to define geometry, the function equals to zero for the points located on the surface of a shape.

Explicit functions are defined as $g=f(x,y,z,t)$. The function value g when evaluated at any point of the 3D space can be used either as a value of some physical property like density or as an argument for other functions (e.g., to define colors or parameters of transformations) or as an indicator of the sampled point location. Thus, explicit functions can be used to define bounded solid objects in the FRep sense [14]. In this case, the function equals to zero for the points located on the surface of a shape. Positive values indicate points inside the shape and negative values are for the points outside the shape.

Parametric functions define surfaces, solid objects, force vector coordinates and colors as:

$$x=f_1(u,v,w,t);\ \ y=f_2(u,v,w,t);\ \ z=f_3(u,v,w,t)$$
$$r=f_1(u,v,w,t);\ \ g=f_2(u,v,w,t);\ \ b=f_3(u,v,w,t)$$

where x, y, z are Cartesian coordinates of the points, r, g, b are values of colors, u, v and w are parametric coordinates, and t is the time.

Thus, geometry, appearance, and physics are defined by implicit, explicit or parametric functions in their own domains and then merged together into one shape by some mapping functions. Appearance (colors and 3D textures) and physical properties are associated with the underlying geometry of the shape.

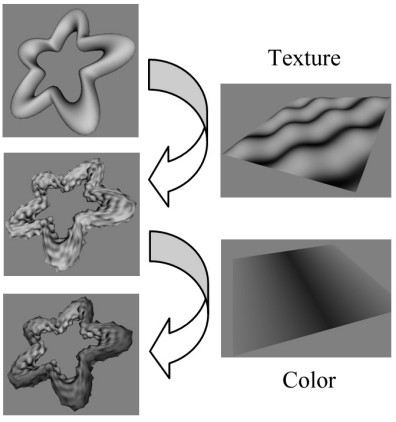


Fig. 2. Modeling shape by consecutive definition of its geometry, texture and color

For example, we can define an original shape parametrically:

$$x = (1 - 0.3\sin 5v) \times \cos v \times (4 + \cos u)$$
$$y = (1 - 0.3\sin 5v) \times \sin v \times (4 + \cos u)$$
$$z = \sin u$$
$$u = [-\pi, \pi], v = [-\pi, \pi]$$

Then, we map to this shape the geometric texture which is Perlin noise displacement defined by the explicit function:

$$g = 0.1(\sin 2\pi x \sin 2\pi y + \sin 2\pi x \sin 2\pi z + \sin 2\pi y \sin 2\pi z)$$

Finally a parametrically-defined color is applied onto the textured shape by mapping the colored plane onto the shape:

$$r = 1; g = |\sin u|; b = 0;$$

With these two parametric and one explicit functions, the final shape is defined as it is shown in Fig. 2.

FVRML offers ten additional nodes, which are *FShape*, *FGeometry*, *FAppearance*, *FMaterial*, *FTexture3D, FPhysics, FDensity, FFriction, FForce* and *FTransform*. These nodes can be used together as well as with the standard VRML and X3D nodes (Fig. 3).

The *FShape* node is a container similar to the VRML's *Shape* node. It contains *FGeometry* or any standard *geometry* node, and *FAppearance* or the standard *Appearance* node. These nodes define the geometry and the appearance of the shape, respectively. *FShape* may be called from *FTransform* node or from the standard *Transform* node.

The *FGeometry* node is designed to define a geometry using implicit, explicit or parametric functions. It contains either one string, or a script defining a function representing the geometry, or a URL to the function definition.

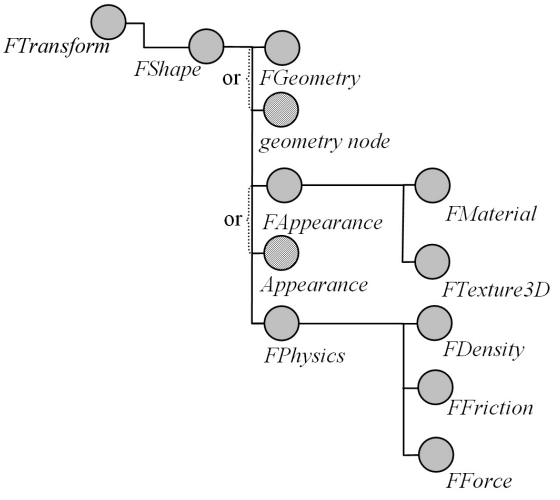


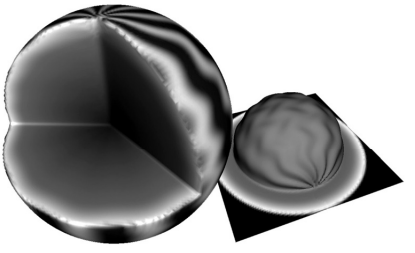
Fig. 3. Scene diagram of the function-defined nodes and their using with the standards nodes of VRML and X3D

The *FAppearance* node may contain *FMaterial* or the standard *Material*, the standard *Texture* node, and *FTexture3D* node. *FTexture3D* defines geometric textures while the standard *Texture* node is used for image texture mapping. *FMaterial* node defines diffuse, specular, and emissive colors, as well as transparency, ambient intensity, and shininess with parametric and explicit functions. Parametric functions directly define color values *r*, *g*, *b*. When the color is defined with an explicit function, its values are linearly mapped to the *r*, *g*, *b* values of the actual color with a user-defined color interpolation map. *FTexture3D* node defines 3D geometric textures using explicit and parametric functions. These functions define displacements of the original shape's geometry defined in *FGeometry* or in any standard *geometry* node. The functions are defined either in one string, or in a function script, or via a URL to the function definition.

If the standard *Geometry* node is used in place of the *FGeometry* node, the standard shapes of VRML will be assigned an appearance defined in the *FAppearance* node. This ability to use the function-based nodes in combination with the standard geometry and appearance nodes of VRML adds many new features to VRML, some of which are illustrated in this chapter.

The *FPhysics* node is used for defining physical properties associated with the defined geometry. *FPhysics* contains *FDensity, FFriction* and *FForse* nodes. *FDensity* node is used for defining material density inside the shape. It can be examined with a moving haptic device handle. Density is defined by explicit functions of coordinates *x, y, z*. *FFriction* node is used for defining friction on the surface. It can be examined with a moving haptic device actuator. Friction is in turn defined by the explicit functions of coordinates *x, y, z*. The surface of the object can also be made non-tangible. *FForce* node is used for defining a force field associated with the geometry of the shape. The force vector at each point of the space is defined by three parametric functions for its *x*, *y*, and *z* components.

FTransform node contains function-defined operations. It may contain *FShape* and other *FTransform* nodes as its children. *FTransform* contains either one string, or a script defining a function representing the operation, or a URL to the function definition as well as other fields to preserve the consistency with the standard *Transform node*. Both, set-theoretic and affine transformations can be defined as functions of time in *FTransform* node.



```
FShape {
    geometry FGeometry {
        definition "(1-x*x-y*y-z*z)&(-((-x+z)&y&z))"
        bboxSize 2 2 2
        parameters 0
        resolution [100 100 100]}
    appearance FAppearance {
        material FMaterial {
        diffuseColor  "sqrt(x*x+y*y+z*z)+
            0.01*(sin(12*atan2(x,z+0.04*sin(y*25)))-0.7)"
            patternKey [0 0.8 0.96 0.98 1]
            patternColor [0.4 0 0,1 0 0,1 1 1,0 1 0,0 0.2 0]
            } }
    physics FPhysics {density FDensity {
            definition "0.3 + sqrt(x*x+y*y+z*z)*0.6"}
            friction FFriction {
            enable TRUE
            definition "0.1" }  }
}
```

Fig. 4. Modeling "watermelon": final shape and the isosurface of the color function

An example of the modeling of a "watermelon" is given in Fig. 4. The geometry, appearance and physical properties of this object are defined with explicit functions. The geometry is defined as a sphere with a piece cut away from it by a semi-infinite solid object defined by 3 planes.

$$(1 - x^2 - y^2 - z^2)\& (-((z - x)\&y\&z))$$

The color is then represented by a 3D field defined in the same geometric coordinate space. The function of this field is defined as a distance from the origin

$$\sqrt{x^2 + y^2 + z^2}$$

with distortions are defined by a noise function

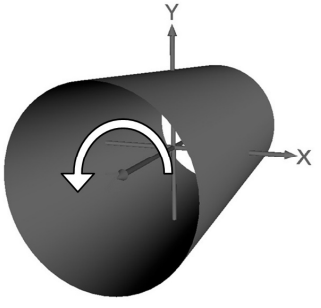
$$0.01(\sin(12\text{atan}(x, z + 0.04\sin(25y))) - 0.7)$$

The function values are then linearly mapped to the color values according to a designed color map. Sampling of this color field is also shown in Figure 4. The uneven shape of the "color" surface was used for making green patterns on the surface of the "watermelon". This was achieved by mapping the respective function values from 0.98 to 1 to RGB values of colors ranging from [0, 1, 0] to [0, 0.2, 0]. The colors inside the watermelon are also created by linear mapping of the respective color function values to different grades from yellow, through white, and finally to red colors.

The density of the watermelon is then defined as a distance function with a value 0.3 in the center of the object increasing to 0.9 at its surface. The friction on the surface is a constant 0.1.

Defining complex shapes, appearances, physical properties and operations usually assumes the use of multiple formulas and temporary variables. This requires a script-like mathematical language. To be consistent with VRML and X3D, which have a JavaScript (VRMLScript) as a standard feature, and to ease the learning curve, FVRML/FX3D emulates a subset of JavaScipt.

Hence, in another example we model a pipe with its geometry defined as a parametric cylinder with no tangible surface and a force field defined as a whirlpool moving along the axis of the cylinder (Fig. 5). For a force field, at each point of the defined geometric volume a force vector will be calculated by the parametric scripts.




```
FShape {
        geometry FGeometry {
                definition "x=cos(u*2*pi); y=sin(u*2*pi); z=-2+4v;"
                parameters [0 1 0 1]
                resolution [100 100 100]}
        appearance Appearance {
                material Material {
                diffuseColor 1 1 1 }  }
        physics FPhysics {
                friction FFriction {
                enable FALSE}
                force FForce { parameters [-1 1 -1 1 -1 1] definition "
function parametric_x(u,v,w) {x=-v; y=u; z=0.3; return  x/sqrt(x*x+y*y+z*z);}
function parametric_y(u,v,w) {x=-v; y=u; z=0.3; return y/sqrt(x*x+y*y+z*z);}
function parametric_z(u,v,w) {x=-v; y=u; z=0.3; return  z/sqrt(x*x+y*y+z*z);}"
 }  }  }
```

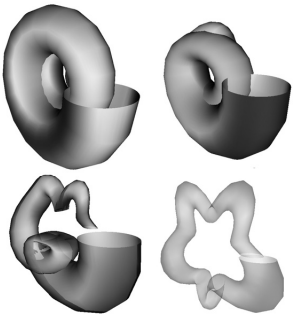
Fig. 5. Modeling a cylindrical tube with a force field inside it

In Fig. 6, four snapshots of an animated geometric morphing from a snail-like shape to a star-like shape, and the respective FVRML code of this morphing transformation, are shown. Both, the initial and final shapes are defined parametrically in the function script. The time-dependent shape is defined as a linear interpolation between the initial and the final shapes. The internal timer of the *FShape* node is used for defining an infinite looping with a period of 5 sec, while the time within the script changes in the range [0, 1]. Fancy color and variable transparency are defined by explicit functions. In this example, the morphing transformations are defined using the linear function interpolation:

$$f(\mathbf{p}) = f_1(\mathbf{p}) + \big(f_2(\mathbf{p}) - f_1(\mathbf{p})\big) \cdot t$$

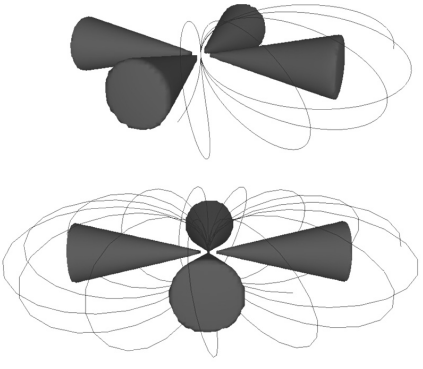
where parameter t is in the range [0, 1]. In this formula, functions f_1 and f_2 may define the initial and the final shape's geometry, texture or color. For implicit and explicit functions, vector **p** defines Cartesian coordinates *x, y, z*, with only one such morphing function needed. For parametric functions, vector **p** defines coordinates *u, v, w* in parametric domain requiring three morphing functions to define Cartesian coordinates *x, y, z*, respectively.

In Fig. 7, there are two snapshots of an animated scene where the counter-clockwise rotation of the implicitly defined cones is defined in *FTransform* node as a function of time. In this example, both internal and external time sensors are used to define a gap between two cycles. The internal timer defines a single 5 sec execution of the node. The external timer defines an infinite looping with a 7 sec duration of



```
FShape {
  enabled TRUE  cycleInterval 5  loop TRUE
    appearance FAppearance {
      material FMaterial  {
          diffuseColor "sin(sqrt(x*x+y*y+z*z) * pi)"
          patternColor [0 1 0 1 0 0]   patternKey [-1 1]
          transparency "0.75*sin(t/2)" }  }
    geometry FGeometry {
      resolution [50 50]  parameters [0 2 0 2 -1 1]
      definition "
        function snail_x(u,v,w) { return 0.03*pi*v*(10+5*cos(pi*u))*cos(3*pi*v); }
        function snail_y(u,v,w) { return 0.03*pi*v*(10+5*cos(pi*u))*sin(3*pi*v);  }
        function snail_z(u,v,w) { return 0.03*pi*v*(6+5*sin(pi*u));  }
        function star_x(u,v,w) { return (0.3-0.1*sin(v*5*pi))*cos(v*pi)*(4+cos(u*pi)); }
        function star_y(u,v,w) { return (0.3-0.1*sin(v*5*pi))*sin(v*pi)*(4+cos(u*pi)); }
        function star_z(u,v,w) { return 0.2*sin(u*pi);  }
        function parametric_x(u,v,w,t){ return snail_x(u,v,w)*(1-t)+star_x(u,v,w)*t;  }
        function parametric_y(u,v,w,t){ return snail_y(u,v,w)*(1-t)+star_y(u,v,w)*t;  }
        function parametric_z(u,v,w,t){ return snail_z(u,v,w)*(1-t)+star_z(u,v,w)*t;  }"
    }  }
```

Fig. 6. Geometric shape morphing and appearance transformation



```
FTransform {
  cycleInterval 5 loop FALSE
  rotation "x=0; y=0; z=1; a=t*6.282;"
  children [
    FShape {
      appearance Appearance {
        material Material { diffuseColor 1 0 0}  }
      geometry FGeometry {
        definition "(2-x*x-y*y-z*z)&((-x*x/0.1-z*z/0.1+y*y/2)|
        (-y*y/0.1-z*z/0.1+x*x/2))"
        resolution [100 100 100]  bboxCenter 0 0 0  bboxSize 3 3 3 }
      }    ] }
    DEF Time TimeSensor { loop TRUE   cycleInterval 7 }
    ROUTE   Time.cycleTime   TO   shape.startTime
    FShape {
      cycleInterval 10   loop TRUE
      polygonizer  "analytical_curve"
        appearance Appearance {material Material
          {diffuseColor 0 0 1   emissiveColor 0 0 1 }  }
        geometry FGeometry {
          resolution [300 300]  parameters [0 30]
          definition "
              x=0.25*(4*cos(u*t*pi)+4)*cos(u*t*pi/15);
              y=-0.25*( (4*cos(u*t*pi)+4)*sin(u*t*pi/15);
              z=0.5*sin(u*t*pi);" }    }
```

Fig. 7. Function-defined rotating cones and 3D curves

each cycle, which results in a 2 sec time gap between the cycles. In this scene there is also a parametrically defined animated curve which spins around the first shape following the internal timer. Note that for the cones and for the curve we use the standard VRML *Appearance* node for defining their appearance instead of *FAppearance*.

Based on FVRML/FX3D, then we have designed and developed a shared virtual laboratory for teaching subjects rich with mathematics, such as geometry, topology, calculus, shape modeling, and physics. This creates a shared virtual environment where the learners are able to collaboratively model complex 3D shapes by defining their geometry, appearance and physical properties using analytical implicit, explicit and parametric functions. The tool visualizes the language of mathematics in forms of point clouds, curves, surfaces and solid objects. The learners may go inside the 3D scene being modeled and explore it by walking/flying through it (Fig. 8) as well as haptically investigating it.

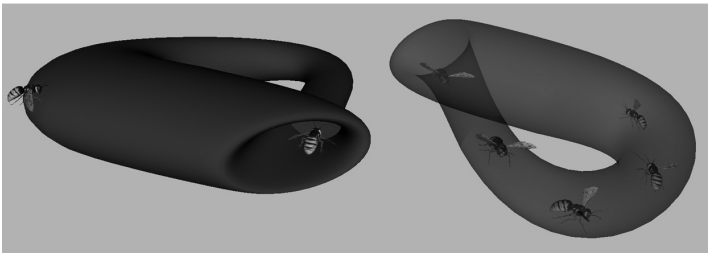


Fig. 8. Flying inside a Klein bottle defined by parametric functions

At a first glance out tool shares some aspects with conventional math software tools mentioned above, however it has totally different aim and goal. Our tool is not intended for making hi-end geometric models such as those that can be created with CAD tools like Maya and 3D Studio MAX. Instead it rather emphasizes the modeling process itself by demonstrating how complex geometry, appearance and physical properties can be defined by small mathematical formulas. Thus the learners are able to enter the defining functions, either as individual formulas or as function scripts. No knowledge of any programming language is required from them. In contract to the math tools which offer an overwhelming variety of complex features and usually have a steep learning curve, our software has an intuitive and relatively simple user interface which learners are interested in. Hence, our aim is not to create a professional 3D modeling package, but rather a simple and intuitive interactive 3D tool to be used within an immersive shared environment for educational and research purposes.

In this virtual environment, the learners define initial shape, tools, geometric and appearance operations analytically or select from the predefined sets. This makes our software open to any customization by the learners and educators. Models defined by analytical formulas or function scripts can be easily modified on-the-fly in the virtual scene either by editing the formulas or by doing iterative interactive modifications. The respective function representations are generated for each interactive modification and displayed to the learners by request.

In the user interface of our tool (Fig. 9) there are three parts: /1/ the shared 3D collaboration scene where haptic rendering is allowed (top-left), /2/ the control panel (top-right) and /3/ the command-chat windows (bottom). The 3D haptic modeling scene and the chat area are shared among the learners in the same session, while the control panel is learner-specific and not shared. The learners can type commands as well as chat in the command-chat pane concurrently. To change the current tool, the pre-defined tools and existing customized tools can be chosen from the drop-down menu in the control panel. The learners may modify a shape by defining its geometry and color either parametrically or with implicit and explicit functions.

There are just a few easy to use commands, which the learners can type in the chat box of the browser and immediately see how the shape changes. The software filters out the shape-modeling commands from other chat messages and processes them accordingly. The function description of the current shape can be displayed in a separate console window, and saved for future use. Shapes can be also defined

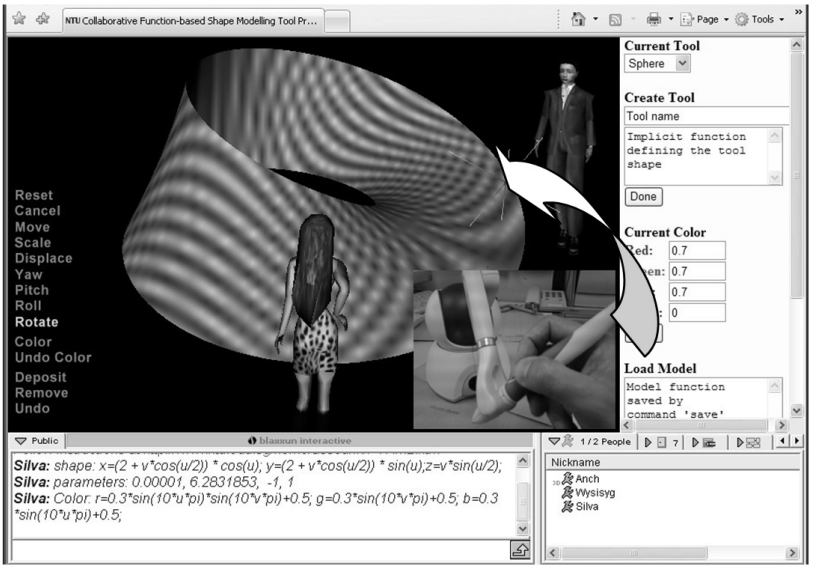


Fig. 9. Haptic exploration of a Möbius Strip

interactively by adding and removing material which is borrowed from the current tool. Any tool shape can be defined by a mathematical function. Before it is applied, the tool can be scaled, rotated and displaced with reference to the shape's surface.

In Fig. 10, we illustrate an example of a free-form collaborative design session using the software. First, we define the original object as a distorted sphere with another smaller sphere subtracted from it. The original sphere is defined by the following function:

$$0.8^2 - x^2 - y^2 - z^2 \geq 0$$

which defines a solid object for the points where the function value is ≥ 0.

The 3D texture function making a distortion of the sphere is defined by the displacement function:

$$0.03\ (\sin(12\ \mathrm{atan}(x,\ z + 0.04\ \sin(25\ y))) - 0.7) \geq 0$$

The smaller sphere to subtract from the distorted one is defined by:

$$0.7^2 - x^2 - y^2 - z^2 \geq 0$$

The final object is defined by typing the following command:

shape: (0.8^2–x^2–y^2–z^2+0.03(sin(12*atan2(x,z+0.04*sin(y*25)))–0.7))&*
(–(0.7^2–x^2–y^2–z^2))

where *&* is a symbol of intersection operation, while *&(- function)* is the defined subtraction operation.

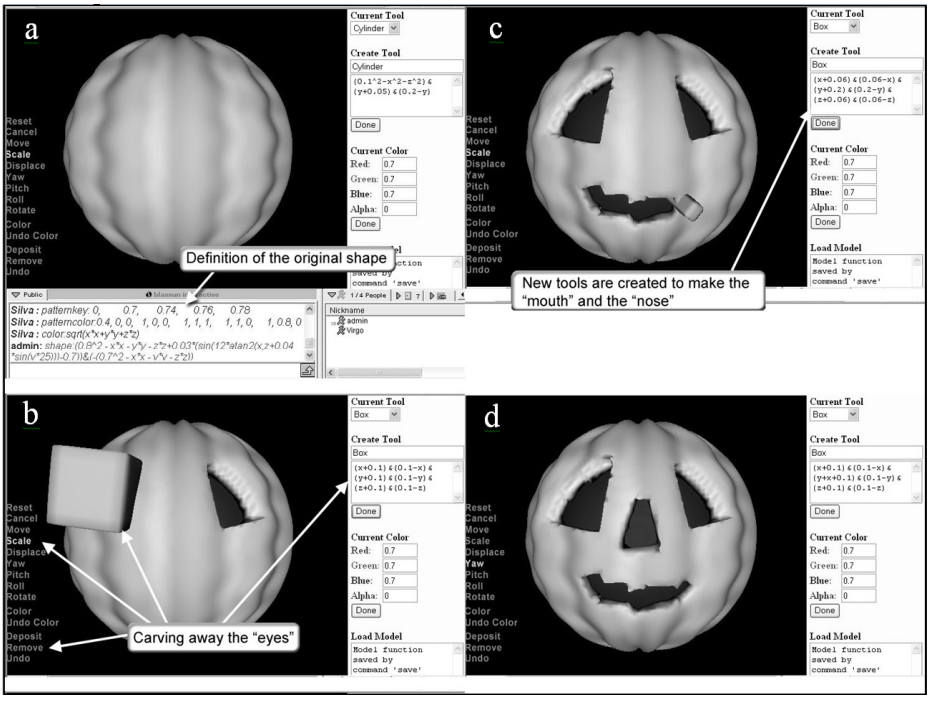


Fig. 10. Example of shape modeling

The 3D color of this object is then defined by an explicit distance function:

$$g = \sqrt{x^2 + y^2 + z^2}$$

which values are linearly mapped to the colors, based on the key-values defined by *patternkey* and color values in *patterncolor* commands. This is achieved by typing commands, which is done in Figure 10a by another user participating in the design session:

patternkey: 0, 0.7, 0.74, 0.76, 0.78
patterncolor: 0.4, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0.8, 0
*color: sqrt(x*x+y*y+z*z)*

This results in a 3D color which varies from yellow-green colors on the surface, to white-pink immediately beneath it, and finally to grades of red color inside the object.

Next, one of the users selects a predefined tool 'box', locates it on the surface of the object, interactively changes its size and orientation, and finally cuts away the material with it to make "eyes" (Figure 10b). The other user can see the changes on his computer as well as concurrently participate in the design. Next, the users change the tool's formula to make a cutter to carve away piece by piece a "mouth" (Fig. 10c). Finally, the toll's formula is changed one more time to make a cutter for carving away a "nose" (Fig. 10d). The final design can be saved either into VRML or X3D file.

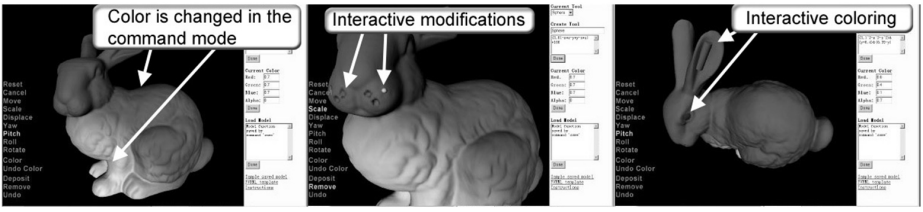


Fig. 11. Interaction work with the model reconstructed from CT data

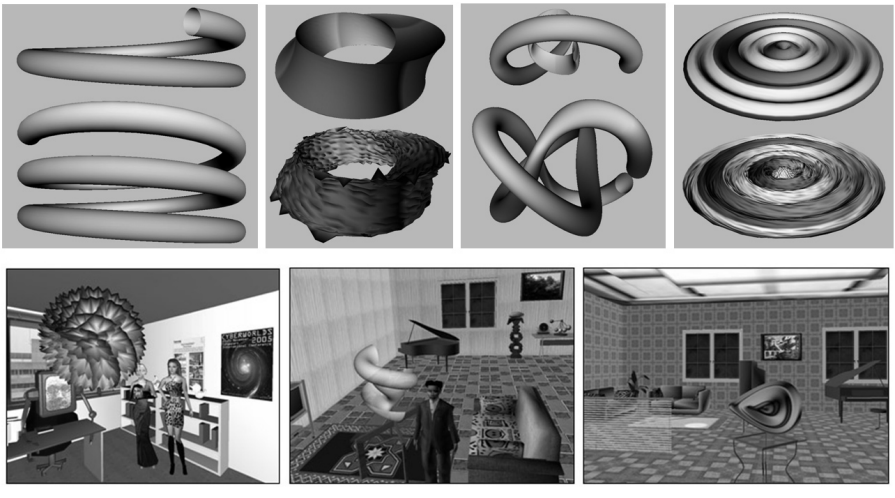


Fig. 12. Student experiments with different types of dynamic geometry and appearance defined by implicit, explicit and parametric functions

In the next example (Fig. 11) the basic volume model of a bunny was acquired by a CT scanner as three dimensional volume density values. The function model was reconstructed by applying a trilinear interpolation function which was added as a native function to the FVRML/FX3D plug-in. At this step, the model is shown as a gray object, which is the default color of VRML and X3D. Some details which can be seen on the original physical object are missing on the reconstructed model due to the scanning precision and the reconstruction procedure. First, the reconstructed model is loaded into the design environment and its color is changed in the command mode so that it varies from bright white on the belly to dark gray on the back. (Fig. 11a). Next, the geometry of the bunny's mouth and nose is modified by applying interactive operations to the object. It is implemented as modifications of the function scripts without changing the original CT volume data (Fig. 11b). Finally, the eyes and the ears of the bunny are colored interactively as dark red and semi-transparent dark yellow, respectively, by various interactive tools (Fig. 11c). The created object's VRML code is very small since it only contains modifications described in FVRML while the original CT volume data is referenced as a hyperlink.

In Fig. 12 we show examples of student experiments with definitions of different types of dynamic geometry and appearance by using implicit, explicit and parametric functions, as well as snapshots of the scenes of a computer graphics assignment: "Implicit Fantasies and Parametric Metamorphoses in Cyberworlds", where the students have to design a shared VRML environment containing animated function-defined objects.

4 Future Trends of Educational Cyberworlds

Increasingly we believe that educational cyberworlds aided by the inclusion of haptic technology will not only flourish but that they will become a mainstay of the educational learning narrative. There are several key imperatives governing this assertion.

One, significant numbers of 'Digital Natives' have grown up with game-based virtual environments, and increasingly they will see them as simply natural extensions of their online experiences—in fact they are likely to demand them. Having experienced throughout their childhood freedom from geographical space-time constraints, students of diverse backgrounds, interests, and talents will increasingly insist that they be able to work together and actively build, share and demonstrate for both themselves and their instructors what they know, and how they come to know it.

Two, from an educational and pedagogical perspective, cyberworlds will be seen to support - in significant ways, the key principles of good practice in higher education [15], identified as crucial to success in the teaching and learning process. More importantly, this reality will eventually translate into an academic administrative impulse to support the active integration of visual immersive environments within the curriculum process. Administrators will eventually realize that their learners have changed and changed significantly, requiring bold and creative applications of such cyberworlds to attract and hold this increasingly sophisticated digital clientele.

Three, bandwidth growth and the increasingly innovative application and integration of Web 2.0 services within existing and rapidly evolving educational cyberworlds, perforce, will require more and more academics to review their teaching and learning practices and processes, especially those that can and will soon be readily supported within educational cyberworlds, i.e., synchronous classrooms, chat lines, streaming videos, interactive lab experiments, just in-time content delivery via pedagogical agents - who access federated content repositories and deliver individually crafted learning experiences tailored to the learner's dominant learning traits - and individual and group created portfolio display spaces.

The future is here. It is tactile, interactive, visually driven, expressive, community oriented and taking flight within immersive shared virtual spaces. These cyberworlds are the tide of the future which, to paraphrase Shakespeare, taken at full flood will lead to fortune; missed will result in nothing but misery.

5 Conclusion

We have developed a function-based approach to web visualization where relatively small mathematical formulas, like some individual DNAs are used for defining the

object's geometry, appearance and physical properties. Based on this approach we propose a new virtual haptic shared environment architecture and tools that can be used in immersive shared virtual applications. With this approach we are able to provide haptic feedback from both standard objects of VRML/X3D, as well as from advanced 3D solid objects and force fields defined by functions. We have designed and developed an innovative application for collaborative teaching of subjects requiring strong 3D geometric interpretations of the theoretical issues, which, in turn, caters to the diverse needs of learners and has the space for continuous improvement and expansion. Our software unifies, under one roof, and provides learners with the ability to: /1/ interactively visualize geometry and appearance defined by analytical formulas, /2/ perform a walkthrough the created scene, as well as haptically explore it, and /3/ make the scene a part of any other shared virtual scene defined by VRML or X3D. As well our tool puts the emphasis on the modeling process itself by demonstrating how geometry, appearance and physical properties can be defined by mathematical formulas. Another benefit is that the software does not require purchasing any licenses and is available anytime anywhere on any Internet-connected computer.

The proposed movement to extend virtual haptic applications within digital learning spaces is a significant first step towards the creation of the Intelligent Memory Systems envisioned by Hawkins [16]. Such systems are thought to be capable of analyzing, synthesizing, evaluating the learner's needs and integrating these within either a tightly or loosely constructed learning sequence.

Acknowledgements

This project is supported by the Singapore National Research Foundation Interactive Digital Media (NRF IDM) grant "Visual and Haptic Rendering in Co-Space".

References

1. Kunimune, S., Nagasaki, E.: Curriculum Changes on Lower Secondary School Mathematics of Japan - Focused on Geometry. In: 8th International Congress on Mathematical Education (1996) (retrieved January 8, 2008), http://www.fi.uu.nl/en/Icme-8/WG13_6.html
2. Mathematics Syllabus Primary. Ministry of Education Singapore (2007) (retrieved January 8, 2008), http://www.moe.gov.sg/cpdd/pri_maths_links.htm
3. Secondary Mathematics Syllabuses. Ministry of Education Singapore (2007) (retrieved January 8, 2008), http://www.moe.gov.sg/cpdd/syllabuses.htm
4. TSG 16: Visualisation in the teaching and learning of mathematics, 10th International Congress of Mathematical Education, Copenhagen, Denmark, July 4-11 (2004), http://www.icme-organisers.dk/tsg16
5. Gibson, W.: Neuromancer. Ace (1984) ISBN-13: 978-0441569595
6. Prasolova-Førland, E., Sourin, A., Sourina, O.: Cyber-campuses: Design Issues and Future Directions. The Visual Computer 22(12), 1015–1028 (2006)
7. Sourin, A.: Nanyang Technological University Virtual Campus. IEEE Computer Graphics & Applications 24(6), 6–8 (2004)

8. Kaufmann, H., Schmalstieg, D.: Designing Immersive Virtual Reality for Geometry Education. In: IEEE Virtual Reality Conference, VR 2006, pp. 51–58. IEEE CS Press, Los Alamitos (2006)
9. Song, K.S., Lee, W.Y.: A virtual reality application for geometry classes. Journal of Computer Assisted Learning 18, 149–156 (2002)
10. Pasqualotti, A., Dal Sasso Freitas, C.M.: MAT3D: A Virtual Reality Modeling Language Environment for the Teaching and Learning of Mathematics. Cyber Psychology & Behavior 5(5), 409–422 (2002)
11. Liu, Q., Sourin, A.: Function-based shape modelling extension of the Virtual Reality Modelling Language. Computers & Graphics 30(4), 629–645 (2006)
12. Liu, Q., Sourin, A.: Function-defined Shape Metamorphoses in Visual Cyberworlds. The Visual Computer 22(12), 977–990 (2006)
13. Wei, L., Sourin, A., Sourina, O.: Function-based Haptic Interaction in Cyberworlds. In: 2007 International Conference on Cyberworlds, pp. 225–232. IEEE CS Press, Los Alamitos (2007)
14. Pasko, A., Adzhiev, V., Sourin, A., Savchenko, V.: Function Representation in Geometric Modeling: Concepts, Implementations and Applications. The Visual Computer 11(8), 429–446 (1995)
15. Chickering, A.W., Gamson, Z.F.: Seven principles for good practice in undergraduate education. American Association of Higher Education Bulletin, 3–7 (1987)
16. Hawkins, J.: On Intelligence. Owl Books, New York (2004)

The Voronoi Diagram of Circles and Its Application to the Visualization of the Growth of Particles

François Anton[1], Darka Mioc[2], and Christopher Gold[3]

[1] Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads
DK-2800 Kongens Lyngby
Denmark
fa@imm.dtu.dk
[2] Department of Geodesy and Geomatics Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, New Brunswick, Canada, E3B 5A3
dmioc@unb.ca
[3] School of Computing
University of Glamorgan
Pontypridd, CF37 1DL, UK
cmgold@glam.ac.uk

Abstract. Circles are frequently used for modelling the growth of particle aggregates through the Johnson-Mehl tessellation, that is a special instance of the Voronoi diagram of circles. Voronoi diagrams allow one to answer proximity queries after locating a query point in the Voronoi zone it belongs to. The dual graph of the Voronoi diagram is called the Delaunay graph. In this paper, we first show a necessary and sufficient condition of connectivity of the Voronoi diagram of circles. Then, we show how the Delaunay graph of circles (the dual graph of the Voronoi diagram of circles) can be computed exactly, and in a much simpler way, by computing the eigenvalues of a two by two matrix. Finally, we present how the Voronoi diagram of circles can be used to model the growth of particle aggregates. We use the Poisson point process in the Voronoi diagram of circles to generate the Johnson-Mehl tesselation. The Johnson-Mehl model is a Poisson Voronoi growth model, in which nuclei are generated asynchronously using a Poisson point process, and grow at the same radial speed. Growth models produce spatial patterns as a result of simple growth processes and their visualization is important in many technical processes.

Keywords: Voronoi diagram of circles, Visualization of nucleation and growth of particles, Johnson-Mehl tessellations, growth models.

1 Introduction

The proximity queries among circles could be effectively answered if the Delaunay graph for sets of circles could be computed in an efficient and exact way.

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 20–54, 2009.

This would require the embedding of the Delaunay graph and the location of the query point in that embedded graph. The embedded Delaunay graph and the Voronoi diagram are dual subdivisions of space, which can be stored in a quad-edge data structure [GS85]. The original contribution of this paper is a necessary and sufficient condition of connectivity of the Voronoi diagram of circles, an exact and much simpler algorithm for the Delaunay graph of circles (the dual graph of the Voronoi diagram of circles, with no assumption on the disjointness of circle sites), and its application to the visualization of the growth of particles.

The first and most explored Voronoi diagram is the Voronoi diagram for a set of points [Vor07, Vor08, Vor10] in the Euclidean plane or in the three-dimensional Euclidean space (see Figure 1). Voronoi diagrams have been generalised in many different ways including by modifying the space in which they are embedded (see [Auren87, OBSC01] for a general survey of Voronoi diagrams): higher dimensional Euclidean spaces, non Euclidean geometries (e.g. Laguerre geometry, hyperbolic geometry, etc.). Fewer generalisations of Voronoi diagrams correspond to extending the possible sites from points to circles, i.e., the additively weighted Voronoi diagram (see Figure 2) [AMG98b, AMG98a] and the Voronoi diagram for circles (set of sites comprising circles, see Figure 3) [KKS01b, KKS01a, KKS00]. The definition of the weighted Voronoi diagram differs from the definition of the ordinary one in that the Euclidean distance is replaced by a weighted distance. In the case of the additively weighted Voronoi diagram, the weighted distance between a point and a generator is the Euclidean distance minus the weight of the generator, but since it must be a distance, it has to be always positive or zero, and thus the additively weighted distance is not defined in the interior of the weight circles (circles centred on a generator and of radius the weight of the generator). The additively weighted Voronoi diagram has been extensively studied by Ash and Bolker [AB86] and Aurenhammer [Auren88] under the name of hyperbolic Dirichlet tessellations and Power Voronoi diagrams, but till [AMG98b] and [AMG98a], there was no dynamic algorithm for constructing the additively weighted Voronoi diagram. This work solves the robustness issue in the work of Anton, Mioc and Gold [AMG98b, AMG98a] and extends it to the Voronoi diagram of circles. This robustness fix and extension are achieved by providing an exact conflict locator.

The exact computation of the additively weighted Voronoi diagram has not been addressed until Anton et al. [ABMY02]. That paper addressed the exact predicate for the off-line construction of the dual graph of the additively weighted Voronoi diagram from the dual of the Power Voronoi diagram of spheres by using the relationship between the additively weighted Voronoi diagram in the plane and the Power Voronoi diagram[1] of spheres in the three-dimensional space. In their independent work, Karavelas and Emiris [KE02, EK06, KE03] provided several exact predicates of maximum degree 16 for achieving the same "in-circle/orientation/edge-conflict-type/difference of radii" test as we do in a single

[1] The Power Voronoi diagram is a generalised Voronoi diagram where sites are hyperspheres and the distance between a point and a site is the power of that point with respect to that site [Auren87].

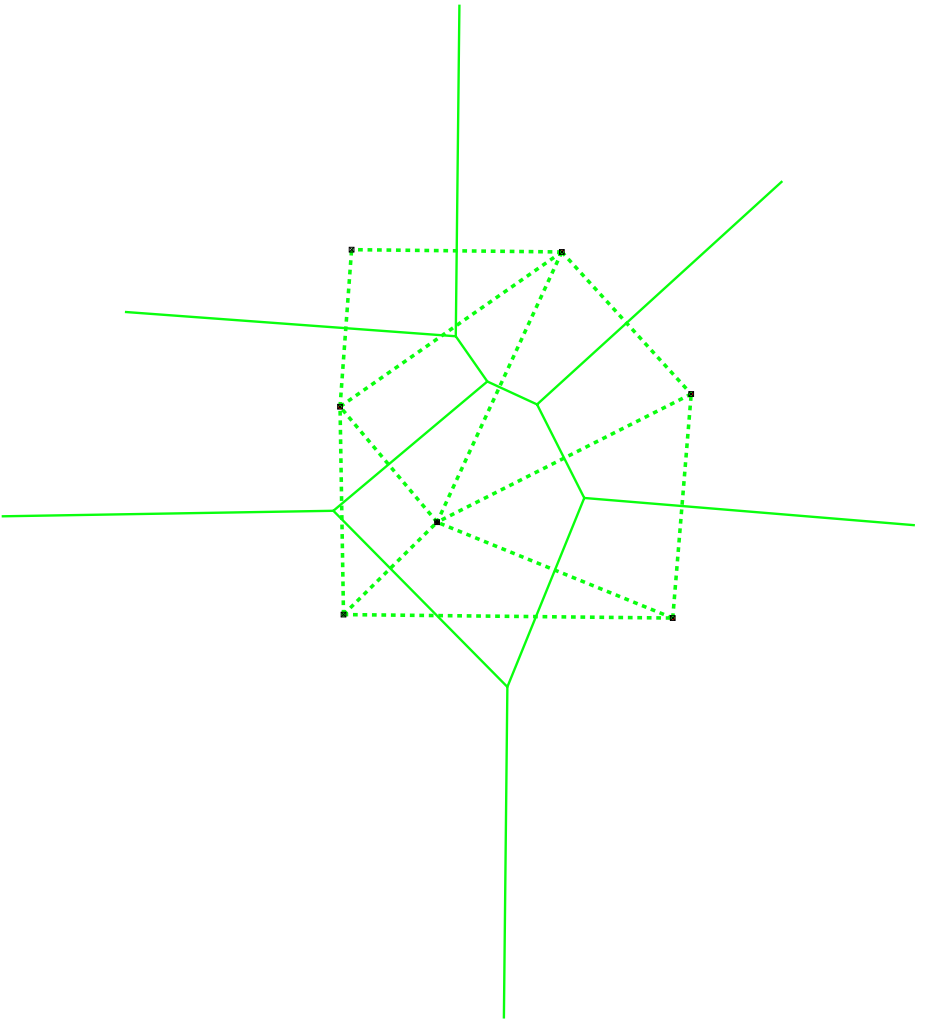


Fig. 1. The ordinary Voronoi diagram (plain lines) of points (squares), and its topology expressed by the Delaunay triangulation (dashed lines)

conflict locator presented in this paper. They reduced the degree of their predicate from 28 to 20 and then to 16 using Sturm sequences and invariants. Their work is more limited in scope than ours, because they compute the additively weighted Voronoi diagram (or Appolonius diagram) rather than the Voronoi diagram of circles, and they assume the circles never intersect (they mention this assumption could be lifted, but they provide no justification), and they also assume no three circles can have a common tangent, or equivalently, no empty circle has infinite radius. The difference between the additively weighted Voronoi diagram (or Appolonius diagram) and the Voronoi diagram of circles is that the additively weighted Voronoi diagram (or Appolonius diagram) is based

on a distance that is not defined in the interior of the (weight) circles, while the Voronoi diagram of circles is based on a distance that is defined everywhere. Thus, there can not be a point of the additively weighted Voronoi diagram in the interior of a circle, because its distance to the enclosing circle is not defined. Thus intersecting circles are not permitted and circles contained in other circles are not permitted either. Indeed, a point of the Voronoi diagram in the interior of the enclosing circle would not have a defined distance to the enclosing circle. The approach adopted in [KE02, EK06, KE03] is also more complex than ours, because they compute exactly not only the Delaunay graph, but also the additively weighted Voronoi diagram, which unlike they state, is not required in the applications. Only the exact computation of the Delaunay graph of circles is required for practical applications, because the Delaunay graph gives the topology of circles. Finally, our approach is much simpler, because we obtain the output of the predicate (in fact a Delaunay graph conflict loctor) by computing the sign of the eigenvalues of a simple two by two matrix.

In this paper, we also provide an application of the Voronoi diagram of circles to the visualisation of the growth of particle aggregates, which justifies the motivation for not only computing the additively weighted Voronoi (or Appolonius) diagram, but also the Voronoi diagram of circles. A comprehensive overview of the Delaunay and Voronoi methods for non-crystalline structures was provided by Medvedev [Med00], and Anishchik and Medvedev [AM95] were the first ones in 1995 to provide the solution of Appolonius problem for sphere packing in three dimensions. The application of the additively weighted Voronoi diagram to visualization of the growth of particle aggregates is based on particle statistics. Particle statistics play an important role in many technical processes (in the industrial production of materials where the phase transition from liquid to solid is a part of the technical process, for example production of metals and ceramic materials) [Stoya98], material science, plant ecology, and spatial analysis. Due to the lack of efficient algorithms for their visualization only the "set-theoretic approach in particle statistics" [Stoya98] has been used as a method of visualization of spatial growth processes in the past.

Growth models produce spatial patterns as a result of simple growth processes operating with respect to a set of n points (nucleation sites), $\mathcal{P} = \{p_1, p_2, ...p_n\}$ at positions $x_1, x_2, ..., x_n$, respectively in $\mathbb{R}^m$ or a bounded region of $\mathbb{R}^m$ $(m = 2, 3)$. The growth processes such as agglomeration, aggregation, packing, etc. lead in a natural way to the Poisson Voronoi tessellation [OBSC01], [Stoya98] and to the Johnson-Mehl tessellation when the members of the generator set $\mathcal{P}$ are not contemporaneous [OBSC01].

The Johnson-Mehl model has been introduced in [JM39] for modelling the growth of particle aggregates. The Johnson-Mehl model is a Poisson Voronoi growth model, in which nuclei are generated asynchronously using a Poisson point process [OBSC01], and grow at the same radial speed v. Each generator $P_i = (\overrightarrow{p_i}, t_i)$ has both a planar location (its position vector) and an associated birth time t_i $(t_i \geq 0)$. The Johnson-Mehl tessellation can be considered as a generalisation of a dynamic version of an additively weighted Voronoi diagram

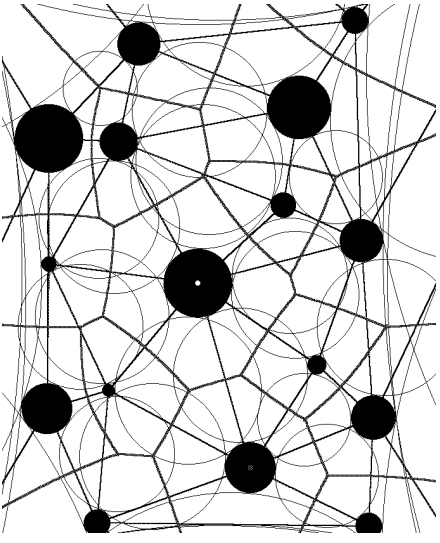


Fig. 2. An additively weighted Voronoi diagram, its dual graph and the empty circumcircles

[AMG98a], in which the weight reflects the arrival time of the point in $\mathbb{R}^2$ [OBSC01]. However, since when nuclei start to touch as they grow, the Johnson-Mehl tessellation might have intersecting nuclei, and edges that correspong to loci of centres of circles internally tangent to two weight circles. In that case, the additively weighted distance would not be well defined, because it would be negative. In that case, the distance used is the distance corresponding to the Voronoi diagram of circles. Thus, the Johnson-Mehl tessellation differs from the Voronoi diagram of circles in that only bisectors that are loci of circles that are either externally tangent to 2 circles or internally tangent to 2 circles are boundaries of Johnson-Mehl cells.

This paper is organised as follows. In Section 2, we present the definitions of the (generalised) Voronoi diagram of a set of sites and its dual Delaunay graph of a set of sites, and the Delaunay graph conflict locator. In Section 3, we provide necessary and sufficient conditions for construction of the Delaunay graph of circles and for connectivity of the Voronoi diagram of circles. In Section 4, we present the Delaunay graph conflict locator, both in the case of the additively weighted Voronoi diagram, and the Voronoi diagram of circles. In Section 5, we present the application of the Voronoi diagram to the modelling and the visualisation of the growth of particle aggregates. Finally, we present the conclusions and future work in Section 6.

2 Preliminaries

Voronoi diagrams are irregular tessellations of the space, where space is continuous and structured by discrete objects [AK00, OBSC01]. The Voronoi diagram

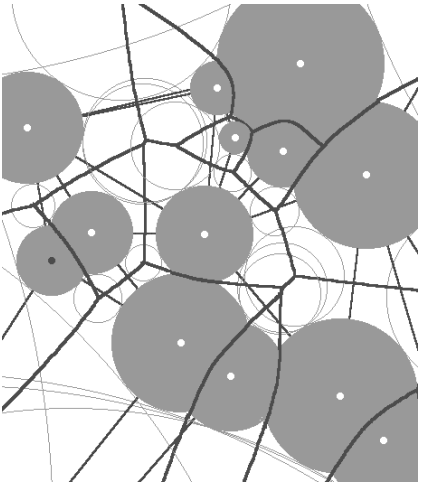


Fig. 3. The Voronoi diagram, the Delaunay graph and the empty circumcircles of circles. The circles hide the edges of the Delaunay graph between intersecting circles and the empty circles corresponding to intersecting circles.

[Vor07, Vor08, Vor10] (see Figure 1) of a set of sites is a decomposition of the space into proximal regions (one for each site). Sites were points for the first historical Voronoi diagrams [Vor07, Vor08, Vor10], but in this paper we will explore sets of circles. The proximal region corresponding to one site (i.e. its Voronoi region) is the set of points of the space that are closer to that site than to any other site of the set of sites [OBSC01]. We will recall now the formal definitions of the Voronoi diagram and of the Delaunay graph. For this purpose, we need to recall some basic definitions.

Definition 1. (Metric) Let M be an arbitrary set. A *metric* on M is a mapping $d : M \times M \rightarrow \mathbb{R}_+$ such that for any elements a, b, and c of M, the following conditions are fulfilled: $d(a,b) = 0 \Leftrightarrow a = b$, $d(a,b) = d(b,a)$, and $d(a,c) \leq d(a,b) + d(b,c)$. (M,d) is then called a *metric space*, and $d(a,b)$ is the distance between a and b.

Remark 2. The Euclidean space $\mathbb{R}^N$ with the Euclidean distance δ is a metric space $\left(\mathbb{R}^N, \delta\right)$.

Let $M = \mathbb{R}^N$, and δ denote a distance between points. Let $\mathcal{S} = \{s_1, ..., s_m\} \subset M, m \geq 2$ be a set of m different subsets of M, which we call *sites*. The distance between a point x and a site $s_i \subset M$ is defined as $d(x, s_i) = \inf_{y \in s_i} \{\delta(x,y)\}$.

Definition 3. (Bisector) For $s_i, s_j \in \mathcal{S}, s_i \neq s_j$, the *bisector* $B(s_i, s_j)$ of s_i with respect to s_j is: $B(s_i, s_j) = \{x \in M | d(x, s_i) = d(x, s_j)\}$ (see Figure 4).

Definition 4. (Influence zone) For $s_i, s_j \in \mathcal{S}, s_i \neq s_j$, the *influence zone* $D(s_i, s_j)$ of s_i with respect to s_j is: $D(s_i, s_j) = \{x \in M | d(x, s_i) < d(x, s_j)\}$ (see Figure 5).

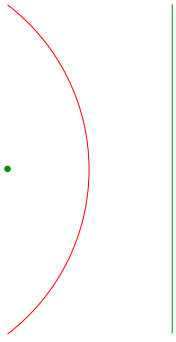


Fig. 4. The bisector (parabola) of a point and a line segment

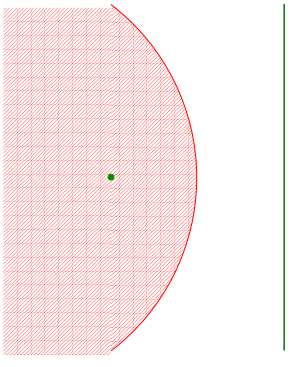


Fig. 5. The influence zone (hashed) of a point with respect to a line

Definition 5. (Voronoi region) The *Voronoi region* $V(s_i, \mathcal{S})$ of $s_i \in \mathcal{S}$ with respect to the set $\mathcal{S}$ is: $V(s_i, \mathcal{S}) = \bigcap_{s_j \in \mathcal{S}, s_j \neq s_i} D(s_i, s_j)$.

Definition 6. (Voronoi diagram) The *Voronoi diagram* of $\mathcal{S}$ is the union $V(\mathcal{S}) = \bigcup_{s_i \in \mathcal{S}} \partial V(s_i, \mathcal{S})$ of all region boundaries (see example on Figure 3).

Definition 7. (Delaunay graph) The *Delaunay graph* $DG(\mathcal{S})$ of $\mathcal{S}$ is the dual graph of $V(\mathcal{S})$ defined as follows:

- the set of vertices of $DG(\mathcal{S})$ is $\mathcal{S}$,
- for each $(N-1)-$dimensional facet of $V(\mathcal{S})$ that belongs to the common boundary of $V(s_i, \mathcal{S})$ and of $V(s_j, \mathcal{S})$ with $s_i, s_j \in \mathcal{S}$ and $s_i \neq s_j$, there is an edge of $DG(\mathcal{S})$ between s_i and s_j and reciprocally, and
- for each vertex of $V(\mathcal{S})$ that belongs to the common boundary of $V(s_{i_1}, \mathcal{S})$,..., $V\left(s_{i_{N+2}}, \mathcal{S}\right)$, with $\forall k \in \{1, ..., N+2\}, s_{i_k} \in \mathcal{S}$ all distinct, there exists a complete graph K_{N+2} between the s_{i_k}, and reciprocally (see example on Figure 3).

The one-dimensional elements of the Voronoi diagram are called Voronoi edges. The points of intersection of the Voronoi edges are called Voronoi vertices. The Voronoi vertices are points that have at least $N + 1$ nearest neighbours among the sites of $\mathcal{S}$. In the plane, the Voronoi diagram forms a network of vertices and edges. In the plane, when sites are points in general position, the Delaunay graph is a triangulation known as the Delaunay triangulation. In the plane, the Delaunay graph satisfies the following empty circle criterion: no site intersects the interior of the circles touching (tangent to without intersecting the interior of) the sites that are the vertices of any triangle of the Delaunay graph.

Once the Voronoi region a query point belongs to has been identified, it is easy to answer proximity queries. The closest site from the query point is the site whose Voronoi region is the Voronoi region that has been identified. The Voronoi diagram defines a neighbourhood relationship among sites: two sites are neighbours if, and only if, their Voronoi regions are adjacent, or alternatively, there exists an edge between them in the Delaunay graph.

The exact computation of the Delaunay graph is important for two reasons. By exact computation, we mean a computation whose output is correct. First, unlike the Voronoi diagram, the Delaunay graph is a discrete structure, and thus it does not lend itself to approximations. Second, the inaccurate computation of this Delaunay graph can induce inconsistencies within this graph (see Section 4.2), which may cause a program that updates this graph to crash. This is particularly true for the randomised incremental algorithm for the construction of the Voronoi diagram of circles. In order to maintain the Delaunay graph after each addition of a site, we need to detect the Delaunay triangles that are not empty any longer, and we need to detect which new triangles formed with the new site are empty, and thus valid. In the reminder, sites are generators of the Voronoi diagram or the Delaunay graph, while points are any location in the plane unless specified otherwise. The algorithm that certifies whether the triangle of the Delaunay graph whose vertices are 3 given sites is empty (i.e. does not contain any point of a given site in its interior) or not empty is used for checking which old triangles are not empty any longer and which new triangles formed with the new site are empty, and thus valid. This algorithm is called the "*Delaunay graph conflict locator*" in the reminder of this paper.

When the old triangles are checked, its input is a 4-tuple of sites, where the first three sites define an old triangle, and the fourth site is the new site being inserted. When the new triangles are checked, its input is also a 4-tuple of sites, where the first three sites define a new triangle, the first two sites being linked by an existing Delaunay edge, and the fourth site forms an old Delaunay triangle with the first two sites. Its output is the list of all the Voronoi vertices corresponding to the $1-$dimensional facets of the Delaunay graph having the first 3 sites as vertices whose circumcircles contain a point of the fourth site in their interior, and a value that certifies the presence of each Voronoi vertex in that list. The fact that a circumcircle (the circle that is externally tangent to three given circles) is not empty is equivalent to the triangle formed by those three circles being not Delaunay, and this is called a conflict. Thus, it justifies

the name of "Delaunay graph conflict locator". In the context of the ordinary Voronoi diagram of points in the plane, the concept that is analogous to the Delaunay graph conflict locator is the *Delaunay graph predicate*, which certifies whether a triangle of the Delaunay triangulation is such that its circumcircle does not contain a given point.

The exact knowledge of the Delaunay graph for curved objects may sound like a purely theoretical knowledge that is not central in practical applications. This is not always the case in some applications. These applications include material science, metallography, spatial analyses and VLSI layout. The Johnson-Mehl tessellations (which generalise several weighted Voronoi diagrams) [OBSC01] play a central role in the Kolmogorov-Johnson-Mehl-Avrami [JM39, Kol37] nucleation and growth kinetics theory. The Kolmogorov theory provides an exact description of the kinetics during the heating and cooling processes in material science (the Kolmogorov equation [JM39, Kol37]). The exact knowledge of the neighbourliness among molecules is central to the prediction of the formation of particle aggregates. In metallography, the analysis of precipitate sizes in aluminium alloys through Transmission Electronic Microscopy [Des03, Section 1.2.2] provides an exact measurement of the cross sections of these precipitates when they are "rodes" with a fixed number of orientations [Des03, Section 1.2.2]. In VLSI design, the second order Voronoi diagram of the layout is used in the computation of the critical area, a measure of a circuit layout's sensitivity to spot defects [CPX02, Section 1]. An important concern on critical area computation is robustness [CPX02, Section 1].

Another limitation of approximative algorithms for the computation of the Delaunay graph is that when approximate computations are performed on objects defined approximately (within some geometric tolerance), the propagation of the errors can be critical, especially if the final computation involves approximate intermediary computations.

Finally, the exact computation of the Delaunay graph participates to the recent move in the development of numerical and simulation software as well as computer algebra systems to exact systems [BCSS98].

3 The Necessary and Sufficient Conditions of Construction of the Delaunay Graph of Circles and of Connectivity of the Voronoi Diagram of Circles

In this section, we will examine how the Delaunay graph conflict locator can be used to maintain the Voronoi diagram of circles in the plane as those circles are introduced one by one. Finally, we will give a necessary and sufficient condition for the connectivity of the Voronoi diagram of circles in the projective plane that has a direct application in the representation of spatial data at different resolutions.

Knowing the Voronoi diagram $V(\mathcal{S})$ of a set $\mathcal{S}=\{s_1,\ldots,s_m\} \subset \mathbb{R}^2$ of at least two circles ($m > 1$) and its embedded Delaunay graph $DG(\mathcal{S})$ stored in a quad-edge data structure, we would like to get the Voronoi diagram $V(\mathcal{S} \cup \{s_{m+1}\})$,

where s_{m+1} is a circle of $\mathbb{R}^2$. In all this section, we will say that a circle $\mathcal{C}$ *touches* a circle s_i if, and only if, $\mathcal{C}$ is tangent to s_i and no point of s_i is contained in the interior of $\mathcal{C}$.

The Voronoi edges and vertices of $V(\mathcal{S})$ may or may not be present in $V(\mathcal{S} \cup \{s_{m+1}\})$. Each new Voronoi vertex w induced by the addition of s_{m+1} necessarily belongs to two Voronoi edges of $V(\mathcal{S})$, because two of the three closest sites to w necessarily belong to $\mathcal{S}$. The new Voronoi edges induced by the addition of s_{m+1} will clearly connect Voronoi vertices of $V(\mathcal{S})$ to new Voronoi vertices induced by the addition of s_{m+1} or new Voronoi vertices between themselves.

Any of these later Voronoi edges e' must be incident to one of the former Voronoi edges at each extremity of e' (because the Voronoi vertex at each extremity of e' belongs to only one new Voronoi edge, i.e. e'). Any of the former Voronoi edges e must be a subset of a Voronoi edge of $V(\mathcal{S})$, since e must be a new Voronoi edge between sites of $\mathcal{S}$ (otherwise the Voronoi vertex belonging to $V(\mathcal{S})$ at one of the extremities of e by the definition of e would be a new Voronoi vertex). Thus, to get $V(\mathcal{S} \cup \{s_{m+1}\})$, we need to know which Voronoi vertices and edges of $V(\mathcal{S})$ will not be present in $V(\mathcal{S} \cup \{s_{m+1}\})$, which Voronoi edges of $V(\mathcal{S})$ will be shortened in $V(\mathcal{S} \cup \{s_{m+1}\})$ and which new Voronoi edges will connect new Voronoi vertices between themselves.

We can test whether each Voronoi vertex v of $V(\mathcal{S})$ will be present in $V(\mathcal{S} \cup \{s_{m+1}\})$. Let us suppose that v is a Voronoi vertex of s_i, s_j and s_k. v will remain in $V(\mathcal{S} \cup \{s_{m+1}\})$ if, and only if, no point of s_{m+1} is contained in the interior of the circle centred on v that touches s_i, s_j and s_k. This is a sub-problem of the Delaunay graph conflict locator that can be tested by giving s_i, s_j, s_k and s_{m+1} as input to the Delaunay graph conflict locator, and then retain only the solutions where the Voronoi vertex is v.

We can test whether each Voronoi edge e of $V(\mathcal{S})$ will be present in $V(\mathcal{S} \cup \{s_{m+1}\})$. Let us suppose that e is a locus of points having s_i and s_j as closest sites. e will disappear entirely from $V(\mathcal{S} \cup \{s_{m+1}\})$ if, and only if, a point of s_{m+1} is contained in the interior of each circle centred on e and touching s_i, s_j and each common neighbour s_k to s_i and s_j in $DG(\mathcal{S})$ in turn. This can be tested by giving s_i, s_j, s_k and s_{m+1} as input to the Delaunay graph conflict locator and then retaining only the solutions where the Voronoi vertex belongs to e. e will be shortened (possibly inducing one or more new Voronoi edges) in $V(\mathcal{S} \cup \{s_{m+1}\})$ if, and only if, there exists Voronoi vertices of s_i, s_j and s_{m+1} on e and there is no point of any common neighbour s_k to s_i and s_j in $DG(\mathcal{S})$ in the interior of a circle centred on e and touching s_i, s_j and s_{m+1}. The centre of each one of such circles will be a new Voronoi vertex in $V(\mathcal{S} \cup \{s_{m+1}\})$. This can be tested by giving s_i, s_j, s_{m+1} and s_k as input to the Delaunay graph conflict locator and then retaining only the solutions where the Voronoi vertex belongs to e.

The Delaunay graph conflict locator is sufficient to maintain the Voronoi diagram of circles. Tests might be limited to edges and vertices on the boundaries of the Voronoi regions $V(s_i, \mathcal{S}), s_i \in \mathcal{S}$ that intersect s_{m+1} and of the Voronoi

regions $V(s_j, \mathcal{S}), s_j \in \mathcal{S}$ adjacent to a Voronoi region $V(s_i, \mathcal{S})$. Indeed, a point (and thus a circle) can steal its Voronoi region only from the Voronoi region it belongs to and the adjacent Voronoi regions.

We will finish this section with a necessary and sufficient condition for the connectivity of the Voronoi diagram of connected circles in the projective plane. This result allows the characterisation of dangling edges in the Delaunay graph corresponding to the presence of closed edges in the Voronoi diagram. In order to proceed, let us recall some notations used in point set topology: let $\overline{s}$ denote the closure of s, and $\overset{\circ}{s}$ denote the interior of s in the sense of the point set topology in $\mathbb{R}^2$. Note that if s bounds a closed domain then the interior of s is meant to be the interior of the closed domain bounded by s.

Proposition 8. *(Connectivity of the Voronoi diagram in the plane) The Voronoi diagram $V(\mathcal{S})$ of a set $\mathcal{S} = \{s_1, \ldots, s_m\} \subset \mathbb{R}^2$ of at least two connected circles ($m > 1$) considered in $\mathbb{P}^2$ is not connected if, and only if, there exist a subset I of $[1, \ldots, m]$ and one index j of $[1, \ldots, m]$ such that $\forall i \in I, s_i \subset \overset{\circ}{s_j}$ and $\forall k \in [1, \ldots, m] \setminus I, \overline{s_i} \cap \overline{s_k} = \overline{s_j} \cap \overline{s_k} = \emptyset$.*

Proof. If: Assume there exist a subset I of $[1, \ldots, m]$ and one index j of $[1, \ldots, m]$ such that $\forall i \in I, s_i \subset \overset{\circ}{s_j}$ and $\forall k \in [1, \ldots, m] \setminus I, \overline{s_i} \cap \overline{s_k} = \overline{s_j} \cap \overline{s_k} = \emptyset$. Let $s_l \in \mathcal{S}$ with $l \in [1, \ldots, m] \setminus I$. Let $S = \bigcup_{i \in I} s_i$. Since $S \subset \overset{\circ}{s_j}$, any circle touching both a $s_i, i \in I$ and s_j must be contained in $\overline{s_j}$. Since $\overline{S} \cap \overline{s_l} = \overline{s_j} \cap \overline{s_l} = \emptyset$, no circle can touch each of an $s_i, i \in I$, s_j and s_l. Thus, there is no point that has a $s_i, i \in I$, s_j and s_l as nearest neighbours. Thus, there is no Voronoi vertex of a $s_i, i \in I$, s_j and s_l. Since there is no Voronoi vertex of a $s_i, i \in I$, s_j and an s_l with $l \in [1, \ldots, m] \setminus I$, there are no Voronoi vertices on the bisector of S and s_j. Since $\overline{S} \cap \overline{s_l} = \overline{S} \cap \overline{s_l} = \emptyset$, any circle centred on the bisector of S and s_j and touching both S and s_j does not intersect any site s_k with $k \in [1, \ldots, m] \setminus I$. Thus, the bisector of S and s_j is contained in $V(\mathcal{S})$. Since s_j is connected and $S \subset \overset{\circ}{s_j}$, the bisector of S and s_j is a closed curve. Thus, the Voronoi diagram of $\mathcal{S}$ is not connected in $\mathbb{P}^2$.

Only if: Assume the Voronoi diagram of $\mathcal{S}$ is not connected in $\mathbb{P}^2$. Then, $V(\mathcal{S})$ has at least two connected components. Thus, at least one of these connected components does not have points at infinity. Let us consider the connected component (let us call it C_1) that does not have points at infinity. Since C_1 is composed of Voronoi edges[2], each edge in C_1 must end at either a Voronoi vertex or a point at infinity. Since C_1 does not have any point at infinity, all Voronoi edges in C_1 connect Voronoi vertices. Thus C_1 is a network of vertices and edges linking those vertices. The regions that this network defines are Voronoi regions. Let $\mathcal{D}$ be the union of the closure of those Voronoi regions. $\mathcal{D}$ is a closed set by its definition. Let us consider now the circles $s_l, l \in L$ whose Voronoi regions are contained in $\mathcal{D}$. Let $S = \bigcup_{l \in L} s_l$. Thus S is a union of circles.

[2] A one-dimensional component of the Voronoi diagram, which is also the locus of points having two nearest sites.

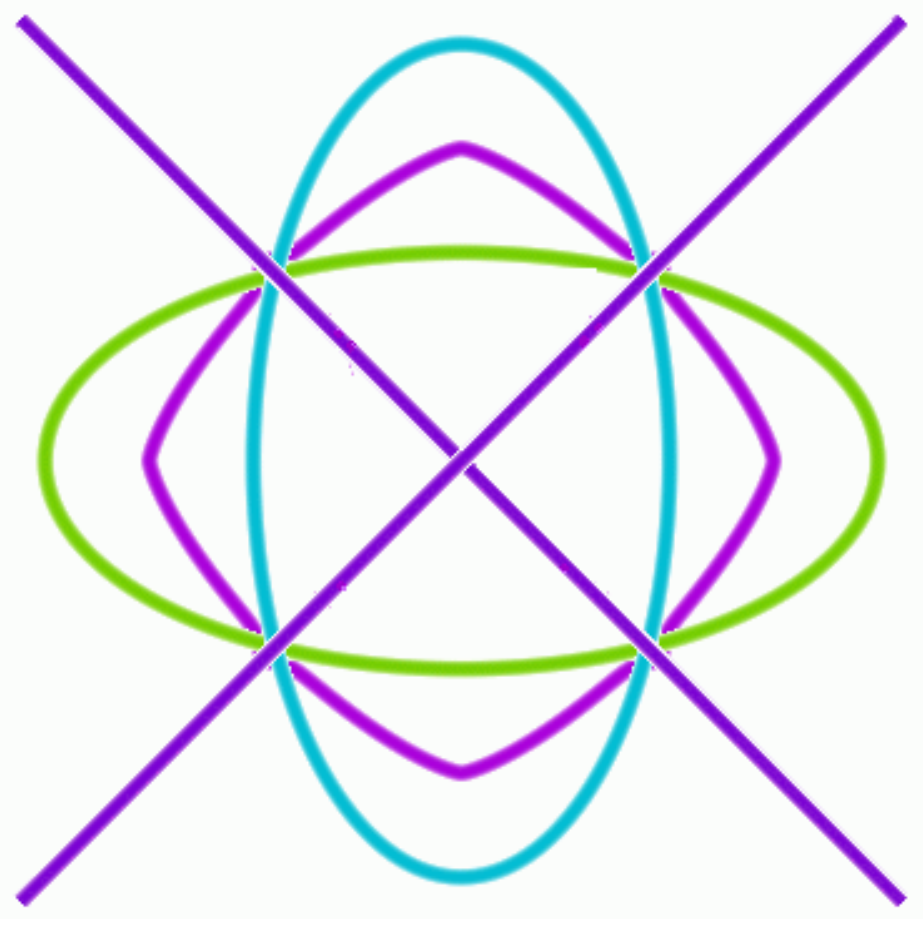


Fig. 6. The relative position with respect to the bisector must be constant

We will now consider S as a site instead of each one of the $s_l, l \in L$. The influence zone of $S = \bigcup_{l \in L} s_l$ is clearly $\mathring{\mathcal{D}}$, because the influence zone of a union of circles is clearly the closure of the union of the Voronoi regions of those circles. Let $e = \partial\mathcal{D}$. It is a portion of the bisector of S and another circle. Let us call it s_j. If not all the bisector of S and s_j was contained in $V(\mathcal{S})$, then e would end at Voronoi vertices (a point on the Voronoi diagram has at least two closest sites) or the point at infinity, a contradiction with e not being connected. Thus, the bisector of S and of s_j is contained in $V(\mathcal{S})$, and it is equal to e. By the definition of e, e must be a closed curve. Assume the positions of S and s_j with respect to e are not always the same. Then, S and s_j must intersect. The bisector of S and s_j must have two branches near the intersection points (see Figure 6). Since e is a closed curve and S is contained in the interior of e, s_j must be closed, and the other branches must be unbounded (a contradiction with e not being connected in $\mathbb{P}^2$). Thus, the positions of S and s_j with respect to e are always

the same along e. Since s_j is connected, S is contained in the interior of e and the positions of S and s_j with respect to e are always the same along e, $S \subset \overset{\circ}{s_j}$. Since e is the bisector of S and s_j and belongs to $V(\mathcal{S})$, any circle centred on e and touching both S and s_j does not intersect any site s_k with $k \in [1, \ldots, m] \setminus I$. Thus, $\forall k \in [1, \ldots, m] \setminus I, \overline{s_i} \cap \overline{s_k} = \overline{s_j} \cap \overline{s_k} = \emptyset$.

□

The only cases of disconnected (considered in $\mathbb{P}^2$) Voronoi diagrams correspond to one or more sites (circles) contained in the interior of another site. This property has a direct application in Geographic Information Systems. When the same region $\mathcal{R}$ bounded by a circle S is represented at different scales, the representation of the details inside $\mathcal{R}$ does not change the Voronoi diagram outside $\mathcal{R}$. The edges of the Delaunay graph corresponding to a disconnected Voronoi diagram (considered in $\mathbb{P}^2$) are respectively dangling edges or cut edges (the Delaunay graph is not bi-connected and removing a cut edge induces two connected components). It is possible to detect if there exists one or more sites $s_i, i \in I$ contained in the interior of another site s_j by checking that there exists no Voronoi vertex of s_i, s_j and any $s_k \in \mathcal{S}$ distinct from s_i and s_j. This is again a subproblem of the Delaunay graph conflict locator.

4 The Exact Symbolic Delaunay Graph Conflict Locator for Circles

We will first present the exact symbolic Delaunay graph conflict locator for additively weighted points when weighted points are introduced one by one, and then introduce what changes for circles. For this purpose, we will present some preliminaries about additively weighted Voronoi diagrams.

4.1 Preliminaries

Let $\mathbb{N}$ be the set of integers, $\mathbb{R}$ be the set of real numbers, and $\mathbb{R}^2$ be the Euclidean plane. Let $\mathcal{P} = \{P_1, ..., P_N\}$ be the set of generators or sites, where P_i is the *weighted point* located at $p_i \in \mathbb{R}^2$ and of weight $w_i \in \mathbb{R}$. Let C_i be the circle centred at p_i and of radius w_i, which we call *weight circle* hereafter.

The definitions of bisector, influence zone, Voronoi region and Voronoi diagram presented in Section 2 generalise to the case where the set of sites $\mathcal{S}$ is a set of weighted points $\mathcal{P}$, and the distance $d(M, P_i)$ (called *additive distance*) between a point M and a site P_i is $d(M, P_i) = \delta(M, p_i) - w_i$, where δ is the Euclidean distance between points.

The Voronoi region of P_i with respect to the set $\mathcal{P}$ is defined by: $\mathcal{V}(P_i, \mathcal{P}) = \left\{M \in \mathbb{R}^2 | \forall j \neq i : \delta(M, p_i) - w_i < \delta(M, p_j) - w_j\right\}$.

The *Additively Weighted Voronoi diagram* of $\mathcal{P}$ is defined by:

$V(\mathcal{P}) = \bigcup_{P_i \in \mathcal{P}} \partial V(P_i, \mathcal{P})$. The additively weighted Voronoi diagram is illustrated in Figure 7: the weight circles are drawn as plain disks with small holes at their centres, the additively weighted Voronoi diagram is drawn in plain thick hyperbola segments, and the Delaunay graph is drawn in dashed lines.

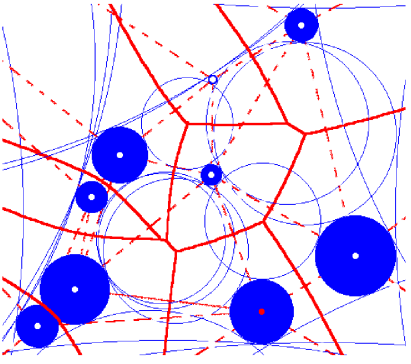


Fig. 7. The additively weighted Voronoi diagram

The additively weighted Voronoi diagram defines a network composed of edges (loci of points having two nearest neighbours), and vertices (loci of points having three nearest neighbours).

The additively weighted Voronoi diagram is related to the Apollonius Tenth problem. The Apollonius Tenth problem is to find a circle Γ tangent to three given circles C_1, C_2, C_3 (see Figure 8). For additively weighted points, we will see later in this section that only the circles that are either externally tangent to each of three given circles C_1, C_2, C_3 or internally tangent to each of C_1, C_2, C_3, are relevant to the Delaunay graph conflict locator. The centres of the circles that are solutions to the Apollonius Tenth problem are the first example encountered in this paper of generalised Voronoi vertices (a concept that we introduced in [Anton04]). Informally, generalised Voronoi vertices are the centres of circles tangent to $N + 1$ sites, where N is the dimension of the Euclidean space.

Hereafter we will call the solutions of the Apollonius Tenth problem *Apollonius circles*. The centres of the Apollonius circles that are either externally tangent to each of three given circles C_1, C_2, C_3 or internally tangent to each of C_1, C_2, C_3 are the first example encountered in this paper of true Voronoi vertices (i.e. centres of circles that touch $N + 1$ sites where N is the dimension of the Euclidean space).

4.2 The Delaunay Graph Conflict Locator for Additively Weighted Points

In this subsection, we present an exact algebraic conflict locator for the Delaunay graph of additively weighted points (i.e. the dual graph of the additively weighted Voronoi diagram). The maximum degree of the polynomials which need to be evaluated to compute this Delaunay conflict locator is 16 (thus, we say that the degree of the conflict locator is 16). This Delaunay graph conflict locator would be the core of a randomised incremental algorithm for constructing the additively weighted Voronoi diagram since the additively weighted Voronoi diagram is an abstract Voronoi diagram [Kle89], and thus, it can be constructed with the randomised incremental algorithm of Klein [Kle89].

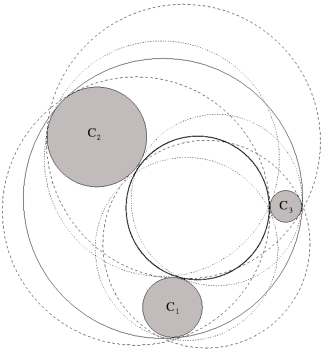


Fig. 8. The Apollonius Tenth problem

The motivation for an exact conflict locator lies in the fact that without an exact computation of the Delaunay graph of additively weighted points, some geometric and topologic inconsistencies may appear. This is illustrated with an example. The starting configuration is shown on Figure 9. There are three weighted points (whose corresponding weight circles are drawn). The Delaunay graph is drawn in dashed lines. The Apollonius circles tangent to the weight circles have been drawn in dotted lines. The real configuration after addition of a fourth weighted point is shown on Figure 10. The configuration that might have been computed by an approximate algorithm is shown on Figure 11: the difference between real and perceived situations has been exaggerated to show the difference. The old Apollonius circles have been adequately perceived to be invalid with respect to the newly inserted weighted point. About the new Voronoi vertices, while on the right of the figure two new Voronoi vertices have been identified as valid with respect to their potential neighbours, on the left of the figure, only one Voronoi vertex has been identified as being valid with respect to its potential neighbours. While the new Voronoi edge between the middle and bottom weighted points can be drawn between the two new Voronoi vertices of the new, middle and bottom weighted points; the Voronoi edge between the top and new weighted points cannot be drawn, because there is no valid Voronoi vertex on the left. There is an inconsistency within the topology: there is one new Voronoi vertex (the Voronoi vertex of the top new and middle weighted points) that cannot be linked by a new Voronoi edge to any other new Voronoi vertex and thus, that Voronoi vertex is incident to only two Voronoi edges. This additively weighted Voronoi diagram might have been computed by an approximative algorithm that is not an additively weighted Voronoi diagram. Thus, even if we perturbate the input weighted points, we will never get this additively weighted Voronoi diagram.

We consider the maintenance of the Delaunay graph of additively weighted points in an incremental way: we check the validity of all the triangles of the Delaunay graph whose vertices are P_1, P_2, P_3 with respect to a newly inserted weighted point P_4 [AKM02] or the validity of all the triangles of the Delaunay graph whose vertices are P_1, P_2, where the edge between P_1 and P_2 exists in

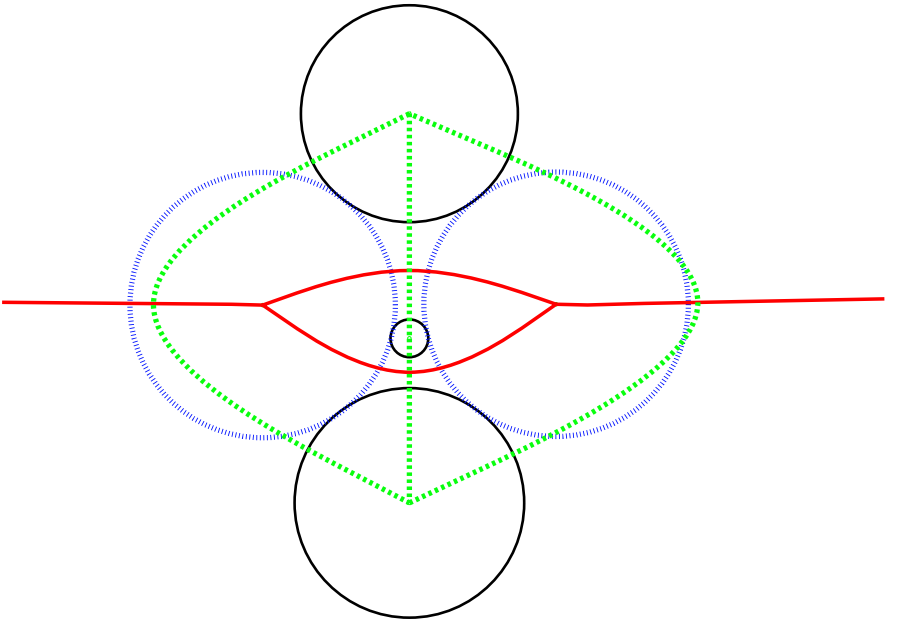


Fig. 9. The starting configuration

the Delaunay graph, and the newly inserted weight point P_3 with respect to an existing point P_4. Thus, the input of the conflict locator is constituted by four points: the first three are supposed to define a triangle in the Delaunay graph, and the last one is the tested point. Let (x_i, y_i) be the coordinates of p_i, for $i = 1, 2, 3, 4$. There are two possible outcomes to the above test of validity: either the triangles are valid with respect to the fourth weighted point and the triangles must appear in the Delaunay graph, or one or two triangles are not valid with respect to the fourth weighted point and those triangles will not be present in the Delaunay graph. We can see an example of the later case in Figure 12. A triangle having $P_1P_2P_3$ as vertices is not valid with respect to the weighted point P_4, because the circle externally tangent to both the weight circles C_1, C_2 and C_3 (of weighted points C_1, C_2 and C_3) contains a point of the weight circle C_4 (of the weighted point P_4). Thus, it must not appear in the Delaunay graph.

When the old triangles are checked, the conflict locator consists of determining which of the additively weighted Voronoi vertices of P_1, P_2 and P_3 will not remain after the insertion of P_4. When the new triangles are checked, the conflict locator consists of determining which new Voronoi vertices of weighted points P_1, P_2 and the newly inserted weighted point P_3 will appear, where P_1P_2 is an old Delaunay edge. When the new triangles are checked, this conflict locator tests the new triangle $P_1P_2P_4$ with respect to any point P_4 such that $P_1P_2P_4$ is an old Delaunay triangle. In both cases, the Delaunay graph conflict locator is equivalent in turn to the additive distance from which of the additively weighted Voronoi vertices of P_1, P_2 and P_3 to P_4 is smaller than the additive distance of that Voronoi vertex to P_1 (or P_2 or P_3 (see Figure 12).

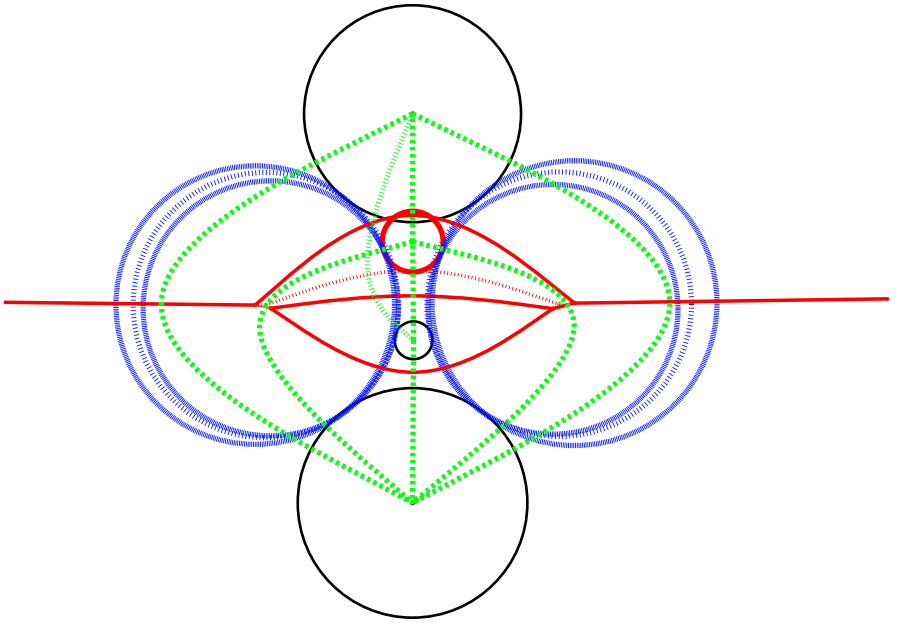


Fig. 10. The real configuration after addition of the fourth weighted point (bold weight circle)

Any *additively weighted Voronoi vertex* I of P_1, P_2, and P_3 with coordinates (x, y) can be obtained algebraically by computing the common intersection of the three circles C_1', C_2' and C_3' expanding (see Figure 13), or shrinking (see Figure 14) from the three first circles C_1, C_2 and C_3 all at the same rate. The common signed expansion of the first three circles is denoted by r. Each circle C'' centred on (x, y) and of radius r is either externally tangent to the first three circles (if the expansion r is positive) or internally tangent to the first three circles (if the expansion r is negative).

The centres coordinates x, y and radii r of the circles C'' centred on the intersections $I = C_1' \cap C_2' \cap C_3'$ and either externally or internally tangent to each of C_1, C_2, and C_3 can be computed algebraically as the solutions of the following system of three quadratic equations in the variables x, y and r:

$$\begin{cases} c_1'(x, y, r) = (x - x_1)^2 + (y - y_1)^2 - (w_1 + r)^2 = 0 \\ c_2'(x, y, r) = (x - x_2)^2 + (y - y_2)^2 - (w_2 + r)^2 = 0 \\ c_3'(x, y, r) = (x - x_3)^2 + (y - y_3)^2 - (w_3 + r)^2 = 0 \end{cases}$$

Subtracting one of the equations (say $c_1'(x, y, r) = 0$) from the remaining two ($c_2'(x, y, r) = 0$ and $c_3'(x, y, r) = 0$) results in a system of 2 linear equations, from which x and y may be expressed as linear functions of r. Substitution in the first equation $c_1'(x, y, r) = 0$ then leads to a quadratic equation in r. This means that the unknown quantities x, y, r can be expressed with quadratic radicals as functions of the given centres and radii.

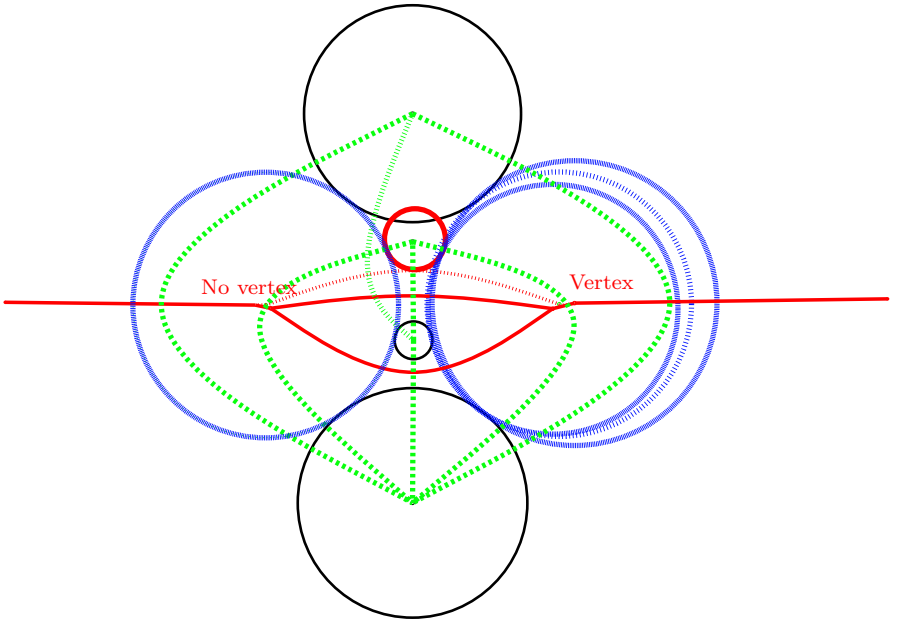


Fig. 11. The configuration computed by an approximate algorithm

Though the simplest thing to do now would be to compute the two Voronoi vertices and use their computed coordinates and corresponding signed expansion in the computation of the values certifying the output of the Delaunay graph conflict locator, it is not desirable because this method would not guarantee the topology of the Voronoi diagram of circles, nor its generalisation to conics or higher degree algebraic curves. We will detail hereafter only the computation of the values certifying the presence of Voronoi vertices in the output list.

To get the exact Delaunay graph conflict locator in a more elegant and generalisable way, we evaluated the values certifying the conflict locator output without relying on the computation of the Voronoi vertices as an intermediary computation. This is done by evaluating the values taken by the polynomial function expressing the relative position of C_4 with respect to C'' on the set of solutions of the system (i.e. the common zeroes of the three polynomials c'_1, c'_2 and c'_3). This is possible due to the translation that exists between geometry and algebra.

More specifically, to the geometric set X of the set of common zeroes of the three polynomials c'_1, c'_2 and c'_3 in K^3, where K is an algebraically closed field [Lan02, Definition before Theorem 1, Section 2, Chapter VII], we can associate the set of all polynomials vanishing on the points of X, i.e., the set of polynomials $f_1c'_1 + f_2c'_2 + f_3c'_3$ where the $f_i, i = 1, 2, 3$ are polynomials in the three variables x, y, r with coefficients in K. This set is the *ideal* [GP02, Definition 1.3.1] $\langle c'_1, c'_2, c'_3 \rangle$. The set of polynomials with coefficients in K, forms with the addition and the multiplication of polynomials, a ring: the *ring of polynomials* [GP02, Definition 1.1.3]. A polynomial function $g\,(x, y, r)$ on K^3 is mapped to

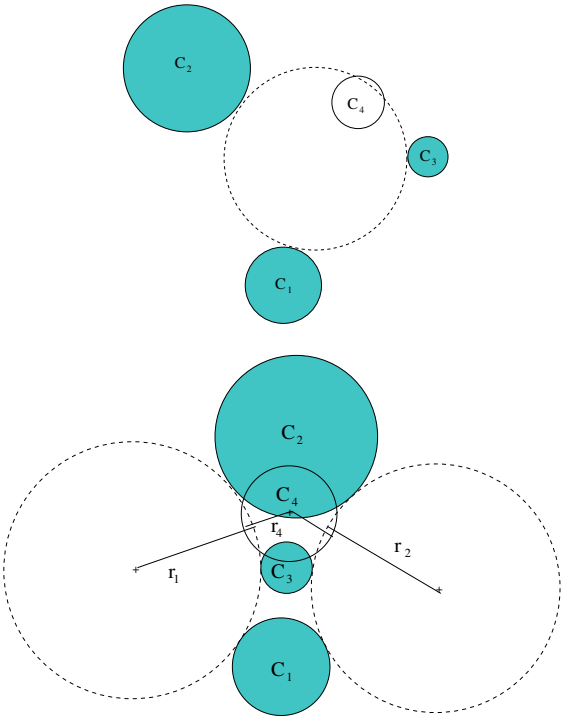


Fig. 12. The Delaunay graph conflict locator for the additively weighted Voronoi diagram: only the weight circles C_i or the weighted points P_i for $i = 1, ..., 4$ are shown. Up: there is only one Voronoi vertex to check; down: there are two Voronoi vertices to check.

a polynomial function on X if we recursively subtract from g any polynomial in g belonging to $\langle c'_1, c'_2, c'_3\rangle$ until no monomial in g can be divided by each one of the lexicographically highest monomials in c'_1, c'_2 and c'_3. The result of this mapping gives a canonic representative of the remainder of the Euclidean division of the polynomial g by the polynomials c'_1, c'_2 and c'_3. The image of the ring of polynomials by this mapping is called the *quotient algebra* [Lan02, Section 3, Chapter II] of the ring of polynomials by the ideal $\langle c'_1, c'_2, c'_3\rangle$. Moreover, $\langle c'_1, c'_2 - c'_1, c'_3 - c'_1\rangle$=$\langle c'_1, c'_2, c'_3\rangle$. Finally, if we recursively subtract from g any polynomial in g belonging to $\langle c'_1, c'_2 - c'_1, c'_3 - c'_1\rangle$ till the only monomials in g are 1 and r, we get the same result as the preceding mapping. The polynomials $c'_1, c'_2 - c'_1, c'_3 - c'_1$ constitute what is called a *Gröbner basis* [GP02, Definition 1.6.1] of the ideal $\langle c'_1, c'_2, c'_3\rangle$.

Gröbner bases are used in Computational Algebraic Geometry in order to compute a canonic representative of the remainder of the division of one polynomial by several polynomials generating a given ideal I. This canonic representative belongs to the quotient algebra of the ring of polynomials by the ideal I. The Gröbner basis for this system provides a set of polynomials that define uniquely the algebraic relationships between variables for the solutions of the system.

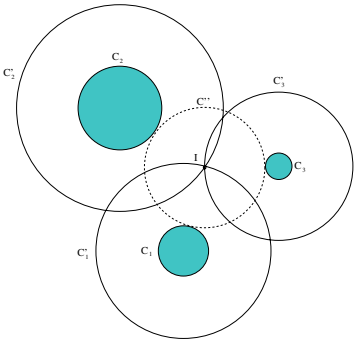


Fig. 13. The additively weighted Voronoi vertex as the common intersection of three expanding circles

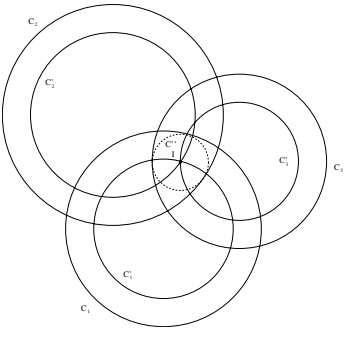


Fig. 14. The additively weighted Voronoi vertex as the common intersection of three shrinking circles

The initial (largest with respect to some monomial order [CLO98]) monomials of each one of the polynomials of the Gröbner basis form an ideal. The monomials that do not pertain to this ideal form a basis for the representatives of the equivalence class of the remainders of the division of a polynomial by the polynomials of the system in the quotient algebra. These monomials are called standard monomials. For the above Gröbner basis, the standard monomials are 1 and r. The size of this basis equals the *dimension* [GP02, see definition on page 414] of the quotient algebra and the number of solutions of the system counted with their multiplicity [Lan02]. In the case of the conflict locator for the additively weighted Voronoi diagram, there are two solutions.

The polynomial $g = (x_4 - x)^2 + (y_4 - y)^2 - (r + r_4)^2$ expresses the relative position of C_4 with respect to C''. Indeed C'' is tangent to C_4 if, and only if, the Euclidean distance between the centres of C'' and of C_4 (i.e., (x, y) and p_4) equals the sum of the radii r and r_4, i.e. $(x_4 - x)^2 + (y_4 - y)^2 - (r + r_4)^2 = 0$. The open balls bounded by C'' and C_4 intersect if, and only if, the Euclidean distance between the centres of C'' and of C_4 is smaller than the sum of the radii r and r_4, i.e. $(x_4 - x)^2 + (y_4 - y)^2 - (r + r_4)^2 < 0$. The circles C'' and C_4 are disjoint if,

and only if, the Euclidean distance between the centres of C'' and of C_4 is greater than the sum of the radii r and r_4, i.e. $(x_4 - x)^2 + (y_4 - y)^2 - (r + r_4)^2 > 0$. We considered the operation of multiplication of polynomials by the polynomial g. This *multiplication operator* is a linear mapping. The operation of this mapping on the canonic representative of the reminder of the division of a polynomial by c'_1, c'_2 and c'_3 is also a linear mapping that can be expressed by a matrix since the quotient algebra has a finite dimension.

First, we compute the matrix $M_g = \begin{pmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{pmatrix}$ of the following multiplication operator on the quotient algebra:
$m_g : [f] \longrightarrow [gf]$.

The eigenvalues of M_g are the values of g taken on X (see Theorem 4.5, page 54 in [CLO98]). The eigenvalues of M_g are the solutions of $det\,(M_g - \lambda I) = 0$, where I denotes the 2×2 identity matrix, i.e. the roots of

$$\lambda^2 - \lambda\,(m_{00} + m_{11}) + (m_{00}m_{11}) - (m_{01}m_{10}) = 0 \tag{4.1}$$

The values certifying the presence of Voronoi vertices in the list output by the Delaunay graph conflict locator are the signs of the values taken by g, and they are determined by the sign of the roots of Equation 4.1 (which are the eigenvalues of M_g). If there is only one eigenvalue and it is 0 then the fourth circle is tangent to the circle externally tangent to the first three circles. The sign of Δ (where $\Delta = (m_{00} + m_{11})^2 - 4\,(m_{00}m_{01} - m_{01}m_{10})$) cannot be negative when the first three sites of the input correspond to a Delaunay triangle, because this would be equivalent to the fact there would be no triangle with vertices C_1, C_2 and C_3 in the old Delaunay graph (because of the absence of real Voronoi vertex, see Figure 15). Thus, if $sign\,(\Delta)$ is negative that means we have one circle contained in another circle, and then we just need to link them by a Delaunay edge. Otherwise, $sign\,(\Delta)$ is 0 or positive, and we have to evaluate the sign of the roots of the quadratic equation.

When there is only one double root of Equation 4.1 then we have the following two possibilities. Either the value of the root of Equation 4.1 is positive or 0

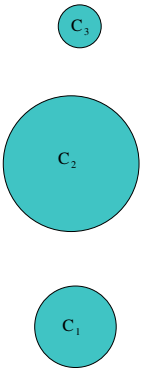


Fig. 15. There is no such triangle in the old Delaunay graph because of the absence of a real Voronoi vertex

and the triangle will exist in the new Delaunay graph, or the value of the root of Equation 4.1 is negative and the triangle will not exist in the new Delaunay graph (see Figure 12). When there are two real roots of Equation 4.1, we have two triangles to consider (see Figure 16). The triangles that correspond to the roots with a negative value will disappear in the new Delaunay graph (see Figure 16).

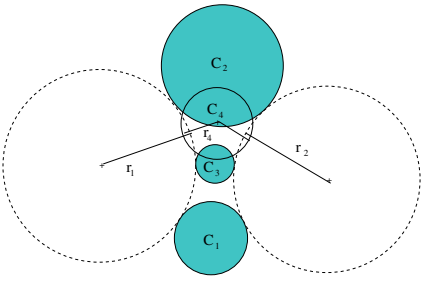


Fig. 16. Two triangles can possibly disappear simultaneously by the addition of a single weighted point

There is not much interest in showing the elements of the matrix of the multiplication operator here, but the Macaulay 2 [GS] code is presented in Appendix 1. The exact algebraic computation of the Delaunay graph conflict locator we have presented in the previous paragraph is not generalisable to the other proper conics or higher degree algebraic curves. Indeed, the size of the multiplication operator matrix is greater than 4 for the other proper conics and for higher degree algebraic curves, and an algebraic equation of degree 5 or more is not necessarily solvable by radicals (see [BB96, Theorem 8.4.8]). Even if we can obtain the matrix of the multiplication operator symbolically, we will need numerical methods for computing the eigenvalues of that matrix, which give the answer to the Delaunay graph conflict locator.

We will now present the Delaunay graph conflict locator for circles, emphasising the changes with respect to the Delaunay graph of additively weighted points presented in this subsection.

4.3 The Delaunay Graph Conflict Locator for Circles

Let $\mathcal{C} = \{C_1, ..., C_N\}$ be the set of generators or sites, with all the C_i being circles in $\mathbb{R}^2$. Let p_i be the centre of C_i and r_i be the radius of C_i.

The definitions of bisector, influence zone, Voronoi region and Voronoi diagram presented in Chapter 2 generalise to the case where the set of sites $\mathcal{S}$ is a set of circles $\mathcal{C}$, and the distance $d(M, C_i)$ between a point M and a site C_i is the Euclidean distance between M and the closest point on C_i from M, i.e. $d(M, C_i) = |\delta(M, p_i) - r_i|$, where δ is the Euclidean distance between points. Observe that assuming C_i is centred on p_i and $r_i = w_i$ for $i = 1, .., N$, this distance

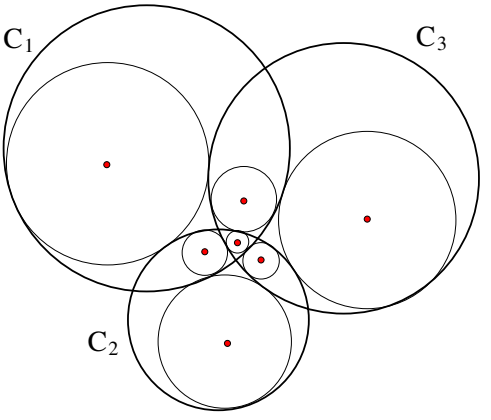


Fig. 17. Seven Apollonius circles centres that are true Voronoi vertices (first case)

is the absolute value of the additive distance used in the previous subsection. The Voronoi region of C_i with respect to the set $\mathcal{C}$ is thus defined by:

$$\mathcal{V}(C_i, \mathcal{C}) = \left\{ M \in \mathbb{R}^2 | \forall j \neq i : |\delta(M, p_i) - r_i| < |\delta(M, p_j) - r_j| \right\}.$$

The Voronoi diagram of $\mathcal{C}$ is defined by: $V(\mathcal{C}) = \bigcup_{C_i \in \mathcal{C}} \partial V(C_i, \mathcal{C})$.

In the previous subsection, we observed that two Apollonius circles centres are true Voronoi vertices of the additively weighted Voronoi diagram (the circles that are either externally or internally tangent to three given circles). When the sites are circles, up to seven of the eight Apollonius circles may be relevant to the Delaunay graph conflict locator (see Figure 17).

We consider the maintenance of the Delaunay graph of circles in an incremental way: we check first the validity of all the old triangles of the Delaunay graph whose vertices are a given triple of circles with respect to a given newly inserted circle. When old triangles are checked, four circles C_1, C_2, C_3 and C_4 are given: the first three are supposed to define one or more triangles in the Delaunay graph, and the last one is the newly inserted circle. Let (x_i, y_i) be the coordinates of p_i for $i = 1, 2, 3, 4$. There are two possible outcomes to the above test of validity. Either the triangles are valid with respect to the newly inserted weighted point and the triangles remain in the new Delaunay graph, or there is at least one triangle that is not valid with respect to the newly inserted weighted point and these triangles will not be present in the Delaunay graph any longer. We also need to check the validity of new triangles $C_1C_2C_3$ with respect to a circle C_4, where $C_1C_2C_4$ is an old Delaunay triangle and C_3 is the newly inserted circle. There are two possible outcomes to this test of validity. Either the triangles formed with an old Delaunay edge C_1C_2 and the newly inserted weighted point C_3 are valid with respect to any circle C_4, where $C_1C_2C_4$ is an old Delaunay triangle, and the triangles will appear in the new Delaunay graph, or there is at least one triangle that is not valid and these triangles will not be added in the Delaunay graph. In both cases, we check the validity of a triangle $C_1C_2C_3$ with respect to a circle C_4.

The Apollonius circles of C_1, C_2 and C_3 can be obtained algebraically by computing the common intersection of the three circles C_1', C_2' and C_3' (see Figure 13) expanding or shrinking from the three first circles C_1, C_2 and C_3 all with the same absolute value of the rate. The common unsigned expansion of the first three circles is denoted by r. The coordinates of the intersection I of C_1', C_2' and C_3' are denoted (x, y). The circle C'' centred on (x, y) and of radius r is tangent to the first three circles.

Thus, the Apollonius circles are the solutions of one of the eight following systems (I) of three quadratic equations in three unknowns x, y, r:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 - (r_1 \pm r)^2 = 0 \\ (x - x_2)^2 + (y - y_2)^2 - (r_2 \pm r)^2 = 0 \\ (x - x_3)^2 + (y - y_3)^2 - (r_3 \pm r)^2 = 0 \end{cases}.$$

By replacing r by $-r$ in one of the preceding systems of equations, we still get another one of the preceding systems of equations. Thus, let us suppose r is the signed expansion of C_1. Then, we can reformulate the preceding systems of equations as the following systems (II) of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 - (r_1 + r)^2 = 0 \\ (x - x_2)^2 + (y - y_2)^2 - (r_2 \pm r)^2 = 0 \\ (x - x_3)^2 + (y - y_3)^2 - (r_3 \pm r)^2 = 0 \end{cases}$$

Now let us consider for each system (II) the set X of solutions of the system (II) of equations in K^3, where K is an algebraically closed field.

Subtracting one of the equations from the remaining two results in a system of 2 linear equations, from which x and y may be expressed as linear functions of r. Substitution in the first equation then leads to a quadratic equation in r. This means that the unknown quantities x, y, r can be expressed with quadratic radicals as functions of the given centres and radii for each one of the systems of equations above.

As before, though the simplest thing to do now would be to compute the two Voronoi vertices and use their computed coordinates and corresponding signed expansion in the computation of the values certifying the output of the Delaunay graph conflict locator, it is not desirable because this method would not be generalisable to conics or higher degree curves.

For the Delaunay graph of additively weighted points, the true Voronoi vertices are the solutions of one system of algebraic equations. Unlike the previous case, for the Delaunay graph of circles, the true Voronoi vertices are not all the solutions of one system of algebraic equations, but a subset of the solutions of four systems of algebraic equations. The solutions of the algebraic equations are the Apollonius circles, whose centres are generalised Voronoi vertices (a concept that was introduced in [Anton04]). We thus need to determine which Apollonius circles centres are potentially true Voronoi vertices (only the real Apollonius circles centres can be true Voronoi vertices).

There are four possible determinations of the true Voronoi vertices from Apollonius circles centres of C_1, C_2 and C_3:

First case. If C_1, C_2 and C_3 mutually intersect, then the real circles among the seven Apollonius circles that are not internally tangent to each of C_1, C_2 and C_3 correspond to true Voronoi vertices (their centres are true Voronoi vertices, see Figure 17), and reciprocally.

Second case. If one circle (say C_1) intersects the two others (C_2 and C_3) which do not intersect, then only the real Apollonius circles that are either externally tangent to each of C_1, C_2 and C_3, or internally tangent to C_1 and externally tangent to C_2 and C_3 correspond to true Voronoi vertices (their centres are true Voronoi vertices, see Figure 18).

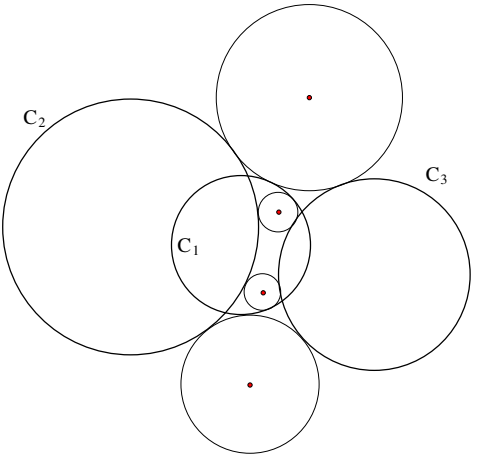


Fig. 18. Four Apollonius circles centres that are true Voronoi vertices (second case)

Third case. If two circles (say C_1 and C_2) intersect the interior of the third one (C_3) and at least one of them (say C_1) is contained in the interior of C_3, then only the real Apollonius circles that are externally tangent to C_1 and C_2 and internally tangent to C_3 correspond to true Voronoi vertices (their centres are true Voronoi vertices, see Figure 19).

Fourth case. Otherwise (if none of the three situations above apply), only the real Apollonius circles that are externally tangent to C_1, C_2 and C_3 correspond to true Voronoi vertices (their centres are true Voronoi vertices, see Figure 20).

When the old Delaunay triangles are checked, the case where one circle (say C_1) lies in the interior of a second circle (say C_2), which lies in the interior of the third circle (C_3), or only one circle (say C_1) lies within the interior of one of the other ones (say C_2) cannot happen because then, there would be no Voronoi vertices and the triangle $C_1C_2C_3$ would not exist in the Delaunay graph. If we check new triangles, we can check if the situation described just above happens by computing the sign of the determinant of the multiplication matrix for the fourth case.

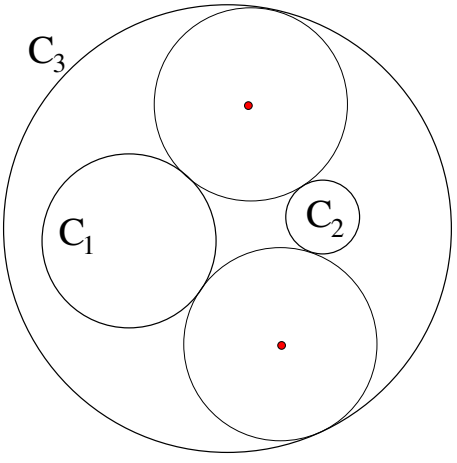


Fig. 19. Two Apollonius circles centres that are true Voronoi vertices (third case)

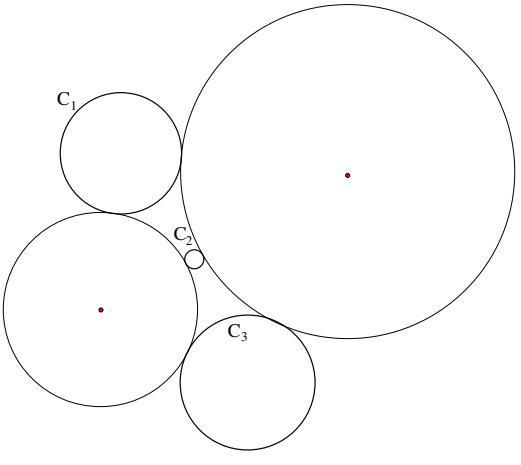


Fig. 20. Two Apollonius circles centres are true Voronoi vertices (fourth case)

Now that we have seen the different cases of true Voronoi vertices, we will see how we can test in which case we are and which solutions of the systems of equations (II) described above correspond to true Voronoi vertices.

First case. C_1, C_2 and C_3 mutually intersect if, and only if, $d(p_1, p_2) - r_1 - r_2 \leq 0$ and $d(p_1, p_3) - r_1 - r_3 \leq 0$ and $d(p_2, p_3) - r_2 - r_3 \leq 0$. The computation of this test can be done exactly, since the only variables that are not input to the Delaunay graph conflict locator are the distances, and these distances are expressed by radicals. Indeed, we need to test the sign of the difference of a radical and a number which do not depend on intermediary computations. The true Voronoi vertices are the real solutions of all the systems of equations (II) such that $r > 0$.

Second case. C_1 intersects C_2 and C_3, and C_2 and C_3 have no point of intersection if, and only if, $d(p_1, p_2) - r_1 - r_2 \leq 0$ and $d(p_1, p_3) - r_1 - r_3 \leq 0$ and $d(p_2, p_3) - r_2 - r_3 > 0$. The computation of this test can be done exactly for the same reasons as the previous case. The true Voronoi vertices are the real solutions of the system of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 - (r_1 \pm r)^2 = 0 \\ (x - x_2)^2 + (y - y_2)^2 - (r_2 - r)^2 = 0 \\ (x - x_3)^2 + (y - y_3)^2 - (r_3 - r)^2 = 0 \end{cases}$$

with $r < 0$.

Third case. C_1 lies in the interior of C_3 and C_2 intersects the interior of C_3 if, and only if, $d(p_1, p_3) + r_1 - r_3 < 0$ and $d(p_2, p_3) - r_2 - r_3 < 0$ and $(x_1 - x_3)^2 + (y_1 - y_3)^2 - r_3^2 < 0$. The computation of this test can be done exactly for the same reasons as the previous case. The true Voronoi vertices are the real solutions of the system of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 - (r_1 + r)^2 = 0 \\ (x - x_2)^2 + (y - y_2)^2 - (r_2 + r)^2 = 0 \\ (x - x_3)^2 + (y - y_3)^2 - (r_3 - r)^2 = 0 \end{cases}$$

such that $r > 0$.

Fourth case. this is the case if all the previous three tests failed. The true Voronoi vertices are the real solutions of the system of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 - (r_1 + r)^2 = 0 \\ (x - x_2)^2 + (y - y_2)^2 - (r_2 + r)^2 = 0 \\ (x - x_3)^2 + (y - y_3)^2 - (r_3 + r)^2 = 0 \end{cases}$$

with $r > 0$.

As before, we used the same algebraic machinery to compute the values of polynomials that are taken by the true Voronoi vertices without solving any intermediate system of equations. We computed the Gröbner basis of the ideal of X for each one of the systems (II) encountered. Each one of these Gröbner bases consists of the earlier mentioned quadratic equation in r and linear equations in x, y and r.

For the Delaunay graph of additively weighted points, we observed that evaluating the signs of a single polynomial ($g = (x_4 - x)^2 + (y_4 - y)^2 - (r + r_4)^2$) taken on the real points of X was enough to provide the values certifying the presence of Voronoi vertices in the list output by the conflict locator. As before, we can check for the existence of real solutions by evaluating the sign of the discriminant of the characteristic polynomial. We will suppose the real solutions to the systems (II) have been tested. Unlike in the previous case, here we need to evaluate the signs taken by both g and r on each one of the points of X. Indeed, we need not only to check the relative position of C_4 with respect to the Apollonius circles, but we need for each Apollonius circle, to check the relative position of C_4 with respect to that Apollonius circle, and to check whether that Apollonius circle corresponds to a true Voronoi vertex.

As before, we considered the operation of multiplication of polynomials by the polynomial g, whose sign expresses the relative position of C_4 with respect to C''. We also considered the operation of multiplication of polynomials by the polynomial r, whose sign allows one to check whether the solutions correspond to true Voronoi vertices. These operations are linear mappings. The operations of these mappings on the canonic representative of the remainder of the Euclidean division of a polynomial by the three polynomials of the system are also linear mappings that can be expressed by a matrix.

We need to be able to associate the signs of the values of g with the signs of the values of r taken on the (real) solutions of each system (II). For a given system (II), let M_g and M_r be the matrices of the result of the multiplication by g and by r respectively on the canonic representative of the remainder of the division of a polynomial by the three polynomials of the system. Since these multiplication maps commute, it is possible to use the transformation matrix obtained during the computation of the Jordan form of one of these matrices to triangularise the other matrix by a simple multiplication of matrices [CLO98]. Indeed, the computation of the Jordan form for M_g gives the triangular matrix $P^{-1}M_gP$ of the Schur form of that matrix where P is a unitary matrix called the transformation matrix; and $P^{-1}M_rP$ is triangular. Finally, we can obtain the solutions by reading the diagonal entries in turn in each one of the Jordan forms of these matrices (the diagonal entries of the Jordan form of a matrix are its eigenvalues). The row number on each one of these matrices corresponds to the index of the solution. By evaluating the signs of the diagonal entries in the Jordan forms of M_g and of M_r on the same line, we associate the signs of the values of g with the signs of the values of r taken on the solutions of each system (II).

5 The Application to the Visualization of the Nucleation and Growth of Particles

The algorithm described in the previous section is applied in this section to the computation of the Johnson-Mehl tessellation, which is a special case of Voronoi diagram of circles. The dual graph of the Johnson-Mehl tessellation is

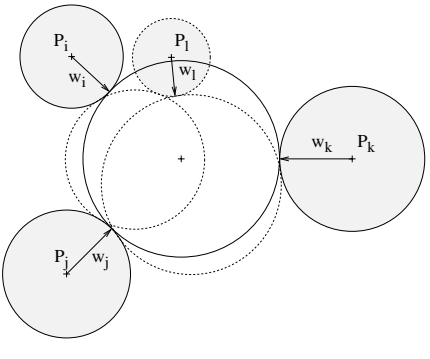


Fig. 21. The event that changes the topology

a triangulation. Now, we will examine the events that affect this triangulation (see Figure 21).

Proposition 9. *(The empty circumcircle criterion for the dual graph of the Johnson-Mehl tessellation): A triangle* $P_i \overset{\triangle}{P_j} P_k$ *exists in the triangulation if, and only if, the circle externally (respectively internally) tangent to the weight circles* $\mathcal{C}(P_i, w_i)$, $\mathcal{C}(P_j, w_j)$, *and* $\mathcal{C}(P_k, w_k)$, *does not intersect properly (non tangentially) any other circle* $\mathcal{C}(P_l, w_l)$, $l \notin \{i, j, k\}$.

Proof. If a fourth circle $\mathcal{C}(P_l, w_l)$ happens to be externally (respectively internally) tangent to the circle $\mathcal{C}_{t_{\{i,j,k\}}}$ that is externally (respectively internally) tangent to $\mathcal{C}(P_i, w_i)$, $\mathcal{C}(P_j, w_j)$, and $\mathcal{C}(P_k, w_k)$, then the vertex $v_{\{i,j,k\}}$ (intersection of $\mathcal{B}_{ij}$, $\mathcal{B}_{ik}$, and $\mathcal{B}_{jk}$) is 4-valent, and the triangle exists in the Delaunay graph.

Otherwise, if the intersection of $\mathcal{C}(P_l, w_l)$ and $\mathcal{C}_{t_{\{ijk\}}}$ was constituted by two different points, then $\mathcal{C}_{t_{\{i,j,l\}}}$ and $\mathcal{C}_{t_{\{j,k,l\}}}$ would be externally (respectively internally) tangent to $\mathcal{C}(P_i, w_i)$, $\mathcal{C}(P_j, w_j)$, and $\mathcal{C}(P_l, w_l)$; and $\mathcal{C}(P_j, w_j)$, $\mathcal{C}(P_k, w_k)$, and $\mathcal{C}(P_l, w_l)$ respectively. Then we would have the triangles $P_i \overset{\triangle}{P_j} P_k$, $P_i \overset{\triangle}{P_j} P_l$, and $P_j \overset{\triangle}{P_k} P_l$, which would contradict the fact that the dual graph of the Johnson-Mehl tessellation is a triangulation (see Figure 22). □

We should therefore make a triangle switch: replace $P_i \overset{\triangle}{P_j} P_k$ and $P_i \overset{\triangle}{P_k} P_l$ by $P_i \overset{\triangle}{P_j} P_l$ and $P_j \overset{\triangle}{P_k} P_l$. Proposition 1 implies that the triangulation mentioned above obeys the Delaunay triangulation "empty circumcircle criterion". This follows the algorithm of Guibas and Stolfi [GS85] for the ordinary Voronoi diagram, extending it to this case of a generalized Dirichlet tessellation. This proposition is the basis of the incremental algorithm that we implemented for the dynamic construction and maintenance of additively weighted Voronoi diagrams. When a new point is added, we locate the triangle T in which it lies, then we connect this new point to the triangulation by replacing T by three new triangles whose vertices are the vertices of T and the new point. Then we check every circle externally respectively internally tangent to the weight circles of the points of every new triangle. If a triangle switch (see Figure 22) has to be performed (see end of the Proof of Proposition 9), we perform the same check for all the externally respectively internally tangent circles corresponding to the triangles generated by the triangle switch (see Figure 2 where the triangle switch is shown: replacing $P_i \overset{\triangle}{P_j} P_k$ and $P_i \overset{\triangle}{P_k} P_l$ by $P_i \overset{\triangle}{P_j} P_l$ and $P_j \overset{\triangle}{P_k} P_l$).

When an existing point is deleted, we locate its nearest neighbour, then we transfer all its neighbours to the nearest neighbour and we remove it and its topological relationships from the triangulation. Then we check every circle externally respectively internally tangent to the weight circles of the points of every modified triangle. If a triangle switch has to be performed (see end of the Proof of Proposition 9), we perform the same check for all the externally respectively internally tangent circles corresponding to the triangles generated by the triangle switch. This is the basis of the incremental algorithm [AMG98a], that we

implemented for the dynamic construction and maintenance of Johnson-Mehl tessellation.

Our algorithm proceeds in a fashion analogous to the algorithm of Devillers, Meiser, and Teillaud [DMT90] for the dynamic Delaunay triangulation based on the Delaunay tree. They proved using the Delaunay tree that each insertion and point location has an expected running time of $O(\log n)$, and each deletion has an expected running time of $O(\log \log n)$. Our algorithm has an efficiency of $O(\log n)$.

5.1 The Johnson-Mehl Tessellation

The algorithm for the construction of the Voronoi diagram of circles has been adapted in order to get the incremental algorithm for the construction and maintenance of the Johnson-Mehl model. After each arrival of a new nucleus, the Johnson-Mehl tessellation changes, and we recompute it as follows. The new nucleus is inserted in the Johnson-Mehl tessellation (a new Voronoi region appears), and the neighbouring Voronoi cells are changed. The size of the spheres is then increased by the growth corresponding to the time interval between the previous insertion and this one $(t_i - t_j)$. Consequently, the spheres will be increased for this time interval (see Figures 22 and 23). This type of spatial growth uses a Poisson point process [OBSC01], and we will now introduce two different cases of radial speed for spatial growth processes.

Time homogeneous Poisson point process. The uniform radial growth of the nuclei and appearance of their Voronoi regions at two different times is shown in Figures 22 and 23. On Figures 22 and 23, we can see the growth of the spheres between two time units. We notice that the Voronoi regions are changed only when a new particle appears.

We assume [Stoya98] that the radial growth speed is the same for all the spheres, and the growth of the spheres in the portion of contact is stopped (see Figures 22 and 23). In the early stages of growth and nucleations spheres do not overlap, but after a certain time a sphere may touch another sphere [OBSC01].

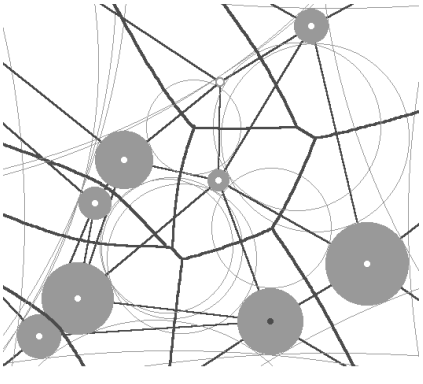


Fig. 22. The growth of particles at $t = 93$

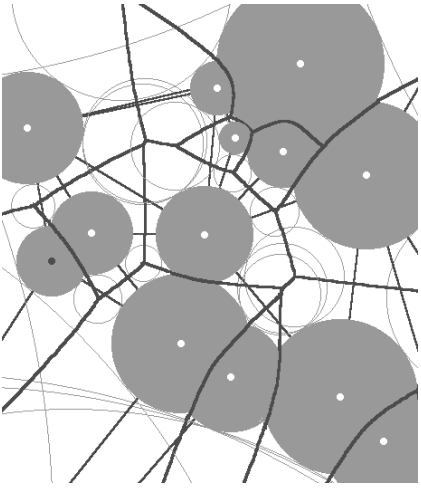


Fig. 23. The growth of particles at $t = 163$

Time inhomogeneous Poisson point process. The Johnson-Mehl model has been generalized [OBSC01] in three different ways: changing the spatial location process for the generators (nuclei), changing the birth rate of the generators, or both. The most extensively studied generalization is the generalization corresponding to the change of the nuclei birth rate as a function of time without changing the spatial location process (the homogeneous Poisson point process). This generalization is known as the time inhomogeneous Johnson-Mehl model. The algorithm for the construction and maintenance of the Johnson-Mehl model is also applied in the case of a time inhomogeneous Poisson point process. In that case, all the nuclei grow at the same radial speed for each time interval and therefore, as long as a new nucleus does not arrive, the difference between the weights of neighbouring nuclei is constant, and the Johnson-Mehl tessellation does not change.

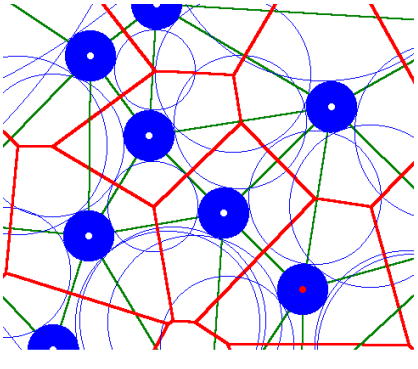


Fig. 24. The Voronoi growth model at $t = 31$

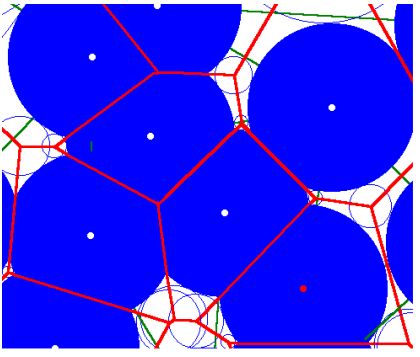


Fig. 25. The Voronoi growth model at $t = 96$

5.2 The Voronoi Growth Model

The Additively Voronoi diagram reduces to the ordinary Voronoi diagram when all the w_i are equal to some constant. In that type of particle growth, nucleation occurs simultaneously. In Figure 24 we can see the simultaneous appearance of the nuclei that are all of the same size. Figure 25 shows the growth of these particles after 65 time units (shown in increased weights). We notice that the tessellation has not changed.

Thus, for the nucleation sites that are appearing simultaneously we have a non-Poisson point process [Stoya98] and we can apply our algorithm that reduces the Johnson-Mehl model to the Voronoi growth model.

6 Conclusions

We have provided a predicate for the incremental construction of the Delaunay graph and the Voronoi diagram of circles that amounts to computing the sign of the eigenvalues of a two by two matrix. Unlike other independent research, our work proposes a single predicate that can compute the Delaunay graph even in the case of one circle being entirely in another circle or intersecting circles. We have also provided an application of the Voronoi diagram of circles to the modelling and the visualisation of the growth of crystal aggregates. We have been also working on the Delaunay graph of conics and of semi-algebraic sets (see [Anton04]), and future work include the Delaunay graph and Voronoi diagram of quadrics and its applications.

References

[AB86] Ash, P.F., Bolker, F.D.: Generalized Dirichlet Tessellations. Geometria Dedicata 20, 209–243 (1986)

[ABMY02] Anton, F., Boissonnat, J.-D., Mioc, D., Yvinec, M.: An exact predicate for the optimal construction of the Additively Weighted Voronoi diagram. In: Proceedings of the European Workshop on Computational Geometry 2002, Warsaw, Poland (2002)

[AK00] Aurenhammer, F., Klein, R.: Voronoi diagrams. In: Handbook of computational geometry, pp. 201–290. North-Holland, Amsterdam (2000)

[AKM02] Anton, F., Kirkpatrick, D., Mioc, D.: An exact algebraic predicate for the maintenance of the topology of the additively weighted voronoi diagram. In: The Fourteenth Canadian Conference on Computational Geometry, Lethbridge, Alberta, Canada, pp. 72–76 (2002)

[AM95] Anishchik, S.V., Medvedev, N.N.: Three-dimensional Apollonian packing as a model for dense granular systems ll. Phys. Rev. Lett. 75(23), 4314–4317 (1995)

[AMG98a] Anton, F., Mioc, D., Gold, C.M.: Dynamic Additively Weighted Voronoi Diagrams Made Easy. In: Proceedings of the 10^{th} Canadian Conference on Computational Geometry (CCCG 1998), Montréal, Canada (1998)

[AMG98b] Anton, F., Mioc, D., Gold, C.M.: An algorithm for the dynamic construction and maintenance of Additively Weighted Voronoi diagrams. In: Proceedings of the 14^{th} European Workshop on Computational Geometry (CG 1998), Barcelona, Spain, pp. 117–119 (1998)

[Anton04] Anton, F.: Voronoi diagrams of semi-algebraic sets, Ph.D. thesis, The University of British Columbia, Vancouver, British Columbia, Canada (2004)

[Auren87] Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. 16(1), 78–96 (1987)

[Auren88] Aurenhammer, F.: Voronoi diagrams - A survey, Institute for Information Processing, Technical University of Graz, Report 263 (1988)

[BB96] Beachy, J.A., Blair, W.D.: Abstract Algebra. Waveland Press Inc. (1996)

[BCSS98] Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and real computation. Springer, New York (1998) (with a foreword by R.M. Karp)

[Berge79] Berger, M.: Géométrie. espaces euclidiens, triangles, cercles et sphères, CEDIC/FERNAND NATHAN, Paris, vol. 2 (1979)

[Boots73] Boots, B.N.: Some models of random subdivision of space. Geografiska Annaler 55B, 34–48 (1973)

[CLO98] Cox, D., Little, J., O'Shea, D.: Using algebraic geometry. Springer, New York (1998)

[CPX02] Chen, Z., Papadopoulou, E., Xu, J.: Robust algorithm for k-gon voronoi diagram construction. In: Abstracts for the Fourteenth Canadian Conference on Computational Geometry CCCG 2002, Lethbridge, Alberta, Canada, August 2002, pp. 77–81. University of Lethbridge (2002)

[Des03] Deschamps, A.: Analytical Techniques for Aluminium Alloys. In: Handbook of Aluminum. Alloy Production and Materials manufacturing, vol. 2, pp. 155–192. Marcel Dekker, Inc., New York (2003)

[DMT90] Devillers, O., Meiser, S., Teillaud, M.: Fully Dynamic Delaunay Triangulation in Logarithmic Expected Time per Operation, Rapport INRIA 1349, INRIA, BP93, 06902 Sophia-Antipolis cedex, France (1990)

[EK06] Emiris, I.Z., Karavelas, M.I.: The predicates of the Apollonius diagram: algorithmic analysis and implementation. Comput. Geom. 33(1-2), 18–57 (2006)

[GP02] Greuel, G.-M., Pfister, G.: A Singular introduction to commutative algebra. In: Bachmann, O., Lossen, C., Schönemann, H. (eds.), With 1 CD-ROM (Windows, Macintosh, and UNIX). Springer, Berlin (2002)

[GS] Grayson, D.R., Stillman, M.E.: Macaulay 2, a software system for research in algebraic geometry, http://www.math.uiuc.edu/Macaulay2/

[GS85] Guibas, L., Stolfi, J.: Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. ACM Transactions on Graphics 4(2), 74–123 (1985)

[Horal79] Horalek, V.: The Johnson-Mehl tessellation with time dependent nucleation intensity in view of basic 3-D tessellations. Mathematical research 51, 111–116 (1979)

[JM39] Johnson, W.A., Mehl, F.R.: Reaction kinetics in processes of nucleation and growth. Transactions of the American Institute of Mining, Metallurgical and Petroleum Engineers 135, 416–456 (1939)

[KE02] Karavelas, M.I., Emiris, I.Z.: Predicates for the Planar Additively Weighted Voronoi Diagram. ECG Technical Report ECG-TR-122201-01, INRIA (2002)

[KE03] Karavelas, M.I., Emiris, I.Z.: Root comparison techniques applied to computing the additively weighted Voronoi diagram. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, pp. 320–329. ACM, New York (2003)

[KKS00] Kim, D.-S., Kim, D.-U., Sugihara, K.: Voronoi diagram of a circle set constructed from Voronoi diagram of a point set. In: Lee, D.T., Teng, S.-H. (eds.) ISAAC 2000. LNCS, vol. 1969, pp. 432–443. Springer, Heidelberg (2000)

[KKS01a] Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set. I. Topology. Comput. Aided Geom. Design 18(6), 541–562 (2001)

[KKS01b] Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set. II. Geometry. Comput. Aided Geom. Design 18(6), 563–585 (2001)

[Kle89] Klein, R.: Concrete and abstract Voronoï diagrams. Springer, Berlin (1989)

[Kol37] Kolmogorov, A.N.: A statistical theory for the recrystallization of metals. Akad. nauk SSSR, Izv., Ser. Matem. 1(3), 355–359 (1937)

[Lan02] Lang, S.: Algebra, 3rd edn. Graduate Texts in Mathematics, vol. 211. Springer, New York (2002)

[Med00] Medvedev, N.N.: Voronoi-Delaunay method for non-crystalline structures. SB Russian Academy of Science, Novosibirsk (2000)

[OBSC01] Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, Chichester (2001)

[Stoya98] Stoyan, D.: Random sets: Models and Statistics. International Statistical Review 66, 1–27 (1998)

[Vor07] Voronoï, G.F.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier mémoire. sur quelques propriétés des formes quadratiques positives parfaites. Journal für die reine und angewandte Mathematik 133, 97–178 (1907)

[Vor08] Voronoï, G.F.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélo̊edres primitifs. première partie. partition uniforme de l'espace analytique à n dimensions à l'aide des translations d'un même polyèdre convexe. Journal für die reine und angewandte Mathematik 134, 198–287 (1908)

[Vor10] Voronoï, G.F.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélo̊edres primitifs. seconde partie. domaines de formes quadratiques correspondant aux différents types de parallélo̊edres primitifs. Journal für die reine und angewandte Mathematik 136, 67–181 (1910)

Appendix 1

The Macaulay 2 program for the exact Delaunay graph conflict locator for circles

```
gbTrace 4
dim FractionField  :=   F  ->  0
P  =  frac(QQ[a,b,c,d,e,f,g,h,i,j,k,l])
R = P[x,y,t]
cercle1 = (x-a)^2+(y-b)^2-(c+t)^2
cercle2 = (x-d)^2+(y-e)^2-(f+t)^2
cercle3 = (x-g)^2+(y-h)^2-(i+t)^2
emptycircle  =  ideal(cercle1,cercle2,cercle3)
ecgb  =  gb  emptycircle
print  ecgb
eckb  =  basis  cokernel  gens  ecgb
print  eckb
 kl  =  sort(flatten(entries(eckb)))
 kmind  =  splice {0..#kl - 1}
scan(kl,entry->print ring entry);
hashlist  =  pack(2,mingle(kl,kmind));
 feetmon  =  applyKeys(hashTable hashlist, key->toString(key));
compmat = f -> (htl=apply(kl,be->
hashTable(pack(2,mingle(apply(flatten(entries((coefficients((f*be)
%ecgb))#0)),
item  ->  feetmon#(toString(item))),flatten(entries((coefficients
((f*be)%ecgb))#1))))));
matrix(table(#kl,#kl,(i,j)->if  (htl#i)#?j  then  (htl#i)#j  else
0)));
matp2  =  compmat((x-j)^2+(y-k)^2-(t+l)^2);
m00  =  matp2_(0,0)
m01  =  matp2_(0,1)
m10  =  matp2_(1,0)
m11  =  matp2_(1,1)
cm00  =  coefficients m00
cm000  =  cm00#0
cm001  =  cm00#1
cm01  =  coefficients m01
cm010  =  cm01#0
cm011  =  cm01#1
cm10  =  coefficients m10
cm100  =  cm10#0
cm101  =  cm10#1
cm11  =  coefficients m11
cm110  =  cm11#0
cm111  =  cm11#1
```

Designing Aircraft Cockpit Displays: Borrowing from Multimodal User Interfaces

Mladjan Jovanovic[1], Dusan Starcevic[2], and Zeljko Obrenovic[3]

[1] Center for Command Information Systems, Military of Serbia
mladjan@afrodita.rcub.bg.ac.yu
[2] School of Business Administration, University of Belgrade
starcev@fon.bg.ac.yu
[3] Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands
zeljko.obrenovic@cwi.nl

Abstract. In this paper, we present an approach to design of command tables in aircraft cockpits. To date, there is no common standard for designing this kind of command tables. Command tables impose high load on human visual senses for displaying flight information such as altitude, attitude, vertical speed, airspeed, heading and engine power. Heavy visual workload and physical conditions significantly influence cognitive processes of an operator in an aircraft cockpit. Proposed solution formalizes the design process describing instruments in terms of estimated effects they produce on flight operators. In this way, we can predict effects and constraints of particular type of flight instrument and avoid unexpected effects early in the design process.

Keywords. Aircraft cockpit, multimodal user interfaces, aircraft instrument, formal description of cockpit display.

1 Introduction

Today's modern aircraft operators rely on vast amount of data that has to be presented in real-time. The meaning of this data is difficult to asses in its raw format. Therefore, we need sophisticated methods to interpret and present data to the user in a suitable format [1]. There is also a need for a data visualization platform that can distribute flight data to a variety of animated graphical displays for easy interpretation by the aircraft operator. Large amounts of airflow velocity data presented in real-time cause numerous effects on human sensory and perceptual apparatus. In situations where the operator must react in a limited period of time and avoid hazardous situations, it is very important to present flight data in a form that can be easily interpreted and processed having in mind user's abilities and preferences. This paper addresses the problem of adapting immense amount of visualization data to the operator in an aircraft cockpit based on the ideas from the multimodal human-computer interaction [13, 16].

This paper is structured as follows. In next section, we give an overview of the research field and some existing solutions. Then, we discuss basic concepts of multimodal systems from the point of our interests. After that, we describe the proposed

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 55–65, 2009.

approach, where we present formal description technique based on the existing meta-model of multimodal human-computer interaction. In Section 4, we demonstrate our solution giving the case study example of Unmanned Aerial Vehicle's (UAV) visual instrument that we have developed. Finally, we give short discussion and conclusions.

2 Background

In our approach, we are reusing ideas from multimodal user interfaces, and applying them in the designing of aircraft cockpit displays. In this section, we give an overview of these two fields, emphasizing their similarities.

2.1 Problem Background

Over the last few decades, the continuous global growth of air traffic has led to increasing problems with respect to airspace capacity and delays [2]. This situation has initiated the research for new operational concepts and aircraft systems that aim for more independent aircraft systems in order to probe the human factors of pilots when operating in aircraft cockpit. Key aspects of this research include modeling interaction in complex time-critical environments like aircraft cockpit and providing timely context-sensitive information in real time without overloading or distracting the human operator [3]. In aircrafts, human-machine interaction is the key issue in providing situational awareness and maintaining safety. The operator functions as an observer who monitors the displays information from the flight computer, pays attention to the environment and concentrates on communication tasks. To facilitate the amount of work and tasks he or she has to accomplish, the aircraft becomes more and more computerized. However, the displays in the cockpit of an aircraft can be quite complex and have to function in a harsh visual environment that may strongly affect the quality of the displayed information. Numerous reports and studies clarify specific fields of research such as situation awareness [4], tactile sensation [5], color patterns [6] and so forth. Major drawback of existing solutions is a lack of operational feedback regarding human performances connected with audio, visual and haptic cues in highly interactive environments such as aircraft cockpit.

If we consider interfaces developed in the field of highly interactive (also called post-WIMP) applications, the dynamicity of interaction objects in terms of existence, reactivity and interrelations appears as a new characteristic [7]. These interfaces may include new interactors such as graphical representations of aircrafts at any time during the use of application. Even though this kind of problems is easily handled by programming languages it is hard to master it in terms of models. This is why classical formal description techniques must be improved to be able to describe highly interactive environment in a complete and unambiguous way. The reason for deployment of formal description techniques lies in the fact that they are means for modeling all components of an interactive system (presentation, dialogue and functional module). Besides, they are usually applied to early phases of development process and clarify their limits when it comes to evaluation.

2.2 Multimodal Human-Computer Interaction

Multimodal systems represent a research-level paradigm shift from conventional WIMP interfaces toward providing users with greater expressive power, naturalness, flexibility and portability [8]. Multimodal research focuses on human perceptual channels [9]. User communicates with the system through set of communication channels which use different modalities, such as visual display, audio, and tactile feedback, to engage human perceptual, cognitive, and communication skills in understanding what is being presented. Multimodal systems integrate various modalities simultaneously, sequentially or independently what is defined by multimodal integration patterns [10].

Various systems offering multimodal interaction techniques have been provided since the early work Bolt in early 80's [11]. Although some real systems have been presented, development process of multimodal interactive systems remains difficult task usually carried out by an ad hoc process.

Previous study on multimodal interaction [8] has shown that multimodal interaction presents several advantages:

- Multimodality increases the overall efficiency of interaction. Task-critical errors decrease during multimodal interaction. This advantage justifies the use of multimodal techniques in highly interactive environments (for instance aircraft cockpit).
- Flexibility of a multimodal interface can accommodate a wide range of users, tasks and environments-including users who are temporarily or permanently handicapped and usage in adverse surroundings (aircraft cockpit, for example).
- Users have a strong preference to interact multimodally. This preference is most pronounced in spatial domain systems when describing spatial information about location, number, orientation or shape of an object.
- Multimodality provides greater naturalness and flexibility of interaction that facilitates learning process. This can be very useful for the flight simulator training.

We find multimodal interaction techniques very useful for designing user interfaces in an aircraft cockpit from the point of quantity (they can increase the bandwidth between user and system) and quality (extracting and adapting content according to user abilities and preferences).

For all these reasons, multimodal human-computer interaction appears to be very useful in the field of interactive systems. It permits enhancing human-computer interaction in these systems, but formal description technique is needed to describe entire multimodal interactive system in a way that can be incorporated in current software development practices.

3 Proposed Solution

In this section, we describe how we model aircraft cockpit displays as a multimodal interface. We begin with an overview of the vocabulary of modeling primitives. Then we define basic steps for describing aircraft displays as complex modalities, and describe how these models can be used in evaluating human performances. In the following section, we give a concrete example of a formal description of a visual instrument as a complex modality.

3.1 Metamodel of Sensory, Motor, Perceptual and Cognitive Effects

The engineering of multimodal systems introduces additional complexity to the development of interactive software systems rarely addressed by current software development methodologies. For example, the UML Unified Software Development Process [12] devotes only a short paragraph to the design of the user interface. For describing multimodal interfaces we use set of modeling primitives defined by the semantic metamodel of multimodal interaction which has been previously developed [13].

The main concept of the metamodel is a HCI modality, which is described as a form of interaction designed to engage some of human capabilities, e.g. to produce some effects on users. A HCI modality can be simple or complex. A complex modality integrates other modalities to create simultaneous use of them, while a simple modality represents a primitive form of interaction. Simple HCI modality can be input or output, using the computer as a reference point. Input and output modalities are not symmetric with human input and output modalities because they represent a computer viewpoint, where it is computer code, not neural circuity, that control interaction with users. Each modality engages some of human capabilities, e.g. it produces some effects on the user. Effects are classified in four main categories: sensory, perceptual, motor, and cognitive (Table 1).

Table 1. Classification of sensory, perceptual, motor and cognitive concepts

Classification	*Concepts*
Sensory	Stimulus: light, sound, vibration Sensory excitation Sensory processing: color, sharpness, peripheral vision
Perceptual	Pattern recognition Grouping: similarity, proximity, or voice color or timber Highlighting : color, polarity, or intensity 3D cue such as stereo vision or interaural time difference Illusion
Motor	Movement: translation or rotation Force: pressure or twisting Hand or head movement Degree of freedom
Cognitive	Short- or long-term memory and memory processes such as remembering forgeting Attention: focus and context Reasoning: deductive, inductive, and abductive Problem solving: Gestalt, problem space, and analogical mapping Analogy Skill acquisition: skill level, proceduralization, and generalization Linguistics: speech, listening, reading, and writing

Sensory effects describe processing stimuli performed by human sensory apparatus. Perceptual effects are more complex effects that human perceptual systems get by analyzing data received from sensors. Motor effects describe human mechanical action, such as head movement or pressure. Cognitive effects describe effects that take

place at higher level of human information processing, such as memory processes, attention, and curiosity. Furthermore, effects are often interconnected. For example, all perceptual effects are a consequence of sensory effects. These relations among effects are important because in this way a designer can see what side effects will be caused by his intention to use some effects.

3.2 Proposed Approach

Our approach is inspired by the model-driven development, where software development's primary focus and products are models rather than computer programs. In this way, it is possible to use concepts that are much less bound to underlying implementation technology and are much closer to the problem domain [14].

In the design of the instrument table, we have classified instrument types by analogy with modalities as basic or complex. Basic instrument tracks simple parameter values and changes and engages a specific human sense. According to type of human sensory

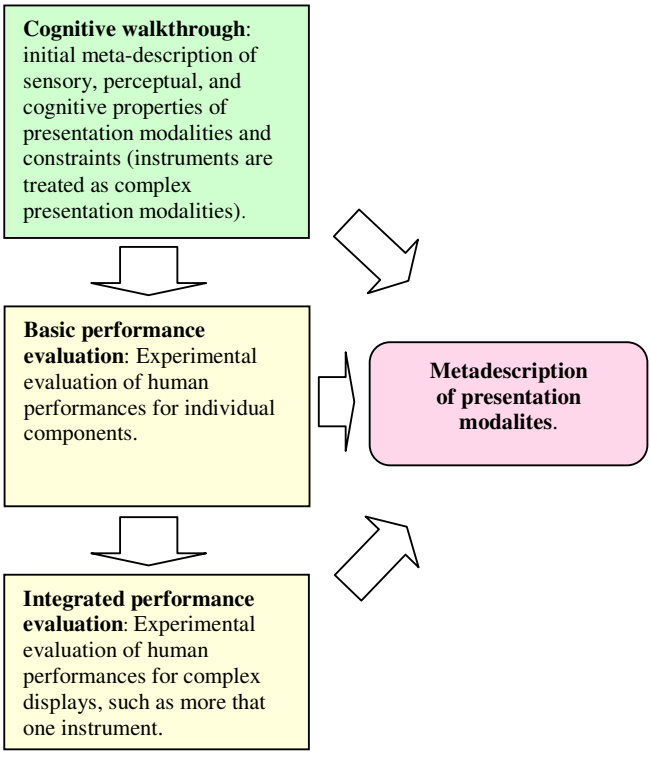


Fig. 1. Proposed design process

system it excites, basic instrument can be visual, audio or haptic. Each basic instrument consists of an instrument scale, instrument pointer, instrument zone, and scale marker. Complex instrument integrates other instruments combining information aimed at specific human sensory apparatus into complex and uniform excitation event.

Each individual instrument engages some human capabilities. Communication channel established between the human and system is parameterized by effects produced on the user. By classifying instruments into categories, we can have an insight into specific effects produced by them, which enables predicting effects conducted in complex instruments where various types of signals interfere and integrate. Next step is connecting estimated effects with cockpit environment characteristics and operator abilities that increase or decrease them. In this way, we can treat each instrument as a presentation modality having some inherit sensory, perceptual or cognitive qualities. Thus, a concrete instance of some instrument will add or change some qualities according to user abilities and preferences, for example, by choosing color scheme or shape pattern that can introduce some analogy. Upon these instruments descriptions experimental evaluation of human performances for individual and complex components is done in order to conclude metadescription of presentation modalities as shown in Figure 1. Given the metadescriptions of presentation modalities, each instrument is considered as an instance of defined metamodels.

Mapping between instruments and effects can serve several purposes. It provides context where we could perceive many relations that are not always obvious. Predicting effects that an instrument causes on humans and connecting these effects with descriptions of limiting environment characteristics gives an opportunity to avoid some undesired situations which can occur (for example, increasing visual workload during instrument scan). Finally, information channels between users (pilot/operator) and device (the aircraft) are described in a uniform and an unambiguous way.

4 Design Case Study: A Visual Instrument

We have applied our approach in designing virtual cockpit for close-range Unmanned Aerial Vehicles (UAV). Requirements for human-computer interface developed are as stated [17]:

- Ergonomic Goal. In order to minimize physical fatigue, the system has to form and fit a human body and to give comfortable environment (temperature and lighting).
- Cognitive Goal. In order to decrease cognitive fatigue, the system should use analog versus digital displays. Placement and font of text and appropriate symbol shapes and colors should minimize scan time.
- Response Goal. This concerns minimizing UAV response time and is achieved by underlying implementation technology.

To realize these three goals we have applied proposed approach. Figure 2 is a UML class diagram, created with defined UML extensions [13], describing the effects of a visual instrument as a complex presentation modality. These effects are used as a basis for achieving our ergonomic, cognitive and response goals. An instrument's basic presentation modality is an instrument pointer that presents the current value of tracked parameter. A *DynamicInstrumentPointer* is modeled as a dynamic output

modality animating presentation of the *StaticInstrumentPointer*. Instrument pointers introduce several perceptual effects: it is recognizable by its shape; orientation denotes its current position; and interposition highlights pointer from instrument scale ticks. By smoothly animating positions of the pointer, a *DynamicInstrumentPointer* gives a notion of motion. An *InstrumentScale* defines global extent in which parameter value can change. Scale is presented to the user as a set of *ScaleMarkers* described as static output modalities. Scale markers add perceptual effects of highlighting by shape, size and color. To distinguish between normal and critical extents of parameter values, an *InstrumentScale* consists of several *InstrumentZones*, also described as static presentation modalities. Each zone defines local extent in which parameter's value varies. Zones distinguish themselves by introducing perceptual effects of highlighting by color and shape, and grouping proximate of visual indicator for parameter values. A *VisualInstrument* presents complex modality integrating *InstrumentScales* and *DynamicInstrumentPointers*. A set of visual instruments represents multimodality *VisualInstrumentPresentation* which engages human cognitive functions of reading and information retrieval.

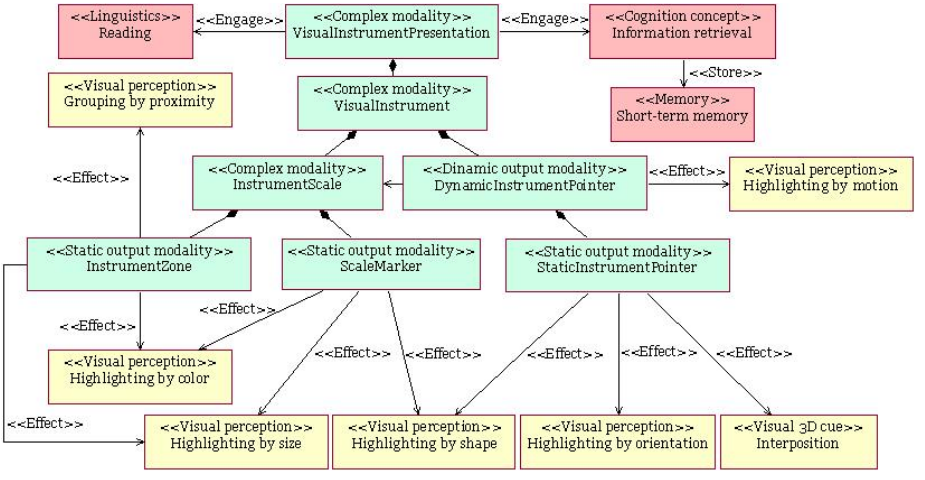
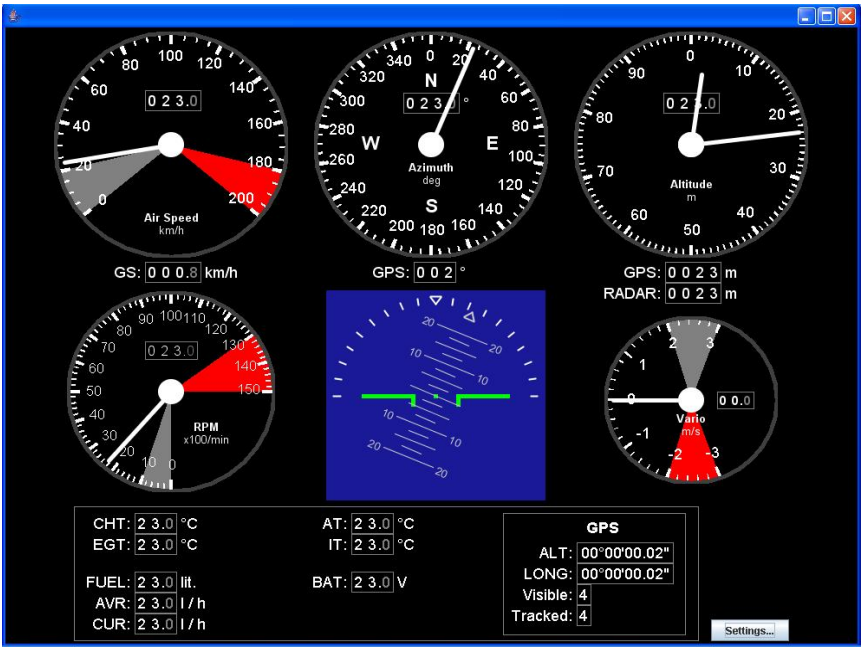
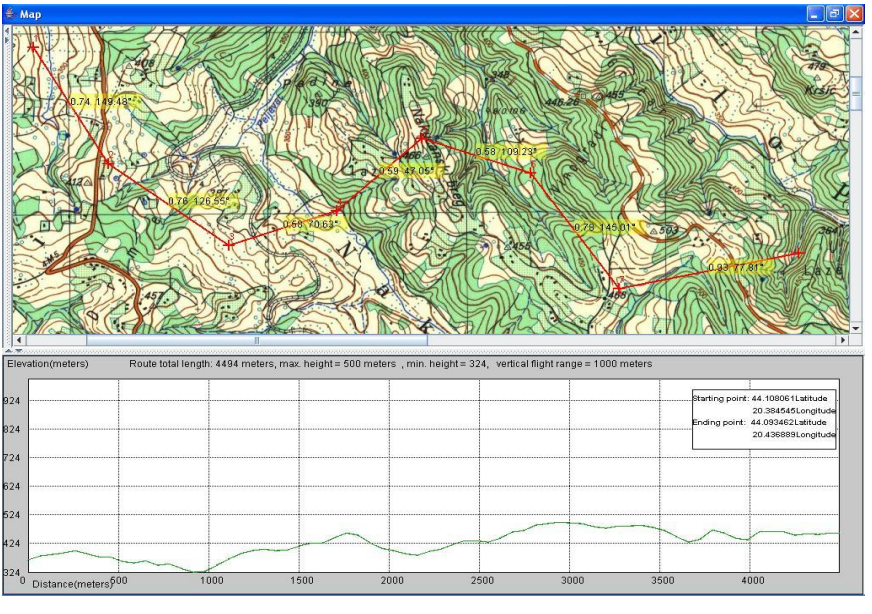


Fig. 2. Visual instrument depicted in terms of effects

Figure 3 shows instrument table developed upon given metadescriptions. These metadescriptions where most useful in cognitive walkthrough phase, as we noticed that most of the designers and programmers are not aware of the huge number of parameters that presentation effects introduced by every part of user interface. Presented display operates in a way that represents an operator's intuitive understanding. Controls that have different functions are distinguishable from one another in order to clearly assess flight status data. Instruments and controls with related functions are grouped together in a logical arrangement, which helps reduce instrument scan time and lowers operator's workload.



(a)



(b)

Fig. 3. Instrument table as instance of complex modality (a), and the window showing aircraft mission route (b)

5 Discussion

The presented work can serve several purposes. First, we demonstrated the ability to predict effects that certain type of instrument produces on humans. Proposed instrument classification connected with the metamodel of multimodal communication gives us predictive and explanatory approach for describing complex effect notions in an aircraft cockpit and connecting them with user and device profiles. Describing a cockpit in a common language, we facilitate more effective user interfaces. Designing displays and information flow at higher level of abstraction enables predicting undesirable effects that can appear early in the design and reduces information overload. What is more important it reduces interdisciplinary gap among designers and allows integration of results from various fields of research. For example, multimodal research techniques introduce results that have been used as a basis for a measurement and enhancement of situational awareness [15]. Metadescriptions of instruments as presentation modalities with some sensory, perceptual and cognitive qualities permits experimental evaluation of human performances for complex displays from which users can clearly benefit. Evaluation results allow seeing if concrete aircraft display suits users abilities and preferences.

Proposed approach describes all effects introduced by the instrument table. However, for more detailed analysis, it is useful to include a notion of a visual scan, which is currently partially addressed by our approach. Visual scan considers a sequence of monitoring tasks associated with flight status. Scan characteristics (where to look, how frequently and how long) are currently determined by the complexity of the information provided by devices and level of operator's expertise. Operator/pilot forms a mental model as a comprehensive understanding of a system and its dynamics. However, mental models are refined with experience, so less experienced operator can employ random scan that is not sensitive to the changing needs for information from one moment to the next. Experienced pilots often feel uncomfortable when transitioning to a new aircraft because of a conflict between their mental model and arrangement of instruments in this new aircraft cockpit. Describing cockpit at higher level of abstraction facilitates transfer of operational skills between various systems and avoids negative learning transfer.

The efficiency of usage of our method depends very much on the efficiency of supporting tools. In our current approach, we are relying on the existing UML modeling tools, and their integration mechanisms. The advantage is that the designers who are familiar with UML can do the design in their natural environments. Additional advantage is that the UML tools, such as Rational Rose, enable integration of custom code connecting the tool with other systems. However, the problem with UML tolls is lack of rigorousness in modeling, which requires discipline at the side of the designer. Tools that can support analysis of the designed models are a subject of our future work.

In order to take into account type of aircraft, level of aircraft operator training, environment, our method allows definition of different models of users and interfaces, at different levels of abstraction. Models can be organized hierarchically and grouped according to different aspects. The models can be reused, which reduce the effort. In our experiences, creation of the initial model is the most time consuming effort.

In the end, we would like to add that one of the advantages is the increased awareness of the designers and programmers about the human factors involved in the design of interfaces.

6 Conclusion

The presented work describes an approach to modeling aircraft cockpit devices in terms of multimodal interfaces using the UML notation [13]. This work could help cockpit designers in analyzing the information presentation to humans and avoiding overload as well as streamlining information acquisition. Each instrument consists of one or more modalities (depending on its complexity) and causes one or more effects on the user/operator. In essence, the instrument is a container for one or more information channels between operator/pilot and the device (the aircraft). If we describe the whole cockpit in terms of modalities, we get a unified way of analyzing the inputs and outputs and the resulting effects on the operator (and the device). This can be used as a basis for analyzing cognitive load as well as studying the expressiveness the inputs provide in controlling the aircraft.

We have illustrated our approach on the example of unmanned aircraft vehicle, but it is applicable for manned aircrafts as well. Presented work is a part of ongoing project and is developed as an experimental prototype. In our future work, we plan to integrate our solution into existing CASE tools and work on implementation of tools for designing aircraft cockpits based on multimodal technique presented as a proof of feasibility of the approach.

References

1. Kaplan, J., Chen, R., Kenney, P., Koval, K., Hutchinson, B.: User's Guide for Flight Simulation Data Visualization Workstation. NASA Technical Memorandum 110294 (1996)
2. Valenti Clari, S.V.M., Ruigrok, C.J.R., Heesbeen, W.M.B., Groeneweg, J.: Research Flight Simulation of Future Autonomous Aircraft Operations. In: Proc. Winter Simulation Conference, pp. 1226–1234 (2002)
3. Irving, S., Mitchel, M.C.: Report on the CHI 1990 Workshop on Computer-Human Interaction in Aerospace Systems. SIGCHI Bulletin 23(1), 17–23 (1990)
4. Hourizi, R., Johnson, P.: Designing To Support Awareness: A Predictive, Composite Model. In: CHI 2004, pp. 159–166. ACM press, New York (2004)
5. Nojima, T., Funabiki, K.: Cockpit Display using Tactile Sensation. In: Proc. of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (2005)
6. Evreinova, T., Raisamo, R.: An Evaluation of Color Patterns for Imaging of Warning Signals in Cockpit Displays. In: Proc. NordiCHI, pp. 205–207 (2002)
7. Jacob, R.: A Software Model and Specification Language for Non-WIMP User Interfaces. ACM Transactions on Computer-Human Interaction 6(1), 1–46 (1999)
8. Oviatt, S.: Ten Myths of Multimodal Interaction. Communications of ACM 42(11), 74–81 (1999)
9. Turk, M., Robertson, G.: Perceptual User Interfaces (Introduction). Communications of the ACM, 33–35 (March 2000)

10. Oviatt, S., DeAngeli, A., Kuhn, K.: Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. In: Proc. CHI 1997, pp. 415–422 (1997)
11. Bolt, R.A.: Put that there: Voice and gesture at the graphics interface. ACM Computer Graphics 14(3), 262–270 (1980)
12. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Adisson-Wessley, Reading (1999)
13. Obrenovic, Z., Starcevic, D.: Modeling Multimodal Human-Computer Interaction. IEEE Multimedia 11(1), 65–72 (2004)
14. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software 20(5), 19–25 (2003)
15. Denford, J., Steele, J.A., Roy, R., Kalantzis, E.: Measurement of Air Traffic Control Situational Awareness Enhancement Through Radar Support Toward Operating Envelope Expansion of an Unmanned Aerial Vehicle. In: Proc. of the 2004 Winter Simulation Conference (2004)
16. Obrenovic, Z., Abascal, J., Starcevic, D.: Universal Accessibility as a Multimodal Design Issue. Communications of the ACM 50(5) (May 2007)
17. Miller, N.: Designing Humans for UAVs: An Operator's Perspective. In: CERICI Workshop (2004)

Parallel Optimal Weighted Links

Ovidiu Daescu[1,⋆], Yam K. Cheung[1], and James D. Palmer[2]

[1] Department of Computer Science,
University of Texas at Dallas Richardson, TX 75080, USA
{daescu,samykcheung}@utdallas.edu
[2] Department of Computer Science,
Northern Arizona University Flagstaff, AZ 86011, USA
James.Palmer@nau.edu

Abstract. In this paper we consider parallel algorithms for computing an optimal link among weighted regions in the plane. The problem arises in several areas, including radiation therapy, geological exploration and environmental engineering. We present a CREW PRAM parallel algorithm and a coarse-grain parallel computer algorithm for this problem. For a weighted subdivision with n vertices, the *work* of the parallel algorithms we propose is only an $O(\log n)$ factor more than that of their optimal sequential counterparts. We further adapt an algorithm for minimizing sum of linear fractionals, that has inherent parallelism, to solve in parallel the global optimization problems associated with our solution for the weighted region optimal link problem.

1 Introduction

Computing optimal paths in the plane in various settings is a fundamental topic in computational geometry. Most of the results on computing optimal paths consider only one optimization criteria, such as the length of the path or the number of turns on the path, and usually assume that the plane is divided into free space and obstacles that the path must avoid. This last condition can be relaxed by assigning positive weights to various regions of the plane. In this scenario, a path can go through any region but it incurs a cost that is proportional to the weight of the region.

A special class of problems, arising frequently in applications, is that of finding paths that are optimal with respect to multiple criteria. For example, one may wish to compute a path that has only a few turns/links (at most k, for some integer value k), and a small length (the smallest among all possible paths with no more than k turns).

In this paper we consider a special case of bicriteria shortest path problems in planar (2-D) weighted polygonal subdivisions. The problem, referred to as the *weighted region optimal link problem*, is defined as follows.

⋆ Corresponding author. Daescu's research was supported in part by NSF award CCF-0635013.

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 66–81, 2009.

Definition 1. *Given a polygonal subdivision* $R = \{R_1, R_2, \ldots, R_m\}$ *of the plane, with* m *weighted regions and a total of* n *vertices, find a* link *(line, semi-line, or line segment)* L *such that: (1)* L *intersects two given regions* $R_s, R_t \in R$ *and (2) the weighted sum* $d(L) = \sum_{L \cap R_i \neq \phi} w_i * d_i(L)$ *is minimized, where* w_i *is the positive weight of region* R_i *and* $d_i(L)$ *is the Euclidean length of* L *within* R_i*.*

Depending on the application, the link L may be (a) unbounded (e.g., a line): the link L "passes through" the regions R_s and R_t; (b) bounded at one end (e.g., a ray): L originates on the boundary of R_s and it stabs R_t and (c) bounded at both ends (e.g., a line segment): L originates on the boundary of R_s and it ends on the boundary of R_t; in this case L is also called a *minimum separation* for R_s and R_t [16]. Throughout the paper, we assume the subdivision is a straight line embedding of a planar graph, with the internal faces triangulated, the inner boundary of the external face a convex polygon, as shown in Fig. 1, and the weight of the external face set to zero. We also assume that R_s and R_t are internal regions of R and they can be separated by a vertical line.

The weighted region optimal link problem is an extension of the optimal weighted penetration problem introduced in [9], where the source region R_s is the external face of the subdivision. The problem arises in several areas such as geographic information systems, radiation therapy, stereotactic brain surgery, geological exploration, environmental engineering and military applications. For example, in military applications the weight w_i may represent the probability to be seen by the enemy when moving through R_i, from a secured source region R_s to another secured target region R_t. In radiation therapy, it has been pointed out that finding the optimal choice for the link (cases (a) and (b)) is one of the most difficult problems of medical treatment optimization [7].

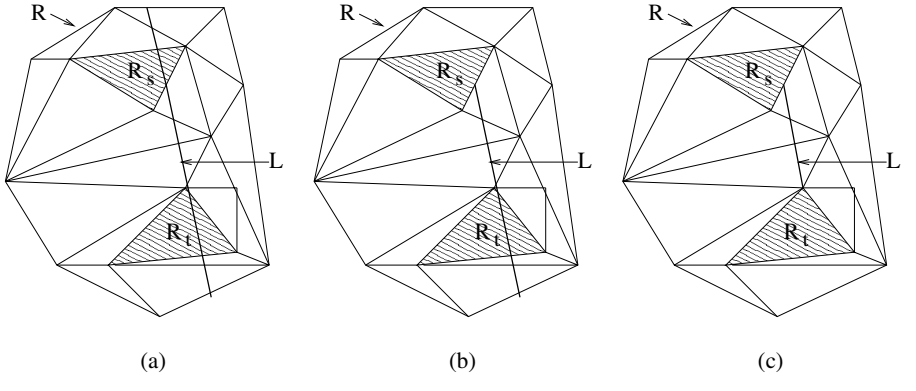


Fig. 1. Illustrating the problem: (a) L intersects R_s and R_t; (b) L originates on the boundary of R_s and stabs R_t and (c) L originates on the boundary of R_s and ends on the boundary of R_t

1.1 Previous Work

Optimal path and bicriteria optimal path problems have been long studied in computational geometry and we refer the reader to [26,29] for comprehensive surveys of the topic. A few results are known for computing or approximating minimum link and k-link shortest paths inside simple polygons and polygons with holes [4,5,28] (an extensive survey is given in [24]).

There are a few results in computational geometry that consider weighted region problems for computing or approximating optimal shortest paths between pairs of points [1,2,3,13,23,25,27,30,31]. Mitchell and Papadimitriou [27] first presented an algorithm that computes an $(1+\epsilon)$ approximate geodesic shortest path between two points in a weighted planar subdivision in $O(n^8 B)$ time and $O(n^4)$ space, where B is a factor representing the bit complexity of the problem instance. The algorithms in [23] take $O(n^5)$ and $O(n^3 \log n)$ time to find an approximate shortest path on triangulated 3-D polyhedral surfaces with weighted facets. The subsequent work on the weighted region shortest path problem is based on placement of Steiner points either on edges [1,3,13,30,31] or on face bisectors [2]. The Steiner points are used to build a discretization graph, in which the nodes are the vertices of R and the Steiner points, and the edges correspond to links between pairs of Steiner points that are on the same face but not the same edge. Results in most of this work depend on some geometric parameters, such as the minimum angle in the subdivision, or the largest integer coordinate of a vertex. The most recent work on the problem [13] describes an algorithm with time bound of $O((n^3\rho/\epsilon)\log\rho\log(\rho n/\epsilon))$, that computes a $(1+\epsilon)$-approximate shortest path, where the weights of R are in $[1,\rho]$ and ϵ is any positive constant. The running time of the algorithm depends on the largest weight of R but it does not depend on any geometric parameters.

The weighted region optimal link problem however has a different structure than the shortest path problem. Important steps towards solving the optimal weighted penetration problem have been made in [9,11], where it has been proven that the 2-D problem can be reduced to at most $O(n^2)$ subproblems, each of which asks to minimize a 2-variable function over a convex domain, where the function is given as a sum of $O(n)$ fractional terms. In [9,11] they show that all subproblems can be generated sequentially in $O(n^2)$ time by sweeping the arrangement of lines dual to the subdivision vertices while maintaining simple data structures that allow to efficiently update the fractional terms in each objective function. Thus, the bulk of computation consists of solving the $O(n^2)$ global optimization subproblems. To compute the optimal solution for each subproblem, a global optimization software has been used in [9]. In [16] it has been proven that an optimal link goes through a vertex of the subdivision R, by showing that the 2-variable objective functions attain their global optimum on the boundary of their feasible domains. Thus, 2-variable functions are reduced to 1-variable functions and the feasible domains become intervals on the real line. They [16] give an $O(n^3 \log^3 n)$ time sequential algorithm for computing an optimal link between two regions R_s and R_t of R, and also describe two efficient approximation schemes. In [17,18] they consider the more general problem

of finding a k-link shortest path between R_s and R_t, where a k-link path is a polygonal path with $k-1$ turns, and a k link shortest path is a shortest possible path with no more than $k-1$ turns. Their best solution is an algorithm that generates paths of weighted length at most $(1+\epsilon)$ times the weight of a k-link shortest path, for any fixed $\epsilon > 0$, while using at most $2k-1$ links.

1.2 Our Results

In [16], it has been pointed out that solving the optimization problems associated with finding an optimal link requires significant time and memory usage if the number of fractional terms in the objective function is large (see also [10]). Since in practical applications having hundreds and even thousands of terms in the objective function is not an uncommon case, sequentially solving the optimization problems on a single processor seems impractical. Instead, one can take advantage of the fact that once the feasible domain and the objective function for each subproblem have been produced, the global optimization problems (GOPs) are independent and can be solved in parallel. After all GOPs are solved, the optimal solution can be obtained by a simple minimum selection.

It is then of interests to efficiently produce the set of GOPs in parallel. We consider this problem and present the following results[1]:

1. We give an $O(\log n)$ time, $O(n \log n + k)$ processors algorithm in the CREW PRAM model, where k is the total complexity description for the feasible domains of the GOPs ($\Omega(n^2)$ in the worst case). The algorithm is based on the arrangement sweeping techniques of Goodrich *et al.* [22]. Our parallel algorithm implies an optimal output sensitive $O(n \log n + k)$ time sequential algorithm for generating all GOPs, by using the optimal segment arrangement construction in [8].
2. We show that, if at most n processors are available, all GOPs can be generated using $O(n^2 \log n)$ work. This algorithm is targeted to coarse-grain parallel computer models, consisting of a relatively small set of nodes (up to a few thousands), where each node has its own processor, with fair computing power, and a large local memory, allowing to store all data involved in (sequentially) solving the problem. Our coarse grained parallel computer algorithm is simple and can be easily implemented.
3. After proving that GOPs can be generated fast in a coarse grained parallel computer model, we show how to efficiently solve a GOP in parallel. Specifically, we adapt an algorithm for minimizing *sum of linear fractionals* (SOLFs) [20], that has inherent parallelism, to solve the GOPs associated with the weighted region optimal link problem.

The paper is organized as follows. In Section 2 we introduce some background and useful structures. Our parallel algorithms to generate the GOPs are presented in Section 3. In Section 4 we discuss a parallel algorithm for solving a GOP. We conclude the paper in Section 5.

[1] A preliminary version of some of these results appeared in [15].

2 Preliminaries and Useful Structures

In this section we introduce some notations and geometric structures.

The PRAM (Parallel Random Access Machine) model of computation is a shared-memory model in which processors act synchronously. In a parallel step, each processor reads or writes a location in the shared memory or it performs some simple, $O(1)$ time computation within its own registers. All communication between processors is done via the shared memory. There are a few variants of the PRAM model. In this paper, we are concerned with the CREW (Concurrent Read Exclusive Write) version of the model, in which many processors may simultaneously access for reading the same memory location on the shared memory space but no two processors can simultaneously write at the same memory location.

A coarse-grain parallel computer model assumes a relatively small set of nodes (up to a few thousands), where each node has its own processor, with fair computing power, and a large local memory, allowing to store all data involved in (sequentially) solving the problem. In contrast, in a fine-grain computing model, one would allow only constant local memory, but unrestrict the number of processing nodes available. In a coarse-grain model the communication between processors is done along network connections, and thus it incurs a cost that could be significantly higher than the cost of local computation.

A planar subdivision R is a partition of the plane into polygons, called the regions of R. We assume that R has one unbounded region, called the external region, and that the union of all the other regions of R, called internal regions, is a convex polygon. A subdivision R is generated by a planar graph embedded in the plane using only straight-line segments. The subdivision is a triangulation if all internal regions are triangles. We assume the subdivision R is a triangulation.

As mentioned in Section 1.1, the optimal link problem can be reduced to solving a number of at most $O(n^2)$ GOPs. Thus, in designing our algorithms we need to consider how to efficiently generate the GOPs and how to solve a given GOP once it is available. In what follows, we introduce the structures needed to generate the GOPs.

Let L be a link intersecting the source and target regions R_s and R_t. Let S be the set of line segments in the subdivision R and let $S_{st} = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\}$ be the subset of line segments in S that are intersected by L. Consider rotating and translating L. An event e_v occurs when L passes a vertex v of R. Such an event corresponds to some line segments (with an endpoint at v) entering or leaving S_{st}. As long as no event occurs, the formula describing the objective function $d(L)$ does not change and has the expression $d(L) = \sum_{i=i_1}^{i_{k-1}} w_i * d_i$, where d_i is the length of L between the two line segments s_i and s_{i+1} that are on the boundary of R_i and have non-empty intersection with L [9,16]. The possible event free movements of L define a hourglass, as shown in Fig. 2.

Let $H = \{l_1, l_2, \ldots, l_n\}$ be a set of n straight lines in the plane. The lines in H partition the plane into a subdivision, called the *arrangement* $A(H)$ of H, that consists of a set of convex regions (cells), each bounded by some line segments on the lines in H. In general, $A(H)$ consists of $O(n^2)$ faces, edges, and vertices and

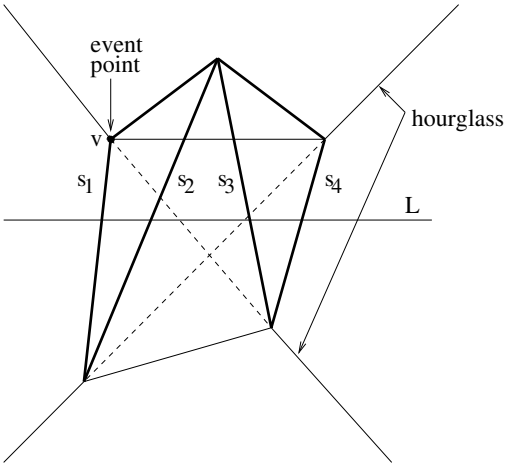


Fig. 2. A set of line segments of R (in bold) defining an hourglass (continuous line), a link L intersecting the segments s_1, s_2, s_3 and s_4, and an event point v

it can be computed in $O(n^2)$ time and $O(n)$ space, by sweeping the plane with a pseudoline [19]. If one is interested only in the part of $A(H)$ inside a convex region, similar results are possible [6,12].

Throughout the paper we refer to a standard point-line duality transform that maps a point $q = (a, b)$ to the dual line $y = ax + b$ and the non-vertical line $y = mx + p$ to the dual point $(-m, p)$. This duality transform preserves incidences and above/bellow relations (e.g., a point q above a line l dualizes to a line that is above the dual point of l). At times, we will refer to the dual plane as the (m, p) plane.

Let $A(R)$ denote the dual arrangement of R, defined by $H_R = \{l_1, l_2, \ldots, l_n\}$, where $l_i \in H_R$ is the dual of the vertex $v_i \in R$. Using the duality transform, all lines intersecting the same subset of segments $S_{st} \in S$ correspond to a cell in the dual arrangement $A(R)$. The case of a semiline, and that of a line segment, can be reduced to that of a line by appropriately maintaining the set of line segments intersected by L and dropping those that arise before a segment in R_s or after a segment in R_t. In [9], it has been shown that sequentially it is possible to obtain the set of line segments of R that are intersected by a semiline in amortized constant time if the segments intersected by the supporting line are available. Their solution can be easily extended to handle a line segment as well.

Generating and sweeping the entire arrangement $A(R)$ however, may not be efficient, since many cells of $A(R)$ may correspond to set of links that do not intersect R_s and/or R_t. As noted in [16], the lines intersecting R_s (resp., R_t), define a "strip" region D_{R_s} (resp. D_{R_t}) sandwiched between two m-monotone, unbounded and nonintersecting chains in the (m, p) plane, and the lines intersecting both R_s and R_t thus correspond to the common intersection D_{st} of D_{R_s} and D_{R_t}. D_{st} is a (possibly unbounded) region bounded by two m-monotone chains with a total of $O(k_s + k_t)$ vertices, where k_s and k_t are the number of vertices of R_s and R_t, respectively.

Lemma 1. *The lines in the set $A(R)$ have at most $O(n)$ intersections with the two chains bounding the region D_{st}.*

Proof. Only $O(1)$ lines tangent to R_s and R_t can pass through a point p. Then, the dual line of p can intersect the chains bounding D_{st} only $O(1)$ times. □

Thus, computing the cells of the arrangement defined by $A(R)$ that correspond to set of lines intersecting both R_s and R_t reduces to computing the arrangement of $O(n)$ line segments in D_{st} (some of these line segments may in fact be semilines, but this does not influence the overall computation).

Since we assume that R is triangulated and R_s and R_t are triangles, we can further simplify the problem by computing optimal links for a constant number of subproblems, where each subproblem corresponds to a pair of edges of R_s and R_t. Thus, from now on, we assume that R_s and R_t are line segments (edges) of R. Then, D_{st} is obtained by intersecting two double wedges in the dual plane, has constant complexity and can be computed in constant time. Without loss of generality we assume that D_{st} is a connected convex region of the dual plane. Note that it is possible D_{st} consists of two connected convex regions, in which case we threat each region separately.

3 Parallel Solutions

In this section we present two parallel solutions for the weighted region optimal link problem. The first algorithm uses the CREW PRAM model of computation. In this model processors act synchronously and may simultaneously access for reading the same memory location on a shared memory space. The second algorithm uses a coarse-grain parallel computer model of computation. In this model, a relatively small number of processors are available and each processor has a large amount of local memory, thus being able to store all data involved in (sequentially) solving the problem, much like a personal computer. In particular, such a processing element would be able to store the region R and its dual arrangement, as well as all data that is required in the process of generating and solving a GOP.

3.1 The CREW PRAM Algorithm

In this section we give an output sensitive CREW PRAM solution for generating the set of GOPs. To obtain an output sensitive algorithm, we use the paradigm in [22]: the pool of virtual processors can grow as the computation proceeds, provided that the allocation occurs globally [21]. Given a subdivision R with a total of n vertices, the algorithm we present runs in $O(\log n)$ time using $O(n \log n + k)$ processors, where k is the size of the output (the total description complexity for the feasible domains of the GOPs to be solved), and it could be $\Omega(n^2)$ in the worst case. If the traditional CREW PRAM model is used, our solution would require $O(n^2)$ processors.

As outlined in the previous section, to compute the feasible domains for the GOPs it suffices to compute the cells in the arrangement $A(D_{st})$ of $O(n)$ line segments in D_{st}, where each line segment has its endpoints on the boundary of D_{st}. Further, in order to produce the corresponding objective functions, with each cell C of $A(D_{st})$ we must associate the subset of line segments in S that are intersected by a line whose dual is a point in C. This computation may be regarded as a set of queries on the line segments in S.

The algorithm we present follows the one in [22], where the following segment intersection problem has been considered and solved: given a set of line segments in the plane, construct a data structure that allows to quickly report the segments intersected by a query line. Their algorithm is based on a parallel persistence data structure termed *array-of-trees* and on fast construction of line arrangements. The main idea in [22] is to build the arrangement, an operation sequence σ for that arrangement, and then use the array-of-trees data structure to evaluate the sequence. A reporting query can then be answered in $O(\log n)$ time per query, resulting in an $O(\log n)$ time, $O(n^2)$ processors CREW PRAM algorithm.

The main difference in the algorithm we present is in defining and handling the operation sequence σ. Given the nature of the optimal link problem, a vertex of the subdivision R may in fact be the endpoint of multiple line segments (e.g., $O(n)$ such segments). Then, while crossing from one cell to an adjacent one, many line segments may enter or leave the set S_{st} and thus many *enable/disable*-like operations in [22] would be associated to such crossing. Rather than defining the enable/disable operations on individual segments, we define these operations on subsets of segments in S.

In order to maintain the processing bounds, we must be able to obtain these subsets in constant time per subset. As in [9], we assume that for each vertex $v \in R$, the vertices adjacent to v (and thus the corresponding edges) are in sorted angular order in the adjacency list of v. If not given as part of the input, this can be easily done in parallel in $O(\log n)$ time using $O(n)$ processors. Let v_s be the list of segments in S_{st} that are incident to v. We also maintain the following information: for each $v \in R$, a list for the segments in v_s and the extreme segments in v_s (based on the angular order); for each edge $\overline{vw} \in S_{st}$, two pointers for its positions in v_s and w_s, respectively.

Let $d(v)$ denote the number of vertices adjacent to a vertex v in R. Since R is a planar subdivision, we have $\sum_{v \in R} d(v) = O(n)$.

Lemma 2. *If the set of line segments intersected by L and corresponding to some cell C_1 in the dual plane is available, then the set of line segments corresponding to an adjacent cell C_2 can be obtained in $O(1)$ time using $O(d(v))$ processors, where v is the dual of the line shared by C_1 and C_2.*

Proof. Let e be the common edge of C_1 and C_2. Then e corresponds to a set of lines passing through v. Crossing e from C_1 to C_2 corresponds to deleting v_s from S_{st} and inserting in S_{st} the edges adjacent to v that are not in v_s. This results in $O(d(v))$ updates in S_{st}. Noting that the two sets above consist of consecutive edges and each delete/insert operation takes constant time, we can assign $d(v)$ processors to perform the $O(d_v)$ delete/insert operations for S_{st}.

Since the first and last entries in the list (for the extreme segments) are known, the processor assignment can be done in constant time. We first produce the list of line segments to be deleted from S_{st}. The list is obtained by having each processor copying an edge of the list while updating adjacency list information for its neighbors, which requires $O(1)$ time. The list corresponding to consecutive insert operations is produced similarly. After the two lists are produced, we can delete the old list and insert the new one in constant time with a single processor. Further, for each deleted segment $\overline{vw}$, w_s can be updated in constant time using the information stored with $\overline{vw}$, and for each inserted segment $\overline{vu}$, u_s can be updated in constant time by inserting $\overline{vu}$ in u_s. Each updating operation takes constant time and it is assigned to one processor resulting in $O(d(v))$ total work. □

Theorem 1. *The feasible domains and the objective functions for the GOPs associated with the region D_{st} can be generated in $O(\log n)$ time using $O(n \log n + k)$ processors, where k is the size of the output.*

Proof. We give an algorithm that constructs the GOPs in the claimed time and processor bounds. The algorithm proceeds as follows. (1) Construct the arrangement of line segments inside D_{st}. This can be done in $O(\log n)$ time with $O(n \log n + k)$ processors, using the algorithm in [21]. We then compute a spanning tree for this arrangement and an Euler tour of this tree, as in [22]. While computing the Euler tour, we use the data structures in [9] to produce the operation sequence σ for the tour. Since the enable/disable operations in σ add only constant time, this computation can still be done in $O(\log n)$ time using $O(k/\log n)$ processors. Constructing the array-of-trees data structure and answering reporting queries can be done as in [22]. Then, the claimed processing bounds follow. □

We mention here that an $O(\log n)$ time, $O(n^2)$ processors algorithm can be obtained by associating an enable/disable operation with each line segment involved in a crossing at a node v (i.e., to $O(d(v))$ segments) and applying the algorithm in [22].

3.2 A Coarse-Grain Parallel Algorithm

In this section we consider a coarse-grain parallel computer model of computation. If at most n processors are available, we present a simple yet efficient algorithm that generates all GOPs using $O(n^2 \log n)$ work and with practically no communications between processors. The GOPs can be solved locally or they can be sent for solving to some external processing clusters. After all GOPs are solved, the optimal solution can be obtained by a simple minimum selection.

We make the following assumptions for our model: (1) processors are connected and can communicate via a global data buss or a communication network that allows efficient data broadcasting (i.e, feed the subdivision R to all processing elements) and (2) processors are numbered and each processor knows its order number. The algorithm we present is more general as it is based on

computing the portion of an arrangement of lines that lies in between two vertical lines. We will show at the end of this section how to simplify it to address our specific problem.

At the start of the algorithm, each processing element stores the subdivision R and the set of lines in $A(R)$ (following a broadcasting operation), and knows its order number. Since each processor will perform similar computation, it then suffices to discuss the computation involved at only one of them, say the k-th processor P_k.

At processor P_k, the algorithm computes the GOPs associated with the portion of the arrangement $A(R)$ that is in between the vertical lines L_{k-1} and L_k passing through the $(k-1)n$-th and kn-th leftmost intersection points of the lines in $A(R)$. We denote these two points as p_{k-1} and p_k. First, the algorithm finds the lines L_{k-1} and L_k by computing the points p_{k-1} and p_k. These points can be computed in $O(n \log n)$ time each using the algorithm in [14]. Next, the algorithm computes the intersection points of the lines in $A(R)$ with L_{k-1} and L_k and runs a topological sweep algorithm [6,12] to produce the GOPs inside the parallel strip. Sweeping the strip, as well as generating the corresponding objective functions, can be done altogether in $O(n \log n)$ time, which follows from [9].

Alternatively, we can obtain the same results using the (optimal) sequential version of our CREW PRAM algorithm (i.e., by computing a line segment arrangement inside the strip and traversing that arrangement). After the GOPs are solved, the last step of the algorithm consists of a minimum selection among the optimal solutions stored "locally" at different processing elements, in order to obtain the minimum over all GOPs. This can be done with only $O(n)$ communication, starting at processor P_1 and with the overall minimum computed at processor P_n.

Thus, we have the following results.

Theorem 2. *In the proposed coarse-grain computing model, the feasible domains and the objective functions for the GOPs can be computed in $O(n \log n)$ time using $O(n)$ processors.*

Corollary 1. *If only p processors are available, where $p \leq n$, the feasible domains and the objective functions for the GOPs can be computed with $O(n^2 \log n)$ total work.*

At this point we recall from Section 1.1 that it is known that an optimal solution goes through a vertex of the subdivision R. Thus, rather than generating the arrangement of lines, we only need to obtain its edges. This can be done directly in the original plane as follows. For each vertex $v \in R$, compute the intersection of the double wedges W_s and W_t at v, defined by the tangent lines from v to R_s and R_t, respectively. This takes constant time for a vertex v. For each W_v, find the set of vertices $VW(v)$ in $R \cap W_v$ by traversing the portion of R that lies between the two bounding lines of W_v, which takes linear time in the number of

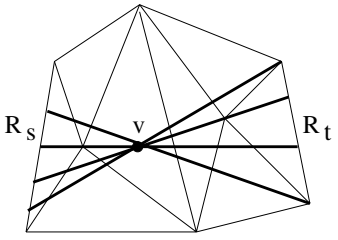


Fig. 3. Three double wedges (bold line) at vertex v, when L is a line segment (case (c))

vertices $|VW(v)|$ in the set ($O(n)$ in the worst case). Then sort the set $VW(v)$ around v in $O(n \log n)$ time and compute the intersection of the lines defined by v and the vertices in $VW(v)$ with the boundaries of R_s and R_t. This results in a set of $O(n)$ double wedges at v, defining the feasible domains for the $O(n)$ possible GOPs at v [16] (see Fig. 3). Finally, compute the objective function associated with the leftmost double wedge at v, then traverse the remaining double wedges at v while updating the objective function in constant time, as described earlier. The computation at v can be carried out in $O(n \log n)$ time [16]. Assigning a processor for each $v \in R$ results in $O(n^2 \log n)$ total work.

Lemma 3. *The simplified algorithm above constructs the global optimization problems for finding an optimal link in $O(n \log n)$ time using $O(n)$ processors.*

Proof. Follows from the algorithm description. □

Some of the advantages of this algorithm over the previous one are that it avoids using duality transforms and (line segment) arrangement computation for generating the optimization problems, which may lead to more robust implementations. It can also be easily adapted to generate only a subset of the optimization problems which may be useful in some applications. Observe that for case (c) of the problem we only need consider the vertices of R that are inside the convex hull of R_s and R_t when generating the subproblems. This subset can be easily obtained in $O(n)$ time using a single processor [16].

There are two important features of our solutions that should be noted here. First, the approaches we propose allow for scalability in solving the GOPs. That is, after a GOP is produced, it can be solved either locally or it can be sent to some external processing cluster, that would in turn compute and return the optimal value for that GOP. Second, once the initial setup for the computation has been completed, it takes constant time to generate a new GOP; since the objective function of a GOP could have $O(n)$ terms, this implies that all GOPs in a strip can be generated in time comparable to that required to perform a single evaluation of a GOP's objective function, and justifies the proposed coarse-grain model of computation.

4 A Parallel Algorithm for Solving a GOP

While the GOPs can be generated fast in a coarse grained parallel computer model, it would also be desirable to solve a GOP in parallel. Thus, we need consider global optimization algorithms that are easily parallelizable.

To solve a GOP in parallel we adapt an algorithm for minimizing a sum of linear fractional functions (SOLF) [20], that has inherent parallelism, to solve the specific GOPs associated with the weighted region optimal link problem.

In [16], it has been shown that since an optimal link goes through a vertex of R the 1-dimensional objective function has the generic form

$$\min_{x\in U}\{d(x)=\sqrt{1+x^2}\sum_{i=1}^{m}\frac{a_i}{b_ix+c_i}\}=\min_{x\in U}\{\sum_{i=1}^{m}\sqrt{1+x^2}\frac{a_i}{b_ix+c_i}\}=\min_{x\in U}\{\sum_{i=1}^{m}r_i(x)\}$$

where a_i, b_i and c_i are constants and $b_ix+c_i>0$ over the feasible domain U, for $i=1,2,\ldots m$.

Thus, the functions we try to minimize are 1-dimensional sum of fractional functions (SOFs) with generic term $r_i(x)=\sqrt{1+x^2}\frac{a_i}{b_ix+c_i}$.

In contrast, a SOLF is a sum of fractional terms, each of which is a ratio of two linear functions. Thus, in a 1-dimensional SOLF the generic term has the form $r_i(x)=\frac{a_i}{x+c_i}$.

Consider minimizing $d(x)=\sum_{i=1}^{m}r_i(x)$, over an interval U. The main steps of the SOLF algorithm [20] are described below (see also [10] for the 1-dimensional SOLF algorithm).

*S*tep 1. Set the step variable, k, equal to zero. Determine an initial lower bound $lb=(lb_1,\ldots,lb_m)$ on the optimal solution for $d(x)$, where we have $lb_i=$ minimize $\{r_i(x)|x\in U\}$, with solutions x^i, for $i=1,\ldots,m$. (This solution gives the minimum value of a single term $r_i(x)$ over U).

*S*tep 2. Compute a set P of m feasible points

$$P=\begin{bmatrix}p_{1,1}&\cdots&p_{1,m}\\&\cdots&\\p_{m,1}&\cdots&p_{m,m}\end{bmatrix},$$

where $p_{i,j}=r_j(x^i)$ for $i=1,\ldots,m$ and $j=1,\ldots,m$. Then compute an isovalue contour f_u defining an upper bound, where

$$f_u=\min\{\sum_{j=1}^{m}p_{i,j},i=1,\ldots,m\}.$$

*S*tep 3. Determine the remaining points of the initial m-simplex:

$$l=\begin{bmatrix}f_u-\sum_{i=2}^{m}lb_i&lb_2&&\ldots&lb_j&\ldots&lb_m\\\ldots&&&\ldots&&&\ldots\\lb_1&\ldots&lb_{j-1}&f_u-\sum_{i\neq j}lb_i&lb_{j+1}&\ldots&lb_m\\\ldots&&&\ldots&&&\ldots\\lb_1&\ldots&lb_j&\ldots&&lb_{m-1}&f_u-\sum_{i=1}^{m-1}lb_i\end{bmatrix},$$

*S*tep 4. Let $k = k + 1$. Solve the m problems:

$$\begin{gathered}\text{minimize } r_1(x)\\ \text{subject to } x \in U \text{ and } t_i \geq r_i(x), i = 2, \ldots, m,\\ \ldots,\\ \text{minimize } r_m(x)\\ \text{subject to } x \in U \text{ and } t_i \geq r_i(x), i = 1, \ldots, m-1,\end{gathered}$$

where $t_i = l_{i,i}$. Let x^i, $i = 1, 2, \ldots, m$, be the solutions to these problems. The new lower bound point lb is the m dimensional point of coordinates $lb_i = r_i(x^i)$, for $i = 1, \ldots, m$.

*S*tep 5. Determine the new m-simplex based on the new lower bound.

$$l = \begin{bmatrix} f_u - \sum_{i=2}^{m} lb_i & lb_2 & \ldots & lb_j & \ldots & lb_m \\ \ldots & & \ldots & & & \ldots \\ lb_1 & \ldots\ lb_{j-1} & f_u - \sum_{i \neq j} lb_i & lb_{j+1} & \ldots & lb_m \\ \ldots & & \ldots & & & \ldots \\ lb_1 & \ldots\ lb_j & \ldots & & lb_{m-1} & f_u - \sum_{i=1}^{m-1} lb_i \end{bmatrix},$$

*S*tep 6. If $lb = l_j$ for any j, where $l_j = (l_{j,1}, \ldots, l_{j,m})$ is a row in the matrix l above, then stop. The optimal value of the objective function is $\sum_{i=1}^{m} l_{j,i}$ and the global optimal solution is x^* such that $r_i(x^*) = l_{j,i}$, for $i = 1, \ldots, m$.

*S*tep 7. If $lb^k \neq lb^{k-1}$, then return to Step 4.

*S*tep 8. Otherwise, the current upper bound lb is the same as the upper bound in the previous iteration and the iterative process has stalled. In this case, use a hyperplane to split the m-simplex containing the optimal solution in half. Relabel f_u as f'_u. Repeat steps 1 and 2 once for each subregion. In step 1, add appropriate constraints to restrict the search to a particular subregion of the m-simplex. In step 2, relabel f_u as f_u^1 and f_u^2 for the first and second subregion, respectively, and let $f_u = \min\{f_u^1, f_u^2\}$. If $f_u < f'_u$, then return to step 3 with the new lb. Otherwise, $f_u \geq f'_u$, so execute steps 3 through 9 in each of the two subregions of the simplex.

The algorithm terminates in step 6 or when the difference between the lower and upper bounds is below a user specified threshold.

The only place in the algorithm above where the expression of the ratio $r_i(x)$ is important is in the optimization subproblems in steps 1 and 4. For the SOF problems associated with a GOP the optimization subproblems in steps 1 and 4 can be solved in constant time each and thus we can apply the algorithm for these SOFs. We refer to the resulting algorithm as the SOF algorithm. Steps one, two and eight correspond to the initialization part while steps three to seven of the algorithm correspond to the iteration part.

Lemma 4. *In the SOF algorithm the initialization part can be performed in $O(m)$ time using $O(m)$ processors and the iteration part can be performed with $O(m)$ total work.*

Proof. The initial lower bound lb for Step 1 can be computed with $O(m)$ work ($O(1)$ time with m processors, or $O(m)$ time with a single processor). Step 2 requires to evaluate m ratios at m points and takes $O(m)$ time with m processors. Computing the $O(m)$ sums take $O(m)$ time with m processors, where the i-th processor computes the i-th summation. After that, the upper bound can be found in $O(m)$ time with a single processor. In Step 8, splitting the m-simplex in half requires $O(m)$ time sequentially and thus can be done by a single processor. This concludes the initialization part.

The iteration part can be performed sequentially, using a single processor, as in [10]. In Steps 3 and 5 use an implicit representation of the m simplex points, since each simplex point differs from the lower bound in only one dimension. After computing $\sum_{i=1}^{m} lb_i$ with $O(m)$ work, the m simplex points can be obtained within the same bound. Step 6 can be performed in a similar way with $O(m)$ total work. Step 7 is trivial and requires $O(1)$ work. Thus, the total work for the iteration part can be carried out with $O(m)$ work. □

From Lemma 4 it follows that we can assign a cluster of $O(m)$ processing nodes to handle a GOP. Each iteration of the SOF algorithm would be executed in $O(m)$ time by a single processor, while the other processors are needed to ensure that a stall can also be handled in $O(m)$ time.

Assuming the SOF algorithm makes progress and returns to Step 3 after each stall in Step 8, the time to solve a GOP becomes $O(m\mathcal{I})$, where $\mathcal{I}$ is the number of iteration to obtain an (approximate) optimal solution. If no progress is made after a stall the two resulting optimization subproblems are executed on two distinct processing nodes.

5 Conclusions

In this paper we have presented parallel CREW PRAM and coarse grained computer algorithms for computing an optimal link among weighted regions in the plane. Given a weighted, triangulated subdivision with a total of n vertices, the *work* of the parallel algorithms we propose is only an $O(\log n)$ factor more than that of their (optimal) sequential counterparts. After proving that GOPs can be generated fast in a coarse grained parallel computer model, we have also shown how to efficiently solve a GOP in parallel by adapting an algorithm for minimizing a sum of linear fractional functions, that has inherent parallelism, to solve the specific optimization subproblems associated with the weighted region optimal link problem.

Our coarse grained parallel computer algorithms are simple and can be easily implemented. They are targeted towards applications in radiation therapy and stereotactic brain surgery, where finding the optimal choice for the link fast and accurate is a key problem in treatment optimization.

References

1. Aleksandrov, L., Lanthier, M., Maheshwari, A., Sack, J.-R.: An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces. In: Arnborg, S. (ed.) SWAT 1998. LNCS, vol. 1432, pp. 11–22. Springer, Heidelberg (1998)
2. Aleksandrov, L., Maheshwari, A., Sack, J.-R.: Determining approximate shortest paths on weighted polyhedral surfaces. Journal of the ACM 52(1), 25–53 (2005)
3. Aleksandrov, L., Maheshwari, A., Sack, J.-R.: Approximation algorithms for geometric shortest path problems. In: Proceedings of the 32nd annual ACM Symposium on Theory of Computing, pp. 286–295 (2000)
4. Alsuwaiyel, M.H., Lee, D.T.: Minimal Visibility Paths Inside a Simple Polygon. Computational Geometry: Theory and Applications 3, 1–25 (1993)
5. Alsuwaiyel, M.H., Lee, D.T.: Finding an Approximate Minimum-Link Visibility Paths Inside a Simple Polygon. Information processing letters 55, 75–79 (1995)
6. Asano, T., Guibas, L.J., Tokuyama, T.: Walking in an arrangement topologically. Int. Journal of Computational Geometry and Applications 4, 123–151 (1994)
7. Brahme, A.: Optimization of radiation therapy. Int. Jouurnal of Radiat. Oncol. Biol. Phys. 28, 785–787 (1994)
8. Chazelle, B., Edelsbrunner, H.: An optimal algorithm for intersecting line segments in the plane. Journal of the ACM 39, 1–54 (1992)
9. Chen, D.Z., Daescu, O., Hu, X., Wu, X., Xu, J.: Determining an optimal penetration among weighted regions in two and three dimensions. Journal of Combinatorial Optimization 5(1), 59–79 (2001)
10. Chen, D.Z., Daescu, O., Dai, Y., Katoh, N., Wu, X., Xu, J.: Optimizing the sum of linear fractional functions and applications. Journal of Combinatorial Optimization 9(1), 69–90 (2005)
11. Chen, D.Z., Hu, X., Xu, J.: Optimal Beam Penetration in Two and Three Dimensions. Journal of Combinatorial Optimization 7(2), 111–136 (2003)
12. Chen, D.Z., Luan, S., Xu, J.: Topological Peeling and Applications. Int. J. Comput. Geometry Appl. 13(2), 135–172 (2003)
13. Cheng, S.-W., Na, H.-S., Vigneron, A., Wang, Y.: Approximate Shortest Paths in Anisotropic Regions. In: Proc. ACM-SIAM Symposium on Discrete Algorithms, pp. 766–774 (2007)
14. Cole, R., Salowe, J., Steiger, W., Szemeredi, E.: Optimal Slope Selection. SIAM Journal of Computing 18, 792–810 (1989)
15. Daescu, O.: Parallel Optimal Weighted Links. In: Alexandrov, V.N., Dongarra, J., Juliano, B.A., Renner, R.S., Tan, C.J.K. (eds.) ICCS 2001. LNCS, vol. 2073, pp. 649–657. Springer, Heidelberg (2001)
16. Daescu, O., Palmer, J.D.: Minimum Separation in Weighted Subdivisions (manuscript, 2003)
17. Daescu, O., Mitchell, J.S.B., Ntafos, S., Palmer, J.D., Yap, C.K.: k-Link Shortest Paths in Weighted Subdivisions. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) WADS 2005. LNCS, vol. 3608, pp. 325–337. Springer, Heidelberg (2005)
18. Daescu, O., Mitchell, J.S.B., Ntafos, S., Palmer, J.D., Yap, C.K.: An Experimental Study of Weighted k-Link Shortest Path Algorithms. In: Proceedings of the 7th International Workshop on the Algorithmic Foundations of Robotics (2006)
19. Edelsbrunner, H., Guibas, L.J.: Topologically sweeping an arrangement. Journal of Computer and System Sciences 38, 165–194 (1989)
20. Falk, J.E., Palocsay, S.W.: Optimizing the Sum of Linear Fractional Functions. In: Floudas, C.A., Pardalos, P.M. (eds.) Collection: Recent Advances in Global Optimization, pp. 221–258 (1992)

21. Goodrich, M.: Intersecting Line Segments in Parallel with an Output-Sensitive Number of Processors. SIAM Journal on Computing 20, 737–755 (1991)
22. Goodrich, M., Ghouse, M.R., Bright, J.: Sweep methods for Parallel Computational Geometry. Algorithmica 15, 126–153 (1996)
23. Lanthier, M., Maheshwari, A., Sack, J.-R.: Approximating weighted shortest paths on polyhedral surfaces. Algorithmica 30(4), 527–562 (2001)
24. Maheshwari, A., Sack, J.R., Djidjev, H.: Link Distance Problems. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Comput. Geom., pp. 519–558. Elsevier Science, Amsterdam (2000)
25. Mata, C., Mitchell, J.S.B.: A new algorithm for computing shortest paths in weighted planar subdivisions. In: Proc. ACM Symp. Comput. Geom., pp. 264–273 (1997)
26. Mitchell, J.S.B.: Geometric Shortest Paths and Network Optimization. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Comput. Geom. Elsevier Science, Amsterdam (2000)
27. Mitchell, J.S.B., Papadimitriou, C.H.: The weighted region problem: Finding shortest paths through a weighted planar subdivision. J. of the ACM 38, 18–73 (1991)
28. Mitchell, J.S.B., Piatko, C., Arkin, E.M.: Computing a shortest k-link path in a polygon. In: Proc. 33rd Annual IEEE Sympos. Found. Comput. Sci., pp. 573–582 (1992)
29. Piatko, C.: Geometric bicriteria optimal path problems. PhD thesis, Cornell U (1993)
30. Reif, J.H., Sun, Z.: An efficient approximation algorithm for weighted-region shortest-path problem. In: Proc. of 4th Workshop on Algorithmic Found. of Robot., pp. 191–203 (2000)
31. Sun, Z., Reif, J.H.: On finding approximate optimal paths in weighted regions. J. Algorithms 58(1), 1–32 (2006)

Parallel Simulation of Oil Reservoirs on a Multi-core Stream Computer

Fadi N. Sibai and Hashir Karim Kidwai

P.O. BOX 17555, College of Information Technology, UAE University,
Alain, United Arab Emirates
{fadi.sibai,hkidwai}@uaeu.ac.ae

Abstract. With the oil barrel price presently crippling the world economy, developing fast oil reservoir simulators is as important as ever. This article describes the parallelization and development of a 2-phase oil-water reservoir simulator on the state-of-the-art IBM Cell computer. The interdependent linear algebraic equations of the reservoir simulator is presented as well as the pipelined time step parallelization approach adopted on the Cell, The performance results reveal that given the largely interdependent nature of the oil reservoir model equations which highly limits parallelism, speedups of 6x or higher could be obtained. This speedup is significant as it results in oil simulation runs cut from weeks to days, allowing for more simulation runs with various well placements to run on the same hardware, and resulting in better reservoir management, and possibly higher oil production. The results also demonstrate that the oil reservoir simulator application is characterized by higher speedups with increasing grid size. However the speedup was shown to go down with increased number of time steps as the main memory transfer overhead becomes an important factor. Proper choice of compiler optimization flags helped boost the performance by a factor of 2x. Our parallelization approach is economically feasible due to the affordable cost of the widely available Cell-based Playstation 3.

Keywords: Oil reservoir simulation, multi-core computing, Cell Broadband Engine, Thomas algorithm.

1 Introduction

With the price of an oil barrel and demand for oil reaching record levels, the importance of effective methodologies and tools for oil exploration, extraction, and production has soared. Oil reservoir simulators [1-6] are crucial to the production of oil, as they help in oil reservoir forecasting, sensitivity analysis, and history matching. As the cost of drilling wells is very high reaching in some cases tens of millions of US dollars [7], simulating oil reser-voirs with virtual wells is necessary. Simulation helps evaluating models for the injection rate, places on the well for injection, and recovery techniques, all crucial for the oil recovery's financial success. After the best placement of wells is done, a reservoir simulator can be used to extend the life of the oil flow plateau. Reservoir simulation is thus a key component of oil reservoir

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 82–98, 2009.

management. The oil reservoir management process repeats the 2 following cycles: i. detects data changes during production and from that, detects reservoir changes; ii. Enter the data and reservoir changes into the dynamic reservoir model, and conduct reservoir simulations to guide future production and data ac-quisition plans.

Many oil reservoir simulators were developed by oil or oil-related companies such as Exxon Mobil, Shell, Amoco, Schlumberger, Saudi Aramco, just to name a few. However none of these implementations are be-lieved to be for the IBM Cell processor [8-11], a recent state of the art processor that is mass produced and is an integral part of the Sony Playstation 3.

The Cell processor is a multi-core processor with 9 cores on one single chip package. With its multi-core parallelism and vector and SIMD processing capabilities, the IBM Cell processor was shown to deliver top computation performance levels compared to other CPUs on graphics and image processing applications [12] and other integer applications. While there have been several implementations of oil reservoir simulators on clusters [13-17] and grid computers [18], we believe that our work is the first implementation of an oil reservoir simulator on this state of the art processor. Because of the Cell's unique architecture, shorter simulator run times are expected on the Cell platform compared to existing systems. Such performance gains can result in faster solutions (in terms of days, depending on the size of the problem). Faster solutions means that the simulation runs are shorter in time, and that more simulation runs with a larger variety of virtual well or other parameter settings can be made in fixed time duration, resulting in better reservoir management and forecasting. Saudi Aramco [13] projects tremendous growths in number of models and in computer capacity requirements for reservoir simulation, both stressing the need for faster and more powerful multiprocessors such as the Cell.

In this paper, we describe the parallelization and implementation of a parallel 1D oil-water reservoir on the IBM Cell Broadband Engine. In Section 2, we review the stream model of computation on which the Cell is based and highlight the Cell's features. In Section 3, we describe the oil reservoir's partial differential equation model, followed by the reservoir's numerical model based on difference equations in Section 4. In Section 5, we present our parallel computation methodology. Implementation details and performance results are then presented in Sections 6 and 7, respectively. Conclusions are drawn in Section 8.

2 The Cell and the Stream Model of Computation

The IBM cell processor is the first commercial proces-sor based on the stream model of computation. Earlier, the IMAGINE processor is based on a stream processor architecture developed by a team of Stanford University researchers. Multimedia and graphics applications are examples of applications suitable for the stream model of computation. In general, applications with large data sets friendly to vector processing will perform well under this model of computation.

The steam model of computation is based on the concepts of streams and kernels. A stream is a set of sequential data requiring similar operations. A stream is created by appending data elements to the tail of the stream. A stream is consumed by popping data elements from its head. Kernels are short programs which take one or

more streams for input and generate an output stream as a result of the execution of the kernel instructions on the input streams' data elements.

Stream processors perform well on media, graphics, and applications with large data sets requiring the execution of similar operations on their data elements such as in vector processing applications because they exchange large portions of superscalar CPU die used to implement out of order instruction execution and large centralized on-chip caches for large numbers of SIMD execution units, large register files, and smaller memory units distributed among the processor's processing elements. Stream processors require smaller memory areas distributed among the processors which hold portions of the code and data as they take advantage of the data locality property of multimedia streams. Furthermore, large cache memories between communicating stream processors are not needed and stream processors' processing elements rarely access main memory to get hold of sparse data. Without various levels of large caches and with local memory serving the computation needs of the processing elements, the L0 cache to L1 cache to L2 cache to main memory latency observed on traditional CPUs is avoided on stream architectures. Various types of parallelisms are exploited by stream processors. Thread level parallelism is achieved for instance on the Cell by running multiple threads on the host RISC processor and another thread on processing elements. The stream processor's many processing elements with distributed local memories facilitate data parallelism. Instruction level parallelism is achieved in the cores' pipelines.

The Cell processor adopts the stream computing model. The Cell is a heterogeneous multi-core chip containing one 64-bit PowerPC Processing Element (PPE), eight specialized co-processors called Synergistic Processing Elements (SPE), and one internal high speed bus called Element Interconnect Bus (EIB) which links PPE and SPEs together. The PPE is itself composed of an L2 cache and a Power Processor Unit (PPU) containing a Power execution unit and an L1 cache. The PPE supports the VMX (AltiVec) vector instruction set to parallelize arithmetic operations. Each SPE contains a Synergistic Processing Unit (SPU), and a SMF unit (including DMA, a memory management unit, and bus interface). Each SPU contains a RISC processor with 128-bit SIMD single and double precision instructions, and a 256 KB instruction and data local memory area (known as the local store or LS) which is visible to the PPE and can be addressed directly by software. The local store does not operate like a superscalar CPU cache since it is neither transparent to software nor contains hardware structures that predict what data to load. The Cell can handle 10 simultaneous threads and over 128 outstanding memory requests. Each SPE can communicate 16 Bytes per cycle to the EIB which has a bandwidth of 96 Bytes per cycle (< 8 x 16=128B/cycle). The Memory Interface Controller and Bus Interface Controller which interface to external memory and I/O peripherals, respectively, also communicate with the EIB at a rate of 16 Bytes/cycle. The Power execu-tion Unit-L1 cache bandwidth is also 16 Bytes/cyle while the L1 cache-L2 cache bandwidth is 32 Bytes/cycle.

3 The Oil Reservoir's PDE Model

The development of the parallel partial differential equation (PDE) reservoir simulation model involves the following steps.

i. Defining and refining a parallel partial diffe-rential equation (PDE) model, and identifying performance-efficient numerical methods [19] and programming techniques [20]. For a 2-phase oil/water reservoir model, the main equations include the oil and water equations, with oil pressure and water saturation being the primary unknowns, and water pressure and oil saturation being the secondary un-known. Many of the coefficients in these equations are also function of these unknown variables.
ii. Dividing the reservoir into grids, discretizing th model in space and time, and parallelizing the model;
iii. Coding the PDE model in C and parallelizing the code;
iv. Debugging this model;
v. Fine tuning and optimizing the performance of this parallel simulator;
vi. Testing simulator by comparing its results of a known reservoir to known results collected from another proven simulator.

The oil reservoir simulation takes for input an accurate geological model of the oil field to be simulated, where the reservoir and its boundaries are clearly defined and rock properties including porosity and permeability are modeled, and well production and injection detailed information. We consider a heterogeneous 1D 2-phase Oil-Water reservoir that is surrounded by impermeable rocks. The reservoir simulation model is based on the following equations that model the 2-phase flow. The differential equations modeling two-phase flow can be derived from the continuity equation (eq. 1) of each phase a which describe the conservation of mass of each component (o for oil and w for water), and Darcy's law that relates the phase velocities Ua to the gradient of the phase pressures Pa (eq. 2).

$$\frac{\partial(\phi_a \rho_a S_a)}{\partial t} + \nabla . U_a = q \tag{1}$$

$$U_a = -\frac{K_a S_a K}{\mu_a}(\rho_a \nabla P_a - \rho_a g \nabla H) \tag{2}$$

$$\sum S_a = 1 \tag{3}$$

Where U_a is the phase velocity, P_a is the phase pressure, ϕ is the porosity, ρ_a S_a is the concentration of component a, S_a is the saturation of phase a, q is the well production or injection rate (where at an injection well, q is non-negative, while at a production well, q is non-positive), K is the rock permeability tensor, $\lambda_a = K_a S_a$, ρ_a / μ_a is the mobility of phase a, ρ_a is the phase density, $g \nabla H$ is the gravity acceleration vector times the elevation, the relative permeability K_a S_a ρ_a models the reduced permeability of one phase due to the presence of the other, and μ_a is the phase viscosity. It is assumed that the phases (a) are oil (o) and water (w) and that the two phases together fill the void space completely so that $S_o + S_w = 1$ as indicated by equation 3 above. Moreover, the phase pressures are related in terms of the capillary pressure $pcow = po - pw$, which we assume to be a known function of water saturation, $pcow(Sw)$.

$$\text{po} - \text{pw} = \text{pcow(Sw)} \tag{4}$$

Equations 1-3 define the PDE model of step i. In step ii, this PDE model is discretized in space and in time resulting in a set of non-linearized finite difference equations which are then linearized. The resulting linearized equations corresponding to Equations 1 and 2 are solved at each discrete time step. The linearized finite difference equation corresponding to equation 2 may have its own time step which is adjusted according to the stability requirements. It is known that a time step of 1 day leads to stable results.

The solution method for solving the set of linearized finite difference equations which we use is the Implicit Pressure Explicit Saturation (IMPES) method which combines the differential equations such that the saturation derivatives are eliminated. As a result, this method combines the equations into a single one in which only one of the phase's pressure is the unknown variable. The resulting linear algebraic equations have coefficients which are a function of the oil or water pressures and saturations and can thus be computed by this method using the pressure and saturation values computed in the previous time iteration. At any time step, when one phase's pressure is solved, the same phase's saturation is obtained from the phase's flow partial differential equation. The other phase's pressure is then obtained from the capillary pressure equation (4). Afterwards, the other phase's saturation is obtained from equation 3 which reduces in our case to So+ Sw=1. These steps are repeated in the next time iterations setting the current pressures and saturations to the values obtained in the previous iteration and solving for the next time step's pressures and saturations until the results converge or until a desired time step is reached.

The equations are typically put in matrix form **A x X = C** where the X variables in the X matrix are the unknown variables representing the set of linear algebraic equations, and various solvers can be used to obtain a solution. The coefficient matrix **A** is usually sparse with one main diagonal and some other diagonals. The reservoir's grid blocks can be ordered differently to obtain various diagonal shapes in the **A** coefficient matrix, as various solvers require the coefficient matrix to be in some certain shape in order to properly work. Grid block ordering affects the storage and computation requirements of simulators. An example of grid ordering is red-black ordering. Solvers fall into direct and iterative categories. The direct ones are faster for small reservoirs at the expense of larger storage costs and result in more accurate solutions. The iterative methods are faster for larger reservoirs requiring less storage than their direct counterparts but provide an approximate solution. In direct solvers such as Thomas algorithm and Gaussian elimination, an exact solution is obtained. Thomas algorithm is superior to other direct solvers as it is based on a tridiagonal coefficient matrix which bypasses computation when the matrix coefficients are zero thus resulting in time savings. The Thomas algorithm has a forward elimination step and a backward substitution step.

A parallel pipelined version of Thomas Algorithm exists and is known as the Immediate Backward Pipelined Thomas Algorithm (IB-PTA) and was developed by Povitsky [21] at NASA allowing the backward step to proceed immediately after the forward step for each line (equation). It allows the completion of some lines by the backward step (and thus variables are solved) immediately after the first lines have completed their forward step, and processors switch between the forward and backward step and communicate with their neighbors to get data. The algorithm prevents some processors from waiting and idling, thus allowing the processor to compute some useful work instead of idling.

4 Numeric Reservoir Model

Fig. 1 shows a homogeneous 1D 2-phase oil-water reservoir with n blocks [5]. The left and right boundaries of the reservoir are assumed to be sealed and prevent flow as denoted by the bold line boundaries. In the grid block 1 on the left, there is a water injection well with flow rate of 75.96 B/D at standard conditions and grid block n on the right has an oil production well with a flow rate of -75.96 STB/D. The reservoir is 1000ft long. It is also assumed that the fluids are incompressible.

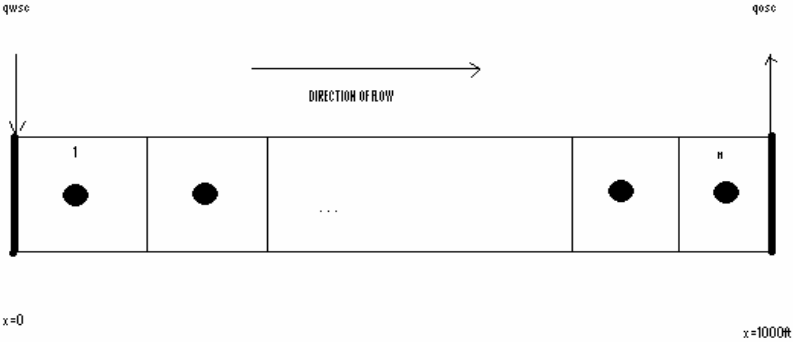


Fig. 1. 1D Oil-Water Reservoir with 2 Wells

When n is 4, the size of a grid is 250 ft with a cross-sectional area of 10000 ft2. The initial reservoir pressures and saturations are known.

At each grid block *i*, the oil pressure equation is of the following form

$$[T^{t}_{w,i,i-1} + T^{t}_{o,i,i-1}]\, p^{t+1}_{o,i-1} + [T^{t}_{w,i,i+1} + T^{t}_{o,i,i+1}]\, p^{t+1}_{o,i+1} - [T^{t}_{w,i,i-1} + T^{t}_{o,i,i-1} + T^{t}_{w,i,i+1} + T^{t}_{o,i,i+1}]\, p^{t+1}_{o,i} = 0 \quad (5)$$

While the water saturation equation at grid block *i* is of the form

$$S^{t+1}_{w,i} = S^{t}_{w,i} + \frac{1}{C_{ww,i}}[T^{t}_{w,i,i-1}(p^{t+1}_{o,i-1} - p^{t+1}_{o,i}) + T^{t}_{w,i,i+1}(p^{t+1}_{o,i+1} - p^{t+1}_{o,i})] \quad (6)$$

where the $T^{t}_{w,i,i-1}$ is the transmissibility between neighboring grid blocks i and i-1 at time step t (previous time step for which the variable's value has been computed and is know, while time step t+1 refers to the next time step for which the variables' values are to be found), $p^{t+1}_{o,i}$ is the oil pressure in grid block I at time t+1, $S^{t}_{w,i}$ is the water saturation in grid block I at time t, the coefficient $C_{ww,i}$ is given by $V_b \phi / \alpha_c \Delta t$. Note how the pressure variable in grid block I is a function of the pressure variables in the neighbouring grids at its left (i-1) and right (i+1). The water saturation variable at grid block i is also a function of the oil pressure variables at grid

blocks i, i-1, and i+1. These create dependencies which inhibit parallelism and are suitable to a few parallelization techniques such as pipelining.

In our case, after computing all coefficients, the resulting pressure (P) and saturation (S) equations for time step 100 days, assuming that at the prior time step, $p_{o1}= p_{o2}= p_{o3}= p_{o4}$=1000 psia, $S_{w1}= S_{w2=} S_{w3=} S_{w4}$ = 0.16, reduce to

$$p_{o1}=1000 \text{ psia}$$

$$\begin{vmatrix} 13.524 & 0 & 0 \\ -27.048 & 13.524 & 0 \\ 0 & 13.524 & -13.524 \end{vmatrix} \text{x} \begin{vmatrix} p_{o2} \\ p_{o3} \\ p_{o4} \end{vmatrix} = \begin{vmatrix} 13448.04 \\ -13524.00 \\ 75.96 \end{vmatrix} \tag{7}$$

where subscripts o and w refer to oil and water, respectively, and subscripts 1-4 refer to the grid block number, and the pressure equations are in matrix form **A** x **P** = **B**, where the **A** coefficients are functions of the transmissibilities and the **B** coefficients are function of the oil and water flow rates.

As the relative water permeabilities and consequently water transmissibilities in all grid blocks are 0, the water saturation equations at time step of 100 days are reduced to

$$\begin{vmatrix} S_{w1} \\ S_{w2} \\ S_{w3} \\ S_{w4} \end{vmatrix} = 0.16 + \frac{1}{890.472} \text{ x } \begin{vmatrix} 0(p_{o2} - p_{o1}) + 75.96 \\ 0(p_{o1} - p_{o2}) \\ 0(p_{o2} - p_{o3}) \\ 0(p_{o3} - p_{o1}) \end{vmatrix} \tag{8}$$

5 Parallel Computation Methodology

The simulator can be parallelized in various fashions. Data parallelism, thread parallelism, functional parallelisms are some common techniques to parallelize applications. In this work, we chose time parallelism and pipelining in which each SPE computes one full iteration including solution of the pressure and saturation equations and the Thomas algorithm for solving the above Matrix but at different time steps. The SPEs transfer the pressure and saturation values of each time step in a pipeline fashion among themselves, as explained later. As solutions at consecutive time steps are dependent on the solution at the previous time step, the SPEs can overlap in time the computation of the various coefficients and parameters and interpolate relative permeability data but would have to wait for the solutions of the pressure and saturation in the previous time step, before proceeding with their solution in the following time step. Thus the SPEs' executions are partially overlapped and not fully parallel.

We chose the following time steps: 100th day, 300th day, 600th day, and 800th day for the case of the 4 time step experiment, and extended these time steps with the following time steps: 1000th day, 1200th day, 1500th day, and 1800th day, for the 8 time step experiment. We chose these specific time steps as these have known solutions and

ease the process of validation. The time difference between the time steps, Δt, therefore varies from one time step to another and is not constant. Specifically, some of the coefficients' computations which are not dependent on the time step can take place concurrently on different SPEs. Similarly as per our implementation as soon as one SPE finishes its computation on the shared data for a particular time step, it releases (writes) the cache line which enables other SPEs to start computation while the former SPE continues its computation of non-shared data. Thus some computations even after the cache line is released occur in parallel.

The PPE is responsible for initializing the coefficient matrix and constant matrix. It is also responsible for initializing relative permeability and saturation values from the input table. Values for water relative permeability and oil relative permeability in the permeability table correspond to a water saturation value and can be interpolated from the table values provided that the water saturation is known. Later on, these values are forwarded to every SPE.

The SPEs are responsible for calculating the above equations, and their coefficients C_{op}, C_{wp}, $Tr_{w2,3}$, $Tr_{w3,2,}$, $Tr_{w4,3}$, $Tr_{w3,4}$, volume $V = A^* \Delta x$, geometric factor G, which are computed in parallel since they are not dependent on the time step. Also the coefficients based on transmissibilities mentioned above are also computed in parallel by the SPEs since they do not change with time.

6 Implementation Details

In the *serial* implementation, we initialized our 2D coefficient arrays/matrices with unknown and constant values. Once the arrays are initialized then we started the time iteration loop. We calculated the pressure distribution and water saturation distribution for the entire tested grids of size 32, 64, and 128 grid blocks, for four time steps (100^{th} day, 300^{th} day, 600^{th} day, and 800^{th} day), and in another experiment for 8 time steps (100^{th} day, 300^{th} day, 600^{th} day, 800^{th}, 1000^{th} day, 1200^{th} day, 1500^{th}, and 1800^{th} day). We also initialized those variables which were time independent before hand at the start and for each iteration we calculated those variables. We calculated Δt which is the time difference between the two consecutive time steps. We updated the corresponding coefficients during each iteration which are relying on the time delta, Δt. In the very first iteration or time step, for calculating the water saturation value for each grid we relied on the initial water saturation value but in the following time iterations we had to calculate the transmissibility value based on bilinear interpolation. Our implementation was flexible enough to be tested for different number of grids as we mentioned before.

In the embedded implementation, because of significant data dependencies and inter-process communications, we employed the atomic Unit and atomic cache technique. We distributed each time step computation or workload to a different SPE so in our experiments with total time of 4 (8) time steps, we used 4 (8) SPEs which partially processed some data in parallel and some data in pipelined and serial fashion. The data which needs to be computed in serial fashion was brought in the SPEs ahead of time for fast serial computation.

All the atomic operations supported by the SPEs are implemented by a specific Atomic Unit inside each MFC (Memory Flow controller) , which contains a dedicated local cache for cache line reservations. This cache is called the Atomic Cache. The Atomic Cache has a total capacity of six 128-byte cache lines, of which four are dedicated to atomic operations. When all the SPEs and the PPE perform atomic operations on a cache line with identical Effective Address, and therefore a reservation for that cache line is present in at least one of the MFC units, the cache snooping and update processes are performed by transferring that cache line contents to the requesting SPE or PPE over the Element Interconnect Bus, without requiring a read/write to main system memory. This constitutes effectively a very efficient hardware support for atomic operations on shared data structures consisting of up to 512 bytes divided in four 128-bytes blocks mapped on a 128-bytes aligned data structure in the SPEs' Local Store, and can be effectively used as a fast inter-processor communication broadcast mechanism. The approach to exploiting this facility is to extend the principles behind the handling of a mutex lock or an atomic addition, ensuring that the operations involved always affect the same four (or eight) cache lines.

For the embedded application, two code files, one for the PPE, and one for the SPEs were developed in the C programming language.

The PPE code performs the following steps.

a) Create separate threads to run PPE code and SPE codes in parallel, using libspe2 in our implementation.
b) Initialize relative permeabilities for both water and oil, oil flow, water flow, saturation values, and coefficient matrices.
c) Initialize the Control block structure and the corresponding member values which include shared input memory address from where every SPE has to retrieve data for processing and output memory address which will be used by every SPE to store the processed data after the computation.
d) Initialize the shared data structure. This data structure will be communicated between each SPE through atomic cache and contains a member time step. This value of this member determines which time step a particular SPE has to process.
e) PPE starts the SPE threads. In one experiment, 4 SPE threads are spawned. In another, 8 SPEs are spawned. The shared data structure is forwarded to every SPE with all member values initialized to zero.
f) Once all the threads/SPE contexts are spawned, the PPE sends the control block to every SPE through a non-blocking mailbox message. Every SPE after being started then has to wait for this mailbox message before it can proceed.
g) Once every SPE finishes the computation of the final time step values, the oil pressure and water saturation values will then be transferred to a shared memory location by the PPE.
h) The above steps are repeated for all grid sizes (32, 64, and 128 blocks), and for both considered total times (4 and 8 time steps).

The SPE code performs the following steps.

a) Every SPE is responsible for computing the pressure and saturation values for all grid blocks for one complete time step. There is no guarantee or prior information of which SPE will compute which time step iteration as they all

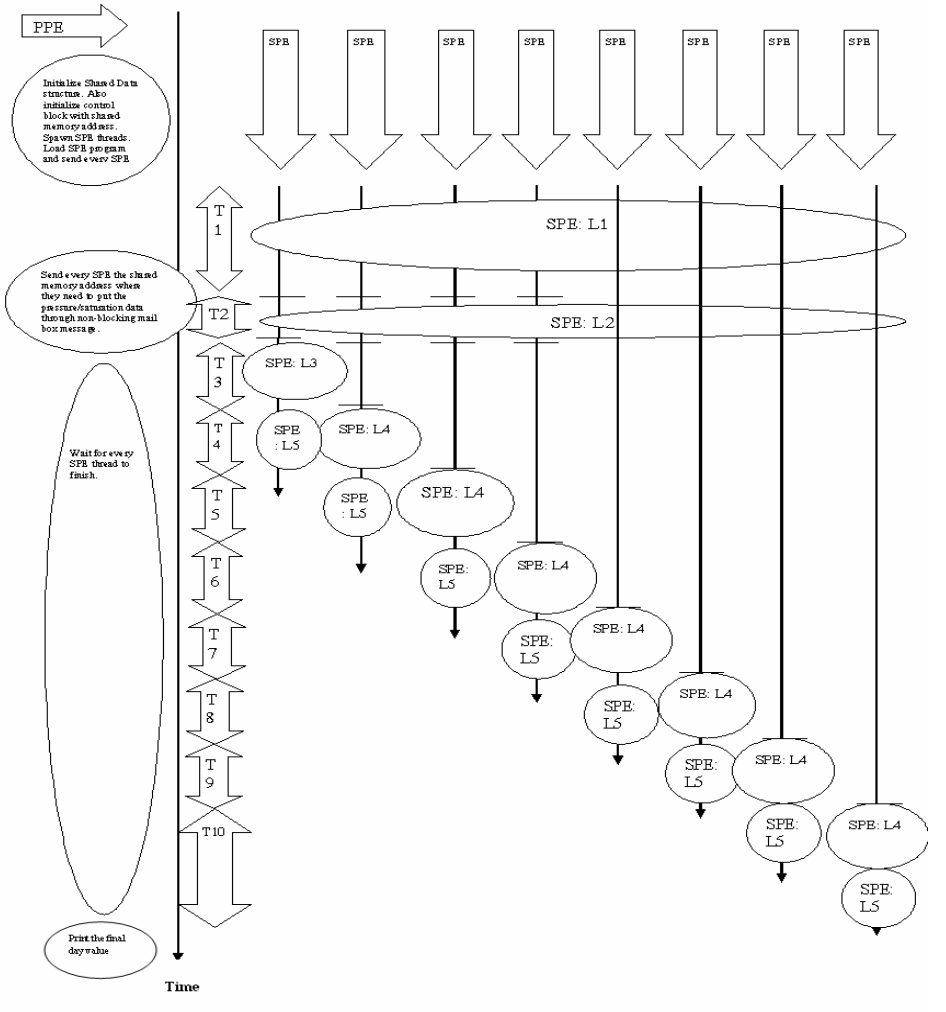
race for the cache line simultaneously and once each SPE gets a lock on the cache line, and based on the processing step, it starts computing the corresponding time step pressure and water saturation values.

b) As soon as the SPE program is invoked, the SPE starts calculating the equations which are independent of time. It then waits for the shared memory address included in the mailbox message sent by the PPE.
c) Every SPE then races for acquiring the lock on the atomic cache line. The first SPE which gets hold of the cache line locks the processing step value. If this value is 0 (this tells the SPE that it is assigned the first time step so it does not need to wait for another SPE to pass on to it previous iteration values), it then computes the time dependent coefficients and equations and then transfers the data to the shared memory location which will be used by the next SPE in line. It then releases or unlocks the cache line.
d) If this value if not 0, other SPEs which when they get hold of the cache line, retrieve the previously processed data (of the previous iteration or time step) from the shared memory location which may or may not be cached and compute the current time step-based coefficients, transmissibilities, oil Pressure and water saturation values. Once finished computing, the SPE transfers back to the main memory the current iteration's pressure and saturation values which will be used by the next SPE, and this is repeated each time by another SPE for as many time steps in the total time (4 or 8).
e) Those equations which are independent of any time iteration, e.g. calculating the volume of the entire grid, transmissibility for both water and oil between neighboring grid points which does not change with time, relative permeability, G_{nm} (geometric factor between grid blocks n and m), q_{osc} (well's oil flow rate), q_{wsc} (well's water flow rate), coefficients of pressure equations, and k_{rw} (relative permeability of water) are immediately calculated in parallel after each SPE thread is spawned.
f) In our implementation, we did not parallelize the Thomas algorithm, since we have a limited data size, and the parallelized Thomas algorithm involves communications between neighboring SPEs which entail DMA transfers which -we believe- may hurt the performance. That is why each SPE locally runs the Thomas algorithm on its own unknown variables to obtain a solution of these variables under its assigned time step.
g) We did not optimize the code manually. Instead we relied on the auto-compiler optimization technique. We used the optimization flags "-O3 -funroll-loops -fmodulo-sched -ftree-vectorize -ffast-math" which are for loop unrolling, software pipelining, auto vectorization and fast floating point calculation.

After finishing the above with one grid size, we increased the grid size (32, 64, 128 blocks) and repeated the same steps. The only changes we had to make is to modify the array sizes which represent the grid size, keeping the number of iterations or time steps constant.

After repeating for all grid sizes, we repeated the same steps with 8 time steps instead of just 4.

Fig. 2 shows the execution timing diagram of PPE and SPEs in the embedded application.



Legend

Label **L1:** Every SPE starts calculating those equations which are independent of time and relative permeability, and filling the coefficient matrices with constant values.

L2: Every SPE waits for the control block sent by the PPE via a mailbox message.

L3: SPE snoops the Cache line and as soon as it sees the atomic cache line free, the SPE locks it so no other processor can simultaneously access it. The SPE finishes the computations and writes the computed pressure and saturation values to the Cache or main memory.

L4: SPE snoops the Cache line and as soon as it sees the atomic cache line free, the SPE locks it so no other processor can simultaneously access it. The SPE retrieves the processed pressure and saturation data -computed in the previous iteration or time step- from the Cache or main memory. Once done, the SPE transfers the processed data back to main memory.

L5: The SPE calculates the oil saturation value, and water pressure distribution for all grid blocks. These operations overlap with the L4 computations.

Fig. 2. Execution timing diagram of PPE and SPEs

Where the vertical axis is time. Every SPE is represented by an arrow on top and its execution process is represented by another arrow below it. During the entire execution process, every computation phase is represented by an oval with the description of the phase. When the oval spans several SPEs, this indicates that this phase is being executed in parallel by the SPEs simultaneously. The label "T= Ti", where i=1-10, represents the time duration of each computation phase. A horizontal bar -crossing a vertical line- represents the end of a particular time period. Computations which are taking place during a phase are represented and detailed by an oval. For example, T1 indicates the time duration immediately after the SPE thread is spawned. During T1, every SPE starts calculating the coefficient matrices, transmissibilities and those equations which are independent of the time step, and will remain constant for the entire simulation. During T2, every SPE waits for getting the control block through the PPE. This waiting is happening in parallel by all involved SPEs as the PPE's mailbox message is non-blocking and does not require any acknowledgement from each SPE. From T3 onwards, the execution process takes a more serial turn (in reality, overlapped computation between a couple of SPEs) as every SPE now has to race for acquiring the atomic cache line which contains the shared data structure, thus each SPE has to wait for the time period T3 (by 1st SPE to get access), or T3+T4 (by 2nd SPE to get through) or T3+T4+T5 (by 3rd SPE to get through) or T3+T4+T5+T6 (by 4th SPE to get through) and so on before it gets hold of the cache line and starts its computation part. Note that this serial computation's waiting time is very minimal and does not affect the overall performance -as we observed from the results- since this process of acquiring the cache line is on-chip. During this serial (overlapped) computation phase, when a new SPE gets hold of the cache line, the SPE which previously held the cache line immediately starts computing the data which does not need to be shared among SPEs, thus some overlapping in the computation takes place between a pair of SPEs.

From the timing diagram of Figure 2, the total execution time can be expressed as

$$\text{Total Execution time} = T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T10 \quad (9)$$

Where T1= time of L1, T2= time of L2, T3=time of L3, T4=T5=T6=T7=T8=T9= time of L4 (with which L5 overlaps), and T10=time of L4 + time of L5. Our implementation's timing diagram clearly indicates that not all SPE (L4 and L5) codes are being executed in parallel because of the data dependency which exists between each time step iteration. For instance, in order to calculate the water saturation values for all grid blocks an SPE must have access to the previous iteration's oil pressure value, and similarly for grid block 1, the computation of the current iteration's oil pressure variable requires the previous iteration's oil pressure value. Thus each SPE has to wait for some time before it can calculate its current time step's oil pressure and water saturation variables and can write them to main memory.

7 Performance Results

We ran our experiments on an IBM Cell machine with 1 Cell blade containing one 9-core Cell processor package. In order to obtain more accurate timing information, we calculated the number of ticks on the serial PPE-only model through the mftb function

which is available through PPU intrinsic header files along with gettimeofday function. In the embedded (PPE+SPE) version, we invoked spu_read_decrementer and spu_write_decrementer functions to get the total number of clock ticks and passed on that information to the PPE code by updating the counter. These functions return more accurate and reliable timing information compared to the gettimeofday function. On the Cell blades, the time base frequency is set to 14.318 MHz with a 3.2 GHz microprocessor. In order to get the execution time, we divided the number of clock ticks by the time base frequency.

In our first experiment, we calculated the time it requires to run the oil reservoir simulation model based on the Thomas Algorithm for solving the oil pressure and water saturation equations for 4 (and 8) time steps on the single PowerPC core (PPE-only model). We repeated this experiment again on the single PowerPC with a varying grid size and calculated the execution time. We ran our experiments for grid sizes of 32, 64 and 128 grid blocks.

In the embedded implementation, we distributed the workload across four (and eight) different SPEs, each SPE core solving the equations for one time step, and the PPE core was only responsible for spawning SPE threads, forwarding the shared data structure and shared memory addresses to the SPEs. Every SPE is then responsible for calculating the oil and water pressure values and water saturation values using the Thomas Algorithm for a single time step. Table 1 and Fig. 3 display the execution times for PPE-only and PPE-SPE embedded implementations.

Table 1. Execution Time (in microseconds)

Grid Size (in blocks) / Mode	32	64	128
Embedded Execution Time	7.98* 16.95▽	20.49* 26.72▽	35.83* 50.25▽
Sequential PPE-only Execution Time	37.96* 57.97▽	79.55* 107.14▽	188.99* 232.23▽

*: 4 time steps

▽: 8 time steps

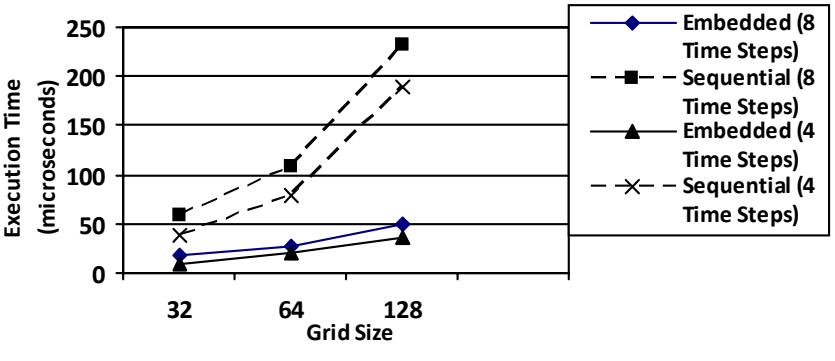


Fig. 3. Plot of execution time (in microseconds) vs. grid size

From Table 1 and Fig. 3, we observe a steady increase in the ratio of execution times of the two implementations, indicating that with increasing number of grid blocks, this application becomes more compute intensive and thus benefits from parallelizing and distributing the workload among different SPEs. The difference in execution time between the serial and parallel implementations is better in the case of 4 time steps than 8 time steps. Note that the Table 1 and Fig. 3 execution times were calculated by running code built with compiler optimizations. We built the executables with the compiler optimization level 3 along with loop unrolling option which duplicates the loop body multiple times and can help SPE performance as it reduces the number of branches. Further details on other compiler optimization flag can be find out in Section 6. In our code, there are multiple branches (loops) and quite a large number of matrix calculation which benefit from these compiler options. After compiler optimization, we observed a 2x speedup compared to building without the above mentioned compiler optimization flags.

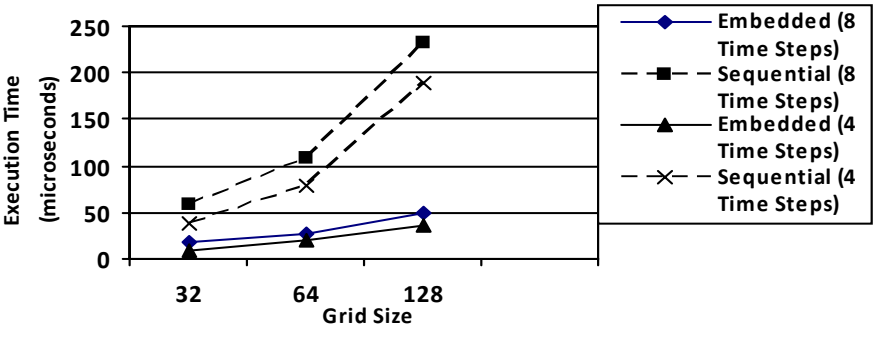


Fig. 4. Plot of speedup vs. grid size, excluding mailbox message wait times

Figure 4 plots the speedup which is the ratio of the embedded application execution time to the serial application execution time versus the grid size. In this Figure, the execution times exclude the mailbox message waiting times. The results are shown for both 4 and 8 days total time steps. The Figure 4 plot for the 8 time steps segment demonstrates that the speedup trend is somewhat linear with increasing grid size. We observe that when the 4 time steps workload is distributed among 4 SPEs, we obtain better speedup as compared to distributing the 8 time steps workload among 8 SPEs. The speedup gap between 4 and 8 times steps increases with increasing grid size. This results from significantly more main memory transfers in the latter case of 8 time steps as the grid size increases, as all the data does not fit entirely in the local stores and must be accessed from main memory.

In the timing diagram of Fig. 2, it is shown that every SPE after initialization has to wait (i.e is blocked) for some time for a mailbox message from the PPE which contains the effective address of the shared main memory address. We used additional shared main memory to introduce flexibility in the code for large data sets. As mentioned before, the atomic cache size is 512 bytes only which enables sharing of small size data sets. However, for large data sets, we had to store the data in main memory which may or may not be cached.

The PPE-to-SPE transfers of the shared memory address and of the shared data structure -which is shared among different SPEs- employ the atomic cache. This data structure contains the current step value which when received by each SPE after getting the atomic cache lock guides the SPE to which code portions (in the single SPE code file) to execute. This is needed as it is not possible to guarantee that one particular SPEs gets the lock before another SPE, so SPEs have to rely on the processing step value (shared among all SPEs through the shared data structure) to determine which iteration or time step each SPE is responsible for.

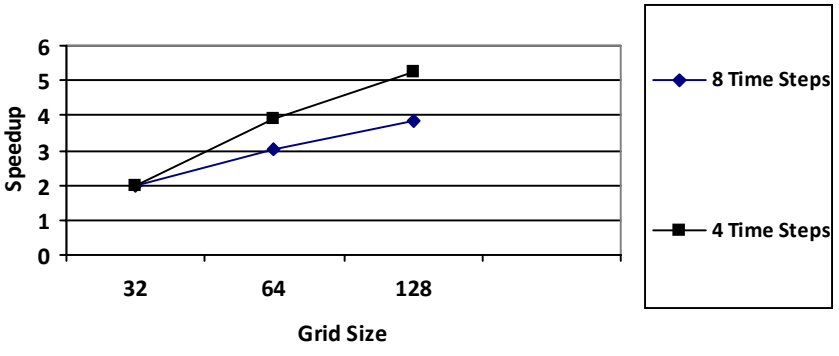


Fig. 5. Plot of speedup vs. grid size, including mailbox wait times

Fig. 5 plots the same speedup but now with execution times including the mailbox message times. We make a number of observations. First, we observe now that both speedup segments are non-linear. For instance in the case of 8 time steps, the grid of size 32 blocks is impacted the most by the blocking mailbox time, namely, a reduction in the speedup by more than 30%. This effect weakens with increasing grid size, resulting in speedup reductions of below 24% and 16%, for 64 and 128 grid blocks respectively. Second, the speedups with the mailbox message wait times included in the executions times (Fig. 5) are lower than those excluding the message wait times (Fig. 4). This is expected as the mailbox message wait time is somehow serial and does not concurrently take place with any other significant operation, resulting in a dilution of the parallel time in the total (serial + parallel) time. Third, comparing the speedups between 4 and 8 time steps, we note that the speedup difference between 4 and 8 time steps is negligible for the smaller grid sizes (i.e. 32) and grows as the grid size increases. This is due to the same rising amount of main memory transfers with increasing grid size and time steps as is also visible in Figure 4.

8 Conclusion

This paper describes the parallelization and development of a 2-phase oil-water reservoir simulator on the state-of-the-art IBM Cell computer. We described the interdependent linear algebraic equations of the reservoir simulator, the matrix solver based on Thomas algorithm, presented the pipelined time step parallelization approach

adopted on the Cell, and the performance results and speedups in reference to the single PowerPC host core.

The result presented in this paper reveal that given the largely interdependent nature of the oil reservoir model equations which highly limits parallelism, significant speedups of 6x or higher could be obtained on the first-generation Cell processor packages. While other applications could generate a much higher speedup or a performance of 1 Petaflop as recently reached by the RoadRunner supercomputer with a number of processors including 12960 Cells, in this oil application the obtained speedup is significant as it results in oil simulation runs cut from weeks to days. This allows reservoir analysts to multiply the number of simulation runs with various well locations and on the same hardware, resulting in better reservoir management and possibly higher oil production, the ultimate goal of this application.

The results also demonstrate that the oil reservoir simulator application is characterized by higher speedups with increasing grid size. This is because we adopt pipelined time step parallelism which assigns the linear equation computation for all grid blocks at each time step to a different SPE, which means that with an increasing number of grid blocks, each SPE will complete more useful computations contributing to widen the overlap times (between L4s and L5s in Fig. 2), and in the process helping to boost the speedup. Thus, although untested, we believe that higher speedups are expected at grid sizes above 128 blocks.

The results also reveal that, the speedup goes down with increased number of time steps, in particular as the grid size increases. With more time steps allocated to the same Cell processor, the main memory transfer overhead limits the speedup. Thus to reach even higher performance, we recommend splitting the time step computations over several Cell processors –if available-- where each Cell only handles 4 time steps rather than fully loading a Cell processor with the computation of 8 time steps. With the affordable cost of the Cell and the Playstation 3, this recommendation is economically appealing and feasible.

On the Cell BE processor, we also observed that the application's performance doubled with the chosen compiler optimization flags. This stresses the importance of compiler optimization flag experimentation during the software build phase which results in quick gains.

In future work, we will experiment with other forms of parallelism, including model and algorithm modification which relatively reduce the serial time and boost the parallel time by

1. Mapping the grid blocks to one SPE each where SPEs cooperate in solving the unknowns of a single time step before moving to the next time step. This results from a direct mapping of the reservoir's grid blocks to the Cell's SPE engines; and
2. Functional parallelism in which cores perform various functions, some dedicated to coefficient computation while others dedicated to solving for the unknowns with the parallel Thomas algorithm or parallel Gaussian elimination technique;
3. Block ordering such as red-black ordering in which some cores alternate in solving for the unknowns;
4. Running the code on the second-generation Cell with improved floating–point performance.

Acknowledgment

We acknowledge generous hardware equipment, software, and service support from IBM Corporation as a result of a 2006 IBM Shared University Research award, and a 2007 IBM Faculty award.

References

1. Adamson, G., et al.: Simulation throughout the Life of a Reservoir. Oilfield Review (1996)
2. Farmer, C.: Flow Through Porous Media and Reservoir Simulation, Mathematical Geophysics and Uncertainty in Earth Models (University of Oxford) (June 2004)
3. Aarnes, J., et al.: Towards Reservoir Simulation on Geological Grid Models. In: 9th European Conference on the Mathematics of Oil Recovery, Cannes, France (September 2004)
4. Oian, E., et al.: Parallel Simulation of a Multiphase/Multicomponent Flow Models. Lecture Notes in Computational Science and Engineering, pp. 99–113. Springer, New York (1999)
5. Ertekin, T., Abou-Kassem, J.H., King, G.: Basic Practical Reservoir Simulation, Society of Petroleum Engineers, Richardson, TX. SPE Textbook Series, vol. 7, p. 406 (September 2001)
6. Abou-Kassem, J.H., Farouq Ali, S.M., Islam, M.R.: Petroleum Reservoir Simulation: A Basic Approach, p. 445. Gulf Publishing Company, Houston (2006)
7. Solving the Oil Equation. IEEE Spectrum, 33–36 (January 2008)
8. http://www.research.ibm.com/cell
9. Gschwind, M., et al.: Synergistic Processing in Cell's Multicore Architecture. IEEE MICRO (March-April 2006)
10. IBM Corporation, Cell Broadband Engine Programming Handbook, http://www-306.ibm.com/chips/techlib/techlib.nsf/techdocs/9F820A5FFA3ECE8C8725716A0062585F
11. IBM Corporation, Cell BE Programming Tutorial, http://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/FC857AE550F7EB83872571A80061F788
12. Habiballah, W., Hayder, M.: Large Scale Parallel Reservoir Simulations on a Linux PC-Cluster. In: 4th LCI International Conference on Linux Clusters, San Jose, CA (June 2003)
13. Li, K., Zamel, N.: An Evaluation of HPF Compilers and the Implementation of a Parallel Linear Equation Solver Using HPF and MPI. In: ACM Conference (1997)
14. Øian, E., Espedal, M.S., Garrido, I., Fladmark, G.E.: Parallel Simulation of Multiphase/Multicomponent Flow Models
15. Sepehrnoori, K., Guler, B., Leng, T., Mashayehki, V., Rooholamini, R.: A High-Performance Computing Cluster for Parallel Simulation of Petroleum Reservoirs, Power Solutions, Dell Corporation (November 2003)
16. Zhang, K., Wu, Y.-S., Ding, C., Pruess, K.: Application of Parallel Computing Techniques to a Large-Scale Reservoir Simulation. In: Proc. of 26th Workshop on Geothermal Reservoir Engineering, California, January 29-31 (2001)
17. Matossian, V.: Autonomic Oil Reservoir Optimization on the Grid, Concurrency and Computation: Practice and Experience, vol. 17, pp. 1–26 (2005)
18. Saad, Y.: Iterative Methods for Sparse Linear Systems, PWS (1996)
19. Saad, M., Zhang, H.: Object Oriented Programming Techniques and FAC Method in Numerical Reservoir Simulation
20. Povitsky, A.: Parallelization of the Pipelined Thomas Algorithm, ICASE NASA Langley Research Center, NASA/CR-1998-20873, ICASE Report No. 98-48 (November 1998)

Efficient Reversible Logic Design of BCD Subtractors

Himanshu Thapliyal[1], Hamid R. Arabnia[2], and M.B. Srinivas[3]

[1] Department of Computer Science and Engineering, University of South Florida, USA
[2] Department of Computer Science , University of Georgia, USA
[3] Centre for VLSI Design and Embedded Systems, IIIT Hyderabad, India
hthapliy@cse.usf.edu, hra@cs.uga.edu, srinivas@iiit.net

Abstract. Reversible logic is emerging as a promising computing paradigm, having its applications in low-power CMOS, quantum computing, nanotechnology and optical computing. Firstly, we showed a modified design of conventional BCD subtractors and also proposed designs of carry look-ahead and carry skip BCD subtractors. The proposed designs of carry look-ahead and carry skip BCD subtractors are based on the novel designs of carry look-ahead and carry skip BCD adders, respectively. Then, we introduced the reversible logic implementation of the modified conventional, as well as the proposed, carry look-ahead and carry skip BCD subtractors efficient in terms of the number of reversible gates used and garbage output produced. To the best of our knowledge, the carry look-ahead and carry skip BCD subtractors and their reversible logic design are explored for the first time ever in literature.

Keywords: Reversible logic, BCD subtractors, BCD adders.

1 Introduction

The decimal arithmetic is receiving significant attention, as financial, commercial, and Internet-based applications cannot tolerate errors generated by conversion between decimal and binary formats [1]. The major consideration in implementing the BCD arithmetic is to enhance its speed as much as possible. Reversible logic is emerging as a promising computing paradigm, having its applications in future computing technologies such as optical computing, nanotechnology and quantum computing [6,7]. Reversible circuits are those circuits that do not lose information, and reversible computation in a system can be performed only when the system comprises of reversible gates. These circuits can generate a unique output vector from each input vector and vice-versa; that is, there is a one-to-one mapping between input and output vectors. Researchers like Landauer have shown that for irreversible logic computations, each bit of information lost generates kTln2 joules of heat energy, where k is Boltzmann's constant and T, the absolute temperature at which computation is performed [2]. Bennett showed that kTln2 energy dissipation would not occur if a computation is carried out in a reversible way [3], since the amount of energy dissipated in a system bears a direct relationship to the number of bits erased during computation.

The major goal in reversible logic design is to minimize the number of reversible gates used and garbage output produced (Garbage output refers to the output that is not used for further computations) [4,5].

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 99–121, 2009.

In this work, first, we showed a modified design of a conventional BCD subtractor. Two novel BCD subtractor architectures termed CLA (carry look-ahead) and CAS (carry skip) BCD subtractors are also proposed. The proposed designs of CLA and CAS BCD subtractors are based on novel designs of carry look-ahead and carry skip BCD adders. It is to be noted that a BCD subtractor internally consists of nine's complementer, BCD adder and parallel adder. Thus, special emphasis has been laid on their architecture to make them carry look-ahead and carry skip, to improve the overall efficiency of the subtractor.

Second, this paper introduces reversible logic implementation of the conventional and the proposed carry look-ahead and carry skip BCD subtractors, efficient in terms of number of reversible gates and garbage output. For achieving this goal, novel reversible gates have been proposed and novel techniques have been adopted which are discussed in appropriate sections in the paper.

The paper is organized as follows. Section 2 deals with the introduction of proposed modified conventional BCD subtractor. Section 3 and Section 4 introduce the proposed carry look-ahead and carry skip BCD subtractors, respectively. Section 5 deals with the introduction of basic reversible gates used in the proposed work. Section 6 deals with the reversible design of conventional BCD subtractor introduced in Section 2. Section 7 and Section 8 deal with the reversible design of proposed carry look-ahead and carry skip BCD subtractors, respectively. Section 9 and 10 provide the implementation results and conclusions of this work, respectively.

2 BCD Subtractor

In the BCD subtraction, the nine's complement of the subtrahend is added to the minuend. In the BCD arithmetic, the nine's complement is computed by nine minus the number whose nine's complement is to be computed. This can be illustrated as the nine's complement of 5 will be 4 (9-5= 4), which can be represented in BCD code as 0100. In BCD subtraction using nine's complement, there can be two possible possibilities [8]:

1. The sum after the addition of minuend and the nine's complement of subtrahend is an invalid BCD Code (an example is when 5 is subtracted from 8) or a carry is produced from the MSB (an example is when 1 is subtracted from 9). In this case, add decimal 6 (binary 0110) and the end around carry (EAC) to the sum. The final result will be the positive number represented by the sum.
2. The sum of the minuend and the nine's complement of the subtrahend is a valid BCD code which means that the result is negative and is in the nine's complement form. An example is, when 8 is subtracted from 5.

In BCD arithmetic, instead of subtracting the number from nine, the nine's complement of a number is determined by adding 1010 (Decimal 10) to the one's complement of the number. The nine's complementer circuit using a 4-bit adder and XOR gates is shown in Fig.1 [8]. We have realized that there is no need to use XOR gates in the nine's complementer for complementing. The use of NOT gates will better suit the purpose and will reduce the complexity of the circuit, both in CMOS as well as reversible logic implementation. The proposed modified design of nine's

complementer is shown in Fig.2; it replaces 4 XOR gates by 4 NOT gates and thus is better compared to the existing design in literature. The one-digit BCD subtractor, using the nine's complementer circuit, is shown in Fig.3. In Fig.3, after getting the nine's complement of the subtrahend, it is added to the minuend using the BCD adder. Then the required 1010 is added by using the complement of the output carry of the BCD adder. The sign represents whether the number stored is positive or negative (for example, 5-8 will be stored as Sign=1 and Magnitude (S3...S0) = 3).

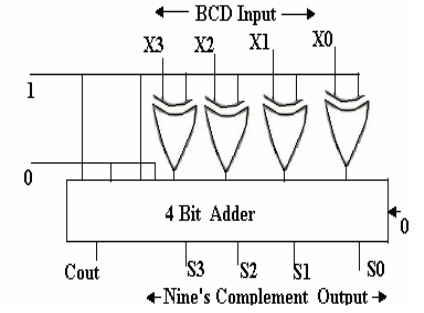


Fig. 1. Nine's Complementer

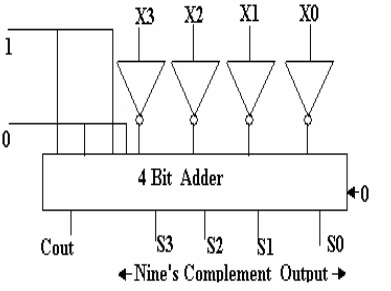


Fig. 2. Proposed Nine's Complementer

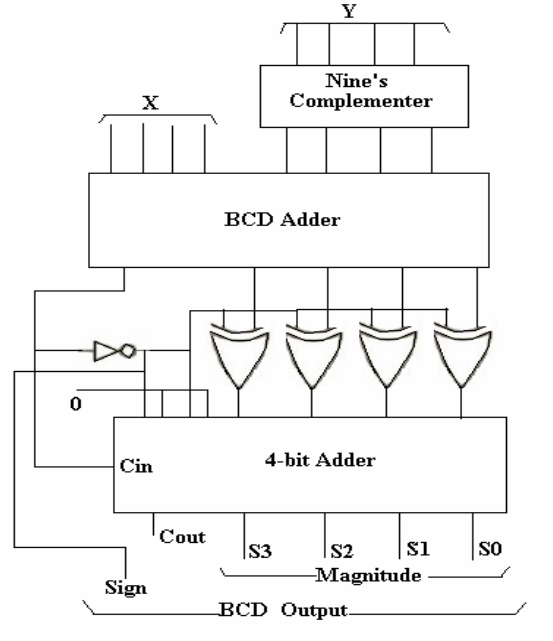


Fig. 3. Modified Conventional BCD Subtractor

3 Proposed Carry Look-Ahead BCD Subtractor

As evident from Fig.3, the nine's complementer, BCD adder and 4-bit adder are the integral components of the BCD subtractor. Thus, we propose the replacement of the conventional nine's complementer, the BCD adder and the 4-bit adder by their carry look-ahead counterparts. This will help us to design a faster and more efficient overall BCD subtractor.

3.1 Carry Look Ahead BCD Adder

A carry look ahead BCD adder is proposed which is a modification over the architecture proposed in [9,10] and is especially improved for making it suitable for CMOS and reversible logic implementation. In the proposed CLA BCD adder, OR gates used in the equations proposed in [9,10] are carefully chosen and replaced by XOR gates. One cannot replace the OR gates in the equations in [9,10] randomly. Hence, a rigorous study has been done and OR gates in the equations in [9, 10] have been replaced in certain places. The functional verification of the proposed CLA BCD adder is done in Verilog HDL using the Active HDL simulator. The advantages of using this approach are as follows:

1. In the conventional CMOS logic, the XOR gate can be designed, with a fewer number of transistors compared to the OR gate.
2. In reversible logic, the multi-input XOR gate can be designed with a less complex reversible gate and one less garbage output compared to the multi-input OR gate. For example, the equation $a \oplus b \oplus c$ can be realized with only one (3x3 reversible gate) and two garbage output compared to a+b+c (here + refers an OR gate), which can be realized with one 4x4 reversible gate and three garbage output, or two 3x3 reversible gates with four garbage output. *Thus, in reversible logic it is better to realize equations as XOR functions.* This advantage of XOR gate will become more dominant as the input size is increased beyond three.

Consider two BCD numbers a and b of 4 bits each, using the proposed approach, the modified functions used to generate the carry look-ahead BCD adder are as follows

// 1^{st} Part
$g[j] = a[j] \bullet b[j]$ $0 \leq j \leq 3$ "generate"
$p[j] = a[j] + b[j]$ $0 \leq j \leq 3$ "propagate"
$h[j] = a[j] \oplus b[j]$ $0 \leq j \leq 3$ "half-adder"

//2^{nd} Part
$k = g[3] \oplus (p[3] \bullet p[2]) + (p[3] \bullet p[1]) \oplus (g[2] \bullet p[1])$
$L = p[3] \oplus (g[2] + (p[2] \bullet g[1]))$
(Here k and L are the carry generate and propagate functions of the first three bits of the decimal number a and b (a[3]a[2]a[1] and b[3]b[2]b[1]), respectively. The details and complete description of k and L can be found in [9])

C1 = g[0] + (p[0] • Cin) "carry out of 1's position"

//3rd Part

S[0] = h[0] ⊕ Cin

S[1] = ((h[1] ⊕ k) • ~C1) + (~(h[1] ⊕ L) • C1)

S[2]=(~p[2]•g[1]) ⊕ (~p[3]•h[2]• ~p[1]) ⊕
((g[3] ⊕ (h[2]•h[1]))• ~C1)+ (((~p[3] • ~p[2] •
p[1]) ⊕ (g[2] • g[1]) ⊕ (p[3] • p[2])) • C1)

S[3]=((~k• L)• ~C1) ⊕(((g[3] • ~h[3]) ⊕
(~h[3] • h[2] • h[1])) • C1)

Cout = k + (L • C1).

In the above equations, **S[3], S[2],S[1], S[0]** represents the sum bits produced by addition of BCD numbers **a** and **b** with input carry **Cin**. The output carry produced by the CLA BCD adder is represented by **Cout.**

3.2 Carry Look-Ahead Binary Adder

As evident in the architectures of the nine's complementer and the modified conventional BCD subtractor shown in Figs.2 and 3, respectively, the improvement in the 4-bit adder is the key requirement to increase their efficiency. We propose the replacement of 4-bit adder blocks with their carry look-ahead counterparts. Recently, a modified carry look-ahead adder (abbreviated as MCLA) is proposed which is similar to CLA (carry look-ahead adder) in basic construction [11]. The drawback of MCLA is that, despite its faster speed, it occupies a larger area due to the excessive number of NAND gates used for faster carry propagation. This problem will significantly increase when MCLA is used to design a higher order CLA. This is the reason why we are proposing a new carry look-ahead adder, modifying the structure of MCLA to make it more economical in terms of the number of gates (area), without losing its speed efficiency. The MCLA [11] uses the modified full adder (MFA) as shown in Fig.4. In our proposed carry look-ahead adder shown in Fig.5, we propose replacing the 4th MFA in the MCLA by a full adder to reduce the area (number of gates) without sacrificing speed improvement.It can easily be verified that there will be a reduction in the number of gates to generate the final carry, as shown in Fig.5. In order to have further savings in terms of the number of gates, the proposed 4-bit CLA can be cascaded in a series to design an expanded width CLA, as shown in Fig.6.

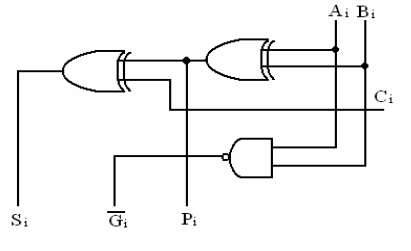


Fig. 4. MFA (modified full adder)[11]

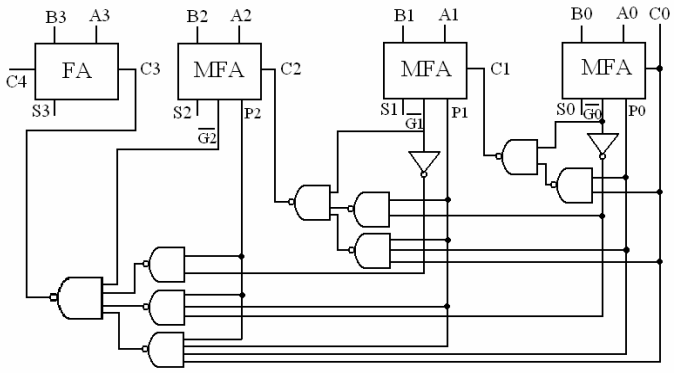


Fig. 5. Proposed 4-bit Carry Look-Ahead Adder

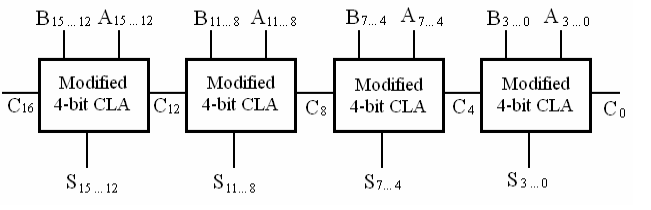


Fig. 6. Cascading for Expanded Width CLA

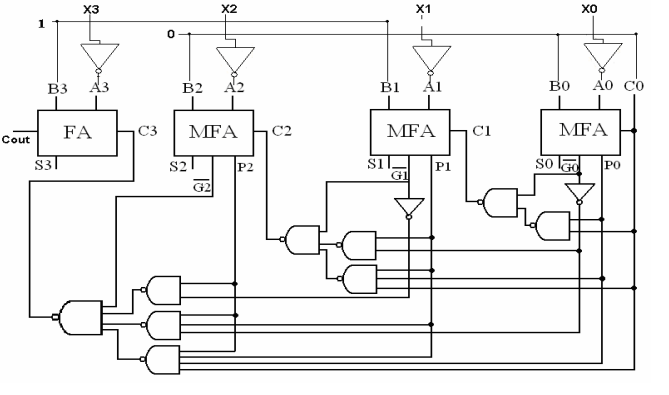


Fig. 7. CLA Nine's Complementer

Figure 7 shows the proposed nine's complementer using the proposed carry look-ahead adder and using the proposed concept of using NOT gates for complementting (rather than XOR gates). The proposed nine's complementer satisfies the requirements of the carry look-ahead approach pertaining to fast speed and reduced area (inherit property of proposed CLA).

Evaluation of the Proposed Approach
The adders are coded in Verilog HDL and synthesized using Xilinx VirtexE FPGA. For 16-bit addition, the CPA (carry propagate adder) has a delay of 26.109 ns with cell usage of 36, while MCLA has a delay of 16.954 ns with cell usage of 46. The proposed CLA takes 21.931 ns of delay with cell usage of 35. It can be concluded from the above results that the proposed carry look-ahead adder approach is of great significance, since it provides a good speed, with cell usage nearly the same as that of the CPA. Thus, the proposed carry look-ahead adder having a delay in between the MCLA and CPA, and an area nearly equal to CPA, is the best choice.

3.3 Carry Look-Ahead BCD Subtractor

After having its key components (BCD adder, 4-bit adder and nine's complementer) designed in carry look-ahead fashion, the carry look-ahead BCD subtractor can be designed by integrating the components. Figure 8 shows the design of the proposed carry look-ahead BCD subtractor. It is to be noted that we have laid emphasis on improving the individual modules of the BCD subtractor, to improve its overall efficiency and make it more suitable for reversible logic implementation.

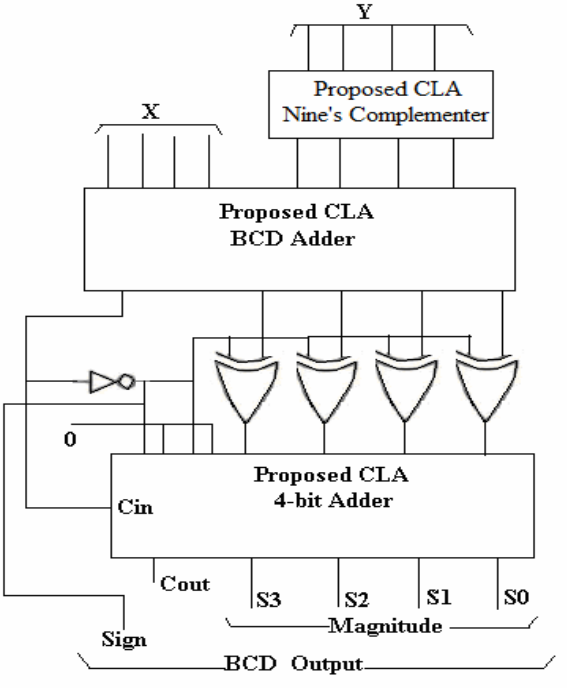


Fig. 8. Proposed CLA BCD Subtractor

4 Proposed Carry Skip BCD Subtractor

In order to design the carry skip equivalent of the BCD subtractor, we propose the carry skip equivalent design of its individual components.

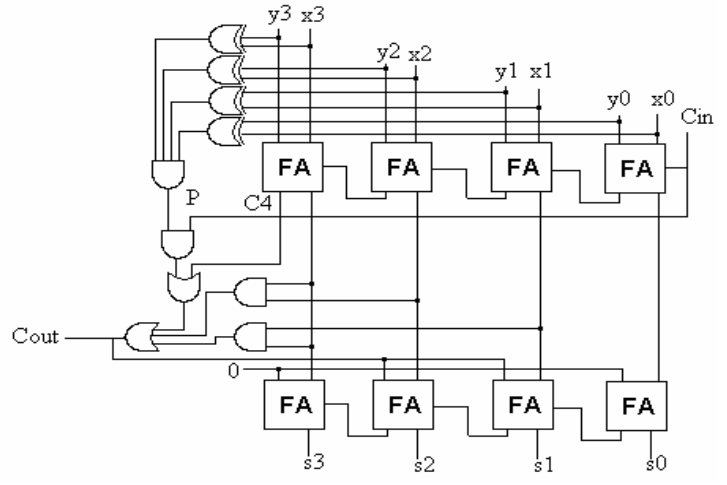


Fig. 9. Proposed Carry Skip BCD Adder

4.1 Carry Skip BCD Adder

In this work, we propose the design of carry skip BCD adder. It is constructed in such a way that the first full adder block consisting of 4 full adders can generate the output carry 'Cout' instantaneously, depending on the input signals and 'Cin'. This avoids carry to be propagated in the ripple carry fashion. Figure 9 shows the proposed carry skip BCD adder. The working of the proposed carry skip BCD adder (CS BCD Adder) can be explained in this manner: In the single bit full adder operation, if either input is a logical one, the cell will propagate the carry input to its carry output. Hence, the i^{th} full adder carry input Ci, will propagate to its carry output C_{i+1} when Pi= Xi$\oplus$Yi, where Xi and Yi represents the input signal to the i^{th} full adder. Thus, the four full adders at the first level making a block can generate a "block" propagate signal 'P'. When 'P' is one, it will make the block carry input 'Cin', to propagate as the carry output 'Cout' of the BCD adder, without waiting for the actual propagation of carry in the ripple carry fashion. An AND gate is used to generate a block propagate signal 'P'. Depending on the value of 'Cout', appropriate action is taken. When 'Cout' is equal to one, binary 0110 is added to the binary sum (correction logic to convert sum in BCD format) using another 4-bit binary adder at the second level or bottom level, as shown in Fig.9. The output carry generated from the bottom binary adder is ignored, since it supplies information already available at the output carry terminal.

4.2 Carry Skip BCD Subtractor

Figure 10 shows our proposed design of the carry skip BCD subtractor. It is to be noted that the carry skip implementation of the nine's complementer in the proposed circuit will not be beneficial, making the carry look-ahead as the best choice for its implementation. The carry skipping property of the BCD adder can be beneficial only when its input carry Cin=1. Thus, in order to extract the benefit of the carry skip

property of the BCD adder in the proposed BCD subtractor, we have made the LSB output (n[0]) of the nine's complementer as input carry 'Cin' of the carry skip BCD adder and passed '0' in its place for addition to the BCD adder (please refer Fig.10). Therefore, the numbers passed for addition in carry skip BCD adder will be X+(n[3]n[2]n[1]'0')+n[0], where n[0] will work as Cin. The last block of the 4-bit adder in the proposed circuit has also been designed in the carry skip fashion to further improve the efficiency of the proposed design. This will result in the generation of **Cout** in Fig.10 in carry skip fashion. As far as existing literature and our knowledge is concerned, the proposed circuit is the maiden attempt to provide the carry skip equivalent of the conventional BCD subtractor.

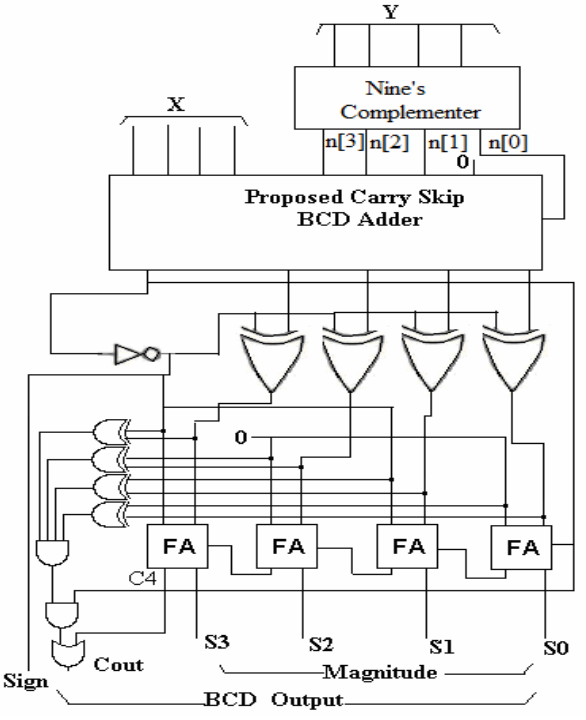


Fig. 10. Proposed Carry Skip BCD Subtractor

5 Basic Reversible Gates

There are a number of existing reversible gates in literature. We have used Fredkin gate [12,13], Feynman Gate [12,13], Toffoli Gate (TG) [12,13], New Gate (NG) [14], New Toffoli Gate (NTG)[15], TKS[17], R2 Gate and TS-3 gate(3*3 and 4*4 Feynman gate, respectively) and TSG Gate[16] to design the reversible BCD subtractors. Since the major reversible gate used in designing the BCD subtractors are Feynman, Modified Toffoli Gate [19], Toffoli, Fredkin and TSG gate, only these reversible gates are discussed in this section.

5.1 Fredkin Gate

Fredkin gate is a (3*3) conservative reversible gate originally introduced by Petri [12, 13] as shown in Fig.11. It is called 3*3 gate because it has three input and three output.

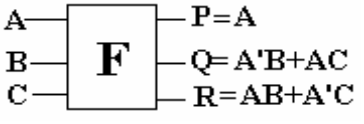


Fig. 11. Fredkin Gate

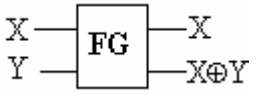


Fig. 12. Feynman Gate

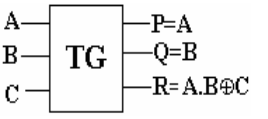


Fig. 13. Toffoli Gate

5.2 Feynman Gate

Feynman gate [12,13] is a 2*2 one-through reversible gate shown in Fig.12. It is called 2*2 gate because it has 2 input and 2 output. One-through gate means that one input variable is also the output. An n input and n output Feynman gate can be described as mapping (x1,x2,x3…..xn) to (x1,x2,x3…..,x1⊕x2⊕x3⊕…xn-1⊕xn).

5.3 Toffoli Gate

Toffoli Gate (TG) [12, 13] is a 3*3 two-through reversible gate as shown in Fig. 13. A n input and n output Toffoli gate can be described as mapping (x1,x2,x3…..xn) to (x1,x2,x3…..,(x1x2x3…xn-1)⊕xn).

5.4 TSG Gate

Recently, a 4*4 one-through reversible gate called TS gate "TSG" was proposed [16]. The reversible TSG gate is shown in Fig.14. It can be verified that the input pattern corresponding to a particular output pattern can be uniquely determined. The TSG gate can implement all Boolean functions. One of the prominent functionalities of the TSG gate is that it can work singly as a reversible full adder unit. Figure 15 shows the implementation of the TSG gate as a reversible full adder. TSG can implement the reversible full adder with a bare minimum of two garbage output (at least two garbage output will be required to realize a reversible full adder).

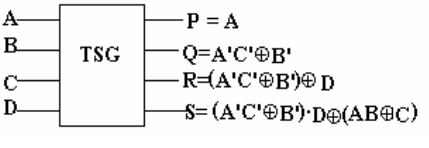


Fig. 14. Reversible 4 *4 TS Gate

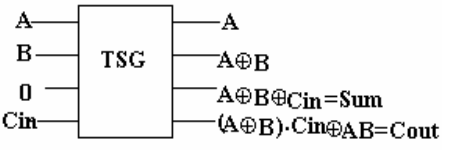


Fig. 15. TSG as a Reversible Full Adder

5.5 Modified Toffoli Gate

Modified Toffoli gate (MTG) is a 3*3 reversible gate and is shown in Fig. 16 [19]. An n input and n output MTG gate can be described as mapping (x1,x2,x3…..xn) to (x1,x2,x3…..,(x1|x2|x3|…xn-1)⊕xn).

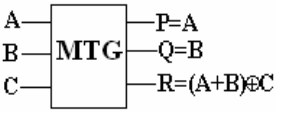


Fig. 16. MTG Gate

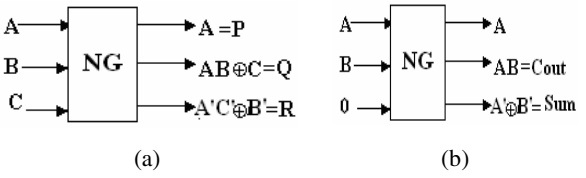


Fig. 17. (a) New Gate (NG), (b) as a reversible half adder

5.6 New Gate (NG)

New gate (NG) [14] is another important 3*3 gate used in our designs of BCD subtractors, shown in Fig.17.a. New gate can work singly as a reversible half adder with minimum of one garbage output, as demonstrated in Fig.17.b.

6 Reversible Design of Conventional BCD Subtractor

It is evident from Fig. 3 that in order to design reversible BCD subtractors, the whole reversible design must be divided into three sub-modules.

1. Design of the reversible nine's complementer (which, in turn, has to be designed using reversible parallel adders).
2. Design of the reversible BCD adder.
3. Integration of the modules using existing reversible gates to design the reversible BCD subtractor.

Our primary goal in this work is to design reversible BCD subtractors with a minimal number of reversible gates and garbage output.

6.1 Reversible Nine's Complementer

Figure 18 shows the proposed reversible nine's complementer using the NOT gates, New gates (NG) and the 3*3 Feynman Gate (FG3). The proposed design is implemented with 7 reversible gates and 3 garbage output. *To minimize the garbage at the bottom 4-bit adder, we have utilized the proposed property of regenerating the constant value at the garbage output (the constant input '1' at the NG gate is regenerated at one its garbage output and is used as input to FG3.*We observed that the **S0** can be directly generated without requiring any addition circuitry (referring to Fig. 1, we observed that second input to the full adder is '0' as well as the Cin is '0'). Further examination showed that there is no need for the full adder in the 2nd place, 3rd place and 4th place of the bottom 4-bit adder. Half adders and 3 input XOR gate can perform the required addition operations. The reversible half adder can be designed

by New gate (NG) with only one garbage output (refer to Fig.17.b), and the 3 input XOR gate can be designed using FG3 with only two garbage output. Utilizing the reversible full adder in those places would have increased the garbage, as at least two garbage output are required in a reversible full adder. Moreover, the output carry is not required in the nine's complementer. Thus, using the reversible full adder would have generated the output carry leading to an increase in garbage count.

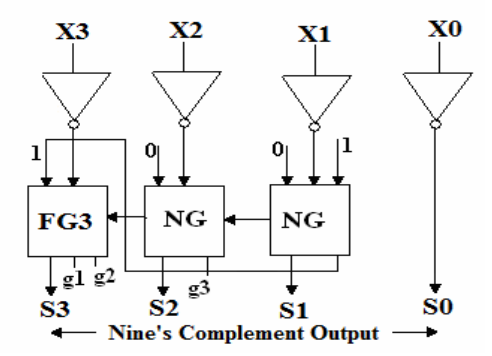


Fig. 18. Reversible Nine's Complementer

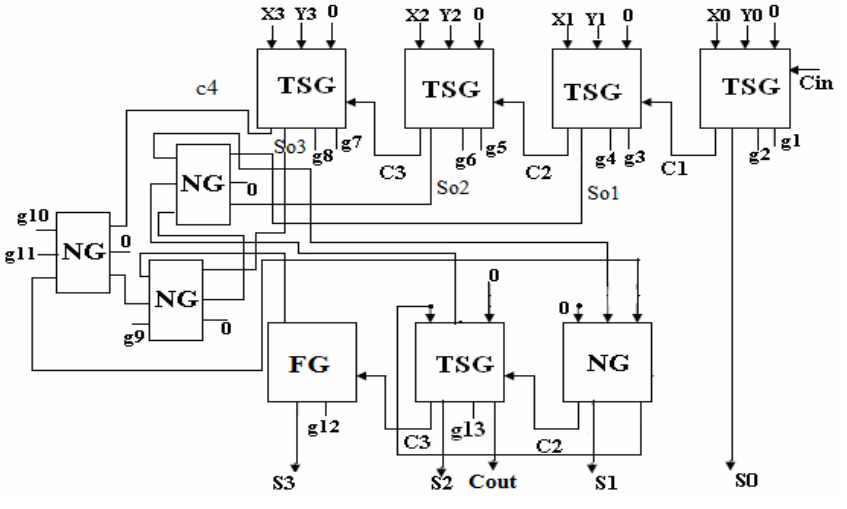


Fig. 19. Reversible Logic Implementation of the Conventional BCD Adder

6.2 Reversible BCD Adder

Figure 19 shows the reversible implementation of the conventional BCD adder using the reversible TSG and New Gate. In BCD addition, the steps are as follows:

Step 1: The two decimal digits (X and Y), together with the input carry (Cin), are first added using a 4-bit binary adder to produce the binary sum (So3,So2,So1,S0) and output carry(c4).

Step 2: When the output carry (c4) is equal to zero, nothing is added to the binary sum. When it is equal to one, binary 0110 is added to the binary sum using another 4-bit binary adder. Instead of directly adding 0110, it is added by generating Cout=C4+ So3(So2+So1). In 0110 addition, where '1' is required, Cout is used instead.

In Fig.19, the 4 TSG gates at the top in Fig.19 perform Step 1. The three New gates generate Cout=c4+So3 (So2+So1). The final addition is performed using NG, TSG and FG reversible gates. *The proposed BCD adder architecture in Fig. 19 uses only 10 reversible gates and produces only 14 garbage output.* As can be observed in Fig.19, the connections are carefully made to avoid the garbage. The proposed BCD adder is shown to be much better than the earlier proposed architecture both in terms of number of reversible gates and garbage output. Recently in [18], the BCD adder is implemented with 23 reversible gates and 22 garbage output. A comparison between our proposed design and the existing design is shown in Table 1. The proposed design in this paper is the most efficient design of reversible BCD adder and achieves an improvement ratio of 2.3 and 1.69 in terms of the number of reversible gates and garbage output, respectively.

Table 1. A comparison of Reversible BCD Adder

	Number of Gates	Number of Garbage Output
Proposed Circuit	10	13
Existing Circuit[18]	23	22
Improvement Ratio	2.3	1.69

6.3 Reversible BCD Subtractor

Figure 20 shows the reversible BCD subtractor using the reversible nine's complementer, reversible BCD adder, TSG, NG and Feynman gate (FG). *In the above sections, we have proven that the proposed reversible designs of the nine's complementer and BCD adder are designed with minimal number of reversible gates and garbage output.* In order to design a more efficient complete BCD subtractor in terms of the number of reversible gates and garbage output, we have used Feynman Gate for generating the XOR/NOT function and copying the output (as fan-out is not allowed in reversible logic). We chose Feynman gate as it can generate the XOR/NOT function and copy the output with minimum number of reversible gates and garbage output. This can be understood by the fact that there are exactly two output corresponding to the input of a Feynman gate, a '0' in the second input will copy the first input in both the output of that gate. It makes the Feynman gate most suitable for a single copy of bit, since it does not produce any garbage output.

It is to be noted that we have carefully examined the architecture of BCD subtractors and in the middle of Fig. 20 used the Feynman gates as chains for generating the XOR, copying and NOT functions, with zero garbage. If the architecture is not deeply examined, it can lead to an inefficiently designed reversible circuit with increased garbage. The reason for this stems from the fact that when the Feynman gate is used for generating the XOR and NOT functions, it produces at least one garbage output in both cases.

The bottom 4-bit binary adder required in BCD subtractor is also designed very efficiently to minimize the garbage. This is achieved by carefully passing the input signal and thereby utilizing the garbage output for further computation along with identifying suitable places where reversible full adders can be replaced by reversible half adders. *It is to be noted that we have designed the bottom 4-bit adder with 4 reversible gates and 4 garbage output. An inefficient approach of simply designing the 4-bit adder with the reversible full adder could lead to 8 garbage output (at least two garbage output are produced in a reversible full adder).* The BCD adder requires 10 reversible gates and 13 garbage output as proven above. The nine's complementer is designed with 7 reversible gates and 3 garbage output. The generation of XOR functions, copying and NOT functions are designed in such an optimal manner that it requires 5 Feynman gates with zero garbage output. The bottom 4-bit reversible adder is designed with 4 reversible gates and 4 garbage output. Thus, the proposed reversible BCD subtractor is designed with 10+7+5+4=26 reversible gates while the garbage output is minimal of 13+3+4=20.

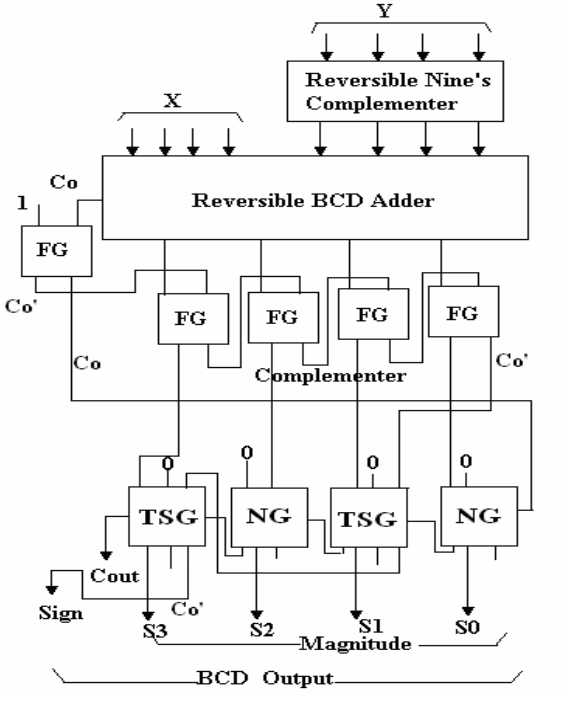


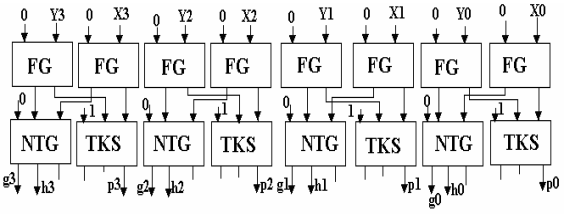
Fig. 20. Proposed Reversible BCD Subtractor

7 Reversible Design of Carry Look Ahead BCD Subtractor

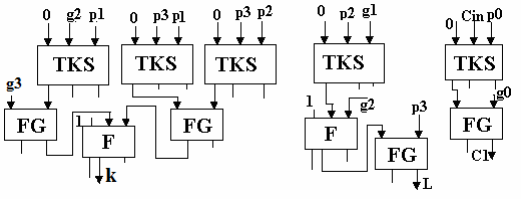
As evident from the earlier discussion, the reversible implementation of carry look-ahead BCD subtractor will require the reversible implementation of carry look-ahead BCD adder, the proposed carry look-ahead nine's complementer and 4-bit carry look-ahead adder.

7.1 Reversible Carry Look Ahead BCD Adder

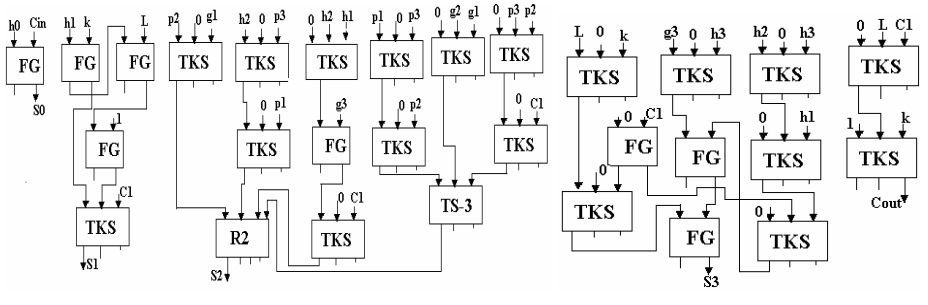
The reversible logic implementation of the carry look-ahead BCD adder is shown in Fig.21. The reversible gates used for designing the proposed reversible carry look-ahead BCD adder are Feynman gate (FG), TKS gate, New Toffoli gate (NTG), and R2 gate (a 4*4 Feynman Gate) and TS-3 gate (the details of these reversible gates are discussed in Section V). In the proposed reversible circuit, Feynman Gates (FG) can be used for copying the output and to avoid the fan-out problem. The proposed reversible CLA BCD adder can be of significant use in future computing technologies. Furthermore, it is a hierarchical architecture; hence, huge power savings can be obtained by switching off the blocks which are not in use, through a control circuitry. It is to be noted that appropriate reversible gates are used in Fig.21, to design it overall efficient in terms of number of reversible gates and garbage output.



(a) Part 1. Generation of g[j], p[j] and h[j] for $0 \leq j \leq 3$



(b) Part 2. Generation of k,L and C

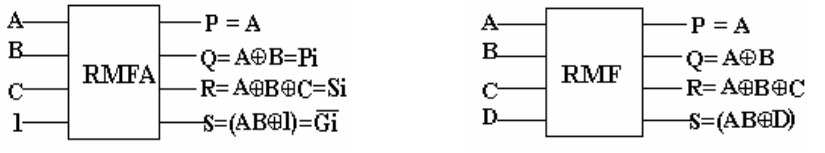


(c) Part 3. Generation of Sum Bits S3,S2, S1,S0 and Cout

Fig. 21. Reversible Implementation of proposed Carry Look Ahead BCD Adder

7.2 Reversible Carry Look Ahead Adder

The key consideration while designing reversible carry look-ahead adder is to generate Pi, Si and Gi' signals with the minimum number of reversible gates and garbage output. Thus, in order to generate Pi, Si and Gi' signals with the best possible case of 1 reversible gate and 1 garbage output (*at least one garbage output will be required to make the function Pi, Si and Gi' reversible. For two cases (input combinations), we will get the same output which can be removed by addition of one garbage bit*), we propose the design of a novel 4*4 reversible gate called RMF gate as shown in Fig.22.a. The RMF gate can realize Pi, Si and Gi' signal as shown in Fig. 22.b (termed as RMFA block). Thus, RMF is able to realize the Pi, Si and Gi' with the lower bound of 1 reversible gate and 1 garbage output.



(a) (b)

Fig. 22. (a) Proposed 4*4 Reversible Gate (b) RMF for Generating Pi,Si & Gi'

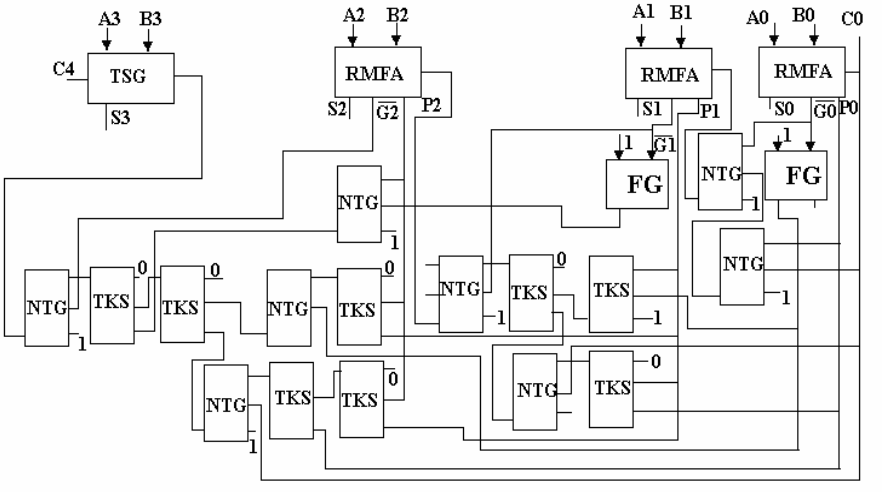


Fig. 23. Reversible Carry Look-Ahead Adder

Figure 23 shows the complete reversible design of the 4-bit carry look-ahead adder. In the complete reversible design of proposed CLA, appropriate reversible gates are used wherever required for generating the function with the minimum number of reversible gates and garbage output. The garbage output is not shown in Fig.23, but it

can be identified as the output which is not used in further computations. The fan-out problem is also not considered just to simplify the circuit, as it can be easily avoided by using the Feynman gate. TKS and Peres Gate (NTG) combination is used for generating the multi-input NAND functions. The 4th block (adding A3 & B3) in Fig.23 consists of only the TSG gate, as only the reversible full adder block is required. The reversible nine's complementer is designed with the proposed reversible CLA, as shown in Fig.24, using NOT gates for complementing.

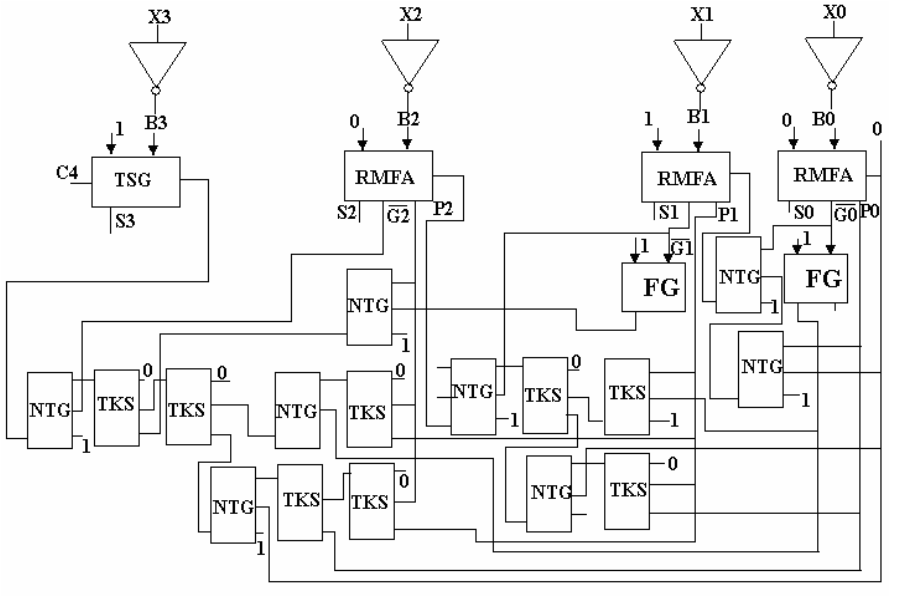


Fig. 24. Reversible Nine's Complementer

7.3 Reversible Carry Look-Ahead BCD Subtractor

After designing the individual reversible components of the carry look-ahead BCD subtractor, the components are combined together to design the complete reversible carry look-ahead BCD subtractor, as shown in Fig.25. It is to be noted that we have used the same strategy of connecting the Feynman gates as chains for generating the XOR, copying and NOT functions, with zero garbage (Please refer to Fig.8, in which four XOR and one NOT gate is required in the middle of the CLA BCD subtractor). Thus, the architecture is designed efficiently in terms of the number of reversible gates and garbage output.

8 Reversible Design of Carry Skip BCD Subtractor

The reversible logic design of carry skip BCD subtractor requires the reversible design of carry skip BCD adder.

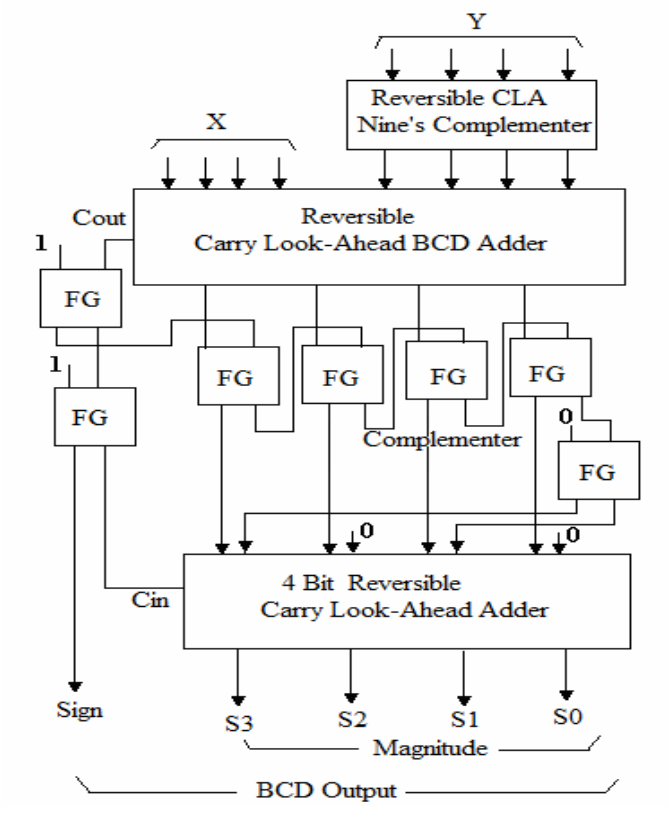


Fig. 25. Proposed Reversible Carry Look-Ahead BCD Subtractor

8.1 Reversible Carry Skip BCD Adder

Figure 26 shows the block diagram of the reversible carry skip adder block constructed with TSG gate, Toffoli gate (TG), Fredkin gate (F) and New gate (NG). *In this work, we have minimized the number of reversible gates and garbage output by adopting various strategies, and designed the reversible carry skip BCD adder with 12 reversible gates and 15 garbage output.* The first strategy is to introduce the 6 input Toffoli gate in the middle of Fig.26 to perform the operation P & Cin, where P is block propagate signal (P=p[0]&p[1]&p[2]&p[3]) and Cin is the input carry. This will minimize the garbage to 5 (if three input Toffoli gates were used for performing P&Cin operation, the garbage count will be 8). Referring to Fig.26, we have efficiently regenerated the '*Cin'* at the garbage output of the 1st full adder (TSG gate), which helps in avoiding the garbage as well as the fan-out problem (the garbage of this TSG gate is reduced to zero). Referring to Fig.9, the generation of Cout as P1+C4+So3 (So2+So1) is done using 4 input MTG gate, where So3(So2+So1) is generated using two NG gates. MTG can implement 3 input OR function with bare minimum of 3 garbage output. Thus, the proposed reversible carry skip BCD adder only has 14 garbage output (that is, with only one more garbage output compared to

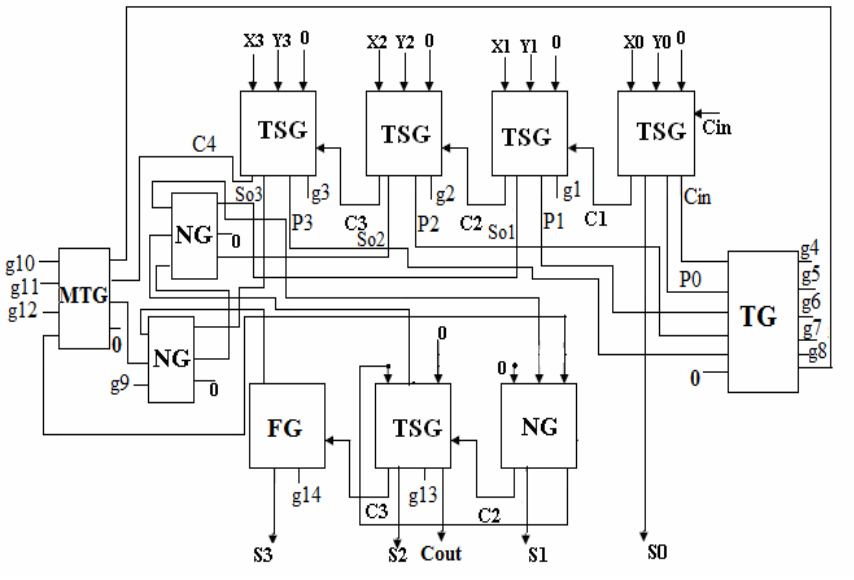


Fig. 26. Reversible Logic Implementation of the Carry Skip BCD Adder

the efficient design of reversible conventional BCD adder proposed in this work). Furthermore, the proposed carry skip BCD adder will be faster due to its carry skipping property.

8.2 Reversible Carry Skip BCD Subtractor

Figure 27 shows the reversible implementation of the proposed carry skip BCD subtractor. It is to be noted that the proposed work is the maiden attempt to design a reversible carry skip BCD subtractor. In the proposed reversible implementation, the reversible nine's complementer can be chosen from the nine's complementer that we designed, as shown in the above sections. The other component, *the reversible carry skip BCD adder*, is already shown in Fig.26. Another interesting component in Fig.27 is the reversible implementation of the bottom 4-bit carry skip adder block, which we have designed with 6 reversible gates and 7 garbage output. Thus, the proposed reversible carry skip BCD subtractor is an efficient design in terms of the number of reversible gates and garbage output. *It led us to conclude that utilizing the garbage output for regenerating the constant input like '1' and '0' will significantly help in reducing the garbage output. This can be considered as the indirect contribution of this paper to the reversible logic community.*

9 Experimental Results

All the BCD adders and subtractors are coded in Verilog HDL for functional verification. The designs are synthesized in Xilinx VirtexE FPGA using Xilinx 9.1 for understanding the delay and area parameters [20]. FPGA synthesis results show that

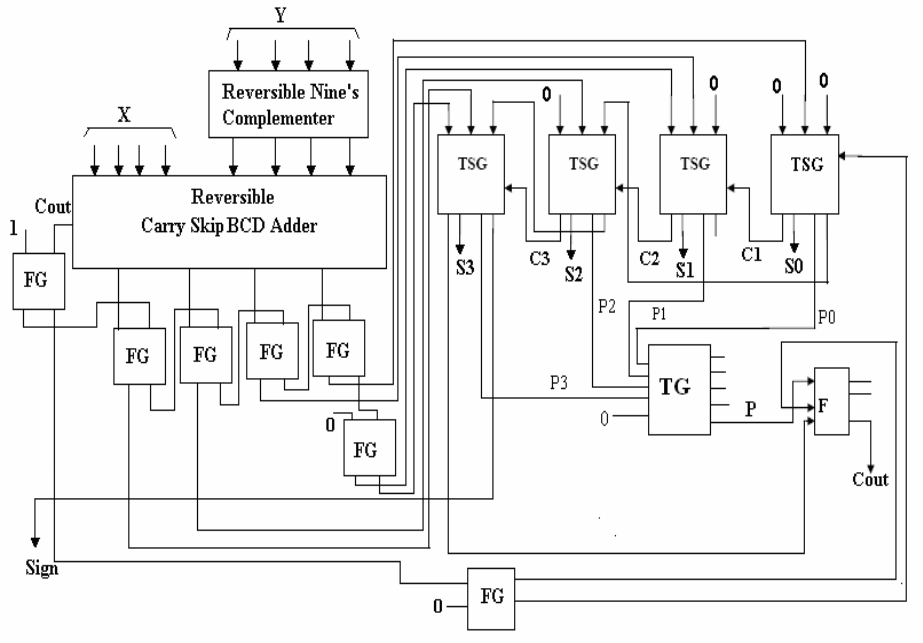


Fig. 27. Reversible Logic Implementation of the Carry Skip BCD Adder

the conventional BCD adder has a propagation delay of 12.441ns with a cell usage of 12. The CLA BCD adder has a propagation delay of 11.619ns with a cell usage of 29. The carry skip (CAS) BCD adder has a cell usage of 21, with a propagation delay of 7.209ns when there is a carry skip. Otherwise, the delay is nearly the same as that of the conventional BCD adder. The conventional modified BCD subtractor has a propagation delay of 16.029ns with a cell usage of 17. The CLA BCD subtractor has a propagation delay of 14.952ns with a cell usage of 37. The carry skip (CAS) BCD subtractor has a cell usage of 24 and propagation delay of 7.553ns when there is carry skipping; otherwise, the delay is the same as that of the conventional modified BCD subtractor. Tables 2 and 3 summarize the FPGA synthesis results for BCD adders and BCD subtractors, respectively. It can be observed that CLA BCD architectures are fastest compared to other designs, but consumes more area. Carry skip designs seem to be the attractive choice when there is an area constraint, and we require the propagation delay better than the conventional BCD subtractors (the propagation delay will be the same as that of the conventional BCD subtractor except when there is carry skipping).

The reversible logic implementation of the BCD adders and subtractors are only functionally verified due to lack of proper technology to implement the reversible gates. One of the existing ways of implementing the reversible gate is using r-MOS technology [21,22]. r-MOS circuits make use of more transistors compared to CMOS circuits for implementing a design, hence dissipating more power. Thus, r-MOS circuits show that it is conceptually possible to implement the reversible gate in MOS transistors. However, they do not guarantee less power consumption compared to

Table 2. Synthesis Results of BCD Adders

	Conventional BCD Adder	CLA BCD Adder	CAS BCD Adder
Delay	12.641 ns	11.619ns	7.20ns* * Carry Skipping
Area (Cell Usage)	12	29	21

Table 3. Synthesis Results of BCD Subtractors

	Conventional BCD Subtractor	CLA BCD Subtractor	CAS BCD Subtractor
Delay	16.029 ns	14.952ns	7.553ns* * Carry skipping
Area (Cell Usage)	17	37	24

CMOS designs due to resistive contributions. To verify this, we have designed various reversible full adders using a combination of existing reversible gates in r-MOS, and compared their power dissipation with CMOS full adders using HSPICE tool [23] in 0.35um TSMC technology. SPICE simulations have proven that r-MOS reversible full adders consume more power than CMOS full adders due to the resistive requirement of MOS technology. Implementing reversible designs in r-MOS technology is equivalent to a functional verification, which can also be done in Verilog HDL. We have built a library of reversible gates in Verilog HDL and used it to code the proposed designs of reversible BCD adders and BCD subtractors. The functional verification is done using the Active HDL simulator [24], which checks the correctness of our proposed designs.

10 Conclusions

In this work, we have proposed novel carry look-ahead and carry skip BCD subtractors based on novel designs of carry look-ahead and carry skip BCD adders, respectively. The architectures are especially designed to make them suitable for reversible logic implementation. We have shown the reversible logic designs of the modified conventional BCD subtractor (also proposed in this work), as well as the proposed carry look-ahead and carry skip BCD architectures, efficient in terms of the number of reversible gates and garbage output. As far as existing literature and our knowledge are concerned, this work is the maiden attempt to design carry look-ahead and carry skip BCD subtractors and provide their reversible logic implementation. All the designs are functionally verified using Verilog HDL and synthesized using Xilinx VirtexE FPGA. In a nutshell, this paper provides the initial direction toward building more complex systems which can execute more complicated operations using reversible BCD arithmetic units.

Acknowledgments

We would like to thank Yvan Van Rentergem and Alexis De VoS, Universiteit Gent, Belgium for helping out in the r-MOS implementation and providing their views to help us understand the power dissipation in r-mos technology. We also thank anonymous reviewers for their suggestions on improving this manuscript.

References

1. Cowlishaw, M.F.: Decimal Floating-Point: Algorithm for Computers. In: Proc. 16^{th} IEEE Symposium on Computer Arithmetic, pp. 104–111 (2003)
2. Landauer, R.: Irreversibility and Heat Generation in the Computational Process. IBM Journal of Research and Development 5, 183–191 (1961)
3. Bennett, C.H.: Logical Reversibility of Computation. IBM J. Research and Development, 525–532 (1973)
4. Perkowski, M., et al.: Regular Realization of Symmetric Functions using Reversible Logic. In: Proc. Euro-Micro., pp. 245–252 (2001)
5. Maslov, D.: Reversible Logic Synthesis. PhD. Thesis, University of New Brunswick, Canada (2003)
6. Gupta, P., Agarwal, A., Jha, N.K.: An Algorithm for Synthesis of Reversible Logic Circuits. IEEE Trans. Computer-Aided Design 25(11), 2317–2330 (2006)
7. Patel, K., Markov, I., Hayes, J.: Optimal Synthesis of Linear Reversible Circuits. Quantum Information and Computation 8(3-4), 282–294 (2008)
8. Jain, R.P.: Modern Digital Electronics, pp. 206–207. Tata McGraw Hill, New York (2003)
9. Schmookler, M.S., Weinberger, A.W.: High Speed Decimal Addition. IEEE Trans. Computers C-20, 862–867 (1971)
10. Erle, M.A., Schulte, M.J.: Decimal Multiplication Via Carry-Save Addition. In: Proc. of the Application-Specific Systems, Architectures, and Processors (ASAP 2003), pp. 348–359 (2003)
11. Pai, Y.T., Chen, Y.K.: The Fastest Carry Look ahead Adder. In: Proc. of the Second IEEE International Workshop on Electronic Design, Test and Applications (DELTA 2004), pp. 434–436 (2004)
12. Fredkin, E., Toffoli, T.: Conservative Logic. Int. J. Theor. Phys. 21(3–4), 219–253 (1982)
13. Toffoli, T.: Reversible Computing. Tech memo MIT/LCS/TM-151, MIT Lab for Computer Science (1980)
14. Khan, M.M.H.A.: Design of Full adder with Reversible Gates. In: Proc. of International Conf. on Computer and Information Technology, pp. 515–519 (2002)
15. Peres, A.: Reversible Logic and Quantum Computers. Physical Review A 32, 3266–3276 (1985)
16. Thapliyal, H., Srinivas, M.B.: A Novel Reversible TSG Gate and Its Application for Designing Reversible Carry Look Ahead Adder and Other Adder Architectures. In: Srikanthan, T., Xue, J., Chang, C.-H. (eds.) ACSAC 2005. LNCS, vol. 3740, pp. 805–817. Springer, Heidelberg (2005)
17. Thapliyal, H., Srinivas, M.B.: Novel Design and Reversible Logic Synthesis of Multiplexer Based Full Adder and Multipliers. In: Proc. 48th IEEE MIDWEST Symposium on Circuits and Systems (MWSCAS 2005), pp. 1593–1596 (2005)
18. Babu, H.M.H., Chowdhury, A.R.: Design of a compact reversible binary coded decimal adder circuit. Journal of Systems Architecture 52(5), 257–314 (2006)

19. Thapliyal, H., Vinod, A.P.: Design of reversible sequential elements with feasibility of transistor implementation. In: Proc. ISCAS 2007, pp. 625–628 (2007)
20. ISE WebPACK Software, http://www.xilinx.com/ise/logic_design_prod/webpack.htm
21. Desoete, B., De Vos, A.: A reversible carry-look-ahead adder using control gates. Integration: the V.L.S.I. Journal 33, 89–104 (2002)
22. Van Rentergem, Y., De Vos, A.: Optimal design of a reversible full adder. Journal of Unconventional Computing 1, 339–355 (2005)
23. HSPICE, http://www.synopsys.com/products/mixedsignal/hspice/hspice.html
24. Active HDL, http://www.aldec.com/

Missing Data Analysis: A Kernel-Based Multi-Imputation Approach

Shichao Zhang[1,2,3], Zhi Jin[4], Xiaofeng Zhu[1], and Jilian Zhang[1]

[1] College of CS and IT, Guangxi Normal University, China
[2] Faculty of EIT, UTS, P.O. Box 123, Broadway NSW 2007, Australia
[3] State Key Lab for Novel Software Technology, Nanjing University, PR China
[4] School of EE and CS, Peking University, PR China
zhangsc@it.uts.edu.au, zhijin@amss.ac.cn,
zhu0011@21cn.com, zhangjilian@yeah.net

Abstract. Many missing data analysis techniques are of single-imputation. However, single-imputation cannot provide valid standard errors and confidence intervals, since it ignores the uncertainty implicit in the fact that the imputed values are not the actual values. Filling in each missing value with a set of plausible values is called multi-imputation. In this paper we propose a kernel-based stochastic non-parametric multi-imputation method under MAR (Missing at Random) and MCAR (Missing Completely at Random) missing mechanisms in nonparametric regression settings. Furthermore, we present a kernel-based stochastic semi-parametric multi-imputation method while we have some priori knowledge about the dataset with missing. Our algorithms are designed specifically with the aim of optimizing the confidence-interval and the relative efficiency. The proposed technique is evaluated by experimentations, using simulation data and real data, and the results demonstrate that our method performs much better than the NORM method, and is promising.

1 Introduction

In machine learning and data mining applications, according to (Cios and Kurgan, 2002), about 20% of the effort is spent on the problem of data understanding, about 60% on data preparation and about 10% on data mining and analysis of knowledge, respectively. Data preparation needs more than half of the project effort. This is because data in real world applications can often be incomplete, redundant, inconsistent, or with noisy. A main kind of incomplete data is missing data. In fact, incomplete information can be caused by error, equipment failure, change of plans, and so on. Missing values presented in a dataset are a common fact in real world applications and, further than, it may generate bias in the data, affecting the quality of the supervised learning process or the performance of classification algorithms. Most learning algorithms are not well adapted to some application domains due to the difficulty with missing data (for example, Web applications). That implies that a reliable method for dealing with those missing values is necessary.

There are many approaches to deal with missing values in (Han and Kamber, 2006; Zhang, et al., 2005): (a) Ignore objects containing missing values; (b) Fill missing

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 122–142, 2009.

values manually; (c) Substitute missing values by a global constant or the mean of the objects; (d) Get the most probable value to fill in the missing values. These methods usually bias the data and the imputed value may not be correct. For instance, one popular method, mean substitution, can result in a distribution with truncated variance. The method of imputation, however, is a popular strategy. In comparison to other methods, it uses as many information as possible from the observed data to predict missing value (Zhang, et al., 2008).

Missing data imputation is a procedure that replaces the missing values in a dataset by some plausible values. One advantage of this approach is that the missing data treatment is independent of the learning algorithm used. This allows users to select the most suitable imputation method for their applications. Commonly used imputation methods for missing response values include parametric and non-parametric regression imputations.

Common regression methods include parametric methods (such as, linear regression, nonlinear imputation method) and non-parametric methods (such as, kernel imputation in (Zhang, et al., 2008)). The parametric regression imputations are superior if a dataset can be adequately modeled parametrically, or if users can correctly specify the parametric forms for the dataset. However, such a parametric approach is potentially more sensitive to model violations than methods based on implicit models. If the regression model is not a good fit, then the predictive power of the model might be poor (Qin, et al., 2007). Moreover, we must expense much time to model the real distribution even if we have some idea to know the real distribute of the datasets. Non-parametric imputation method offers a nice alternative if users have no idea on the actual distribution of a dataset because the method can provide superior fits by capturing structure in datasets (a mis-specified parametric model cannot). In practice, non-parametric imputation method cannot correctly explain some relation if we have some priori knowledge for the data. For example, there are other relations within real world data, and both parametric imputation method and non-parametric imputation method are not adequate to capture the relations. That is, we know a part of relation between independent variables (condition attributes) and dependent variable (target attribute), e.g., we can regard this relation as parametric model, but we have no knowledge on the relation between other independent variables and dependent variable, e.g., we can take it as nonparametric model. However, combined these two parts, it is difficult for us to consider the compound relation with parametric model or nonparametric model. Moreover, the case is very general in real application. In this paper, we regard the relation containing two models as semi-parametric model or partial parametric model. In real application, semi-parametric model is natural than non-parametric model because users can always know some information but no all on the datasets, such some parameters in the datasets.

Recently, much research on missing data analysis has focused on multi-imputation techniques for addressing the issues in single-imputation (Qin, et al., 2007, Zhang, et al., 2007, Zhang, et al., 2006; Peng and Zhu, 2008; Zhang 2004). Multi-imputation is a simulation-based approach to the statistical analysis of incomplete data first proposed by (Little and Rubin, 2002). In multiple imputation methods, each missing datum is replaced by $m>1$ simulated values. The resulting m versions of the complete data can then be analyzed by standard complete-data methods, and the results are thus combined to produce inferential statements (e.g. interval estimates or p-values) that incorporate missing-data uncertainty.

No matter which complete-data analysis is used, the process of combining results from different data sets is essentially the same. Multiple-imputation does not attempt to estimate each missing value through simulated values but rather to represent a random sample of the missing values. This process results in valid statistical inferences that properly reflect the uncertainty due to missing values; for example, valid confidence intervals for parameters.

This paper designs a kernel-based stochastic non-parametric multiple imputation method under MAR (Missing at Random) and MCAR (Missing Completely at Random) missing mechanisms in nonparametric regression settings, with the aim of optimizing the confidence-interval and the relative efficiency. We also design a kernel-based stochastic semi-parametric multi-imputation method: instead of filling in a single value for each missing value, a multi-imputation procedure replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute (Little et al. 1987) when we have some priori knowledge about the dataset.

The rest of this paper is organized as follows. We briefly recall some related work and describe some experimental results on simulation models and real datasets to compare the performances between our methods and the existed method (such as Norm, kernel-based deterministic imputation method). We summarize this paper in Section 5.

2 Related Work

This section reviews some main techniques for missing data imputation and describes some basic concepts.

2.1 Research into Single-Imputation

Some incomplete data handling methods do a better job of maintaining the distribution than others. However, the most appropriate way to handle missing or incomplete data will depend upon how data is missing. A useful reference for general parametric statistical inferences with missing data is (Little and Rubin, 2002), which classified missing data mechanisms into three categories as follows.

1. **Missing Completely at Random** (MCAR)**:** When given the variables X and Y, the probability of response dependents on X but not on Y.
2. **Missing at Random** (MAR)**:** The probability of response independence exists between X and Y. MCAR data exhibits a higher level if randomness than does MAR.
3. **Non-ignorable:** the probability of response depends on variables X and possibly on variable Y.

In practice it is usually difficult to meet the MCAR assumption. Most missing data methods are applied upon the assumption of MAR. And in correspondence to Kim (2001), “Non-igorable missing data is the hardest condition to deal with, but unfortunately, the most likely to occur as well”. In this paper, our experiments base on the missing mechanisms MAR and MCAR.

Single imputation strategies provide a single estimate for each missing data value. Many methods for imputing missing values are single imputation methods, such as, C4.5 algorithm, kNN method, and so on. We can partition single imputation methods into statistical methods and machine learning ones. The most popular method in statistics is regression imputation methods. Common regression methods include parametric methods (such as, linear regression, nonlinear imputation method) and non-parametric methods (such as, kernel imputation in (Zhang, et al., 2008)). While much work focuses on modeling data by parametric or nonparametric approaches, (Engle et al. 1986) have studied the semi-parametric model. They model the electricity demand y as the sum of a smooth function g of monthly temperature t, and a linear function of x_1 and x_2, as well as 11 monthly dummy variables $x_3, \cdots, x_{13}$, to build a semi-parametric model firstly. In fact, semi-parametric model is more ordinary in real application than nonparametric model or parametric model because we always contain a little but no all information on our datasets, however, there are a little literatures, such as, (Qin, et al., 2007), focusing on this issue because of the analysis complexity, in this paper, we introduce SIIA algorithm to model the partial parametric model for filling up iteratively missing target values.

The presentation methods in machine learning include rough set method, C4.5 method, association algorithm (Zhang W., 2000), etc. Most imputation methods focus on imputing missing attribute values rather than the methods in statistics which pay attention on imputing missing target values. (Karmaker and Kwek, 2005) combines EM algorithm with decision tree method to missing condition attribute values, and (Mostafa, et al. 2007) combines regression imputation method with ensemble method to impute missing values. (Peng and Zhu, 2008) designs imputation method based on rough set.

In semi-parametric, (Millimet, 2003) has shown with data for US states that parametric modeling can be rejected in favor of a semiparametric estimator, which does not impose any a priori restriction on the functional form of the relationship. Pickle et al. (2005) compares the parametric and non-parametric methods, present a semiparametric for modeling which combine parametric and non-parametric function to improve the quality of both the mean and variance models. The resulting semi-parametric estimates have smaller bias and variance and result in a better understanding of the process at hand. The above methods about semi-parametric imputation method handle the complete data. In fact, real-world data often includes some form of missing data. (Qin, et al. 2007) develops a kernel regression model under the assumption of nonparametric model.

A disadvantage of single imputation strategies is that they tend to artificially reduce the variability of characterizations of the imputed dataset. The alternatives are to fill in the missing values with multiple imputation methods (e.g., Multiple Imputation (MI)). In multivariate analysis, MI methods provide good estimations of the sample standard errors. However, data must be missing at random in order to generate a general-purpose imputation.

2.2 Research into Multi-Imputation

The theoretical underpinnings of multi-imputation are Bayesian. The central idea is to fill in the missing values by drawing from the posterior predictive distribution of the

missing data given the observed data. The procedure is independently repeated M times. Each filled-in dataset is analyzed separately and the results combined following well-established rules. (Little and Rubin, 2002)'s multiple imputation is a three-step method for handling complex missing data.

At the first step, m (> 1) completed-data sets are created by imputing the unobserved data m times using m independent draws from an imputation model, which is constructed to reasonably approximate the true distributional relationship between the unobserved data and the available information, and thus reduce potentially very serious non-response bias due to systematic difference between the observed data and the unobserved ones. At the second step, m complete data analyses are performed by treating each complete data set as a real complete-data set, and thus standard complete-data procedures and software can be utilized directly. At the last step, the results from the m complete-data analyses are combined in a simple, appropriate way to obtain the so-called repeated-imputation inference, which properly takes into account the uncertainty in the imputed values.

The multi-imputation (MI) procedure (such as (Yuan, 2001), SAS, S-plus) provides three methods for imputing missing values and the choice of methods depends on the type of missing data pattern. These methods are Regression method, Propensity Scores method and Markov chain Monte Carlo (MCMC) method. (Scheffer, 2002) shows how the mean and standard deviation are affected by different methods of imputation, given different missing mechanisms. Better options than the standard default options are available in the major statistical software, offering the chance to do the right thing to the statistical and non-statistical community alike. (Zhang, 2004) propose a method for the multivariate data under the ANOVA model, where both the hot-deck and ABB methods run into difficulties. (Faris et al, 2002) compared approaches in dealing with missing data. Three multiple imputation methods are compared with a method of enhancing a clinical database through merging with administrative data. The different methods produced similar results, with one of the multiple imputation methods demonstrating a slight advantage. It is concluded that the choice of missing data strategy should be guided by statistical expertise and data resources. (Allison, 2000) has evaluated two algorithms for producing multiple imputations or missing data using simulated data about software of SOLAS. Software using a propensity score classifier with the approximate Bayesian boostrap produces badly biased estimates of regression coefficients when data on predictor variables are MAR or MACR. Allison has also showed that list-wise deletion produces un-bias regression estimates whenever the missing data mechanism depends only in the predictor variable, not on the response variable. (Kang, et al., 2007) thinks the MI imputation method is more efficient than single imputation, and can handle auxiliary variables as well as can be made robust against the failure of the imputation model. The experimental results in (Peng and Zhu, 2008) show MI perform better than EM algorithm.

2.3 Our Contribution

In this article, in the first step, we build a kernel-based stochastic non-parametric imputation (detailed in section 3.1) and a kernel-based random semi-parametric imputation model (detailed in section 3.2) as imputation model. In the second step, we

impute the non-response in the dataset and obtain m 'complete' data sets based on the imputation model constructed in section 3.1 or section 3.2. In the third step, we use two methods introduced in section 3.4 based on the theory of (Little and Rubin, 2002) to analyze the m complete-data for verifying the efficiency of proposed algorithms and the experimental results are presented in section 4. The contribution of the paper is presented as follows:

Different from existing single imputation method, our algorithms impute missing multiple times. That can avoid efficient the variability of the imputed dataset in single imputation. Furthermore, our methods are based on the nonparametric model and semi-parametric model which is the practical model in real application.

Different from existing MI techniques in which the parametric model (i.e., Bayes model) is based model, one of our two methods is a stochastic regression multiple imputation method in nonparametric settings. Due to the complex structures of data sets, it is hard or impossible to precisely describe the regression relationship between the response variable and the covariates. In this case, the usual (parametric) regression imputation method cannot be used. In particular, the kernel-based stochastic multiple imputation method aims at optimizing the confidence-interval and the relative efficiency. Furthermore, we propose an efficient random/stochastic semi-parametric regression imputation under the missing mechanisms MCAR and MAR under the circumstance of having a little prior knowledge about the dataset.

3 Proposed Algorithm

3.1 Kernel-Based Non-parametric Imputation Method

Let X be a d-dimensional vector of factors and let Y be a response variable influenced by X. In practice, one often obtains a random sample (sample size = n) of incomplete data associated with a population (X,Y,δ), (X_i,Y_i,δ_i), i = 1,2,…,n. Where all the X_i's are observed and $\delta_i = 0$ if Y_i is missing, otherwise $\delta_i = 1$. Suppose that (X_i,Y_i) 's satisfy the following model:

$$Y_i = g(X_i) + \varepsilon_i \text{ , } i=1,2,\ldots,n, \tag{3.1.1}$$

Where $g(.)$ is an unknown function, and the unobserved ε_i (with population ε) are i.i.d.(independent identically distributed) random errors with mean 0 and unknown finite variance σ^2, which assure the convergence of our algorithm same as the deterministic imputation (Wang, Q. & Rao, J. 2002), and independent of the i.i.d. random variables X_i's.

Let $r = \sum_{i=1}^{n} \delta_i, m = n - r$. Denote the sets of respondents and non-respondents as s_r and s_m, respectively. Let K be a symmetric probability density function and let $h = h_n$ be a bandwidth sequence that decreases toward 0 as the sample size n increases toward ∞. Use $\hat{g}(x)$ to denote the kernel estimator for $g(x)$ ($x \in R^d$) based on the completely observed pairs (X_i,Y_i), i.e.,

$$\hat{g}_n(x)=\frac{\sum_{i=1}^{n}\delta_i Y_i K(\frac{x-X_i}{h})}{\sum_{i=1}^{n}\delta_i K(\frac{x-X_i}{h})+n^{-2}} \tag{3.1.2}$$

Where the term n^{-2} is introduced to avoid the case that the denominator vanishes; *K(.)* is called kernel function. There are some widely used kernel functions in non-parametric inference. For instance, the Gaussian kernel (standard normal density function) $K(\cdot)=(2\pi)^{-1/2}\exp(-x^2/2), x\sim N(1,1)$, and a polynomial kernel:

$$K(.)=\begin{cases}\frac{15}{16}(1-t^2+t^4), & |t|\le 1\\ 0, & otherwise\end{cases}$$

In practice, there is no any significant difference using these kernel functions. In this paper, we use the Gaussian kernel in our experiments of non-parametric multiple imputation, we will discuss the choosing of bandwidth in kernel method later in section 3.3.

Let $Y_i^{(R)}, i\in s_m$ be the imputed values for the missing data based on random imputation method, we uses $Y_i^{(R)}=\hat{g}_n(X_i)+\varepsilon_i^*$, $i\in s_m$, as the imputed values in our non-parametric imputation method, which have same characteristic of convergence as deterministic imputation method, where ε_i^* is a simple random sample of size *m* with replacement from $\{Y_j-\hat{m}_n(X_i), j\in s_r\}$. Denote $Y_{R,i}=\delta_i Y_i+(1-\delta_i)Y_i^{(R)}, i=1,\cdots,n,$ which are 'complete' data based on the above imputations.

3.2 Constructing a Semi-parametric Regression Model

A general semi-parametric regression model is as follows.

$$Y_i = X_i^T\beta + g(T_i) + \varepsilon_i. \tag{3.2.1}$$

Where the $Y_i's$ are iid (independent identically distributed) scalar response variables, the $X_i's$ are iid d-dimensional random covariate vectors, the $T_i's$ are iid d^*-variable random covariate vectors, the function $g(\cdot)$ is unknown, and the model error ε_i are iid random errors with mean 0 and unknown finite variance σ^2.

Let $r=\sum_{i=1}^{n}\delta_i$, $m=n-r$. Denote the sets of respondents and non-respondents as s_r and s_m, respectively. Let *K* be a symmetric probability density function and let $h=h_n$ be a bandwidth sequence that decreases toward 0 as the sample size *n* increases toward $+\infty$. In this section, we define the estimators that we will analyze in this article. From (3.2.1), we have:

$$Y_i - X_i^T \beta = g(T_i) + \varepsilon_i. \quad i = 1, \cdots, r \tag{3.2.2}$$

Assuming β is known, we have a kernel estimator $\hat{g}(t)$ for $g(t)$ based on the completely observed data:

$$\hat{g}(t) = \frac{\sum_{j=1}^{n} \delta_j K\left(\frac{(t - T_j)}{h}\right)(Y_j - X_j \beta)}{\sum_{j=1}^{n} \delta_j K\left(\frac{(t - T_j)}{h}\right) + n^{-2}}. \quad i = 1, \cdots, r. \tag{3.2.3}$$

Where, the term n^{-2} is introduced to avoid the case that the denominator vanishes. $K(.)$ is a kernel function. In this paper, we use the polynomial kernel in our semi-parametric experiments. We will discuss the choosing of bandwidth in kernel method later in Section 3.3.

Using $\hat{g}(T_i)$ to replace $g(T_i)$ in (3.2), we obtain:

$$Y_i - X_i^T \beta \approx \frac{\sum_{j=1}^{n} \delta_j K\left(\frac{(T_i - T_j)}{h}\right)(Y_j - X_j \beta)}{\sum_{j=1}^{n} \delta_j K\left(\frac{(T_i - T_j)}{h}\right) + n^{-2}}, \quad i \in s_r. \tag{3.2.4}$$

Converting (3.2.4), we have

$$Z_i \approx U_i^T \beta, \quad i \in s_r. \tag{3.2.5}$$

Where

$$Z_i = Y_i - \frac{\sum_{j=1}^{n} \delta_j Y_j K\left(\frac{(T_i - T_j)}{h}\right)}{\sum_{j=1}^{n} \delta_j K\left(\frac{(T_i - T_j)}{h}\right) + n^{-2}}, \quad U_i = X_i - \frac{\sum_{j=1}^{n} \delta_j X_j K\left(\frac{(T_i - T_j)}{h}\right)}{\sum_{j=1}^{n} \delta_j K\left(\frac{(T_i - T_j)}{h}\right) + n^{-2}}, \quad i \in s_r. \tag{3.2.6}$$

According to the theory of linear regression model, β is estimated by (3.2.7):

$$\hat{\beta}_n = \left(\sum_{i=1}^{n} \delta_i U_i U_i^T\right)^{-1} \left(\sum_{i=1}^{n} \delta_i U_i Z_i\right). \tag{3.2.7}$$

Combining with (3.2.3), the final estimator for $g(t)$ is given by

$$\hat{g}_n(t) = \frac{\sum_{j=1}^{n} \delta_j K\left(\frac{(t-T_j)}{h}\right)(Y_j - X_j \widehat{\beta_n})}{\sum_{j=1}^{n} \delta_j K\left(\frac{(t-T_j)}{h}\right) + n - 2}. \tag{3.2.8}$$

Let $Y_i^{(D)}$ and $Y_i^{(R)}$, $i \in s_m$ be the imputed values for the missing data based on deterministic and random imputation methods, respectively. Deterministic imputation (Wang & Rao 2002) uses $\hat{g}_n(T_i)$ as the imputed value, i.e.

$$Y_i^{(D)} = X_i^T \hat{\beta}_n + \hat{g}_n(T_i), i \in s_m .$$

We construct the random imputation $Y_i^{(R)} = X_i^T \hat{\beta}_n + \hat{g}_n(T_i) + \varepsilon_i^* = Y_i^{(D)} + \varepsilon_i^*$, $i \in s_m$. as the imputed values, which have same convergence as the deterministic method, where ε_i^* is a simple random sample of size m with replacement from $Y_i - X_i^T \hat{\beta}_n - \hat{g}_n(T_i), i \in s_r$.

Denote

$$Y_{D,i} = \delta_i Y_i + (1-\delta_i) Y_i^{(D)},$$

$$Y_{R,i} = \delta_i Y_i + (1-\delta_i) Y_i^{(R)}, i = 1, ..., n$$

Which are 'complete' data based on the above imputations.

3.3 The Choice of the Bandwidth and the Algorithm Complexity Analysis

Kernel method can be decomposed into two parts: one for the calculation of the kernel and another for bandwidth choice. (Qin, et al., 2007) states that one important factor in reducing the computer time is the choice of a kernel that can be calculated very quickly. Having chosen a kernel that is efficient to compute, one must then choose the bandwidth. (Qin, et al., 2007) turn out that the choice of window width is much more important than the choice o f kernel function. Small value of h make the estimate look 'wiggly' and show spurious features, whereas to big values of h will lead to an estimate which is too smooth in the sense that it is too biased and may not reveal structural features. There is no generally accepted method for choosing the window widths. Methods currently available include 'subjective choice' and automatic methods such as the "plug-in", 'cross-validation' (CV), and 'penalizing function' approaches. In this paper, we use the method of cross-validation to minimize the

approximate mean integrated square error (AMISE) of $\hat{g}(x_i)$ for a given sample of data. Define the CV function to be

$$CV = \sum_{i=1}^{n} (y_i - \hat{g}(x_{-i}, c))^2$$

Where $\hat{g}(x_{-i},c)$ denotes the 'leave-one-out' estimator evaluated for a particular value of c. $\hat{g}(x_{-i},c)$ is obtained by omitting the realization (x_i,y_i) from the estimator of $g(.)$ at the point x_i.

Generally, before imputing, all numeric data are normalized into [0, 1] to avoiding the bias that the result is usually prone to the data with bigger magnitude. While the complexity of the kernel method is $O(mn^2)$, where n is the number of instances of the dataset, m is the number of attributes, so the algorithm complexity of our two methods are $O(mn^2)$.

3.4 Combing Inferences from Imputation Data

We use the method derived in (Little and Rubin, 2002) to analyze the performance for our two algorithms. Suppose that our primary interest lies in a scalar Q (in this article, we specify Q as the mean of the response variable); and m complete datasets under the nonparametric regression model are obtained. In each of these data sets we use standard complete-data methods to obtain an estimate of Q with an associated estimated variance. As might be expected, the multiple imputation point estimate of Q is the average of the m complete-data estimates:

$$\bar{Q}=\frac{1}{m}\sum_{t=1}^{m}\hat{Q}^{(t)} \tag{3.4.1}$$

In the usual way, a $100(1-\alpha)\%$ interval estimate for Q based on (Barnard and Rubin 1999) is

$$\bar{Q}\pm t_{1-\alpha/2,\gamma^*}\times\sqrt{T} \tag{3.4.2}$$

Another useful statistic about the non-response is the fraction of missing information about Q:

$$\lambda=\frac{r+2/(\gamma+3)}{r+1}\ ,\ (\text{where } r=\frac{(1+1/m)B}{W}) \tag{3.4.3}$$

The relative efficiency of using the finite m imputation estimator, rather than using an infinite number for the fully efficient imputation, in units of variance, is approximately a function of m and λ.

$$RE=(1+\frac{\lambda}{m})^{-1} \tag{3.4.10}$$

Below Table 1 shows the relative efficiencies with different values of mm and λ. For cases with little missing information, only a small number of imputations are necessary for our MI analysis.

Table 1. Relative efficiencies (RE) with different values of m and λ

λ		
m	10%	70%
3	0.9677	0.8108
10	0.9901	0.9662

4 Experiments

In order to show the effectiveness of the proposed method, extensive experiments were done on simulation models as well as real dataset using a DELL Workstation PWS650 with 2G main memory, 2.6G CPU, and WINDOWS 2000.

4.1 Experimental Study for Non-parametric Multi-Imputation

4.1.1 Simulations

In this section, we introduce some of our experiments to evaluate the performances of the proposed method in making inference for the mean (Q) of the response variable. We compare the performances of our non-parametric method with the NORM according to their coverage probabilities and average lengths of confidence intervals, as well as their relative efficiencies. Throughout this section, we set α=0.05.

The NORM is a Windows 95/98/NT program for multiple-imputation (MI) of incomplete multivariate data downloaded from (Schafer, 1999). It creates multiple imputations by an algorithm called data augmentation (DA), a special kind of Markov chain Monte Carlo (MCMC) technique. NORM is not designed to replace well-established statistical packages like SAS or SPSS and does not perform statistical analyses (e.g. linear or logistic regression).

We use the model $y = x_1^2 + \sin x_2 + \varepsilon$, with $x_i (\, i = 1, 2)$ from the normal distribution N(1, 1) and ε from N(0, 1). The following two cases of response probabilities under the MAR and MCAR assumptions are considered (Qin, et al, 2007):

Case1 (MAR): $P_1(x) = P(\delta = 1 \,|\, X = x_1, X = x_2) = 0.8 + 0.2\,|x_1 - 1|\,|x_2 - 1|$, *if* $|x_1 - 1|\,|x_2 - 1| \le 1$, and =0.95, elsewhere.

Case2 (MCAR): $P(x)=P(\delta=1 \,|\, X=x_2, X=x_2)=0.9$, for all x_1, x_2 respectively.

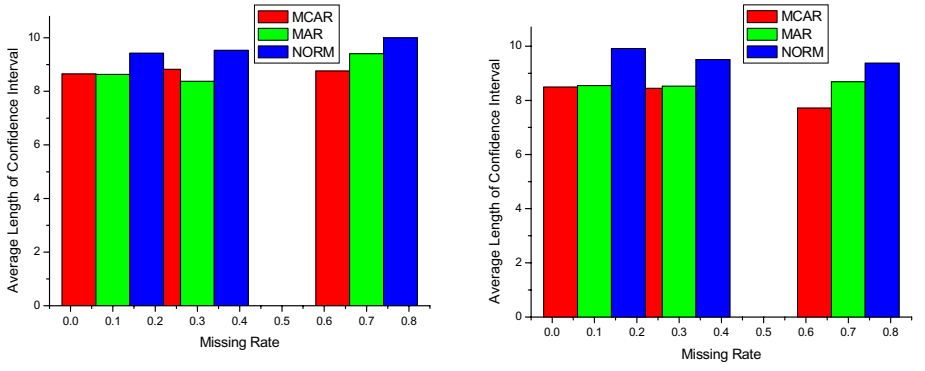


Fig. 1. The result of average length of confidence interval for our kernel-based stochastic non-parametric multi-imputation were compared with the NORM. Sample Size is 1000, Missing Rate is 10%, 30%, 70% respectively, the repeat time is 3 in left figure, and the right's is 5. method under MCAR and MAR compare with NORM.

Table 2. The result of coverage probability for our kernel-based stochastic non-parametric multi-imputation were presented with NORM. Sample Size is 1000, Missing Rate is 10%,20%,70% respectively.

	Repeat Time:3			Repeat Time:10		
MR	MCAR	MAR	NORM	MCAR	MAR	NORM
10%	0.941	0.937	0.918	0.942	0.947	0.925
30%	0.938	0.935	0.919	0.941	0.944	0.919
70%	0.935	0.933	0.926	0.939	0.943	0.918

We obtain the confidence interval from the m 'complete 'data sets according to formula (3.4.2) and then compare the CP and AL (coverage probability and the average length of intervals).

Figure 1 and Table 2 reveal the following results:

1. When the missing rate is relatively small (for example, 10%), the confidence interval based on our algorithm under both of MCAR and MAR perform almost uniformly better than those based on NORM as shown in Table 2, i.e., the CPs based on our algorithms are closer to the nominal level 95% than the CP based on NORM, and the ALs are shorter based on our algorithm than the AL based on NORM in Figure 1.
2. When the missing rate is relatively large(for example, 30% or 70%), the confidence interval based on our algorithm under both of MCAR and MAR perform still better, but not as significantly as the case when missing rate is small,

than those based on NORM as shown in Table 2, that is to say, the CPs based on our algorithm are closer to the nominal level 95% than the CP based on NORM, and the Als are shorter based on our method than the AL based on NORM for all most all C.

We also compare the performances of our method and the NORM in terms of the RE (Relative Efficiency) according to the formula (3.4.3). Table 3 shows the Relative Efficiency in different values of m (repeat times) and λ, in which 10% ,30% and 70% are the missing rates.

Table 3. The result of Relative Efficiency for our kernel-based stochastic non-parametric multi-imputation were compared with NORM. Sample Size is 1000, Missing Rate is 10%,30%,70% respectively.

	Repeat Time:3			Repeat Time:10		
MR	MCAR	MAR	NORM	MCAR	MAR	NORM
10%	0.999983	0.999998	0.9996	1	1	0.999829
30%	0.999983	0.999994	0.99978	0.999973	0.999991	0.99977
70%	0.99976	0.99987	0.995	0.999985	0.999739	0.992

From Table 3 comparing with the standard in Table 1, we can see that the performances of our method and the NORM are similar.

4.1.2 Application in Abalone from UCI

In order to show the effectiveness of our proposed method in making inference for the mean of a population, we conducted some experiments on the real dataset *abalone*, which is downloaded from the UCI machine-learning repository (Blake & Merz 1998). It contains 4177 instances in total and 9 attributes (sex, length, diameter, height, whole weight, shucked weight, viscera weight, shell weight, rings) for each instance, in which there are no missing values. These attributes are used to predict the age of abalone. Obviously, the relation between the age and these attributes is MAR. But we also have experiments for the data set about MACR since we want to show the difference among the three. (Schafe, 1997) has argued that the use of a rich multivariate data set can provide protection against violations of the MAR assumption, and can minimize the biases incurred where the assumption is violated. It may also be possible to guard against violations of the MAR assumption by combining data merging and multiple imputation methods. So we select the other attributes (except the "sex" who is a nominal) to predict.

We conduct the experiments under missing rates 10%, 30% and 70%, with repeated times 3 and 10. We randomly select 1000 instances from 4177 because the maximum instance that NORM can only handle is 2000.

From these experiments, we can see that our method performs better than the NORM similar to the findings in previous simulation study.

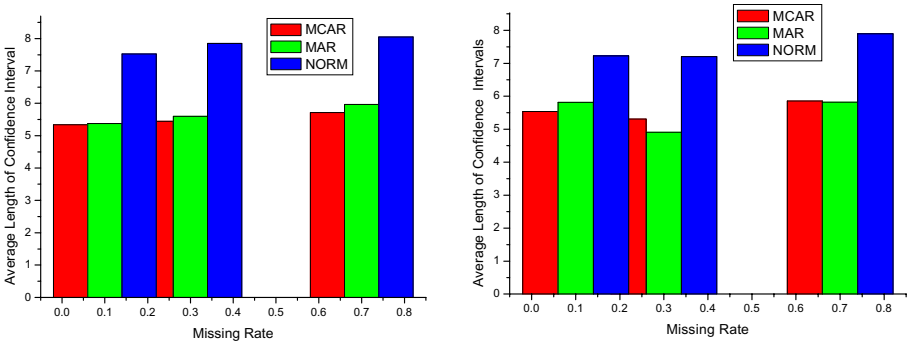


Fig. 2. Sample Size is 1000, Missing Rate is 10%,30%,70% respectively. The result of AL for our kernel-based stochastic non-parametric multi-imputation were compared with NORM.

Table 4. Sample Size is 1000, Missing Rate is 10%,30%,70% respectively. The result of CP for our kernel-based stochastic non-parametric multi-imputation were presented with NORM.

	Repeat Time:3			Repeat Time:10		
MR	MCAR	MAR	NORM	MCAR	MAR	NORM
10%	0.935	0.941	0.928	0.937	0.939	0.929
30%	0.934	0.938	0.918	0.945	0.935	0.922
70%	0.931	0.937	0.909	0.966	0.9249	0.91

Table 5. Sample Size is 1000, Missing Rate is 10%,30%,70% respectively. The result of Relative Efficiency for our kernel-based stochastic non-parametric multi-imputation were compared with NORM.

	Repeat Time:3			Repeat Time:10		
MR	MCAR	MAR	NORM	MCAR	MAR	NORM
10%	0.999879	0.99999	0.9952	0.999993	0.999998	0.9991
30%	0.999509	0.999969	0.99091	1	0.999999	0.9991
70%	0.999797	0.99988	0.98201	0.999911	0.99982	0.9901

4.2 Experimental Study for Semi-parametric Multi-Imputation

4.2.1 Simulation

In this section, we introduce some of our experiments to evaluate the performances of the proposed method in making inference for the mean (Q) of the response variable.

We compare the performances of our stochastic semi-parametric method with the deterministic semi-parametric method according to their coverage probabilities and average lengths of confidence intervals, as well as their relative efficiencies. Throughout this section, we set $\alpha=0.05$.

According to (3.2.8), we used model:

$$\beta = 1.5,$$

and

$$g(t) = 3.2t^2 - 1 \; if \; t \in [0,1], \quad g(t) = 0, otherwise.$$

We generated $X_i s$ from the normal distribution $N(1, 1)$ and $\varepsilon_i s$ from the standard normal distribution $N(0, 1)$, and the following two cases of response probabilities under the MAR and MCAR assumptions from (Wang & Rao, 2002a):

Case 3 (MAR):

$P_1(x,t)=P(\delta=1|X=x)=0.8+0.2(|x-1|+|t-0.5|)$, *if* $|x-1|+|t-0.5|\le 1$, and =0.95, elsewhere.

Case 4 (MCAR):

$P(\delta=1|X=x,T=t)=0.6$, for all x and t.

For each of the two cases, we generated 1,000 (repeated time) random samples of incomplete data { $X_i, Y_i, \delta_i, i=1,...,n$ } for n=100 from the models and specified response probability function. We get the confidence intervals and AL (the average length of the intervals) from m 'complete 'data sets according to (Little and Rubin, 2002), and then we achieve the CP (coverage probability) after scanning the original data sets.

Table 6. The performance of CP under the assumption MCAR and MAR

	MCAR		MAR	
MR	Deterministic	Stochastic	Deterministic	Stochastic
5%	0.91	0.915	0.905	0.903
10%	0.918	0.924	0.899	0.91
20%	0.894	0.901	0.89	0.895
40%	0.885	0.898	0.882	0.902

Table 6 presents the performance of our two imputation methods in terms of CP with various missing rate 5%, 10% , 20%, 40% under the assumption MCAR and MAR; Table 7 presents the performance of the two imputation methods in terms of AL with various missing rate 5%, 10% , 20%, 40% under the assumption MCAR and MAR;

Table 7. The performance of AL under the assumption MCAR and MAR

	MCAR		MAR	
MR	Deterministic	Stochastic	Deterministic	Stochastic
5%	0.782347	0.776642	0.763287	0.761652
10%	0.804854	0.774798	0.75145	0.744785
20%	0.748518	0.68534	0.699242	0.644894
40%	0.740917	0.738762	0.717476	0.57594

Tables 6 and 7 reveal the following results. The performance of stochastic method under both MCAR and MAR is basically better(closer to the nominal coverage probability 95%) than the deterministic method does in terms of CP or AL with different missing rates, such as 5%,10% ,20% and 40%;

Table 8. The performance of RE under the assumption MCAR and MAR

	MCAR		MAR	
MR	Deterministic	Stochastic	Deterministic	Stochastic
5%	0.99899	0.99905	0.99803	0.99822
10%	0.99965	0.99965	0.99928	0.99928
20%	0.99937	0.99939	0.9988	0.99884
40%	0.99861	0.99872	0.99932	0.99953

Table 8 presents the RE of deterministic and stochastic semi-parametric multiple imputations under MCAR and MAR with different missing rate 5%, 10%, 20% and 40%. Comparing to Table 1, we can see that the RE of deterministic and stochastic imputation methods are much better than the standard values in Table 1, and the stochastic imputation method performs better than the deterministic one with different missing rate.

4.2.2 Real Data Applications

We considered the real data set given in (Qin, et al., 2007). The data give the normal average January minimum temperature in degrees Fahrenheit (Denoted as JanTemp) with the latitude (Lat) and longitude (Long) of 56 U.S. cities. For each year from 1931 to 1960, the daily minimum temperatures in January were added together and divided by 31. Then, the averages for each year were averaged over the 30 years. The data set is also available on: http://lib.stat.cmu.edu/DASL/Datafiles/USTemperatures.html

We suppose the dependent variable is JanTemp and the independent variable is Lat. Our experiment present that the value of significant probability of the correlation

between the JanTemp and Lat is 0 in SPSS, after removing the effects of Lat, we get the value of significant probability of the correlation between the JanTemp and Long is 0.861, these result show there is a obviously linear relationship between JanTemp and Lat, the linear relationship between the JanTemp and Long is not clearly, and (Peixoto , 1990) reports a cubic polynomial in Long is used to predict JanTemp. To apply our method to these real data, we denote the variables for JanTemp, Lat and Long to be Y, X and T respectively. We suppose that Y, X and T satisfy the semi-parametric model (3.2.1).

Note that the original data set given by (Peixoto, 1990) is complete. We used the 56 data and random deleted 6, 14 or 23 Y values (Missing Rate is almost 10%, 20% or 40% respectively) and the repeated times are 1000. The deletion mechanisms are designed to be MAR and MCAR. We make inference on the distribution function $\theta = F(y)$ for fixed y and quantile $\theta_q = F^{-1}(q)$ comparing our semi-parametric regression imputation estimator with non-parametric model and linear model same as Section 4.1. When making inference based on non-parametric kernel regression imputation estimator, the kernel function K(t) is used by (3.1.2) and the deletion mechanism were taken to the same as in Section 4.1.

Due to an obviously linear relationship between JanTemp and Lat, we assume the multiple linear regression among JanTemp, Lat and Long, and then we construct a experiment about multiple linear regression imputation comparing with the non-parametric and semi-parametric model.

Table 9. The performance of CP under the assumption MCAR and MAR

	MCAR				MAR			
MR	Semi4Sto	Semi4Det	Non	Linear	Semi4Sto	Semi4Det	Non	Linear
10%	0.925	0.92321	0.91429	1	0.92321	0.91964	0.90893	1
20%	0.9375	0.91964	0.91429	0.99821	0.92321	0.91964	0.91607	1
40%	0.92143	0.91071	0.90357	1	0.93036	0.92143	0.90536	1

Table 10. The performance of AL under the assumption MCAR and MAR

	MCAR				MAR			
MR	Semi4Sto	Semi4Det	Non	Linear	Semi4Sto	Semi4Det	Non	Linear
10%	4.1550	4.4092	4.4630	9.2959	4.1222	4.3824	4.5638	9.0311
20%	3.8179	4.2325	4.3107	9.1249	3.9516	4.2509	4.5346	8.8035
40%	4.1704	4.2723	4.3417	8.9180	4.0851	4.4714	4.5717	8.9916

Table 9 presents the performance of four imputation methods for CP with missing rates 10%, 20% and 40% under the missing mechanism of MCAR and MAR. Table 10 presents the performance of four imputation methods for AL with missing rates 10%,

20% and 40% under the missing mechanism of MCAR and MAR. Table 11 presents the performance of four imputation methods for RE with missing rates 10%, 20% and 40% under the missing mechanism of MCAR and MAR. The 'Semi4Sto', 'Semi4Deter', 'Non', 'Linear' refer to as the imputation method stochastic semi-parametric regression imputation method, deterministic semi-parametric regression imputation method, non-parametric regression imputation method and linear regression method respectively.

Table 11. The performance of RE under the assumption MCAR and MAR

	MCAR				MAR			
MR	Semi4Sto	Semi4Det	Non	Linear	Semi4Sto	Semi4Det	Non	Linear
10%	1	0.99996	0.99942	0.91173	1	0.99998	0.99994	0.91112
20%	1	0.99999	0.99969	0.97045	0.99988	0.99973	0.99985	0.91128
40%	0.99999	0.99998	0.99863	0.91102	0.99997	0.99994	0.99976	0.91106

Tables 9 to 11 reveal the following facts:

1. From Tables 9 to 11, we can see that the performances of two imputation methods based on semi-parametric models are similar with the simulations results shown before. The stochastic imputation method is basically better than the deterministic imputation method in coverage probabilities (CP), average lengths of confidence intervals (AL) and the relative efficiency (RE). The performances based on the two semi-parametric models are significantly better than the non-parametric model and linear model as there is some linear relation between the covariates and the response variable because the semi-parametric model is capable to capture this relation.
2. From Table 9, we can see that the performance of the CP of linear regression imputation is best than the other three methods, such as, the CP of the linear regression imputation is 1, 0.99821 and 1 under the missing rate 10%, 20% and 40% respectively under the mechanism of MCAR. We can analyze the phenomenon in follow: 1) We constructed a $100(1-\alpha)\%$ interval estimate for Q(mean) where α is the significance level and we set $\alpha=0.05$, i.e. the closer to 95% ,the better the result of CP is, we can find the performance of the other three methods are closer to 95% than the linear regression. 2) the wider the average length of the confidence intervals is, the bigger the CP is, from Table 10, we can know the AL of the linear regression imputation method is widest among all four methods, it show that the performance of the linear regression imputation method is worst among these four methods.
3. Comparing Table 11 with Table 1, we can see that the result of RE of the three methods all beyond the standard in Table 1 under different missing rate (10%, 20%, 40% respectively) under the different missing mechanism (MCAR or MAR) when the repeat time is 10 , but the result of the RE of the linear regression imputation method is worse than the standard because the RE of linear is

0.91773, 0.97045 and 0.91102 respectively, under different missing rate (10%, 20%, 40% respectively) under MCAR (the result is 0.91112, 0.91128, 0.91106 respectively under MAR) in Table 11, and the standard is 0.9901, 0.9804 and 0.9615 in Table 1 when the repeat times are 10.

4. Comparing to missing rate, we can see that the performance of our stochastic semi-parametric regression imputation method is better when the response rate is higher about the performance of the CP, AL and RE under the missing mechanism of MCAR or MAR, and the result show the performance of all four imputation method is better when the missing rate is moderate such as 20%, but the performance of our stochastic imputation method is best among the four imputation methods.
5. We conclude from these results that the best efficient method is our stochastic semi-parametric regression imputation method, then the second is deterministic and non-parametric, and the worst is linear method based on this real data. We also get a conclusion: if we want to construct a inference about multiple imputation, we had better use semi-parametric regression imputation method to patch up the missing value when we have a little information about the missing attribute and the observed attributes, such as we know the linear relationship between the dependent variable and one of the independent variables.

5 Conclusion

There are many reports on single-imputation methods for missing data in data mining and machine learning. In this paper we have designed an algorithm of Kernel-Based Stochastic Nonparametric Multi-imputation to impute the incomplete datasets under MAR and MCAR assumptions when we have little priori knowledge about the dataset, and we can construct a kernel-based stochastic semi-parametric multi-imputation method to patch up the missing value if we have a little priori information about this dataset. We have experimentally evaluated the performances of our methods with the other method (such as NORM, deterministic method) using a simulation dataset and a real dataset. The performances are in terms of the confidence intervals and the Relative Efficiencies based on different imputation methods. It has shown that our methods perform much better than the other methods in terms of confidence intervals, and their relative efficiencies are similarly well. When the missing rate is relatively low, the performance of our method is significant better than the other methods. It is interesting to study the choice of C in using our methods. We leave this to the future work.

Acknowledgement

This work was supported in part by the Australian Research Council (ARC) under grant DP0985456, the Nature Science Foundation (NSF) of China under grant (60496327, 90718020, 60625204), the China 973 Program under grant 2008CB317108, the Research Program of China Ministry of Personnel for Overseas-Return High-level Talents, the MOE Project of Key Research Institute of Humanities and Social Sciences at Universities (07JJD720044), and the Guangxi NSF (Key) grants.

References

1. Allison, P.D.: Multiple imputation for missing data: a cautionary tale. Sociological Methods and Research 28, 301–309 (2000)
2. Blake, C., Merz, C.: UCI Repository of machine learning database. University of California, Department of Information and Computer Science, Irvine (1998), http://www.ics.uci.edu/~mlearn/MLResoesitory.html
3. Cios, K., Kurgan, L.: Trends in Data Mining and Knowledge Discovery. In: Pal, N., Jain, L., Teoderesku, N. (eds.) Knowledge Discovery in Advanced Information Systems. Springer, Heidelberg (2002)
4. Engle, R.F., et al.: Semiparametric Estimates of the Relation Between Weather and Electricity Sales Applications, JASA, vol. 394 (June 1986)
5. Farhangfar, A., Kurgan, L.A., Pedrycz, W.: A Novel Framework for Imputation of Missing Values in Databases. IEEE Transactions on SMC(A) (2007)
6. Faris, P., et al.: Multiple imputation versus data enhancement for dealing with missing data in observational health care outcome analyses. Journal of Clinical Epidemiology 55, 184–191 (2002)
7. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2006)
8. Karmaker, A., Kwek, S.: Incorporating an EM-Approach for Handling Missing Attribute-Values in Decision Tree Induction. In: Fifth International Conference on Hybrid Intelligent Systems (2005)
9. Kang, S.S., Koehler, K., Larsen, M.D.: Partial FEFI for Incomplete Tables with Covariates Iowa State University, JSM (2007)
10. Kim, Y.: The curse of the missing data (2001), http://209.68.240.11:8080
11. Little, R., Rubin, D.: Statistical analysis with missing data, 2nd edn. John Wiley & Sons, New York (2002)
12. Millimet, D., List, J., Stengos, T.: The Environmental Kuznets Curve: Real Progress or Misspecified Models? Review of Economics and Statistics 85(4), 1038–1047 (2003)
13. Mostafa, M.H., Amir, F.A., Neamat, E.G., Raafat, E.F.: Regression in the Presence Missing Data Using Ensemble Methods. In: Proc. IJCNN (2007)
14. Peixoto, J.: A property of well-formulated polynomial regression models. American Statistician 44, 26–30 (1990)
15. Peng, C., Zhu, J.: Comparison of Two Approaches for Handling Missing Covariates in Logistic Regression. Educational and Psychological Measurement (2008)
16. Pin, T., James, L.: The Elasticity of Demand for Gasoline: A Semi-Parametric Analysis (2005), http://uiuc.edu/~ng/working/gas.ps
17. Qin, Y.S., et al.: Semi-parametric optimization for missing data imputation. Appl. Intell. 27(1), 79–88 (2007)
18. Schafer, J.: Analysis of incomplete multivariate data, 1st edn. Chapman and Hall, London (1997)
19. Schafer, J.: NORM: Multiple imputation of incomplete multivariate data under a normal model. Version 2 (1999)
20. Scheffer, J.: Dealing with Missing Data. Res. Lett. Inf. Math. Sci. 3, 153–160 (2002)
21. Wang, Q., Rao, J.: Empirical likelihood-based inference under imputation for missing response data. Ann. Statist. 30, 896–924 (2002)
22. Yuan, Y.: Multiple imputation for missing data: concepts and new development SAS/STAT 8.2. SAS Institute Inc., Cary, NC (2001), http://www.sas.com/statistics

23. Zhang, C.Q., et al.: Efficient Imputation Method for Missing Values. In: PAKDD (2007)
24. Zhang, L.: Nonparametric Markov chain bootstrap for multiple imputation (2004)
25. Zhang, S., et al.: Missing is useful: missing values in cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering 17(12) (2005)
26. Zhang, S., Zhang, J., Zhu, X., Qin, Y., Zhang, C.: Missing Value Imputation Based on Data Clustering. In: Gavrilova, M.L., Tan, C.J.K. (eds.) Transactions on Computational Science I. LNCS, vol. 4750, pp. 128–138. Springer, Heidelberg (2008)
27. Zhang, S.C., et al.: POP Algorithm: Kernel-Based Imputation to Treat Missing data in Knowledge Discovery from Databases, Expert Systems with Applications (2008)
28. Zhang, W.: Association based multiple imputation in multivariate datasets: A summary. In: Proc. 16th ICDE (2000)

The Average Solution of a Stochastic Nonlinear Schrodinger Equation under Stochastic Complex Non-homogeneity and Complex Initial Conditions

Magdy A. El-Tawil

Cairo University, Faculty of Engineering, Engineering Mathematics Department, Giza, Egypt
magdyeltawil@yahoo.com

Abstract. In this paper, a stochastic nonlinear Schrodinger equation is studied under stochastic complex non-homogeneity in a limited time interval through homogeneous boundary conditions and complex initial conditions. The analytical solution for the linear case is introduced. The Wiener-Hermite expansion together with the perturbation method, the WHEP technique, is used to get approximate ensemble average of the stochastic solution process. Using Mathematica, the solution algorithm is tested through computing the first order approximation of the solution ensemble average. The method is illustrated through case studies which demonstrate the effects of the initial conditions as well as the input non-homogeneities.

Keywords: Stochastic Nonlinear Schrodinger Equation, Perturbation, Eigenfunction Expansion, WHEP Technique, Wiener-Hermite Expansion.

1 Introduction

The nonlinear Schrodinger equation (NLS)arise as model equations from several areas of physics and applied sciences, see [1,2,3,4,5] for examples. All of which can be considered as perturbations of one sort or another of the linear Schrodinger equation. Needless to say, these equations are more difficult to analyze and are in great need to more developed and efficient computational algorithms.

Wang M. and et al [6] obtained the exact solutions to NLS using what they called the sub-equation method. They got four kinds of exact solutions of the equation

$$i\,\frac{\partial u}{\partial t}+\frac{1}{2}\frac{\partial^2 u}{\partial x^2}+\alpha\left|u\right|^p u+\beta\left|u\right|^{2p}u=0,$$

for which no sign to the initial or boundary conditions type is made. Xu L. and Zhang J. [7] followed the same previous technique in solving the higher order NLS:

$$i\,\frac{\partial u}{\partial x}-\frac{1}{2}\alpha\frac{\partial^2 u}{\partial t^2}+\beta\left|u\right|^2 u+i\,\varepsilon\frac{\partial^3 u}{\partial t^3}+i\,\delta\left|u\right|^2\frac{\partial u}{\partial t}+i\,\gamma u^2\frac{\partial u^*}{\partial t}=0.$$

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 143–170, 2009.

Sweilam N. [8] solved

$$i\,\frac{\partial u}{\partial t}+\frac{\partial^2 u}{\partial x^2}+q\,|u|^2 u=0, t>0, L_0<x<L_1,$$

with initial condition $u(x,0)=g(x)$ and boundary conditions $u_x(L_0,t)=u_x(L_1,t)=0$ which gives rise to solitary solutions using variational iteration method. Zhu S. [9] used the extended hyperbolic auxiliary equation method in getting the exact explicit solutions to the higher order NLS:

$$iq_z-\frac{\beta_1}{2}q_{tt}+\gamma_1|q|^2 q=i\,\frac{\beta_2}{6}q_{ttt}+\frac{\beta_3}{24}q_{tttt}-\gamma_2|q|^4 q,$$

without any conditions. Sun J. and et al [10] solved the NLS:

$$i\,\frac{\partial \psi}{\partial t}+\frac{\partial^2 \psi}{\partial x^2}+a|\psi|^2\psi=0,$$

with the initial condition $\psi(x,0)=\psi_0(x)$ using Lie group method. By using coupled amplitude phase formulation, Parsezian K. and Kalithasan B. [11] constructed the quartic anharmonic oscillator equation from the coupled higher order NLS. Two-dimensional grey solitons to the NLS were numerically analyzed by Sakaguchi H. and Higashiuchi T. [12]. The generalized derivative NLS was studied by Huang D. and et al [13] introducing a new auxiliary equation expansion method.

Moebs G. [14] considered a stochastic NLS equation as a model for signal propagation in optical fibres. He used the split-step Agrawal method together with a new multilevel method to obtain the propagation of a binary signal by solitons. Garnier J. and Abdullaev F. [15] introduced the theory of modulational instability of electromagnetic waves in optical fibres where they used a model of nonlinear Schrodinger equation with random group velocity dispersion and random nonlinear coefficients. Adrian A. and et al [16] found what they called the exact dynamics in mean value for a particular model of the Schrodinger-Langevin equation that preserves norm for all realizations. Abdullaev F. and et al [17] investigated the propagation of optical pulses in two types of fibres with randomly varying dispersion. Gautier E. [18] considered real multiplicative Gaussian noise in a stochastic NLS.

In this paper, the WHEP technique is used to introduce an approximate average solution to a stochastic NLS equation. The technique can be revised in appendix A or in [19-23] for more applications. In section 2, the linear Schrodinger equation is solved analytically to serve the methodology used in section 3 for solving a stochastic NLS equation for which some case studies are illustrated in section 4.

2 The Linear Case

In this section, the analytical solution of the linear Schrodinger equation is obtained which will be considered as a prototype solution when proceeding in the next section, mainly the nonlinear case. Consider the non homogeneous linear Schrodinger equation:

$$i\,\frac{\partial u(t,z)}{\partial z}+\alpha\frac{\partial^2 u(t,z)}{\partial t^2}+i\,\gamma u(t,z)=F_1(t,z)+i\,F_2(t,z),(t,z)\in(0,T)\times(0,\infty) \tag{1}$$

where $u(t,z)$ is a complex valued function which is subjected to initial conditions (I.C.),

$$u(t,0) = f_1(t) + i\, f_2(t), \tag{2}$$

and boundary conditions (B.C.),

$$u(0,z) = 0, u(T,z) = 0\,. \tag{3}$$

The imaginary constant is represented by i while α and γ are physical constants. The real functions F_1 and F_2 are the real and imaginary parts of the non-homogeneity respectively.

Let $u(t,z) = \psi(t,z) + i\,\phi(t,z)$, ψ, ϕ: real valued functions. The following coupled equations are obtained:

$$\frac{\partial \phi(t,z)}{\partial z} = \alpha \frac{\partial^2 \psi(t,z)}{\partial t^2} - \gamma\phi(t,z) - F_1(t,z), \tag{4}$$

$$\frac{\partial \psi(t,z)}{\partial z} = -\alpha \frac{\partial^2 \phi(t,z)}{\partial t^2} - \gamma\psi(t,z) + F_2(t,z), \tag{5}$$

where $\psi(t,0) = f_1(t), \phi(t,0) = f_2(t),$ and all other corresponding I.C. and B.C. are zeros.

To get rid of the loss terms in equations (4) and (5), mainly $-\gamma\phi$ and $-\gamma\psi$, one can use the following effective transformations:

$$\psi(t,z) = e^{-\gamma z} w(t,z), \tag{6}$$

$$\phi(t,z) = e^{-\gamma z} v(t,z), \tag{7}$$

where $w(t,0) = f_1(t), v(t,0) = f_2(t),$ and all other corresponding I.C. and B.C. are zeros. The following coupled equations are obtained:

$$\frac{\partial v(t,z)}{\partial z} = \alpha \frac{\partial^2 w(t,z)}{\partial t^2} - G_1(t,z), \tag{8}$$

$$\frac{\partial w(t,z)}{\partial z} = -\alpha \frac{\partial^2 v(t,z)}{\partial t^2} + G_2(t,z), \tag{9}$$

where

$$G_1(t,z) = e^{\gamma z} F_1(t,z)\,, \tag{10}$$

$$G_2(t,z) = e^{\gamma z} F_2(t,z)\,. \tag{11}$$

Eliminating one of the variables in equations (8) and (9), one can get the following independent equations:

$$\frac{\partial^4 w\ (t,z)}{\partial t^4}+\frac{1}{\alpha^2}\frac{\partial^2 w\ (t,z)}{\partial z^2}=\frac{1}{\alpha^2}\tilde{\psi}_1(t,z), \tag{12}$$

$$\frac{\partial^4 v\ (t,z)}{\partial t^4}+\frac{1}{\alpha^2}\frac{\partial^2 v\ (t,z)}{\partial z^2}=\frac{1}{\alpha^2}\tilde{\psi}_2(t,z), \tag{13}$$

where

$$\tilde{\psi}_1(t,z)=\alpha\frac{\partial^2 G_1(t,z)}{\partial t^2}+\frac{\partial G_2(t,z)}{\partial z}, \tag{14}$$

$$\tilde{\psi}_2(t,z)=\alpha\frac{\partial^2 G_2(t,z)}{\partial t^2}-\frac{\partial G_1(t,z)}{\partial z}. \tag{15}$$

Using the eigenfunction expansion technique [24], the following solution expressions are obtained:

$$w\ (t,z)=\sum_{n=0}^{\infty}T_n(z)\sin(\frac{n\pi}{T})t, \tag{16}$$

$$v\ (t,z)=\sum_{n=0}^{\infty}\tau_n(z)\sin(\frac{n\pi}{T})t, \tag{17}$$

where $T_n(z)$ and $\tau_n(z)$ can be got through the applications of initial conditions and then solving the resultant second order differential equations using the method of the variational parameter [25]. The final expressions can be got as the following:

$$T_n(z)=(C_1+A_1(z))\sin\beta_n z+(C_2+B_1(z))\cos\beta_n z, \tag{18}$$

$$\tau_n(z)=(C_3+A_2(z))\sin\beta_n z+(C_4+B_2(z))\cos\beta_n z, \tag{19}$$

where

$$\beta_n=\alpha(\frac{n\pi}{T})^2, \tag{20}$$

$$A_1(z)=\frac{1}{\beta_n}\int\tilde{\psi}_{1n}(z;n)\cos\beta_n z\,dz, \tag{21}$$

$$B_1(z)=\frac{-1}{\beta_n}\int\tilde{\psi}_{1n}(z;n)\sin\beta_n z\,dz, \tag{22}$$

$$A_2(z) = \frac{1}{\beta_n}\int \tilde{\psi}_{2n}(z;n)\cos\beta_n z\, dz\,, \tag{23}$$

$$B_2(z) = \frac{-1}{\beta_n}\int \tilde{\psi}_{2n}(z;n)\sin(\beta_n)z\, dz\,, \tag{24}$$

in which

$$\tilde{\psi}_{1n}(z;n) = \frac{2}{T}\int_0^T \tilde{\psi}_1(t,z)\sin(\frac{n\pi}{T})dt\,, \tag{25}$$

$$\tilde{\psi}_{2n}(z;n) = \frac{2}{T}\int_0^T \tilde{\psi}_2(t,z)\sin(\frac{n\pi}{T})dt\,. \tag{26}$$

The following conditions should also be satisfied:

$$C_2 = \frac{2}{T}\int_0^T f_1(t)\sin(\frac{n\pi}{T})dt - B_1(0), \tag{27}$$

$$C_4 = \frac{2}{T}\int_0^T f_2(t)\sin(\frac{n\pi}{T})dt - B_2(0). \tag{28}$$

Finally, the following solution is obtained:

$$u(t,z) = e^{-\gamma z}(w(t,z) + iv(t,z)), \tag{29}$$

or in absolute value as

$$\left|u(t,z)\right|^2 = e^{-2\gamma z}(w^2(t,z) + v^2(t,z)). \tag{30}$$

3 The Non-linear Case

Consider the non-homogeneous non-linear Schrodinger equation:

$$i\frac{\partial u(t,z)}{\partial z} + \alpha\frac{\partial^2 u(t,z)}{\partial t^2} + \varepsilon\left|u(t,z)\right|^2 u(t,z) + i\gamma u(t,z) = F_1(t,z;\omega) + i F_2(t,z;\omega),$$
$$(t,z;\omega)\in(0,T)\times(0,\infty)\times\Theta, \tag{31}$$

where $u(t,z)$ is a complex valued function, which is subjected to the initial and boundary conditions (2) and (3), ε is a deterministic nonlinearity scale and $\Theta = (\Omega, \tilde{\beta}, P)$ is a probability space in which Ω is a sample space, $\tilde{\beta}$ is a σ-field associated with the space Ω and *P* is a probability measure. For typing simplicity, ω is dropped.

Lemma
The solution of equation (31) with the constraints (2) and (3) is a power series in ε if the solution exists.

Proof
At $\varepsilon = 0$, the following linear equation is got:

$$i\frac{\partial u(t,z)}{\partial z}+\alpha\frac{\partial^2 u(t,z)}{\partial t^2}+i\gamma u(t,z)=F(t,z),\ (t,z)\in(0,T)\times(0,\infty)$$

which has the solution, see the previous section,

$$u_0(t,z)=e^{-\gamma z}(w_0(t,z)+iv_0(t,z)),$$

Following Pickard approximation, equation (31) can be rewritten as

$$i\frac{\partial u_n(t,z)}{\partial z}+\alpha\frac{\partial^2 u_n(t,z)}{\partial t^2}+i\gamma u_n(t,z)=F(t,z)-\varepsilon\left|u_{n-1}(t,z)\right|^2 u_{n-1}(t,z),n\geq 1.$$

At n=1, the iterative equation takes the following form:

$$\begin{aligned} i\frac{\partial u_1(t,z)}{\partial z}+\alpha\frac{\partial^2 u_1(t,z)}{\partial t^2}+i\gamma u_1(t,z) &= F(t,z)-\varepsilon\left|u_0(t,z)\right|^2 u_0(t,z) \\ &= F(t,z)+\varepsilon h_1(t,z) \end{aligned}$$

which can be solved as a linear case with zero initial and boundary conditions. The following general solution can be obtained:

$$w_1(t,z)=\sum_{n=0}^{\infty}(T_{0n}+\varepsilon T_{1n})\sin(\frac{n\pi}{T})t,$$

$$v_1(t,z)=\sum_{n=0}^{\infty}(\tau_{0n}+\varepsilon\tau_{1n})\sin(\frac{n\pi}{T})t,$$

$$\begin{aligned} u_1(t,z) &= e^{-\gamma z}(w_1(t,z)+iv_1(t,z)) \\ &= u_1^{(0)}+\varepsilon u_1^{(1)}. \end{aligned}$$

At n=2, the following equation is obtained:

$$\begin{aligned} i\frac{\partial u_2(t,z)}{\partial z}+\alpha\frac{\partial^2 u_2(t,z)}{\partial t^2}+i\gamma u_2(t,z) &= F(t,z)-\varepsilon\left|u_1(t,z)\right|^2 u_1(t,z) \\ &= F(t,z)+\varepsilon h_2(t,z) \end{aligned}$$

which can be solved as a linear case with zero initial and boundary conditions. The following general solution can be obtained:

$$u_2(t,z) = u_2^{(0)} + \varepsilon u_2^{(1)} + \varepsilon^2 u_2^{(2)} + \varepsilon^3 u_2^{(3)} + \varepsilon^4 u_2^{(4)}.$$

Continuing like this, one can get

$$u_n(t,z) = u_n^{(0)} + \varepsilon u_n^{(1)} + \varepsilon^2 u_n^{(2)} + \varepsilon^3 u_n^{(3)} + \ldots\ldots\ldots \varepsilon^{(n+m)} u_n^{(n+m)}.$$

As $n \to \infty$, the solution (if exists) can be reached as $u(t,z) = \lim_{n\to\infty} u_n(t,z)$. Accordingly, the solution is a power series in ε.

According to the previous lemma, one can expect that the ensemble average of the solution process and the other statistical moments can be represented also as a power series in ε.

3.1 The Solution Using WHEP Technique

By letting $u(t,z) = \psi(t,z) + i\,\phi(t,z)$ in equation (31), the following coupled equations are obtained:

$$\frac{\partial \phi(t,z)}{\partial z} = \alpha \frac{\partial^2 \psi(t,z)}{\partial t^2} + \varepsilon(\psi^2 + \phi^2)\psi - \gamma\phi(t,z) - F_1(t,z), \quad (32)$$

$$\frac{\partial \psi(t,z)}{\partial z} = -\alpha \frac{\partial^2 \phi(t,z)}{\partial t^2} - \varepsilon(\psi^2 + \phi^2)\phi - \gamma\psi(t,z) + F_2(t,z), \quad (33)$$

where $\psi(t,0) = f_1(t), \phi(t,0) = f_2(t)$ and all corresponding other I.C. and B.C. are zeros.

Taking the Gaussian part of the Wiener-Hermite expansion [21], the input and output stochastic processes of the problem can be represented as follows:

$$F_1(t,z) = F_1^{(0)} + \int_0^t F_1^{(1)}(t,z,t_1) H^{(1)}(t_1)\,dt_1, \quad (34)$$

$$F_2(t,z) = F_2^{(0)} + \int_0^t F_2^{(1)}(t,z,t_1) H^{(1)}(t_1)\,dt_1, \quad (35)$$

$$\psi(t,z) = \psi^{(0)} + \int_0^t \psi^{(1)}(t,z,t_1) H^{(1)}(t_1)\,dt_1, \quad (36)$$

$$\phi(t,z) = \phi^{(0)} + \int_0^t \phi^{(1)}(t,z,t_1) H^{(1)}(t_1)\,dt_1, \quad (37)$$

where the deterministic kernels $\psi^{(0)}, \psi^{(1)}, \phi^{(0)}$ and $\phi^{(1)}$ are unknowns to be evaluated through taking a set of ensemble averages [19]. The final governing equations are

$$\frac{\partial \phi^{(0)}(t,z)}{\partial z}+\alpha\frac{\partial^2 \psi^{(0)}(t,z)}{\partial t^2}-\gamma\phi^{(0)}(t,z)$$
$$+\varepsilon\left[\left[\psi^{(0)}\right]^3+\psi^{(0)}\left[\phi^{(0)}\right]^2+\psi^{(0)}\int_0^t\left[\phi^{(1)}(t,z,t_1)\right]^2dt_1+3\psi^{(0)}\int_0^t\left[\psi^{(1)}(t,z,t_1)\right]^2dt_1+2\phi^{(0)}\int_0^t\psi^{(1)}(t,z,t_1)\phi^{(1)}(t,z,t_1)dt_1\right]$$
$$=F_1^{(0)}(t,z) \tag{38}$$

$$\frac{\partial \psi^{(0)}(t,z)}{\partial z}+\alpha\frac{\partial^2 \phi^{(0)}(t,z)}{\partial t^2}+\gamma\psi^{(0)}(t,z)$$
$$+\varepsilon\left[\left[\phi^{(0)}\right]^3+\phi^{(0)}\left[\psi^{(0)}\right]^2+\phi^{(0)}\int_0^t\left[\psi^{(1)}(t,z,t_1)\right]^2dt_1+3\phi^{(0)}\int_0^t\left[\phi^{(1)}(t,z,t_1)\right]^2dt_1+2\psi^{(0)}\int_0^t\psi^{(1)}(t,z,t_1)\phi^{(1)}(t,z,t_1)dt_1\right]$$
$$=F_2^{(0)}(t,z) \tag{39}$$

$$\frac{\partial \phi^{(1)}(t,z,t_1)}{\partial z}+\alpha\frac{\partial^2 \psi^{(1)}(t,z,t_1)}{\partial t^2}-\gamma\phi^{(1)}(t,z,t_1)$$
$$+\varepsilon\left[3\left[\psi^{(0)}\right]^2\psi^{(1)}+2\psi^{(0)}\phi^{(0)}\phi^{(1)}(t,z,t_1)+\left[\phi^{(0)}\right]^2\psi^{(1)}(t,z,t_1)+3\psi^{(1)}\int_0^t\left[\psi^{(1)}(t,z,t_1)\right]^2dt_1+\right.$$
$$\left.\psi^{(1)}\int_0^t\left[\phi^{(1)}(t,z,t_1)\right]^2dt_1+2\phi^{(1)}\int_0^t\psi^{(1)}(t,z,t_1)\phi^{(1)}(t,z,t_1)dt_1\right]$$
$$=F_1^{(1)}(t,z,t_1) \tag{40}$$

$$\frac{\partial \psi^{(1)}(t,z,t_1)}{\partial z}+\alpha\frac{\partial^2 \phi^{(1)}(t,z,t_1)}{\partial t^2}+\gamma\psi^{(1)}(t,z,t_1)$$
$$+\varepsilon\left[3\left[\phi^{(0)}\right]^2\phi^{(1)}+2\psi^{(0)}\phi^{(0)}\psi^{(1)}(t,z,t_1)+\left[\psi^{(0)}\right]^2\phi^{(1)}(t,z,t_1)+3\phi^{(1)}\int_0^t\left[\phi^{(1)}(t,z,t_1)\right]^2dt_1+\right.$$
$$\left.\phi^{(1)}\int_0^t\left[\psi^{(1)}(t,z,t_1)\right]^2dt_1+2\psi^{(1)}\int_0^t\psi^{(1)}(t,z,t_1)\phi^{(1)}(t,z,t_1)dt_1\right]$$
$$=F_2^{(1)}(t,z,t_1) \tag{41}$$

Following the WHEP technique up to second correction, the following expressions can be assumed

$$\psi^{(0)}(t,z) = \psi_0^{(0)} + \varepsilon\psi_1^{(0)} + \varepsilon^2\psi_2^{(0)}, \tag{42}$$

$$\phi^{(0)}(t,z) = \phi_0^{(0)} + \varepsilon\phi_1^{(0)} + \varepsilon^2\phi_2^{(0)}, \tag{43}$$

$$\psi^{(1)}(t,z) = \psi_0^{(1)} + \varepsilon\psi_1^{(1)} + \varepsilon^2\psi_2^{(1)}, \tag{44}$$

$$\phi^{(1)}(t,z) = \phi_0^{(1)} + \varepsilon\phi_1^{(1)} + \varepsilon^2\phi_2^{(1)}\ , \tag{45}$$

where $\psi_0^{(0)}(t,0) = f_1(t)$, $\phi_0^{(0)}(t,0) = f_2(t)$ and the corresponding other initial and boundary conditions are zeros. Substituting equations (42-45) into the governing equations (38-41), the following set of coupled equations are got;

$$\frac{\partial\phi_0^{(0)}(t,z)}{\partial z} = \alpha\frac{\partial^2\psi_0^{(0)}(t,z)}{\partial t^2} - \gamma\phi_0^{(0)}(t,z) - \tilde{F}_1(t,z), \tag{46}$$

$$\frac{\partial\psi_0^{(0)}(t,z)}{\partial z} = -\alpha\frac{\partial^2\phi_0^{(0)}(t,z)}{\partial t^2} - \gamma\psi_0^{(0)}(t,z) + \tilde{F}_2(t,z), \tag{47}$$

where $\tilde{F}_1$ and $\tilde{F}_2$ are the average of the non-homogeneous functions F_1 and F_2 respectively.

$$\frac{\partial\phi_0^{(1)}(t,z)}{\partial z} = \alpha\frac{\partial^2\psi_0^{(1)}(t,z)}{\partial t^2} - \gamma\phi_0^{(1)}(t,z), \tag{48}$$

$$\frac{\partial\psi_0^{(1)}(t,z)}{\partial z} = -\alpha\frac{\partial^2\phi_0^{(1)}(t,z)}{\partial t^2} - \gamma\psi_0^{(1)}(t,z), \tag{49}$$

$$-\frac{\partial\phi_1^{(0)}}{\partial z} + \alpha\frac{\partial^2\psi_1^{(0)}}{\partial t^2} - \gamma\phi_1^{(0)} + \left[\psi_0^{(0)}\right]^3 + \psi_0^{(0)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2 dt_1 +$$

$$3\psi_0^{(0)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2 dt_1 + 2\phi_0^{(0)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1 = 0, \tag{50}$$

$$\frac{\partial\psi_1^{(0)}}{\partial z} + \alpha\frac{\partial^2\phi_1^{(0)}}{\partial t^2} + \gamma\psi_1^{(0)} + \left[\phi_0^{(0)}\right]^3 + \phi_0^{(0)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2 dt_1 +$$

$$3\phi_0^{(0)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2 dt_1 + 2\psi_0^{(0)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1 = 0 \tag{51}$$

$$-\frac{\partial\phi_1^{(1)}}{\partial z}+\alpha\frac{\partial^2\psi_1^{(1)}}{\partial t^2}-\gamma\phi_1^{(1)}+3\left[\psi_0^{(0)}\right]^2\psi_0^{(1)}+2\psi_0^{(0)}\phi_0^{(0)}\phi_0^{(1)}+3\psi_0^{(1)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+$$

$$\psi_0^{(1)}\left[\phi_0^{(0)}\right]^2+\psi_0^{(1)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1+2\phi_0^{(1)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1=\delta(t-t_1),\tag{52}$$

$$\frac{\partial\psi_1^{(1)}}{\partial z}+\alpha\frac{\partial^2\phi_1^{(1)}}{\partial t^2}+\gamma\psi_1^{(1)}+3\left[\phi_0^{(0)}\right]^2\phi_0^{(1)}+2\phi_0^{(0)}\psi_0^{(0)}\psi_0^{(1)}+3\phi_0^{(1)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1+$$

$$\phi_0^{(1)}\left[\psi_0^{(0)}\right]^2+\phi_0^{(1)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+2\psi_0^{(1)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1=\delta(t-t_1),\tag{53}$$

$$-\frac{\partial\phi_2^{(0)}}{\partial z}+\alpha\frac{\partial^2\psi_2^{(0)}}{\partial t^2}-\gamma\phi_2^{(0)}+3\left[\psi_0^{(0)}\right]^2\psi_1^{(0)}+2\psi_0^{(0)}\int_0^t\phi_0^{(1)}(t,z,t_1)\phi_1^{(1)}(t,z,t_1)dt_1+$$

$$\psi_1^{(0)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1+3\psi_1^{(0)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+6\psi_0^{(0)}\int_0^t\psi_1^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1$$

$$+2\phi_0^{(0)}\int_0^t\phi_1^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1+2\phi_0^{(0)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_1^{(1)}(t,z,t_1)dt_1+$$

$$2\phi_1^{(0)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1=0,\tag{54}$$

$$\frac{\partial\psi_2^{(0)}}{\partial z}+\alpha\frac{\partial^2\phi_2^{(0)}}{\partial t^2}+\gamma\psi_2^{(0)}+3\left[\phi_0^{(0)}\right]^2\phi_1^{(0)}+2\phi_0^{(0)}\int_0^t\psi_0^{(1)}(t,z,t_1)\psi_1^{(1)}(t,z,t_1)dt_1+$$

$$\phi_1^{(0)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+3\phi_1^{(0)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1+6\phi_0^{(0)}\int_0^t\phi_1^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1$$

$$+2\psi_0^{(0)}\int_0^t\psi_1^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1+2\psi_0^{(0)}\int_0^t\psi_0^{(1)}(t,z,t_1)\phi_1^{(1)}(t,z,t_1)dt_1+$$

$$2\psi_1^{(0)}\int_0^t\psi_0^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1=0,\tag{55}$$

$$-\frac{\partial \phi_2^{(1)}}{\partial z}+\alpha\frac{\partial^2 \psi_2^{(1)}}{\partial t^2}-\gamma\phi_2^{(1)}+3\left[\psi_0^{(0)}\right]^2\psi_1^{(1)}+6\psi_0^{(0)}\psi_1^{(0)}\psi_0^{(1)}+2\psi_0^{(0)}\phi_0^{(0)}\phi_1^{(1)}+2\phi_0^{(0)}\psi_1^{(0)}\phi_0^{(1)}$$

$$+2\psi_0^{(0)}\phi_1^{(0)}\phi_0^{(1)}+2\phi_0^{(0)}\psi_0^{(1)}\phi_1^{(0)}+\left[\phi_0^{(0)}\right]^2\psi_1^{(1)}+6\psi_0^{(1)}\int_0^t\psi_0^{(1)}(t,z,t_1)\psi_1^{(1)}(t,z,t_1)dt_1+$$

$$3\psi_1^{(1)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+2\psi_0^{(1)}\int_0^t\phi_1^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1+\psi_1^{(1)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1$$

$$+2\phi_0^{(1)}\int_0^t\phi_1^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1+2\phi_0^{(1)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_1^{(1)}(t,z,t_1)dt_1$$

$$+2\phi_1^{(1)}\int_0^t\phi_0^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1=0,$$

(56)

$$\frac{\partial \psi_2^{(1)}}{\partial z}+\alpha\frac{\partial^2 \phi_2^{(1)}}{\partial t^2}+\gamma\psi_2^{(1)}+3\left[\phi_0^{(0)}\right]^2\phi_1^{(1)}+6\phi_0^{(0)}\phi_1^{(0)}\phi_0^{(1)}+2\phi_0^{(0)}\psi_0^{(0)}\psi_1^{(1)}+2\psi_0^{(0)}\phi_1^{(0)}\psi_0^{(1)}+$$

$$2\phi_0^{(0)}\psi_1^{(0)}\psi_0^{(1)}+2\phi_0^{(1)}\psi_0^{(0)}\psi_1^{(0)}+\left[\psi_0^{(0)}\right]^2\phi_1^{(1)}+$$

$$6\phi_0^{(1)}\int_0^t\phi_0^{(1)}(t,z,t_1)\phi_1^{(1)}(t,z,t_1)dt_1+3\phi_1^{(1)}\int_0^t\left[\phi_0^{(1)}(t,z,t_1)\right]^2dt_1$$

$$+2\phi_0^{(1)}\int_0^t\psi_1^{(1)}(t,z,t_1)\psi_0^{(1)}(t,z,t_1)dt_1+\phi_1^{(1)}\int_0^t\left[\psi_0^{(1)}(t,z,t_1)\right]^2dt_1+$$

$$2\psi_0^{(1)}\int_0^t\psi_1^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1$$

$$+2\psi_0^{(1)}\int_0^t\psi_0^{(1)}(t,z,t_1)\phi_1^{(1)}(t,z,t_1)dt_1+2\psi_1^{(1)}\int_0^t\psi_0^{(1)}(t,z,t_1)\phi_0^{(1)}(t,z,t_1)dt_1=0.$$

(57)

The set of coupled equations (46-57) can be represented as the prototype linear equations (4) and (5) and so can be solved analytically using the linear case algorithm.

3.2 Zero Order Approximation

Solving equations (46-49), the zero order approximation takes the following form:

$$u_{(0)}=\psi_{(0)}+i\ \phi_{(0)} \tag{58}$$

where $\psi_{(0)}=\psi_0^{(0)}$ and $\phi_{(0)}=\phi_0^{(0)}$, hence the absolute average function can be got using:

$$\left|\mu_{u_{(0)}}\right|^2 = \left[\psi_0^{(0)}\right]^2 + \left[\phi_0^{(0)}\right]^2 . \tag{59}$$

3.3 First Order Approximation

Solving equations (50-53), the first order approximation can be represented as

$$u_{(1)} = \psi_{(1)} + i\ \phi_{(1)} \tag{60}$$

where

$$\psi_{(1)} = \left[\psi_0^0 + \varepsilon\psi_1^0\right] + \varepsilon\int_0^t \psi_1^1(t,z,t_1)H^{(1)}(t_1)dt_1 \tag{61}$$

and

$$\phi_{(1)} = \left[\phi_0^0 + \varepsilon\phi_1^0\right] + \varepsilon\int_0^t \phi_1^1(t,z,t_1)H^{(1)}(t_1)dt_1 . \tag{62}$$

Accordingly, the absolute average function can be got using:

$$\left|\mu_{u_{(1)}}\right|^2 = \left|\mu_{u_{(0)}}\right|^2 + 2\varepsilon\left[\psi_0^{(0)}\psi_1^{(0)} + \phi_0^{(0)}\phi_1^{(0)}\right] + \varepsilon^2\left[\left[\psi_1^{(0)}\right]^2 + \left[\phi_1^{(0)}\right]^2\right] \tag{63}$$

The covariance function can be got from:

$$\left|Cov(u_{(1)}(t,z),u_{(1)}(\tau,z))\right|^2 = \varepsilon^4\left(\left[\int_0^t \psi_1^{(1)}(t,z,t_1)\psi_1^{(1)}(\tau,z,t_1)dt_1 - \int_0^t \phi_1^{(1)}(t,z,t_1)\phi_1^{(1)}(\tau,z,t_1)dt_1\right]^2\right.$$
$$\left. + \left[\int_0^t \psi_1^{(1)}(t,z,t_1)\phi_1^{(1)}(\tau,z,t_1)dt_1 + \int_0^t \phi_1^{(1)}(t,z,t_1)\psi_1^{(1)}(\tau,z,t_1)dt_1\right]^2\right). \tag{64}$$

From which, the variance can be computed at $t = \tau$ as follows:

$$\left|Var(u_{(1)}(t,z)\right|^2 = \varepsilon^4\left(\left[\int_0^t (\psi_1^{(1)}(t,z,t_1))^2 dt_1 - \int_0^t (\phi_1^{(1)}(t,z,t_1))^2 dt_1\right]^2\right.$$
$$\left. + 4\left[\int_0^t \psi_1^{(1)}(t,z,t_1)\phi_1^{(1)}(t,z,t_1)dt_1\right]^2\right). \tag{65}$$

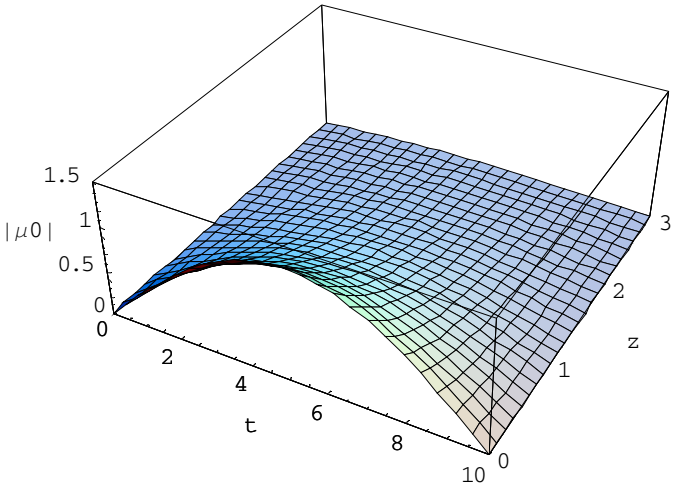


Fig. 1. The zero order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 0$ using only two terms for the solution expansions, mathematica 5

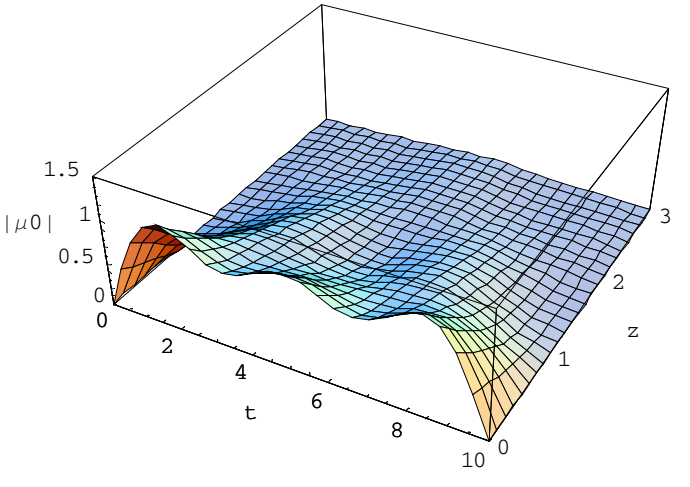


Fig. 2. The zero order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 0$ using only five terms for the solution expansions, mathematica 5

4 Case Studies

The main goal of this section is to illustrate the efficiency of the proposed symbolic solution algorithm presented in section 3. The effect of the nonlinearity strength parameter ε is shown through a lot of case studies under the changes of the input functions; mainly, the initial conditions $f_1(t)$ and $f_2(t)$, and the non-homogeneity functions $F_1(t,z)$ and $F_2(t,z)$.

4.1 Case Study-1

In this case, $f_1(t)=1, f_2(t)=0, F_1(t,z)=0,$ and $F_2(t,z)=0$. Substituting these input values in the expression of the zero order approximation (59) and its consequent related equations, mainly the solution of equations (46) and (47) under zero non-homogeneities, the following results are obtained.

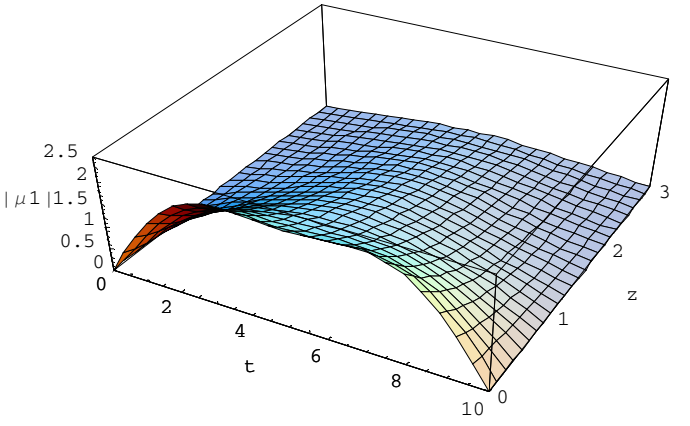


Fig. 3. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=1$ for only two terms, mathematica 5

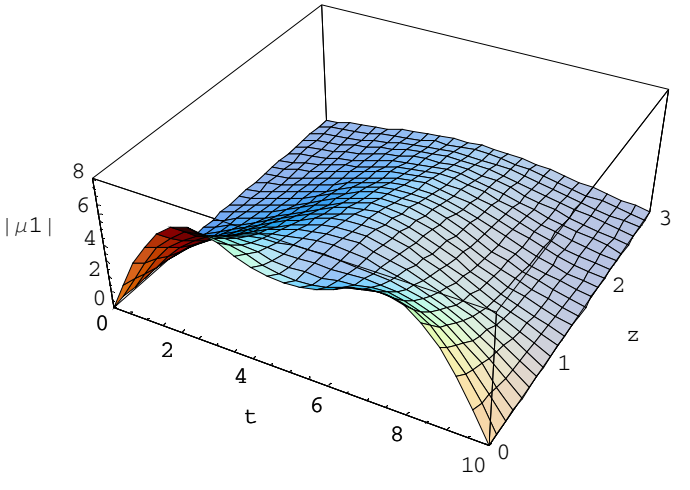


Fig. 4. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=5$ for only two terms, mathematica 5

One can notice that the extra solution expansion terms does not affect the solution level but it increases its fluctuations.

Now, let us substitute the input values in the first order solution expression (63) and its consequent, mainly the solution of equations (50) and (51). Using mathematica-5 to compute symbolically the resultant solution expression, the following results are obtained.

One can notice that the increase of ε values, i.e. increasing the nonlinearity, highly increases the solution level.

One can notice the high decrease of the solution average with the increase of the value of z.

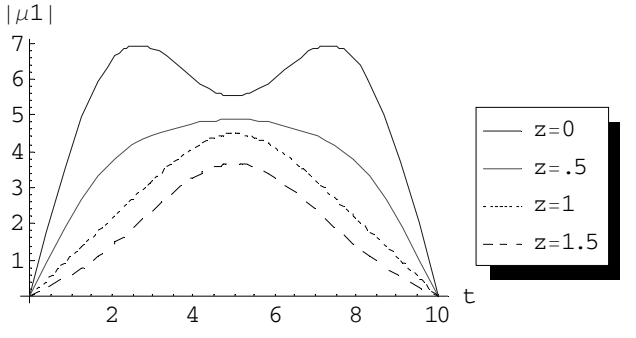


Fig. 5. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of z and for only two terms, mathematica 5

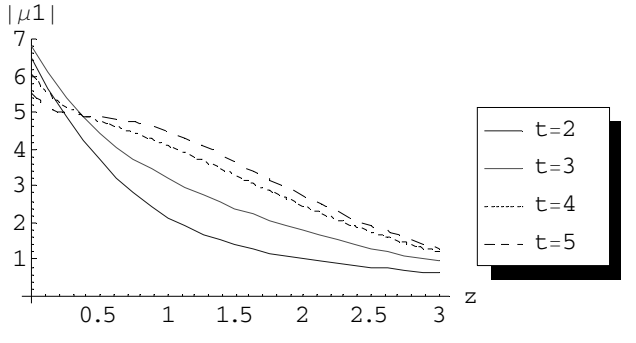


Fig. 6. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only two terms, mathematica 5

4.2 Case Study-2

In this case, $f_1(t) = 0, f_2(t) = 1, F_1(t, z) = 0,$ and $F_2(t, z) = 0$. The following results are obtained.

One can notice that there is no difference between case-1 and case-2.

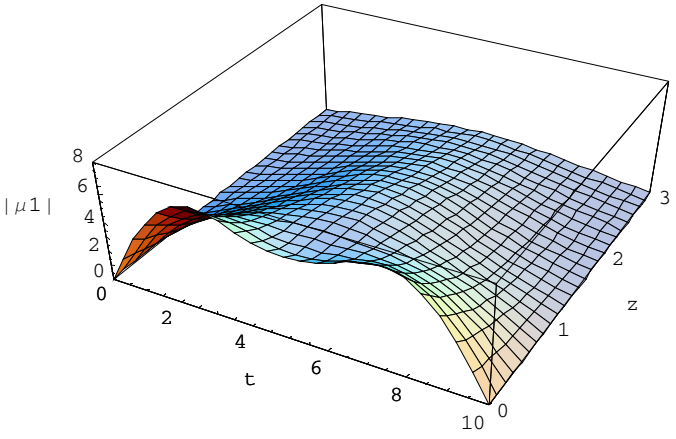


Fig. 7. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for only one term, mathematica 5

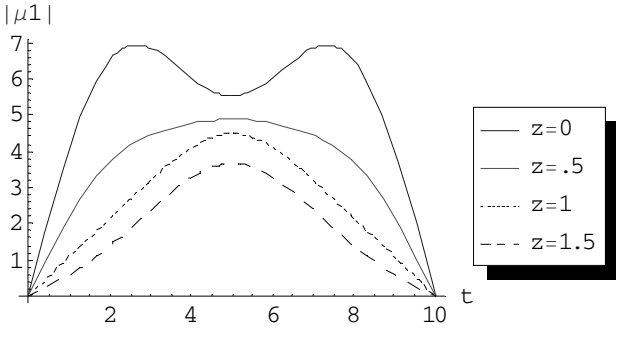


Fig. 8. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of z and for only one term

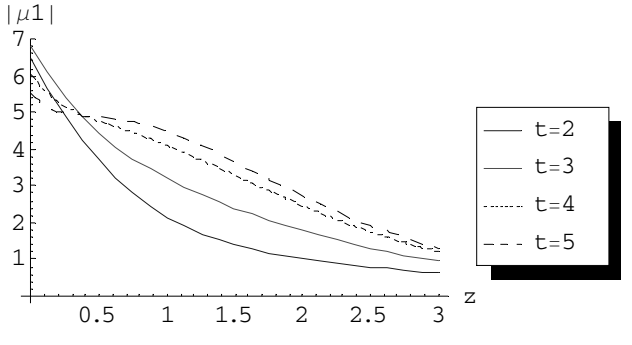


Fig. 9. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only one term

4.3 Case Study-3

In this case, $f_1(t)=0, f_2(t)=0, F_1(t,z)=1,$ and $F_2(t,z)=0$. The following results are obtained.

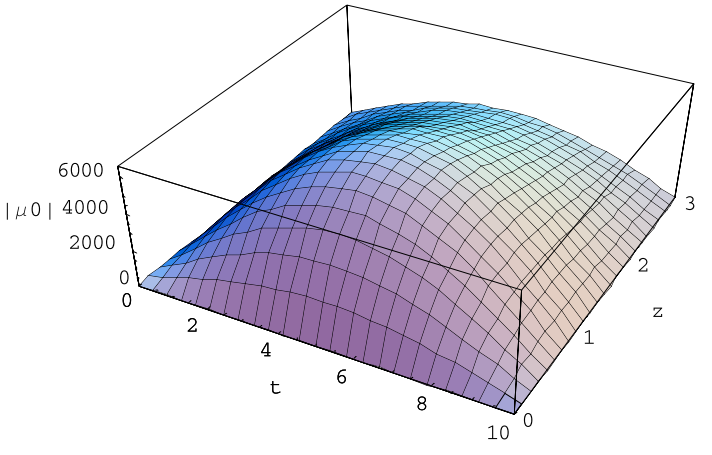


Fig. 10. The zero order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=0$ for only one term, mathematica 5

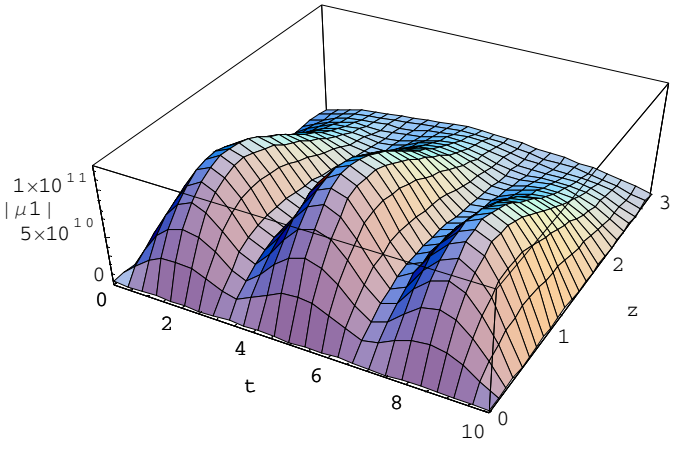


Fig. 11. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=1$ for only one term

One can notice the huge effect of F_1, the real part of the non-homogeneity, on the value of the average.

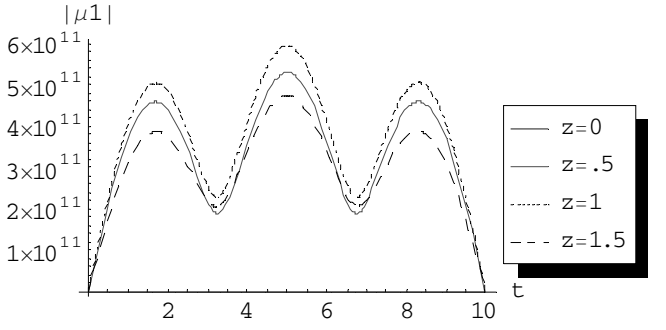


Fig. 12. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 1$ for different values of z and for only one term

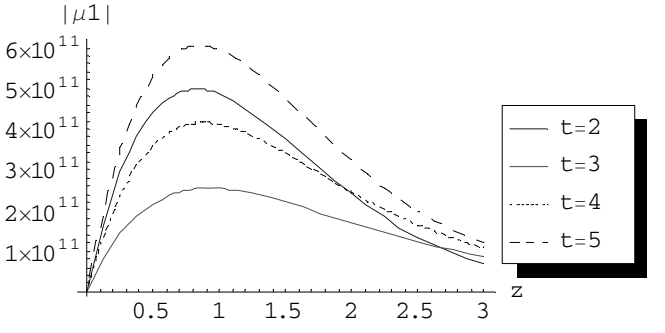


Fig. 13. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only one term

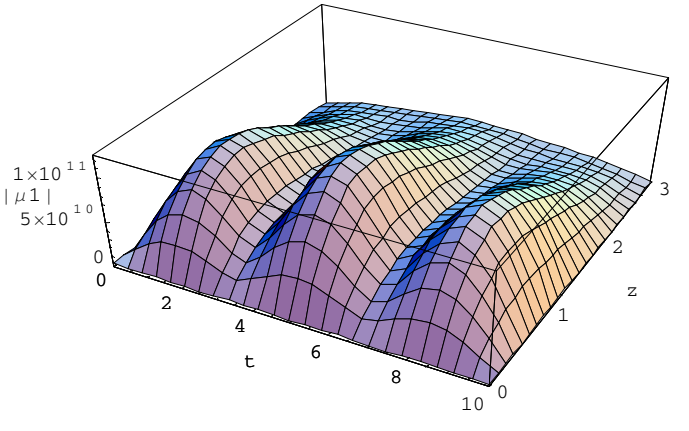


Fig. 14. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 1$ for only one term

4.4 Case Study-4

In this case, $f_1(t)=0, f_2(t)=0,\ F_1(t,z)=0,$ and $F_2(t,z)=1$. The following results are obtained.

One can notice that identical results to case-3 are obtained.

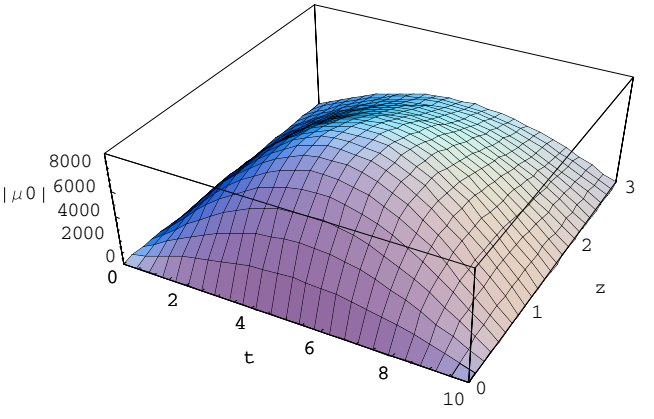


Fig. 15. The zero order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=0$ for only one term

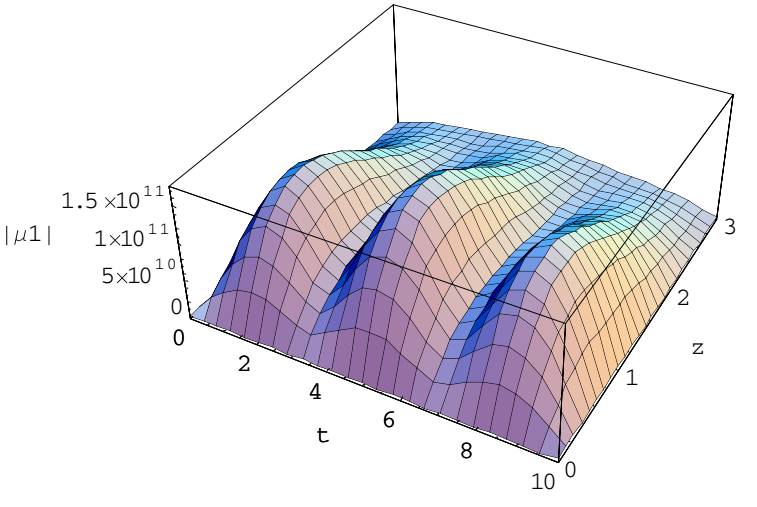


Fig. 16. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=1$ for only one term

4.5 Case Study-5

In this case, $f_1(t)=0, f_2(t)=0,\ F_1(t,z)=1,$ and $F_2(t,z)=1$. The following results are obtained.

One can notice the high increase of the value of the average compared to case-3 or 4.

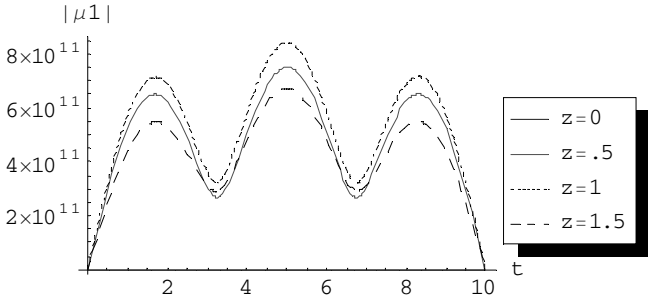


Fig. 17. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 1$ for different values of z and for only one term

4.6 Case Study-6

In this case, $f_1(t) = 1, f_2(t) = 1, F_1(t, z) = 0,$ and $F_2(t, z) = 0$. The following results are obtained.

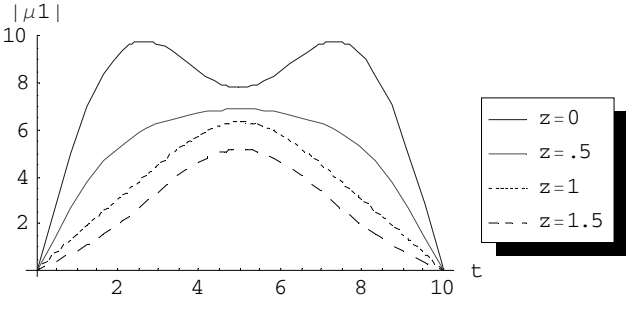


Fig. 18. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of z and for only one term

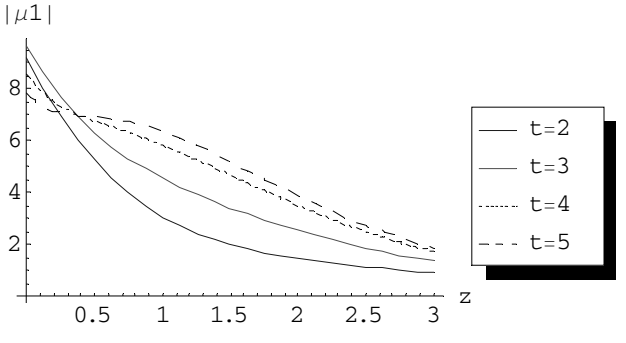


Fig. 19. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only one term

One can notice the slight increase for the average in this case when compared with case-1 or 2.

4.7 Case Study-7

In this case, $f_1(t)=e^{-t}$, $f_2(t)=e^{-t}$, $F_1(t,z)=0,$ and $F_2(t,z)=0$. The following results are obtained.

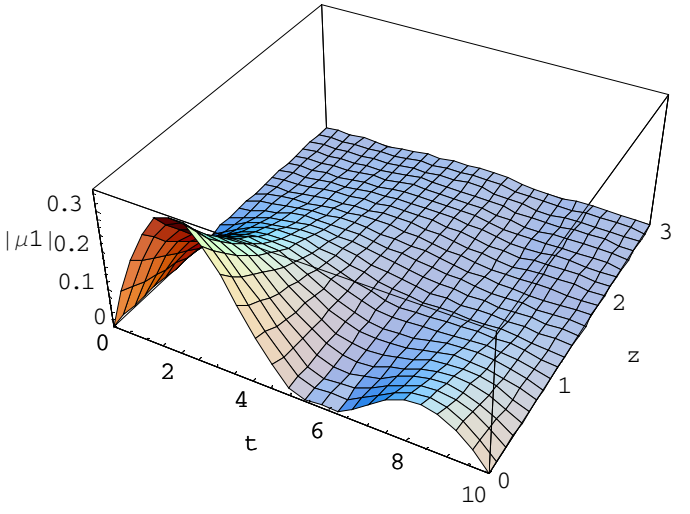


Fig. 20. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=1$ for only one term

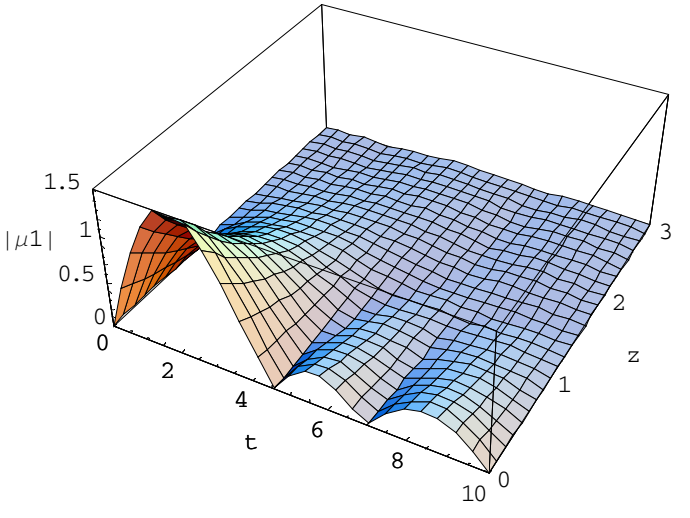


Fig. 21. The first order approximation of the absolute average of the solution at $\alpha,\gamma=1,T=10$, $\varepsilon=5$ for only one term

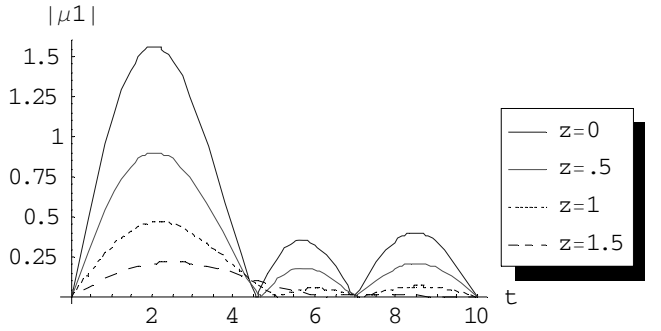


Fig. 22. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of z and for only one term

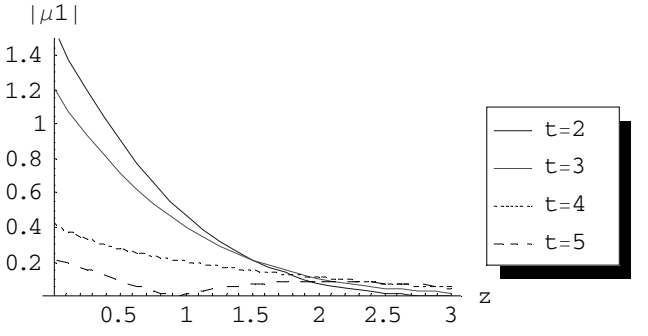


Fig. 23. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only one term

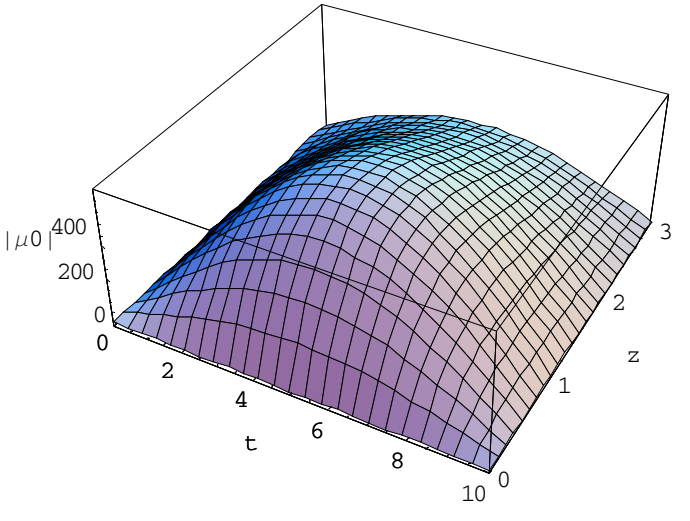


Fig. 24. The zero order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 0$ for only one term

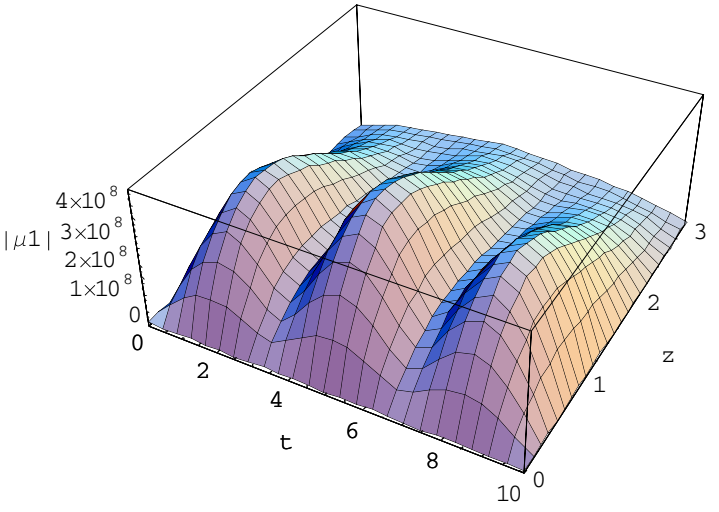


Fig. 25. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for only one term

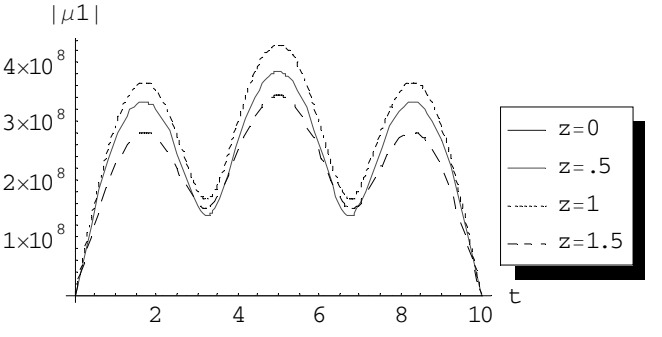


Fig. 26. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of z and for only one term

One can notice the high reduction in the magnitude of the average compared with case-6 and shape changes are also noticed.

4.8 Case Study-8

In this case, $f_1(t) = 0, f_2(t) = 0, F_1(t, z) = e^{-t}$, and $F_2(t, z) = e^{-t}$. The following results are obtained.

One can notice the high reduction in the average value.

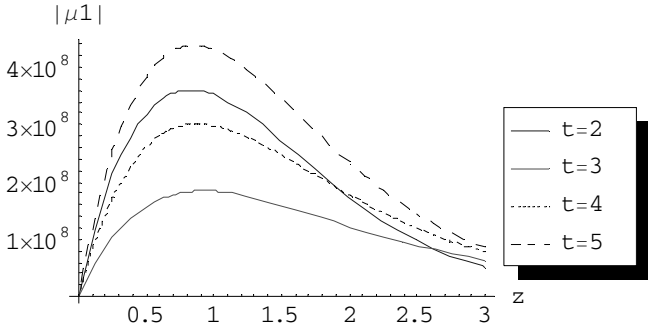


Fig. 27. The first order approximation of the absolute average of the solution at $\alpha, \gamma = 1, T = 10$, $\varepsilon = 5$ for different values of t and for only one term

5 Conclusions

The WHEP technique introduces an efficient approximate solution to the NLS equation with a perturbative nonlinear term for a finite time interval. Using mathematica, the difficult and huge computations were fronted to some extent, we still have computations problems when trying to evaluate higher orders of the solution average or trying to evaluate the covariance or variance of the solution process in the future. In general, there is a reduction in the magnitude of solution when z increases. The control of the input non-homogeneity functions $F_1(t)$ and/or $F_2(t)$ leads to a huge magnification in the solution's magnitude level while there is a reverse impact on the solution when controlled by the initial condition functions $f_1(t)$ and/or $f_2(t)$. The solution level highly increases with the increase of the nonlinearity scale parameter ε.

References

[1] Cazenave, T., Lions, P.: Orbital Stability of Standing Waves for Some Nonlinear Schrodinger Equations. Commun. Math. Phys. 85, 549–561 (1982)
[2] Faris, W.G., Tsay, W.J.: Time Delay in Random Scattering. SIAM J. on Applied Mathematics 54(2), 443–455 (1994)
[3] Bruneau, C., Menza, L., Lehner, T.: Numerical Resolution of Some Nonlinear Schrodinger-like Equations in Plasmas. Numer. Meth. PDEs 15(6), 672–696 (1999)
[4] Abdullaev, F., Garnier, J.: Solitons in Media With Random Dispersive Perturbations. Physica (D) 134, 303–315 (1999)
[5] Corney, J.F., Drummond, P.: Quantum Noise in Optical Fibres. II. Raman Jitter in Soliton Communications. J. Opt. Soc. Am. B 18(2), 153–161 (2001)
[6] Wang, M., et al.: Various Exact Solutions of Nonlinear Schrodinger Equation With Two Nonlinear Terms. Chaos, solitons and Fractals 31, 594–601 (2007)
[7] Xu, L., Zhang, J.: Exact Solutions to Two Higher Order Nonlinear Schrodinger Equations. Chaos, solitons and Fractals 31, 937–942 (2007)
[8] Sweilam, N.: Variational Iteration Method for Solving Cubic Nonlinear Schrodinger Equation. J. of computational and applied mathematics (to appear)

[9] Zhu, S.: Exact Solutions for The High Order Dispersive Cubic Quintic Nonlinear Schrodinger Equation by The Extended Hyperbolic Auxiliary Equation Method. Chaos, solitons and Fractals (to appear)
[10] Sun, J., et al.: New Conservation Schemes for The Nonlinear Schrodinger Equation. Applied Mathematics and Computation 177, 446–451 (2006)
[11] Parsezian, K., Kalithasan, B.: Cnoidal and Solitary Wave Solutions of The Coupled Higher Order Nonlinear Schrodinger Equations in Nonlinear Optics. Chaos, solitons and Fractals 31, 188–196 (2007)
[12] Sakaguchi, H., Higashiuchi, T.: Two Dimensional Dark Soliton in The Nonlinear Schrodinger Equation. Physics letters A (to appear)
[13] Huang, D., et al.: Explicit and Exact Traveling Wave Solution for The Generalized Derivative Schrodinger Equation. Chaos, solitons and Fractals 31, 586–593 (2007)
[14] Moebs, G.: A Multilevel Method for The Resolution of a Stochastic Weakly Damped Nonlinear Schrodinger Equation. Applied Numerical Mathematics 26, 353–375 (1998)
[15] Garnier, J., Abdullaev, F.: Modulational Instability Induced by Randomly Varying Coefficients for The Nonlinear Schrodinger Equation. Physica D 145, 65–83 (2000)
[16] Adrian, A., et al.: Averaged Exact Dynamics of a Stochastic Non-Markovian Wave Vector. Physica A 292, 383–391 (2001)
[17] Abdullaev, F., et al.: Optical Pulse Propagation in Fibers With Random Dispersion. Physica D 192, 83–94 (2004)
[18] Gautier, E.: Uniform Large Deviations for The Nonlinear Schrodinger Equation With Multiplicative Noise. Stochastic Processes and Their Applications 115, 1904–1927 (2005)
[19] Gawad, E., El-Tawil, M.: General Stochastic Oscillatory Systems. Applied Mathematical Modeling 17(6), 329–335 (1993)
[20] El-Tawil, M., Mahmoud, G.: The Solvability of Parametrically Forced Oscillators Using WHEP Technique. Mechanics and Mechanical Engineering 3(2), 181–188 (1999)
[21] El-Tawil, M.: The Application of WHEP Technique on Stochastic Partial Differential Equations. Int. J. of differential equations and applications 7(3), 325–337 (2003)
[22] Xu, Y., et al.: On a Complex Duffing System With Random Excitation. Chaos, Solitons and Fractals, expected (2007)
[23] El-Tawil, M.A.: The Homotopy Wiener-Hermite Expansion and Perturbation Technique (WHEP). In: Gavrilova, M.L., Tan, C.J.K. (eds.) Transactions on Computational Science I. LNCS, vol. 4750, pp. 159–180. Springer, Heidelberg (2008)
[24] Farlow, S.: P.D.E. for Scientists and Engineers. John Wiley & sons, N.Y. (1982)
[25] Pipes, L., Harvill, L.: Applied Mathematics for Engineers and Physicists. McGraw-Hill, Tokyo (1970)
[26] El-Tawil, M., Al-Johani, A.: On The Solution of Stochastic Oscillatory Quadratic Nonlinear Equations Using Different Techniques, A Comparison Study. TMNA, J. of Juliusz Schauder Center (to appear)

Appendix-A: WHEP Technique

The Wiener-Hermite expansion (WHE) method utilizes the Wiener-Hermite polynomials which are the elements of a complete set of statistically orthogonal random functions [19]. The Wiener-Hermite polynomial $H^{(i)}(t_1,t_2,...t_i)$ satisfies the following recurrence relation:

$$H^{(i)}(t_1,t_2,...t_i) = H^{(i-1)}(t_1,t_2,...t_{i-1}).H^{(1)}(t_i) - \sum_{m=1}^{i-1} H^{(i-2)}(t_{i_1},t_{i_2},...t_{i_{i-2}}).\delta(t_{i-m}-t_i), i \geq 2 \quad \text{(A-1)}$$

where

$$H^{(0)} = 1,$$

$$H^{(1)}(t) = n(t),$$

$$\mathrm{H}^{(2)}(t_1,t_2) = H^{(1)}(t_1).H^{(1)}(t_2) - \delta(t_1-t_2),$$

$$H^{(3)}(t_1,t_2,t_3) = \mathrm{H}^{(2)}(t_1,t_2).H^{(1)}(t_3) - H^{(1)}(t_1).\delta(t_2-t_3) - H^{(1)}(t_2).\delta(t_1-t_3),$$

$$H^{(4)}(t_1,t_2,t_3,t_4) = H^{(3)}(t_1,t_2,t_3).H^{(1)}(t_4) - \mathrm{H}^{(2)}(t_1,t_2).\delta(t_3-t_4) - \mathrm{H}^{(2)}(t_1,t_3).\delta(t_2-t_4) - \mathrm{H}^{(2)}(t_2,t_3).\delta(t_1-t_4), \quad \text{(A-2)}$$

in which n(t) is the white noise with the following statistical properties

$$E\,\mathrm{n}(t) = 0, \qquad \mathrm{E}\,\mathrm{n}(t_1).n(t_2) = \delta(t_1-t_2), \quad \text{(A-3)}$$

where δ(-) is the Dirac delta function and E denotes the ensemble average operator.

The Wiener-Hermite set is a statistically orthogonal set, i.e.

$$E\,\mathrm{H}^{(i)}.H^{(j)} = 0 \;\; \forall\, \mathrm{i} \neq \mathrm{j}. \quad \text{(A-4)}$$

The average of almost all H functions vanishes, particularly,

$$E\,\mathrm{H}^{(i)} = 0 \;\text{ for } \mathrm{i} \geq 1. \quad \text{(A-5)}$$

Due to the completeness of the Wiener-Hermite set ,any random function $G(t;\omega)$ can be expanded as

$$G(t;\omega) = G^{(0)}(t) + \int_{-\infty}^{\infty} G^{(1)}(t;t_1)H^{(1)}(t_1)dt_1 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G^{(2)}(t;t_1,t_2)H^{(2)}(t_1,t_2)dt_1dt_2 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G^{(3)}(t;t_1,t_2,t_3)H^{(3)}(t_1,t_2,t_3)dt_1dt_2dt_3 + ... \quad \text{(A-6)}$$

where the first two terms are the Gaussian part of $G(t;\omega)$. The rest of the terms in the expansion represent the non-Gaussian part of $G(t;\omega)$. The average of $G(t;\omega)$ is

$$\mu_G = E\,G(t;\omega) = G^{(0)}(t) \tag{A-7}$$

The covariance of $G(t;\omega)$ is

$$\begin{aligned} Cov(G(t;\omega),G(\tau;\omega)) &= E(G(t;\omega)-\mu_G(t))(G(\tau;\omega)-\mu_G(\tau)) \\ &= \int_{-\infty}^{\infty} G^{(1)}(t;t_1)G^{(1)}(\tau,t_1)dt_1 + 2\int_{-\infty-\infty}^{\infty\infty} G^{(2)}(t;t_1,t_2)G^{(2)}(\tau,t_1,t_2)dt_1dt_2 \\ &+ 2\int_{-\infty-\infty-\infty}^{\infty\infty\infty} G^{(3)}(t;t_1,t_2,t_3)[G^{(3)}(\tau,t_1,t_2,t_3) \\ &\quad + G^{(3)}(\tau,t_1,t_3,t_2) + G^{(3)}(\tau,t_2,t_3,t_1)]dt_1dt_2dt_3 + \ldots \end{aligned} \tag{A-8}$$

The variance of $G(t;\omega)$ is

$$\begin{aligned} Var\,G(t;\omega) &= E(G(t;\omega)-\mu_G(t))^2 \\ &= \int_{-\infty}^{\infty}[G^{(1)}(t;t_1)]^2dt_1 + 2\int_{-\infty-\infty}^{\infty\infty}[G^{(2)}(t;t_1,t_2)]^2dt_1dt_2 \\ &+ 2\int_{-\infty-\infty-\infty}^{\infty\infty\infty}[G^{(3)}(t;t_1,t_2,t_3)]^2dt_1dt_2dt_3 \\ &+ 2\int_{-\infty-\infty-\infty}^{\infty\infty\infty}[G^{(3)}(t,t_1,t_2,t_3)G^{(3)}(t,t_1,t_3,t_2)]dt_1dt_2dt_3 \\ &+ 2\int_{-\infty-\infty-\infty}^{\infty\infty\infty}[G^{(3)}(t,t_1,t_2,t_3)G^{(3)}(t,t_2,t_3,t_1)]dt_1dt_2dt_3 + \ldots. \end{aligned} \tag{A-9}$$

The WHE method can be elementary used in solving stochastic differential equations by expanding the solution process as well as the stochastic input processes via the WHE. The resultant equation is more complex than the original one due to being a stochastic integral-differential equation. Taking a set of ensemble averages together with using the statistical properties of the WH polynomials, a set of deterministic integral-differential equations are obtained in the deterministic kernels $G^{(i)}(t;\omega), i = 0,1,2,\ldots$. To obtain approximate solutions for these deterministic kernels, one can use perturbation theory in the case of having a perturbed system

depending on, say, ε . Expanding the kernels as a power series of ε, another set of simpler iterative equations in the kernel series components are obtained. This is the main algorithm of the WHEP technique. The technique was successfully applied to several nonlinear stochastic equations, see [19-23] and also [26].

By taking the only first two terms in the solution process expansion, the Gaussian part, the first order approximation is obtained. The second order approximation is got when adding the third term and the other orders are obtained by consequent terms additions. The WHEP technique has an advantage over the other known iterative techniques that every order of approximation can be corrected to any required order of corrections. The n^{th} correction represents the ε series of the deterministic kernels up to ε^n . Accordingly, one can have only first order approximate solution with different correction levels depending on how efficient the computing tool which may encourage computing other approximation orders.

Neural Network Representation for the Forces and Torque of the Eccentric Sphere Model

Mostafa Y. Elbakry[1], Mohammed El-Helly[2], and Mahmoud Y. Elbakry[2]

[1] Faculty of Education for girls, Tabuk, P.O.Box 796, KSA
[2] Faculty of science for girls, Dammam, KSA
elbakre1987@hotmail.com, m_el_helly@hotmail.com

Abstract. An artificial neural network (ANN) has been designed to simulate and predict the torque and force acting on the outer stationary sphere due to steady state motion of the second order fluid between two eccentric spheres by a rotating inner sphere with an angular velocity Ω The (ANN) model has been trained based on the experimental data to produce the torque and force at different eccentricities. The experimental and trained torque and force are compared .The designed ANN shows a good match to the experimental data.

Keywords: Neural Network, Eccentric Sphere, Eccentricity, Torque, Force.

1 Introduction

An artificial neural network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANN, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Neural networks are widely used for solving many scientific linear and non-linear problems [1,2,3,4,5] The theoretical and experimental studies concerning the flow field of viscous and viscoelastic fluids in annular region between two rotating bodies are very interesting boundary value problems in rheology [7,8,9,10] The solution of these boundary value problems based on microscopic models [11,12] or phemenological state equations of state [13] allow a number of experimental measurements sufficient to determine a specific set of material parameters such as viscosity and first normal stress difference which is very important in different branches of industries. A large number of theoretical and experimental works are done on the viscous flow between two eccentric spheres; [14,15,16,17,18]. Recently, M.Y.El-Bakry et .al. studied the flow of viscoelastic fluid between two eccentric spheres theoretically [19] and experimentally [20]. Determination of the torque and the force on the outer stationary sphere while the inner sphere

M.L. Gavrilova and C.J.K. Tan (Eds.): Trans. on Comput. Sci. III, LNCS 5300, pp. 171–183, 2009.

is rotated with angular velocity at different eccentricities is theoretically investigated in [19] and the experimental data are obtained in [20] .In the present work, we have applied the Artificial Neural Network ANN technique to simulate and predict the torque and the force on the outer stationary sphere as a function of angular velocity of the inner sphere in eccentric sphere model at different eccentricities using 0.3 polyacrylamide in 50/50 glycerin/water.

2 Eccentric Sphere Model

The flow of a fluid of a second order between two eccentric (two centers are not coincide on each other) spheres is considered. This study is carried out in terms of the Bispherical Coordinates α, β, γ.The inner sphere is rotated with angular velocity while the outer sphere is kept at rest. The equations of motion of first and second orders are formulated and solved. The solution of the problem is carried out within the frame of retarded motion approximation [13]. The velocity field up to a second order is being a superposition of a first order primary flow distributed uniformly around the axis of rotation and a secondary flow which is every where perpendicular to the streamlines of the primary flow. The forces and torque acting on the outer sphere, when kept at rest, are calculated in [19]. Since experimental measurements can be carried out about the axis of rotation and any two arbitrary axes perpendicular to it, the total forces and torques are expanded in the X,Y,Z-directions, where Z-axis is taken along the symmetry axis. The two component of forces in X and Y-directions are vanished and the non vanishing component of the force is in Z-direction which is dented by F_z . The non-vanishing resultant torque is obtained about the z-axis. The force and torque results are functions of eccentricity of the two spheres which represent distance between two centers of the two spheres, the angular velocity of the inner sphere and material parameters of the fluid (viscosity and first normal stress difference). The experimental setup is built to measure force and toque of the eccentric sphere model in order to determine viscosity and first normal stress difference in [20]. These results are measured at certain values of the eccentricity of the two spheres. In the present work we are used ANN to simulate and predict these results of the forces and torque at different eccentricities which enable us to predict these values of force and torque at any value of eccentricity.

3 Neural Network Representation for the Torque and Force Acting on the Outer Sphere in Eccentric Sphere Model

3.1 Feed-Forward Neural Networks

An artificial neural network (ANN) is made up of a number of simple and highly inter-connected computational elements. There are many types of ANNs, but all of them have three things in common: individual neurons (processing elements), connections (topology) and a learning algorithm. The processing element

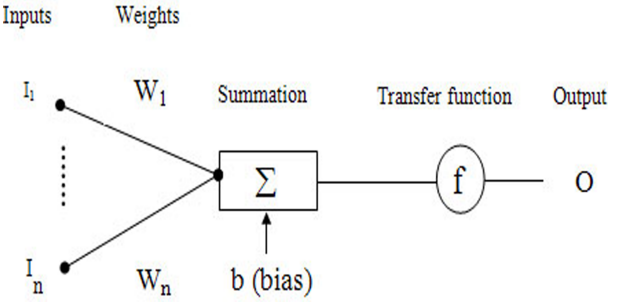


Fig. 1. Neuron Model

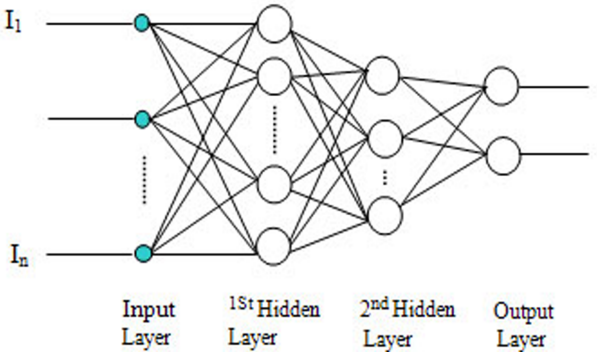


Fig. 2. Example of two hidden layer neural network

calculates the neuron transfer function of summation of weighted inputs. A simple neuron structure is shown in Figure (1).

The neuron transfer function, f , is typically step or sigmoid function that produces a scalar output (O) as follows

$$O = f(\sum_i W_i + b) \tag{1}$$

where I_i , W_i and b are i^{th} input, i^{th} weight and bias, respectively. A network consists of one or more layers of neurons. An example of a multi-layer network is shown in Figure (2). This example consists of one input layer and two hidden layers and one output layer. Each neuron can have multiple inputs and only one output as shown in Figure (2). Inputs to neurons could be from external stimuli or could be from output of other neurons. Copies of the single output that comes from a neuron could be input to many other neurons in the network. There is an inter-connection strength, weight, associated with each connection.

When the weighted sum of the inputs to the neurons exceeds a certain threshold, specified by a threshold (transfer) function with bias, the neuron is fired and an output signal is produced. The network can recognize input-output relation (mapping function) once the weights are tuned via some kind of learning

process [21] Neural networks (NN) can approximate a function, associate input patterns with specific output (desired or target) patterns, or classify input patterns in an appropriate way as defined by the user. The essential features for a feed-forward NN are reviewed below employing a two-layer NN [22,23]. However, the results generally hold for any multiple layers NN. It is assumed that, the NN consists of an input; a hidden layer and an output layer (see Fig.2). The objective is to associate a P-pattern input to their corresponding P-pattern target. The following definitions are necessary'

NI,NH and NO =the number of nodes in the input,hidden and output layers, respectively
I (i, r) = the i^{th} input value, i ∈ [1, NI] ; in the r^{th} input pattern, r ∈ [1, P],
W_1 (i, j) = the weight connecting the i^{th} input value to the j^{th} hidden neuron,
W_2(j, k) = the weight connecting the j^{th} hidden neuron to the k^{th} output neuron,
B_2(k) , k ∈ [1, NO] = the bias associated with the k^{th} output neuron,
O (k, r) = output of the k^{th} output neuron, k ∈ [1, NO], for the r^{th} input pattern,
T (k, r) = target of the k^{th} output neuron, k ∈ [1, NO], for the r^{th} input pattern,
Y = all weights and biases for the whole NN which starts with the random values. The output of the j^{th} hidden neuron at the r^{th} input pattern is given by:

$$H(j,r) = f\left[\sum_{i=1}^{NI} W_1(i,j)I(i,r) + B_1(j)\right] \tag{2}$$

where f, is an approximate transfer function. Typical transfer functions are the hyperbolic tangent function defined as:

$$f_1(\theta) = \tanh(\theta) \tag{3}$$

and the linear function defined as:

$$f_2(\theta) = \tanh(\theta) \tag{4}$$

Similarly, the output of the k^{th} output neuron is given by:

$$O(j,r) = f\left[\sum_{j=1}^{NH} W_2(j,k)H(i,r) + B_2(k)\right] \tag{5}$$

The NN output O, is required to mimic a target output T. To achieve that, the NN is trained to find an approximate set of weights and biases Y, which minimizes an index E defined as:

$$E = \sum_{k=1}^{NO}\sum_{r=1}^{P}\left[O(k,r) - T(k,r)\right]^2 \tag{6}$$

An algorithm is employed to minimize the index E over Y, employing gradients estimated using the partial derivatives of E with respect to Y. The gradients are determined, employing the backpropagation technique which involves

performing computations backward in the network [24]. The training is performed employing the Levenberg - Marquardt algorithm (LMA) [25]. The LMA employs a Newton - like update in the form,

$$Y_v = Y_{v-1} - \left[J_t - \mu \tilde{I} \right]^{-1} J^t e \tag{7}$$

where J is the Jacobian matrix which contains the first derivatives of the NN errors with respect to the weights and biases, e is a vector of NN errors, μ is is a scalar changed adaptively by the algorithm (the adaptation constant), v is the iteration number, t denotes transposition and $\tilde{I}$ is the identity matrix. The term between brackets in the R.H.S. of equation(7) is an approximation of the Hessian matrix, while the term after it is the gradient.

3.2 The Proposed Neural Network

An ANN was constructed for simulation and prediction of the force and torque on the outer stationary sphere in eccentric sphere model. There are many different types of ANNs. The most widely used is the Back Propagation ANN. This type of ANN is excellent for performing classification task [26,27]. The developer of ANN has to answer the main question: which configuration of the ANN is appropriate for a good out-of-sample prediction? The configuration of ANN needs to be determined accurately to give an optimal classification result. This configuration includes the number of layers and the number of neurons for each layer. It is known that too many neurons degrade the effectiveness of the model and leads to the undesirable consequences, long training times and local minima. Large number of connection weights in the ANN model may cause over-fitting and loss of the generalization capacity. On the other hand, choosing too few neurons may not capture the full complexity of the data. Many heuristic

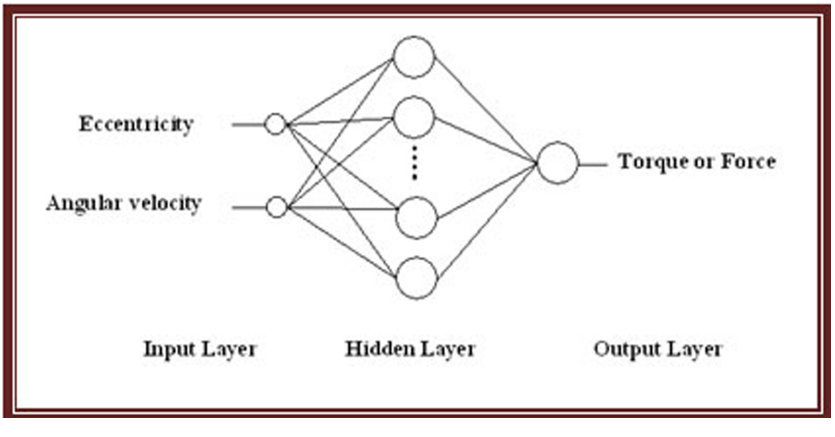


Fig. 3. The Proposed neural network configuration for torque or force

rules were suggested for finding the optimal number of neurons in the hidden layer. Most of them employ trial-and-error methods, in which the ANN training starts with a small number of neurons, and additional neurons are gradually added until some performance goal is satisfied [28]. Unfortunately, there is no theoretical consideration for determining the optimal network topology for the specific problem. A number of configurations sample of a feed-forward NN was considered to select among them the best one that gives the highest classification accuracy for our problem.

In each configuration two factors are changed. A configuration of NN started with one hidden layer and five neurons. Gradually the number of neurons is increased by five for five times to reach to forty neurons. Then, the number of hidden layers is increased by one for the same number of neurons. The proposed network consists of input layer, one hidden layer and an output layer. The input layer composes of two inputs in two cases the first is the angular velocity and the second is the eccentricity of the two spheres while the output in the first case is the force acting on the outer sphere and in the second case is the torque acting on the outer sphere. The output layer uses single neuron (force or torque) based on a linear activation function.

After many trials, we have used feed-forward neural networks with one hidden layer and forty neurons per hidden layer and the standard back propagation as the training algorithm. The hidden layer nodes use a sigmoid transfer function and the node in the output layer has a linear transfer function and the objective function to be minimized was the Mean Square Error (MSE). Figure (3) shows the structure of the proposed network for torque and force.

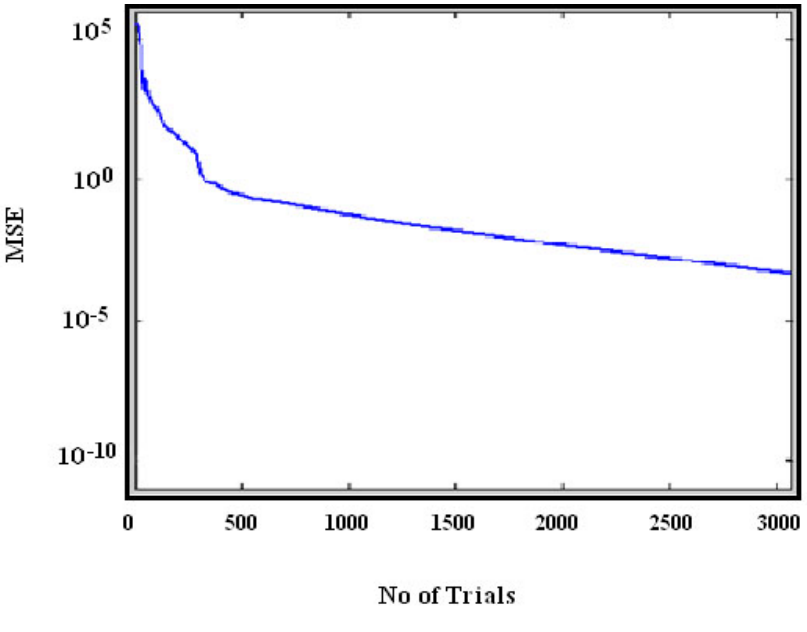


Fig. 4. Neural Network Training Performance for Force

The actual weights and biases for the designed networks are given in the appendix.

4 Results and Conclusion

The chosen neural network was trained on four cases of different eccentricities of the two spheres for force F_z which is acting in the z direction of the Cartesian coordinates on the outer stationary sphere as a function of the angular velocity. These values of eccentricities are 0.2, 0.4, 0.6 and 0.8. The training performances of the

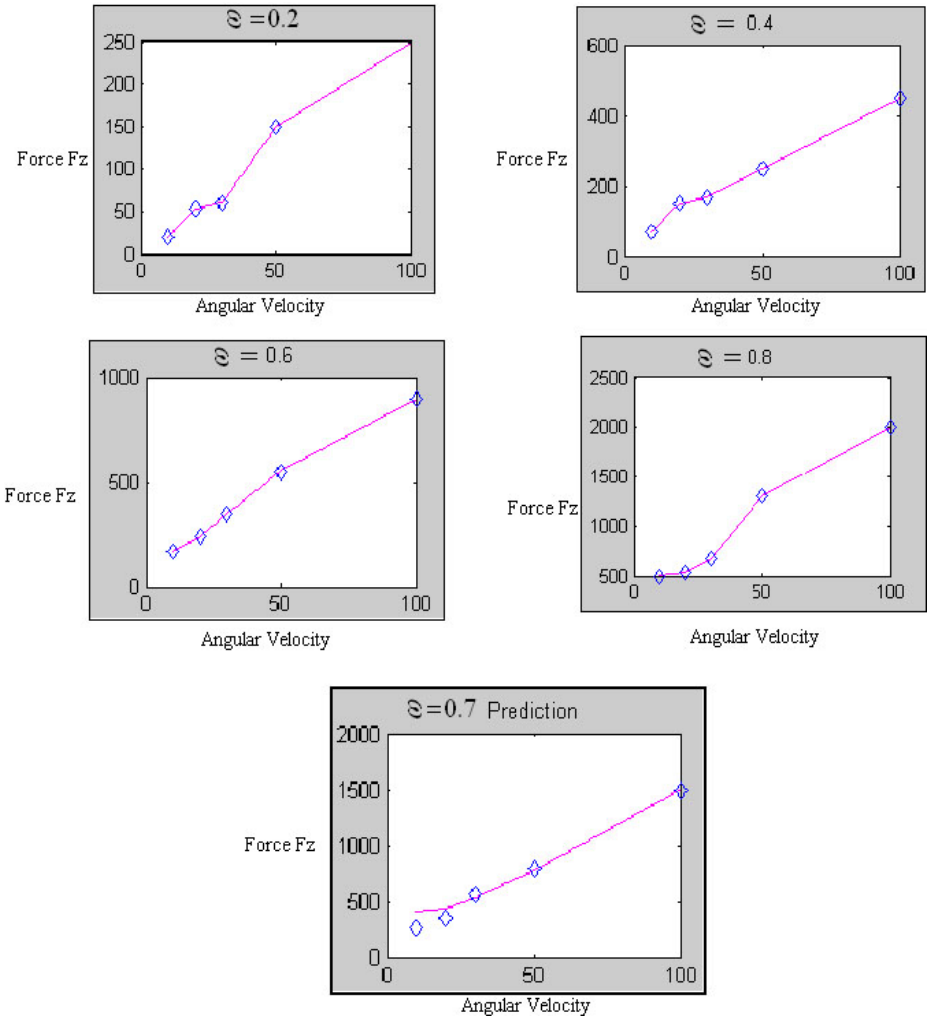


Fig. 5. The NN simulation of the force on the outer sphere in the z-direction of the eccentric sphere model

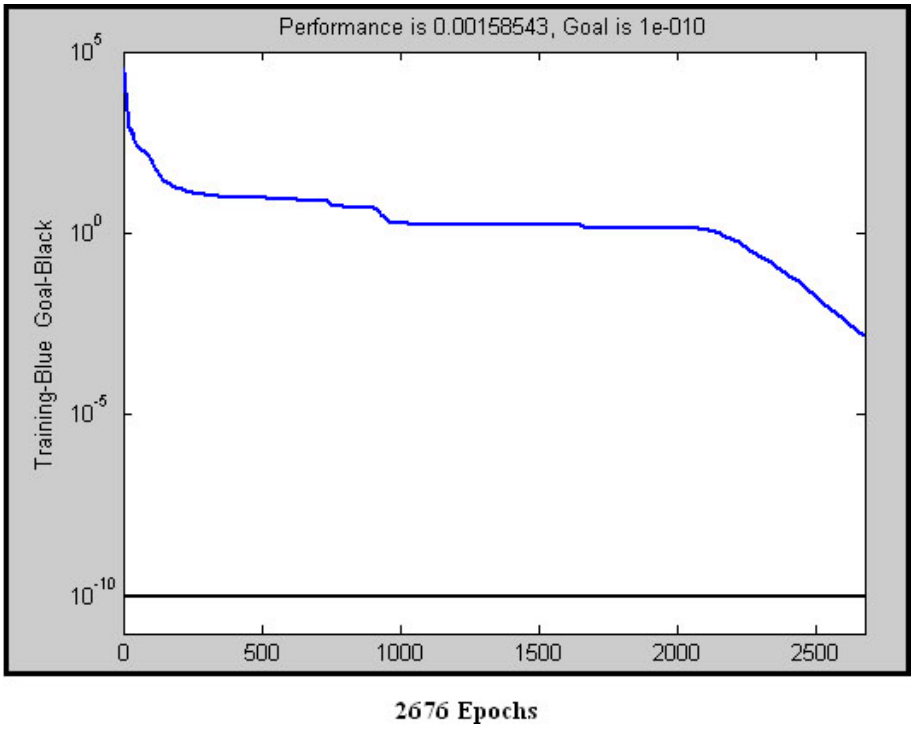


Fig. 6. Neural Network training performance for torque

obtained networks are shown in figure (4). In this figure, the training is stopped when the difference between the desired output and NN simulation output nearly equal to zero (0.00045). The obtained networks were tested for choosing the best one. This network was tested on the above mentioned four cases and used for predicting the case at eccentricity 0.7. Figure (5) shows the neural networks results of the four cases training and one predicted case for the change of the force versus angular velocity Figure (6) shows also the training performances of the torque acting on the outer stationary sphere as a function of the angular velocity. The values of training at eccentricities are 0.2, 0.4, 0.6 and 0.8 while the prediction value at eccentricity $\varepsilon = 0$ The four cases training and one predicted case for torque acting on the outer stationary sphere as a function of the angular velocity are shown in Figure (7). It was observed that these figures illustrate an excellent performance in two cases (the training and prediction).

4.1 Force Results

The following figure shows the four cases tested and one predicted data of the force acting on the outer sphere in (S.I.units)with angular velocity in (S.I.units) compared to the experimental data for 0.3 polyacrylamide in 50/50 glycerin/water.

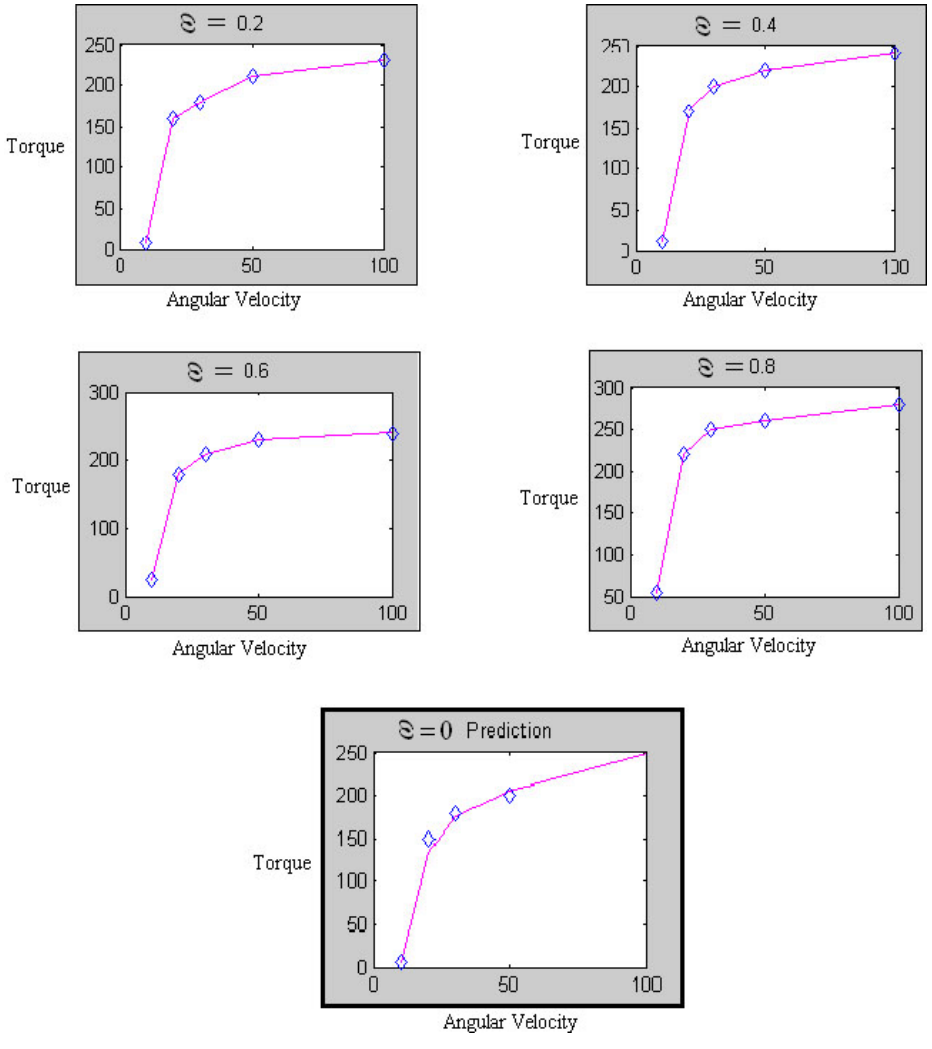


Fig. 7. The NN Simulation of the Torque on the Outer Sphere of the Eccentric Sphere Model

4.2 Torque Results

The Neural Network performance for torque acting on the outer sphere are shown in the figure(6). The following figure shows the four cases tested and one predicted data of the torque acting on the outer sphere in (S.I.units)with angular velocity in (S.I.units) compared to the experimental data for 0.3 polyacrylamide in 50/50 glycerin/water. Finally, the present work presents a new technique for modeling the force and torque on the outer stationary sphere in eccentric sphere model at different eccentricities based on ANN approach. The designed

ANN shows a good match to the experimental data.we used the ANN technique which gives a numerical solution which easier than many complicated theoretical methods and obtain a best fitting with the experimental data.

References

1. Labonte, G.: Neural Network reconstruction of fluid flows from tracer-particle displacements. Exp. Fluid 30, 399–409 (2001)
2. El Bakry, M.Y., El-Metwally, K.A.: Neural Network for Proton- Proton Collision at High Energy. Chaos, Solitons and Fractals 16(2), 279–285 (2003)
3. El Bakry, M.Y.: Feed Forward Neural Networks Modeling for K-P Interactions. Chaos, Solitons and Fractals 16(2), 279–285 (2003)
4. Altun, H., Bilgil, A., Fidan, B.C.: Treatment of multi-dimensional data to enhance neural network estimators in regression problems. Expert Systems with Applications 32(2), 599–605 (2007)
5. Sreenivasa Rao, K., Yegnanarayana, B.: Modeling durations of syllables using neural networks. Computer Speech and Language 21, 282–295 (2007)
6. Abu-ElHasan, A., Abdel Wahab, M., El-Bakry, M., Hemaid, S., Zidan, M.: Flow of a viscoelastic fluid between two rotating confocal ellipsoids. Proc. Math. phys. soc. Egypt 81, 37–57 (2004)
7. Abdel Wahab, M., Giesekus, H., Zidan, M.: A new eccentric-cylinder rheometer. Rheol. Acta 29, 16 (1990)
8. Wimmer, M.: Experiments on viscous flow between two rotating concentric spheres. J.of fluid mech. 78, 317 (1976)
9. Yamajuchi, H., Fujiyoshi, J., Matsui, H.: Viscoelastic fluid in spherical Couette flow, Part I; Experimental study for the inner sphere rotation. J. Non-Newtonian Fluid Mech. 69, 29 (1997)
10. Yamajuchi, H., Nishiguchi, B.: Spherical Couette flow of a viscoelastic fluid, Part III; A study of outer sphere rotation. J. Non-Newtonian Fluid Mech. 84, 45 (1999)
11. Bird, R.B., Armstrong, R.C., Hassager, O., Curtiss, C.F.: Dynamics of polemeric liquids. fluid mechanics, kinetic theory, vol. I, II . Wiley, New York (1987)
12. Giesekus, H.W., Kroner, E., Kirchasner, K.: Trends in application of pure mathematics to mechanics. Lecture notes in physics, vol. 244, pp. 331–348. Springer, Berlin (1986)
13. Truesdell, C., Noll, W.: Encyclopedia of physics. The Non linear field theories of mechanics, vol. III/3. Springer, Heidelberg (1965)
14. Jeffery, G.B.: Proc. London Math. Soc. 14(2), 327 (1915)
15. Stimson, M., Jeffery, G.B.: The motion of two spheres in a viscous fluid. Proc. Roy. Soc. A 3, 110–116 (1926)
16. Majumdar, S.R.: On the slow motion of viscous liquid in space between two eccentric spheres. J. Phy. Soc. Japan 26, 827–840 (1969)
17. Munson, B.R.: Viscous incompressible flow between eccentric coaxially rotating spheres. The Phys. of Fluids 17, 528 (1974)
18. Menguturk, M., Munson, B.R.: Experimental results for low reynolds number flow between eccentric rotating spheres. The Phys. of Fluids 18, 128 (1975)
19. Abu-El Hasan, A., Abdel Wahab, M., El-Bakry, M., Zidan, M.: Flow of fluids of grade two between two eccentric rotating spheres. ZAMP 47, 313 (1996)
20. El-Bakry, M.: Theoretical investigation as basis for a new rheometer which measure the Non linear mechanical properties of polymeric solutions or melts, Ph.D. thesis,Faculty of science, Benha, Egypt (1999)

21. Haykin, S.: Neural Networks: A comprehensive foundation. IEEE Press, Los Alamitos (1994)
22. Chang, T.S., Abdel-gaffar, K.A.S.: IEEE Transactions Signal Processing 40(12), 3022 (1992)
23. Haweel, T.I.: Neural digital filters design and implementation using signal decomposition. In: Proceeding of the fourth International Conference on Signal Processing Applications and Technology, Santa Clara, California, September 28, pp. 337–342 (1993)
24. Lippmann, R.: IEEE Trans.Acoust. Speech and signal processing 38(6), 938 (1990)
25. Hagan, M.T., Menhaj, M.B.: Training feed forward network with the Marquardt algorithm. IEEE Trans. on Neural Networks 5(6), 861–867 (1994)
26. Hecht-Nielsen, R.: Neurocomputing. Addison Wesley Publishing Co., Reading (1990)
27. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City (1991)
28. Boger, Z., Weber, R.: Finding an Optimal Artificial Neural Network Topology in Real-Life Modeling. In: Proceedings of the ICSC Symposium on Neural Computation, Article No. 1403/109 (2000)

Appendix

This appendix presents two figures Fig.8 and Fig.9 respectively, the first one is the produced weights and biases values of the best trained network of force while the second one presents the produced weights and biases values of the best trained network of the torque. Note: LW $(2,1)^T$ represents the transpose of LW (2,1). All titles in the these tables that appears in Fig.8 and Fig.9 were described in architecture of the proposed network.

Force F_z				
Weights			Biases	
IW (1,1)		$LW(2,1)^T$	b (1)	b(2)
4.261857	-47.7722	4.447034	37.35763	4.358905
0.467182	-20.4985	4.007949	48.48224	
0.437114	-17.2203	3.037318	46.32981	
3.379865	-54.7498	3.480524	51.15259	
0.157007	60.69904	100.196	-39.532	
0.037464	31.38419	65.41972	-21.6619	
4.428376	-12.6481	3.292672	3.900466	
0.089945	52.98905	480.3859	-48.01	
3.11767	-46.8697	4.091917	50.25568	
-0.04231	48.57828	47.00571	-23.6553	
0.126813	47.64036	399.7155	-44.447	
4.916478	-6.86304	3.083641	-3.2908	
0.624507	12.6929	41.39953	-34.6977	
0.014773	55.64327	92.1143	-37.4162	
0.96713	35.71022	20.59282	-36.7294	
3.998881	-53.8784	3.984947	41.25222	
5.326718	-20.7314	4.276593	1.5186	
16.36376	-38.2779	-60.5813	7.020694	
1.416684	0.155908	3.491424	25.63488	
2.648007	-35.4169	2.743388	39.03276	
6.245592	-45.2522	4.283485	20.54307	
0.19827	28.03398	279.276	-31.493	
5.957971	-52.3803	3.219833	29.31515	
4.901481	-52.7743	3.331039	35.15203	
0.033841	44.64499	56.59378	-31.5374	
3.741873	-32.4926	3.325593	34.32564	
-0.214	53.30805	94.76438	-10.2447	
4.347071	-47.7061	2.980916	33.91784	
-0.32466	5.781841	1.906155	21.28778	
6.807591	8.490578	-42.3204	-7.10709	
0.752319	-4.95679	119.8509	-32.8552	
0.222978	4.007054	40.72892	-14.4817	
-0.09813	57.8128	24.18572	-25.5271	
2.950587	-14.9586	4.121077	20.7862	
1.641093	-44.8135	78.96745	-7.79258	
7.681366	-23.3282	4.228015	-9.88688	
0.331801	-11.8973	31.11621	-30.9419	
4.660616	-31.0335	3.777067	24.23424	
0.669568	-26.286	-56.1253	-21.9746	
0.046316	7.517631	175.6748	-8.25563	

Fig. 8. The produced weights and biases values of the best trained network of Force

Torque				
Weights			Biases	
IW (1,1)		$LW(2,1)^T$	b (1)	b(2)
2.86298	21.15694	8.872771	-43.5721	4.000564
-0.21375	44.02262	-114.178	-33.0707	
0.54277	17.6137	6.036898	-20.7129	
2.277469	12.7735	7.505518	-39.0144	
-0.53161	-4.07104	-39.9175	9.318646	
0.666336	-31.1917	5.36674	6.27678	
1.93191	25.65304	9.873435	-36.9473	
-0.84235	45.12387	-26.1068	-27.559	
1.438805	-15.1418	3.310889	34.48979	
0.796502	-44.3019	-8.74923	13.00816	
0.174284	-35.2783	3.470635	24.44213	
0.113479	4.013984	3.874864	-6.29374	
0.52299	38.61472	6.187222	-31.5338	
0.400067	17.44183	8.995279	-34.5549	
1.737016	23.75001	6.913798	-33.7756	
1.506075	-6.72413	9.526839	-13.7758	
1.58286	11.31362	8.501808	-27.6601	
0.212515	4.854011	4.294449	-5.07133	
0.11114	-43.5463	-22.1511	16.74606	
0.413947	28.26519	3.286346	-23.8338	
0.044284	-17.6564	3.140207	8.734215	
1.746776	14.43479	6.635955	-29.4302	
0.025076	-35.0547	50.75662	20.84582	
0.285244	28.64431	3.614828	-27.3815	
1.127803	16.45358	3.723491	-23.0685	
-0.05997	-1.63403	-34.8209	1.173702	
0.537639	18.14731	6.661433	-21.9328	
1.350239	-3.16966	8.965819	-20.334	
0.145234	9.133998	4.496178	-7.92334	
1.323438	5.4345	4.776229	-21.471	
1.817706	-13.9994	7.580408	-25.8897	
0.052911	-3622.79	22.76741	-2.51668	
0.783472	10.65375	6.269141	-16.2823	
0.000398	28.43553	131.8271	-22.6112	
1.46371	-25.4356	9.656231	-18.4497	
0.115797	-19.8198	13.31453	5.846889	
0.588638	8.561829	22.53962	-12.1574	
0.040182	26.6735	11.29443	-12.3826	
0.785214	378.5184	4.716749	-43.0192	
0.061481	6.822822	3.817806	37.02959	

Fig. 9. The produced weights and biases values of the best trained network of the torque

Author Index