Comput Mech (2014) 54:999–1012
DOI 10.1007/s00466-014-0990-8

ORIGINAL PAPER

A matrix-form GSM–CFD solver for incompressible fluids and its application to hemodynamics

Jianyao Yao · G. R. Liu

Received: 25 November 2013 / Accepted: 27 January 2014 / Published online: 16 February 2014

Abstract A GSM–CFD solver for incompressible flows is developed based on the gradient smoothing method (GSM). A matrix-form algorithm and corresponding data structure for GSM are devised to efficiently approximate the spatial gradients of field variables using the gradient smoothing operation. The calculated gradient values on various test fields show that the proposed GSM is capable of exactly reproducing linear field and of second order accuracy on all kinds of meshes. It is found that the GSM is much more robust to mesh deformation and therefore more suitable for problems with complicated geometries. Integrated with the artificial compressibility approach, the GSM is extended to solve the incompressible flows. As an example, the flow simulation of carotid bifurcation is carried out to show the effectiveness of the proposed GSM–CFD solver. The blood is modeled as incompressible Newtonian fluid and the vessel is treated as rigid wall in this paper.

Keywords Incompressible GSM–CFD · Gradient smoothing · Matrix-form · Carotid bifurcation

1 Introduction

Computational fluid dynamics (CFD) has become a standard and popular tool widely utilized in almost all aspects of fluid dynamics for both engineering and academic fields, including aerospace, automotive, chemical reaction, biomechanics and so on. With the dramatic increases of computer power and advances in algorithm efficiency, the development and application of CFD packages are always an active topic in many decades for better robustness in handling complicated engineering problems. The numerical methods used in CFD generally fall into three categories, finite difference method (FDM), finite element method (FEM) and finite volume method (FVM), and the detailed comments on the comparisons of the methods are given in [28]. There are also some other approaches like smoothed-particle hydrodynamics (SPH) [24,25,32] and lattice Boltzmann method [1,8] that are developed for complex flows.

For hemodynamics, an important issue is how to model the blood and to simulate its dynamic features during the cardiac cycle. Physically, the blood is a complex mixture of cells, proteins, lipoproteins, and ions, by which nutrients and oxygen are delivered to the cells and the wastes and carbon dioxide are transported outside [18,20]. During the CFD computation at continuum-scale, the blood can be modeled as incompressible Newtonian [20,38,40,46,48,53] or non-Newtonian [13–17,36] flow. The non-Newtonian constitutive equations are necessary if the shear rate is less than $100\,\mathrm{s}^{-1}$ when the shear-thinning phenomenon become significant. In the regions of mid-range to high shear rate, the Newtonian constitutive laws can give a good approximation [13,15]. The detailed distributions of velocity, pressure and wall shear stress (WSS) can be obtained by using the CFD and fluid–structure interaction (FSI) tools, combined with the advanced medical image and modeling techniques, more accurate patient-specific model can be constructed and solved [2,6,7,38,40,42,46,48]. The advanced computational tools can help to investigate the pathogenesis, predict disease progression, design surgical planning, and can play an important role in the development of medical devices.

The general purpose CFD codes developed based on FEM [14,41,43–46], FVM [15,17] or commercial pack-

J. Yao (✉) · G. R. Liu
Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati, Cincinnati, Ohio 45221, USA
e-mail: yaojy@uc.edu

G. R. Liu
e-mail: liugr@uc.edu

ages [16,33] have been already used as the fluid solver or FSI solver for hemodynamics. Tezduyar and his colleagues [37] have conducted extensive research on patient-specific arterial FSI modeling of cerebral aneurysms for the recent decade using the deforming-spatial-domain/stabilized space-time (DSD/SST) formulation. This formulation is developed based on the streamline-upwind/Petrov–Galerkin (SUPG) and pressure-stabilizing/Petrov–Galerkin methods, and designed with moving mesh features to handle the flow problems with moving boundaries. The stabilized space-time FSI (SSTFSI) technique is developed based on DSD/SST formulations with various special techniques for the arterial FSI simulations, with emphasis on arteries with aneurysm. These techniques include a special mapping technique for determining the inlet velocity profile with non-circular shape [38], techniques for using variable arterial wall thickness [38,39], mesh generation techniques for building layered meshes near the arterial walls for CFD computation [38,39,46], the sequentially coupled arterial FSI (SCAFSI) technique [46,47], and techniques for the projection of fluid–structure interface stresses, calculation of the WSS [39] etc. The detailed descriptions of these techniques are well summarized in [37].

Another popular approach for FSI simulation of hemodynamics is to use arbitrary Lagrangian–Eulerian (ALE) formulation to handle the moving mesh and the interaction between fluid and solid [11,31,35,50]. In the ALE formulation, the displacement of the fluid domain is introduced as a third field of solution variables in the coupled systems. An auxiliary solver may be needed to compute the deformation of the meshes, and proper coupling boundary conditions (continuity of velocities and traction forces) are applied on the fluid–structure interface.

As the first step of the development of a new FSI formulation, the purpose of this paper is to present an incompressible CFD solver based on the gradient smoothing method (GSM) [21,22,28]. The GSM has been recently developed as a strong form method for solving general partial differential equations (PDEs) for both solid and fluid dynamics, named as smoothed finite element method (S-FEM) [23,26,27] and GSM–CFD [19,28,51,52,54], respectively. In the proposed GSM–CFD solver, the gradient smoothing operation is consistently used to approximate the first and second order derivatives at different locations. Different types of smoothing domains and smoothing functions can be constructed based on compactness, efficiency and accuracy. The previous research on GSM–CFD has shown that the solver is conservative, conformal, stable, efficient and accurate [28,51,54]. Another very attractive feature of GSM–CFD is that it is insensitive to mesh distortion. This is favorable to FSI simulation where the mesh deforms due to the moving boundaries and thus the mesh quality cannot be guaranteed.

The Green's theorem is adopted for gradient approximation in both GSM and widely used FVM–CFD solvers, and thus the two methods can be considered as similar approaches in this regard. However, when approximating the first-order derivatives, different smoothing functions [19] can be adopted to provide results with different orders of accuracy. That means the GSM is a more versatile, flexible and designable method. Another crucial difference is how to approximate the second-order derivatives of a field variable, for which the first-order derivatives at the midpoint of cell edges are needed. In the standard FVM, the first-order derivatives at the midpoint of cell edges are usually obtained by interpolating the derivatives at the two end points of the edge. Therefore, special ad hoc techniques are required to "correct" the interpolation to overcome the checkerboard problem. A typical choice of this kind of ad hoc techniques is the directional correction that are currently widely used for CFD solvers [4,34]. Such an technique could be very useful for many cases, however, it can fail in some cases, especially on domain boundaries. A systematic and fully proofed technique for second-order derivative approximation is needed. The GSM uses, in a consistent manner, smoothing operations for approximating both first- and second-order derivatives on the smoothing domains created in a nested fashion. This not only removes the need for any ad hoc technique for correction, it also ensures conservation at all times even for highly distorted meshes. This has been confirmed for 2D cases [28,54]. In this paper, we focus on the verification of 3D GSM on the structured and hybrid meshes.

It should be noted that the GSM–CFD is originally designed for unstructured T-mesh (triangular for 2D and tetrahedral for 3D) to reduce the meshing difficulties of complex geometries. However, for viscous and turbulent flows, we prefer the structured mesh with large aspect ratio to discretize the boundary layer regions to reduce the number of nodes and elements, especially for 3D cases where the hexahedral or prismatic mesh should be used instead of tetrahedral mesh. During the generation of hybrid mesh, first, the boundary surfaces are discretized using triangles or quadrangles or their mixture, and several layers of hexahedral or prismatic elements "grow" orthogonally from the discretized surfaces. For the other regions, the unstructured mesh can still be adopted. Since the previous research on GSM–CFD mainly focus on 2D unstructured mesh, we need to extend the GSM–CFD to 2D and 3D structured and hybrid meshes, for which a clean data structure is needed for effective implementations. In this paper, a new matrix-form algorithm for approximating gradients using GSM is developed, and the corresponding data structure is also devised to improve the numerical efficiency. As indicated in [13,18], for a medium-sized artery, the blood can be treated as laminar incompressible Newtonian flow, thus only the GSM–CFD solver for incompressible Navier–Stokes is developed using the artificial compressibility approach at this step, and will be further developed for non-Newtonian and turbulent flows.

The codes for GSM–CFD is developed using C++ based on an open-source SU2 (Stanford University Unstructured) tool suite [34], which is a collection of C++ software tools for solving PDEs, especially focusing on fluid dynamics. During the development of GSM–CFD solver, we can use the framework of SU2 and reuse the basic codes such as mesh manipulation and linear equation solvers. Our efforts are focused on the development of new modules for construction of gradient smoothing domains (GSD) and spatial gradients approximation using GSM.

There are six more sections in this paper. In Sect. 2, different kinds of GSDs are constructed on 3D primitive meshes, and a uniform expression for GSD is given. The construction of local and global sparse matrix for gradient calculation and the algorithm for matrix-form GSM are introduced in Sect. 3. The verifications of gradient approximation on structured meshes are carried out in Sect. 4. In Sect. 5, the incompressible GSM–CFD solver are developed using GSM and artificial compressibility method. The numerical example for 3D unsteady carotid bifurcation is given in Sect. 6 to show the effectiveness of the proposed GSM–CFD solver and some conclusions are drawn in the final section.

2 Construction of smoothing domains

2.1 Gradient smoothing operations

The gradients of a field variable U at a point $\mathbf{x}_i$ in domain Ω_i can be approximated in the integral form of [22,25,30]

$$\nabla U_i \equiv \nabla U(\mathbf{x}_i) \approx \int\limits_{\Omega_i} \nabla U(\mathbf{x})\hat{w}(\mathbf{x}-\mathbf{x}_i)\mathrm{d}\Omega \tag{1}$$

where ∇ is the gradient operator, and $\hat{w}$ is a smoothing function. The smoothing function should satisfy the following conditions:

1. Unity condition: the integration of smoothing function over the smoothing domain should produce unity.

$$\int\limits_{\Omega_i} \hat{w}(\mathbf{x}-\mathbf{x}_i)\mathrm{d}\Omega = 1 \tag{2}$$

2. Compact condition: define the effective (nonzero) domain.

$$\hat{w}(\mathbf{x}-\mathbf{x}_i) = 0 \quad \text{when} \quad \mathbf{x} \notin \Omega_i \tag{3}$$

If the above conditions are satisfied, the smoothing function can be properly chosen based on the accuracy required by the gradient approximation. The piecewise constant smoothing function is mostly used in our previous works, and it can be written as

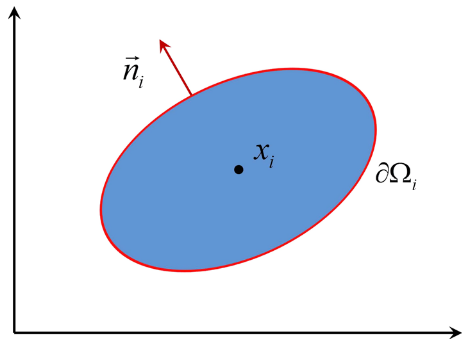


Fig. 1 Illustration for gradient smoothing operation

$$\hat{w} = \begin{cases} 1/V_i, & \mathbf{x} \in \Omega_i \\ 0, & \mathbf{x} \notin \Omega_i \end{cases} \tag{4}$$

where V_i is the volume of smoothing domain Ω_i.

Integrating Eq. (1) by parts gives

$$\nabla U_i \approx \int\limits_{\partial\Omega_i} U(\mathbf{x})\hat{w}(\mathbf{x}-\mathbf{x}_i)\mathbf{n}\mathrm{d}s - \int\limits_{\Omega_i} U(\mathbf{x})\nabla\hat{w}(\mathbf{x}-\mathbf{x}_i)\mathrm{d}\Omega \tag{5}$$

where $\partial\Omega_i$ represents the external boundary of the gradient smoothing domain, and $\mathbf{n}$ denotes the outward unit normal vector of $\partial\Omega_i$, as shown in Fig. 1. We can see that there are two parts of the integration, the boundary integration and the domain integration. If the piecewise constant smoothing function is adopted, the domain integration vanishes and we have

$$\nabla U_i \approx \frac{1}{V_i} \oint\limits_{\partial\Omega_i} U\mathbf{n}ds \tag{6}$$

For the approximation of second-order spatial gradients, we can successively use the gradient smoothing operation with piecewise constant smoothing function, and have

$$\nabla \cdot (\nabla U_i) \approx \frac{1}{V_i} \oint\limits_{\partial\Omega_i} \mathbf{n} \cdot \nabla U \mathrm{d}s \tag{7}$$

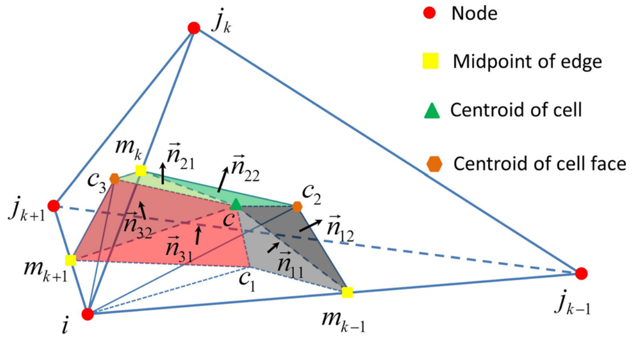
We should note that the first-order derivatives at the boundary of the smoothing domain is needed for second-order gradients approximation. Hence, the spatial derivatives at any location of interest can be approximated using Eqs. (6) and (7) together with the properly defined smoothing domains.

The first step for gradient approximation using GSM is to construct smoothing domains on the primitive mesh. Basically, there are two kinds of smoothing domains, the node-associated gradient smoothing domain (nGSD) for the gradient approximation at node, and the midpoint-associated gradient smoothing domain (mGSD) for the gradient approximation at midpoint of edge. The triangular and quadrilateral meshes are usually used to discretize the 2D domains, and the tetrahedral and hexahedral, as well as the prismatic and pyramidal meshes are usually used for 3D domains. The construction of nGSD and mGSD will be introduced in the following subsections.

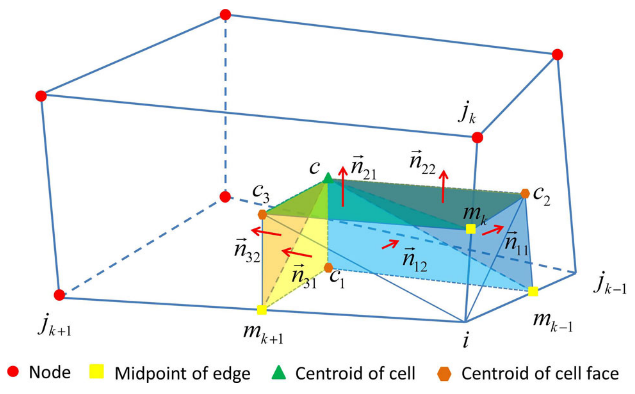
2.2 Construction of nGSD

The nGSD in 3D is constructed in a similar way as 2D [28] by connecting the midpoint of edges, the centroid of cell faces (denoted as c_i in the figure) and the centroid of cells surrounding the node of interest, as shown in Fig. 2. To illustrate the smoothing domains more clearly, we only show one partition of the nGSD within the cell. In Fig. 2, the shaded faces denote the outer surfaces of the nGSD, and $\mathbf{n_{ij}}$ denotes the outward normal vectors of the boundary surfaces of the smoothing domain. It should be pointed out that the four points which form the quadrilateral outer surface, for example, the edge midpoint m_k, two cell face centroids c_2 and c_3, and the cell centroid c, may not lie on the same plane, especially when moving mesh is involved. Thus, during the computation, we usually divide this quadrilateral into two triangles both connected by the edge midpoint, cell face centroid and the cell centroid. In this way, all the nGSDs are conservative and compact, and thus the local and global conservation are ensured.

To sum up, the nGSDs are constructed by all the subedges connecting the edge midpoint and cell centroid in 2D domains, or all the subtriangles connecting the edge midpoint, face centroid and cell centroid in 3D domains, which finally form a closed polygon for 2D or a closed polyhedron for 3D which surrounds the node of interest.



(a)

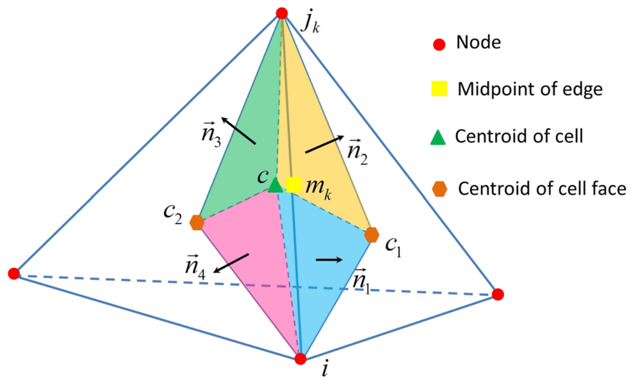


(b)

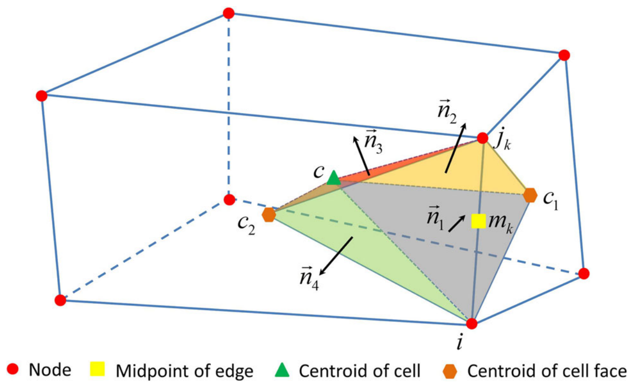
Fig. 2 Node-associated gradient smoothing domain (nGSD) within a cell. **a** nGSD in a tetrahedral cell and **b** nGSD in a hexahedral cell

2.3 Construction of mGSD

For the compact mGSD in 3D domain, the smoothing domain is constructed by connecting the endpoints of the edge of interest, the centroids of cell faces and the centroids of of cells surrounding the edge of interest. To show the smoothing domain more clearly, we plot the mGSD within a cell in Fig. 3. The shaded faces in the figure denote the outer surfaces of a part of the smoothing domain for edge $i - j_k$ within the current cell.



(a)



(b)

Fig. 3 Compact midpoint-associated gradient smoothing domain (mGSD) within a cell. **a** mGSD in a tetrahedral cell and **b** mGSD in a hexahedral cell

To sum up, the mGSDs are constructed by all the subedges connecting the edge endpoint and cell centroid in 2D domains, or all the subtriangles connecting the edge endpoint, face centroid and cell centroid in 3D domains, which finally form a closed polygon in 2D or a closed polyhedron in 3D surrounds the edge of interest. The mGSDs constructed in this fashion are also compact, and both locally and globally conservative.

3 Spatial gradient approximation based on GSM

Using the gradient smoothing operation, the spatial gradients can be approximated in an integral form on the nGSD and mGSD constructed on the primitive mesh. In this section, the matrix-form GSM is introduced for the gradient approximation, by which the gradients at nodes or midpoint of edges can be calculated as the product of a sparse matrix and a vector storing the field variables. In finite element method, the strains (gradients of displacements at nodes) can be calculated using a strain–displacement matrix **B**. Similarly, we can construct a **B** matrix for gradient approximation based on GSM rather than the shape function in FEM, and the gradients can be calculated as

$$\frac{\partial \mathbf{U}}{\partial s} = \mathbf{B} \cdot \mathbf{U} \tag{8}$$

Apparently, **B** is a block-wise sparse matrix, and can be assembled from the element matrix $\mathbf{B}_e$ like in FEM. The first and the most important task of gradient approximation is to assemble global **B** matrix. It should be noted that the assemblies of the global **B** matrix for both nGSD and mGSD are only need to be carried out once if the mesh keeps unchanged.

3.1 Approximation of first-order gradients at nodes

The global **B** matrix for gradient approximation at nodes should have $N \times N$ blocks, including all the zero and nonzero blocks, where N is the number of nodes in the domain. For the ith row of **B**, the nonzero blocks only exist at the nodes associated or surrounding node i. When we assemble the global matrix, we only need to accumulate the element matrix $\mathbf{B}_e$ to its corresponding global indexes.

During assembling process, for any element or cell, each edge has its associated normals as shown in Fig. 4a. It should be noted that these kinds of edge data structures are independent of element types. If the piecewise constant smoothing function and the one-point quadrature scheme [28] are used, the flux contribution of current edge $i - j_k$ to its two endpoints can be written as

$$\mathbf{U}_{ij_k,i} = U_{m_k} A \frac{\mathbf{n}}{|\mathbf{n}|}, \quad \mathbf{U}_{ij_k,j_k} = -U_{m_k} A \frac{\mathbf{n}}{|\mathbf{n}|} \tag{9}$$

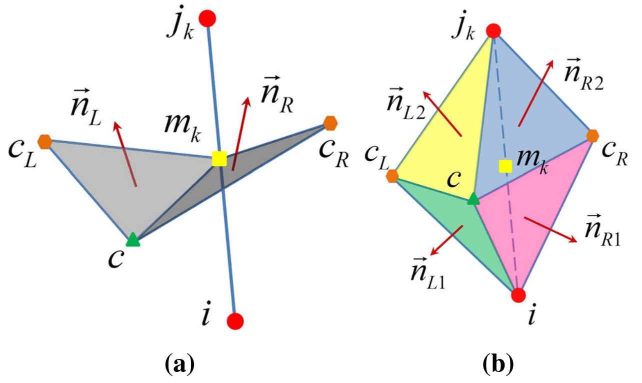


Fig. 4 Edge structure for assembling $\mathbf{B}_e$. **a** Edge structure for nGSD and **b** edge structure for mGSD

given that the direction of vector **n** is defined as from node i to node j. In the above equation, A denotes the sum of two subtriangles ($C_L - c - m_k$ and $C_R - c - m_k$), and $U_{m_k} = (U_i + U_{j_k})/2$. Therefore, the block-wise contribution of edge $i - j_k$ to the current element written as: for node i, i.e., the ith row of $\mathbf{B}_e$,

$$\begin{aligned} \mathbf{B}_e(i,i) &= \mathbf{B}_e(i,i) + \frac{1}{2} A \frac{\mathbf{n}}{|\mathbf{n}|} \\ \mathbf{B}_e(i,j_k) &= \mathbf{B}_e(i,j_k) + \frac{1}{2} A \frac{\mathbf{n}}{|\mathbf{n}|} \end{aligned} \tag{10}$$

and for node j_k, i.e., the j_kth row of $\mathbf{B}_e$,

$$\begin{aligned} \mathbf{B}_e(j_k,j_k) &= \mathbf{B}_e(j_k,j_k) - \frac{1}{2} A \frac{\mathbf{n}}{|\mathbf{n}|} \\ \mathbf{B}_e(j_k,i) &= \mathbf{B}_e(j_k,i) - \frac{1}{2} A \frac{\mathbf{n}}{|\mathbf{n}|} \end{aligned} \tag{11}$$

The element $\mathbf{B}_e$ matrix is fully assembled as soon as the loop of edges in this element is finished, and we have $\mathbf{B}_e$ as a $N_n \times N_n$ block-wise matrix as

$$\mathbf{B}_e = \begin{bmatrix} \mathbf{B}_e(1,1) & \cdots & \mathbf{B}_e(1,j) & \cdots & \mathbf{B}_e(1,N_n) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathbf{B}_e(i,1) & \cdots & \mathbf{B}_e(i,j) & \cdots & \mathbf{B}_e(i,N_n) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathbf{B}_e(N_n,1) & \cdots & \mathbf{B}_e(N_n,j) & \cdots & \mathbf{B}_e(N_n,N_n) \end{bmatrix} \tag{12}$$

where N_n is the number of nodes in the element.

When the element $\mathbf{B}_e$ is obtained, we can accumulate it to the global **B** as

$$\mathbf{B}(I,J) = \mathbf{B}(I,J) + \mathbf{B}_e(i,j) \tag{13}$$

where I and J are the global indexes of local nodes i and j. It should be noted that if **B** is formed using the above procedures, we only accumulate the boundary fluxes around the node of interest, the gradients calculate by Eq. (8) still need to be divided by the volume of the smoothing domain V_i, i.e.,

$$\frac{\partial \mathbf{U}_i}{\partial s} = \frac{\partial \mathbf{U}_i}{\partial s} / V_i \tag{14}$$

3.2 Approximation of first-order gradients at edge midpoint

The approximation of gradients at edge midpoints can also be realized using matrix vector multiplication as Eq. (8), and the global **B** matrix for mGSD can also be assembled in a similar way as for nGSD. The **B** matrix for mGSD should have $N_e \times N$ blocks, including all the zero and nonzero blocks, where N_e is the number of edges in the domain. For the ith row of **B**, the nonzero blocks only exist at the nodes associated or surrounding edge i. When we assemble the global matrix, we only need to accumulate the element matrix $\mathbf{B}_e$ for mGSD to its corresponding global indexes. During assembling process, for any element or cell, each edge has its associated normals for mGSD as shown in Fig. 4b. It should be noted that these kinds of edge data structures are also independent of element types same as for nGSD.

For each edge in the current element, the edge data structure includes the node indexes of the two endpoints, and the corresponding normals of boundary surfaces of part of the mGSD within this element. Since the normals on the boundaries are quite different, we cannot use the one-point quadrature for mGSD, and we have to calculate the fluxes on each boundary separately. For 3D cases, it can be written as the sum of the fluxes through the four subtriangles

$$\mathbf{U}_{ie} = \sum_{s=1}^{4} \frac{\mathbf{n}_s}{|\mathbf{n}_s|} U_s A_s \tag{15}$$

In the above equation, $\mathbf{n}_s/|\mathbf{n}_s|$ and A_s denote the unit outward normal and the area, and U_s denotes the average field variable at center of the boundary surface of smoothing domain.

For 3D cases, since there are three points in the each subtriangle which serves as the boundary face of the mGSD, the nodes in the elements will fall into three levels with different contributions to the edge flux. For each node in the element, they will all contribute to the element average U_c like

$$\mathbf{B}_e(ie, m) = \mathbf{B}_e(ie, m) + \frac{1}{3N_n}\left(\frac{\mathbf{n}_{L1} A_{L1}}{|\mathbf{n}_{L1}|} + \frac{\mathbf{n}_{L2} A_{L2}}{|\mathbf{n}_{L2}|} + \frac{\mathbf{n}_{R1} A_{R1}}{|\mathbf{n}_{R1}|} + \frac{\mathbf{n}_{R2} A_{R2}}{|\mathbf{n}_{R2}|}\right) \tag{16}$$

For the nodes on the left or the right surface, there will be an extra contribution to the element face center. Take the left face as example, we have

$$\mathbf{B}_e(ie, k) = \mathbf{B}_e(ie, k) + \frac{1}{3}\left(\frac{\mathbf{n}_{L1} A_{L1}}{|\mathbf{n}_{L1}|}\frac{1}{N_{fcL1}} + \frac{\mathbf{n}_{L2} A_{L2}}{|\mathbf{n}_{L2}|}\frac{1}{N_{fcL2}}\right) \tag{17}$$

For the two endpoints of the edge $i - j_k$, the extra contribution to the node itself in the boundary surfaces of the smoothing domain can be written as

$$\begin{aligned}
\mathbf{B}_e(ie, i) &= \mathbf{B}_e(ie, i) + \frac{1}{3}\left(\frac{\mathbf{n}_{L1} A_{L1}}{|\mathbf{n}_{L1}|} + \frac{\mathbf{n}_{R1} A_{R1}}{|\mathbf{n}_{R1}|}\right) \\
\mathbf{B}_e(ie, j_k) &= \mathbf{B}_e(ie, j_k) + \frac{1}{3}\left(\frac{\mathbf{n}_{L2} A_{L1}}{|\mathbf{n}_{L2}|} + \frac{\mathbf{n}_{R2} A_{R1}}{|\mathbf{n}_{R2}|}\right)
\end{aligned} \tag{18}$$

From the above equations, we can clearly see the different weights of all the nodes in the elements to the edge fluxes. The two endpoints have the largest weights and then the nodes share the same faces as the edge endpoints, and all the other points in the element have the same smallest weights.

The element $\mathbf{B}_e$ matrix is fully assembled after finishing the loop of all the edges in the current element, and we have an $N_e \times N_n$ block-wise matrix with N_e is the number of edges and N_n is the number of nodes in the element. Also, we can accumulate the element $\mathbf{B}_e$ matrix to the global **B** matrix according to the edge and node indexes as

$$\mathbf{B}(IE, I) = \mathbf{B}(IE, I) + \mathbf{B}_e(ie, i) \tag{19}$$

where IE and I are the global indexes of local edge index ie and local node index i. When the global **B** matrix is obtained, the gradients at edge midpoints can be calculated as the product of sparse matrix **B** and variable vector **U** as Eq. (8). Similarly to nGSD, we have to divide the gradients by the volume of corresponding mGSD.

4 Verification of gradient approximation on mGSD

The GSM with piecewise constant smoothing function on nGSD is the same as the Green–Gauss theorem used for gradient approximation in traditional FVM. The validation and verification of this kind of method are already well established. The GSM for gradient approximations are also extensively verified on unstructured meshes [19,28,54]. Therefore, we mainly focus on the verification of mGSD on hexahedral and prismatic mesh for 3D. For CFD applications, the gradient at midpoint of edges are used to calculate the viscous terms. In this region, the structured meshes with large aspect ratio are usually used and the mesh quality can be affected greatly when moving boundaries are involved.

4.1 Basic equations for gradient verification

The most straightforward method to calculate the gradients at the midpoint of an edge is to obtain simple arithmetic mean of the gradients at the two endpoints of the edge as

$$\frac{\partial U_{m_k}}{\partial s} = \frac{1}{2}\left(\frac{\partial U_i}{\partial s} + \frac{\partial U_{j_k}}{\partial s}\right) \tag{20}$$

However, it is known that it can lead to decoupling solution (checkerboard problem) when approximating the second-order gradients (gradients of viscous fluxes for CFD). The average gradient method with directional correction [10,29, 49] (denoted as CAVG) is proposed to circumvent the decoupling problem. The directional correction takes the form of

$$\nabla U_{m_k} = \overline{\nabla U}_{m_k} - \left[\overline{\nabla U}_{m_k} \cdot \mathbf{t}_{ij_k} - \left(\frac{\partial U}{\partial l}\right)_{ij_k}\right] \tag{21}$$

where $\overline{\nabla U}_{m_k}$ is the simple average of gradients calculated using Eq. (20), and

$$\left(\frac{\partial U}{\partial l}\right)_{ij_k} \approx \frac{U_{j_k} - U_i}{\Delta l_{ij_k}} \tag{22}$$

$$\mathbf{t}_{ij_k} = \mathbf{x}_{j_k} - \mathbf{x}_i \tag{23}$$

$$\Delta l_{ij_k} = |\mathbf{x}_{j_k} - \mathbf{x}_i| \tag{24}$$

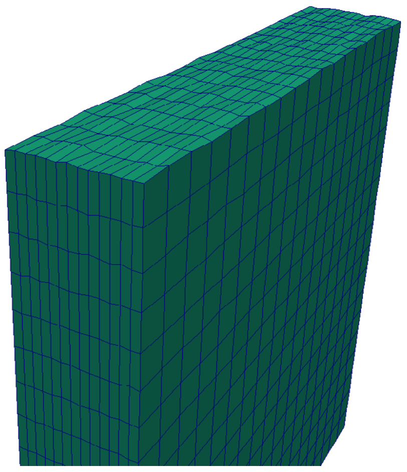
The modification leads to strongly coupled stencils on tetrahedral as well as on prismatic or hexahedral grids [4], and therefore prevent the checkerboard problem. Here, we will compare the gradient results obtained from GSM and CAVG. The magnitude of gradient errors at each edge midpoint is calculated as

$$\epsilon = \sqrt{\sum_{i=1}^{nDim}\left(\left.\frac{\partial U_m}{\partial x_i}\right|_a - \left.\frac{\partial U_m}{\partial x_i}\right|_n\right)^2} \tag{25}$$

where $|_a$ and $|_n$ denote the analytical and numerical solution of the gradient. The relative error of gradient approximation is defined as the ratio of absolute error norm to the norm of accurate gradients.

4.2 Verification of mGSD on perturbed hexahedral and prismatic mesh

The verifications of 3D gradient approximation are carried on a cube domain by 0.5 × 0.5 × 0.1. The perturbed hexahedral and prismatic meshes are used to discretize the 3D domain as shown in Fig. 5. The perturbed meshes are generated by slightly and randomly changing the locations of nodes and keeping the boundaries unchanged. The variable fields are chosen to be linear, quadratic polynomial, harmonic and exponential.


(a)

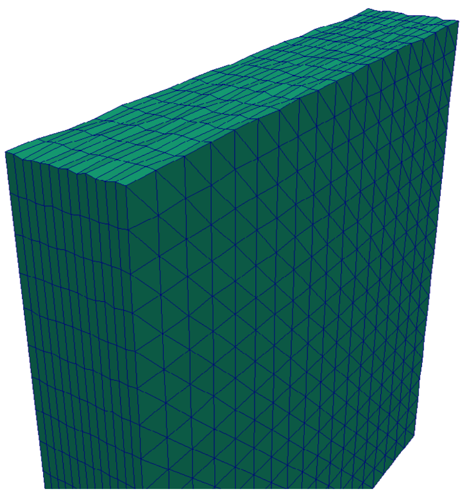

(b)

Fig. 5 3D perturbed structured hexahedral and prismatic mesh of a square domain for gradient verification of mGSD. There are 6,000 cells and 7,506 nodes in the mesh in the hexahedral mesh, and 12,000 cells and 7,506 nodes in the prismatic mesh. **a** Perturbed hexahedral mesh and **b** perturbed hexahedral mesh

$$\begin{aligned} U(x,y,z) &= x + 2y + 3z \\ U(x,y,z) &= (x + 2y + 3z)^2 \\ U(x,y,z) &= z\sin 2\pi x \cos 2\pi y \\ U(x,y,z) &= e^{x+2y+3z} \end{aligned} \tag{26}$$

The norms of relative errors of gradient approximations on the regular and perturbed meshes are summarized in Table 1. From the error data, we can see that:

1. On the perturbed hexahedral mesh, the GSM can exactly reproduce the linear field almost to machine accuracy while CAVG produces much large error, see Fig. 6.

Table 1 Relative errors of different variable fields on 3D domains

Function	Linear	Polynomial	Harmonic	Exponential
CAVG-H[a]	1.27E−2	1.49E−2	1.22E−2	5.87E−3
GSM-H[a]	1.40E−13	1.01E−3	9.39E−3	5.16E−4
CAVG-P[b]	1.29E−2	1.47E−2	1.74E−2	5.82E−3
GSM-P[b]	2.34E−13	1.07E−3	1.07E−2	5.02E−4

[a] H denotes the results are calculated on perturbed hexahedral mesh
[b] P denotes the results are calculated on perturbed prismatic mesh

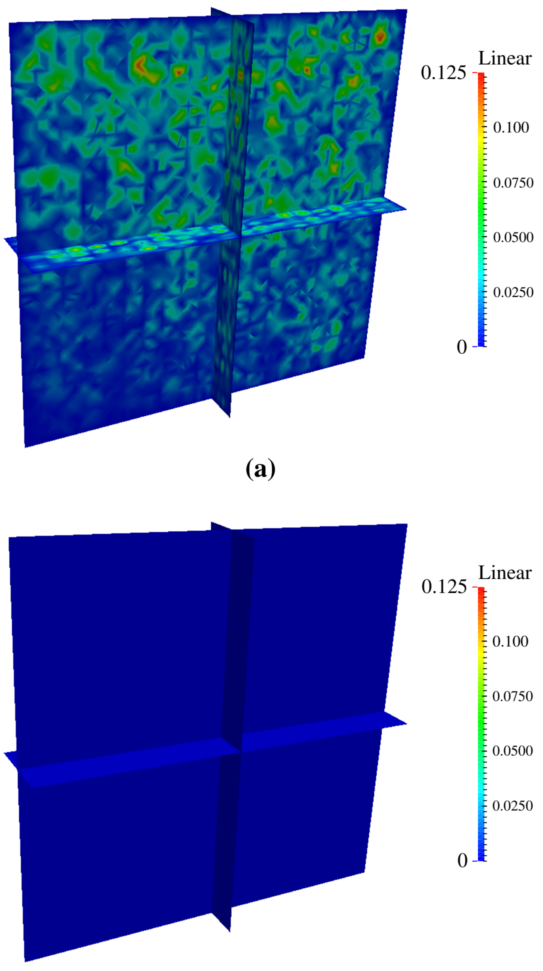


Fig. 6 Distributions of gradient errors of linear field $U(x, y, z) = x + 2y + 3z$ on perturbed hexahedral mesh. **a** CAVG and **b** GSM

2. On this perturbed mesh, neither method can reproduce the quadratic field anymore, but the gradient results obtained using GSM are much better than the those obtained using CAVG. For both quadratic and exponential fields, the relative error norms of CAVG are several times or 1–2 orders of magnitude larger than those obtained using GSM. For harmonic field, the two methods produce similar results, but the GSM results have smaller errors. The error distributions of exponential and harmonic fields are shown in Figs. 7 and 8.
3. When the regular prismatic mesh slightly perturbed as shown in Fig. 5b is used, GSM can still reproduce the linear field. For all the four kinds of variable fields, GSM consistently performs better CAVG method on the given perturbed mesh. The relative error norms obtained using GSM can be several times or even one order of magnitude smaller than the errors calculated using CAVG method, depending on the type of variable field. The error distributions on all the four variable fields are shown from Figs. 9, 10, 11 and 12.
4. GSM can exactly reproduce the linear field for all kinds of mesh due to the conservative property of the method. On the perturbed hexahedral and prismatic meshes, the GSM performs better than CAVG method for all the four variable fields considered here, and from the distributions of the gradient approximation errors, we can observe that the errors of CAVG are globally larger than those of GSM where the mesh is perturbed.

Through the gradient verifications, we can see that when the regular meshes are slightly perturbed, the GSM can get much better results than CAVG for all the variable fields considered here. Thus, we can conclude that the proposed GSM are more robust to mesh deformation than the CAVG method.

5 Solution procedures of GSM–CFD solver for incompressible flows

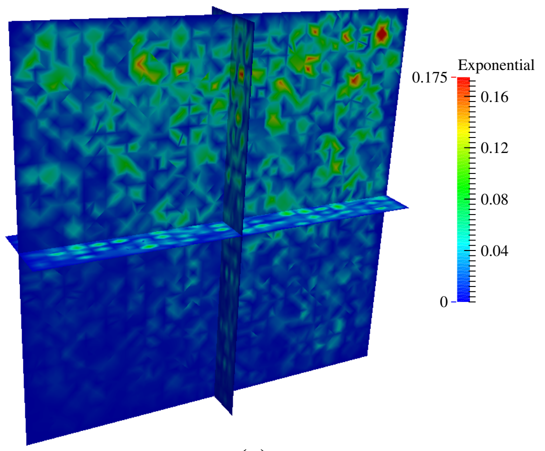
In this work, the artificial compressibility method [9] is used to solver the incompressible N–S equations. In this approach, the fluid is treated as being slightly compressible, so that an additional pseudo-time (τ) derivative of the pressure is added to the mass conservation equation, and the compact vector form governing equations can be written in a dual time step fashion as

$$\frac{\partial \mathbf{Q}}{\partial \tau} + \frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot (\mathbf{F}_c - \mathbf{F}_v) = 0 \tag{27}$$

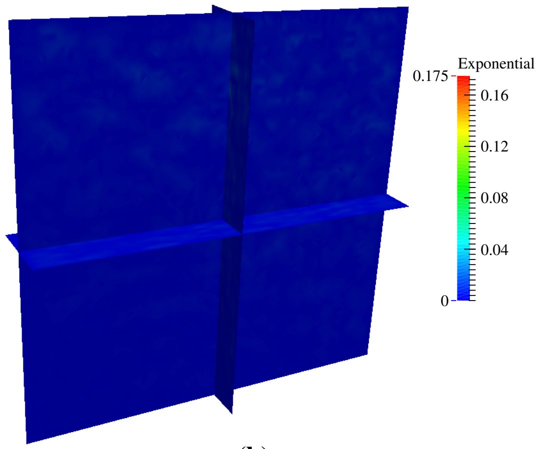
where $\mathbf{W}$, $\mathbf{F}_c$ and $\mathbf{F}_v$ represent, respectively, the vectors of conservative variables, the convective and viscous fluxes, and

$$\mathbf{Q} = \begin{bmatrix} p \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} 1 \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \end{bmatrix}, \quad \mathbf{F}_{cj} = \begin{bmatrix} \beta \rho u_j \\ u_1 u_j + p\delta_{1j} \\ u_2 u_j + p\delta_{2j} \\ u_3 u_j + p\delta_{3j} \end{bmatrix},$$

$$\mathbf{F}_{vj} = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \end{bmatrix} \tag{28}$$



(a)



(b)

Fig. 7 Distributions of gradient errors of exponential field $U(x, y, z) = e^{x+2y+3z}$ on perturbed hexahedral mesh. **a** CAVG and **b** GSM

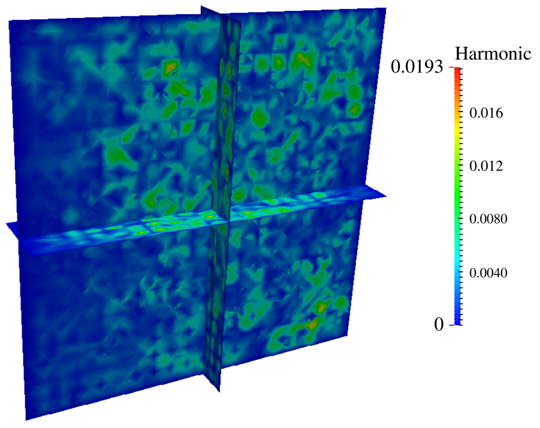
with δ_{ij} is the Kronecker delta, and u_j $(j = 1, 2, 3)$ denotes the three components of velocity. In the above equations, ρ, p, u, v and w, denote the density, static pressure and velocity components in x-, y- and z-direction under Cartesian coordinate system, respectively. The the viscous stresses of incompressible $(\nabla \cdot \mathbf{v} = 0)$ Newtonian fluids can be written as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{29}$$

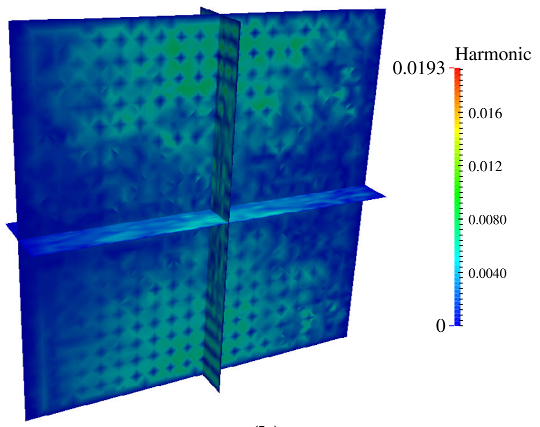
with μ denotes the dynamic viscosity.

Physically speaking, the parameter $\sqrt{\beta}$ is the speed of artificial pressure wave [52], but it can be tuned to speed up the convergence rate of the overall iterative solution procedures. Based on numerical experiments, the value of β is defined as [55]

$$\beta^2 = \max \left(\beta^2_{\min}, C_\beta |\mathbf{u}| \right) \tag{30}$$



(a)



(b)

Fig. 8 Distributions of gradient errors of exponential field $U(x, y, z) = z \sin 2\pi x \cos 2\pi y$ on perturbed prismatic mesh. **a** CAVG and **b** GSM

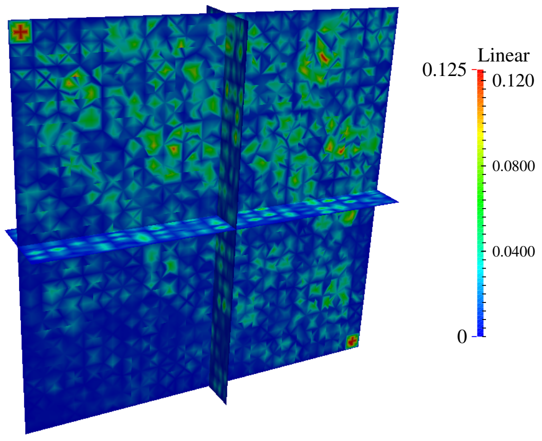
with $\beta^2_{\min} = 1$ and $C_\beta = 2.5$ for non-dimensionalized equations.

5.1 Approximation of convective fluxes

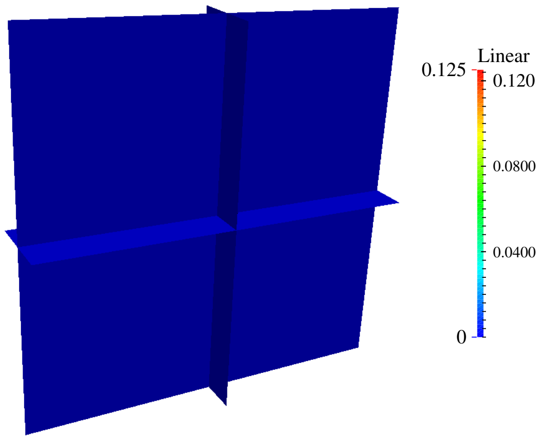
The most straightforward way to calculate the convective flux at the midpoint of cell edge is to use the average of the fluxes at the two endpoints as

$$(\mathbf{F_c})_{ij_k} = \frac{1}{2} \left[\mathbf{F}_c(\mathbf{Q}_L) + \mathbf{F}_c(\mathbf{Q}_R) \right] \tag{31}$$

where $\mathbf{Q}_L$ and $\mathbf{Q}_R$ denote the left and right state of the field variables, and they can be the variables at nodes for the most simple cases, but may need some reconstructions (such as different kinds of limiters [4]) to improve the stability or accuracy. However, for convection dominated flow, it is necessary to use upwind schemes rather than the simple linear interpolation of the fluxes at the two constitutive nodes to discretize the convective fluxes to guarantee the numerical stability and reduce the numerical diffusion. There are vari-

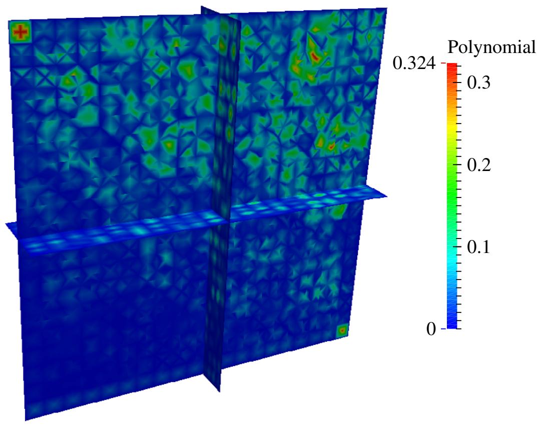


(a)

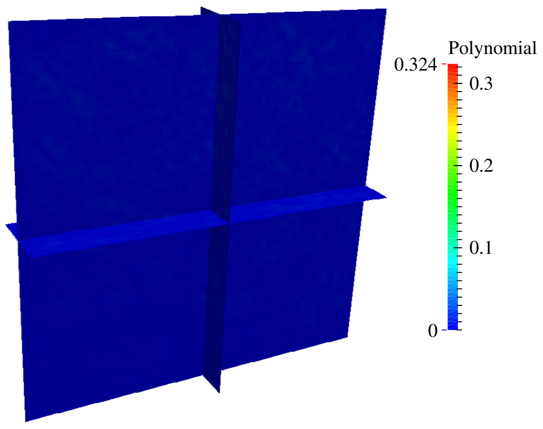


(b)

Fig. 9 Distributions of gradient errors of linear field $U(x, y, z) = x + 2y + 3z$ on perturbed prismatic mesh. **a** CAVG and **b** GSM



(a)



(b)

Fig. 10 Distributions of gradient errors of polynomial field $U(x, y, z) = (x + 2y + 3z)^2$ on perturbed prismatic mesh. **a** CAVG and **b** GSM

ous methods to approximate the convective fluxes, such as the central differential with dissipation, Roe's scheme, AUSM, etc. These methods are quite standard in FVM solvers, so we will not elaborate them here.

5.2 Approximation of viscous fluxes

During solving the Navier–Stokes equations shown as Eq. (27), the viscous stresses τ_{ij} at the midpoint of edges are needed for calculating the gradients of viscous fluxes using the GSM. According to Eq. (29), the spatial derivatives of velocity at the edge midpoint are needed. In traditional FVM–CFD solvers, the gradients at edge midpoint are approximated using simple interpolation method shown in Eq. (20) or corrected interpolation method in Eq. (21). In the GSM–CFD solver, the spatial gradients are calculated using GSM on the compact mGSD introduced in Sect. 3. We have proven that the proposed GSM can be more accurate than the interpolation method when approximating the gradients at midpoints on perturbed structured mesh which is mostly used to discretize the boundary-layer regions. The viscous fluxes at midpoint of edge $i - j_k$ can be directly calculated as

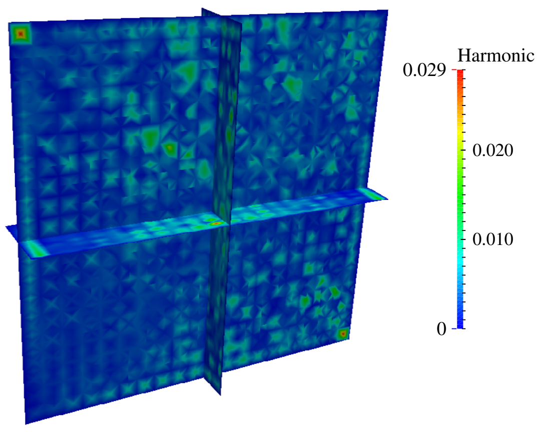
$$F_{vrs}^{ij_k} = \mu\left(\frac{\partial u_{m_k,r}}{\partial x_s} + \frac{\partial u_{m_k,s}}{\partial x_r}\right) \tag{32}$$

where r and s denote the axial directions in Cartesian coordinate system, and $\frac{\partial u_{m_k,r}}{\partial x_s}$ and $\frac{\partial u_{m_k,s}}{\partial x_r}$ are calculated using GSM.

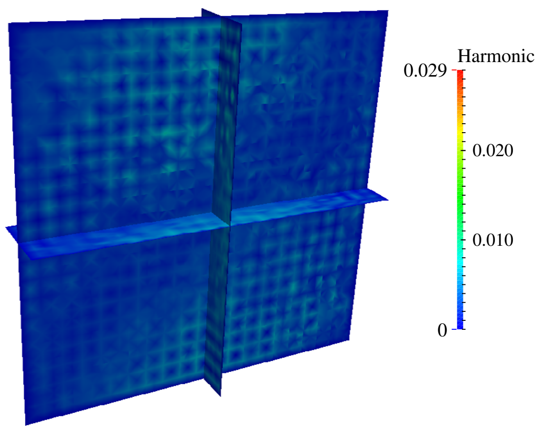
After the residuals are calculated based on the gradients of fluxes, the N–S equations can be solved explicitly or implicitly [4,34].

6 Application GSM–CFD to blood flow simulation of carotid bifurcation

In this section, the 3D unsteady blood flows through carotid bifurcation are solved. The geometry and mesh are shown

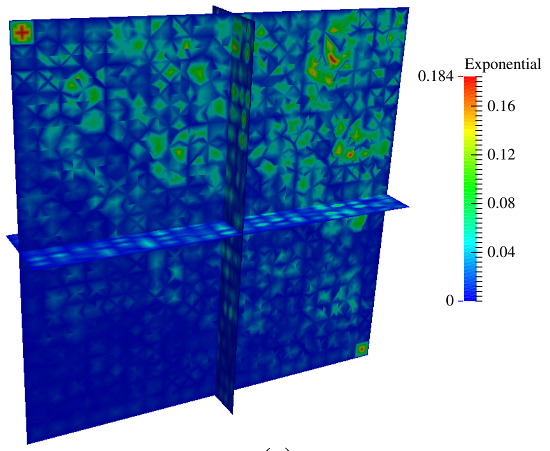


(a)

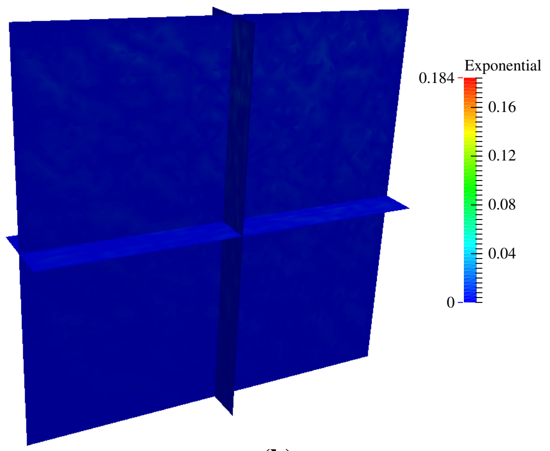


(b)

Fig. 11 Distributions of gradient errors of harmonic field $U(x, y, z) = z \sin 2\pi x \cos 2\pi y$ on perturbed hexahedral mesh. **a** CAVG and **b** GSM



(a)



(b)

Fig. 12 Distributions of gradient errors of exponential field $U(x, y, z) = e^{x+2y+3z}$ on perturbed prismatic mesh. **a** CAVG and **b** GSM

in Fig. 13[1], and the dimensions of the model are listed in Table 2. The geometry surface is discretized using triangular mesh, and several layers of prismatic mesh are generated to resolve the boundary layer. During the simulation, we apply velocity inlet boundary condition at common carotid artery (CCA), zero back pressure (free of constrains) outlet boundary condition at internal carotid artery (ICA) and external carotid artery (ECA), and non-slip wall boundary condition at the artery walls. Several layers of elements are extruded from the inlet and outlet surfaces to reduce the influences of the boundary conditions on the internal flow.

In this case, the blood flow through the bifurcation is only determined by inflow, geometry and friction. The blood is modeled as incompressible Newtonian flow with density $\rho = 1{,}041\ \mathrm{kg/m^3}$, and the dynamic viscosity $\mu = 2.9 \times 10^{-3}$ Pa s [12]. The unsteady flow rate in a cardiac cycle shown in Fig. 14 is applied at the CCA as inlet boundary condition.

[1] The model is obtained from http://grabcad.com/library/carotid-bifurcation

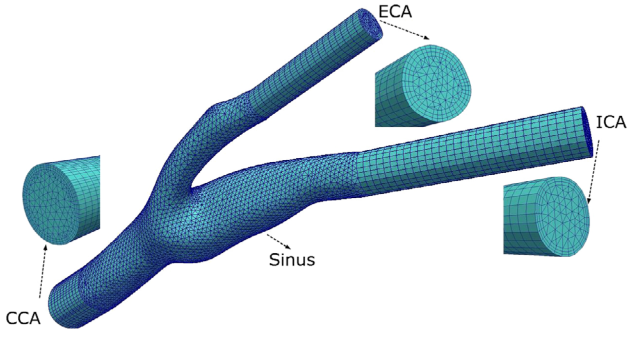


Fig. 13 Geometry and mesh of carotid bifurcation. There are 111,949 elements and 38,760 nodes in the mesh. In the figure, *CCA* denotes common carotid artery, *ICA* internal carotid artery and *ECA* external carotid artery

The fast Fourier transform is carried out for the discretized data to obtain the approximated flow rate at any time.

The streamlines at $t = 0s$, $t = 0.14s$ (near peak systole), $t = 0.52s$ (second peak systole) and $t = 0.9s$ (diastole) are shown in Fig. 15. In the figures, the streamlines are colored

Table 2 Geometry information of the carotid bifurcation

The extended parts at inlet and outlet are not included

A_{CCA}	R_{CCA}	A_{ICA}	R_{ICA}	A_{ECA}	R_{ECA}	L
29.82 mm^2	3.08 mm	14.82 mm^2	2.17 mm	7.22 mm^2	1.52 mm	35.8 mm

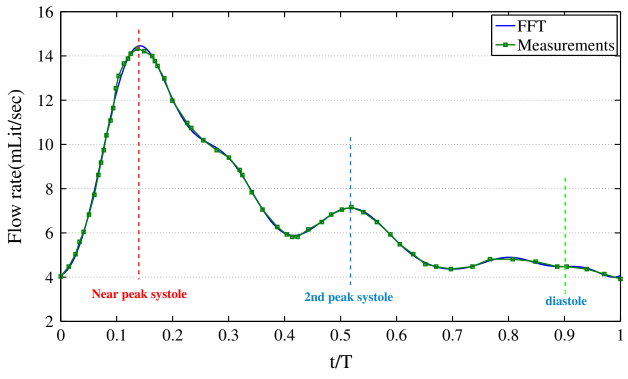


Fig. 14 The unsteady flow rates obtained from phase-contrast MRI velocity measurements [5] in vivo

with axial velocity. The velocity contours at different axial locations at $t = 0.14s$ are shown in Fig. 16. A steady-state solution with inlet flow rate 4.0 mL/s is used as the initial step for the unsteady simulation. At the beginning of the cardiac cycle, the blood has the lowest speed, and smallest amount of flow rate at the ECA. With the rapid increase of flow rate (actually the inlet blood pressure), the blood flow speed also increases rapidly. After the peak systole, the blood velocity decreases gradually. We can also observe that, under the given boundary conditions, much more blood will flow through the ICA during the cardiac cycle. For example, the the peak velocity at ECA is about 0.262 m/s, and ICA 0.411 m/s, and the peak velocity ratio $V_{ECA}/V_{ICA} \approx 0.64$ (which usually ranges from 0.4 to 0.7 [3]).

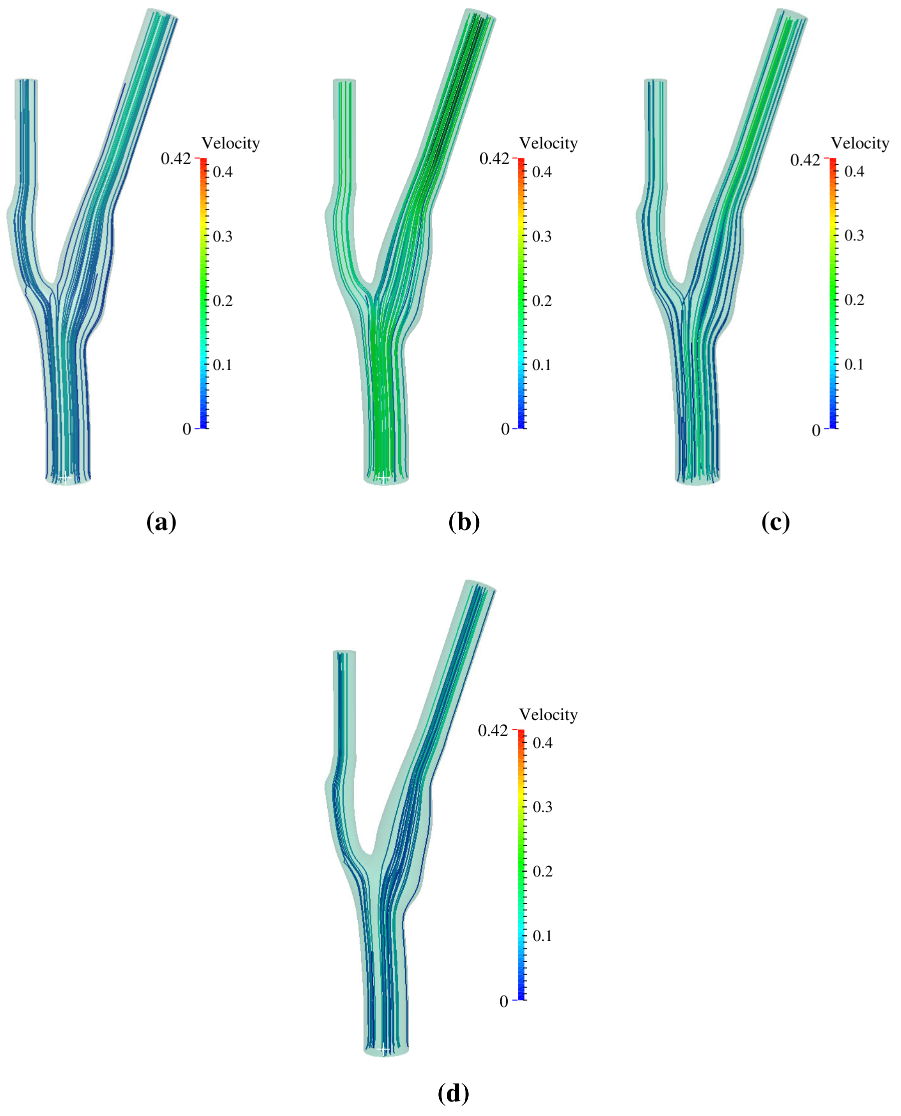


Fig. 15 Streamlines at different times during the cardiac cycle of the carotid bifurcation. **a** $t = 0s$, **b** $t = 0.14s$, **c** $t = 0.52s$ and **d** $t = 0.9s$

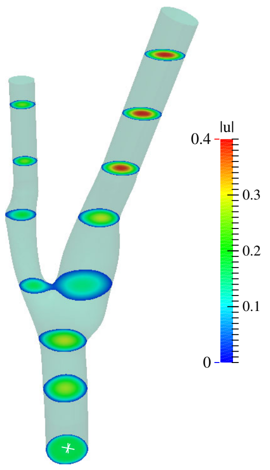


Fig. 16 Velocity contours at different axial locations of the carotid bifurcation near peak systole

7 Conclusions

The main purpose of this paper is to develop a 3D GSM–CFD solver for incompressible flow. The governing equations of incompressible flows are solved in a strong form but all the spatial gradients are approximated in a weak form based on the GSM. A matrix-form algorithm is devised to calculate the spatial derivatives of field variables on the compact smoothing domains. The proposed GSM–CFD solver is used to solve the blood flows through the carotid bifurcation. According to the results of gradient approximation and flow simulation, the following conclusions can be drawn:

1. On the perturbed structured meshes, much more accurate gradients results at midpoint of cell edges can be obtained using GSM than using average gradient method with directional correction. Thus, we can conclude that the GSM is more robust than the modified interpolation method for gradient calculation.
2. Even on perturbed hexahedral and prismatic meshes, the GSM can still exactly reproduce the linear field. This is the most distinguished feature of the first-order conservative GSM method, and would play an important role in solving problems with complicated geometries and in the fluid structure interaction simulations.
3. The GSM is also used to calculate gradients of the convective and viscous fluxes in incompressible governing equations to form a GSM–CFD solver with artificial compressibility approach. The carotid bifurcation example shows the effectiveness of the proposed solver.

As the first step for the blood flow simulation using GSM–CFD solver, two important assumptions are made: the blood is incompressible Newtonian flow and the vessel is treated as solid wall. In further development, we will add the non-Newtonian properties and the interaction between flexible and deformable vessel to the proposed GSM–CFD solver.

Acknowledgments This research is supported mainly by ARO under Contract No. W911NF-12-1-0147 and partially by NSF under Award No. 1214188

References

1. Aidun CK, Clausen JR (2010) Lattice–Boltzmann method for complex flows. Annu Rev Fluid Mech 42:439–472
2. Antiga L, Piccinelli M, Botti L, Ene-Iordache B, Remuzzi A, Steinman DA (2008) An image-based modeling framework for patient-specific computational hemodynamics. Med Biol Eng Comput 46(11):1097–1112
3. Blackshear WM, Phillips DJ, Chikos PM, Harley JD, Thiele BL, Strandness DE Jr (1980) Carotid artery velocity patterns in normal and stenotic vessels. Stroke 11:67–71
4. Blazek J (2005) Computational fluid dynamics: principles and applications. Elsevier, Amsterdam
5. Cebral JR, Yim PJ, Lohner R, Soto O, Choyke PL (2002) Blood flow modeling in carotid arteries with computational fluid dynamics and MR imaging. Acad Radiol 9(11):1286–1299
6. Cebral JR, Castro MA, Appanaboyina S, Putman CM, Millan D, Frangi AF (2005) Efficient pipeline for image-based patient-specific analysis of cerebral aneurysm hemodynamics: technique and sensitivity. IEEE Trans Med Imaging 24(4):457–467
7. Cebral JR, Castro MA, Burgess JE, Pergolizzi RS, Sheridan MJ, Putman CM (2005) Characterization of cerebral aneurysms for assessing risk of rupture by using patient-specific computational hemodynamics models. AJNR Am J Neuroradiol 26:2550–2559
8. Chen S, Doolen GD (1998) Lattice Boltzmann method for fluid flows. Annu Rev Fluid Mech 30:329–364
9. Chorin AJ (1967) A numerical method for solving incompressible viscous flow problems. J Comput Phys 2:12–26
10. Crumpton PI, Moinier P, Giles MB (1997) An unstructured algorithm for high Reynolds number flows on highly stretched grids. In: 10th International conference on numerical methods in laminar and turbulent flow, Montreal
11. Filipovic N, Teng Z, Radovic M et al (2013) Computer simulation of three-dimensional plaque formation and progression in the carotid artery. Med Biol Eng Comput 51(6):607–616
12. Gijsen FJH, van de Vosse FN, Janssen JD (1999) The influence of the non-Newtonian properties of blood on the flow in large arteries: steady flow in a carotid bifurcation model. J Biomech 32:601–608
13. Ho H, Wu J, Hunter P (2011) Blood flow simulation in a giant intracranial aneurysm and its validation by digital subtraction angiography. In: Computational biomechanics for medicine. Springer, Berlin, pp 15–26
14. Janela J, Moura A, Sequeira A (2010) A 3D non-Newtonian fluid–structure interaction model for blood flow in arteries. J Comput Appl Math 234(9):2783–2791
15. Johnston BM, Johnston PR, Corney S, Kilpatrick D (2003) Non-Newtonian blood flow in human right coronary arteries: steady state simulations. J Biomech 37:709–720
16. Kabinejadiana F, Ghistab DN (2012) Compliant model of a coupled sequential conoray arterial bypass graft: effects of vessel wall elasticity and non-newtonian rheology on blood flow regime and hemodynamic parameters distribution. Med Eng Phys 34:860–872
17. Kanaris AG, Anastasiou AD, Paras SV (2012) Modeling the effect of blood viscosity on hemodynamic factors in a small bifurcated artery. Chem Eng Sci 71:202–211
18. Ku DN (1997) Blood flow in arteries. Annu Rev Fluid Mech 29:399–434
19. Li E, Tan V, Xu GX, Liu GR, He ZC (2011) A novel linearly-weighted gradient smoothing method (LWGSM) in the simulation of fluid dynamics problem. Comput Fluids 50:104–119

20. Li E, Liu GR, Xu GX, Tan V, He ZC (2012) Numerical modeling and simulation of pulsatile blood flow in rigid vessel using gradient smoothing method. Eng Anal Bound Elem 36(3):322–334
21. Liu GR (2008) A generalized gradient smoothing technique and the smoothed bilinear form for Galerkin formulation of a wide class of computational methods. Int J Comput Methods 5(2):199–236
22. Liu GR (2009) Mesh free methods: moving beyond the finite element method. CRC Press, Boca Raton
23. Liu GR (2010) Smoothed finite element methods. CRC Press, Boca Raton
24. Liu MB, Liu GR (2010) Smoothed particle hydrodynamics (SPH): an overview and recent developments. Arch Comput Methods Eng 17:25–76
25. Liu GR, Liu MB (2003) Smoothed particle hydrodynamic: a meshfree particle method. World Scientific, Singapore
26. Liu GR, Dai KY, Nguyen TT (2007) A smoothed finite element method for mechanics problems. Comput Mech 39(6):859–877
27. Liu GR, Nguyen TT, Dai KY, Lam KY (2007) Theoretical aspects of the smoothed finite element method (SFEM). Int J Numer Methods Eng 71(8):902–930
28. Liu GR, Xu GX (2008) A gradient smoothing method (GSM) for fluid dynamics problems. Int J Numer Methods Fluids 58(10):1101–1133
29. Liu GR, Zhang J, Lam KY, Li H, Xu GX, Zhong ZH, Li GY, Han X (2008) A gradient smoothing method (GSM) with directional correction for solid mechanics problems. Comput Mech 41(3):457–472
30. Lucy LB (1977) A numerical approach to the testing of the fission hypothesis. Astron J 82(12):1013–1024
31. Moireau P, Xiao N, Astorino M et al (2012) External tissue support and fluid–structure simulation in blood flows. Biomech Model Mechanobiol 11:1–18
32. Monaghan JJ (1992) Smoothed particle hydrodynamics. Annu Rev Astron Astrophys 30:543–574
33. Morbiducci U, Gallo D, Massai D, Ponzini R, Deriu MA, Antiga L, Redaelli A, Montevecchi FM (2011) On the importance of blood rheology for bulk flow in hemodynamic models of the carotid bifurcation. J Biomech 44(13):2427–2438
34. Palacios F, Colonno MR, Aranake AC, Campos A, Copeland SR, Economon TD, Lonkar AK, Lukaczyk TW, Taylor TWR, Alonso JJ (2013) Stanford university unstructured (SU2): an open-source interated computational environment for multi-physics simulation and design. In: 51st AIAA aerospace sciences meeting, Grapevine
35. Razzaq M, Damanik H, Hronand J, Ouazzi A, Turek S (2012) FEM multigrid techniques for fluid–structure interaction with application to hemodynamics. Appl Numer Math 62(9):1156–1170
36. Soulis JV, Giannoglou GD, Chatzizisis YS, Louridas GE, Seralidou KV, Parcharidis GE (2008) Non-Newtonian models for molecular viscosity and wall shear stress in a 3D reconstructed human left coronary artery. Med Eng Phys 30:9–19
37. Takizawa K, Tezduyar TE (2014) Fluid–structure interaction modeling of patient-specific cerebral aneurysms. In: Visualization and simulation of complex flows in biomedical engineering. Springer, Delft, pp 25–45
38. Takizawa K, Christopher J, Tezduyar TE, Sathe S (2010) Space–time finite element computation of arterial fluid–structure interactions with patient-specific data. Int J Numer Methods Biomed Eng 26(1):101–116
39. Takizawa K, Moorman C, Wright S, Christopher J, Tezduyar TE (2010) Wall shear stress calculations in spacetime finite element computation of arterial fluid–structure interactions. Comput Mech 46:31–41
40. Takizawa K, Moorman C, Wright S, Purdue J, McPhail T, Chen PR, Warren J, Tezduyar TE (2011) Patient-specific arterial fluid–structure interaction modeling of cerebral aneurysms. Int J Numer Methods Fluids 65(1–3):308–323
41. Takizawa K, Schjodt K, Puntel A, Kostov N, Tezduyar TE (2013) Patient-specific computational analysis of the influence of a stent on the unsteady flow in cerebral aneurysms. Comput Mech 51(6):1061–1073
42. Taylor CA, Figueroa CA (2009) Patient-specific modeling of cardiovascular mechanics. Annu Rev Biomed Eng 11(1):109–134
43. Tezduyar TE, Sathe S (2007) Modeling of fluid–structure interactions with the space–time finite elements: solution techniques. Int J Numer Methods Fluids 54:855–900
44. Tezduyar TE, Sathe S, Cragin T, Nanna B, Conklin BS, Pausewang J, Schwaab M (2007) Modeling of fluid–structure interactions with the space–time finite elements: arterial fluid mechanics. Int J Numer Methods Fluids 54:901–922
45. Tezduyar TE, Sathe S, Schwaab M, Conklin BS (2008) Arterial fluid mechanics modeling with the stabilized space–time fluid–structure interaction technique. Int J Numer Methods Fluids 57:601–629
46. Tezduyar TE, Schwaab M, Sathe S (2009) Sequentially-coupled arterial fluid–structure interaction (SCAFSI) technique. Comput Methods Appl Mech Eng 198:3524–3533
47. Tezduyar TE, Takizawa K, Moorman C, Wright S, Christopher J (2010) Multiscale sequentially coupled arterial FSI technique. Comput Mech 46:17–29
48. Torii R, Oshima M, Kobayashi T, Takagi K, Tezduyar TE (2008) Fluid–structure interaction modeling of a patient-specific cerebral aneurysm: influence of structural modeling. Comput Mech 43:151–159
49. Weiss JM, Maruszewski JP, Smith WA (1999) Implicit solution of preconditioned Navier–Stokes equations using algebraic multigrid. AIAA J 37(1):29–36
50. Wu Y, Cai X-C (2014) A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation. J Comput Phys 258:524–537
51. Xu GX, Liu GR, Tani A (2009) An adaptive gradient smoothing method (GSM) for fluid dynamics problems. Int J Numer Method Fluids 62:499–529
52. Xu GX, Li E, Tan V, Liu GR (2012) Simulation of steady and unsteady incompressible flow using gradient smoothing method (GSM). Comput Struct 90–91:131–144
53. Yao J, Liu GR, Narmoneva DA, Hinton RB, Zhang Z-Q (2012) Immersed smoothed finite element method for fluid–structure interaction simulation of aortic valves. Comput Mech 50(6):789–804
54. Yao J, Liu GR, Chen C-L (2013) A moving-mesh gradient smoothing method for compressible cfd problems. Math Models Methods Appl Sci 23(2):273–305
55. Zhang J, Liu GR, Lam KY, Li H, Xu GX (2008) A gradient smoothing method (GSM) based on strong form governing equation for adaptive analysis of solid mechanics problems. Finite Elem Anal Des 44(15):889–909