Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 225 (2007) 120–144
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A matrix-free implicit unstructured multigrid finite volume method for simulating structural dynamics and fluid–structure interaction

X. Lv, Y. Zhao *, X.Y. Huang, G.H. Xia, X.H. Su

School of Mechanical and Aerospace Engineering, College of Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Received 29 May 2006; received in revised form 17 October 2006; accepted 15 November 2006
Available online 17 January 2007

Abstract

A new three-dimensional (3D) matrix-free implicit unstructured multigrid finite volume (FV) solver for structural dynamics is presented in this paper. The solver is first validated using classical 2D and 3D cantilever problems. It is shown that very accurate predictions of the fundamental natural frequencies of the problems can be obtained by the solver with fast convergence rates. This method has been integrated into our existing FV compressible solver [X. Lv, Y. Zhao, et al., An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3d unsteady compressible flows with moving objects, Journal of Computational Physics 215(2) (2006) 661–690] based on the immersed membrane method (IMM) [X. Lv, Y. Zhao, et al., as mentioned above]. Results for the interaction between the fluid and an immersed fixed-free cantilever are also presented to demonstrate the potential of this integrated fluid–structure interaction approach.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Vertex-based finite volume method; Fluid–structure interaction; Unstructured multigrid; Dual time stepping; Computational solid mechanics

1. Introduction

Over the last five decades a wide variety of numerical methods have been proposed for the numerical solution of partial differential equations. Among them the finite element (FE) method has firmly established itself as the standard approach for problems in computational solid mechanics (CSM), especially with regard to deformation problems involving non-linear material analysis [1,2]. As a contemporary, the FV method developed from early finite difference techniques and has similarly established itself within the field of computational fluid dynamics (CFD) [3,4]. Both classes of methods integrate governing equations over pre-defined control volumes [3,5], which are associated with the elements making up the domain of interest. Furthermore,

* Corresponding author. Tel.: +65 6790 4545; fax: +65 792 4062.
E-mail address: myzhao@ntu.edu.sg (Y. Zhao).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.11.023

both approaches can be classified as weighted residual methods where they differ in the weighting functions adopted [6].

In many engineering applications, there is an emerging need to model multiphysics problems in a coupled manner. In principle, because of their local conservation properties the FV methods should be in a good position to solve such problems effectively. Over the last decade a number of researchers have applied FV methods to problems in CSM [7] and it is now possible to classify these methods into two approaches, cell-centered [8–12] and vertex-based ones [6,7,13,14]. The first approach is based upon traditional FV methods [3] that have been widely applied in the context of CFD [4]. Subsequently, such techniques have been applied to CSM problems using structured [8,9] and unstructured meshes [10–12,15]. With regard to these techniques, it should be noted that when solid bodies undergo deformation the application of mechanical boundary conditions is the most effective if they can be imposed directly on the physical boundary. Obviously, the cell-centred approximation may have difficulty in prescribing the boundary conditions, when complex geometries are considered and where displacements at the boundary are not prescribed directly and in a straight forward manner.

The second approach is based on some basic ideas of traditional FE methods, which employs shape functions to describe the variation of an independent variable, such as displacement, over an element and is therefore well suited to complex geometries [6,7,13,14]. The approach can be roughly classified as a cell-vertex FV method [4,6].

Both the above FV approaches apply strict conservation laws over a control volume and are comparable, if not better in accuracy and efficiency, to the traditional FE methods [7,10]. Some researchers have attributed this to the local conservation of independent variables as enforced by the control-volume methods employed [13,14] and others to the enforced continuity of the derivatives of the independent variables across cell boundaries [10].

The objectives of this paper are to describe the development and validation of a new vertex-based unstructured multigrid FV method for structural dynamic problems. It should be noted that the approach presented in this paper belongs to a special class of cell-vertex methods that employ non-overlapping control volumes [7,16–18]. It should also be noted that the approach is different from previous non-overlapping FV methods [6,7,13,14] in that we do not utilize shape functions at all and we have developed an effective implicit unstructured multigrid method for fast solution convergence.

The paper is structured as follows. In Section 2, the mathematical formulation and detailed numerical treatments are described. In Section 3, a brief description of the implemented convergence acceleration techniques is presented. This is followed by a short description of the coupling with the fluid solver TETRAKE in Section 4. In Section 5, the method is validated by applying it to several test cases. Finally, several concluding remarks are given in Section 6.

2. Governing equations and numerical methods

2.1. Governing equations

The governing equations for structural dynamics based on the continuum model are the Cauchy's equations:

$$\begin{aligned} b_x + \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} &= \rho \frac{\partial^2 d_x}{\partial t^2}, \\ b_y + \frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} &= \rho \frac{\partial^2 d_y}{\partial t^2}, \\ b_z + \frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} &= \rho \frac{\partial^2 d_z}{\partial t^2}, \end{aligned} \tag{2.1}$$

where σ_{ij} is the stress tensor defined for either a fluid or solid medium, with the correlation $\sigma_{ij} = \sigma_{ji}$. b_x, b_y and b_z are the body force components in the three directions of Cartesian coordinates. ρ is the material density. d_x, d_y and d_z are the three components of displacement vector. This system of equations can be expressed in a more compact form:

$$\vec{b} + \nabla \cdot \sigma_{ij} = \rho \vec{a}, \tag{2.2}$$

where $\vec{a}$ is the acceleration vector. In the theory of elasticity, Eq. (2.2) is variously described as the stress equation of small motion [21], the equation of equilibrium [22] or the equation of motion [23]. The term equation of dynamic equilibrium will be employed in this paper to distinguish the dynamic problems considered in this research from static structural problems. Damping is the ability of a structure to dissipate energy and in structural mechanics the most common damping device is the ideal linear viscous damper [24–26]. The ideal linear viscous damper opposes structural motion with a force proportional to velocity. Thus, for those cases where damping is required, the linear viscous damping term is incorporated into Eq. (2.2) as follows:

$$\vec{b} + \nabla \cdot \sigma_{ij} = \rho \vec{a} + c\vec{U}, \tag{2.3}$$

where $\vec{U}$ is velocity vector and c is the coefficient of viscous damping.

2.2. Constitutive relationship for stress and strain

The generalized form of Hooke's law gives the following constitutive relationship between stress and strain for an isotropic homogeneous material undergoing small strains [27] in three dimensions:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{zx} \end{pmatrix} = D \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{pmatrix} = \left[\frac{E(1-v)}{(1+v)(1-2v)} \begin{pmatrix} 1 & \frac{v}{1-v} & \frac{v}{1-v} & 0 & 0 & 0 \\ \frac{v}{1-v} & 1 & \frac{v}{1-v} & 0 & 0 & 0 \\ \frac{v}{1-v} & \frac{v}{1-v} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2v}{2(1-v)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2v}{2(1-v)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2v}{2(1-v)} \end{pmatrix} \right] \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{pmatrix}, \tag{2.4}$$

where E and v are, respectively, Young's modulus and Poisson's ratio, the stress vector is $\sigma^{\mathrm{T}} = [\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma_{zx}]$ and the strain vector $\varepsilon^{\mathrm{T}} = [\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{xy}, \varepsilon_{yz}, \varepsilon_{zx}]$. The elastic strain at any instant in time may be expressed in terms of the total and initial strains. Thus the constitutive relationship for an isotropic homogeneous material undergoing linear elastic strains is given by

$$\sigma - D(\varepsilon - \varepsilon^0) = 0 \quad \text{in } \Omega_{\mathrm{s}}, \tag{2.5}$$

where D is the constitutive property matrix, given by the term in the large square brackets in Eq. (2.4) and ε and ε^0 are the total and initial strains, respectively. Ω_{s} is the structural domain.

2.3. Displacement formulation

This work is based on a linear strain–displacement formulation using the small strain assumption, which is valid for strains of the order of a few percent [21]. Thus the strains may be defined in the general displacement form as

$$\varepsilon = \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{pmatrix} = L\vec{d} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial x} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{pmatrix} \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} \frac{\partial d_x}{\partial x} \\ \frac{\partial d_y}{\partial y} \\ \frac{\partial d_z}{\partial z} \\ \frac{\partial d_x}{\partial y} + \frac{\partial d_y}{\partial x} \\ \frac{\partial d_y}{\partial z} + \frac{\partial d_z}{\partial y} \\ \frac{\partial d_x}{\partial z} + \frac{\partial d_z}{\partial x} \end{pmatrix}, \tag{2.6}$$

where $\vec{d}$ is the vector of displacements and L is the matrix of differential operators. Applying the constitutive stress–strain equation (2.4) and the strain–displacement equation (2.6) to the dynamic equilibrium equatuion (2.3) yields the displacement formulation:

$$b + \nabla \cdot (DL\vec{d} - D\varepsilon^0) - \rho\vec{a} - c\vec{U} = 0 \quad \text{in } \Omega_s, \tag{2.7}$$

where Ω_s represents the structural domain. Eq. (2.7) is subject to the boundary conditions:

$$\begin{aligned} &\vec{d} - \vec{d}_P = 0 \quad \text{for } \Gamma_d, \\ &T(DL\vec{d} - D\varepsilon^0) - \vec{t}_P = 0 \quad \text{for } \Gamma_t, \end{aligned} \tag{2.8}$$

where the structural boundary is a combination of prescribed displacement and traction boundaries, i.e. $\Gamma_s = \Gamma_d \cup \Gamma_t$ and T is the matrix of outward normal operators such that

$$T = \begin{pmatrix} n_x & 0 & 0 & n_y & 0 & n_z \\ 0 & n_y & 0 & n_x & n_z & 0 \\ 0 & 0 & n_z & 0 & n_y & n_x \end{pmatrix}, \tag{2.9}$$

where n is the outward unit normal vector to the domain boundary with components n_x, n_y and n_z.

2.4. Discretization of the displacement equations

Eq. (2.7) is discretized on an unstructured tetrahedral grid and a cell-vertex scheme is adopted here, i.e., all computed variables in vector $\vec{d}$ are stored at vertices of the tetrahedral cells. For every vertex, as shown in Fig. 1, a control volume is constructed using the median dual of the tetrahedral grid [19]. In Fig. 1, nodes A, P, B and C form the vertices of the tetrahedral cell and O is the centre of the element $APBC$. Points a, b and c are the centres of the edges AP, BP and CP. 1, 2 and 3 are the centroids of triangles APC, CBP and ABP, respectively. In the cell-vertex scheme, the computed variables are stored at vertices A, P, B and C. Triangles $O1a$, $O3a$, $O3b$, $O2b$, $O1c$ and $O2c$ form part of the control volume surface for node P within the tetrahedral cell. Likewise, the control volume surfaces for different nodes A, B and C are constructed based on this idea.

The finite-volume discretization is based on the governing equations in integral form, on which spatial discretization is performed over the control volume surrounding a node (P for example):

$$\int\int\int_{CV} (\nabla \cdot (DL\vec{d} - D\varepsilon^0))\,\mathrm{d}V + \int\int\int_{CV} (b - c\vec{U})\,\mathrm{d}V - \int\int\int_{CV} \rho\vec{a}\,\mathrm{d}V = 0. \tag{2.10}$$

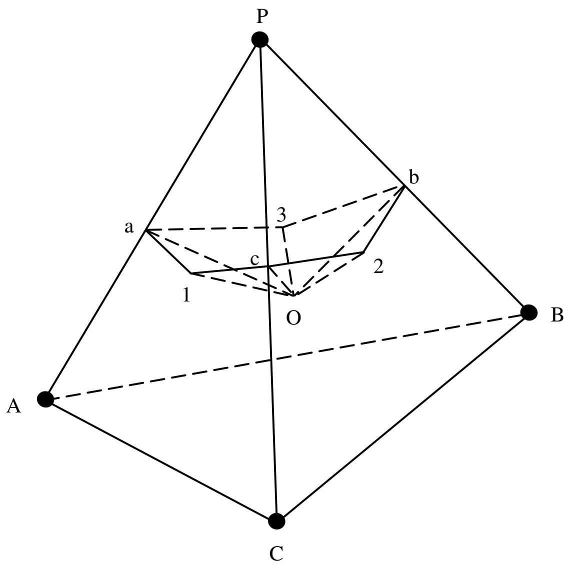


Fig. 1. Portion of control volume around vertex P within tetrahedron ABCD.

The first term on the left-hand side is calculated using a cell-based method:

$$\int\int\int_{CV}(\nabla\cdot(DL\vec{d}-D\varepsilon^0))\,\mathrm{d}V=\oiint_{S_{CV}}(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\,\mathrm{d}S=\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_c]_i, \tag{2.11}$$

where *ncell* is the number of cells associated with node P and ΔS_{ci} is the part of control volume surface in cell i. By using the following relation for a given cell

$$\oiint_{S_{CV}}\mathrm{d}\vec{S}=0,$$

the total vector surface of the control volume in a cell i becomes

$$\vec{n}\Delta S_{ci}=\frac{1}{3}(\vec{n}\Delta S_{pi}),$$

where $\vec{n}\Delta S_{pi}$ is the surface vector of the face opposite node P of the tetrahedron under consideration. Thus, the calculation of the first term on the left-hand side of Eq. (2.10) can be simplified as

$$\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_c]_i=\frac{1}{3}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i. \tag{2.12}$$

Here the value $(DL\vec{d}-D\varepsilon^0)_i$ is calculated at the center of the tetrahedron with a node P, and can be obtained by using Green's theorem based on the variables at the four vertices of the tetrahedron. Similar to the Galerkin type of formulation, the gradient of a variable ϕ at the center of a tetrahedron is evaluated as follows:

$$\mathrm{grad}\,\phi_c=-\frac{\sum_{i=1}^4\phi_i 9S_i}{27V}=-\frac{1}{3}\frac{\sum_{i=1}^4\phi_i S_i}{V}, \tag{2.13}$$

where ϕ_i is the variable at a vertex i of the tetrahedron and S_i is the surface area that is opposite to node i, V is the volume of the tetrahedron. Gradients at the vertices are obtained by volume averaging of the gradients at the centers of cells associated with the vertex under consideration.

If we use the spatial averages of density, acceleration and velocity, then Eq. (2.10) can be written as

$$\frac{1}{3}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i+(b-c\vec{U})\cdot V-\rho\vec{a}\cdot V=0, \tag{2.14}$$

where V is the volume of the tetrahedron. Knowing that $\vec{a}=\frac{\partial\vec{U}}{\partial t}$, after arrangement we have

$$\frac{1}{3\rho V}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i+\frac{(b-c\vec{U})}{\rho}=\frac{\partial\vec{U}}{\partial t}. \tag{2.15}$$

An implicit scheme is adopted for Eq. (2.15) and the time dependent term is discretized using an implicit three-level asymmetric scheme. In order to obtain time-accurate solutions, we add a pseudo time term to this equation:

$$\frac{1}{3\rho V}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i+\frac{(b-c\vec{U}^{n+1})}{\rho}=\left(\frac{1.5\vec{U}^{n+1}-2.0\vec{U}^n+0.5\vec{U}^{n-1}}{\Delta t}\right)+\frac{\partial\vec{U}}{\partial\tau}, \tag{2.16}$$

where the superscript $(n+1)$ denotes the physical time level and all the variables are evaluated at this time level. The derivative with respect to a pseudo time τ is discretized using a first-order-accurate forward differencing scheme. After moving the rest of the terms to another side of the equation, we have

$$\begin{aligned}\frac{\vec{U}^{n+1,m+1}-\vec{U}^{n+1,m}}{\Delta\tau}&=\widetilde{R}^{n+1,m}\\&=\frac{1}{3\rho V}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i+\frac{(b-c\vec{U}^{n+1,m})}{\rho}-\left(\frac{1.5\vec{U}^{n+1,m}-2.0\vec{U}^n+0.5\vec{U}^{n-1}}{\Delta t}\right).\end{aligned} \tag{2.17}$$

The solution of the velocity vector is sought by marching the solution to a pseudo steady state in τ. Here m and $m+1$ denote the initial and next pseudo time levels. Once the artificial steady state is reached, the derivative of $\vec{U}$ with respect to τ becomes zero, and the solution satisfies $\widetilde{R}^{n+1,m}=0$. Hence, the original equation (2.15) is fully recovered. Therefore, instead of solving the equations in each time step in the physical time domain (t), the problem is transformed into a sequence of steady-state computations in the artificial time domain (τ). Eq. (2.17) can be integrated in pseudo time by an explicit five-stage Runge–Kutta scheme. However, the pseudo time step size may be restricted if the physical time step size is very small. Hence, a fully implicit dual time stepping scheme is adopted here. A Taylor series expansion is performed for the residual in Eq. (2.17) with respect to the pseudo time for node P:

$$\widetilde{R}_p^{n+1,m+1}=\widetilde{R}_p^{n+1,m}+\frac{\partial\widetilde{R}_p}{\partial(\vec{U})_p}\Delta(\vec{U})_p+\sum_{j=1}^{nbseg}\frac{\partial\widetilde{R}_p}{\partial(\vec{U})_j}\Delta(\vec{U})_j, \tag{2.18}$$

where *nbseg* is the number of edges connected to P, which is also equal to the number of neighbouring points connected to point P through the edges. And for the viscous damping and physical time-dependent terms, we have

$$(\vec{U})_p^{n+1,m+1}=(\vec{U})_p^{n+1,m}+\Delta(\vec{U})_p. \tag{2.19}$$

After combining all the residual terms at every node in the structural domain into a vector and dropping the third term of the right-hand side of Eq. (2.18), we have

$$\widetilde{R}^{n+1,m+1}=\widetilde{R}^{n+1,m}+A\Delta(\vec{U})_p-\frac{1.5}{\Delta t}\Delta(\vec{U})_p-\frac{c}{\rho}\Delta(\vec{U})_p, \tag{2.20}$$

where

$$A=\frac{1}{3M_p}\left(\frac{\partial\left\{\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i\right\}}{\partial(\vec{U})_p}\right)=\frac{1}{3M_p}\left(\frac{\partial\left\{\sum_{i=1}^{ncell}[(DL\vec{d})\cdot\vec{n}\Delta S_p]_i\right\}}{\partial(\vec{U})_p}\right).$$

And the whole-field equivalent equation (2.17) can then be re-written as

$$\begin{aligned}\left(\frac{\Delta t+1.5\Delta\tau}{\Delta t}+\frac{c\Delta\tau}{\rho}-A\Delta\tau\right)\frac{\Delta(\vec{U})_p}{\Delta\tau}&=\widetilde{R}^{n+1,m}\\&=\frac{1}{3M_p}\sum_{i=1}^{ncell}[(DL\vec{d}-D\varepsilon^0)\cdot\vec{n}\Delta S_p]_i+\frac{(b-c\vec{U}^{n+1,m})}{\rho}\\&\quad-\left(\frac{1.5\vec{U}^{n+1,m}-2.0\vec{U}^n+0.5\vec{U}^{n-1}}{\Delta t}\right),\end{aligned} \tag{2.21}$$

where $\Delta(\vec{U})_p=\vec{U}^{n+1,m+1}-\vec{U}^{n+1,m}$ and $M_p=\rho V$ is the mass of the control volume around the current node P. We need to evaluate the derivative $\frac{\partial(\partial\vec{d}/\partial X_i)}{\partial\vec{U}}$ $(X_i=x,y,z)$ in order to determine the Jacobian A. It can be calculated as follows:

$$\frac{\partial(\partial\vec{d}/\partial X_i)}{\partial\vec{U}}=\frac{\partial(\partial\vec{d}/\partial X_i)/\partial X_i}{\partial\vec{U}/\partial X_i}=\frac{\partial^2\vec{d}/\partial X_i^2}{\partial\vec{U}/\partial X_i}, \tag{2.22}$$

where $\vec{d}$ is the displacement vector at node P. That is

$$\widetilde{A}\frac{\Delta(\vec{U})}{\Delta\tau}=\widetilde{R}^{n+1,m}; \tag{2.23}$$

thus,

$$\frac{\Delta(\vec{U})}{\Delta\tau}=\widetilde{\widetilde{R}}^{n+1,m}, \tag{2.24}$$

where $\widetilde{\widetilde{R}}^{n+1,m}=\widetilde{A}^{-1}\widetilde{R}^{n+1,m}$ and $\widetilde{A}=\frac{\Delta t+1.5\Delta\tau}{\Delta t}+\frac{c\Delta\tau}{\rho}-A\Delta\tau$.

Further approximation can be introduced in order to achieve matrix-free computation. If we employ point implicit treatment to the preceding equations, then only the diagonal term in $\widetilde{A}$ is used in the pseudo time stepping. As a result, the equation for every node can now be written as

$$\frac{\Delta(\vec{U})}{\Delta\tau} = \widetilde{\widetilde{R}}^{n+1,m}, \tag{2.25}$$

where $\widetilde{\widetilde{R}}_p^{n+1,m} = \widetilde{A}_{pp}^{-1}\widetilde{R}_p^{n+1,m}$ and $\widetilde{A}_{pp}^{-1} = \mathrm{diag}[(\frac{\Delta t+1.5\Delta\tau}{\Delta t} + \frac{c\Delta\tau}{\rho} - A\Delta\tau)^{-1}]$.

Pseudo time stepping is then performed in Eq. (2.24). In this work, a five-stage Runge–Kutta time integration algorithm is used to march the numerical solution in pseudo time τ until convergence is reached [19]. Therefore, the converged solution from the pseudo time steady-state equations becomes the time accurate solution at current physical time. To advance the solution in pseudo time from m to $m+1$, the formulation of a five-stage Runge–Kutta scheme is as follows:

$$\begin{aligned}
(\vec{U})_p^{(0)} &= (\vec{U})_p^{m},\\
(\vec{U})_p^{(1)} &= (\vec{U})_p^{(0)} - \alpha_1\Delta\tau\widetilde{\widetilde{R}}[(\vec{U})_p^{(0)}],\\
(\vec{U})_p^{(2)} &= (\vec{U})_p^{(0)} - \alpha_2\Delta\tau\widetilde{\widetilde{R}}[(\vec{U})_p^{(1)}],\\
(\vec{U})_p^{(3)} &= (\vec{U})_p^{(0)} - \alpha_3\Delta\tau\widetilde{\widetilde{R}}[(\vec{U})_p^{(2)}],\\
(\vec{U})_p^{(4)} &= (\vec{U})_p^{(0)} - \alpha_4\Delta\tau\widetilde{\widetilde{R}}[(\vec{U})_p^{(3)}],\\
(\vec{U})_p^{(5)} &= (\vec{U})_p^{(0)} - \alpha_5\Delta\tau\widetilde{\widetilde{R}}[(\vec{U})_p^{(4)}],\\
(\vec{U})_p^{(m+1)} &= (\vec{U})_p^{(5)},
\end{aligned} \tag{2.26}$$

where the stage coefficients for a five-stage Runge–Kutta time integration is as follows:

$$\alpha_1 = 1/4, \quad \alpha_2 = 1/6, \quad \alpha_3 = 3/8, \quad \alpha_4 = 1/2, \quad \alpha_5 = 1.$$

After the velocity vector at physical time levels $n+1$ has been solved, we calculate the delta displacement during this time step as

$$\Delta\vec{d} = \frac{(\vec{U}^{n+1} + \vec{U}^{n})}{2}\cdot\Delta t. \tag{2.27}$$

3. Convergence acceleration techniques

3.1. Local time stepping in pseudo time

Due to the disparity in cell sizes in unstructured grid calculations, the chosen time step size for the entire mesh will be the minimum of the local time steps of all the control volumes for time accurate calculations. In this work, the large variation in grid size for the unstructured mesh will restrict the time step used and the smallest control volume dictates the maximum time step size. In order to overcome the above problems, each control volume can be advanced in pseudo time by its own maximum local time step, which greatly enhances the convergence rate. In this study, the maximum permissible local time step size is determined by the critical time step size:

$$\Delta\tau \leqslant \Delta\tau^{\mathrm{crit}} = \frac{2}{\omega_{\max}}, \tag{3.1}$$

where $\omega_{\max}$ is the largest natural circular frequency. For the cantilever case, the natural frequency is given by

$$\omega_{\max} = \frac{2c}{\Delta l},$$

where $c = \sqrt{E/\rho(1-\upsilon^2)}$ is the wave propagation velocity. Δl is the characteristic length scale associated with a node under consideration. Normally, it is taken as the smallest height of all the tetrahedral cells sharing the node. Substituting $\omega_{\max}$ into the equation for critical time step yields

$$\Delta\tau = \frac{\Delta l}{c}.$$

$\Delta\tau$ is the time needed for the wave to propagate through the cell of characteristic length Δl. The local time step size is estimated via CFL stability condition as

$$\Delta\tau = \mathrm{CFL}\cdot\frac{\Delta l}{c} = \mathrm{CFL}\cdot\frac{\Delta l}{\sqrt{E/\rho(1-\upsilon^2)}}. \tag{3.2}$$

The global physical time step size Δt for the entire mesh will be the minimum of the local time steps of all the control volumes for time accurate calculations. That is

$$\Delta t = \min\{\Delta\tau_1, \Delta\tau_1, \ldots, \Delta\tau_{ns}\}, \tag{3.3}$$

where *ns* is the total number of nodes.

3.2. Implicit residual smoothing

In order to speed up the convergence rate, an implicit residual smoothing scheme developed for unstructured grids is employed. The idea behind this is to replace the residual at one point of the domain with a smoothed or weighted average of the residuals at the neighbouring points [19]. The averaged residuals are calculated implicitly in order to increase the maximum CFL number, thus increasing the convergence rate. Normally this procedure allows the CFL number to be increased by a factor of 2 or 3. The smoothing equation for a vertex k can be expressed as follows:

$$\overline{R}_k = R_k + \varepsilon\nabla^2\overline{R}_k, \tag{3.4}$$

where $\overline{R}$ is the smoothed residual, R is the original residual, and ε is the smoothing coefficient, which can be defined as

$$\varepsilon = \max\left\{\frac{1}{4}\left[\left(\frac{\mathrm{CFL}}{\mathrm{CFL}^*}\right)^2 - 1\right], 0\right\}, \tag{3.5}$$

where CFL* is the maximum CFL number of the basic scheme. The solution to the preceding equations can be obtained on an unstructured grid by using the Jacobi iterative method as follows:

$$\overline{R}_k^{(m)} = R_k^{(0)} + \varepsilon\sum_{i=1}^{numnod(k)}\left[\overline{R}_i^{(m)} - \overline{R}_k^{(m)}\right],$$

i.e.

$$\overline{R}_k^{(m)} = \frac{R_k^{(0)} + \varepsilon\sum_{i=1}^{numnod(k)}\overline{R}_i^{(m-1,m)}}{1 + \varepsilon\cdot numnod(k)}, \tag{3.6}$$

where $numnod(k)$ is the number of neighboring nodes of vertex k.

3.3. The multigrid method

It is well known that multigrid methods can dramatically reduce the overall cost of CFD simulations. The basic idea of the multigrid method is to carry out early iterations on a fine grid and then progressively transfer these solutions and residuals to a series of coarser grids. On the coarser grids, the low frequency errors become high frequency ones and they can be easily eliminated by a time stepping scheme. The equations are then solved on the coarser grids and the corrections are then interpolated back to the fine grid. The process is repeated over a sufficient number of times until satisfactory convergence on the fine grid is achieved. In this study, based on the early works done by Zhao et al. [19,20], an efficient unstructured multigrid scheme has been developed for the equation of dynamic equilibrium, allowing convergence to be achieved with much less CPU time than the single-grid scheme. The multigrid algorithm is described as follows. The discretized dynamic equation can be expressed by Eq. (2.23), which is repeated here for convenience:

$$\frac{\Delta(\vec{U})}{\Delta\tau} = \tilde{\tilde{R}}^{n+1,m}, \tag{3.7}$$

where $\vec{U}$ is the velocity vector. This equation is solved iteratively by a dual-time stepping scheme. An pseudo code is given below to illustrate the basic procedures of the multigrid scheme. The outer cycles are based on the physical time t, which is numbered from 1 to *ktmax*, whereas the inner cycles are based on the pseudo-time τ, which is numbered from 1 to *max_itr_sub*. The grid levels range from 1 to *nmg*, where 1 is the finest level and *nmg* is the coarsest level. P_{h+1}^{h} is the prolongation operator from level $h+1$ to h, and Q_h^{h+1} and T_h^{h+1} are the residual transfer and restriction operators from level h to $h+1$.

```
ALGORITHM START
  call setupinterconnect !—> build up inter-connectivity relationship between levels
  DO kt = 1, ktmax !—> start of physical time step
    DO mg = 1, nmg
      call cvvol(mg) !—> calculate the volumes for control volume (CV) and cell
      call clhaut(mg) !—> calculate CV characteristic length for determination
                      !—> of $\Delta\tau$ and $\Delta t$ computation
    ENDDO
    call dtsize(mg = 1) !—> computation of the time-step size for the finest level
    DO itersub = 1, max_itr_sub !—> start of sub-iteration
      DO ialpha = 1, 5 !—> 5-stage Runge–Kutta time integration process
        call fvsolver(mg = 1) !—> solve Eq. (3.1) for $\vec{U}_1^{kt}$ following the way
                              !—> described in forgoing sections
      ENDDO
      DO mg = 2, nmg
        call transfrsol(mg − 1) !—> restrict solution from h to h + 1
                                !—> $(\vec{U}_{h+1}^{(0)} = T_h^{h+1}\vec{U}_h)$.
        call transfresd(mg − 1) !—> restrict residual from h to h + 1
                                !—>$(\tilde{\tilde{R}}_{h+1}^{(0)} = Q_h^{h+1}\tilde{\tilde{R}}(\vec{U}_h))$.
      IF (itersub.eq.1) THEN
        call dtsize(mg) !—> calculate the time-step size for the current level
      ENDIF
      DO ialpha = 1, 5
        call fvsolver(mg) !—> solve Eq. (3.1) for $\vec{U}_{mg}^{kt}$ based on the initial
                          !—> solution $\vec{U}_{h+1}^{(0)}$ and initial residual $\tilde{\tilde{R}}_{h+1}^{(0)}$
        ENDDO
      ENDDO
      DO mg = nmg, 2, −1
        call transfrcorct(mg) !—> prolongate correction from h + 1 to h
                              !—> $(\vec{U}_h^{+} = \vec{U}_h + I_{h+1}^{h}(\vec{U}_{h+1}^{+} - \vec{U}_{h+1}^{(0)}))$. Here $\vec{U}_h^{+}$ is
          !—> the updated solution for the finer grids
        ENDDO
      ENDDO !—> end of sub-iteration
      DO mg = 2, nmg
        call transfrsol(mg − 1) !—> restrict solution from h to h + 1
      ENDDO
```

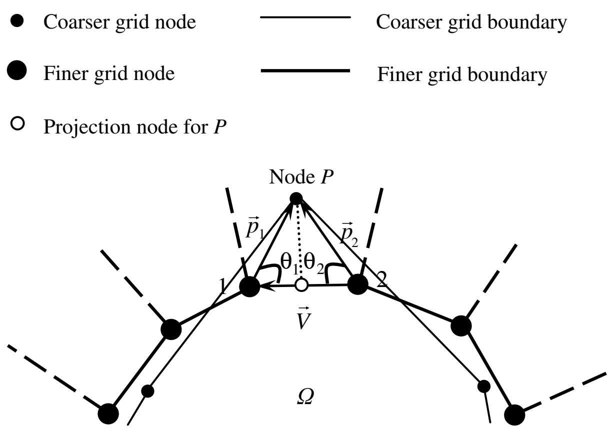


Fig. 2. Schematic of boundary node projection algorithm, Ω is the computational domain.

```
        DO mg = 1, nmg
          call updgrid(mg) !—> finally update the solid mesh
        ENDDO
      ENDDO !—> end of physical time step
  ALGORITHM END
```

For detailed discussion of the mesh-to-mesh transfer operators, please refer to Refs. [19,20]. One should note that we always use the initial inter-mesh connectivity, which is built up before the first time step starts, to perform the mesh-to-mesh transfer operations. In order to enable the solver to tackle more geometrically complex structures, the search for nodes within the boundary faces (edges for 2D) will continue after the search for nodes within the cells are performed if the user specifies that the physical wall of the geometry is curved. See Fig. 2 for example, a normal procedure cannot find a corresponding finer grid cell for the coarser grid node P. It means the node P cannot get a proper solution vector from the finer grid, which can pose a serious problem while solving the dynamic equation (3.1). Following the boundary node projection algorithm proposed in [20], we resort to finding a projection node for node P in the nearest finer mesh cell face and then use this projected node to perform the necessary multigrid operations.

4. Coupling with the fluid solver

We have already reported the development of a new parallel unstructured multi-grid preconditioned compressible Navier–Stokes solver in [19]. Here we aim to extend it for fluid–structure interaction simulation. The biggest challenge here is how to couple the two modules and synchronize them. Our solution algorithm is shown in Fig. 3, from which it may be seen that different time stepping sizes between the fluid and the structure solvers are allowed. In the beginning of every physical time step, the fluid solver (TETRAKE) is run to solve the fluid domain. And during this stage, IMM [19] is used to impose the boundary conditions across the fluid–structure interface, where those fluid cells crossing the interface need special treatment. As shown in Fig. 4 for example, a fluid cell *1234* is cut by the interface. Nodes *1*, *2* and *3* lie in the fluid domain while node 4 is in the structural domain. *abc*, *def* and *ghi* are triangles from the shell mesh of the structural domain. As described in [19], convection fluxes are computed based on mesh edges. In the computation of the convection flux along edge *14*, the flow conditions at node *1* and ghost node *4* (*g14*) will be involved. The conditions at ghost node *4* (*g14*) is determined as follows:

(1) Identify the intersection point I_1 between the edge *14* and the interface.
(2) Identify the interface surface triangle in which intersection point I_1 lies.
(3) Determine the velocity $\vec{u}_I$ at intersection point I_1 using a area weighted scheme with the knowledge of velocities at node a, b and c.

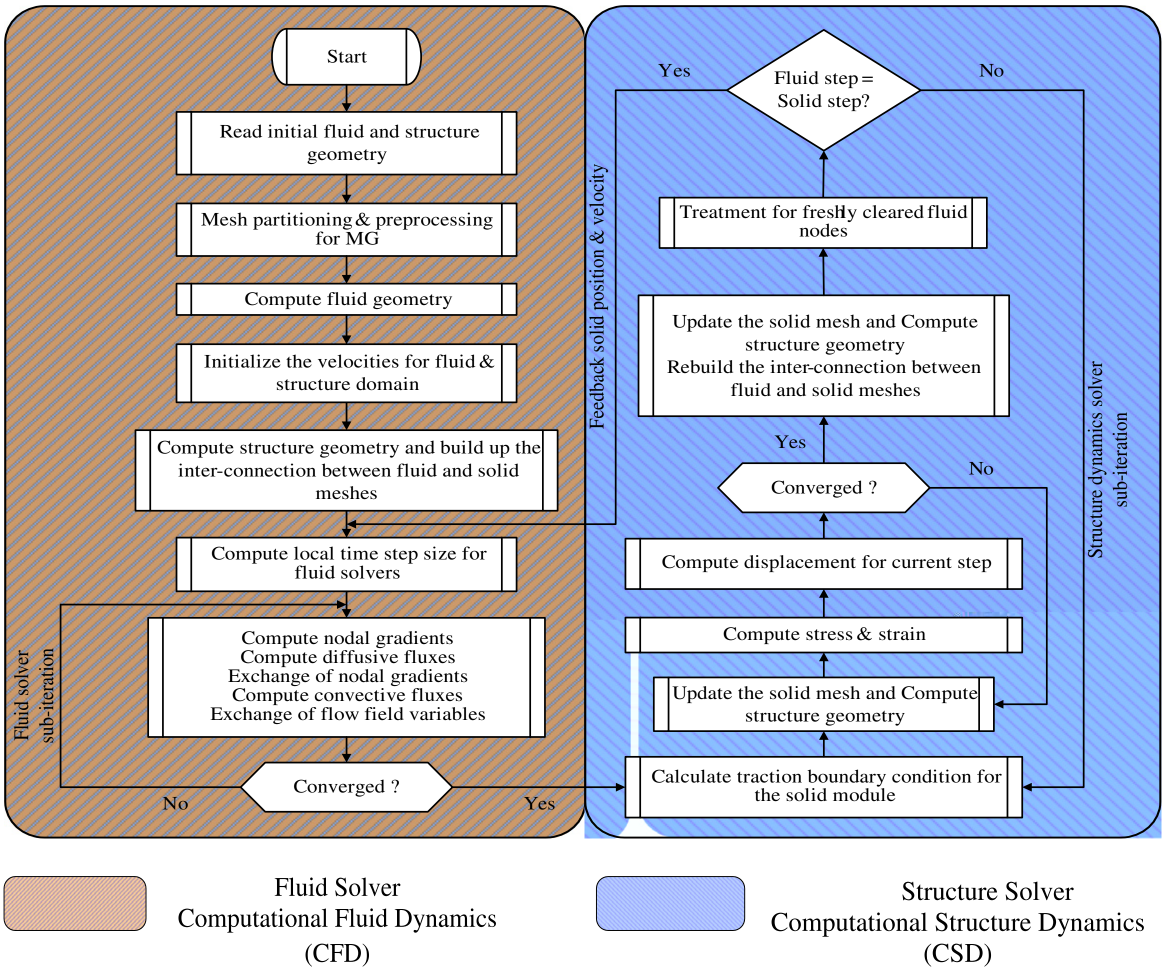

Start
Read initial fluid and structure geometry
Mesh partitioning & preprocessing for MG
Compute fluid geometry
Initialize the velocities for fluid & structure domain
Compute structure geometry and build up the inter-connection between fluid and solid meshes
Compute local time step size for fluid solvers
Compute nodal gradients
Compute diffusive fluxes
Exchange of nodal gradients
Compute convective fluxes
Exchange of flow field variables
Converged ?
No
Yes
Fluid solver sub-iteration
Feedback solid position & velocity
Fluid step = Solid step?
Yes
No
Treatment for freshly cleared fluid nodes
Update the solid mesh and Compute structure geometry
Rebuild the inter-connection between fluid and solid meshes
Yes
Converged ?
No
Compute displacement for current step
Compute stress & strain
Update the solid mesh and Compute structure geometry
Calculate traction boundary condition for the solid module
Structure dynamics solver sub-iteration
Fluid Solver
Computational Fluid Dynamics
(CFD)
Structure Solver
Computational Structure Dynamics
(CSD)

Fig. 3. Schematic of FSI solution algorithm.

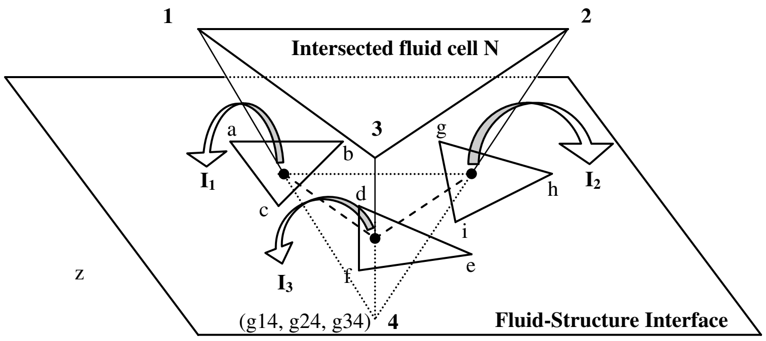


Fig. 4. Schematic of immersed membrane method.

(4) Extrapolate to find velocity at ghost node *g14* using the flow conditions of node *1* and the velocity of the intersection point I_1.
(5) Extrapolate to obtain pressure and density at ghost node *g14* using the known pressure, density and gradients at node *1*.

Extrapolations of ghost-node velocities are illustrated in Fig. 5a and b. Taking node *1* as the real node, velocity at ghost node *g14* is calculated as follows:

$$\begin{cases} \frac{\vec{u}_{g14}-\vec{u}_I}{\vec{u}_1-\vec{u}_I} = -\frac{|r_{4I}|}{|r_{1I}|} & \text{for } 0 \leqslant |r_{4I}| \leqslant |r_{1I}|, \\ \frac{\vec{u}_{g14}-\vec{u}_I}{\vec{u}_2-\vec{u}_I} = -\frac{|r_{4I}|}{|r_{2I}|} & \text{for } |r_{1I}| < |r_{4I}|. \end{cases} \tag{4.1}$$

Therefore,

$$\begin{cases} \vec{u}_{g14} = -\frac{|r_{4I}|}{|r_{1I}|}\cdot(\vec{u}_1 - \vec{u}_I) + \vec{u}_I & \text{for } 0 \leqslant |r_{4I}| \leqslant |r_{1I}|, \\ \vec{u}_{g14} = -\frac{|r_{4I}|}{|r_{2I}|}\cdot(\vec{u}_2 - \vec{u}_I) + \vec{u}_I & \text{for } |r_{1I}| < |r_{4I}|, \end{cases}, \tag{4.2}$$

where $\vec{u}_I, \vec{u}_1$ and $\vec{u}_{14}$ are the velocity vectors at intersection point I_1, node 1, ghost node *g14*. In Fig. 5b, we have $|r_{2I}| = |r_{4I}|$ and $|r_{11'}| = |r_{14}|$. $\vec{u}_2$ is evaluated as follows:

$$\vec{u}_2 = \frac{(|r_{11'}| - |r_{12}|)\vec{u}_1 + (|r_{2I}| - |r_{1I}|)\vec{u}_{1'}}{|r_{14}|},$$

where $\vec{u}_{1'} = \vec{u}_1 + \vec{r}_{41}\cdot\nabla\vec{u}_1$. In this work, the structure and fluid domains are coupled by enforcing the velocity continuity condition:

$$\vec{u}_\mathrm{s} = \vec{u}_\mathrm{f}, \tag{4.3}$$

where $\vec{u}_\mathrm{s}$ is the velocity vector of a structure surface node and $\vec{u}_\mathrm{f}$ the neighbouring fluid velocity. And Eq. (4.3) is used to extrapolate fluid velocity to its corresponding ghost nodes. The extrapolation of ghost-node pressure is illustrated in Fig. 5c. Taking node *1* as real node, the ghost-node pressure and density of *g14* is obtained by

$$\begin{aligned} p_{g14} &= p_1 + \vec{r}_{14}\cdot\nabla p_1, \\ \rho_{g14} &= \rho_1 + \vec{r}_{14}\cdot\nabla\rho_1, \end{aligned} \tag{4.4}$$

where p_{g14} and ρ_{g14} are the ghost-node pressure and density at *g14*, ∇p_1 and $\nabla\rho_1$ are their gradients at node *1*. $\vec{r}_{14}$ is the distance vector pointing from node *1* to node *4*. Suppose edge *24* is the next edge along which the convection flux will be computed. We need to determine the ghost conditions for node *4* (*g24*) again with the conditions of node *2* and intersection node I_2. As a result, every node can become multiple ghost nodes with their corresponding ghost values since a node can be connected by multiple edges, which is especially true

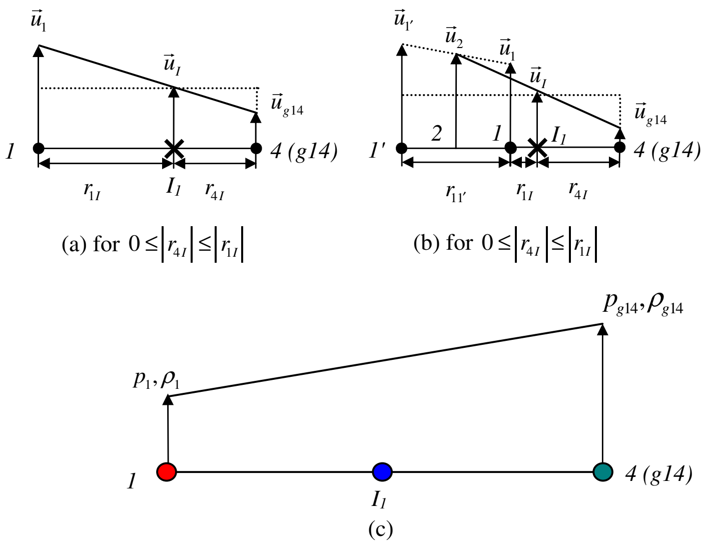


Fig. 5. (a) Velocity extrapolation for ghost node *4*; (b) pressure and density extrapolation for ghost node *4*.

for 3D unstructured grids. The selection of a particular ghost value depends on which edge and node the computation involves, which is the novel feature of IMM [19].

This linear interpolation results in a second-order accurate scheme. It is noted that higher-order MUSCL interpolation can also be applied here if higher-order accuracy is required. As depicted in Fig. 6, edge *12* is cut by the fluid–structure interface and *I* is the intersection point. *C* is the centre point of edge *12*. Local third-order accuracy can be achieved by introducing $\vec{u}_L$ and $\vec{u}_R$:

$$\begin{aligned}\vec{u}_L &= \vec{u}_1 + \frac{1}{4}[(1-\kappa)\Delta_1^- + (1+\kappa)\Delta_1^+],\\ \vec{u}_R &= (\vec{u}_1 + \vec{u}_{g12})/2,\end{aligned} \tag{4.5}$$

where

$$\begin{aligned}\Delta_1^+ &= \vec{u}_{g12} - \vec{u}_1,\\ \Delta_1^- &= 2\cdot\vec{12}\cdot\nabla\vec{u}_1 - (\vec{u}_{g12} - \vec{u}_1) = 2\cdot\vec{12}\cdot\nabla\vec{u}_1 - \Delta_1^+.\end{aligned} \tag{4.6}$$

And $\vec{u}_{g12}$ is the ghost velocity vector of node *2*, $\nabla\vec{u}_1$ velocity gradient at node *1*.$\vec{12}$ is the vector representing the edge. κ is set to 1/3, which corresponds to a nominally third-order accuracy. Pressure and density can be interpolated in a similar way. The viscous fluxes and gradients are computed based on fluid mesh cells. In the cell *N* in Fig. 4, when the viscous flux is computed for node *1*, the flow conditions at node *1*, *2* and *3* will be needed, as well as the ghost node conditions at node *4* (It is *g14* in this case. But it would be *g24* if the viscous flux for node *2* is computed).

After the computation of fluid domain, fluid forces acting on the structure are calculated on the fluid–structure interface and they are applied to advance the movement of the structure. In the structural domain, Eq. (2.1) is solved by the techniques described above with the fluid forces exerted by ambient fluids as boundary conditions. Boundary conditions on the surface of the structural domain Ω_s are described in terms of prescribed traction $\vec{t}_P$ on the boundary Γ_t and prescribed displacement $\vec{d}_P$ on the boundary Γ_d. In the IMM, the loosely coupling relationship between the fluid domain and the structure domain is depicted in Fig. 7. The fluid forces include pressure p, shear stress σ_t and normal stress σ_n, Since the fluid and structural meshes are non-confirmatory, the fluid pressure needs to be extrapolated to the fluid–structure interface for the computation of fluid forces. And again this extrapolation is edge based. In Fig. 4, I_3 is the intersection point

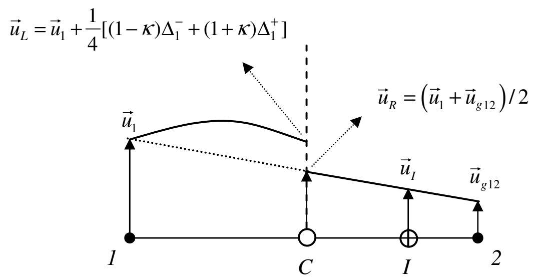

$\vec{u}_L = \vec{u}_1 + \frac{1}{4}[(1-\kappa)\Delta_1^- + (1+\kappa)\Delta_1^+]$
$\vec{u}_R = (\vec{u}_1 + \vec{u}_{g12})/2$
$\vec{u}_1$
$\vec{u}_I$
$\vec{u}_{g12}$
1
C
I
2

Fig. 6. Introduction of higher-order MUSCL like interpolation.

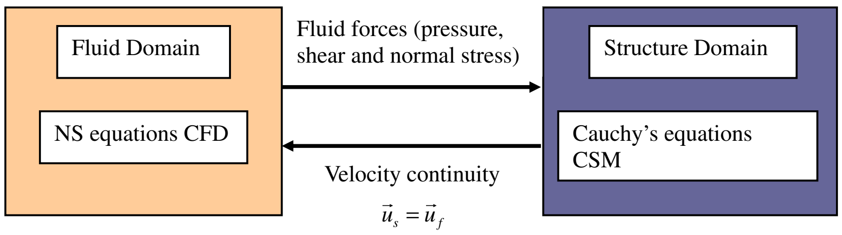


Fig. 7. Relation between the structure domain and the fluid domain in fluid–structure interaction.

between edge *34* and the interface, and we need to calculate the pressure at this point to compute the fluid force. This extrapolation procedure can be expressed as following:

$$p_{I_3} = p_3 + \vec{r}_{3I} \cdot \nabla p_3, \tag{4.7}$$

where p_{I_3} is the fluid pressure on the fluid–structure interface on the side of vertex *3*, ∇p_3 are the pressure gradient at vertex *3*, $\vec{r}_{3I}$ is the distance vector pointing from node *3* to point I_3. This linear extrapolation leads to a second-order accuracy. Note that the higher-order MUSCL scheme similar to Eqs. (4.5) and (4.6) can also be applied here. And then the result out pressure will be distributed to the structure nodes *d*, *e* and *f* based on an area weighted scheme. For Newtonian fluids, the fluid stress tensor is given as follows:

$$\bar{\bar{t}} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}; \tag{4.8}$$

and

$$\begin{aligned}
\sigma_{xx} &= 2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), \quad \sigma_{xy} = \sigma_{yx} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right),\\
\sigma_{yy} &= 2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), \quad \sigma_{xz} = \sigma_{zx} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right),\\
\sigma_{zz} &= 2\mu\frac{\partial w}{\partial z} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), \quad \sigma_{yz} = \sigma_{zy} = \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right).
\end{aligned}$$

where the fluid viscosity μ is property of the fluid and a function of temperature. So the fluid stress $\vec{\sigma}$ on the interface can be calculated as

$$\vec{\sigma} = \bar{\bar{t}} \cdot \vec{n} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}, \tag{4.9}$$

where $\vec{n}$ is the unit normal vector of the structure element. In the fluid solver, the velocity gradients are assumed to be constant within one fluid cell [19]. It means that on the fluid–structure interface the gradients of fluid velocities are equal to those at the cell centroids. But for a cell intersected by the immersed structure, the gradients are only uniform and continuous for centroids on the same side of the structure. So the computations of fluid stresses on the interface should use the right gradients of velocity. Use edge *34* in Fig. 4 for example, the fluid stresses at intersection point I_3 should be computed using the gradients of vertex 3, i.e.:

$$\sigma_3 = \bar{\bar{t}}_3 \cdot \vec{n} = \begin{bmatrix} \sigma_{3,xx} & \sigma_{3,xy} & \sigma_{3,xz} \\ \sigma_{3,yx} & \sigma_{3,yy} & \sigma_{3,yz} \\ \sigma_{3,zx} & \sigma_{3,zy} & \sigma_{3,zz} \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}. \tag{4.10}$$

5. Results and discussion

5.1. Deformation of point loaded fixed-free cantilever structures

5.1.1. Two-dimensional cantilever

A standard problem in structural mechanics is that of a fixed-free cantilever supporting an applied load at the free end [9,28,29]. The fixed-free cantilever is shown in Fig. 8, where $b = 2.0$ is the breadth, $L = 20.0$ the length of the cantilever and F the applied load. It is assumed that the depth $d = 1.0$. The static solution to this problem given by Timoshenko and Goodier [22] allows a slight distortion at the fixed end of the cantilever, whereas the solution given by Fenner [21] allows no such phenomenon. This test case requires no such displacement or distortion at the fixed end of the cantilever, hence at $x = L$ the y displacement at the free end of the cantilever according to Fenner [21] is given by

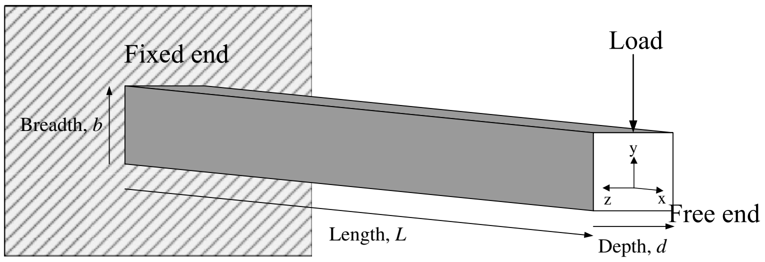


Fig. 8. Schematic of flexural (bending) deformation test of fixed-free cantilever.

$$d_y = -\frac{4FL^3}{Edb^3}, \tag{5.1}$$

where E is Young's modulus and d is the depth of the cantilever. The fixed-free cantilever has a load of 200 N at the free end, as depicted by Fig. 8. Note that the gravity effect is not considered in this study. The static solution given in Eq. (5.1) is independent of Poisson's ratio and is applicable to a cantilever undergoing pure flexure, i.e. no axial loads are supported and the out of plane load on the cantilever is zero. Thus for comparison with the analytic solution a zero Poisson's ratio is assumed. With the parameters as given in Table 1, Eq. (5.1) gives the static displacement in y direction at the tip of the cantilever as -0.08 m. Before embarking on a dynamic problem, a grid convergence study for the static displacement problem is carried out. The domain is meshed using triangular elements. The following five meshes are studied for increasingly higher mesh density:

- Coarse mesh, $20*2$ elements and 33 nodes.
- First refinement, $40*4$ elements and 105 nodes.
- Second refinement, $80*8$ elements and 369 nodes.
- Third refinement, $100*10$ elements and 561 nodes.
- Final refinement, $200*20$ elements and 2121 nodes.

For all of the grids the simulation is initiated by exerting the same load on the right tip of the cantilever until the solution is converged. The percentage errors in the y displacement are shown in Fig. 9, where percentage errors of less than 1.5% for the second mesh and less than 0.1% for the final mesh are observed. Thus the second mesh is considered to be sufficiently accurate and is used for the rest of the analysis. Fig. 10 shows the stress distribution in the equilibrium state.

In order to test the capability of this method for predicting dynamics of solid structures, we also perform a dynamic bending simulation for the fixed-free cantilever. We use a dynamic load, which is a sinusoidal function of time t:

$$F = 200\sin(0.05t). \tag{5.2}$$

With the other parameters as given in Table 1 and depth $d = 1.0$ m, Eq. (5.1) also gives the maximum displacement in y direction at the tip of the cantilever as 0.08 m. The simulation is kept running for a total of 2315 s. The cantilever tip displacement–time history in y direction is then plotted in Fig. 11, which has a maximum displacement of 0.081 m in good agreement with the analytic solution.

Table 1

2D Fixed-free cantilever computational parameters	Value
Load, F	200 N
Length, L	20.0 m
Breadth, b	2.0 m
Density, ρ	2600.0 kg/m^3
Young's modulus, E	10 MPa
Poisson's ratio, v	0.0

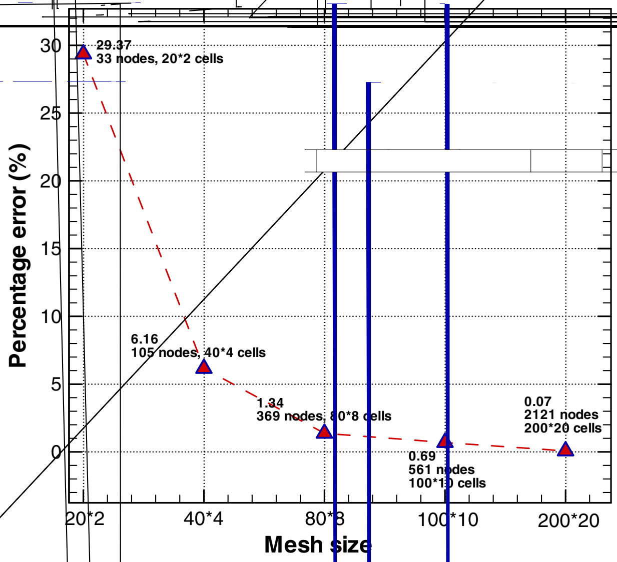


Fig. 9. Mesh refinement versus percentage error.

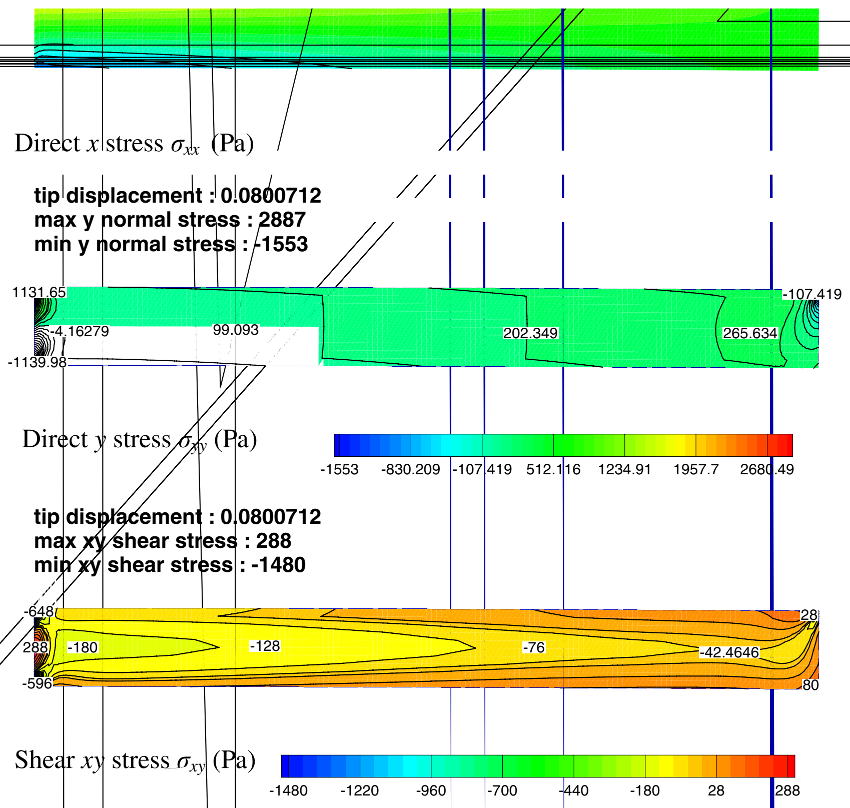
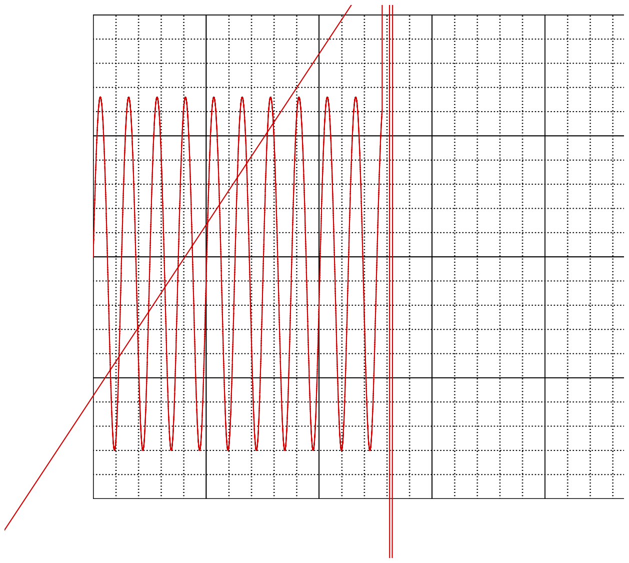


Fig. 10. Direct and shear stress distribution in the equilibrium state for the static point loaded free-fixed 2D cantilever.



5.1.2. Three-dimensional cantilever

The 2D problem studied in previous section is then extended to three dimensions and all of the computational parameters are given in Table 2.

Thus the static y displacement at the neutral axis of the tip given by Eq. (5.1) is about -0.1124 m, which provides an upper bound to the amplitude. Similar to the 2D simulation, the 3D statically loaded cantilever is first studied. The following three meshes are tested for grid convergence and multigrid studies:

- Coarse mesh, 2400 tetrahedral elements.
- First refinement, 3200 tetrahedral elements.
- Final refinement, 10,800 tetrahedral elements.

According to our results, the solutions for single grid and 3-level multigrid are identical. And from the numerical solution for the final mesh an amplitude error of 1.36% is observed. The tip displacement for this mesh turns out to be -0.11365 m. Fig. 12 shows the residual history and the convergence acceleration due to the 3-level multigrid method, which makes the convergence at least three times faster than single-grid solver in terms of CPU time (CPU times on a SGI O3400 workstation).

Next we will study the performance of the scheme for simulating dynamic loading. The natural frequency of a fixed-free cantilever [28] is given by

$$f = \frac{3.516}{2\pi L^2}\sqrt{\frac{EI}{\rho db}}, \tag{5.3}$$

Table 2

3D Fixed-free cantilever computational parameters	Value
Load, F	1.0 MN
Length, L	20.0 m
Breadth, b	2.0 m
Depth, d	2.0 m
Density, ρ	2600.0 kg/m^3
Young's modulus, E	17.8 GPa
Poisson's ratio, ν	0.3

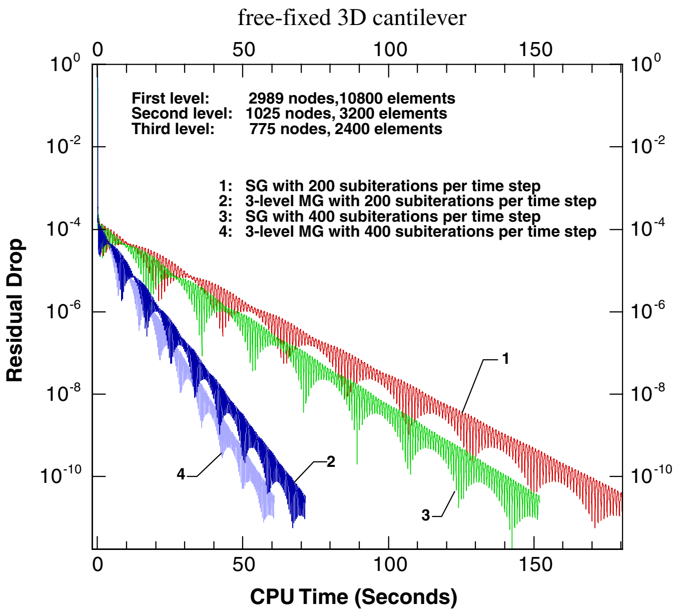


Fig. 12. Convergence rate acceleration given by 3-level MG method. In all of the tests, the solver stops only if the residual drop to a given level, which is $R/R0 = 10^{-12}$.

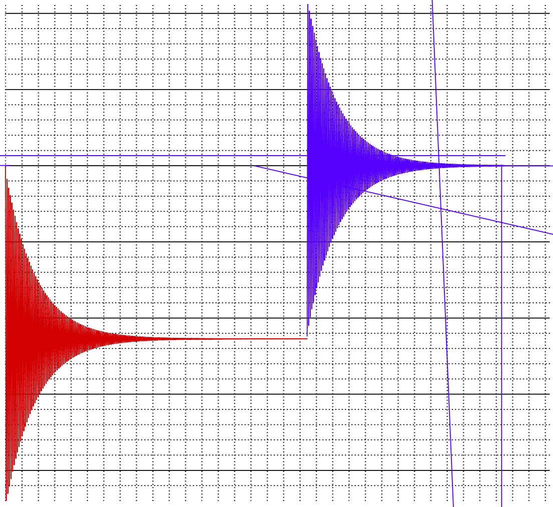
where I is the moment of inertia in the plane or the second moment of area of the cantilever about the neutral axis [13] and is given by

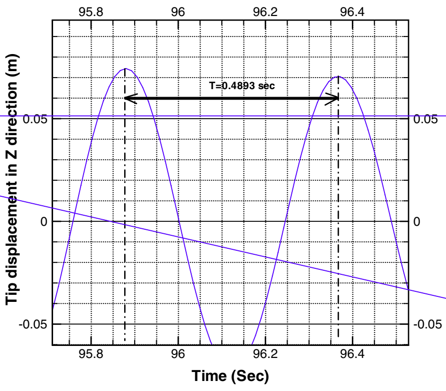
$$I = \int_{-\frac{1}{2}b}^{\frac{1}{2}b} z^2 \, \mathrm{d}A = \frac{db^3}{12}. \tag{5.4}$$

Using the same parameters in Table 2, from Eqs. (5.3) and (5.4), the 3D fixed-free cantilever has a fundamental natural frequency of 2.1134 Hz with a period of oscillation of 0.4732 s. To capture the sinusoidal motion of the structure accurately, the time step for the solver is set to 0.05 s. The load is exerted on the initially un-deformed 3D cantilever and then kept until the static equilibrium state of the cantilever is reached, after which the load is suddenly removed. The calculated tip displacement history is depicted in Fig. 13. The computed period is 0.4893 s, which again agrees well with the theoretical one. As pointed out by Slone in [28], for accuracy considerations, the updating of the cantilever grid in this study is based on the initial position and the total displacement method. In order to accelerate the convergence rate, the multigrid method is used for all of the 3D problems.

5.2. Three-dimensional fixed-free cantilever immersed in fluid flow

The basic configurations of the problem are shown in Fig. 14 and the boundary conditions include an inflow boundary on the top of the domain, outflow boundary at the bottom and a non-slip wall boundary to which the cantilever is clamped. A symmetry plane is also used to divide the field into two halves because of its geometric symmetry. The fluid domain is meshed using 1,307,075 tetrahedral elements and the overall dimensions are: depth $d = 2.0$ m, breadth $b = 2.0$ m and length $l = 20.0$ m. The cantilever is meshed using 12,196 tetrahedral elements. The Reynolds number and inflow Mach number are set to 18.7×10^5 and 0.05, respectively, for this case. The material properties of the cantilever are: Young's modulus 21.0 GPa, Poisson's ration 0.3 and density 2600 kg/m^3. The fluid is air and the free stream density is 1.293 kg/m^3. According to Eq. (5.3), the cantilever has a period of oscillation of 0.4356 s. The simulation time step is taken as 0.025 s. The gravity force is also ignored in the simulation. The stress distribution in the cantilever in its final equilibrium state is shown in Fig. 15 and the tip vertical displacement history is depicted in Fig. 16, while the flow field in the final steady state is shown in Figs. 17–19, respectively. They show that while the beam is oscillating,

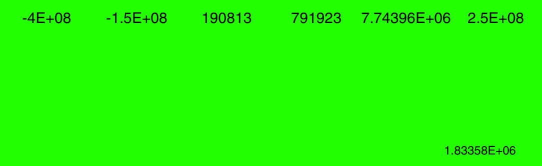


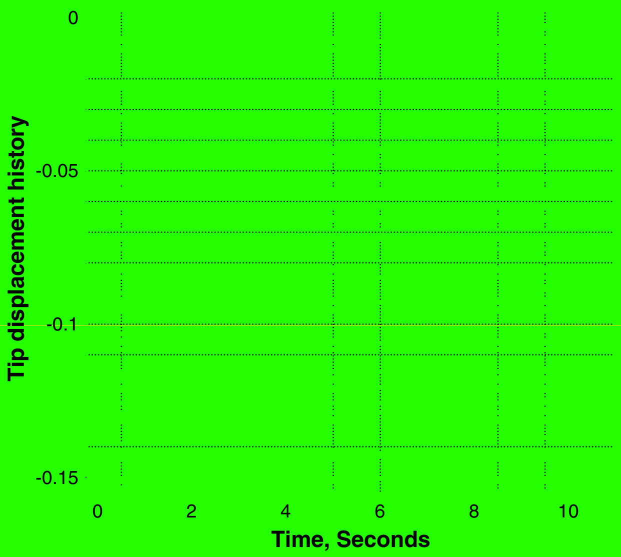


a complex flow field is also developing under the beam and the wake region fluctuates with the beam accordingly.

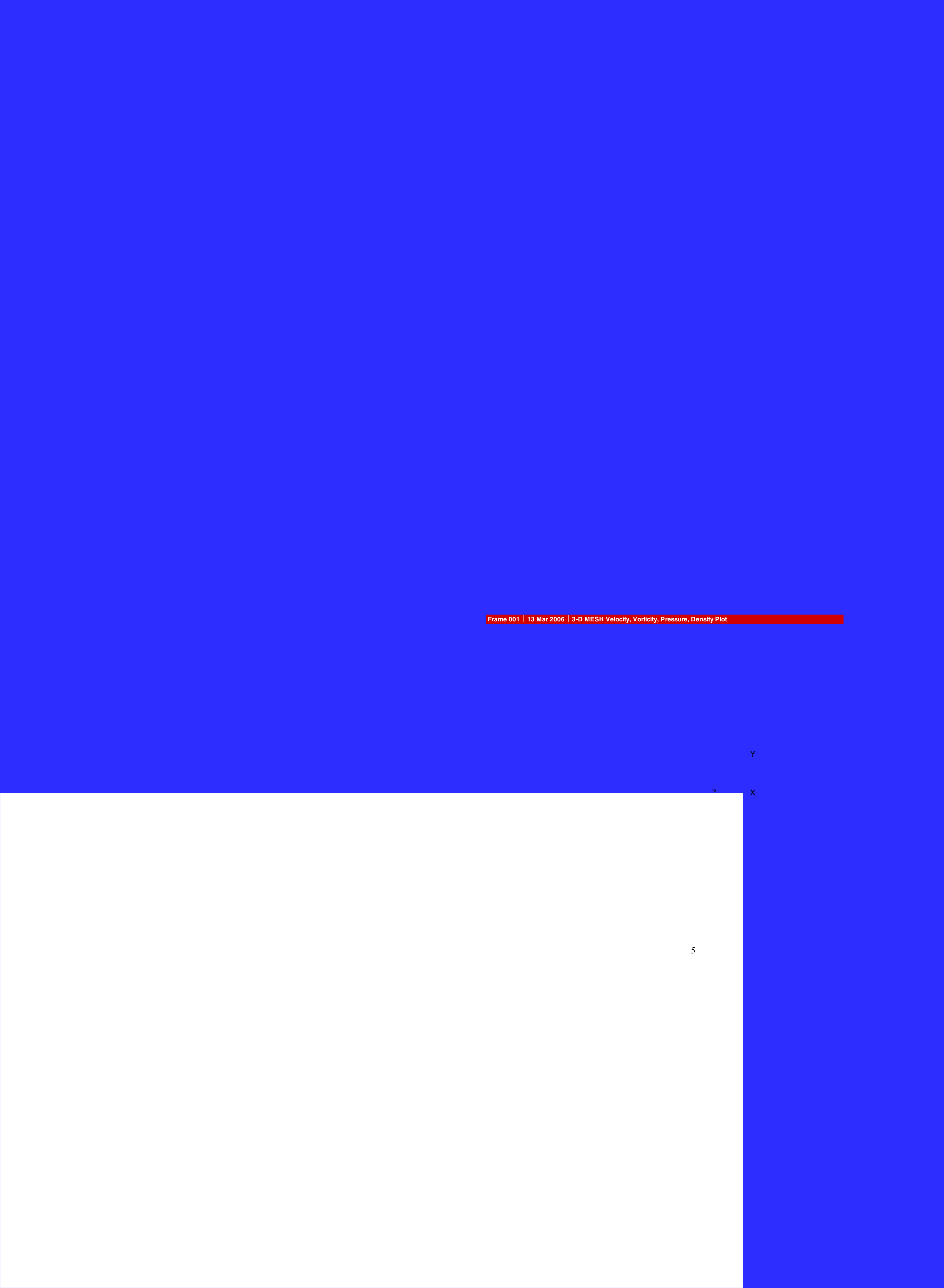
A grid convergence study for this case is also performed to evaluate the accuracy of the proposed numerical methods. Four successively finer meshes, with 434,572, 921,586, 1,307,075, and 1,908,265 tetrahedral cells, respectively, are used for error analysis, and the finest-mesh solution is considered to be the 'exact' solution. On all the grids the same physical time step ($\Delta t = 0.01T$) is employed in order to concentrate on the spatial resolution of the method. For all the grids, the L_∞ and L_q norms of the u-velocity errors are calculated as follows:

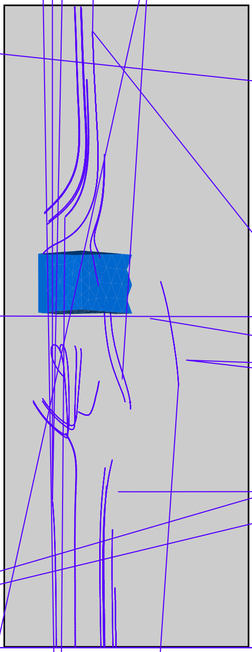
$$\varepsilon_j^\infty = \max_{i=1,N} |u_i^- u_i^{\mathrm{e}}|, \quad \varepsilon_j^q = \left[\frac{1}{N}\sum_{i=1}^{N} |u_i^- u_i^{\mathrm{e}}|^q\right]^{1/q} \qquad (j = 1, 4\ j = 1 \text{ is the coarsest grid}),$$





The immersed cantilever tip vertical displacement history during the FSI process.


Frame 001 | 13 Mar 2006 | 3-D MESH Velocity, Vorticity, Pressure, Density Plot
Y
Z
X



5.3. A comparison with an existing solver

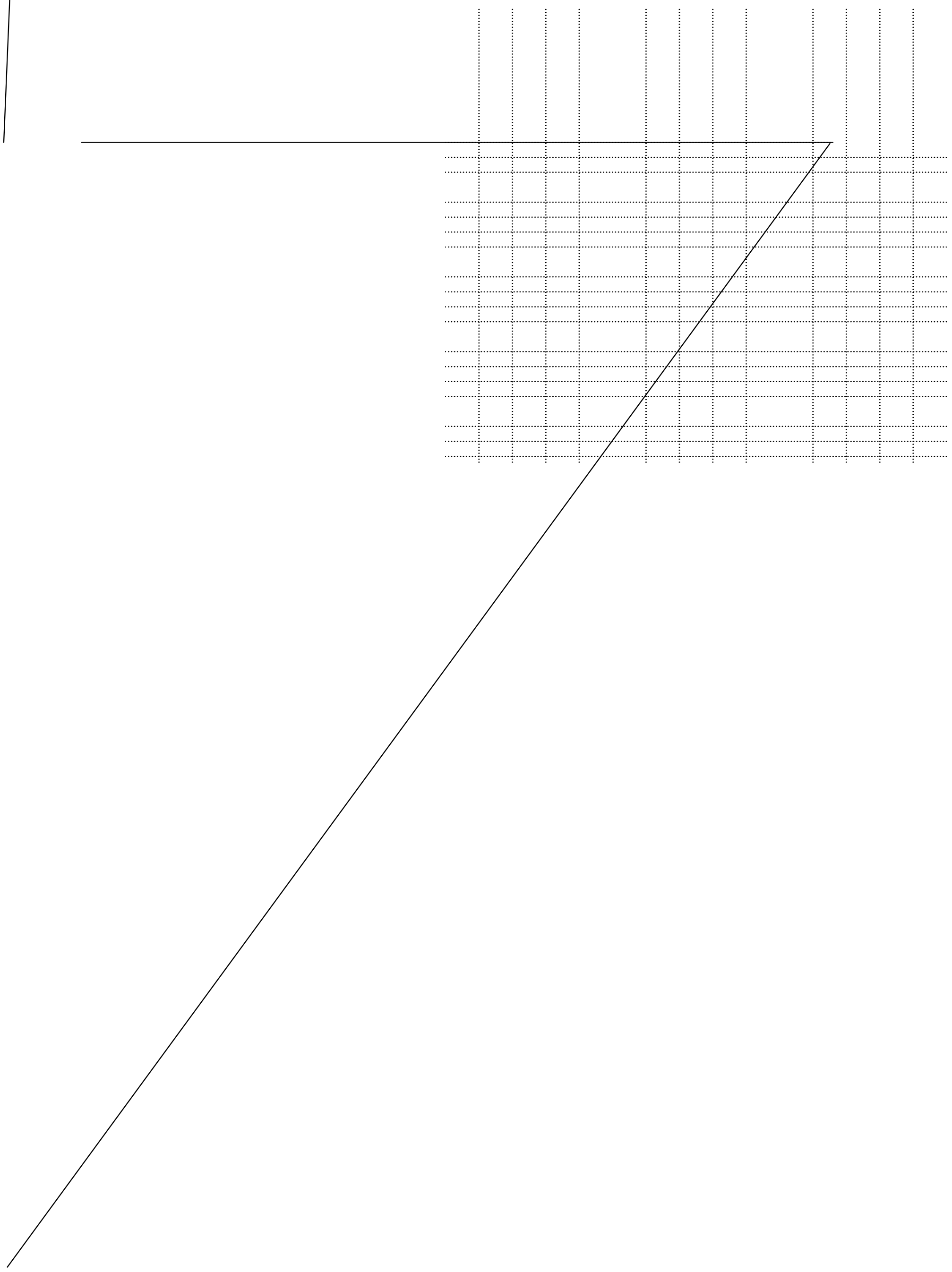
To further demonstrate and evaluate the performance of the proposed basic structural solver, both the solver and a commercial solver (ANSYS 9.0) are used to simulate the equilibrium state of the point loaded 3D fixed-free cantilever as used in the above FSI calculation, with a load of 1.0 MN. The geometry and material properties of the cantilever remain the same.

Table 3
Rate of convergence γ calculated for different error norms

Norm	Grids
	$j = 2, 3, 4$
L_∞	1.93
L_1	2.18
L_1	2.15

Using the given parameters, the theoretical value of the vertical displacement of the tip is 0.0952381 m. Four meshes are tested here:

- grid one, 3950 tetrahedral elements;
- grid two, 6412 tetrahedral elements;



- grid three, 12,196 tetrahedral elements;
- grid four, 19,183 tetrahedral elements.

Fig. 21 shows a comparison of the computational times (CPU times on a SGI O3400 workstation) used by the proposed basic structural solver and ANSYS 9.0. It is found that the two solvers use almost the same amount of CPU time to make the residual drop to the same level (at $R/R0 = 10^{-5}$), with the current solver being slightly more efficient when the mesh density is increased. Fig. 22 shows a comparison of grid convergence for the proposed solver and ANSYS 9.0. It can be observed that the current solver can produce faster convergence toward the exact solution than ANSYS. From the above comparisons, we can conclude that the proposed basic solver is comparable to established solvers in terms of efficiency and accuracy. With the use of multigrid, the efficiency will be significantly improved, as demonstrated in Section 5.1.2.

6. Conclusions

A novel 3D matrix-free implicit unstructured multigrid structural dynamic finite-volume solver has been successfully developed and validated. The convergence of numerical solutions is found to be significantly improved with the help of the implicit unstructured multigrid method. The efficiency and accuracy of the solver is fully validated using a point loaded fixed-free cantilever, for which both 2D and 3D static and dynamic cases are throughoutly tested. Through the case involving a fixed-free cantilever immersed in fluid flow it is found that the current FV structural dynamic solver works well with our unstructured grid FV compressible fluid solver TETRAKE as well as the immersed membrane method [19]. These studies demonstrate the potential capability of the proposed method for large-scale complex fluid–structure interaction simulation.

References

[1] D.R.J. Owen, E. Hinton, Finite Elements in Plasticity: Theory and Practice, Pineridge Press Ltd, Swansea, UK, 1980.

[2] O.C. Zienkiewicz, R.L. Taylor, The Finite Element MethodBasic Formulation and Linear Problems, vol. 1, McGraw-Hill, Maidenhead, UK, 1989.

[3] S.V. Patanker, Numerical Heat Transfer and Fluid Flow, Hemisphere, Washington, DC, 1980.

[4] C. HirschNumerical Computation of Internal and External Flows: Fundamentals of Numerical Discretization, vol. 1, Wiley, New York, 1988.

[5] O.C. Zienkiewicz, Origins, milestones and directions of the finite element method – a personal view, Archives of Computational Methods in Engineering 2 (1995) 1–48.

[6] E. Onate, M. Cervera, O.C. Zienkiewicz, A finite volume format for structural mechanics, International Journal for Numerical Methods in Engineering 37 (1994) 181–201.

[7] G.A. Taylor, A vertex-based discretization scheme applied to material non-linearity within a multi-physics finite volume framework, Ph.D. Thesis, The University of Greenwich, 1996.

[8] I. Demirdzic, D. Martinovic, Finite volume method for thermo-elasto-plastic stress analysis, Computer Methods in Applied Mechanics and Engineering 109 (1992) 331–349.

[9] J.H. Hattel, P.N. Hansen, A control volume-based finite difference method for solving the equilibrium equations in terms of displacements, Applied Mathematical Modelling 19 (1995) 210–243.

[10] M.A. Wheel, A geometrically versatile finite volume formulation for plane elastostatic stress analysis, Journal of Strain Analysis 31 (2) (1996) 111–116.

[11] M.A. Wheel, A mixed finite volume formulation for determining the small strain deformation of incompressible materials, International Journal for Numerical Methods in Engineering 44 (1999) 1843–1861.

[12] H. Jasak, H.G. Weller, Application of the finite volume method and unstructured meshes to linear elasticity, International Journal for Numerical Methods in Engineering 48 (2000) 267–287.

[13] Y.D. Fryer, C. Bailey, M. Cross, C.-H. Lai, A control volume procedure for solving the elastic stress–strain equations on an unstructured mesh, Applied Mathematical Modelling 15 (1991) 639–645.

[14] C. Bailey, M. Cross, A finite volume procedure to solve elastic solid mechanics problems in three dimensions on an unstructured mesh, International Journal for Numerical Methods in Engineering 38 (1995) 1757–1776.

[15] I. Demirdzic, S. Muzaferija, Finite volume method for stress analysis in complex domains, International Journal for Numerical Methods in Engineering 37 (1994) 3751–3766.

[16] B.R. Baliga, S.V. Patanker, A new finite-element formulation for convection–diffusion problems, Numerical Heat Transfer 3 (1980) 393–409.

[17] V. Selmin, The node-centred finite volume approach: bridge between finite differences and finite elements, Computer Methods in Applied Mechanics and Engineering 102 (1992) 107–138.

[18] S.R. Idelsohn, E. Onate, Finite volumes and finite elements: two ‘Good Friends’, International Journal for Numerical Methods in Engineering 37 (1994) 3323–3341.
[19] X. Lv, Y. Zhao, et al., An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3d unsteady compressible flows with moving objects, Journal of Computational Physics 215 (2) (2006) 661–690.
[20] C.H. Tai, Y. Zhao, A finite volume unstructured multigrid method for efficient computation of unsteady incompressible viscous flows, International Journal for Numerical Methods in Fluids 46 (1) (2004) 59–84.
[21] R.T. Fenner, Engineering Elasticity: Applications of Numerical and Analytical Techniques, Ellis Horwood, 1986.
[22] S.P. Timoshenko, J.N. Goodier, Theory of Elasticity, McGraw-Hill, New York, 1982.
[23] T.J.R. Hughes, The Finite Element MethodLinear Static and Dynamic Finite Element Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[24] R.D. Blevins, Flow Induced Vibration, Van Nostrand, Rheinhold, 1977.
[25] R.R. Craig Jr., Structural Dynamics, Wiley, New York, 1981.
[26] S.W. Key, Transient response by time integration, in: J. Donéa (Ed.), Advanced Structural Dynamics, Applied Science Publishers, 1978, pp. 71–95.
[27] O.C. Zienkiewicz, R.L. TaylorThe Finite Element Method, vols. 1–3, Butterworth-Heinemann, 2000.
[28] A.K. Slone, C. Bailey, M. Cross, Dynamic solid mechanics using finite volume methods, Applied Mathematical Modeling 27 (2003) 69–87.
[29] A.K. Slone, K. Pericleous, C. Bailey, M. Cross, C. Bennett, A finite volume unstructured mesh approach to dynamic fluid–structure interaction: an assessment of the challenge of predicting the onset of flutter, Applied Mathematical Modeling 28 (2004) 211–239.