2016

Finite Volume Based Fluid-Structure Interaction

Mohamed M. Selim
University of Alabama at Birmingham

FINITE VOLUME BASED FLUID-STRUCTURE INTERACTION

by

MOHAMED M. SELIM

ROY P. KOOMULLIL, COMMITTEE CHAIR
DAVID R. MCDANIEL
DAVID L. LITTLEFIELD
HAIBIN NING
AMR A. HASSAN

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2016

FINITE VOLUME BASED FLUID-STRUCTURE INTERACTION

MOHAMED M. SELIM

INTERDISCIPLINARY ENGINEERING

ABSTRACT

Fluid-Structure Interaction (FSI) is an important topic that needs to be addressed during the design and analysis of air vehicles. The main components of an FSI analysis framework include: 1) computational fluid dynamics (CFD) solver, 2) computational structural dynamics (CSD) solver, 3) mesh deformation module, and 4) module for data transfer between computational fluid and structural solvers.

In this PhD research work a loosely-coupled FSI methodology with all of the above components in a single framework has been developed. An in-house CFD solver has been used for the solution of the fluid dynamics equations. A structural solver has been developed by discretizing the linear elasticity equations using the finite-volume method that is used in the in-house CFD solver. The developed finite volume based structural mechanics solver has been validated against analytical and finite element based numerical results. Results were found to be in a good agreement for bending and tensile deflections as well as for distributed and concentrated loads. Furthermore, the implemented CSD methodology was tested for the prediction of large deflection cases. Two different dimensionless load magnitudes have been applied. The error in the deflection amplitude for the lower load and higher load were 5.5% and 7.19%, respectively, with frequency shifts of 1.0% and 4.8%, respectively.

In addition, an extensive survey of the existing mesh deformation techniques has been conducted. Per this survey, Radial Basis Functions (RBF) based mesh deformation technique has been adopted for further improvements. A novel concept to solve the RBF system of equations incrementally combined with a greedy algorithm has been introduced. This improvement decreased the computational complexity of solving the RBF's system of equations from $O(n^3)$ *to* $O(n^2)$, where n is the total number of fluid mesh boundary nodes. Benchmark test cases with four different analytic deformations were used to evaluate the performance of the presented approach. Mesh deformation results were also presented for deflections of a cantilever beam and a rectangular supercritical wing. These simulations showed that the developed incremental approach saves up to 67% of CPU time as compared to the traditional RBF solvers.

The developed FSI methodology is a general purpose one that can be applied to different types of problems and is capable of handling any mesh topology. The numerical compatibility between the CFD and CSD solvers implies same mesh requirements for both domains. Therefore, an identical surface meshes at the interface have been used for both fluid and structural domains. This eliminated the interpolation errors during the data transfer between the structural and fluid solvers. The developed FSI approach has been tested on the case of flow-induced cantilever beam vibration. Four different flow inlet speeds have been analyzed. The predicted beam vibration has preserved the natural frequency for all cases. Furthermore, increasing the flow velocity increases the magnitude of the beam deflection, as expected. Moreover, two different structural densities were simulated. The results were validated against the FSI results produced by coupling a finite element structural solver with a finite volume fluid solver. The predicted structural

response was found to be in a good agreement for both density values. The proposed FV-FSI solver under-predicted the maximum deflection by 7% and over-predicted the vibration frequency by 0.16%.

Keywords: structural mechanics, finite volume, mesh deformation, radial basis functions, cantilever beam, flow induced vibration

DEDICATION

I dedicate my dissertation work to my father who has believed in me and prepared me for success since I was a child. A special feeling of gratitude goes to my loving parents who have always loved me unconditionally, and whose words of encouragement ring in my ears. To my beloved wife, Mariam, for her unlimited love and support. I wish to express my heartfelt love to my little daughter, Leilah, for coping with the undue paternal deprivation during the course of my study. Most of all I pledge allegiance to the Lord Almighty for the strength and encouragement He has given me.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support and help of a great number of people, and it is impossible to properly acknowledge everyone who made this work possible. First, I would like to thank my supervisor, Dr. Roy P. Koomullil, for the guidance, advice, encouragement and patience he has provided throughout my time at University of Alabama at Birmingham. Dr. Koomullil supported me in my dissertation work, as well as in my professional and personal life, and I have been extremely lucky to have such a supervisor who was always present to respond to my questions and queries so promptly. I may never fully comprehend the totality of the support and guidance of my advisors through these years; Drs. David McDaniel, David Littlefield, Haibin Ning, and Amr Hassan. You shall always have my respectful gratitude, and I will be always indebted for the knowledge and experience you have sent my way. Thanks to Dr. Thomas Attard from Civil Engineering for his time and attention on this endeavor.

I would like to thank Dr. Zeljko Tukovic from the University of Zagreb (Croatia), Department of Mechanical Engineering for providing valuable instruction and sharing important information which have been beyond helpful to make this research possible. Also, thanks go to all the past and present graduate students and friends who have given more than can be expressed, especially: Ahmed Hattab, Benjamin M.G. Willis, Ahmed Khourshed, Ossama Ramadan and Eric Winardi. A special thank you goes out to Ahmed

Arabi who is a friend, brother and great support on my entire journey. From my undergraduate studies to the PhD studies he always kept my spirits high.

The financial assistance from the Arab Academy for Science and Technology and Maritime Transport (AASTMT) is greatly appreciated; without your help and support this work would not have been possible. AASTMT provided me with the proper education and research experience during my B.Sc. and M.Sc. studies which gave me the qualifications and the capabilities to perform this Ph.D. level work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CFD	Computational Fluid Dynamics
CSD	Computational Structural Dynamics
FDM	Finite Difference Method
FEM	Finite Element Method
FSI	Fluid-Structure Interaction
FVM	Finite Volume Method
IDW	Inverse Distance Weighting
MSA	Moving Submesh Approach
RBF	Radial Basis Function
TFI	Transfinite Interpolation
RMS	Root Mean Square

CHAPTER 1

INTRODUCTION

BACKGROUND

The importance of obtaining numerical results in engineering applications has increased greatly over recent years. The ability to model and simulate engineering systems allows more informed decisions to be made during the design as well as the operating phase. In many cases numerical analysis can provide a level of detail that is difficult or even impossible to obtain using experimental methods. Moreover, the rapid improvement in computer hardware makes the associated cost of performing numerical simulations very feasible in comparison to the computational cost of performing experiments.

Numerous computational fluid dynamics (CFD) and computational structural dynamics (CSD) solvers have been developed independently over the decades. The results from computational simulation are now accurate enough to predict real physical phenomena. However, in many physical problems it may not only be the fluid medium in isolation or the structural medium in isolation that is of interest; these two mediums might interact together and also many other physical processes might contribute to the problem, such as chemical reaction, heat transfer, radiation, and acoustics. An efficient and accurate numerical scheme is required to solve such multiphysics problems.

Fluid-Structure Interaction (FSI) is one of the most important topics that need to be addressed during the design and analysis of many engineering problems. There are numerous applications which involve FSI effect. Typical examples include: the wind

induced oscillations of structures, the floating offshore components under wave motion, the aeroelastic deformation of wings, and the expansion and contraction of blood vessels.

There are three different categories of FSI analysis based on their degree of detail. These approaches include: strongly-coupled, loosely-coupled, and reduced order modeling. In the strongly-coupled approach, the fluid and the structural state equations are combined and treated as a single monolithic system of equations. On the other hand, the loosely-coupled approach solves the equations governing the dynamics of the fluid and the structure in an alternating manner. Reduced order modeling is usually thought of as computationally inexpensive mathematical representations that offer the potential for near real-time analysis using a certain number of system responses, i.e. mode shapes. The main components of an FSI framework consist of: 1) computational fluid dynamics (CFD) solver, 2) computational structural dynamics (CSD) solver, 3) mesh deformation module, and 4) module for data transfer between computational fluid and structural solvers. Figure 1 shows how the information is being transferred between the different FSI components.

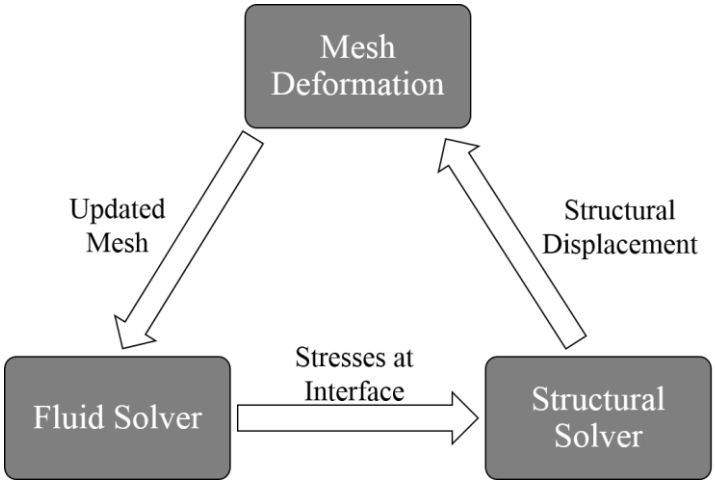


Figure 1 FSI main components

The strongly-coupled scheme provides more accurate results than the loosely-coupled one. However, it requires more effort to formulate and implement, and it also requires higher computational resources which tend to be more feasible with the recent advances in hardware [1, 2]. Another advantage is the seamless coupling of the fluid and structure domains, which can lead to improved solution stability. On the other hand, the main drawbacks of this approach are:

1. Generating a single mesh that is of high quality for both domains creates a challenge [3]
2. A single time step must be used for both domains, which can lead to inefficiencies when totally different time scales present [4]
3. A highly ill-conditioned matrix system may arise, especially when the fluid and the structure domain's rigidities are highly incompatible [5, 6]

The loosely-coupled scheme is the simplest approach for simulating FSI problems in which separate programs are used to simulate the fluid flow and solid deformation. The interface stresses and deformations are being transferred between the programs. However, since the two programs may use different numerical grids based on the numerical schemes; interpolation is needed to transfer the information between these programs. Furthermore, an interface code is required to transfer the data between the two programs, since each package has different data structures for storing the information. In addition, the use of commercial CSD solvers for the FSI analysis decreases the efficiency due to lack of access to the source codes, file based data transfer, and script based synchronization. This can also affect the parallel performance due to the use of separate processors for the CFD and CSD solvers. Data interpolation at the interface between the two solvers and transferring the data between different data structures lead to solution inaccuracy, particularly for unsteady problems with large structure deformations.

The limitations of using loosely-coupled FSI scheme can be overcome by eliminating the use of commercial CSD and CFD solvers; instead use CSD and CFD solvers that are compatible with each other. If the two mediums are solved in the same exact manner with respect to the numerical discretization approach, as well as the same data structures for storing the solutions, then a similar numerical grid can be used; and accordingly there will be no need to interpolate the solutions over the interface between the fluid and the structure. Consistent implementation also removes the need for a software interface, allowing more efficient computation and parallel implementation.

Most of the CSD solvers used at present are based on Finite Element Methods (FEM) [7]. However, state-of-the-art CFD solvers are based on Finite Volume Methods (FVM) [7]. The mesh requirements for these solvers are different and that necessitates interpolation of forces from the CFD solver to the CSD solver and displacement from the CSD solver to the CFD solver. FEM based structural dynamics solvers are known for a better order of accuracy as compared to FVM based solvers. However, it has been reported in the literature that mesh refinement for finite volume schemes would result in comparable solution accuracy to an FEM based simulation [8-10]. Even with a refined mesh for the structural domain, the mesh size will be much smaller as compared to the mesh for the fluid domain. Therefore, the computational time that is needed for the structural solver using a finite volume scheme will be small as compared to that of the CFD solver.

Another important element within the FSI framework is the mesh deformation algorithm. A successful fluid mesh deformation, due to structural deflections, is a critical step in the analysis of this class of problems. For an accurate simulation, the fluid volume mesh needs to move conformal to the structure, with very little degradation in quality. Due

to the repeatability of the mesh deformation and the large number of fluid cells, an efficient and reliable approach is needed for a successful analysis. In addition, the fluid mesh is typically partitioned and handled by different processors. This necessitates that the mesh deformation algorithms be parallelizable for efficient simulations. The main complexity of the moving mesh approach is to find an optimum technique that is suitable for different mesh topologies and physical situations. At the same time, it should preserve, as much as possible, the quality of the mesh while keeping computational cost low.

Strategies for deforming the fluid mesh conforming to the deformation of structure can be divided into two basic classes: physical analogy or interpolation. The physical analogy approach describes the fluid mesh deformation according to a physical process that can be modeled using numerical methods. One of the popular methods in this class is the tension spring analogy by Batina [11]. In the interpolation approach an interpolation function is used to transfer prescribed boundary point displacements to the fluid mesh. In general, these schemes do not require connectivity information. Therefore, these algorithms can be applied to arbitrary mesh types that contain general polyhedral elements or hanging nodes [12].

STUDY OBJECTIVES

Since the existing in-house CFD solver is finite-volume based, it was decided that the main objective of this work is to develop a finite-volume based FSI analysis methodology. This was achieved by solving the structural dynamics equations using the finite volume numerical approach and coupling this solver to the in-house CFD solver. This will enforce same mesh requirements and same surface meshes for both CSD and CFD solvers, alleviating the data interpolation problem. Also, a conservative stresses

transfer between the two domains will be enforced by default. Furthermore, a novel Radial Basis Function (RBF) interpolation based mesh deformation approach was developed to handle the fluid mesh deformation to accommodate the structural deflections.

The main objectives of this research can be concluded in the following four objectives:

1- Develop and validate a finite volume based linear elasticity methodology

2- Conduct an extensive survey of the available mesh deformation techniques

3- Develop an efficient mesh deformation technique

4- Couple the developed modules in objective 1 and 3 with the in-house CFD code and perform and validate FSI cases

OUTLINE OF DISSERTATION

Chapter 2 provides a literature review of the recent studies in the field of Fluid-Structure Interaction, with a focus on the studies dealt with finite volume based structural mechanics approaches. Chapter 3 describes the governing equations for linear elasticity in integral form and the numerical implementation of the finite volume stand-alone structural solver. Test cases are also presented in Chapter 3 for the purpose of validation of the numerical approach. Following this, Chapter 4 provides an extensive survey of the available mesh deformation techniques. Chapter 5 describes the development of a novel mesh deformation approach. The FSI coupling procedure and FSI test cases are presented in Chapter 6. Finally, the conclusions from this work and recommendations for future work are presented in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

FLUID-STRUCTURE INTERACTION

The main approaches to solve FSI problems, which are strongly-coupled and loosely-coupled, are reviewed in this section. An overview of modeling of the FSI problem in aero-elasticity can be found in Bennett et al. [13] and Dowell et al. [14]. The review of FSI in liquid filled pipe systems can be found in Tijsseling et al. [15].

The most common approach to carry an FSI analysis is to employ a loosely-coupled or partitioned approach. The main advantages of the loosely-coupled approach are the ability to separately develop and advance the flow modeling and structural modeling software [16-19] and to generate individual meshes for the structural and the fluid domains, which allow for different mesh resolutions. The main drawback to this approach is the possible degradation of solution stability due to small errors during the coupling step [20, 21].

One of the major applications of FSI that has been studied by many researchers using strongly-coupled scheme is Aero-elasticity. Behr [22] analyzed incompressible flow over two-dimensional airfoil and cylinder using implicit stabilized space-time formulation for moving boundaries and interfaces, and a new stabilized velocity-pressure-stress formulation. Carstens et al. [23] analyzed the flutter behavior of turbo-machinery blades in the time domain. The structural part of the governing aeroelastic equations is time-integrated according to the Newmark algorithm, while the unsteady air-loads are computed

at every time step by a Navier–Stokes code. However, their applications with the strongly-coupled methods are limited to two-dimensional or simple three-dimensional geometry because the possibility of divergence of solution is increased as geometries become complicated. Another problem associated with the strongly-coupled scheme is to yield ill-conditioned matrix system from the differences of properties and scales used in CFD and CSD formulations. Other applications with this method in aero-elasticity can be found in Refs. [24-28].

Demirdzic and Muzaferija [29] coupled a finite-volume CFD solver to a finite volume solid solver. The fluid solver included the SIMPLE pressure correction algorithm and k–ε turbulence model, whilst the solid solver used the thermo-elastic form of the constitutive relations. A coupled FSI simulation was performed for a simplified air-cooled engine; however, only qualitative results were given to demonstrate the possibilities of the method, rather than to test its accuracy.

Yates [30] developed a finite volume based two-dimensional FSI solver. He used the SIMPLE method of Patankar and Spalding [31] for pressure-velocity linkage in the CFD solver with the Quadratic Upwind Interpolation for Convective Kinematics (QUICK) scheme of Leonard [32] in order to increase the accuracy of the discretization by including higher order terms. The system of discretized equations for both fluid and structure solvers had been solved using the Tri-Diagonal Matrix Algorithm (TDMA). The developed algorithm used a single numerical mesh to cover both fluid and solid sub-domains, and it was capable of handling only structured meshes. The coupled solver was verified and validated through three test cases: steady laminar flow through an initially straight tube with a compliant wall section; steady flow (laminar and turbulent) through a compliant

walled stenosis, and unsteady turbulent flow through a compliant aneurysm. Only one of the test cases was in a good agreement with the data existed in the literature.

A wide range of applications involves the analysis of a flow-induced cantilever beam vibration. Wang et al. [33] used the Euler-Bernoulli beam theory and the normal mode method to analyze the cantilever structural response and used the finite element method to resolve the flow field. Other possible applications are the reed valve, two-stroke engines, compressors, and shock absorbers [34]. Cantilevers with cylindrical shapes are also important for cardiovascular applications, where the internal flow of the blood exerts stresses leading to deformation in arteries and veins [35]. Furthermore, fluid–structure interaction is dominant for very small-scale dynamics problems in biomedical applications [36].

COMPUTATIONAL STRUCTURAL DYNAMICS

Many computational techniques have been used in the field of structural dynamics, but the most widely adopted technique is the finite element method. This method was developed by Richard Courant in 1943 [37]. The method was put to practical use on computers in the mid 1950's by aeronautical structures engineers M. J. Turner, R. W. Clough, H. C. Martin and L. J. Topp in the United States [38] and by J. H. Argyris and S. Kelsey in Britain [39]. In the most common version of the finite element method, the domain to be analyzed is divided into elements, and the displacement field within each element is interpolated in terms of displacements at a few points around the element boundary, and sometimes within it, called nodes. The interpolation must ensure the continuity of the displacement field across element boundaries for any choice of the nodal

displacements. Then it is required that the stress-strain relations of the material satisfy the principle of virtual work for arbitrary variation of the nodal displacements [40].

FEM was the choice in the solid mechanics area due to its ability in dealing with arbitrary domains and its less complex mathematical models for elasticity problems. On the other hand, fluid flow equations which include high level of nonlinearities are more suitable to be coupled and solved using finite volume methods (FVM) [41]. However, for solving coupled problems that involve both structural and fluid systems, the use of the same discretization methodology provide the advantage of using the same mesh and avoid the need for solutions interpolation and coupling interfaces. Thus, more focus is being concentrated in the past few years into providing a more robust and efficient FVM for solving structural dynamics problems.

The first use of the FV method to numerically solve structure problems was by Demirdzic et al. [42], who used the method to simulate thermo-elastic problems. Following this, Demirdzic and Martinovic [43] used the FV method to simulate thermo-elastoplastic problems. This method solved the unsteady form of the equations governing thermal energy conservation and momentum balance, with displacement components and temperature as dependent variables. The elasto-plastic form of the solid body constitutive relations, which relate stresses to displacement gradients, and Fourier's law, which relates heat flux to temperature gradient, were used to close the system of equations. Both works used a structured numerical grid. The authors concluded the following advantages for the use of the FV method to solve structure problems: 1) the method is simple and efficient; 2) the method is conservative on both the local and global scale; 3) boundary conditions are simple to prescribe; and 4) non-linearities can be handled at little extra cost. Demirdzic and

Muzaferija [29] then applied the FV method for unstructured grids for improving the ability to deal with complex geometries. In this case the thermo-elastic constitutive relations were used. As a further development, Bijelonja et al. [40] used an FV based method to solve incompressible elastic problems. Wheel [44] presented an FV based method to calculate the stresses and displacements within axisymmetric bodies. He stated that the finite-volume method would be particularly useful in the fluid-structure interaction analysis of pressure vessels, pipes and heat exchangers.

Several studies focused on application of FV method to linear elasticity [8, 10, 45, 46]. Jasak and Weller [46] used a segregated cell-centered FV approach to discretize the linear elasticity governing equation. Moreover, for the sake of efficient parallelization, they treated the displacement vector implicitly and treated the inter-component coupling term explicitly. Tukovic and Jasak [45] used a similar approach to numerically calculate large deformation dynamic response of an elastic body. Xia and Lin [10] employed an implicit dual time-stepping cell-vertex FV method to obtain time accurate solutions for linear elasticity problems. They adopted a dual time-stepping scheme by adding a pseudo time derivative term to the governing equation. Then they sought the solution by marching into the pseudo time using five-stage Runge-Kutta scheme. Suliman et al. [8] also adopted the dual time-stepping scheme; however, they proposed a hybrid finite volume method. Instead of using the standard cell-vertex FV method, they proposed a hybrid scheme where they calculated the shear stress component at the cell center and calculated the normal stress components at the cell vertices. They demonstrated their methodology on two-dimensional structured test cases.

COMPUTATIONAL FLUID DYNAMICS

Computational fluid dynamics, CFD, is the branch of fluid mechanics that uses numerical analysis to solve and analyze problems that involve fluid flows. The first attempt to use a discrete numerical model to simulate fluid motion was made by the British meteorologist, L. F. Richardson, at the beginning of the 20th century [47]. He developed the first numerical weather prediction system when he divided physical space into grid cells and used a primitive finite difference approximations. His attempt to calculate weather for a single eight-hour period took six weeks of real time and ended in failure [47].

During the 1960s, NASA scientists at Los Alamos in the U.S. contributed many numerical methods that are still in use in CFD today: Particle-In-Cell (PIC), Marker-and-Cell (MAC), Vorticity-Stream function methods, Arbitrary Lagrangian-Eulerian (ALE) methods, and the ubiquitous k-ε turbulence model [48].

In the 1973, the CFD group at Imperial College, London, developed Parabolic flow codes that predict simple shear flows, free and confined jet flows [49]. In 1974, they extended their codes to include Vorticity-Stream function based codes, the SIMPLE algorithm and the TEACH code, as well as the form of the k-ε equations that are used today [50]. Another key event in CFD history was in 1980 when Patankar published his book which is considered the most influential book on CFD to date [51].

In-House CFD Solver (HYB3D)

When considering the simulation of a flow field on a given spatial domain, the physical problem can be modeled by a set of partial differential equations conveniently written in a conservative form

$$\frac{\partial(Q)}{\partial t} + \frac{\partial F_i(Q)}{\partial x_i} = \frac{\partial G_i(Q)}{\partial x_i} + S(Q) \quad (1)$$

where Q is the vector of conserved variables, $F_i(Q)$ is the convective flux vector along the i-direction, $G_i(Q)$ is the diffusive flux vector along the i-direction and $S(Q)$ is a source term. In the case of viscous flows, equation (1) is given by the conservative formulation of the Navier-Stokes equations. While in the case of inviscid flows, equation (1) is given by the Euler equations which are a simplified form of the Navier-Stokes equations. In order to solve this equation numerically, the domain is required to be discretized into a set of discrete points for which the equations will be solved after providing some initial and boundary conditions [52]. Few of the most popular spatial discretization methods are briefly described below.

The Finite Difference Method

The finite difference method (FDM), considered the simplest method that has an historical importance. FDM is based on filling the computational domain with regularly spaced discretization points and the derivatives are approximated using Taylor series expansion. One of the earliest demonstrations of the use of applying finite difference approximation was the study by Courant et al. [53]. They used a discrete analogue of Dirichlet's principle to define an approximate solution by means of the five point approximation of Laplace's equation. By extending Taylor expansions higher orders of accuracy can be obtained. However, this requires extended stencils and can rapidly become oscillatory [54]. Despite the simplicity of the FDM, it was limited to simple geometry problems because of the difficulty associated with generating structured meshes for complicated geometries. However, in the early 1990's, NASA developed an overset grid

CFD solver, called OVERFLOW, that is capable of solving problems of complicated geometries by means of finite difference approximations [55, 56].

The Finite Element Method

The finite element method, FEM, is very popular for the structural analysis of solids but is also applicable to fluids. In the Finite Element method, a continuous function fitting to a finite element space is used to approximate the solution. The solution is assumed to be of the form:

$$Q(\vec{x},t) = \sum_{j=1}^{n} N_j Q_j \tag{2}$$

where N_j is the nodal shape function defined at node j and Q_j is the solution at node j. There are two methods of defining the shape functions, 1) the Galerkin method in which the shape function is defined locally, and 2) the Collocation method in which the shape function is defined globally [57].

The Finite Volume Method

The finite volume method, FVM, is the main method used in this. FVM uses a volume integral formulation of the problem with a finite partitioning set of volumes, defined by a numerical grid, to discretize the equations. The cells of the grid usually consist of rectangles or triangles in two-dimensional, and tetrahedral, pyramids, prisms, or hexahedra in three-dimensional. Generally, any type of volume that is bounded by a number of planar surfaces is possible; see Figure 2.

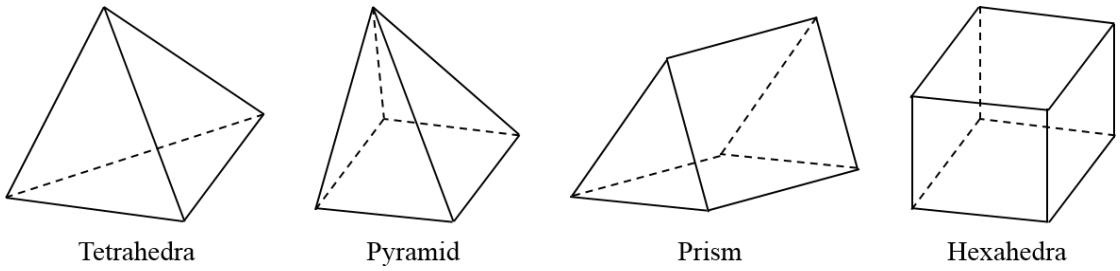


Figure 2. Control volume shapes.

After discretizing the computational domain, Ω, into a finite number of control volumes, Ω_n, the conservative equation is expresses in the integral form

$$\int_{\Omega_n} \frac{\partial Q}{\partial t}\, d\Omega_n + \int_{\Omega_n} \frac{\partial F_i}{\partial x_i}\, d\Omega_n = \int_{\Omega_n} S(Q)\, d\Omega_n \tag{3}$$

By applying the Gauss theorem the following expression is obtained

$$\int_{\Omega_n} \frac{\partial Q}{\partial t}\, d\Omega_n + \oint_{\partial \Omega_n} F_i n_i\, ds = \int_{\Omega_n} S(Q)\, d\Omega_n \tag{4}$$

where *n* is the outward pointing unit normal field of the boundary ds.

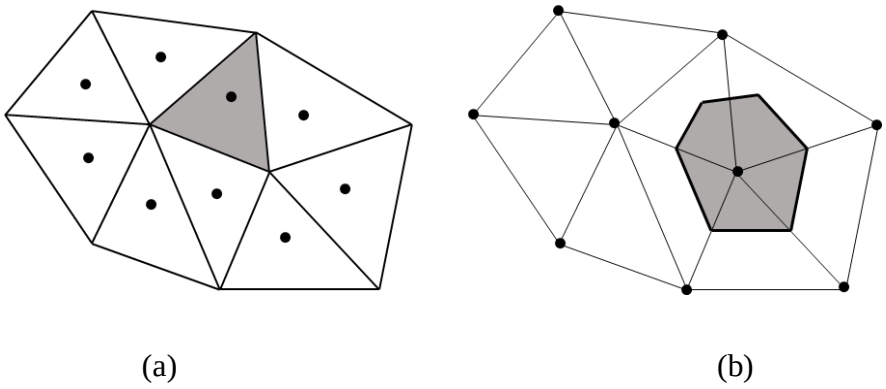
The spatial discretization term is computed by summing the contribution of the fluxes through the faces of the control volume.

$$\oint_{\partial \Omega_n} F_i n_i\, ds = \sum_f F_i n_i \cdot ds \tag{5}$$

The flux terms are conserved for any adjacent control volumes that share a common face. This is considered to be the main advantage of FVM.

There are two basic approaches for defining the shape and position of the control volume within the numerical grid. First, the cell-centered approach in which the control volumes are identical to the grid cells and solution variables are stored at the centroid of the grid cells. Second, the cell-vertex approach in which the solution variables are stored

at the grid points and the control volume is formed by a volume around each grid point. Figure 3 illustrates each approach of defining the control volume shapes.



(a) (b)

Figure 3. Control volume defining approaches: (a) cell-centered and (b) cell-vertex.

HYB3D FEATURES

The in-house fluid dynamics solver, HYB3D, is an Euler/Navier-Stokes flow simulation solver for generalized grids. It is capable of dealing with any type of numerical grids, e.g. structured, unstructured or an agglomeration of cells with an arbitrary number of faces. The spatial discretization of the governing equations is based on a cell-centered, finite volume upwind scheme. The upwind scheme is one of the most stable discretization schemes that uses the upstream information along with the flow direction to evaluate the flow properties at the boundaries. Similar to the developed structural solver, the control volume that is used for the flux integration is taken as the cell itself.

The convective fluxes at the cell-faces are evaluated using either Roe's approximate Riemann solver or van Leer's flux vector split scheme. The viscous flux at a cell-face is taken as the average of those on either sides of the cell-face. The turbulent viscosity at the cell center is estimated using the Spalart-Allmaras one equation model. The gradients at

the cell center for the Taylor series expansion are estimated using the Gauss theorem together with a weighted averaging procedure. In order to avoid the creation of local extrema during the reconstruction process, a limiter function is added to the Taylor's series expansion. Time integration is performed implicitly and the flux Jacobian is evaluated either analytically assuming the Roe averaged matrix to be constant or numerically. The block sparse matrix system resulting from the linearization of the governing equations is solved using a symmetric Gauss-Seidel algorithm. Numerical methods used in this framework and the results from the validation are presented in the literature [58, 59].

MESH DEFORMATION

Strategies for deforming the fluid mesh conforming to the deformation of structure can be divided into two basic classes: physical analogy or interpolation. The most popular algorithms of each of these classes are discussed below.

Mesh Deformation using Physical Analogy

The physical analogy approach describes the fluid mesh deformation according to a physical process that can be modeled using numerical methods. One of the popular methods in this class is the tension spring analogy by Batina [11]. In this approach, each edge of the mesh is replaced by a tension spring with the spring stiffness taken as inversely proportional to the edge length. The drawback of this approach, especially for fine meshes or high amplitude movements, is the mesh crossing. One of the improvements to this approach is the torsional spring approach introduced by Farhat et al.[60], in which fluid mesh is considered as a network of torsional springs added at the nodes to prevent cell collapse and to allow cell rotation. There are modifications reported in the literature in

which the mesh is considered as a combination of tension and torsion springs and also the stiffness of the springs is varied locally based on mesh cell size or proximity to the deforming surface. The main drawback of these methods is that they involve large systems of equations, implying a higher computational cost. Besides, these methods require grid connectivity information which results in more storage requirements and difficulties in parallelization.

A later advancement under the physical analogy approach is the use of multidimensional linear elasticity analogy [61], in which the mesh is interpreted as a continuous elastic medium. The modulus of elasticity is chosen as inversely proportional to the cell volume or to the distance from the deforming boundaries. In this approach, each displacement component of a mesh movement is governed by a partial differential equation, such as Laplacian equation [62].

Mesh Deformation using Interpolation Analogy

In this approach, an interpolation function is used to transfer prescribed boundary point displacements to the fluid mesh. In general, these schemes do not require connectivity information. Therefore, these algorithms can be applied to arbitrary mesh types that contain general polyhedral elements or hanging nodes [12].

Recently, a novel interpolation based scheme has been developed by Luke [12]. In this scheme, the deformation of the volume mesh is viewed as a projection of the surface deformation into the volume. Using a tree-code optimization, the algorithm cost is demonstrated to be $O(n\log(n))$, where n is the total number of nodes in the simulation, with mesh quality that is competitive to radial basis function (RBF) scheme.

McDaniel and Morton [63] developed a technique that is based on a two-pronged approach where the viscous layers of nodes are deformed rigidly and the outer region is deformed with two different interpolation techniques. Several different rigid deformation schemes were investigated. However, the results showed that the best performing scheme was based on a semi-rigid connection to the owner surface nodes defined as part of the mesh parsing, which provided smoother deformation in convex regions. The last layer of the viscous region was used as the deforming boundary surface for the outer region deformation.

The radial basis function interpolation method, such as the method developed by Boer et al. [64], is one of the promising interpolation schemes. RBF's have become a well-established tool to interpolate scattered data. RBF can also be used as an interpolation function to transfer the displacements known at the boundaries of the structural mesh to the fluid mesh. This scheme produces high-quality meshes with reasonable orthogonality preservation near deforming boundaries. Other advantages of RBF includes: 1) it avoids the need for mesh connectivity information, 2) the system of equations which needs to be solved is linear, and 3) the size of the linear system of equation is proportional to the number of boundary nodes, not all fluid nodes. Moreover, many studies have investigated different techniques for improving RBF's interpolation based mesh deformation. The most influential study was made by Rendall and Allen [65]. They proposed the use of data reduction algorithm along with RBF interpolation. This technique will be discussed later in details. Another study, which builds up on the previous technique, is the work done by Sheng and Allen [66], in which they put forward specific criteria for selecting the nodes involved in the interpolation.

CHAPTER 3

OBJECTIVE 1: FINITE VOLUME BASED STRUCTURAL SOLVER

INTRODUCTION

In this chapter, an approach to solve the governing equations of linear elasticity is presented. The main principles, simplifications and assumptions inherent in a linear elastic analysis are also highlighted in this chapter. The equations in this chapter have been written with reference to use Cartesian coordinate system.

The implementation of FV methods can be classified into two main categories: the cell-centered approach [67-69] and the cell-vertex approach [3, 70, 71]. Both approaches are locally and globally conservative. In the cell-vertex approach, the displacement and stress variables are stored at the nodes of the control volumes formed by the mesh elements. In the cell-centered approach, the variables are stored at the centroids of the control volumes that are formed by the mesh elements. In this work, the main goal is to couple the structural solver with the in-house CFD solver based on the cell-centered approach. Therefore, for a better compatibility, the cell-centered approach was chosen to develop the linear elasticity structural solver.

GOVERNING EQUATIONS

Cauchy's First Law

The governing equations for a structural domain undergoing motion can be written as,

$$\frac{\partial(\rho V)}{\partial t} - \nabla \cdot \sigma + \rho\, C\, V = \rho f \tag{6}$$

where V is the velocity vector, ρ is the density, C is the damping coefficient, f is the body force per unit mass and σ is the stress tensor.

Equation (6) is also known as the stress equation of small motion, the equation of equilibrium, or the equation of motion [70].

The Hooke's Law

The generalized form of Hook's law provides the following stress-strain relationship for isotropic homogeneous material undergoing small strains [72].

$$\sigma = 2\mu\varepsilon + \lambda\, tr(\varepsilon)\boldsymbol{I} \tag{7}$$

Or in matrix notations

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{31}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix}2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0\\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0\\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0\\ 0 & 0 & 0 & 2\mu & 0 & 0\\ 0 & 0 & 0 & 0 & 2\mu & 0\\ 0 & 0 & 0 & 0 & 0 & 2\mu\end{bmatrix}\begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \varepsilon_{23}\\ \varepsilon_{31}\\ \varepsilon_{12}\end{bmatrix} \tag{8}$$

Note: in this case, the stress and strain tensors are assumed to be symmetric (i.e. $\sigma_{12} = \sigma_{21}$).

Here, $\boldsymbol{I}$ is the unit tensor and μ and λ are the Lamé's coefficients, relating to the Young's modulus of elasticity E and the Poisson's ratio ν as [46]:

$$\mu = \frac{E}{2(1+v)}, \qquad \lambda = \begin{cases} \frac{vE}{(1+v)(1-v)} & \text{Plane Stress} \\ \frac{vE}{(1+v)(1-2v)} & \text{Plane Strain and three-dimensional} \end{cases}$$

The strain tensor ε is defined in terms of the displacement vector U [73]:

$$\varepsilon = \frac{1}{2}[\nabla U + (\nabla U)^T] \tag{9}$$

Where U is the displacement vector.

Or in matrix notation

$$\begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{1}{2}\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right) & \frac{1}{2}\left(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1}\right) \\ \frac{1}{2}\left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2}\right) & \frac{\partial u_2}{\partial x_2} & \frac{1}{2}\left(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}\right) \\ \frac{1}{2}\left(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3}\right) & \frac{\partial u_3}{\partial x_3} \end{bmatrix} \tag{10}$$

By substituting equation (10) into equation (8), we get

$$\sigma = \begin{bmatrix} \lambda\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z}\right) + 2\mu\frac{\partial u_1}{\partial x} & \mu\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) & \mu\left(\frac{\partial u_1}{\partial z} + \frac{\partial u_3}{\partial x}\right) \\ \mu\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) & \lambda\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z}\right) + 2\mu\frac{\partial u_2}{\partial y} & \mu\left(\frac{\partial u_2}{\partial z} + \frac{\partial u_3}{\partial y}\right) \\ \mu\left(\frac{\partial u_3}{\partial x} + \frac{\partial u_1}{\partial z}\right) & \mu\left(\frac{\partial u_3}{\partial y} + \frac{\partial u_2}{\partial z}\right) & \lambda\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z}\right) + 2\mu\frac{\partial u_3}{\partial z} \end{bmatrix} \tag{11}$$

For convenience, the stress tensor within equation (6) is rewritten in terms of the gradient of the displacement ∇U:

$$\frac{\partial(\rho V)}{\partial t} - \nabla \cdot [\mu\nabla U + \mu(\nabla U)^T + \lambda \boldsymbol{I}\, tr(\nabla U)] + \rho\, C\, V = \rho f \tag{12}$$

Special Cases

Plane Stress

For plane stress conditions, the normal and tangential stresses in the 3rd direction are ignored [74], which will result into

$$\sigma_{13} = \sigma_{23} = \sigma_{33} = 0 \tag{13}$$

Plane Strain

For plane strain conditions, the deformations in the third direction are neglected [74], which will result into

$$\sigma_{13} = \sigma_{23} = 0 \tag{14}$$

$$\varepsilon_{13} = \varepsilon_{23} = \varepsilon_{33} = 0 \tag{15}$$

DISCRETIZATION OF THE EQUATIONS

Integrating equation (12) over a control volume v_P with ∂v_P as the control surface and applying the Gauss' theorem will result in:

$$\int_{v_P} \frac{\partial(\rho V)}{\partial t}\, dv_P - \oint_{\partial v_P} \mathrm{ds} \cdot [\mu \nabla U + \mu(\nabla U)^T + \lambda \boldsymbol{I}\, tr(\nabla U)] + \int_{v_P} \rho\, C\, V\, dv_P = \int_{v_P} \rho f dv_P \tag{16}$$

Temporal Derivative

The temporal derivative is calculated using properties at *n+1*, *n*, and *n-1* time levels

$$\int_{v} \frac{\partial(\rho V)}{\partial t}\, dv_P = \rho v_P \left[\frac{3V^{n+1} - 4V^{n} + V^{n-1}}{2\Delta t}\right] \tag{17}$$

where $\mathrm{V}^{\mathrm{n+1}} = \mathrm{V}(\mathrm{t} + \Delta\mathrm{t})$, $\mathrm{V}^{\mathrm{n}} = \mathrm{V}(\mathrm{t})$ and $\mathrm{V}^{\mathrm{n-1}} = \mathrm{V}(\mathrm{t} - \Delta\mathrm{t})$.

Spatial Discretization

The continuous surface integrals in equation (16) are split into the sum of integrals over the cell faces

$$\oint_{\partial v_P} \mathrm{ds} \cdot [\mu \nabla U + \mu (\nabla U)^T + \lambda \, \boldsymbol{I} \, tr(\nabla U)] = \sum_{faces} s \cdot [\mu \nabla U + \mu (\nabla U)^T + \lambda \, \boldsymbol{I} \, tr(\nabla U)] \tag{18}$$

where $\boldsymbol{s}$ is the face area vector.

By assembling all terms and performing the integration, we get

$$\rho v_P \left[\frac{3V^{n+1} - 4V^n + V^{n-1}}{2\Delta t}\right] - \sum_{faces} s. [\mu \nabla U + \mu (\nabla U)^T + \lambda \, \boldsymbol{I} \, tr \, (\nabla U)] + \rho v_P (C \, V^n - f) = 0 \tag{19}$$

The term ∇U in equation (19) could be evaluated based on the current (available) solution U^n; in this case the solution is said to be *explicit*, or it could be evaluated based on the next iteration solution U^{n+1}, in this case the solution is said to be *implicit* [52].

Dual Time-Stepping Scheme

To achieve a matrix-free operation and to use a larger time step size, a dual time-stepping scheme is adopted [8, 10] by adding a pseudo time derivative term to the left hand side of equation (19). This dual-time-stepping procedure is independent of the spatial discretization approach employed, i.e. finite volume or finite element [75, 76]. After adding the pseudo term into equation (19) will be:

$$\rho v_P \frac{dV^n}{d\tau} = \sum_{faces} s. [\mu \nabla U + \mu (\nabla U)^T + \lambda \, \boldsymbol{I} \, tr \, (\nabla U)] - \rho v_P \left[\frac{3V^{n+1} - 4V^n + V^{n-1}}{2\Delta t}\right] - \rho v_P (C \, V^n - f) \tag{20}$$

By dividing both sides by the mass of the control volume and grouping the spatial terms together the equation could be re-written as follows:

$$\frac{dV^n}{d\tau} = \frac{1}{\rho v_P}\sum_{faces} s.[\mu\nabla U + \mu(\nabla U)^T + \lambda\, \boldsymbol{I}\, tr\,(\nabla U)] - \left[\frac{3V^{n+1} - 4V^n + V^{n-1}}{2\Delta t}\right] - C\,V^n + f$$
$$= \mathcal{R}(V^{n+1}) \tag{21}$$

In order to retrieve the displacement from the calculated velocity the following relation is used

$$V = \frac{\partial U}{\partial t} \tag{22}$$

Similarly, by adding a pseudo time derivative term and using properties at *n+1*, *n*, and *n-1* time level for approximating the temporal derivative and grouping the spatial terms together, equation (22) can be re-written as

$$\frac{\partial U}{\partial \tau} = V^n - \left[\frac{3U^{n+1} - 4U^n + U^{n-1}}{2\Delta t}\right]$$
$$= H(U^{n+1}) \tag{23}$$

Then the solution is sought by marching in τ to a pseudo steady state. We adopt a four-stage Runge-Kutta scheme for integrating equation (21) and equation (23) for stability and convergence.

$V^{(0)} = V^{(n,m)}$	$U^{(0)} = U^{(n,m)}$
$V^{(1)} = V^{(0)} + \alpha_1\,\Delta\tau\,\mathcal{R}^{(0)}$	$U^{(1)} = U^{(0)} + \alpha_1\,\Delta\tau\,H^{(0)}$
$V^{(2)} = V^{(0)} + \alpha_2\,\Delta\tau\,\mathcal{R}^{(1)}$	$U^{(2)} = U^{(0)} + \alpha_2\,\Delta\tau\,H^{(1)}$
$V^{(3)} = V^{(0)} + \alpha_3\,\Delta\tau\,\mathcal{R}^{(2)}$	$U^{(3)} = U^{(0)} + \alpha_3\,\Delta\tau H^{(2)}$
$V^{(4)} = V^{(0)} + \alpha_4\,\Delta\tau\,\mathcal{R}^{(3)}$	$U^{(4)} = U^{(0)} + \alpha_4\,\Delta\tau\,H^{(3)}$
$V^{(n,m+1)} = V^{(4)}$	$U^{(n,m+1)} = U^{(4)}$

where the superscript *m* denotes the pseudo time level and the coefficients for the four-stage Runge-Kutta time integration are taken as

$$\alpha_1 = 0.15, \quad \alpha_2 = 0.3275, \quad \alpha_3 = 0.57, \quad \alpha_4 = 1.0$$

Pseudo Time Step Calculation

The dual-time stepping procedure is a conditionally stable scheme since it is explicit in pseudo-time, which means maximum pseudo-time step size, $\Delta\tau$, is limited. However, the procedure is implicit in real-time, thus the scheme is stable for any choice of the real-time step size Δt.

To ensure solver stability, the pseudo time step must be less than the critical pseudo time step which is defined by the following expression [75]

$$\frac{L}{\Delta\tau_{cr}} = \sqrt{\frac{K}{\rho}} + \sqrt{\frac{\mu}{\rho}} \tag{24}$$

where *L* is the effective length scale of the mesh spacing and *K* is the bulk modulus which is defined as

$$K = \frac{E}{3(1-2\nu)} \tag{25}$$

Physical Damping Coefficient

Damping is the dissipation of energy in an oscillating system. In cases where the static state solution is to be determined, physical damping must be added. Based on the assigned value for the damping coefficient, the motion of the system will be termed as underdamped, overdamped, or critically damped.

$$\text{if C} \quad \begin{cases} < C_{cr} & \text{Underdamped} \\ = C_{cr} & \text{Crtically damped} \\ > C_{cr} & \text{Overdamped} \end{cases} \tag{26}$$

where C_{cr} is the critical damping coefficient and is defined as

$$C_{cr} = 2\ M\ \omega_n \tag{27}$$

where M is the mass per unite length and ω_n is the natural frequency of the structure.

$$\omega_n = \alpha_n \sqrt{\frac{EI}{ML^4}} \tag{28}$$

where I is the area moment of inertia and L is the length span. The value of α_n is based on the support condition, i.e. 1.57 for the pinned supports, 2.45 for fixed/pinned supports, 3.56 for fixed both ends and 0.56 is for fixed/free (cantilever) ends [77].

FORCES ON INTERNAL FACES

The forces acting on an internal face are calculated based on the gradients of the displacement of this face. The gradients of the displacement at any internal face are calculated based on the neighboring cells gradients; see Figure 4, as the following

$$\nabla U_i = \left[\overline{\nabla U}_{Pj} - \left(\overline{\nabla U}_{Pj} \cdot \frac{\vec{s}}{\|\vec{s}\|}\right)\frac{\vec{s}}{\|\vec{s}\|}\right] + \left[\left(\frac{U_j - U_P}{\|\vec{s}\|}\right)\frac{\vec{s}}{\|\vec{s}\|}\right] \tag{29}$$

where

$$(\overline{\nabla U})_{Pj} = \frac{1}{2}\left((\nabla U)_P + (\nabla U)_j\right) \tag{30}$$

where $\vec{s}$ is the vector pointing from the centroid of cell P to the centroid of cell j, as shown in Figure 4.

The force acting on the internal face can be calculated using the relation

$$\vec{F}_i = \vec{S}_b.[\mu \nabla U_i + \mu(\nabla U_i)^T + \lambda \, \boldsymbol{I} \, tr \, (\nabla U_i)] \tag{31}$$

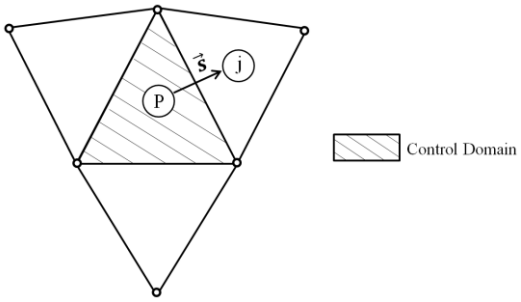


Figure 4. Sample 2D grid showing the cell which the gradient is being evaluated for, cell p, and its nearest neighbors.

FORCES ON BOUNDARY FACES

Fixed Displacement Boundary

This condition specifies the value of U_b on the boundary face. The necessary face gradient is computed using the cell center value in the neighboring cell.

$$\nabla U_{face} = \nabla U_{cell} - [\nabla U_{cell} \cdot \vec{n}]\vec{n} + \left[\left(\frac{U_b - U_{cell}}{\|\vec{d}\|}\right)\right]\vec{n} \tag{32}$$

Where $\vec{n}$ is the unit surface normal, $\|\vec{d}\|$ is the distance between the cell-center and the face-center, and U_b is the displacement at the boundary face.

The force acting on the surface can be calculated using the relation

$$\vec{F}_d = \vec{S}_b.\left[\mu \nabla U_{face} + \mu\left(\nabla U_{face}\right)^T + \lambda \, \boldsymbol{I} \, tr \, \left(\nabla U_{face}\right)\right] \tag{33}$$

Specified Traction Boundary

The traction boundary condition specifies the force on the boundary face in terms of the pressure and the shear stress. The force acting on the surface can be calculated using the relation

$$\vec{F}_t = \vec{S}_b.[\mu\nabla U + \mu(\nabla U)^T + \lambda\, \boldsymbol{I}\, tr\,(\nabla U)] = |\vec{S}_b|\vec{t} - \vec{S}_b\, p \tag{34}$$

Where $\vec{F}_t$ is the traction force acting on the boundary, $\vec{S}_b$ is the outward pointing boundary face area vector, $\vec{t}$ is the specified traction, and p is the pressure.

TOTAL FORCES

By combining the forces from all type of faces, equations (31), (33), and (34), equation (21) can be written as

$$\frac{dV^n}{d\tau} = -\left[\frac{3V^{n+1} - 4V^n + V^{n-1}}{2\Delta t}\right] - \frac{1}{\rho v_P}\left[\sum_i \vec{F}_i + \sum_d \vec{F}_d + \sum_t \vec{F}_t\right] + CV^n - f \tag{35}$$

CELL-CENTER DISPLACEMENT GRADIENTS

The displacement gradient at the cell-center is calculated based on the Green-Gauss theory

$$\nabla U_{cell} = \left[\sum_{faces} \vec{n}.U_{face}dA\right] \Big/ v_P \tag{36}$$

Where U_{face} is the face displacement value, which is calculated based on a simple averaging of the displacement values at the face nodes, dA is the face area, and v is the cell volume.

In the current cell-center finite volume scheme the displacement values are being calculated at the center of each control volume. In order to transfer the displacement values from the control volume centers to the control volume nodes, a weighted averaging interpolation is used as follows [45]

$$U_{node} = \left[\sum_{cells} w_{nc}(U_{cell} + \vec{r}_{nc} \cdot \nabla U_{cell})\right] \Big/ \sum_{cells} w_{nc} \tag{37}$$

where U_{node} is the node displacement value, U_{cell} is the cell displacement value, $\vec{r}_{nc}$ is the vector pointing from the cell center to the designated node, ∇U_{cell} is the displacement gradients evaluated at the previous time step, the summation is executed over all cells sharing the designated node, w_{nc} is the weighting coefficient which is inversely proportional with the distance between the cell center and the node

$$w_{nc} = \frac{1}{|\vec{r}_{nc}|} \tag{38}$$

CONVERGENCE CRITERIA

Convergence is achieved when no further change in solution is being recorded. This condition is implemented by calculating the L2 norm of the residuals at the physical time level and the pseudo time level. The L2 norm is calculated based on the change in displacement difference, ΔU, within all elements, as follow

$$\|L_2\| = \sqrt{\sum_{cells} \left(\Delta {U_x}^2 + \Delta {U_y}^2 + \Delta {U_z}^2\right)} \tag{39}$$

Where at the physical time level $\Delta U = U^{t+\Delta t} - U^t$ and at the pseudo time level $\Delta U = U^{\tau+\Delta\tau} - U^\tau$

When L2 norm for the pseudo time level falls below a pre-defined tolerance the inner pseudo time iterations are considered converged and the code marches to the next physical time step. At this stage the displacement at the physical time level, $U^{t+\Delta t}$, is set equal to the converged displacement at the pseudo time level, $U^{\tau+\Delta\tau}$. Before starting the next physical time step, the L2 norm for the physical time level is calculated and compared against a pre-defined tolerance as well. If it is found to be less than this tolerance, the solution is considered fully converged and the simulation is stopped.

TEST CASES

The developed methodology has been tested using a three-dimensional fixed-free cantilever beam for which an exact analytical solution exists. Three different types of loading have been used: 1) suddenly applied uniform distributed pressure, 2) suddenly applied axial tension pressure, 3) suddenly applied traction force. For each case, the dynamic (undamped) solution and the static (damped) solution have been analyzed.

The density ρ and Young's modulus of elasticity E are selected to obtain the beam's first natural frequency (f_n) as 1 Hz. The beam has a length of 2.0 m with a cross section of 0.2 x 0.2 m^2, as shown in Figure 5. Table 1 lists the material and geometrical specifications of the beam. Note that the gravity effect has not been taken into account.

In order to demonstrate the capability of the developed solver to handle any mesh type, an unstructured mesh with mixed elements has been generated for the cantilever beam. The generated unstructured mesh, shown in Figure 5, consists of 20,346 tetrahedral cells.

Table 1

Cantilever beam material specifications

Beam specification	**Value**
Density	1000 kg/m^3
Modulus of elasticity	15.293 MPa
Poisson's ratio	0.3
Natural Frequency	1 Hz
Area moment of inertia	$1.333x10^{-4}$ m^4

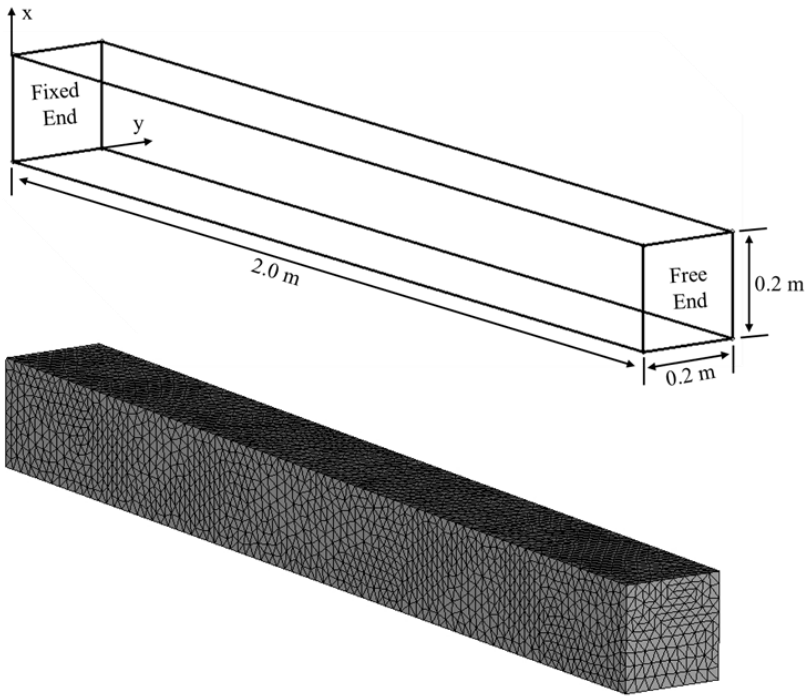


Figure 5. Schematic of the cantilever beam dimensions (top) and the generated unstructured mesh (bottom).

The physical time step was set to $1x10^{-3}$ seconds and the pseudo time step was set to $1x10^{-5}$ seconds which satisfies the solver stability condition. Within each physical time step a maximum of 300 pseudo iterations were performed. The convergence tolerance for the pseudo iteration residuals was $1x10^{-7}$. For all cases, the full load was suddenly applied

at t = 0 seconds and kept till the end of the simulation. Moreover, the structure was assumed to be initially stationary.

3D Cantilever Beam under Uniform Distributed Load

A uniform distributed pressure has been added as shown in Figure 6. The static solution for this type of problem is determined analytically to be

$$U_{max} = \frac{P_d A_p L^4}{8EI} \tag{40}$$

Where A_p is the area on which the uniform pressure, P_d, is applied.

The applied uniform pressure, P_d, value has been selected to be 510.0 Pa, which would result in a maximum static deflection of 0.1 m at the free end of the beam.

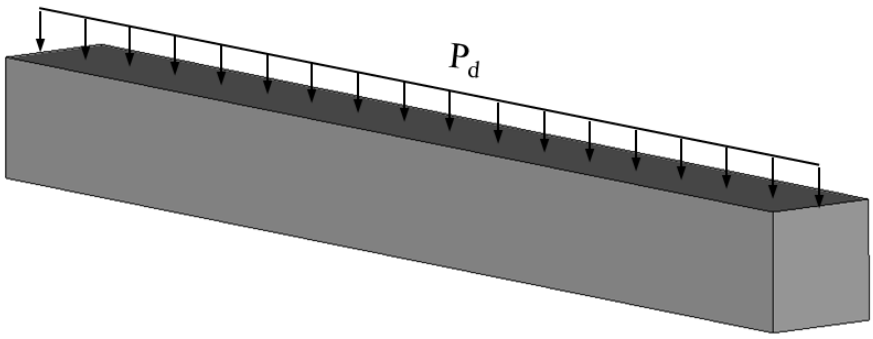


Figure 6. Schematic of the cantilever beam under a distributed pressure load.

First, the dynamic (undamped) response was simulated. Based on the analytical solution and the calculated natural frequency, the maximum dynamic deflection at the free end of the beam should be 0.2 m and should take place at t = 0.5 seconds. Figure 7 shows a comparison between the finite element approach and the proposed finite volume approach of the predicted dynamic response at the center of the beam free end.

Table 2 lists the deflection and frequency error percentages associated with using the finite volume approach. The finite volume approach shows a good agreement with the finite element solution with a maximum deflection of 0.196 m at t = 0.51 seconds and a natural frequency of 0.992 Hz. This represents a 2% amplitude error and a 1.6% frequency shift. Second, the static state solution was obtained using the proposed finite volume approach. Using a damping coefficient, *C*, of 3 N.m/s, the simulation converges at a maximum of 0.0995 m deflection at the free end of the cantilever beam, as shown in Figure 8.

Table 2

Uniform distributed load deflection and frequency comparison

	Dynamic Analysis		**Static Analysis**
	U_{max} (m)	**Frequency (Hz)**	**U_{steady} (m)**
Finite Element	0.2	1.0	0.1
Finite Volume	0.196	0.992	0.0995
Error %	2.0%	1.6%	0.05%

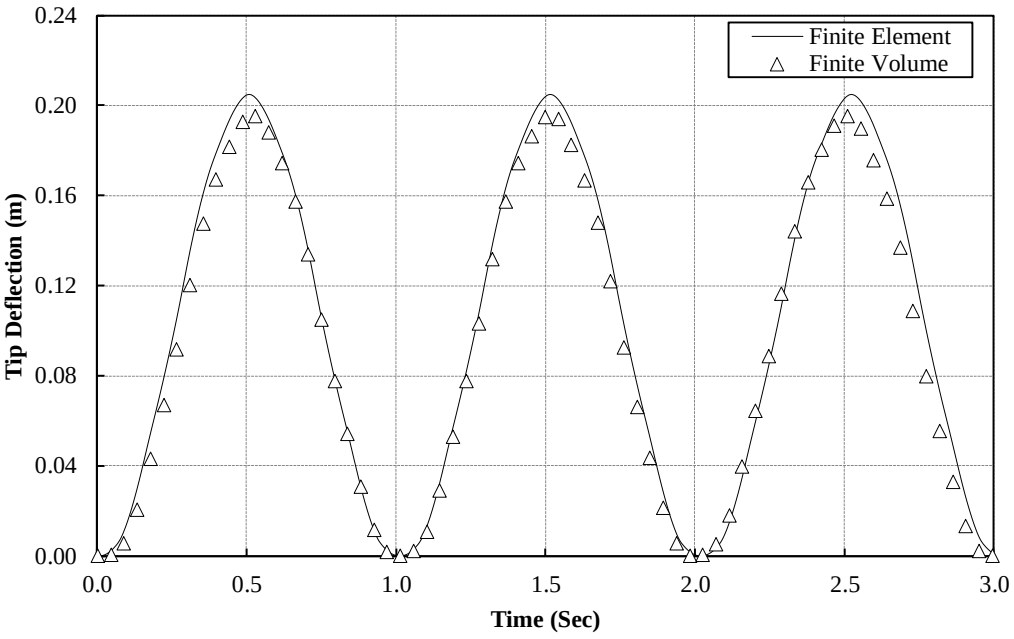


Figure 7. Dynamic response of the cantilever beam under uniform distributed pressure at the center of the free end.

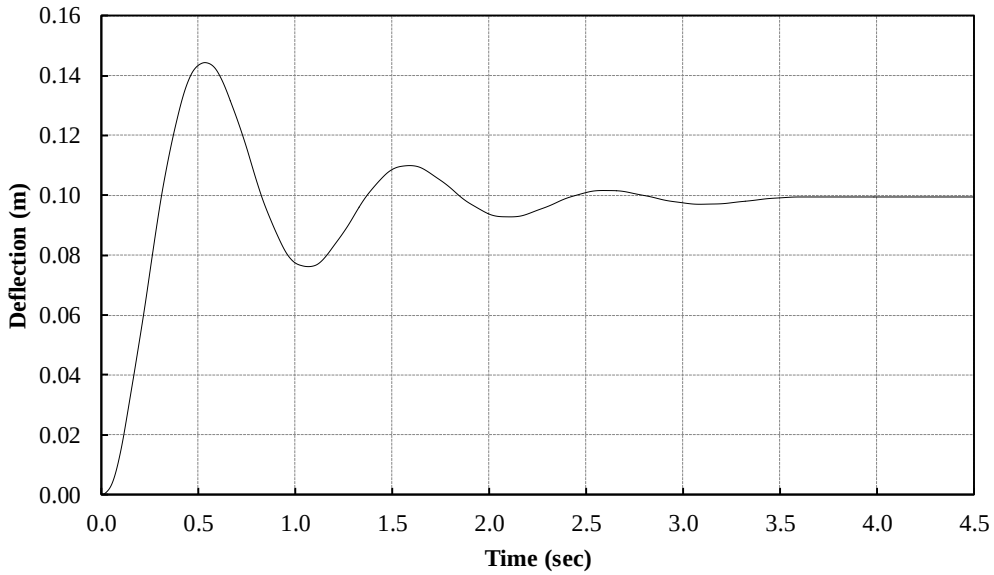


Figure 8. Static response of the cantilever beam under uniform distributed pressure at the center of the free end with C = 3 N.m/s.

The deflection at any section across the beam length can be determined analytically as

$$U_x = \frac{P_d A_p z^2}{24EI}(z^2 + 6L^2 - 4Lz) \tag{41}$$

Where z represents the location across the beam length (z-direction).

Figure 9 shows the computed steady state beam shape, which illustrates consistent deflections throughout the beam length (z-direction). Also Figure 9 shows a comparison between the analytical and finite volume deflections across the beam length. It can be seen from the figure that the error in the deflection across the length is minimal close to the fixed end of the beam and increasing in the direction of the free end. However, the maximum error at the free end is 0.05%.

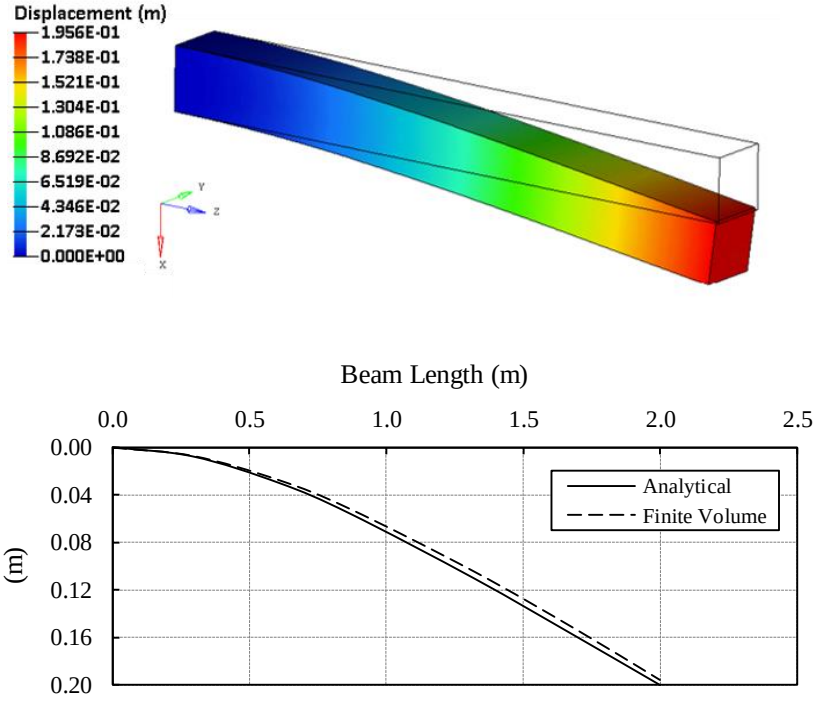


Figure 9. Steady state cantilever shape under uniform distributed load (top) and deflection values across the beam length (bottom).

3D Cantilever Beam under Axial Tension Pressure

A normal pressure has been added to the free end of the cantilever beam, as shown in Figure 10. The static solution for this type of problem predicts a maximum deflection to be

$$U_{max} = \frac{PA_pL}{E} \tag{42}$$

The applied normal pressure, *P*, value has been selected to be 1911.1 Pa, which would result in a maximum static deflection of 2.5×10^{-4} m at the free end of the beam.

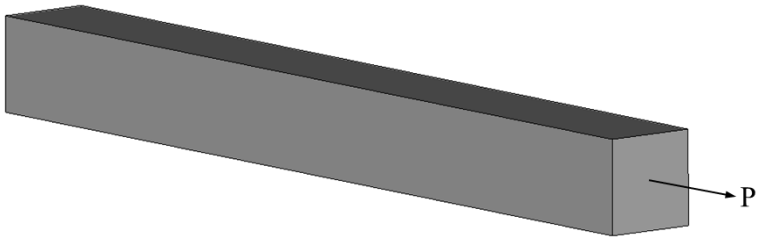


Figure 10. Schematic of the cantilever beam under an axial tension pressure

For the dynamic (undamped) response, the developed finite volume approach predicts an accurate response when compared to the response obtained by the finite element method. Table 3 lists the deflection and frequency error percentages associated with using the finite volume approach for dynamic and static analyses. The maximum deflections predicted by finite element and finite volume were 4.85×10^{-4} m and 4.57×10^{-4} m, respectively. While the predicted vibration frequencies were 15.0 Hz and 14.3 Hz, respectively. The dynamic responses at the center of the free end for both methods, finite element and finite volume, are shown in Figure 11. However, when damping is added the finite volume simulation under-predicts the maximum deflection by 6% at 2.35×10^{-4} m. The damped response at the center of the free end predicted by the finite volume approach is shown in Figure 12.

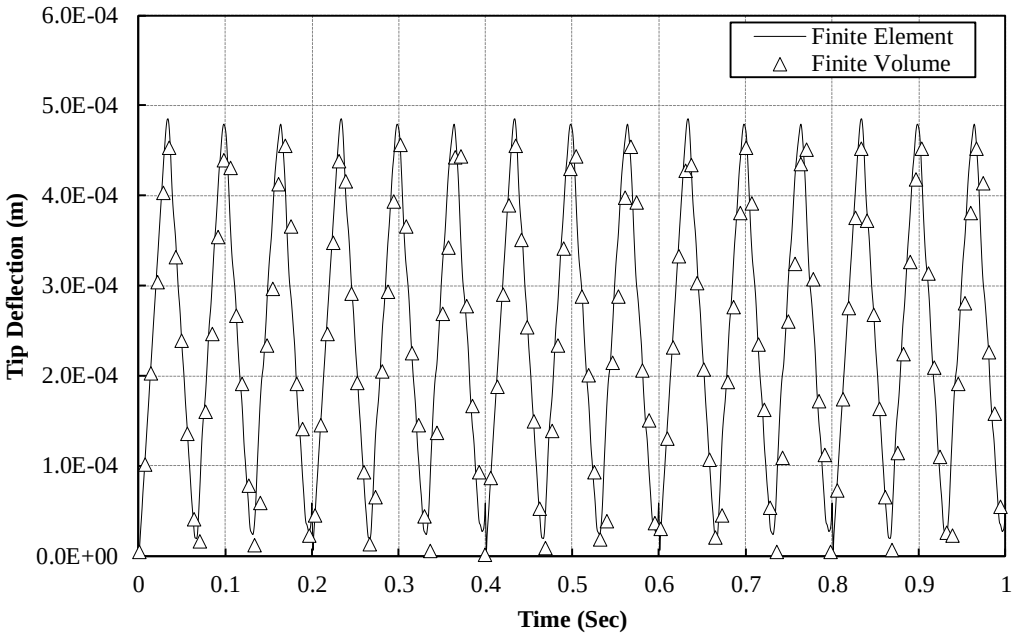


Figure 11. Dynamic response of the cantilever beam under axial tension pressure at the center of the free end.

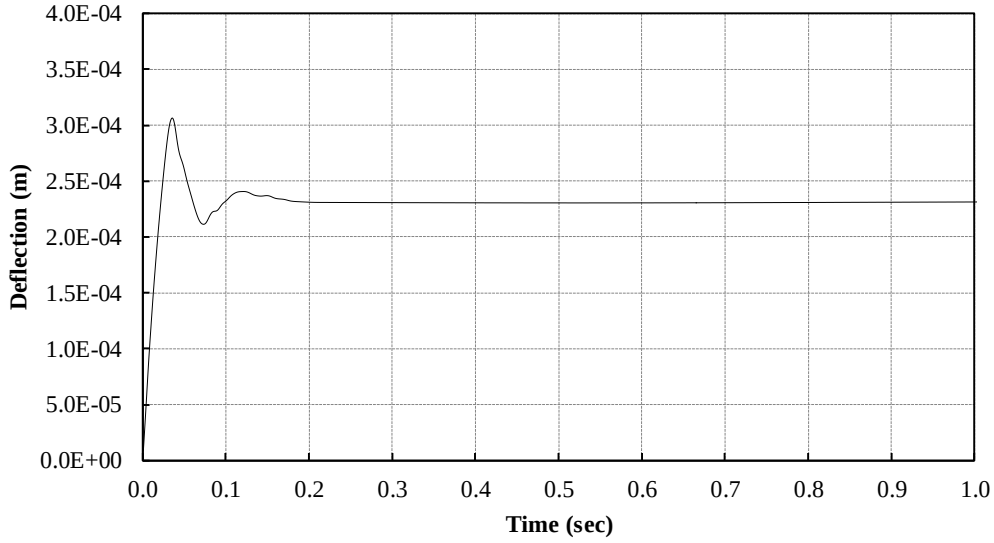


Figure 12. Static response of the cantilever beam under axial tension pressure at the center of the free end with C = 70 N.m/s.

Table 3

Axial Tension Load deflection and frequency comparison

	Dynamic Analysis		Static Analysis
	U_{max} (m)	**Frequency (Hz)**	**U_{steady} (m)**
Finite Element	4.85×10^{-4}	15.0	2.50×10^{-4}
Finite Volume	4.57×10^{-4}	14.3	2.35×10^{-4}
Error %	5.8%	4.7%	6.0%

3D Cantilever Beam under Traction Force

In this test case, a pure bending traction pressure was added to the free end of the cantilever beam as shown in Figure 13. For this case, the load is concentrated and not uniformly distributed along wide area. For this reason, it is expected that the maximum amplitude to be less than the load calculated using equation (40).

The applied traction pressure, *P*, value has been selected to be 1911.1 Pa. The maximum deflection was calculated using finite element analysis and determined to be 0.196 m.

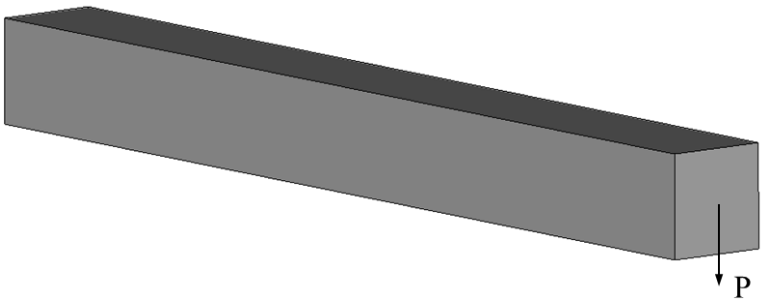


Figure 13. Schematic of the cantilever beam under a traction pressure.

As shown in Figure 14 and Figure 15, the finite volume based dynamic and static responses show a good agreement when compared to the finite element based response. Table 4 lists the deflection and frequency error percentages associated with using the finite volume approach for dynamic and static analyses. For the dynamic structural response, the finite volume solution predicted a maximum deflection and a vibration frequency of 0.187 m and 0.94 Hz, respectively. This represents a deflection error of 4.7% and a frequency error of 6%. For the static damped structural response, the finite volume solution predicted a steady state deflection of 0.092 m and the finite element predicted a deflection of 0.098 m. This represents an error of 6%. Using a smaller pseudo time tolerance and performing more sub-iterations might lead to a more accurate solution.

Table 4

Bending traction load deflection and frequency comparison

	Dynamic Analysis		**Static Analysis**
	U_{max} (m)	**Frequency (Hz)**	**U_{steady} (m)**
Finite Element	0.196	1.0	0.098
Finite Volume	0.187	0.94	0.092
Error %	4.7%	6.0%	6.0%

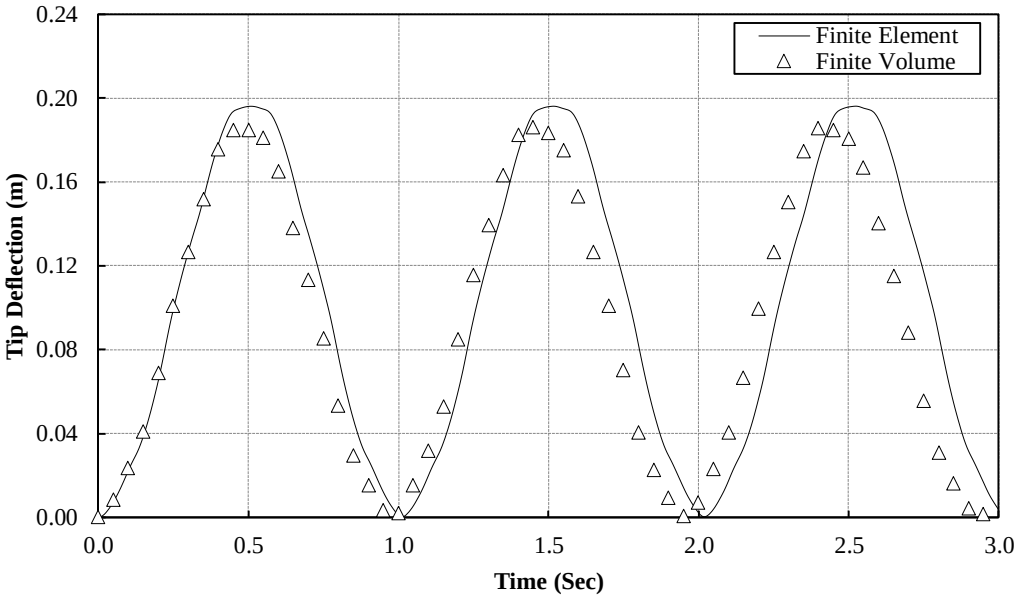


Figure 14. Dynamic response of the cantilever beam under traction pressure at the center of the free end.

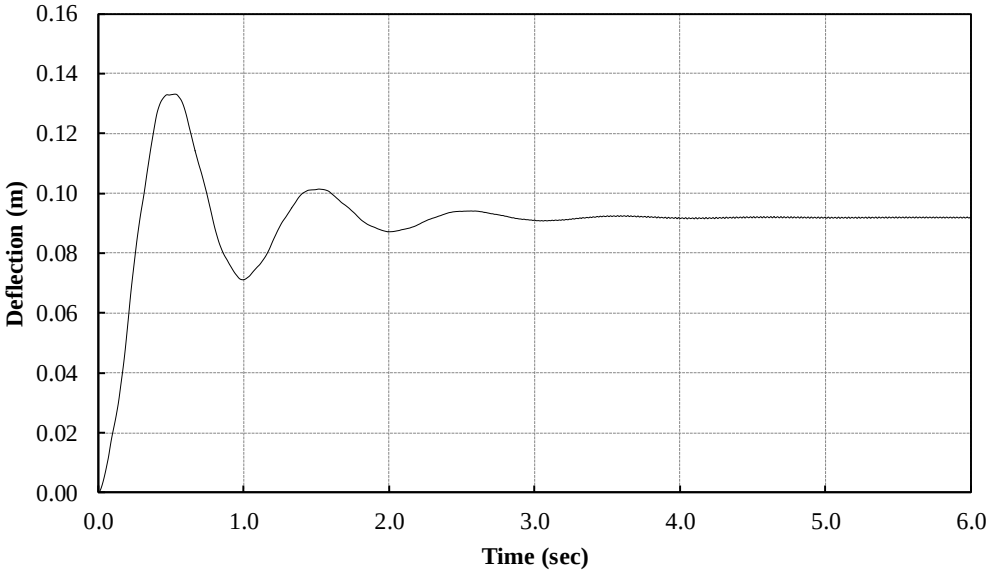


Figure 15. Static response of the cantilever beam under traction pressure at the center of the free end with C = 3 N.m/s.

Large Amplitude Deflections of a 3D Cantilever Beam

In this test case, a diagonal traction load is applied to the free end of the cantilever beam, as shown in Figure 16. Higher load magnitudes were used in order to test the developed approach for large deflections capabilities. The load, *P*, is specified at the faces of the free-end of the cantilever beam as a traction vector, $\vec{t} = \left(\frac{P}{\sqrt{2}}, \frac{P}{\sqrt{2}}, 0\right)$ N/m^2. The applied load, *P*, was determined based on the dimensionless traction force,

$$\mathcal{F} = \frac{PAL^2}{EI} \tag{43}$$

The maximum deflection was calculated at mid-point on the free-end of the beam as,

$$d_{max} = \sqrt{{u_x}^2 + {u_y}^2} \tag{44}$$

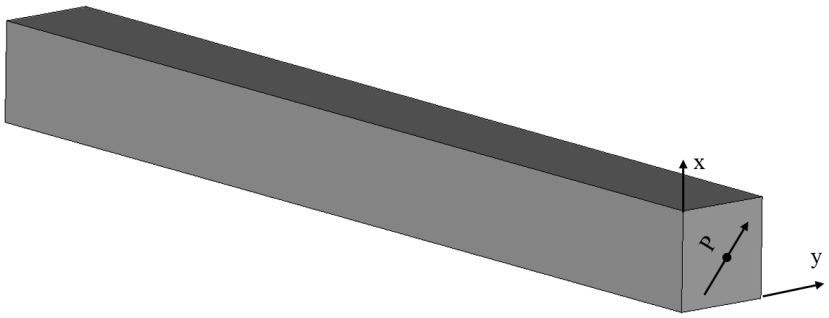


Figure 16 Schematic of the cantilever beam under diagonal traction load

In order to reach a steady state solution, a damped analysis has been performed. The calculated deflection was compared with the exact deflection calculated by Mattiasson [78] using the numerical evaluation of elliptic integrals. Figure 17 shows the steady state

shape of the beam colored by the deflection values. It can be seen from the figure that the predicted deflections are uniform throughout the beam length.

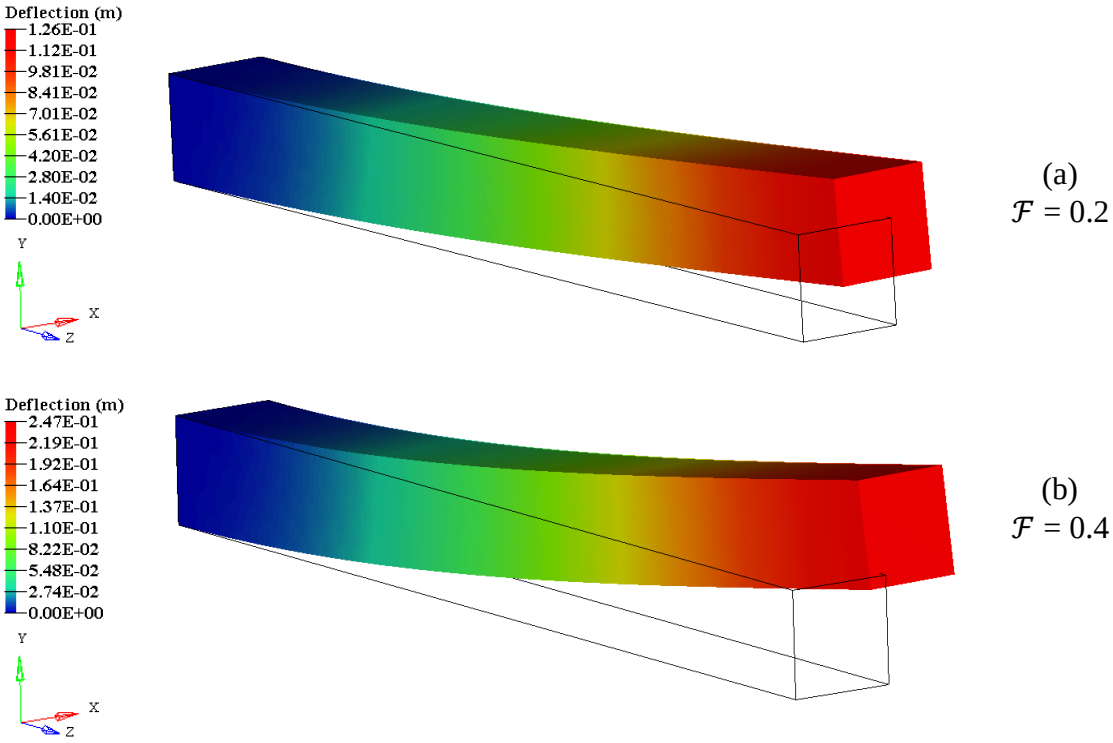


Figure 17 Steady State shape colored by deflection value for different loads.

Moreover, the dynamic response of the beam with no damping effect has been analyzed. Figure 18 shows the recorded deflection history at the free end of the beam. The simulation was run for 9 cycles to ensure the stability of the calculation over a long period of time, which is important for an FSI simulation. It is clear that the predicted dynamic response shows no effect of damping, which implies that there are no numerical dissipation associated with the applied numerical approach. Table 5 presents the applied loads specifications and the comparison between the exact numerical deflection and the

calculated deflection. It also compares the predicted dynamic frequency with the beam's natural frequency.

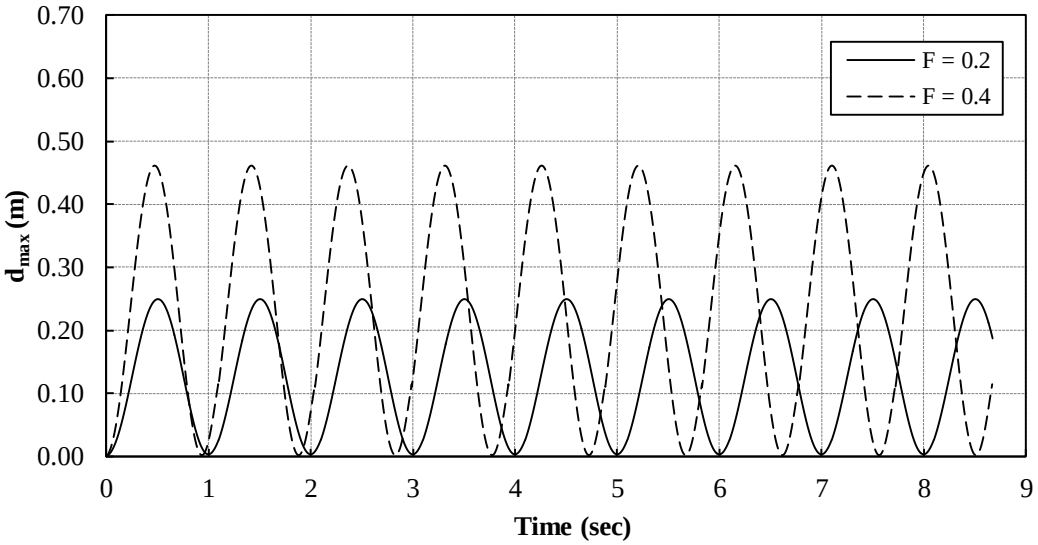


Figure 18 Dynamic response of the cantilever beam under diagonal traction load at the center of the free end.

Table 5

Relative error in steady state deflection and dynamic frequency

	$\mathcal{F}$ = 0.2		$\mathcal{F}$ = 0.4	
	d_{max} (m)	Frequency (Hz)	d_{max} (m)	Frequency (Hz)
Exact	0.13272	1.0	0.26196	1.0
Finite Volume	0.1255	1.01	0.2431	0.952
Error %	5.5%	1.0%	7.19%	4.8%

Mesh Sensitivity Analysis

In order to study the effect of the mesh refinement on the accuracy of the proposed method, five different unstructured meshes were generated, i.e. 270, 2100, 4000, 10000 and 20000 cells. Table 6 lists the relative errors for the dynamics solutions for the five different meshes. The dynamic (undamped) solution, for the cantilever beam under uniform distributed load of 510.0 Pa, shows that the accuracy of the finite volume solver increases with refining the mesh. However, for the 20000 cells mesh the maximum tip deflection recorded was 0.196 m, while the analytical solution indicates a maximum tip deflection of 0.2 m.

Table 6

Dynamic (undamped) solution's maximum deflection and relative error

	Mesh 1	**Mesh 2**	**Mesh 3**	**Mesh 4**	**Mesh 5**	**Analytical**
No. of Cells	270	2100	4000	10000	20000	--
Max. Deflection	0.1747	0.1862	0.1906	0.1918	0.1955	0.2
Relative Error %	12.7%	6.9%	4.7%	4.1%	2.2%	0.0%

By performing a Fourier transform, the frequencies recorded by the finite volume solver for each mesh were obtained. Figure 19 illustrates the increase in the solution's accuracy with the increase of mesh refinement. It is clear that the error in the solution's vibration amplitude and frequency kept decreasing with the increase of the number of cells. Increasing the number of cells allows for a more accurate estimation of the displacement gradients and accordingly a more accurate stresses to be calculated.

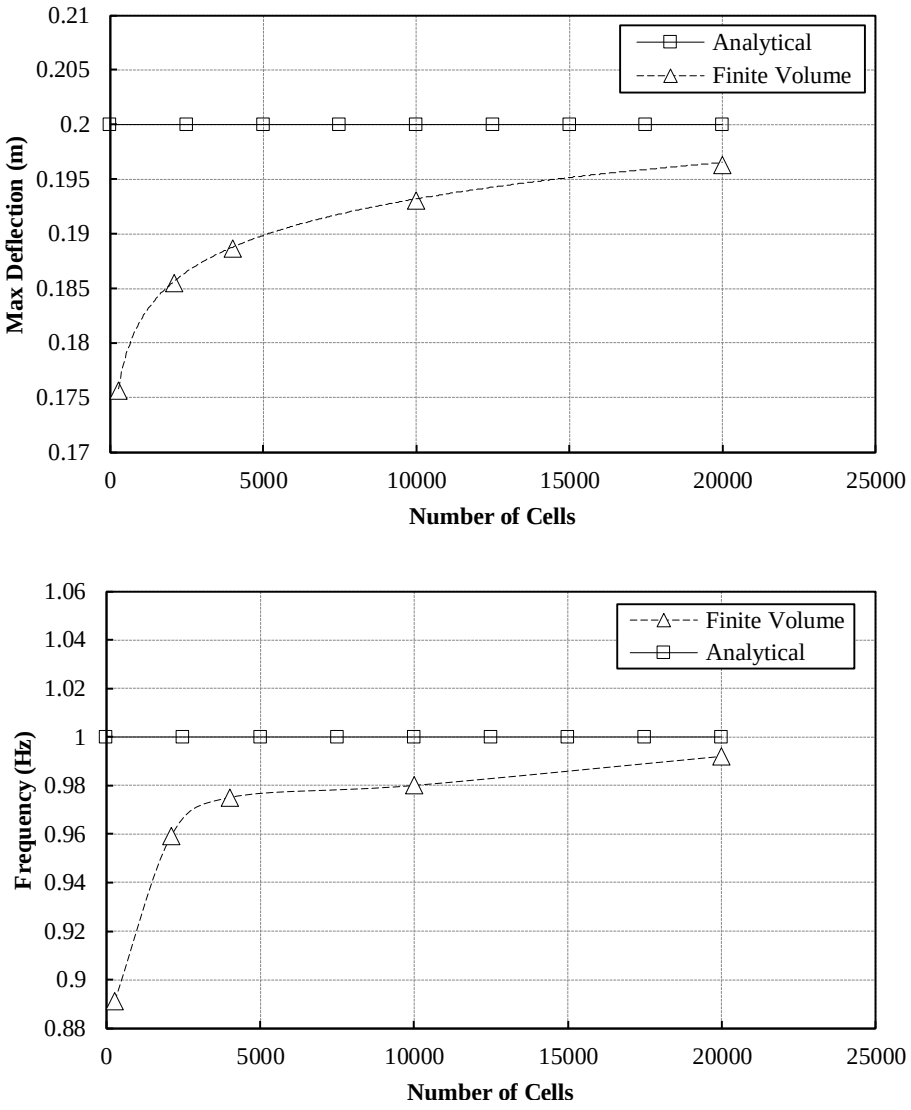


Figure 19. Mesh number of cells versus recorded maximum deflection (top) and frequency (bottom).

CPU Time

The current implementation of the finite volume structural solver is in serial. A CPU time analysis was performed for the investigated test cases. In these test cases a 20,000 unstructured tetrahedral mesh was used. It was found that the average CPU time required to perform one time step with 300 pseudo inner iterations is 48.6 seconds. This CPU time is considered to be relatively high. By analyzing the solver performance, it was

found that the step of calculating the gradient is consuming around 90% of the total CPU time.

The Green-Gauss method is used for calculating the gradients of the displacement. This method requires the displacement values to be known at the center of each face. Since the displacement values are being calculated at the center of the control volume, it was essential to use a weighted averaging interpolation to transfer the displacement values to the nodes, then transfer the nodal values into the center of the face. The Green-Gauss method is more suitable for CFD solvers where the number of inner iterations required is low, typically 3 to 5 [52]. While in the developed CSD methodology, the number of inner iterations required is of order of hundreds. For the test cases investigated in this chapter, the CSD solver performed 300 inner iteration to reach a convergence tolerance of $1x10^{-7}$, as shown in Figure 20.

Thus, optimizing the Green-Gauss theory implementation within the code or replacing it with an alternative less expensive method, e.g. least square fitting is essential. Due to time limitations, it was decided to consider this step as part of the future work.

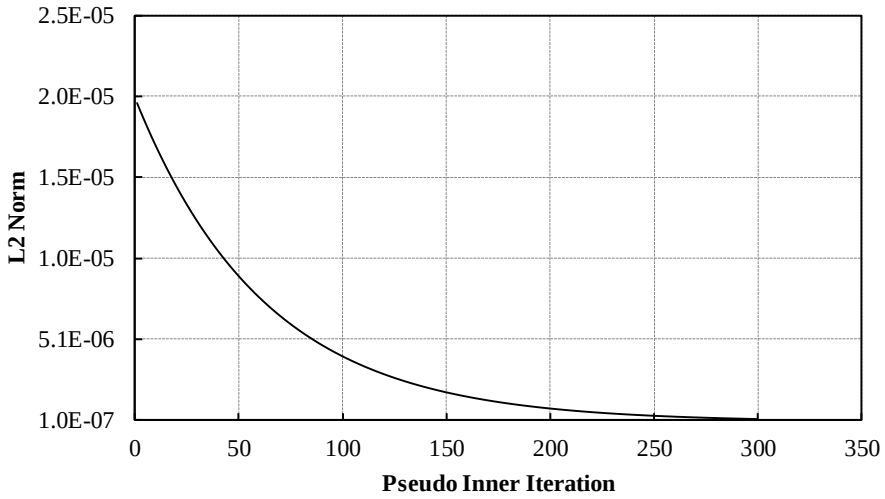


Figure 20 Convergence history of L2 Norm versus pseudo inner iteration.

CONCLUSIONS

A second order cell-centered finite volume approach to model three-dimensional linear elastic structures was developed and investigated. The case of a 3D cantilever beam under three different loading conditions was tested. The obtained results were compared against the traditional finite element method and against analytical solutions. The proposed finite volume methodology results show a good agreement. Furthermore, the developed solver was tested for large amplitude deflection cases. The load was applied as a diagonal traction load acting on the free-end of the beam. Two different dimensionless load magnitudes have been applied. The finite volume results well-predicted the dynamic as well as the damped responses. The error in the deflection amplitude for the lower load and higher load were 5.5% and 7.19%, respectively, with frequency shifts of 1.0% and 4.8%, respectively.

Based on the obtained results, the proposed finite volume approach can be used as an alternative for the well-established finite element method. The developed finite volume methodology adapts the exact numerical discretization scheme applied in state-of-the art generalized mesh based CFD solvers. This allows for efficient FSI coupling without the need for data interpolation and will greatly improve the parallelization capabilities.

CHAPTER 4

OBJECTIVE 2: EXTENSIVE SURVEY OF MESH DEFORMATION TECHNIQUES

INTRODUCTION

This survey reviews the recent development of mesh deformation methods. During the past two decades a vast number of researches have been concerned with developing an efficient and robust mesh deformation technique. This has been achieved either by proposing a novel approach, improving an existing one, or by combining two existing approaches together resulting in a new hybrid approach. It is important to keep track of the most up to date developments in the field of mesh deformation, in order to allow the researchers to adopt the most efficient and application compatible mesh deformation scheme, as well as propose new methods of improvements. In this survey the mesh deformation techniques have been classified into two main categories, 1) physical analogy based techniques and 2) interpolation based techniques.

The numerical simulation of dynamically updated three-dimensional (3D) meshes arises in many engineering applications, such as moving boundary problems [79], bio-fluid mechanics problems [80], free surface flows, and Fluid–Structure Interaction (FSI) problems. FSI are of great importance in many real-life applications, such as industrial processes, aero-elasticity, and bio-mechanics. In such applications, when the flow domain boundary undergoes a motion, the most common approach is to conform the fluid mesh to confine the changing flow domain. This can be achieved either by deleting the old mesh and regenerating a new mesh or by dynamically deforming the mesh. For applications that

require updating the mesh at every time step, regenerating a new mesh consumes high CPU cost and requires special mesh quality controls which makes it impractical approach [81-83]. Moreover, mapping the solution from the old mesh to the new mesh consumes extra CPU cost. Therefore, the mesh deformation option is more practical and flexible.

Various mesh deformation methodologies have been proposed. Some of them are robust with respect to elements overlapping and crossing but very time consuming, while others are computationally efficient but less robust. Since 1980 it has been a challenge to develop an efficient and robust mesh deformation technique. However, various types of simulations have different mesh deformation requirements. The simplest problem is when an object undergoes a translational or rotational motion, followed by an object experiencing both translational and rotational motions; then the most complicated is when the problem involves higher frequency components or multiple objects motions. Moreover, handling structured grids is easier than unstructured grids and dealing with small deformations is obviously simpler than large deformations. Also, viscous flows need a special treatment for preserving the quality of the boundary layer mesh, whereas the mesh layers are tightly packed and needs to move rigidly with the boundary surfaces.

Strategies for deforming the fluid mesh conforming to the deformation of solid body can be divided into two basic classes: physical analogy or interpolation. The physical analogy approach describes the fluid mesh deformation according to a physical process that can be modeled using numerical methods. In the interpolation based approaches, an interpolation function is used to transfer prescribed boundary point displacements to the fluid mesh. Figure 21 shows the reviewed approaches of each of these classes which are discussed below in detail.

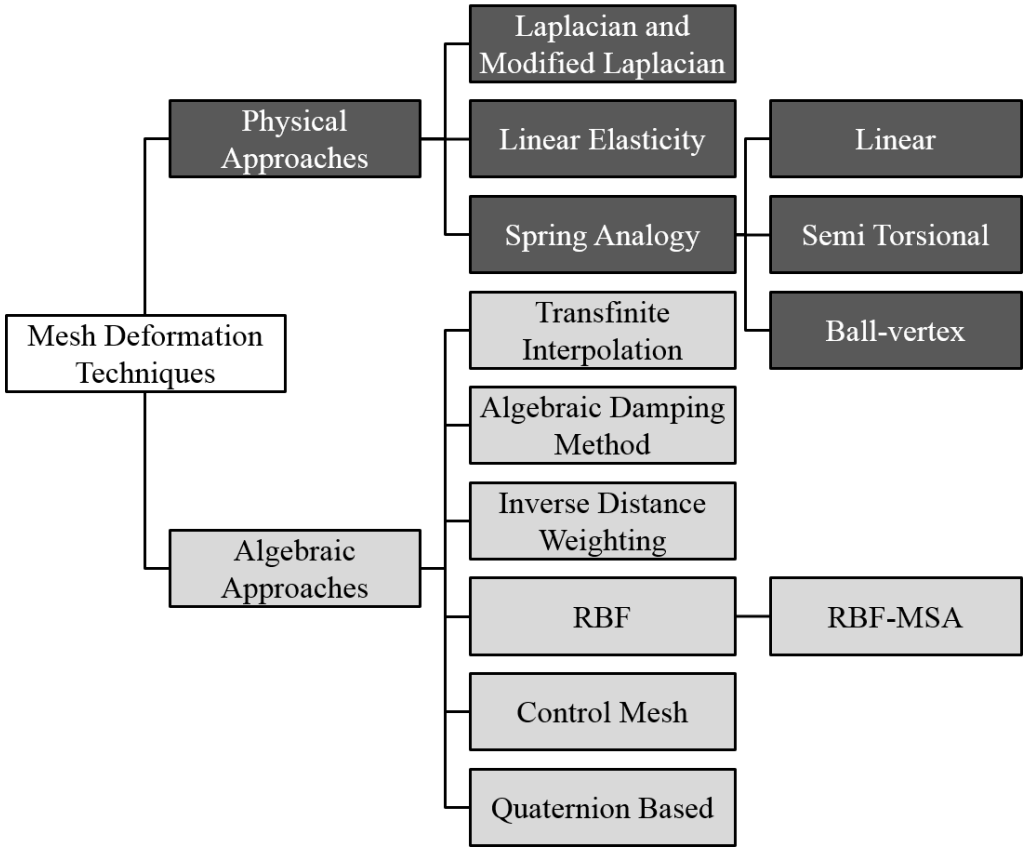


Figure 21 Reviewed mesh deformation techniques.

MESH DEFORMATION USING PHYSICAL ANALOGY

Techniques lying under this category are based on spring analogy or solutions of partial differential equations. The main drawback of physical analogy methods is that they involve large systems of equations, implying a higher computational cost. Besides, these methods require grid connectivity information which results in more storage requirements and difficulties in parallelization.

Linear Spring Analogy

One of the popular methods in this class is the tension spring analogy developed by Batina [84]. In this approach, each edge of the mesh is replaced by a tension spring with the spring stiffness is taken as inversely proportional to the edge length. Many researchers have adopted the spring analogy and also used the same assumption for the stiffness [85-87]. In this method, the equilibrium lengths of the springs are set equal to the initial lengths of the edges. By applying Hook's law to the nodes displacements, the force is written as

$$\vec{F}_i = \sum_{j=1}^{n_i} \alpha_{ij}\left(\vec{\delta}_j - \vec{\delta}_i\right) \tag{45}$$

where α_{ij} is the stiffness of the spring between node i and j, $\vec{\delta}$ is the node displacement and n_i is the number of neighbors of node i. For static equilibrium, the force at every node i has to be zero. The iterative equation to be solved is

$$\vec{\delta}_i^{k+1} = \frac{\sum_{j=1}^{n_i} \alpha_{ij}\vec{\delta}_j^{k}}{\sum_{j=1}^{n_i} \alpha_{ij}} \tag{46}$$

where the known displacements at the boundaries are used as the boundary conditions.

After iteratively solving equation (46), the nodes coordinates are updated by adding the final displacement to them.

Blom [88] analyzed the stiffness of the springs of a one-dimensional linear spring system. He proved that by setting the stiffness equal to the inverse of the edge length the nodes are prevented from colliding when they are placed on a line and move along this line. In other words, this stiffness choice prevents the cells from colliding if they are placed on the same axis and moving across this axis. However, for triangular grids it is possible

for the triangle edges to rotate and cross each other, see Figure 22. In order to prevent this, Farhat et al. [89] proposed the use of torsional springs that are placed in the corner between adjacent edges. An alternative solution is to divide the edge stiffness by the angle formed by the other two edges in the triangle. However, this will result in a non-linear system of equations, since α_{ij} will become a function of the displacement. This approach is usually referred to as the *semi-torsional spring* approach.

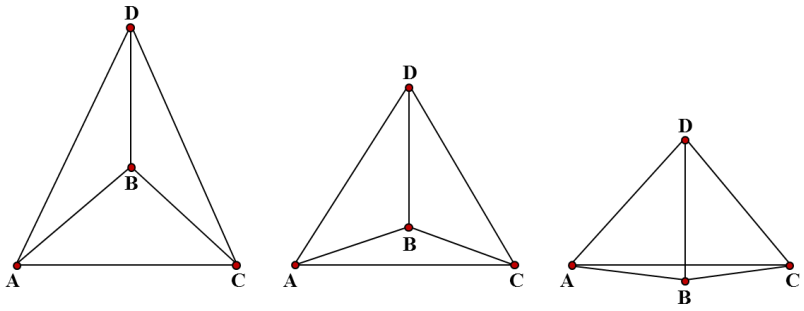


Figure 22. Negative areas produced by linear spring method [90].

Modified Spring Analogies

In order to prevent the element inversion problem associated with the linear spring analogy, multiple modifications have been proposed. The most influential modifications are discussed in this section. These modifications are the *torsional spring* [89], the *semi-torsional spring* [88], the *ball-vertex* [91], and the *Ortho-Semi-Torsional (OST) spring* approach [92].

Torsional Spring Method. To prevent element inversion on a 2D triangular mesh, three torsional springs where attached, one at each vertex of the triangle and their stiffness coefficients were depended on the angle θ of the corresponding vertex in the triangle. The stiffness of the torsional spring is given by

$$\alpha_A^{ABC} = \frac{1}{\sin^2 \theta_A^{ABC}} \tag{47}$$

where the subscript *A* specifies the vertex on which the calculation applies, and the superscript *ABC* specifies the triangle which the vertex *A* belongs to.

This method was extended to 3D meshes by Degand and Farhat [93]. They constructed 12 triangles within each tetrahedron, where three triangles are constructed for each vertex. Consider the tetrahedron *ABCD*, to prevent the vertex D from penetrating through its opposite face *ABC*, three triangles can be constructed by projecting the vertex *D* on each of the edges forming the face ABC. The first triangle can be constructed as follow, consider the projection of the vertex *D* on the edge AB as node *X*, then the first triangle is *DXC*, as shown in Figure 23. Similarly, consider the projection of the vertex *D* on the edges *AC* and *BC* as node *Y* and node *Z* respectively. Then, the other two triangles are *DYB* and *DZA* respectively.

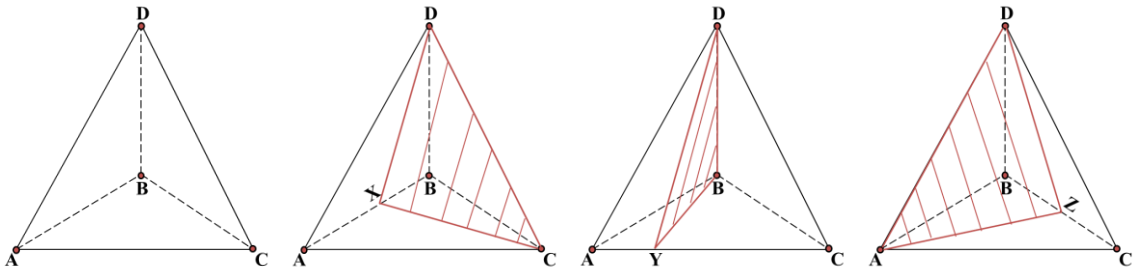


Figure 23. Three triangles constructed for vertex d of the tetrahedron ABCD.

The stiffness coefficients are then calculated using equation (47). For points *X*, *Y*, and *Z*, the displacement is calculated by interpolating the displacements of the two vertices of the corresponding edge. Finally, the local stiffness matrix for the tetrahedron is obtained by assembling all matrices associated with the twelve inserted triangles.

Burg [90] generalized the extension of the method for 3D meshes in order to be applicable for higher order elements, such as quadrilateral.

Ball-Vertex Spring Method. The concept of this method is to modify the original linear spring method by introducing additional linear springs. These additional springs resist the motion of a vertex towards its opposite faces, see Figure 24. Point *i* position is computed as the normal projection of the vertex *s* on the face *pqr*.

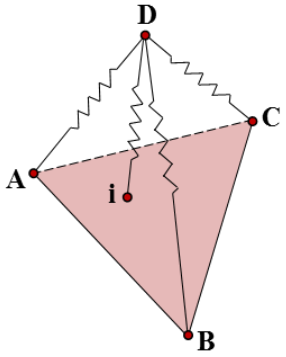


Figure 24. Ball-vertex additional linear spring [92].

The displacement at *i* can be calculated by the interpolation of the three vertices of the corresponding face. The stiffness of the added spring is,

$$\alpha_{is} = {}^{1}/_{L_{is}} \tag{48}$$

where

$$L_{is} = \sqrt{(x_i - x_s) \cdot (x_i - x_s)} \tag{49}$$

Semi-Torsional Spring Method. This method considers the same springs of the original linear spring method, but with different stiffness calculation approach. Here, the stiffness coefficient of each linear spring is equal to the torsional stiffness coefficient of

the facing angle. As was proposed in [88], for 2D triangular element, a semi-torsional stiffness coefficient of an edge *ij* is,

$$\alpha_{ij} = \frac{1}{l_{ij} \cdot \theta} \tag{50}$$

where l_{ij} is the length of the edge ij and θ is the edge facing angle.

This semi-torsional 2D model is not directly applicable to 3D problems. Zeing and Ethier [94] extended this method for 3D applications. They suggested defining the spring stiffness as the sum of its linear and semi-torsional stiffness, as follows,

$$\alpha_{ij} = {}^{1}/_{l_{ij}} + C \sum_{m=1}^{NE_{ij}} {}^{1}/_{sin^2(\theta_m^{ij})} \tag{51}$$

where NE_{ij} is the number of elements sharing edge *ij*, θ_m^{ij} is the edge facing angle on the m^{th} element sharing the edge *ij*, and *C* is a coefficient which is related to the dimension of the stiffness. Figure 25 shows the facing angle of an edge on a tetrahedron.

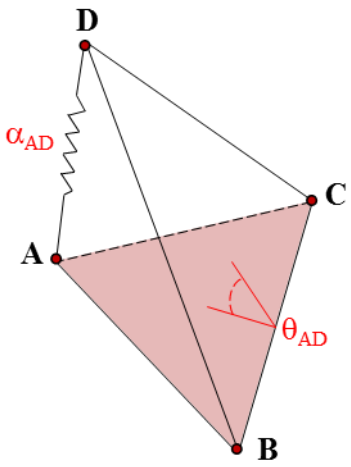


Figure 25. Definition of facing angle in a tetrahedron [92].

Thus, for edge *pq* the following system of equations results,

$$\begin{Bmatrix} F_{px} \\ F_{py} \\ F_{pz} \\ F_{qx} \\ F_{qy} \\ F_{qz} \end{Bmatrix} = \left(\frac{1}{l_{pq}} + \frac{1}{sin^2\,\theta^{pq}} \right) \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} u_{px} \\ u_{py} \\ u_{pz} \\ u_{qx} \\ u_{qy} \\ u_{qz} \end{Bmatrix} \tag{52}$$

This method requires the calculation of 6 torsional stiffness coefficients corresponding to the 6 angles created by four tetrahedral faces. Hence, this method is more efficient than the torsional spring method, which requires 12 triangles to be created for each tetrahedron. In 2D problems the calculations to form the fictitious stiffness matrix for the torsional spring analogy model involve 72 additions and 117 multiplications per element, while for the semi-torsional spring analogy model these operations are reduced to 18 additions and 20 multiplications per element [94].

Ortho-Semi-Torsional Spring Method. In this method four additional springs, within each tetrahedron element, have been considered in addition to the springs of the original spring method. These four springs are connecting each vertex with its projection on its opposite face, as shown in Figure 26. The stiffness coefficients of these imaginary springs were assumed to be inversely proportional to their lengths. These additional springs are used only to alter the original springs' stiffness coefficients, and they are discarded in the final formulation.

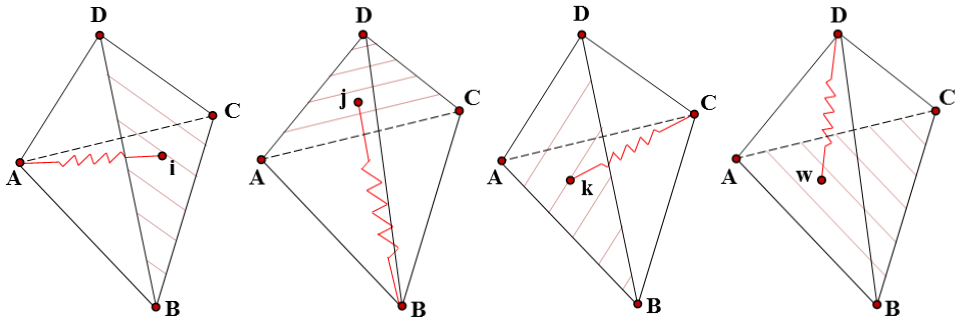


Figure 26. Additional projected spring from each node toward its opposite face [92].

Each additional stiffness coefficient is divided into three parts, one for each neighboring edge, as follow,

$$\lambda_{si,1} = \frac{l_{qs}}{l_{qs}+l_{ps}+l_{rs}},\ \lambda_{si,2} = \frac{l_{ps}}{l_{qs}+l_{ps}+l_{rs}},\ \lambda_{si,3} = \frac{l_{rs}}{l_{qs}+l_{ps}+l_{rs}} \tag{53}$$

Then each of the neighboring edges stiffness coefficients are calculated as,

$$\alpha_{qs}^{total} = {}^{1}/_{l_{qs}} + C\sum_{m=1}^{NE_{qs}} {}^{1}/_{sin^2(\theta_m^{qs})} + \left[\left(\frac{\alpha_{si}}{(\lambda_{si,1})^{k1}}\right) + \left(\frac{\alpha_{qi}}{(\lambda_{si,1})^{k1}}\right)\right]^{k2} \tag{54}$$

where $k1$ is a coefficient related to the closeness of the projection *si* to the neighboring edges and k2 is a coefficient affects the contribution of the additional linear spring stiffness coefficients to the total stiffness of the edge.

This method combines the simplicity of the semi-torsional spring method and the robustness of the torsional spring method.

Summary. Markou et al. [92] conducted several tests in order to analyze each of these four schemes. They concluded that the ball-vertex and semi-torsional spring analogy methods could be an appropriate choice for large problems with relatively small deformations. On the other hand, for large boundary surfaces deformations, then the

torsional spring analogy is a better choice. However, for both cases, the OST spring analogy method appeared to be ensuring robustness and computational efficiency.

Linear Elasticity

In this method, mesh deformation is accomplished by solving the linear elasticity equations for the mesh point displacements throughout the field. Since the elasticity equations contain material properties, the modulus of elasticity (E) and Poisson's ratio (υ), these properties are related to the mesh characteristics. One common approach is to set υ as a constant, within the valid physical range from 0 to 1/2, and E either to be calculated as the inverse of the distance between the interior node and the nearest boundary surface or to be set inversely proportional to the cell volume [95-97]. This turned out to be very beneficial for avoiding invalid mesh cells, especially near to the boundaries. An alternative approach is to use a constant E and manipulate the υ so that the term $1/(1-2\upsilon)$ is equal to the aspect ratio of the cell [61, 98] or to use constant υ and set E equal to the aspect ratio of the element [99]. This increases the stiffness in regions with high aspect ratio cells leading to more rigid motion near boundary surfaces. Another rarely used option is to set E equal to the element condition number which according to [100, 101] should result in the same effect of the previous approach.

Yang and Mavriplis [102] implemented an adjoint-based optimization procedure for producing a more optimal distribution of E. In this study, an objective function, that was selected to be proportional to the cell volume, was minimized by varying E in each cell. Even though the optimization resulted in avoiding invalid elements generation for highly stretched mixed element meshes, its solution consider to be expensive in terms of CPU time.

Hsu et al. [103] proposed to perform the deformation through two consecutive steps. The first step is performed with uniform modulus of elasticity (E = 1) and the second one with varying modulus of elasticity. The element strain energy density output from the first analysis is used to compute the modulus of elasticity for the second analysis.

Most of the studies concerned with the linear elasticity mesh deformation method used the Finite Element method for discretizing the linear elasticity equations and then solved the resulting linear system using GMRES method [95, 99, 103-105].

Laplacian and Modified Laplacian

Using the Laplace smoothing equations for mesh deformation proved to be an efficient technique. The idea behind using the Laplace equations for mesh deformation is that the solution of the Laplace equations satisfies the minimum/maximum principle. In other words, this means that the values of the interior displacements are bounded by the values on the boundary surfaces. This ensures that the interior nodes will not cross the boundaries. The traditional Laplacian method is to consider the Laplace smoothing equation,

$$\nabla \cdot (\nabla u) = 0 \tag{55}$$

where u is the mesh deformation velocity such that,

$$x_{new} = x_{old} + \Delta t\, u \tag{56}$$

The modified Laplacian includes the factor γ raised to some exponent q,

$$\nabla \cdot (\gamma^q \nabla u) = 0 \tag{57}$$

where γ is the diffusion coefficient, If $q = 0$, then the traditional Laplacian is recovered.

The selection of variable γ should depend on the specific mesh motion problem. Jasak and Tuković [106] investigated several possibilities to set the γ value based on distance (linear, quadratic, and exponential) from some boundary or mesh characteristics (orthogonality and skewness). These methods all have their merits and shortcomings. Lohner et al. [107] varied the diffusion coefficient with the distance from the viscous surfaces, and Crumpton and Giles [62] used diffusion coefficient inversely proportional to the cell volume.

One disadvantage that limits the use of traditional Laplacian method is that the three components of the mesh deformation are solved independently of each other. For example, if the boundary surface is moved only along x-direction, the interior mesh points will be moved only along coordinate x [103].

Choosing an appropriate exponent to the modified Laplacian for single frequency deformations would highly improve the capability of handling extreme deformations. Therefore, the modified Laplacian yields excellent results in cases of rigid translations and rotations. However, in practical problems, the mesh deformation has multiple frequencies which make the use of optimal exponent impractical and leads to invalid mesh generation [108].

MESH DEFORMATION USING INTERPOLATION ANALOGY

In general, these schemes do not require connectivity information. Therefore, these algorithms can be applied to arbitrary mesh types that contain general polyhedral elements or hanging nodes [12]. Interpolation based schemes attain higher computational efficiencies and less memory requirements compared to physical schemes. However, any interpolation process is associated with some sort of error margin.

Transfinite Interpolation

Most structured grid regeneration and deformation techniques are based on transfinite interpolation (TFI) [109]. In this method, an interior fluid node motion is assumed to be equal to the motion of the moving boundary times a scale factor. This scale factor depends on the distances from the node to the moving and the fixed surfaces [110, 111]. This method is very computationally efficient but suffers from robustness issues. Obviously, TFI method does not have any mechanism for preventing element crossing and overlapping; thus this method is not suitable for handling unstructured meshes especially when large deformations take place.

Algebraic Damping Method

The method, proposed by Zhao and Forhad [112], works by assigning a boundary node (x_{bi}) for each interior node (x_i) that needs to be deformed, then the deformation of this interior node is calculated as the product of a distance function and the displacement of the associated boundary node as follows:

$$\vec{D}(x_i) = f(x_{bi})\vec{D}(x_{bi}) \tag{58}$$

where $\vec{D}$ is the displacement vector and f is the distance function.

The boundary node that has the shortest distance to the interior node is selected as the associated node (x_{bi}). A generic distance function was chosen so it tends to 1 when $\|x_i - x_{bi}\|$ tends to 0 and the function tends to 0 when $\|x_i - x_{bi}\|$ tends to the $max\|x_i - x_{bi}\|$ for all interior nodes. This results in having a very rigid mesh in areas near to the boundary walls and far away from the boundary wall while having a very elastic mesh in between. In order to improve the robustness of this method for large mesh

deformations, a smoothing procedure was also incorporated to eliminate highly skewed or overlapping cells.

Inverse Distance Weighting Method

This method has been originally used for the generation of contour maps in geography [113]. In this method, the interpolated displacement value is an average of the known values at the boundary nodes weighted by the inverse of the distance to the interior fluid node. Thus, the displacement at any interior node x_i can be calculated by

$$\vec{D}(x_i) = \frac{\sum_{k=1}^{nb} x_k w(r_k)}{\sum_{k=1}^{nb} w(r_k)} \tag{59}$$

with weighting function

$$w(r) = r^{-c} \tag{60}$$

where nb is the total number of boundary nodes, $r_k = \|x_i - x_k\|$, and c is a power parameter which is usually set to a value of 2 in order to influence the distance-decay effect.

The IDW technique has the capability of treating boundary rotations separately. Therefore, the IDW technique can be used as a tool to improve the mesh orthogonality near the boundary surfaces. To illustrate this property, a single-block structured C-type inviscid mesh was tested. Different power parameters were considered. The results showed that for power parameter $c = 1$ the mesh orthogonality tends to be worst; however, by increasing c, the mesh quality improves spectacularly. The mesh quality measure shows a perfect value of 90^o over more than 95% of the airfoil surface for $c = 3$. Further increasing of c does not have noticeable effect on mesh quality. However, the effect of this optimization on the global mesh quality was not reported in this study.

In the research conducted by Witteveen [114], the 3D AGARD aeroelastic wing case has been tested as a full fluid-structure interaction problem. The results for IDW and RBF mesh deformation were compared. The results reported a reduction of computational costs for IDW mesh deformation with respect to the RBF method of a factor 20. Furthermore, by neglecting mesh rotation and considering only mesh translation the reduction of computational costs reduce the CPU time to a factor 50 with respect to RBF mesh deformation. However, the RBF technique produced 4% higher mesh quality than the IDW. It should be noted here that the study did not declare whether the used RBF approach is the most straightforward approach or the improved approach.

The main advantages of this approach can be summarized as, 1) it does not involve the solution of a matrix system of equations, and 2) it treats boundary node displacements and rotations separately.

Radial Basis Functions Interpolation

The radial basis function interpolation method, such as the method developed by Boer et al. [65], is one of the promising interpolation schemes. RBF's have become a well-established tool to interpolate scattered data. RBF can also be used as an interpolation function to transfer the displacements known at the boundaries of the structural mesh to the fluid mesh. This scheme produces high-quality meshes with reasonable orthogonality preservation near deforming boundaries. Other advantages of RBF includes: 1) Avoid the need for mesh connectivity information, 2) the system of equations which needs to be solved is linear, and 3) the size of the linear system of equation is proportional to the number of boundary nodes, not all fluid nodes. Moreover, many studies have investigated different techniques for improving RBF's interpolation based mesh deformation. The most

influencing study was made by Rendall and Allen [66]. They proposed the use of data reduction algorithm along with RBF interpolation. This technique will be discussed later in details. Another study, which builds up on the previous technique, is the work made by Sheng and Allen [115], in which they put forward specific criteria for selecting the nodes involved in the interpolation.

RBF Formulation. Using RBF, the interpolation function, *S*, describing the displacement in the whole domain can be approximated by a sum of basis functions as,

$$S(X) = \sum_{j=1}^{n_b} \alpha_j \, \phi\left(\left\| X - X_{b_j} \right\|\right) + P(X) \tag{61}$$

where $X_{bj} = [x_{bj} , y_{bj} , z_{bj}]$ are the boundary nodes in which the deformations are known and these are called the centers for RBF, P is a polynomial, n_b is the number of boundary nodes, and ϕ is the selected basis function with respect to the Euclidean distance $\|x\|$. The coefficients α_j and the polynomial P are determined by the interpolation conditions

$$S\left(X_{b_j}\right) = d_{b_j} \tag{62}$$

$$\sum_{j=1}^{n_b} \alpha_j = \sum_{j=1}^{n_b} \alpha_j \, x_j = \sum_{j=1}^{n_b} \alpha_j \, y_j = \sum_{j=1}^{n_b} \alpha_j \, z_j = 0 \tag{63}$$

The values for the coefficients α_j and the linear polynomial coefficient can be obtained by solving the system

$$\begin{bmatrix} \phi_{b,b} & p \\ p^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} d_b \\ 0 \end{bmatrix} \tag{64}$$

Where α is a vector containing the coefficients α_j, β is a vector containing the coefficients of the linear polynomial P, $\phi_{b,b}$ is an n_b x n_b matrix containing the evaluation

of the basis function $\phi_{b_i,b_j} = \phi\left(\left\|X_{b_i} - X_{b_j}\right\|\right)$, and p is an n_b x 4 matrix with row j given by $\left[1\ x_{b_j}\ y_{b_j}\ z_{b_j}\right]$ [64, 65]. In this study, the polynomial P was omitted, since it was concluded in previous studies that it does not have a large influence on the quality of the deformed mesh [116]. In this case, the system in equation (64) will be simplified as the following

$$[\phi_{b,b}][\alpha] = [d_b] \tag{65}$$

RBF interpolation method produces high-quality meshes with good orthogonality preservation near deforming boundaries. On the other hand, in its most straightforward implementation, it is too costly to use for large three-dimensional problems. A direct solution of such systems require $O(n_b^3)$ operations and $O(n_b^2)$ memory usage which becomes prohibitive for more than a few thousand data points. Great progress has been made in recent years towards alleviating this computational burden.

An approximation algorithm for RBF mesh deformation has been suggested by Rendall and Allen [66]. In this algorithm, the RBF is applied using a coarsened subset of the surface mesh. Displacements of the omitted surface nodes are calculated using the interpolation method and the error is calculated as the difference between the interpolated values and the actual displacement. A greedy algorithm is used to add points that have the largest error. Rendall and Allen reported that this algorithm improves the performance of the RBF method by approximately two orders of magnitude. The use of the greedy algorithm reduces the cost remarkably without compromising the accuracy. Michler [117] proposed a confinement technique that restricts the mesh deformation to the surrounding region of the moving surface. He achieved this by assigning an auxiliary geometry encompassing the region targeted by the interpolation, instead of using a cut-off function. Moreover, Michler [117] proposed to choose different centers for different directions

instead of using the same set of centers for all displacement directions, which results in reducing the required CPU time.

Control Mesh Methods

This scheme is based on the creation of a Delaunay graph of the original mesh. The Delaunay graph is used as an intermediate map. Only boundary nodes are used to create the graph. Then each interior node is assigned to the Delaunay element that it belongs to. Finally, by deforming the Delaunay graph, which the displacements are already known for, the interior nodes new locations are easily interpolated [118]. Basically, this method includes the following four steps:

1. Generating the Delaunay graph
2. Locating the mesh points in the graph
3. Moving the Delaunay graph according to the specified geometric change
4. Relocating the mesh points in the new graph

Generating the Delaunay graph. The Delaunay graph must be generated to cover the whole computational domain. All moving boundary nodes with few stationary boundary nodes must be used for generating the graph. If not all moving boundary nodes are used, the integrity of the grid is not guaranteed. If the stationary boundary surface has a curved shape, more points on the surface must be used. Then the Delaunay graph is generated using the Delaunay criterion [119, 120].

Locating the mesh points in the Delaunay graph. Any node within the graph must belong to one of the triangles or tetrahedrons elements in the graph, since the graph covers the whole solution domain. In order to locate the mesh nodes in the graph, the relative area/volume coefficients to define the points for 2D/3D meshes are used. For 2D, assume

that the node P is to be located within the Delaunay element ABC. By connecting the node P with each node of the Delaunay graph element ABC, three sub-triangles will be created. The area coefficients can be calculated as the area of the sub-triangle divided by the area of the element ABC. Similarly, for 3D the volume coefficients can be calculated as the volume of the sub-tetrahedron divided by the volume of the tetrahedron ABCD. Obviously, the summation of all coefficients must equal to 1. However, if and only if all the signs of the above coefficients are positive or zero, the point is within the graph element. Otherwise the point is classified outside the element.

An efficient walk-through algorithm [121, 122] was adopted for locating the nodes within the Delaunay graph. The complexity of the walk-through algorithm is of $O(n^{1/d})$, where n is the total number of the Delaunay elements and d is the spatial dimension.

Moving the Delaunay graph. After deforming the mesh, the Delaunay graph might not satisfy the Delaunay rules. This should not cause a problem unless some nodes move across each other, which will result in negative coefficients. In this case, the movement is broken into two smaller steps.

Relocating the mesh points in the graph. After the graph movement, the mesh points can now be located based on the area/volume coefficients stored for the points with the associated graph element number. This requires keeping the coefficients constant during the deformation.

Test Cases. Liu et al. [118] showed the robustness of the method through a series of test cases including inviscid and viscous flow grids with large deformations. The performance of the method was compared with a standard spring analogy method in the

wing-body test case. An order of magnitude improvement in CPU has been achieved, but on the other hand the memory requirements have been increased.

Main Disadvantage. Intersections occur occasionally between the Delaunay graph's elements for complex geometries with large relative movements. Under this condition, the Delaunay graph has to be regenerated and the grid nodes are required to be relocated, which increase the time consumption.

RBFs-MSA Hybrid Method

This method, recently proposed by Yu et al. [123], combines the benefits of Moving Submesh Approach (MSA) and RBFs interpolation, which is called RBFs-MSA. The MSA method can be considered as an extension to the Delaunay graph interpolation method. The main difference between the two methods is that in the MSA the background mesh is not a Delaunay graph anymore, it is a coarse unstructured mesh. This change avoids the elements crossing problem associated with using the Delaunay graph as the background mesh. On the other hand, since the background mesh now is not formed only by boundary nodes, the displacement is not known for all background nodes. Hence, an extra step needs to be performed before transferring the movement from the background mesh to the computational mesh. This step involves interpolating the displacements between the boundary nodes and the non-boundary nodes of the background mesh. Therefore, the authors proposed the use of RBFs interpolation to perform this step. Since the background mesh is a coarse mesh, performing RBFs interpolation will be more efficient.

RBFs-MSA Hybrid Algorithm's Steps.

1. Generate the background mesh
2. Locate the computational mesh nodes on the background mesh element and compute the area/volume ratio coefficients
3. Update the coordinates of the boundary points of the background mesh
4. Solve the RBF interpolation system and evaluate the non-boundary points new coordinates of the background mesh
5. Update the coordinates of computation mesh points by interpolation using the relative area/ volume ratio coefficients
6. Repeat Step 3 to Step 5 until the end of computation

Test Cases Results. Two 2D test cases and one 3D test case have been presented in [123]. Firstly, a rectangular block rotation 2D test case was analyzed and compared with RBFs interpolation and semi torsional spring scheme. The results showed that the semi torsional spring produces an invalid mesh after 50^{o} rotation. On the other hand, the RBFs-MSA and the RBFs interpolation produce a valid mesh for 90^{o} rotation. However, the RBFs interpolation produced slightly better mesh quality than the RBFs-MSA.

Secondly, a double flapping wings 2D test case was analyzed and compared with RBFs interpolation. The RBFs-MSA produced slightly better mesh quality than the RBFs interpolation. However, the RBFs-MSA reduced the CPU cost by about two orders of magnitude in comparison to RBFs interpolation.

Finally, a 3D test case was constructed by artificially bending the ONERA M6 wing. Since the computational mesh in this case has 10,419 boundary nodes, the authors stated that using RBFs interpolation is unpractical. Thus, the RBFs-MSA has been compared to the semi-torsional spring method. The comparison was in favor of the RBFs-MSA method. It showed that the RBFs-MSA method produced an acceptable mesh quality

while the semi-torsional method produced poor mesh quality. Moreover, the RBFs-MSA was 6.35 times faster than the semi-torsional spring method.

Quaternion Based Method

A quaternion can be interpreted as a scalar with a direction. It is composed of one real number and three imaginary numbers, on the form,

$$Q = q_0 + q_1 i + q_2 j + q_3 k \tag{66}$$

where $ii = jj = kk = -1$, $ij = -ji = k$, $jk = -kj = i$, and $ki = -ik = j$.

Quaternions are widely used to describe rotations in computer graphics, animations, control theories in robotics, attitude controls of spacecraft, etc. Quaternions have several unique properties, such as [124]:

1. Conjugate of a quaternion, $Q^* = q_0 - q_1 i - q_2 j - q_3 k$
2. Magnitude of a quaternion, $\|Q\| = \sqrt{QQ^*} = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}$
3. Unit quaternion, $\|Q\| = 1$
4. Inverse of quaternion, $Q^{-1} = Q^*/(QQ^*)$
5. For unit quaternion, $Q^{-1} = Q^*$

The nature of the quaternion makes it ideal to carry on rotational information, for which the scalar term could represent the magnitude of the rotation, and the last three terms represent the axis of rotation. Moreover, the unique properties those quaternions have make it easier to perform different mathematical operations on them.

Samareh [124] proposed a technique to use quaternions for mesh deformation. He proposed a three-steps approach to translate the known boundary displacements into boundary quaternions and then use these quaternions to deform the mesh. Step 1) is to translate the undeformed nodes, deformed nodes, and the neighboring nodes to the origin.

Step 2) is to rotate the undeformed nodes so their normal vectors align with the deformed nodes normal vectors. This step is performed using a quaternion, which is calculated as follows,

$$Q_1 = [cos(\alpha/2), n_u \times n_d \, sin(\alpha/2)] \tag{67}$$

where α is the angle between the two normal, and n_u and n_d are the undeformed and deformed normal, respectively. Step 3) is to rotate the undeformed nodes about the deformed boundary normal vectors in order to minimize the angle between neighboring nodes. Similarely, another quaternion can be used to perform this rotation,

$$Q_2 = [cos(\theta/2), n_d \, sin(\theta/2)] \tag{68}$$

where θ is the average angle between corresponding neighboring nodes. Step 4) is to compute the total rotation of any node as the composition of Q_1 and Q_2 as,

$$Q = Q_1 Q_2 \tag{69}$$

And the translation vector *T* is calculated as,

$$T = X_d - Q X_u Q^* \tag{70}$$

Where X_d and X_u is the deformed and undeformed node position vector, respectively. The translation vectors and quaternions are propagated into the field mesh using the spring analogy. Maruyama et al. [125] proposed a modification to this method in order to generalize the formulation and obtain the quaternions independently of the coordinate system. They used Laplacian smoothing to propagate the translation vectors and the quaternions into the field. Moreover, they reported that this method is at least one order of magnitude more CPU intensive than other interpolation based mesh deformation methods, such as RBF interpolation method.

Worth Mentioning Approaches

Recently, a novel interpolation based scheme has been developed by Luke [12]. In this scheme, the deformation of the volume mesh is viewed as a projection of the surface deformation into the volume. Using a tree-code optimization, the algorithm cost is demonstrated to be O(*nlog(n)*), where n is the total number of nodes in the simulation, with mesh quality that is competitive to radial basis function (RBF) scheme.

McDaniel and Morton [63] developed a technique that is based on a two-pronged approach where the viscous layers of nodes are deformed rigidly and the outer region is deformed with two different interpolation techniques. Several different rigid deformation schemes were investigated. However, the results showed that the best performing scheme was based on a semi-rigid connection to the owner surface nodes defined as part of the mesh parsing, which provided smoother deformation in convex regions. The last layer of the viscous region was used as the deforming boundary surface for the outer region deformation.

CONCLUSIONS

In this chapter, the mesh deformation methods are categorized into physical analogy based methods and interpolation based methods. Basically, the physical analogy approaches treat the mesh deformation problem as a physical process that can be modeled using numerical methods. These approaches are accurate, reliable, and can be case-optimized by changing the physical parameters. However, these methods involve solving large systems of equations, implying a higher computational cost. Besides, these methods require grid connectivity information which results in more storage requirements and difficulties in parallelization.

On the other hand, in the interpolation based approaches, an interpolation function is used to transfer prescribed boundary point displacements to the fluid mesh. These approaches are fast, easy to implement and stable. Despite that interpolation error cannot be completely avoided, it can be limited to be below certain tolerance. Also, these methods can accommodate any mesh topology and do not require mesh connectivity information; thus they are easy to parallelize.

Based on the survey conducted in this chapter, it was concluded that interpolation based approaches are more suitable for the FSI applications that are targeted by this study. The aim of this study is to develop a generic FSI solver that is capable of handling any mesh topology. Moreover, more avenues for improvements are seen within the interpolation based approaches. It has been decided that the RBF based mesh deformation technique is the most promising technique among the reviewed techniques. As stated earlier in this chapter, RBF scheme produces high-quality meshes with reasonable orthogonality preservation near deforming boundaries. Moreover, it produces linear system of equations, and the size of this system of equations is proportional to the number of boundary nodes, not all fluid nodes. In the following chapter the implementation of a novel improvement in the conventional RBF based mesh deformation technique is described and tested.

CHAPTER 5

OBJECTIVE 3: RBF BASED MESH DEFORMATION DEVELOPMENT AND IMPROVEMENT

INTRODUCTION

Many dynamic computational simulations, such as Fluid Structure Interaction (FSI) and control surface deflections involve moving/deforming meshes. A critical step in the analysis of this class of problems is a successful mesh deformation due to structural deflections. For an accurate simulation, the fluid volume mesh needs to move conformal to the structure, with very little degradation in quality. Due to the repeatability of the mesh deformation and the large number of fluid cells, an efficient and reliable approach is needed for a successful analysis. In addition, the fluid mesh is typically partitioned and handled by different processors. This necessitated the mesh deformation algorithms be parallelizable for efficient simulations. The main complexity of the moving mesh approach is to find an optimum technique that is suitable for different mesh topology and physical situations. At the same time, it should preserve, as much as possible, the quality of the mesh while keeping computational cost low. The objective of this chapter is to investigate the feasibility of using radial basis function (RBF) based interpolation technique with greedy algorithm to deform large-scale generalized meshes and to improve the traditional greedy algorithm using an incremental approach. In the following sections, the RBF formulation is revisited in more details and the proposed novel improvement approach is described.

RBF FORMULATION

Using RBF, the interpolation function, *S*, describing the displacement in the whole domain can be approximated by a sum of basis functions as,

$$S(X) = \sum_{j=1}^{n_b} \alpha_j \, \phi\left(\left\| X - X_{b_j} \right\|\right) + P(X) \tag{71}$$

where $X_{bj} = [x_{bj}, y_{bj}, z_{bj}]$ are the boundary nodes in which the deformations are known and these are called the centers for RBF, P is a polynomial, n_b is the number of boundary nodes, and ϕ is the selected basis function with respect to the Euclidean distance $\|x\|$. The coefficients α_j and the polynomial P are determined by the interpolation conditions

$$S\left(X_{b_j}\right) = d_{b_j} \tag{72}$$

$$\sum_{j=1}^{n_b} \alpha_j = \sum_{j=1}^{n_b} \alpha_j x_j = \sum_{j=1}^{n_b} \alpha_j y_j = \sum_{j=1}^{n_b} \alpha_j z_j = 0 \tag{73}$$

equation (73) ensures that the polynomial P is coinciding with the interpolant S, by forcing the polynomial in equation (71) to zero when $X = X_{b_j}$ [126].

The values for the coefficients α_j and the linear polynomial coefficients β_j can be obtained by solving the system

$$\begin{bmatrix} \phi_{b,b} & p \\ p^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} d_b \\ 0 \end{bmatrix} \tag{74}$$

Where α is a vector containing the coefficients α_j, β is a vector containing the coefficients of the linear polynomial β_j, $\phi_{b,b}$ is an n_b x n_b matrix containing the evaluation

of the basis function $\phi_{b_i,b_j} = \phi\left(\left\|X_{b_i} - X_{b_j}\right\|\right)$, and p is an n_b x *4* matrix with row j given by $\left[1\ x_{b_j}\ y_{b_j}\ z_{b_j}\right]$ [64, 65].

In this study, the polynomial *P* was omitted, since it was concluded in previous studies that it does not have a large influence on the quality of the deformed mesh [127]. In this case, the system of equations in (74) will be simplified as the following

$$[\phi_{b,b}][\alpha] = [d_b] \tag{75}$$

Generally, RBFs can be divided into two groups 1) functions with compact support, and 2) functions with global support. The main difference between the two types is that the function with compact support is scaled with a support radius (*r*) to control the extent of influence of the basis function. Functions with global support cover the entire interpolation space, which leads to dense matrix systems.

Table 7 lists some RBFs for both these types [64, 65].

Functions with compact support are forced to satisfy the following condition

$$\phi_{b,b} = \begin{cases} \phi\left(\left\|X_{b_i} - X_{b_j}\right\|\right) & if\ 0 \le \xi \le 1 \\ 0 & if\ \xi > 1 \end{cases} \tag{76}$$

Table 7

Examples of Different RBFs Types

No.	Function ϕ	Type	Remarks
1	$(1-\xi)^2$	Compact Support	$\xi = {}^{x}/_{r}$
2	$(1-\xi)^4(4\xi+1)$	Compact Support	$\xi = {}^{x}/_{r}$
3	$(1-\xi)^6\left({}^{35}/_{3}\xi^2 + 6\xi + 1\right)$	Compact Support	$\xi = {}^{x}/_{r}$
4	e^{-x^2}	Global Support	Gaussian
5	$x^2\log(x)$	Global Support	Thin Plate Spline
6	$1+x^2$	Global Support	Quadric Biharmonics

Direct RBF Scheme Limitations and Improvements

The radial basis function interpolation method produces high-quality meshes with good orthogonality preservation near deforming boundaries. On the other hand, in its most straightforward implementation, it is too costly to be used for large three-dimensional problems. A direct solution of such systems requires $O(n_b^3)$ operations and $O(n_b^2)$ memory usage which becomes prohibitive for more than a few thousand data points. Great progress has been made in recent years towards alleviating this computational burden.

An approximation algorithm for RBF based mesh deformation has been suggested by Rendall and Allen [66]. In this algorithm, the RBF is applied using a coarsened subset of the surface mesh. Displacements of the omitted surface nodes are calculated using the interpolation method and the error is calculated as the difference between the interpolated values and the actual displacement. A greedy algorithm is used to add points that have the largest error. Rendall and Allen report that this algorithm improves the performance of the RBF method by approximately two orders of magnitude. The use of the greedy algorithm reduces the cost remarkably without compromising the accuracy.

The presented approach is an extension of the work done by Rendall and Allen [66] by utilizing incremental approaches to solve the system of equations resulting from RBF formulation. This study takes advantage of solving similar systems of equations within each iteration by using an incremental approach [128] to reduce the CPU time by avoiding the use of any expensive linear system solver such as full LU decomposition. Two incremental approaches will be presented and compared in this chapter: 1) Matrix inversion based approach, and 2) Incremental LU decomposition based approach. The proposed method does not require any pre-conditioning, since it is not affected by the matrix's

condition number, which makes it very robust and suitable for any compact support RBF. The procedure of picking the centers using the greedy algorithm and the use of the incremental approaches are described in the following sections.

GREEDY ALGORITHM

A direct RBF method would use all boundary nodes as interpolation centers. For large-scale unstructured-grid problems, the number of boundary nodes is usually of the order of 10^4 to 10^5, therefore calculating the interpolation weight coefficients tends to be very costly. Moreover, if the mesh is required to be deformed at each time step, direct RBF method cost will be impractical. It was reported in the literature that to achieve high mesh deformation accuracy, only a small subset of the total boundary nodes is sufficient [66, 115, 117].

Centers selection is highly dependent on the shape of the geometry, displacement vector, and the support radius in the case of compact support RBFs. Therefore, there is no direct way to predict which nodes to be selected in advance. Hence, a greedy algorithm is used to iteratively select these centers. Initially, two boundary nodes are selected during the first iteration. In each subsequent iteration, the selected centers are used to predict the deformation of the unselected centers. Based on this prediction the error is calculated at all boundary points, which is the difference between the actual and the predicted deformations. The greedy algorithm is stopped if the maximum error is below a specified tolerance. Otherwise, the node with the maximum error is added to the selected subset, and the iteration continues. Sheng and Allen [115] investigated the possibility of selecting the centers based only on the geometry. Thus, the greedy algorithm is used only once and the

same subset of centers is used for all time steps for different deformations. A typical greedy algorithm procedure is expressed in Algorithm 1.

Algorithm 1: Greedy Algorithm Procedure

Procedure *Greedy_RBF*

Read in real deformation for all boundary nodes

Choose an initial subset (two nodes) to begin with

Do While (Maximum Error > Tolerance)

Calculate $\phi_{s,s}$ & $\phi_{u,s}$

Solve equation (75)

Evaluate deformation for all unselected centers

Calculate the error

Select node with largest error and add it to selected centers list

End Do

Evaluate volume deformation using only reduced subset of centers

Update volume mesh

Assume the greedy algorithm starts with *n* selected centers, equation (75) becomes

$$\begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,n} \\ \vdots & & \vdots \\ \phi_{n,1} & \cdots & \phi_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} \tag{77}$$

or

$$[\phi_n][\alpha_n] = [d_{b_n}] \tag{78}$$

where $\phi_{i,j} = \phi(\|X_i - X_j\|)$.

Then for the next iteration, the node with the largest error will be added to the list, and the system will become

$$\begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,n} & \phi_{1,n+1} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{n,1} & \cdots & \phi_{n,n} & \phi_{n,n+1} \\ \phi_{n+1,1} & \cdots & \phi_{n,n+1} & \phi_{n+1,n+1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_{n+1} \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \\ d_{n+1} \end{bmatrix} \tag{79}$$

or

$$[\phi_{n+1}][\alpha_{n+1}] = \left[d_{b_{n+1}}\right] \tag{80}$$

Note that

$$[\phi_{n+1}] = \begin{bmatrix} \phi_n & \phi_{i,n+1} \\ \phi_{n+1,i} & \phi_{n+1,n+1} \end{bmatrix} \tag{81}$$

where i = [1, 2, …, n].

For compact support RBFs, the matrix $\phi_{b,b}$ is symmetrical with ones in the diagonal. Therefore, the system can be simplified as

$$\begin{bmatrix} \phi_n & \phi_{add} \\ {\phi_{add}}^T & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_{n+1} \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \\ d_{n+1} \end{bmatrix} \tag{82}$$

where ϕ_{add} is a vector that contains the RBF evaluations for the newly added node with respect to each selected center.

The following sections describe the proposed approaches for solving equation (75) incrementally which is considered the novel contribution of this chapter.

Incremental Matrix Inversion Based Approach

The most straight-forward technique for solving the above system is by calculating the inverse of the coefficients matrix. Since the coefficient matrix consists of the matrix from the previous iteration with an addition of one row and one column, it is possible to use an incremental approach to compute the inverse using the inverse from the previous iteration.

The inverse of a general block matrix can be calculated as [129]

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \tag{83}$$

By applying this principle to the calculation of the inverse of the coefficient matrix in equation (82) will result in,

$$[\phi_{n+1}]^{-1} = \begin{bmatrix} \phi_n & \phi_{add} \\ \phi_{add}{}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} (\phi_n - \phi_{add}\,\phi_{add}{}^T)^{-1} & -\frac{1}{k}\phi_n{}^{-1}\,\phi_{add} \\ -\frac{1}{k}\,\phi_{add}{}^T\,\phi_n{}^{-1} & \frac{1}{k} \end{bmatrix} \tag{84}$$

where $k = 1 - \phi_{add}{}^T\,\phi_n{}^{-1}\phi_{add}$

Since the matrix ϕ_n is symmetrical, the previous equation can be simplified as

$$[\phi_{n+1}]^{-1} = \frac{1}{k}\begin{bmatrix} k\phi_n{}^{-1} + \xi\xi^T & -\xi \\ -\xi^T & 1 \end{bmatrix} \tag{85}$$

where $\xi = \phi_n{}^{-1}\,\phi_{add}$

This step has $O(n^2)$ complexity [128] instead of $O(n^3)$ complexity required by Gauss elimination or LU decomposition [130]. The matrix system resulting from RBFs based formulation is highly ill-conditioned [115, 131], which makes it difficult to use traditional iterative matrix solvers. The current incremental approach is independent of the condition number of the matrix. As shown in the following section, it requires less computational time as compared to LU decomposition. Moreover, this step is repeated for every iteration until the greedy algorithm converges, which makes the complexity reduction more significant. However, it must be noted here that during the first greedy algorithm iteration, the inverse of the matrix must be computed using any traditional method. At this specific iteration, the system is very small in size, typically 2x2, which is very feasible to solve.

Incremental LU Decomposition Based Approach

LU decomposition of the matrix ϕ_{n+1} defined in equation (84) can be calculated incrementally using the LU decomposition of the matrix ϕ_n. This can be performed by the addition of one row and one column to the LU decomposition of the matrix ϕ_n, as shown in equation (86). In this case, performing full LU factorization is needed only once at the first iteration. For all subsequent iterations an incremental LU decomposition will be used.

$$[LU_{n+1}] = \begin{bmatrix} u_{1,1} & \cdots & u_{1,n+1} \\ \vdots & \ddots & \vdots \\ l_{1,n+1} & \cdots & u_{n,n+1} \end{bmatrix} = \begin{bmatrix} & [LU_n] & & u_{1,n+1} \\ & & & \vdots \\ l_{1,n+1} & \dots & l_{n,n+1} & u_{n,n+1} \end{bmatrix} \tag{86}$$

The incremental LU decomposition procedure requires only two nested do loops to perform the decomposition instead of three nested do loops which are regularly required by the full LU decomposition. The resulted LU_{n+1} from this code is used, alongside with d_{b_n}, to calculate the interpolation weight coefficients using forward and backward substitutions.

RESULTS AND DISCUSSION

Two-dimensional test functions

To investigate the accuracy and efficiency of the presented method, four analytical test functions were chosen to deform a structured mesh. A mesh refinement study is also conducted to analyze influence of the total number of nodes on the performance of the algorithm. Three different structured meshes used for this study are shown in Figure 27. The *x*- and *y*-coordinates of the computational domain varies from -1.5 to 1.5. The *z*-coordinate of the computational domain is specified as zero before the deformation and it

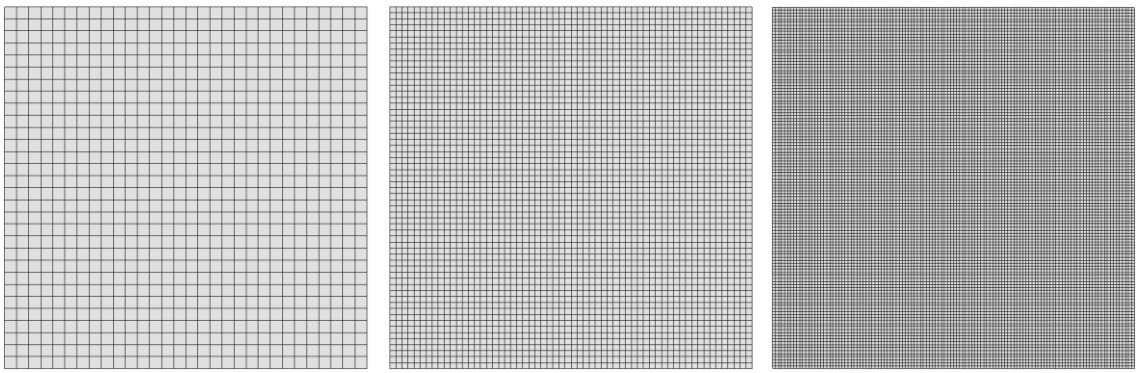
is specified as the analytic functions after the deformation. The specified deformations of the mesh for this study are listed below.

$$F1(x,y) = (1 + 9x^2 + 16y^2)^{-1} \quad (87)$$

$$F2(x,y) = 1 - \left((x^2 + y^2)/2\right)^{1/2} \quad (88)$$

$$F3(x,y) = 1.5xe^{-x^2-y^2} \quad (89)$$

$$F4(x,y) = \left(1.25 + \cos(5.4y)\right) / (6 + 6(3x - 1)^2) \quad (90)$$



(a) (b) (c)

Figure 27. Surface mesh with three different element spacing: (a) h = 0.1, (b) h = 0.05, and (c) h = 0.025.

The displacement at each node is pre-calculated using the above functions, and two random nodes are selected as initial centers for the RBFs. The greedy algorithm uses the initial centers to calculate the RBF weight coefficients and evaluate the displacement for all the remaining nodes. The error is calculated as the difference between the pre-calculated displacement and the evaluated displacement to check the stopping criteria for the greedy algorithm. If the maximum error is greater than the specified tolerance, the node with the maximum error is added to the list of centers for RBF. If the maximum error is less than the specified tolerance, the greedy algorithm is stopped and the fluid mesh deformation is performed using only the selected RBF centers.

From a computational cost point of view, the greedy algorithm consists of three main steps:

1. Evaluating the RBF between the nodes and forming $\phi_{s,s}$ & $\phi_{u,s}$
2. Solving equation (75) for the weight coefficients
3. Evaluating the interpolated deformations for all unselected centers

Using conventional methods for performing solving equation (75) makes Step 2 the most expensive step within the greedy loop. Figure 28 illustrates the computational cost of each step as a percentage of the total CPU time of the greedy algorithm for the different shape functions deformations when using Full LU decomposition. In this specific case, Step 2 formed approximately an average of 70% of the total CPU time of the greedy algorithm. Thus, reducing the cost of solving equation (75) would highly improve the greedy algorithm's performance, which the proposed method aims to.

To assess the enhancement gained by using the greedy algorithm with the incremental approach, the performances of the proposed algorithms were compared against the use of a full LU decomposition. In this test case, the compact support RBF $\emptyset = (1-\xi)^4(4\xi+1)$ was used with a predefined error tolerance of 5×10^{-4}. Table 8 shows the results of this comparison, for the case of element spacing h = 0.05, which clearly illustrates the superiority of the use of greedy algorithm along with incremental approach. Shape function F2 requires picking relatively a higher number of centers, to achieve the predefined tolerance, compared to other shape functions. This leads to perform more iterations for the greedy algorithm, which consequently leads to a longer CPU time especially for the full LU decomposition case.

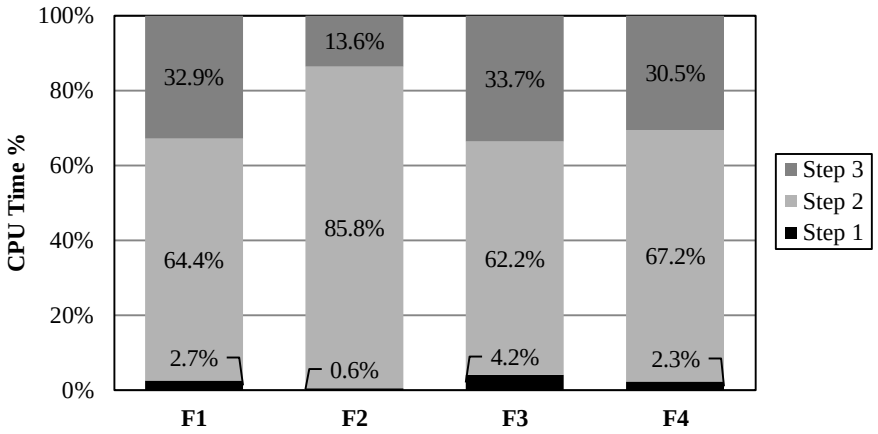


Figure 28. Computational cost percentages of greedy algorithms steps for different shape functions using full LU decomposition for element size (h) = 0.05.

Table 8
Time comparison between RBF with greedy algorithm using full LU decomposition and incremental approaches for element size (h) = 0.05

Analytical Shape Function	**Greedy RBF (Tolerance = 5×10^{-4})**					
	Centers Percentage	**CPU Time (Seconds)**			**Time Saving**	
		Full LU	**Matrix Inv.**	**LU Decomp.**	**Matrix Inv.**	**LU Decomp.**
F1	6.9 %	0.92	0.4	0.41	56.4 %	55.6 %
F2	11.5 %	4.84	1.57	1.59	67.6 %	67.3 %
F3	3.5 %	0.32	0.13	0.14	57.5 %	56.8 %
F4	5.6 %	0.82	0.31	0.32	61.6 %	60.9 %

The number of selected centers and total number of nodes on the surface for three meshes with different resolution and for four different deformations are compared in Figure 29. In these calculations $1.0\text{x}10^{-3}$ is set as the tolerance criteria for stopping the greedy algorithm and a support radius of 1.5 units is used.

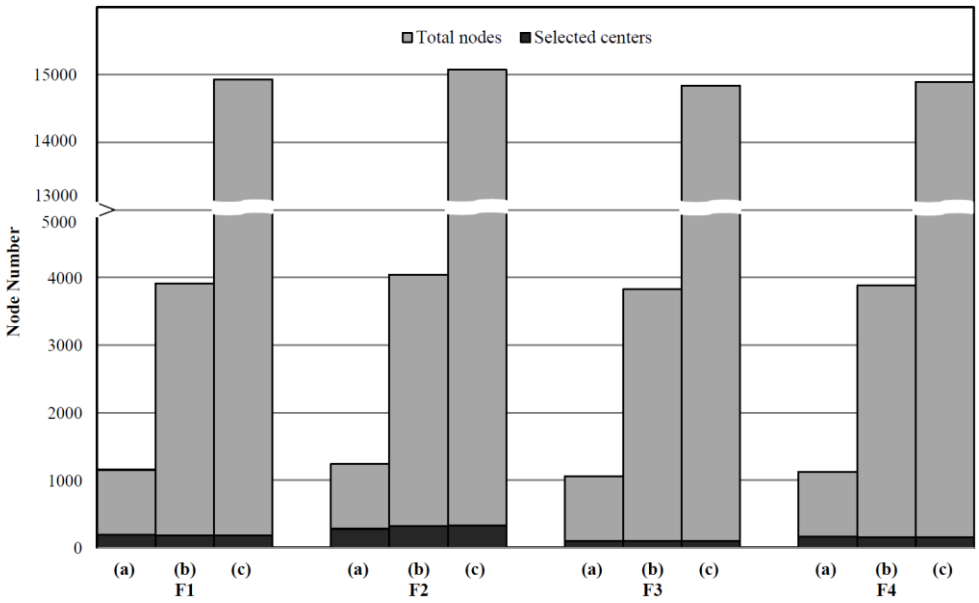


Figure 29. Total number of nodes versus selected number of centers for the three different mesh refinements: (a) h = 0.1, (b) h = 0.05, and (c) h = 0.025.

It can be seen from the figure that the mesh refinement did not cause the number of selected centers to increase. The number of centers required for each of the analytical deformation remains more or less the same irrespective of the number of surface nodes. However, the number of selected nodes varies with the analytical functions used for the deformation. This confirms that, for the same geometry, the number of selected centers is a function of the type of deformation.

The specified four different analytic deformations are used for this study. The selected RBF centers for the interpolation for three different meshes are shown in Figure 30. In this figure the color contour is specified based on the deformation. It can be seen from the figure that the pattern of the selected RBF centers are the same for all different mesh resolutions. Figure 31 shows the deformed mesh shaded with the absolute error for a specified error tolerance of 1×10^{-4} for all four analytical benchmark deformations. This

figure shows the error is smaller in the vicinity of the selected nodes and relatively higher near the unselected nodes.

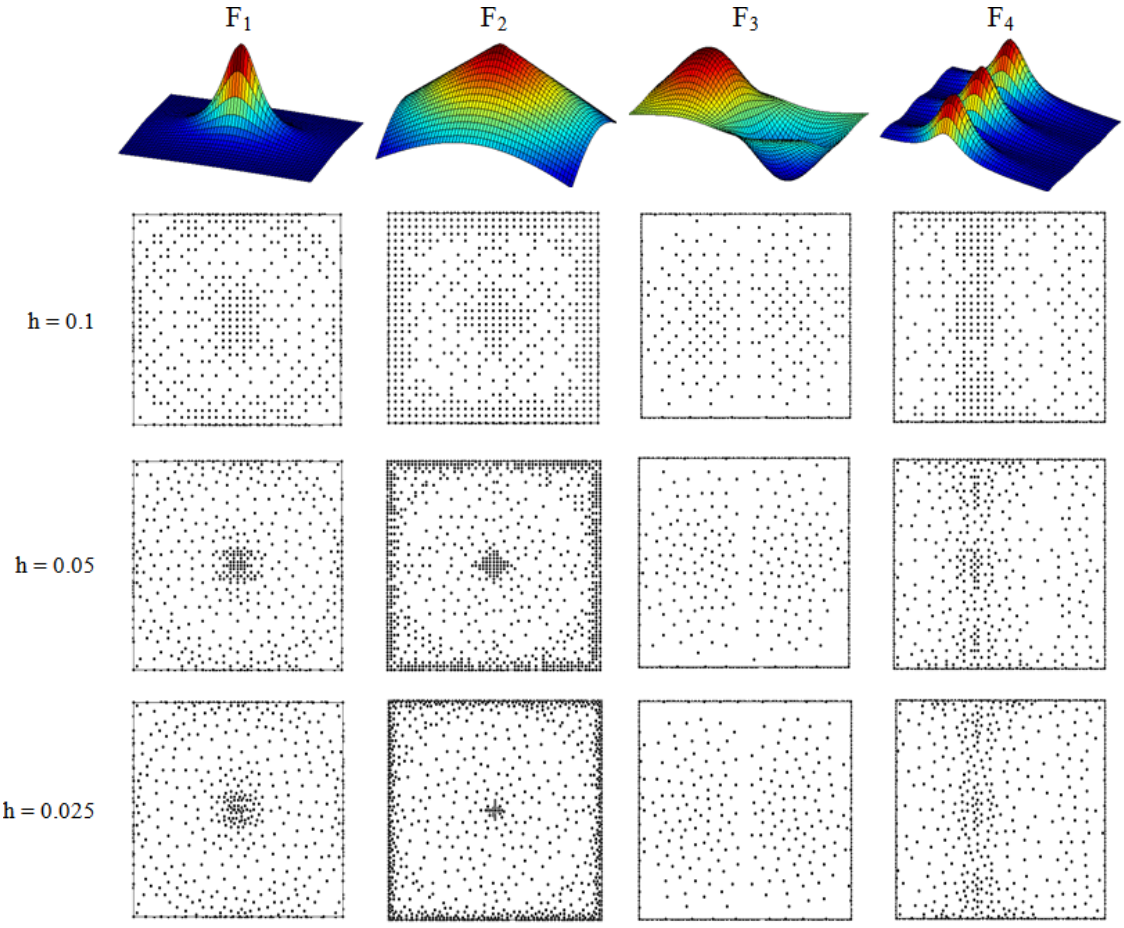


Figure 30. Original surface mesh and selected centers (top view) for coarse, medium, and fine meshes.

A study is also conducted to analyze the effect of error tolerance on the number of selected centers and the CPU time required for the selection of appropriate centers. The fine mesh (h = 0.025) from the mesh refinement study is used for this analysis. The number of selected centers and CPU time requirements were calculated for error tolerances of 1×10^{-3}, 5×10^{-4}, and 1×10^{-4}. As expected, the finer the specified tolerance, the higher the number of centers picked and longer the CPU time required. Figure 32 shows the variation

of CPU time and number of selected centers versus the specified tolerance. It is noticeable from the chart that the number of selected centers and the CPU time for the greedy algorithm increases with the decrease in error tolerance.

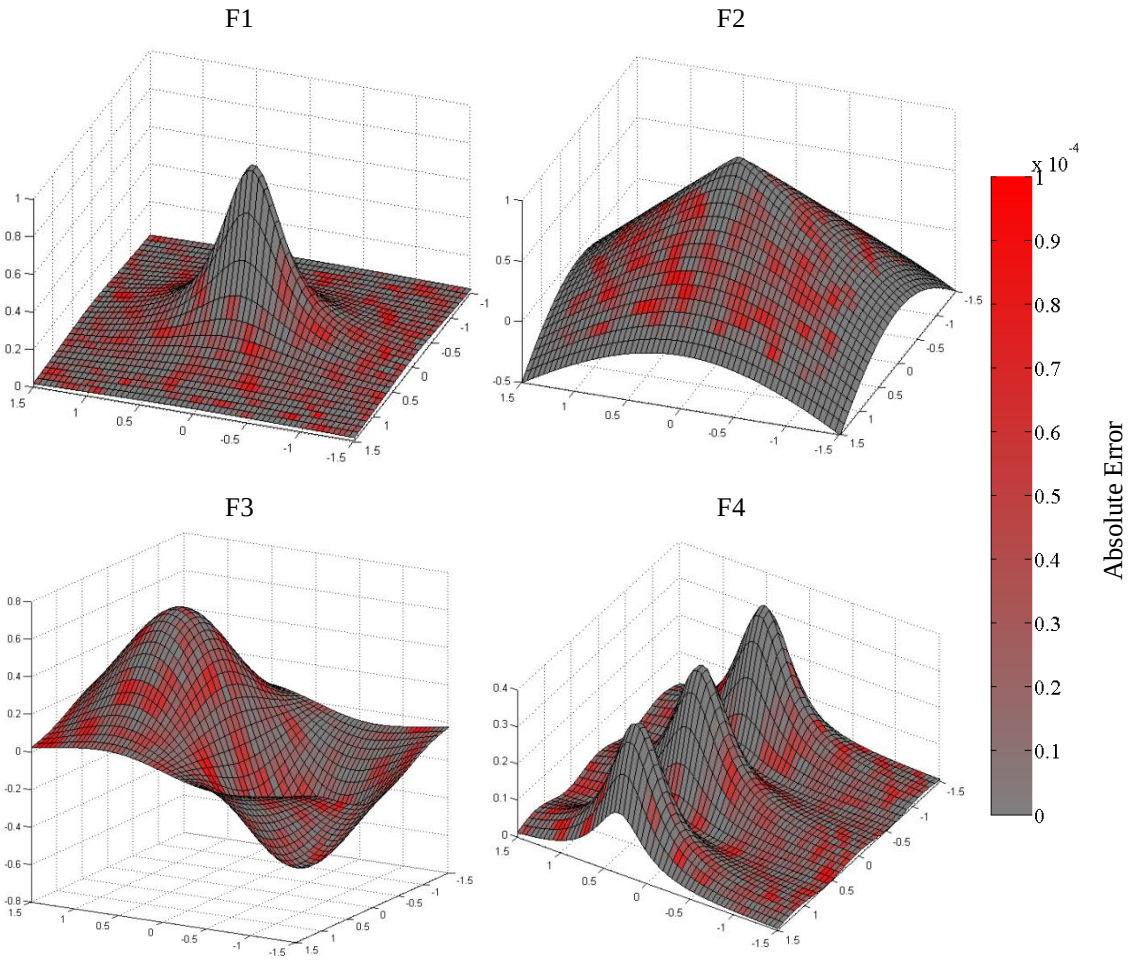


Figure 31. Deformed mesh shaded by the absolute error.

To determine the dependency of the total number of nodes (N) on the interpolation error, the Root Mean Square (RMS) error is plotted against the number of nodes using a logarithmic scale in Figure 33. It can be seen from the figure that all different analytical functions for deformation showed an approximate slope of 0.4 or less, indicating that the total number of nodes has a minimal effect on the interpolation error.

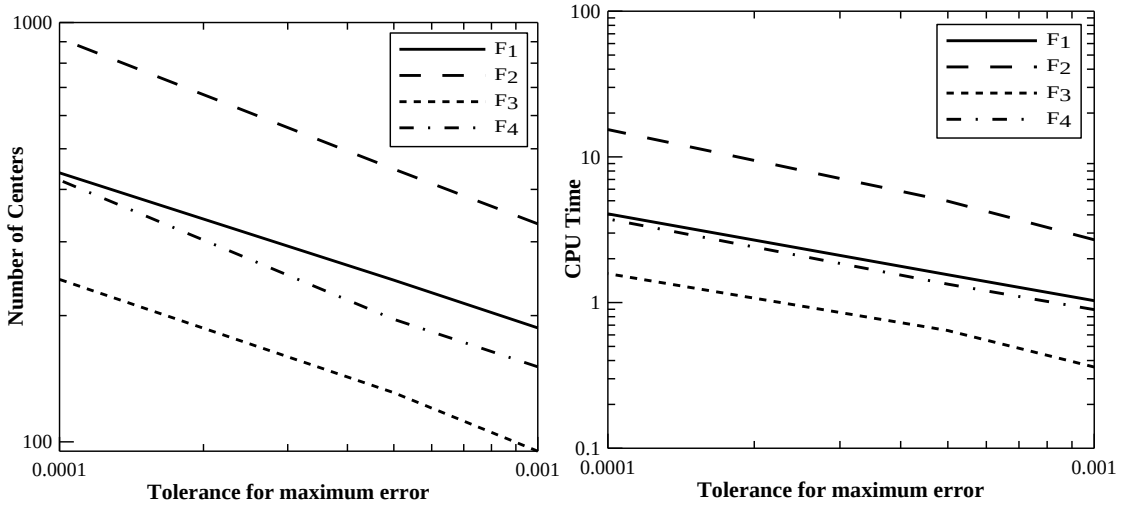


Figure 32. Tolerance versus number of centers (left), tolerance vs. CPU time (right).

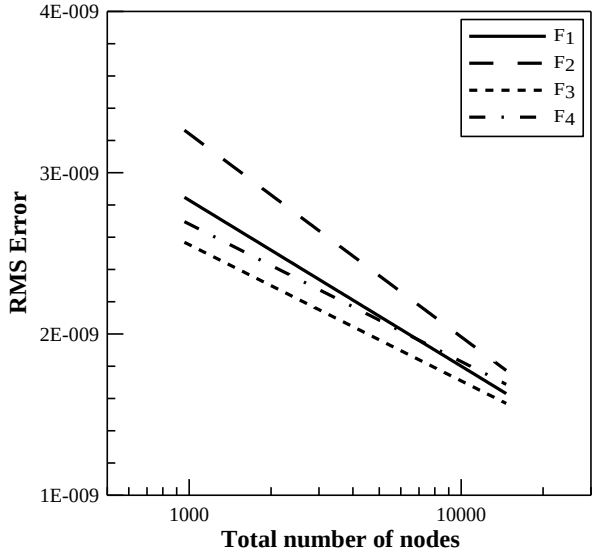


Figure 33. Variation of the RMS error with the total number of centers.

Cantilever beam vibration in an oscillating flow field

To study the effect of the support radius on the mesh quality, a well-known three-dimensional FSI problem for flow-induced vibration of a beam structure is considered [45, 132]. The geometry of the beam and the computation domain are depicted in Figure 34. The generated computational mesh consists of prism elements in the boundary layer and

tetrahedral elements away from the beam, as shown in Figure 35. The mesh consists of 5,427 nodes, out of which 2,471 are the boundary nodes. The deformation is set to zero for all the boundary nodes, except the ones on the beam. The nodes on all the boundary surfaces are considered for the interpolation, even though some of the boundary nodes have zero deformation.

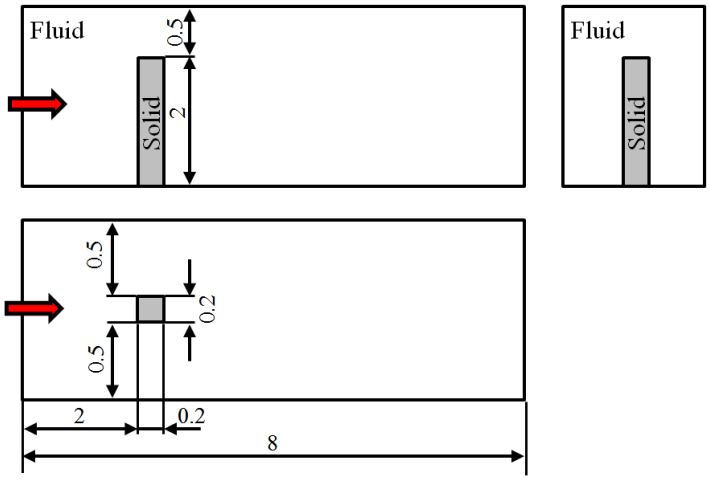


Figure 34. Cantilever beam vibration computation domain [132].

The beam is assumed to be fixed at the bottom, and it is assumed to deflect in the cross-flow direction based on the sinusoidal shape with an angle of 7 degrees at the tip of the beam. The tolerance to stop the greedy algorithm was set to $5x10^{-4}$ and the support radius is set as 1.45 units, which is half the distance between the furthest fluid node and any boundary node. The greedy algorithm selected 307 centers, which is only 12% of the total number of boundary nodes. The centers selection process and calculating the RBF weighting coefficients took approximately 0.3 seconds using single Intel CPU with 2.6 GHz clock speed. The deformed mesh is shown in Figure 35. Also, Figure 35 illustrates the influence of different support radii on deformed mesh region. In this figure, the region with red cells represents the region influenced by the deformation. The use of a larger

support radius will lead to include more far away nodes in the interpolation and increase the CPU requirements. On the other hand, the use of a smaller support radius will lead to better performance but it will compromise the resulting mesh quality as the deformation will be confined to a small region.

The mesh skewness distribution for the original mesh and deformed mesh with different support radii is shown in Figure 36. The mixed element mesh generator called MEGG3D is used to evaluate the skewness of the mesh elements [133]. Assuming, V as the volume of a tetrahedron, and V_{opt} as the volume of the equilateral tetrahedron with the same circumsphere, the skewness ε is defined as $\varepsilon = (V_{opt} - V)/V_{opt}$. A skewness value of zero represents an equilateral tetrahedron, while a value of unity represents a degenerate element. For a support radius of 0.6 units, more elements tend to have higher skewness, which leads to lower mesh quality. On the other hand, skewness distribution for support radii of 1.45 and 2.9 is almost identical to the undeformed mesh indicating the preservation of mesh quality.

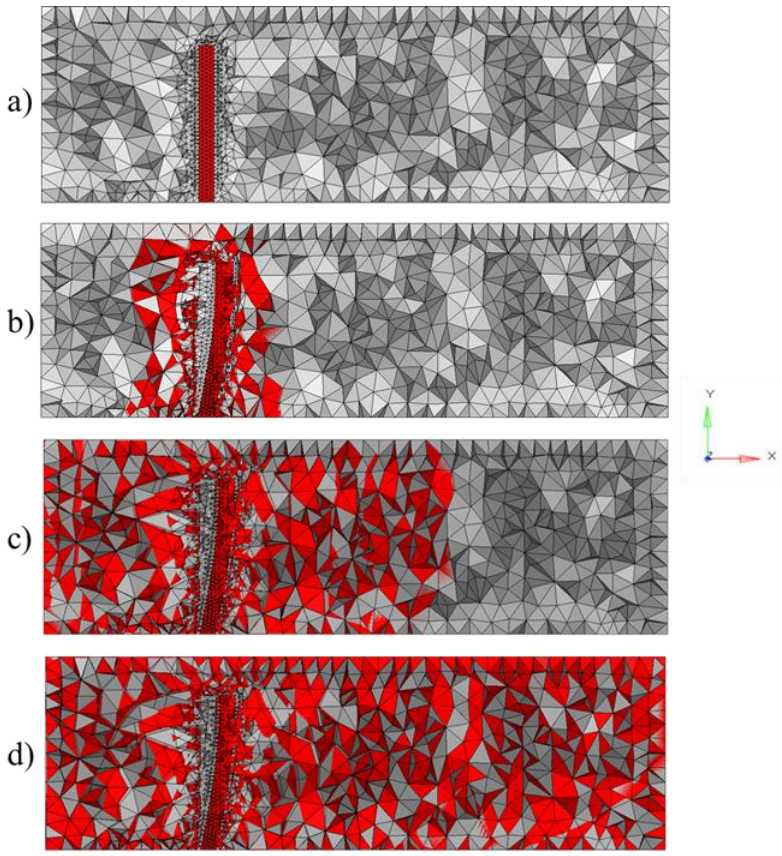


Figure 35. XY-plane cross-section of original and deformed meshes for several compact radii (r): a) undeformed, b) r=0.6, c) r=1.45, d) r=2.9.

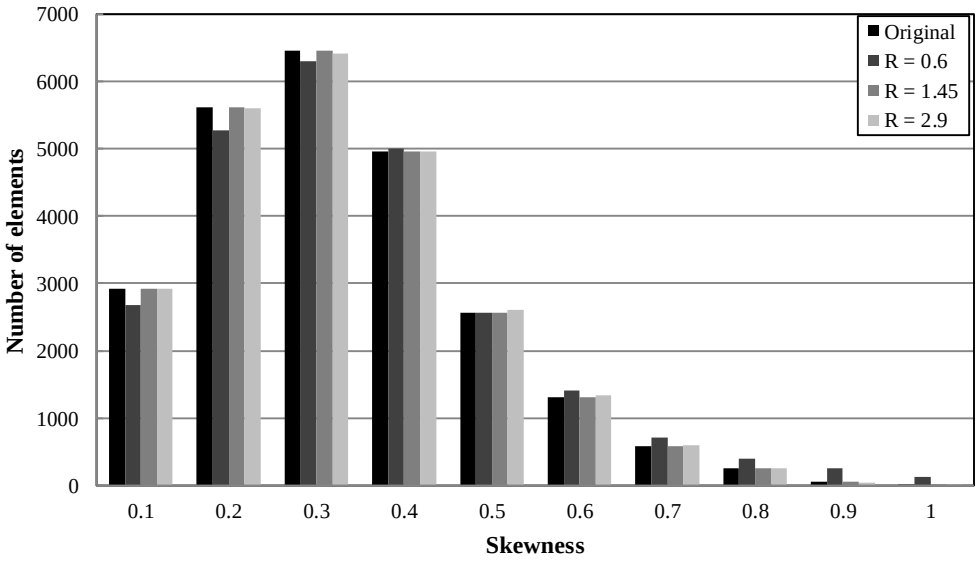


Figure 36. Histogram of the skewness of the original mesh and deformed mesh using different support radii.

Rectangular Supercritical Wing (RSW)

To assess the proposed approaches on a real-life FSI application, the computational time savings for the presented RBF interpolation with greedy algorithm and incremental approach, and traditional RBF interpolation with greedy algorithm are compared using the rectangular supercritical wing. RSW test case was used at the Aeroelastic Prediction Workshop sponsored by the Structural Dynamics Technical Committee, American Institute of Aeronautics and Astronautics (AIAA) [134, 135]. This wing has a span of 48 inches and root chord length of 24 inches. The mesh used for this analysis is one of the meshes provided for the Aeroelastic Prediction Workshop and is available to down load from the workshop website[†]. A fully tetrahedral mesh was selected for this test case as shown in Figure 37. The definition used for the skewness results in approximately 34% of the total number of elements in the original mesh having a skewness of 1.0 due to the highly skewed tetrahedral elements within the boundary layer; see Figure 38. The fluid domain has the dimensions of 4824x2400x4800. The mesh consists of 17,453,792 cells, 2,944,006 nodes, and 56,272 boundary nodes. A direct RBF interpolation technique will require operations of the order of $(56{,}272)^3$ to evaluate the interpolation coefficients and operations of the order of 2,944,006 x 56,272 to evaluate the deformation for all interior nodes. In addition, this process will be repeated for each time step, which will make it impractical for FSI analysis.

[†] https://c3.nasa.gov/dashlink/static/media/other/RSW.htm

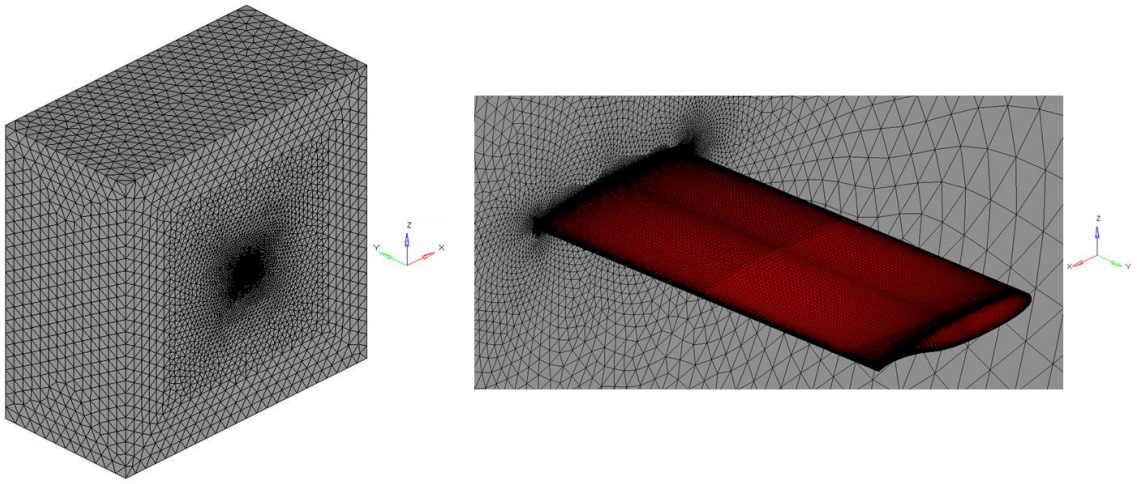


Figure 37. RSW volume mesh (left) and rsw wing view (right).

The RSW structure is assumed to be fixed at root of the wing and deflected using sinusoidal shape function. Three deformation angles are tested (1°, 3.5°, and 5°) to assess the performance of the presented approach. In these simulations, a specified tolerance of $5x10^{-4}$ for the greedy algorithm and a support radius of 1200 units are used. The support radius was calculated as half the distance between the furthest fluid node and any boundary node.

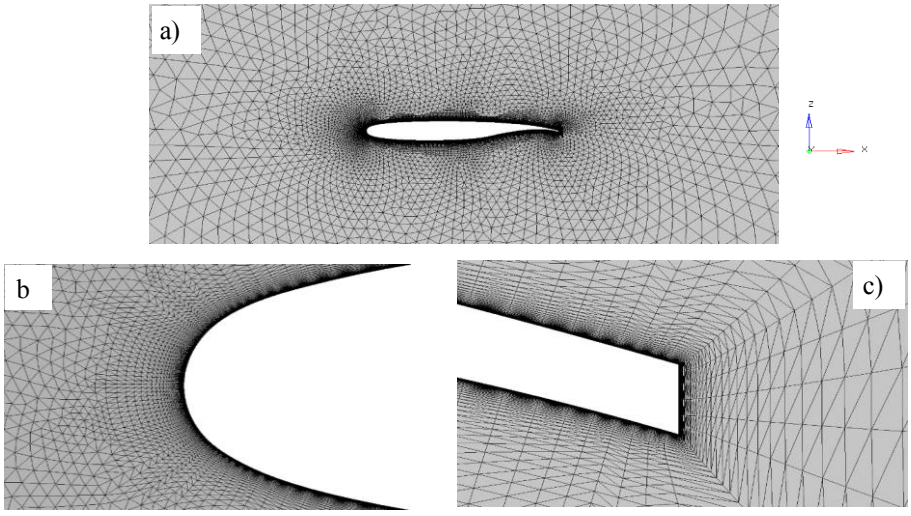
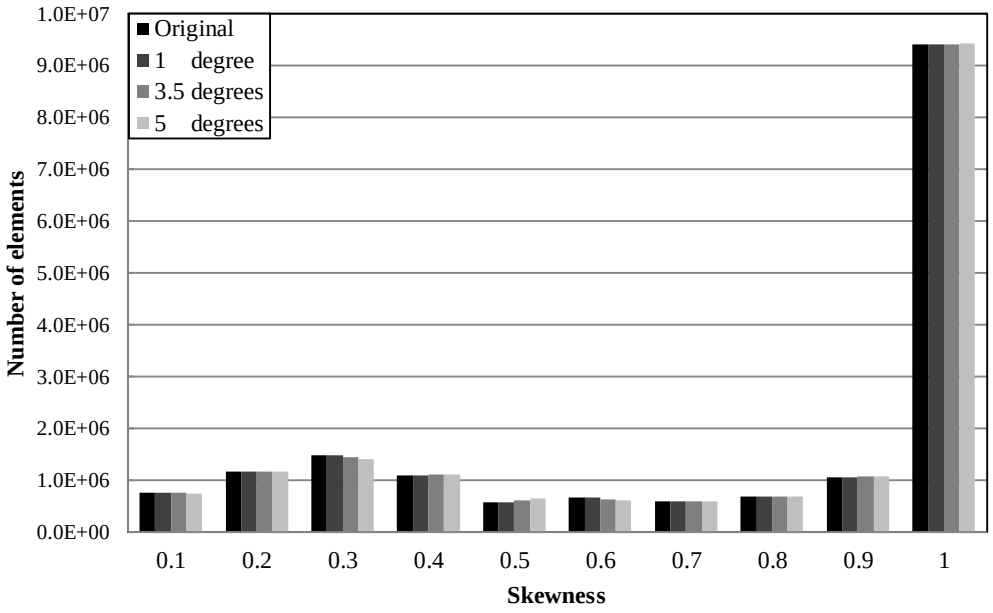


Figure 38. Highly skewed elements within the boundary layer: a) mounting wall normal view, b) zoom into the wing leading edge, and c) zoom into the wing trailing edge.

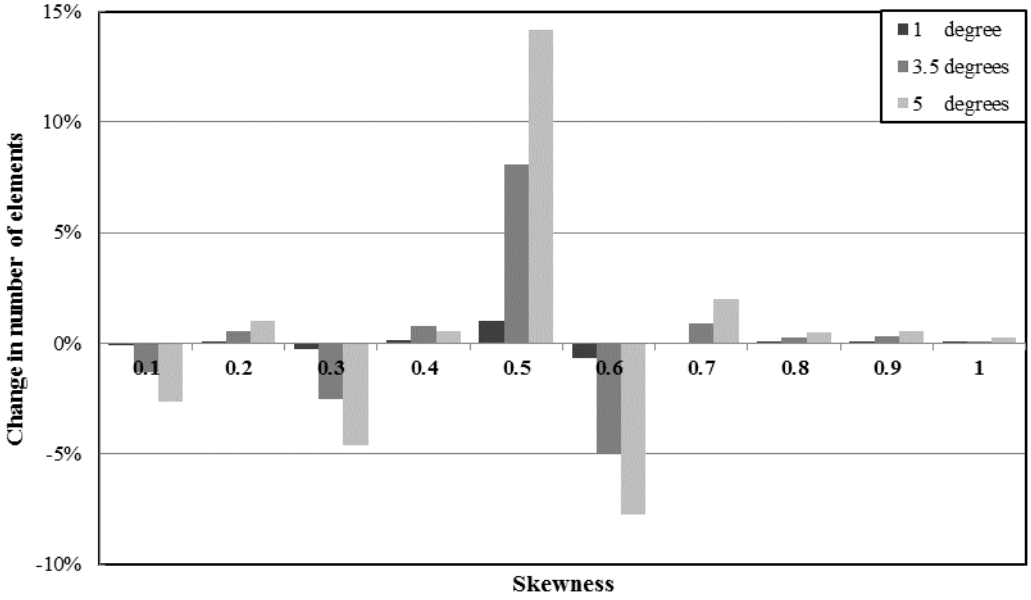
Figure 39 shows no noticeable change in the skewness before and after the deformation. However, to get a better understanding about the effect of the magnitude of boundary deformation on the mesh quality, the percentage change of number of elements with respect to the skewness before and after the deformation is plotted in Figure 39. From this graph, it is evident from the figure that the new distribution of skewness is within an acceptable range and the number of elements that became highly skewed after the deformation is below 0.1% for 3.5 degrees deformation and below 0.2% for 5 degrees deformation. Furthermore, all the deformed meshes have been tested and neither negative volume cells nor cells degeneration has occurred. Pictorial views of the mesh after deformation are shown in Figure 40. The computational costs and number of selected centers for the full LU decomposition method compared against the incremental matrix inversion and LU decomposition based approaches are compared in Table 9. The reason for using LU decomposition method as the base method for comparison instead of using an iterative method is that the matrix system is highly ill-conditioned and convergence issues were reported in the literature [115, 131].

It is clear that the number of selected centers and accordingly the corresponding computational time increases for larger deformations. However, it can be noted from the results that the time required for the greedy algorithm with incremental approaches is significantly smaller than that of performing a full LU decomposition. Moreover, the matrix inversion based approach is found to have slightly better performance over LU decomposition based approach for all cases. Also the superiority of the incremental approaches over the full LU decomposition increases for larger deformations and

accordingly for larger meshes, as illustrated by the increase of the computational time saving with the increase of the deformation angle.



(a)



(b)

Figure 39. (a) Mesh skewness versus number of elements and (b) mesh skewness versus percentage change in number of elements.

Table 9

Number of selected centers and CPU time in seconds for different deformation angles

Deformation Angle	Tolerance	No. of Centers	Greedy Algorithm Time			Total Saving %	
			Full LU	Matrix Inv.	LU Decomp.	Matrix Inv.	LU Decomp.
1.0°	$5.0E^{-4}$	285	10.31	5.29	5.32	48.7%	48.4%
3.5°	$5.0E^{-4}$	570	39.34	18.80	18.88	52.2%	52.0%
5.0°	$5.0E^{-4}$	703	78.84	25.48	25.62	67.7%	67.5%

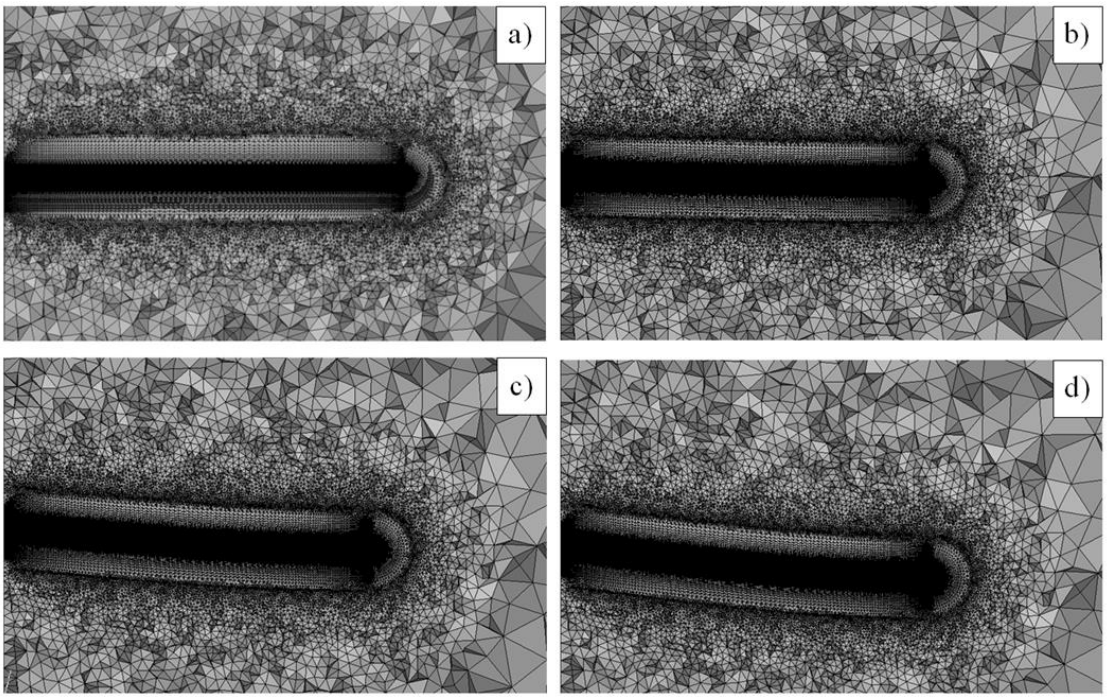
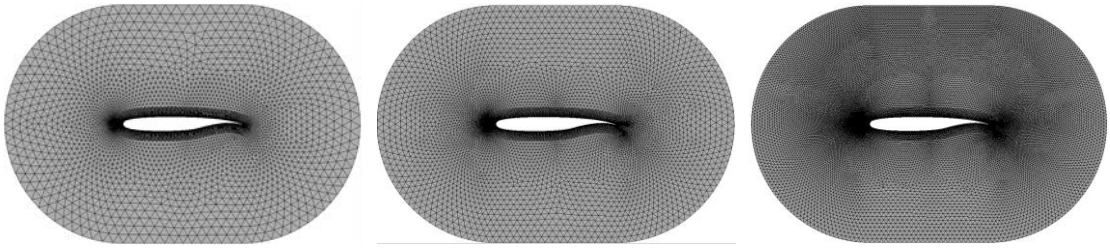


Figure 40. Mid YZ-plane: a) undeformed, b) 1.0° deformation, b) 3.5° deformation, b) 5° deformation.

Bending and Twisting RSW Deformation

The computational time savings for hybrid mesh topologies are demonstrated using a combined bending and twisting deformations of the RSW case. Three different meshes were used for this study. The count of boundary and interior nodes and the count of cells for each mesh are given in Table 10. These meshes were also attained from the Aeroelastic

Prediction Workshop website. Figure 41 illustrates the different refinements for the three meshes by visualizing a part of side wall around the wing.



(a) **(b)** **(c)**

Figure 41. Meshes for three different refinements (side wall view): (a) coarse mesh, (b) medium mesh, (c) fine mesh.

Table 10

Count of boundary and interior nodes for the three mesh refinements

	No. of Boundary Nodes	No. of Interior Nodes	No. of Cells
Coarse	30,146	1,325,155	1,355,301
Medium	60,101	3,062,697	3,122,798
Fine	134,754	8,476,975	8,611,729

The deformation of the wing is specified as a combination of a vertical bending displacement and a twist using the following formula.

$$\Delta z = \delta_0 \, sin\left(\frac{y}{2b}\pi\right); \qquad \delta_0 = 0.5c \tag{91}$$

$$\varphi = \varphi_0 \, sin\left(\frac{y}{2b}\pi\right); \qquad \varphi_0 = 30^o \tag{92}$$

Where y is the coordinate along the span of the wing, z is the coordinate normal to the wing, b is the span of the wing, c is the root chord length, and φ is the twist angle. A comparison of the original and the deformed wings is shown in Figure 42. This test case represents a real-application of FSI problem since it involves more complicated

deformations with more than 130 thousand boundary nodes and 8.4 million interior nodes. The focus here will be directed toward determining which incremental approach provides better performance.

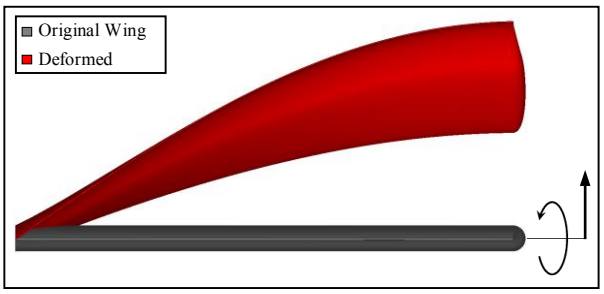


Figure 42. Side view of the wing before and after deformation.

In this test case, the support radius is taken as four times the span of the wing (220 units). Therefore, the deformation is confined within 220 units away from all boundaries. The error tolerance for stopping the greedy algorithm is taken as 5.0×10^{-4}. Figure 43 compares the CPU time required by the greedy algorithm using the matrix inversion and LU decomposition based approaches for each of the three meshes. For all meshes the matrix inversion based approach required slightly less CPU time and resulted in average time saving of 5.5% over LU decomposition based approach.

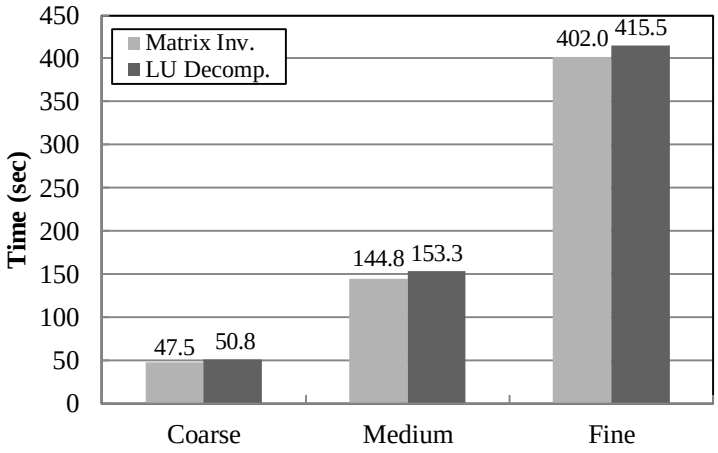


Figure 43. CPU time's comparison for the two proposed incremental approaches.

It should be mentioned that in some cases, due to round off error and slight variation in solvers accuracies, matrix inversion and LU decomposition based approaches do not pick exactly the same number of centers. However, the difference in the final number of selected centers is within the range of ±5 centers. Figure 44 shows the convergence trends for the case of the fine mesh. The convergence trends for both algorithms are identical for all test cases. However, the variation in the total number of centers occurs at the end when a few centers are added to reach the predefined tolerance. Therefore, for ensuring fair time comparison, in such case, both algorithms were forced to pick the same subset of centers with the same sequence.

Moreover, the matrix inversion based approach shows unstable behavior when reducing the error tolerance below $1x10^{-4}$, unlike the LU decomposition based approach (see Figure 45). This instability is due to the accumulation of numerical errors for the matrix inverse calculation during the greedy iterations and which results in error in the weight calculation. The matrix inverse at the current level is calculated as a function of the matrix inverse from the previous step through the calculation of k and ξ in equation (85).

Therefore, any error in the calculation of the inverse in the previous step will be carried to the next level. Based on this observation, it is recommended to use the LU decomposition based approach for the greedy algorithm, even though the CPU time requirement is slightly higher.

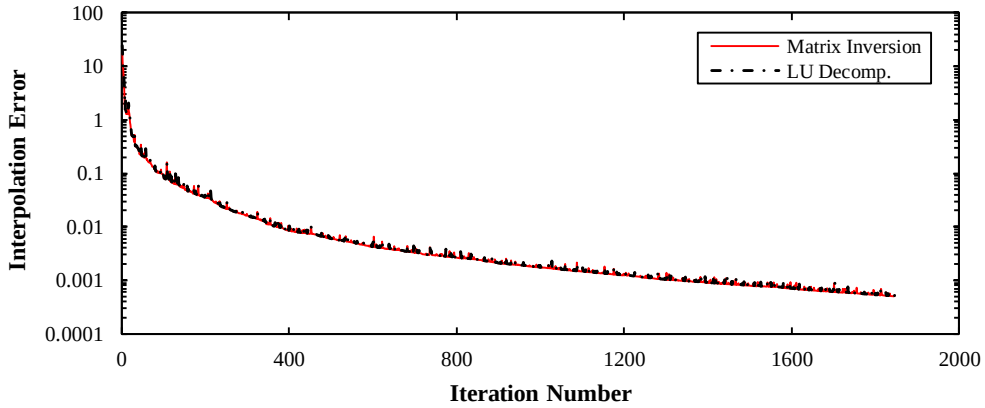


Figure 44. Convergence history of error versus number of iterations.

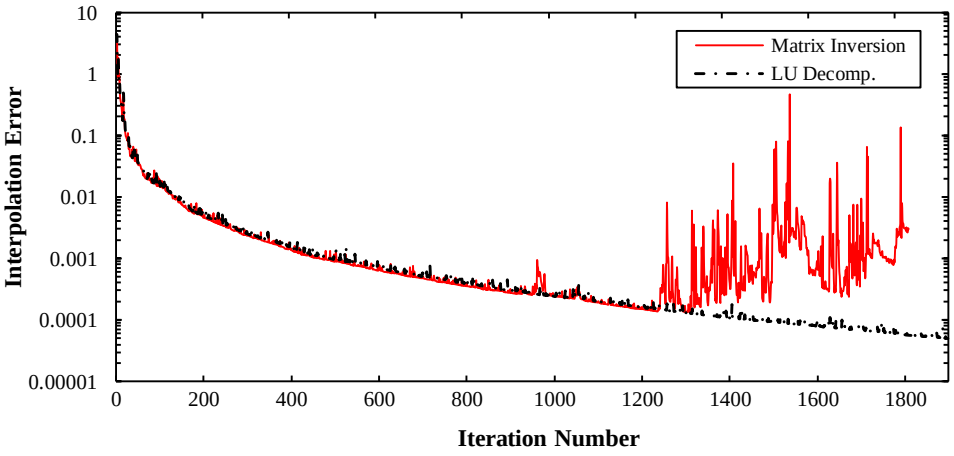


Figure 45. Convergence history of error versus number of iterations for smaller tolerance.

CONCLUSIONS

An efficient mesh deformation technique using radial basis functions by combining the advantages of merging the use of both greedy algorithm and incremental approach is presented in this chapter. A greedy algorithm is used to reduce the number of centers used for the RBF interpolation and an incremental approach is used for the inversion of the matrix system during each greedy iteration. Two different incremental approaches were implemented and tested: 1) Matrix inversion based, and 2) LU decomposition based. The use of the incremental approach decreased the computational complexity of solving the system of equations within each greedy algorithm's iteration from $O(n^3)$ to $O(n^2)$. This technique does not need any cell, face or edge connectivity information, and it depends only on the set of points and the boundary deformation. Therefore, it could be efficiently parallelized. However, in this study, the proposed approaches have not been implemented in a parallel framework.

Benchmark test cases with four different analytic deformations are used to evaluate the performance of the presented approach. The results from the numerical experiments showed that the number of centers required to perform the interpolation is independent of the total number of nodes and mainly depends on the deformation. This makes the technique optimal for fluid-structure interaction simulations where the meshes are very fine. Moreover, the algorithm's order of accuracy is also independent of the total number of nodes. Results are also presented for mesh deformations for deflections of a cantilever beam and a rectangular supercritical wing. These simulations showed both proposed incremental approaches take significantly less CPU time as compared to the traditional full LU decomposition. The present results show that improvement in CPU time saving

increases as the number of selected centers for RBF increases. Moreover, the matrix inversion based approach showed instability issues when error tolerance is less than $1x10^{-4}$. Therefore, it is recommended to use the LU decomposition based approach for the greedy algorithm, even though the CPU time requirement is slightly higher.

CHAPTER 6

OBJECTIVE 4: FLUID-STRUCTURE COUPLING

INTRODUCTION

This chapter summarizes the coupling process between the three main elements of the FSI analysis framework, namely 1) CFD solver, 2) CSD solver, and 3) Mesh deformation module. Once the newly developed CSD methodology as well as the mesh deformation module had been verified independently, they were coupled to the existing in-house CFD code. This task was greatly simplified by the care taken to implement the CSD solver in a manner consistent with the CFD solver. The data transfer procedures between the three components of the FSI framework are described in this chapter. The general procedure for solving coupled FSI problems is shown in Figure 46. The FSI procedure starts with reading the CSD and CFD meshes, calculating various mesh parameters (e.g. areas centroids, volumes, etc.), and constructing mapping information between CSD-CFD interface faces. Then a steady state CFD simulation is run in order to obtain a fully developed converged solution. Afterwards, a global time step is determined and stresses are transferred from CFD mesh to CSD mesh. Then a single FSI time step is performed which includes a CSD step, a mesh deformation step, and a CFD step. Finally, the time level is updated and the stopping criteria is checked to decide whether more steps are needed or not.

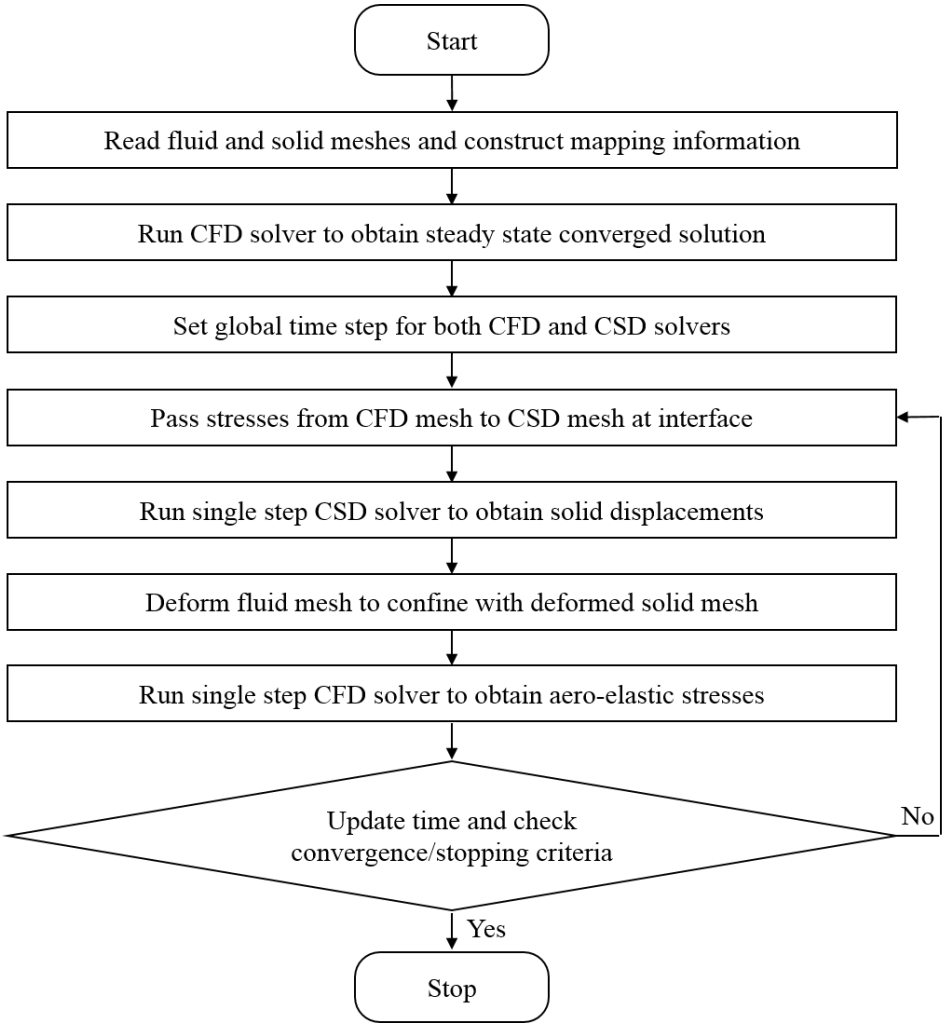


Figure 46. Coupled FSI procedure.

DATA TRANSFER

Since the CSD methodology has been implemented in a consistent manner with the existing CFD numerical approach, both solvers are storing the solutions at the cell-centers. This implies easier data transfer between the two solvers. Moreover, the numerical compatibility between the solvers implies same mesh requirements for both domains. Therefore, an identical surface meshes at the interface will be used for both CSD and CFD domains. Thus, a simple mapping module is needed to map each CFD face/node ID to its

corresponding CSD face/node ID at the interface. This module's function is to create two pointer arrays. The first array maps the CFD interface faces to their corresponding CSD interface faces. While the second array maps the CFD interface nodes to their corresponding CSD interface nodes. The former array is used to transfer the CFD interface stresses to the CSD interface. The later array is used to transfer the CSD nodal deflections to the CFD nodes at interface, as a preparation for the mesh deformation process. Since the RBF mesh deformation technique does not change the mesh connectivity, there is no need to update the mapping information during the simulation and the mapping module runs only once at the beginning of the FSI simulation.

The fluid exerts two types of stresses into the structural interface: pressure and shear stress. CFD solver computes the pressure as one of the flow variables. However, the wall shear stresses are not calculated by default. Thus, the velocity gradients at each interface face must be calculated by knowing the velocity gradients stored at the neighbor cell. Then these face gradients are multiplied by the fluid viscosity and projected on the face normal direction resulting in the shear traction stresses. The structural traction stress is calculated as

$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = -\frac{\mu}{2} \begin{bmatrix} 2\frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} + \frac{\partial v_2}{\partial x} & \frac{\partial v_1}{\partial z} + \frac{\partial v_3}{\partial x} \\ \frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y} & 2\frac{\partial v_2}{\partial y} & \frac{\partial v_2}{\partial z} + \frac{\partial v_3}{\partial y} \\ \frac{\partial v_3}{\partial x} + \frac{\partial v_1}{\partial z} & \frac{\partial v_3}{\partial y} + \frac{\partial v_2}{\partial z} & 2\frac{\partial v_3}{\partial z} \end{bmatrix} . \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{93}$$

or

$$\vec{t} = -\frac{\mu}{2}(\nabla V + \nabla V^T)_{face} \cdot \vec{n} \tag{94}$$

Where $\vec{t}$ is the traction stress, μ is the fluid dynamic viscosity, ∇V is the velocity gradients at the fluid face, and $\vec{n}$ is the fluid face normal. Since the face normal is pointing into the outward direction, each CSD face and its corresponding CFD face at the interface have opposite normal directions. This justifies the presence of the negative sign. The transferred traction, $\vec{t}$, and pressure, p, at each interface face are applied into equation (34) in Chapter 3 in order to calculate the overall traction force, $\vec{F}_t$, acting on each particular face.

It must be noted that all variables within the CFD solver are stored in a non-dimensional form. Thus, the pressure and shear stresses are converted into dimensional form while being transferred to the CSD mesh.

GLOBAL TIME STEP

One requirement for the FSI analysis is to ensure that CFD and CSD simulations are synchronized together. Both solvers must march in physical time simultaneously. Since the time step size for the CSD solver is not limited by a stability condition, unlike CFD solver, the CFD solver time step is used for both solvers during the FSI analysis.

$$\Delta t_{CSD} = \Delta t_{CFD} \tag{95}$$

Where Δt_{CSD} and Δt_{CFD} are the time steps used for CSD and CFD solvers.

However, the CSD pseudo time step must be smaller than the critical pseudo time step and must not be larger than the CSD physical time step. Thus, the CSD pseudo time step is selected based on the following condition

$$\Delta \tau_{CSD} = \text{Min}(\alpha \cdot \Delta \tau_{cr}, \Delta t_{CFD}) \tag{96}$$

Where α is a constant varying between $0 < \alpha < 1$ which ensures the satisfaction of the stability condition in equation (24).

Since both meshes are being deformed through the analysis, the time step stability conditions must be examined and updated if needed every several time levels. The previous condition in equation (96) is revisited after every time step update.

CONVERGENCE CRITERIA

Convergence is achieved when no further change to either fluid or solid solutions are being recorded. This condition is implemented by calculating the L2 norm of the residuals for both of the solvers. The L2 norm is calculated based on the change in solution within all elements for each solver. When L2 norms of both solvers fall below a certain tolerance, which means that there is no change in the solution, the coupled FSI simulation convergence is said to be achieved. If convergence did not occur, the FSI simulation will continue until the maximum physical time is reached.

FLOW-INDUCED CANTILEVER BEAM VIBRATION

To examine the coupled fluid-structure interaction methodology along with the mesh deformation algorithm, the case of a cantilever beam under the effect of incoming flow is investigated. The cantilever beam has the same structural properties as well as the exact dimensions of the cantilever beam case used in Chapter 3. As shown in Figure 47, the beam has a cross section area of 0.2 m x 0.2 m and a height of 2 m. The computational domain is identical to the domain that has been used to test the mesh deformation algorithm in Chapter 4. As also shown in Figure 47, the computational domain has the dimensions of 8 m x 1.2 m x 2.5 m and the beam is placed at 2 m from the inlet.

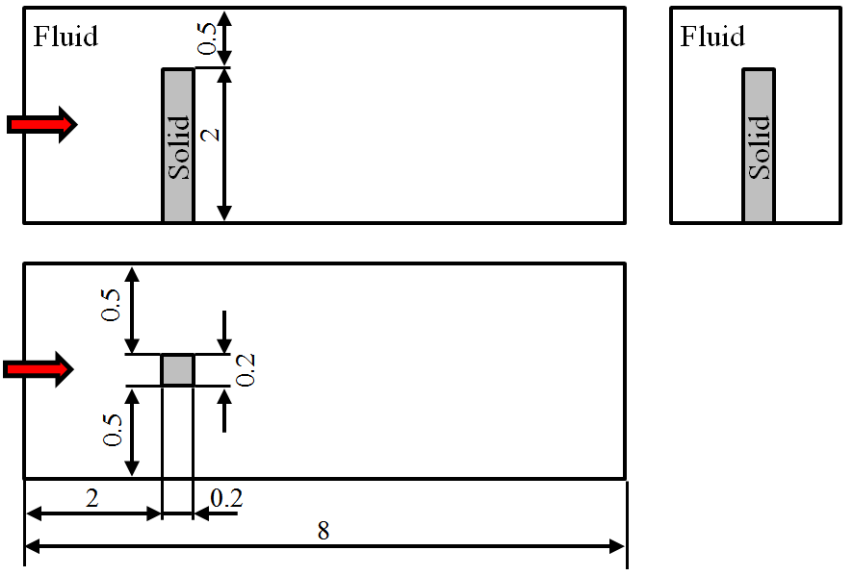


Figure 47. Cantilever beam vibration computation domain.

The fluid was assumed to be water with a density of 1000 kg/m^3. The fluid flow was assumed to be inviscid and incompressible. Regarding the boundary conditions, all walls, including interface walls, were assigned as slip walls, thus no boundary layer was needed. An inlet velocity boundary and an atmospheric pressure outlet boundary were used. Four different inlet velocities have been analyzed, i.e. 0.25, 0.5, 0.75, and 1.0 m/s. For the structural side, the same material properties of the cantilever beam have been kept the same as stated in Table 1 in Chapter 3. The faces forming the bottom surface of the beam were assigned as a zero displacement boundary. The remaining boundary faces were considered interface faces where a specified traction boundary was applied.

Unstructured tetrahedral meshes has been used for both domains. Figure 48 shows the generated surface meshes for both fluid and structural domains. The CFD mesh consists of 105,568 cells, 206,408 faces, and 19,164 nodes. The CSD mesh consists of 20,346 cells, 37,114 faces, and 5,169 nodes. The surface meshes for both domains at the interface are identical.

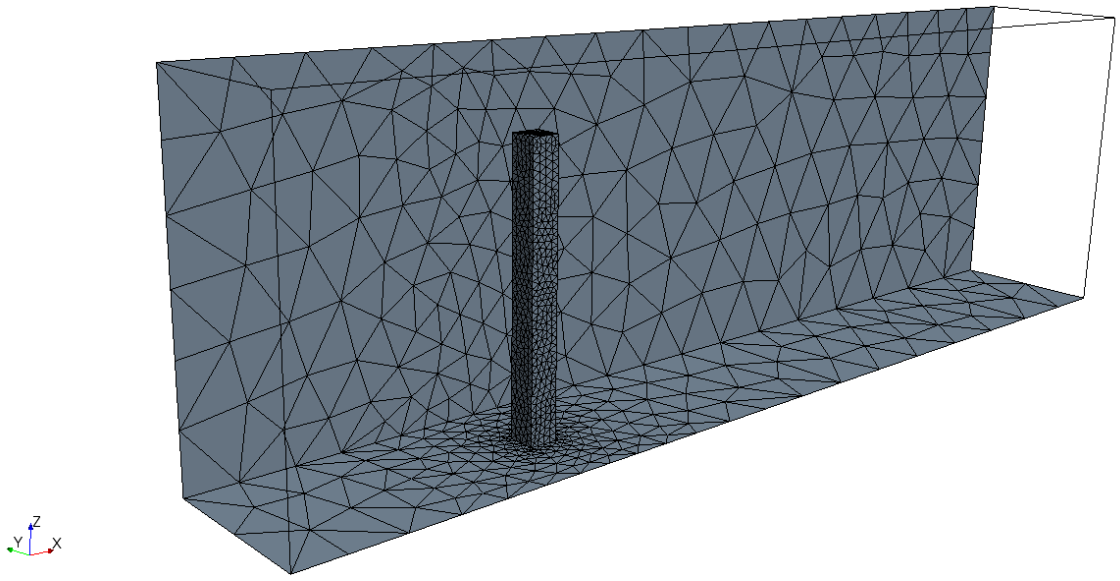


Figure 48. Unstructured tetrahedron mesh for the induced flow cantilever beam case.

Steady State CFD Analysis

A fully-developed steady state fluid flow was obtained prior to starting the FSI analysis. Since the fluid flow is assumed to be inviscid and incompressible, the same pressure profile has been obtained for the different inlet velocity cases. The difference can be noticed in the magnitude of pressure for each case. Figure 49 shows the maximum and minimum pressure on the beam surface for the analyzed range of inlet velocities.

Figure 50 shows the pressure distribution on the surfaces of the beam after obtaining a steady state CFD condition. It is obvious that the pressure is highest in magnitude and positive at the surface facing the incoming flow which pushes the surface opposite to its surface normal direction (positive x-direction). Moreover, it is noticeable that the pressure around the edges tends to fall to a negative value which is pulling the surfaces toward its normal outward direction. However, opposite surfaces will cancel the displacement in y-direction and z-direction.

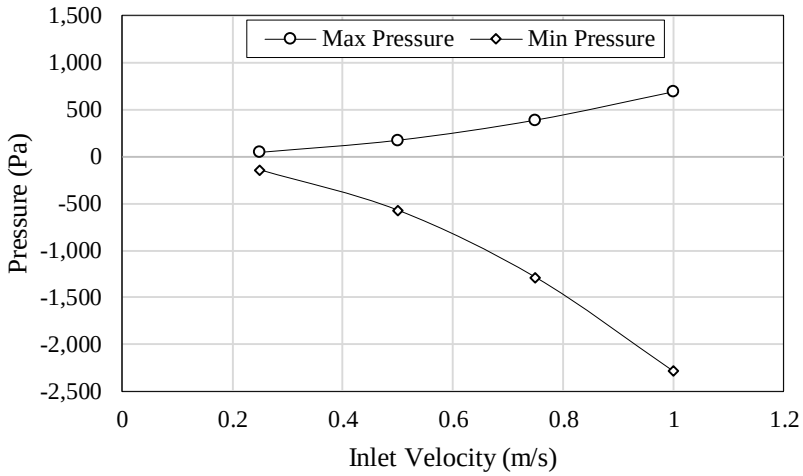


Figure 49. Maximum and minimum pressure versus inlet velocity.

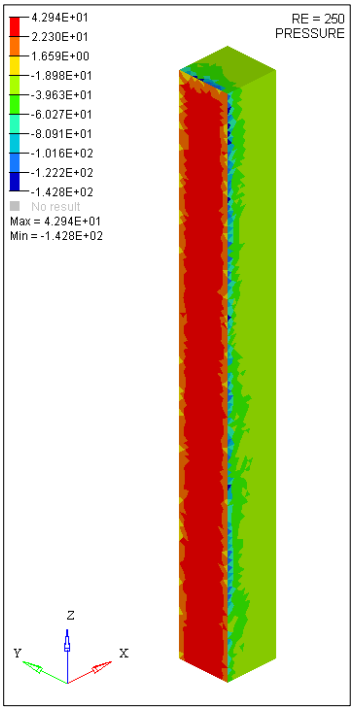


Figure 50. Pressure distribution on beam surface after obtaining converged CFD solution.

Figure 51 shows the pressure and velocity distributions of a mid-plane within the fluid domain. Identical pressure and velocity profiles were obtained for different inlet velocities. The pressure is highest on the beam surface facing the incoming flow and lowest behind the beam.

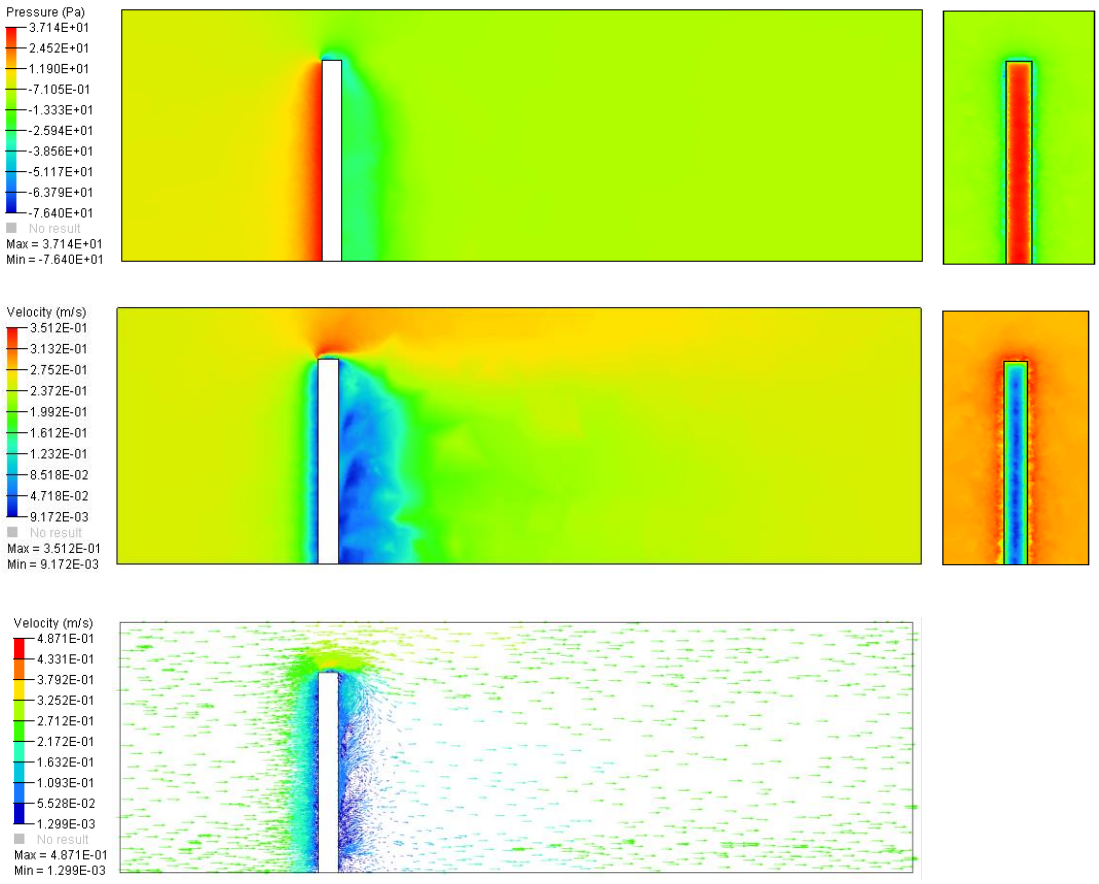


Figure 51. Mid-plane CFD solution for V = 0.25 m/s: pressure distribution (top), velocity distribution (middle), and velocity vectors distribution (bottom).

FSI Analysis

After obtaining converged steady state solutions successfully, the required FSI parameters were initialized. These parameters include setting a global time step to insure CFD and CSD solvers stability, constructing of mapping information needed to transfer stresses between the two meshes, and setting the mesh deformation's support radius and error tolerance.

Specified traction boundary condition was assigned to all CSD mesh boundary faces except to the faces on the bottom which were of a zero displacement type. The specified traction values were updated based on equation (34) in Chapter 3 after each CFD time step. The physical time step, Δt_{CSD}, was set to $1x10^{-3}$ seconds and the pseudo time step was set to $1x10^{-5}$ which satisfies the solver stability condition. Within each physical time step a maximum of 500 pseudo iterations were allowed. The order of pseudo iteration residual was $1x10^{-10}$.

An RBF support radius of 1.0 m has been used for this test case. However, regarding the error tolerance, it has been found that using a fixed error tolerance is not practical for such FSI analysis. Since the beam is expected to have a wide range of deflections, using a fixed error tolerance that suits the minimum deflection will cause the greedy algorithm to pick almost all boundary nodes when the deflection reaches its peak. This will lead to a very time consuming mesh deformation process. Instead, a relative error tolerance has been used in order to relate the error tolerance to the maximum deflection predicted at each time step. In this case, the error tolerance has been set as 0.1% of the maximum total beam deflection. At the same time, the minimum allowed error tolerance was set to $1x10^{-6}$ in order to avoid expensive mesh deformation at the beginning of the vibration cycle.

Furthermore, it is worth mentioning that the fluid mesh is being deformed based on the original undeformed mesh at every time step. Hence, instead of deforming the mesh based on the previous time step location and the newly calculated deflection (ΔU), the mesh is deformed based on the undeformed location and the total deflection (U). This approach avoids the accumulation of interpolation error.

Each case has been run for a total physical time of three seconds. Figure 52 shows the recorded tip deflections for each case throughout the analysis. It is clear that the predicted deflections preserved the natural frequency of the cantilever beam. Moreover, as expected, increasing inlet velocity led to predicting higher deflections. Table 11 lists the predicted vibration frequency and maximum deflection for each case. By comparing the predicted frequency against the beam natural frequency (1.0 Hz), the maximum error was determined to be below 2.7% within all cases.

By increasing the inlet velocity the flow momentum tend to resist the beam vibration and act as a damping effect. For higher inlet velocities, i.e. V = 0.75 m/s and V = 1.0 m/s, the damping effect is more substantial. This damping effect is expected to vanish after certain number of cycles when the vibration response reaches a steady state.

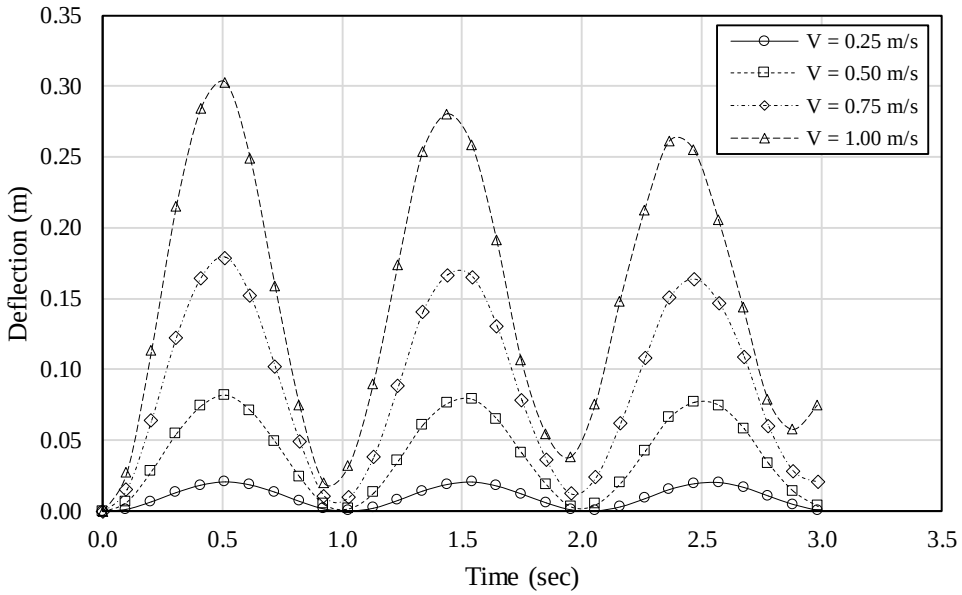


Figure 52. Tip deflection histories for four inlet velocities.

Table 11

Vibration frequency and maximum deflection for different inlet velocities

	Inlet Velocity (m/s)			
	0.25	**0.50**	**0.75**	**1.00**
U_{max} (m)	0.021	0.082	0.180	0.305
Frequency (Hz)	1.028	1.004	0.994	0.980
Frequency Error	2.7%	0.4%	0.6%	2.0%

Figure 53 shows the velocity distributions around the cantilever beam at simulation time of 0.5 seconds for the different inlet velocities. It is clear from the figure that increasing the inlet velocity leads to higher beam deflections which reduces the beam resistance to the flow. As can be seen on the side view velocity distributions, the area showing minimum velocity is shrinking with the increase of the inlet velocity.

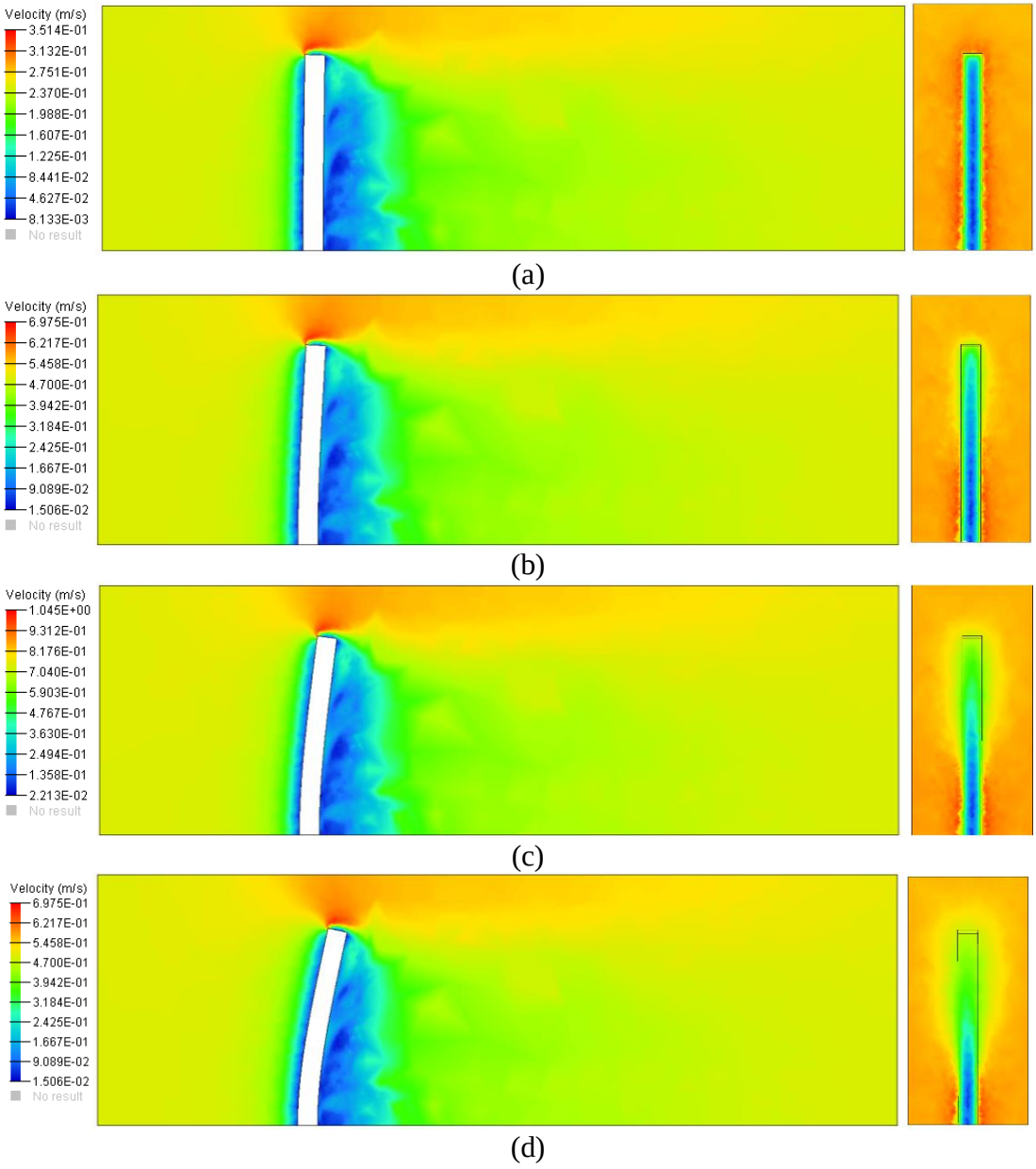


Figure 53. Velocity distribution of cutting planes normal to Y (right) and X (left): (a) V = 0.25 m/s, (b) V = 0.5 m/s, (c) V = 0.75 m/s, and (d) V = 1.0 m/s.

Mesh Deformation Performance Analysis

The RBF based mesh deformation approach with incremental solver showed a very efficient and robust performance. Using a varying error tolerance, that is a percentage of the maximum deflection, improved the algorithm's performance dramatically. For all cases, the total number of the nodes in the deformed region is 19,164 and the surface mesh consists of 4730 points. Only an average of 470 surface points, i.e. around 10%, were selected to drive the volume mesh motion. A support radius of 1 m was selected, which gave a deforming region that was large enough to reasonably accommodate the motion, as shown in Figure 54. It is noticeable in this figure that for higher beam deflections, the mesh cells located above the beam tend to become more skewed. This issue happened because the mesh is coarse on the top wall, and the gap between the top wall and the beam tip is relatively small. This can be avoided by refining the mesh at this region.

Figure 55 shows the percentage of selected centers versus simulation time. It is noticeable that the number of selected centers increases up to 16% at the beginning of each cycle. Since the predicted deflections tend to be relatively small at the beginning of each cycle, the algorithm picks more nodes in order to achieve the error tolerance. However, limiting the minimum value of the error tolerance to be $1x10^{-6}$ prevents the algorithm from keep adding more points at this stage of the deflection cycle.

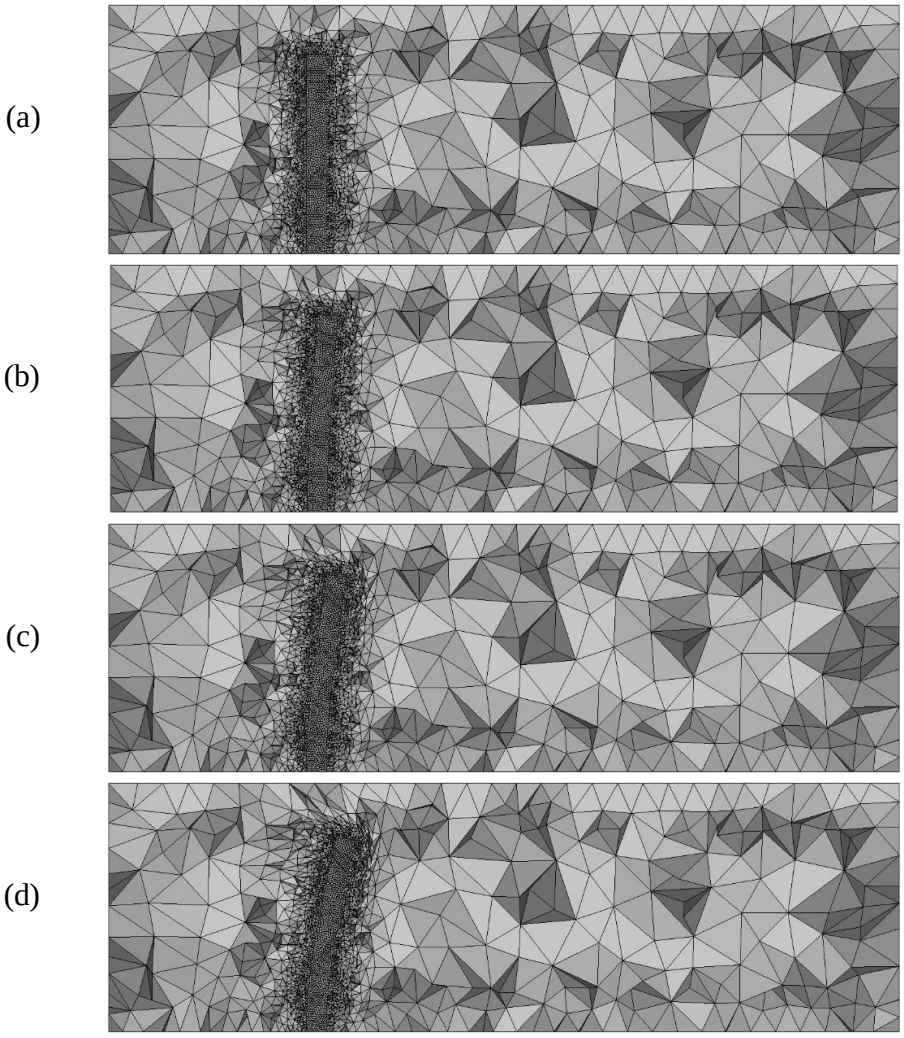


Figure 54. Mid-plane section of the CFD mesh for V = 1.0 m/s at different simulation times: (a) 0.0 sec, (b) 0.2 sec, (c) 0.3 sec, and (d) 0.5 sec.

The RBF interpolation error is calculated based on the known deflections of the unselected centers. The interpolation error percentage is shown in Figure 56 for different inlet velocities. It can be seen that the error is fixed at 0.1%. However, when the maximum deflection reaches a very small value (less than $1x10^{-3}$ m in this case), the error exceeds 0.1%, since the tolerance is not allowed to fall below $1x10^{-6}$. This behavior can be seen at the beginning of each deflection cycle for the case of V = 0.25 m/s where the deflections are the smallest.

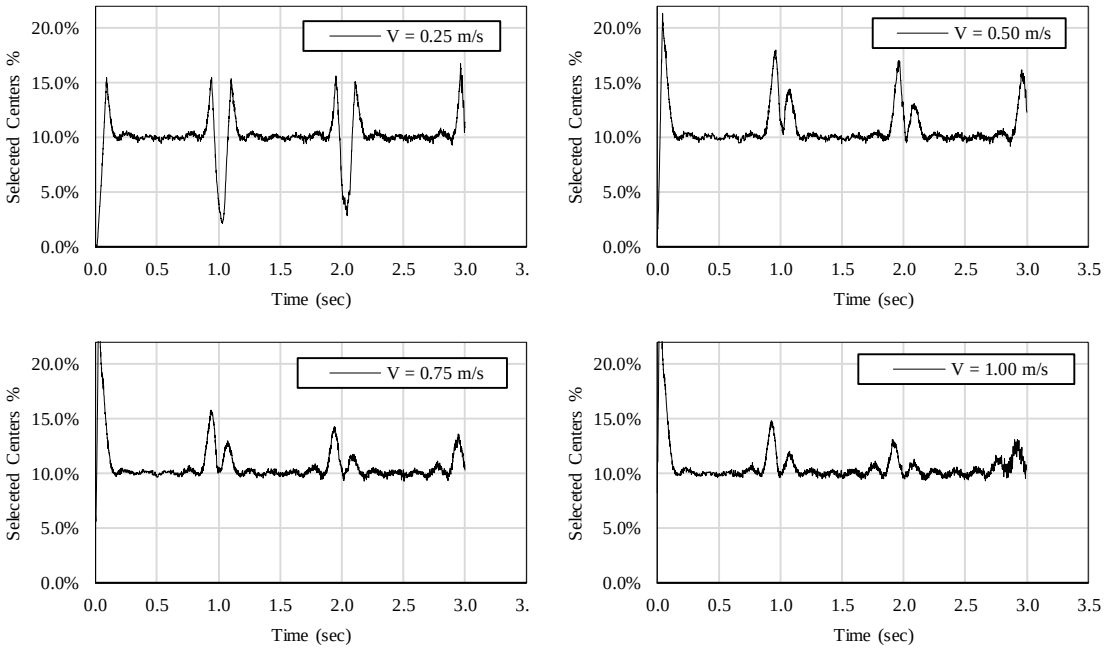


Figure 55. Percentage of selected centers versus simulation time for four inlet velocities.

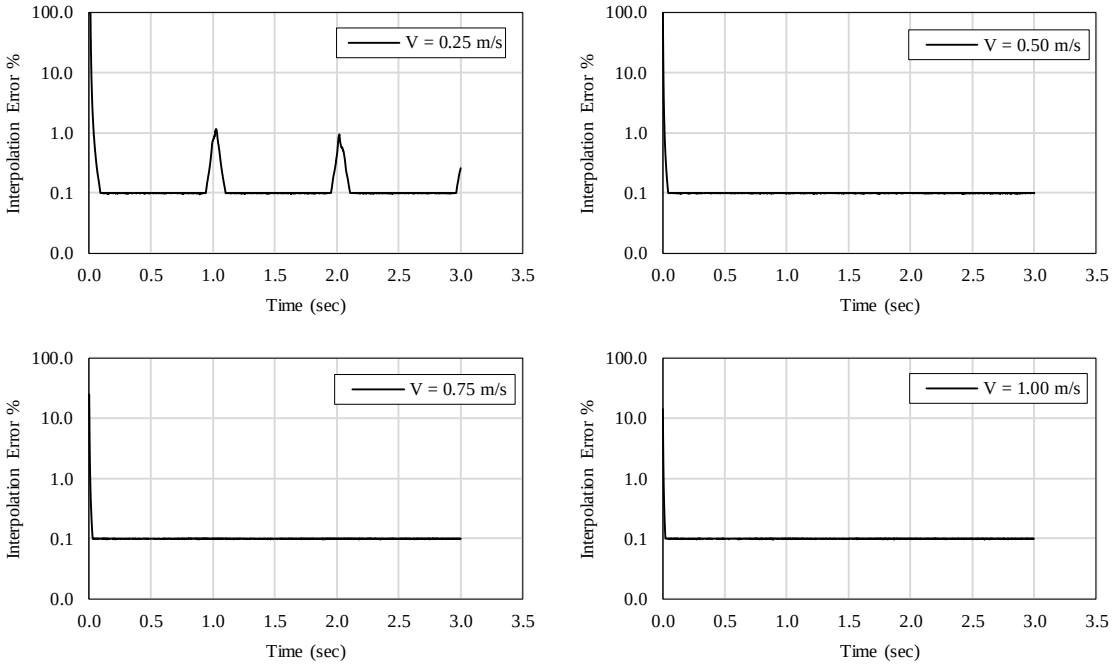


Figure 56. Percentage of interpolation error versus simulation time for four inlet velocities.

CPU Time Analysis

A CPU time analysis was performed for all cases. The total CPU time for the FSI solver was distributed into three main elements: 1) CSD solver CPU time, 2) CFD solver CPU time, and 3) RBF mesh deformation CPU time. The cost unit is considered to be the CPU time required for one time step. Table 12 presents the CPU time in seconds for the three element of the FSI solver for different inlet velocity values. The CSD solver consumes 95% of the total required time. This performance was expected since the CSD solver requires performing many additional inner iterations when compared to CFD solver. The costly implementation of the Green-Gauss theorem for calculating the displacement gradients within the CSD solver is not suitable for such high number of inner iterations. On the other hand, the RBF algorithm shows an efficient computational time with an average of 1.4 seconds per FSI time step. The minimum CPU time through the FSI analysis of this test case was consumed by the CFD solver. This was expected due to the low number of elements within the fluid mesh. However, it is expected for the CPU time consumed by the CFD solver to exceed the time consumed by the RBF module when larger and more complex test cases are being simulated.

Table 12

Average CPU time in seconds for one FSI time step

	V = 0.25 m/s	**V = 0.5 m/s**	**V = 0.75 m/s**	**V = 1.0 m/s**	**Average**
CSD	47.414	48.684	48.967	48.983	48.512
CFD	1.052	1.058	1.061	1.062	1.058
RBF	1.285	1.471	1.448	1.458	1.415
FSI (Total)	49.751	51.213	51.476	51.502	50.985

FSI RESULTS VALIDATION

There are several benchmark test cases with experimental data available for comparison in the field of aero-elasticity. Among several well-known test cases the Rectangular Supercritical Wing case [134, 135] and the AGARD wing case [136] are considered to be milestones for validating newly developed FSI tools because the experimental data of the analysis are well documented. However, most of the well-documented cases that deal with the flutter wing problem require larger meshes, usually millions of elements. Based on the current serial development of the proposed solver, it is prohibitive to consider such large complicated cases.

One study that investigated a similar flow-induced cantilever beam case was the study by Lorentzon [137]. He used an open source packages, i.e. DEAL.II and OpenFOAM, to create a coupling between a finite element formulation for structural dynamics and a finite volume formulation for fluid dynamics. In his study, the cantilever beam vibration results were compared against empirical data and found to be in a good agreement. The beam has the same dimensions as the previous case, which is 0.2 m x 0.2 m x 2 m. However, the fluid domain dimensions are changed to be 5.2 m x 1.2 m x 2.5 m. The beam is placed at 1.0 m from the inlet and 0.5 m from the side walls while the outlet is placed at 4.0 m from the beam.

Unstructured tetrahedral meshes have been used for both domains. Figure 57 shows the generated surface meshes for both fluid and structural domains. The CFD mesh consists of 182,805cells, 358,027 faces, and 32,816 nodes. The CSD mesh consists of 24,013 cells, 45,317 faces, and 5,264 nodes. The surface meshes for both domains at the interface are identical.

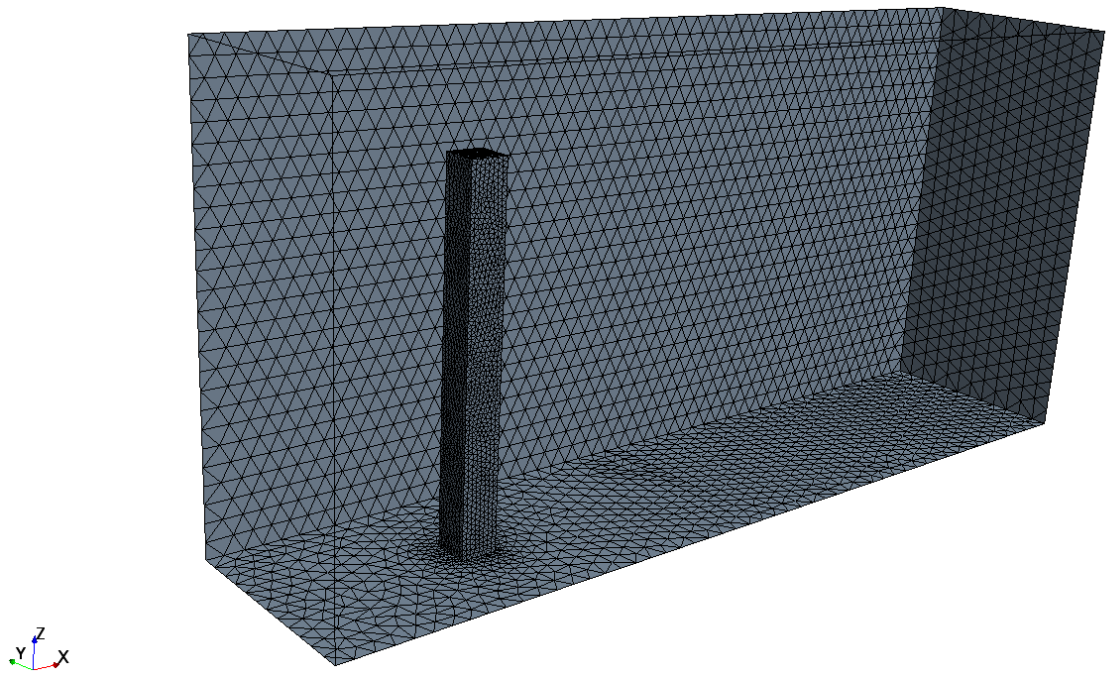


Figure 57 Unstructured tetrahedron mesh for FSI validation case

The fluid was assumed to have a density of 0.1 kg/m^3 and the inlet velocity is set to be 10 m/s. The structure was assumed to have a modulus of elasticity of 20 KPa and a Poisson's ratio of 0.3. Two different structural densities were investigated, 10 kg/m^3 and 50 kg/m^3. The physical time step, Δt_{CSD}, was set to 1x10^{-3} seconds and the pseudo time step was set to 2x10^{-6} seconds which satisfies the solver stability condition for the newly generated structural mesh. Within each physical time step a maximum of 500 pseudo iterations were allowed. The convergence tolerance for the pseudo iteration residual was set to 1x10^{-10} in the simulation.

First, a CFD analysis was run to obtain a fully-developed converged fluid flow solution. Then, the coupled FSI analysis was run for 0.3 seconds of simulation time. Within the RBF mesh deformation algorithm, a support radius of 1.0 m has been used for this test case. The greedy algorithm's error tolerance has been set as 0.1% of the maximum total beam deflection and the minimum allowed error tolerance was set to $1x10^{-6}$ m.

Figure 58 shows the recorded tip deflection throughout the simulation for structural density of 10 kg/m^3. By comparing the obtained results against the results presented by Lorentzon [137], it can be seen that both solutions are in a good agreement. Table 13 compares the predicted maximum deflection and frequency values against the results of Lorentzon [137]. The proposed FV-FSI solver under-predicted the maximum deflection by 7% and predicted almost the exact vibration frequency with error margin of 0.23%.

Since the natural frequency is a function of the beam mass, changing the structure density should affect the frequency of vibration. However, the deflection magnitude, which is a function of the applied load, modulus of elasticity, and area moment of inertia, should remain unchanged. Figure 59 shows the recorded tip deflection throughout the simulation for structural density of 50 kg/m^3. By comparing the obtained results against the results obtained by Lorentzon [137], it can be seen that both solutions are in a good agreement. The proposed FV-FSI solver under-predicted the maximum deflection by 7% and well-predicted the vibration frequency with error margin of 0.16%. It can be seen in Table 13 that the maximum deflection has not been affected by changing the density, and only a change in the response frequency was recorded.

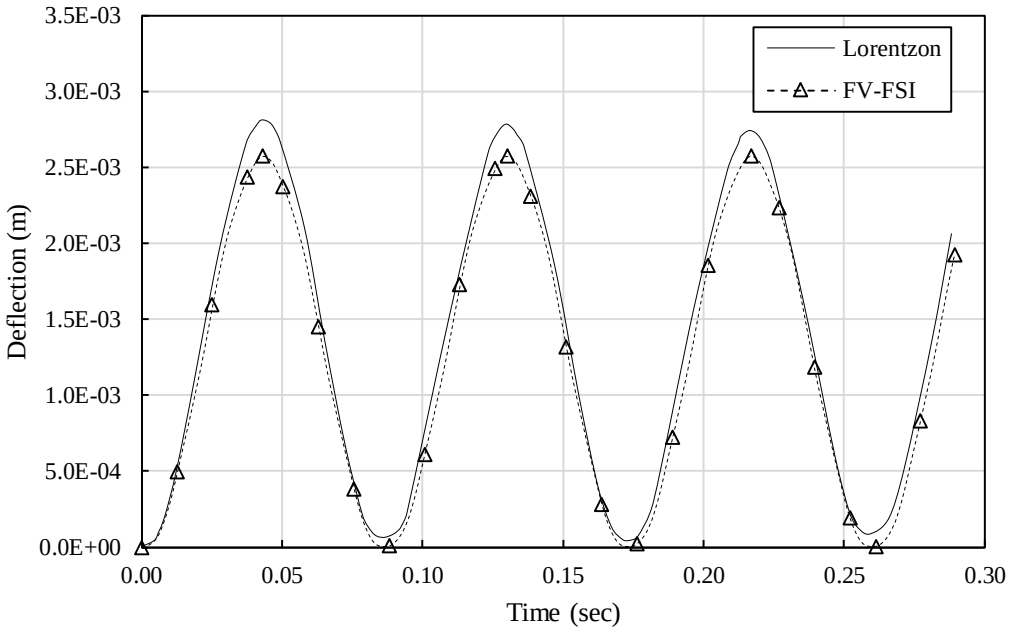


Figure 58 Tip deflection history comparison for ρ_s = 10 kg/m^3.

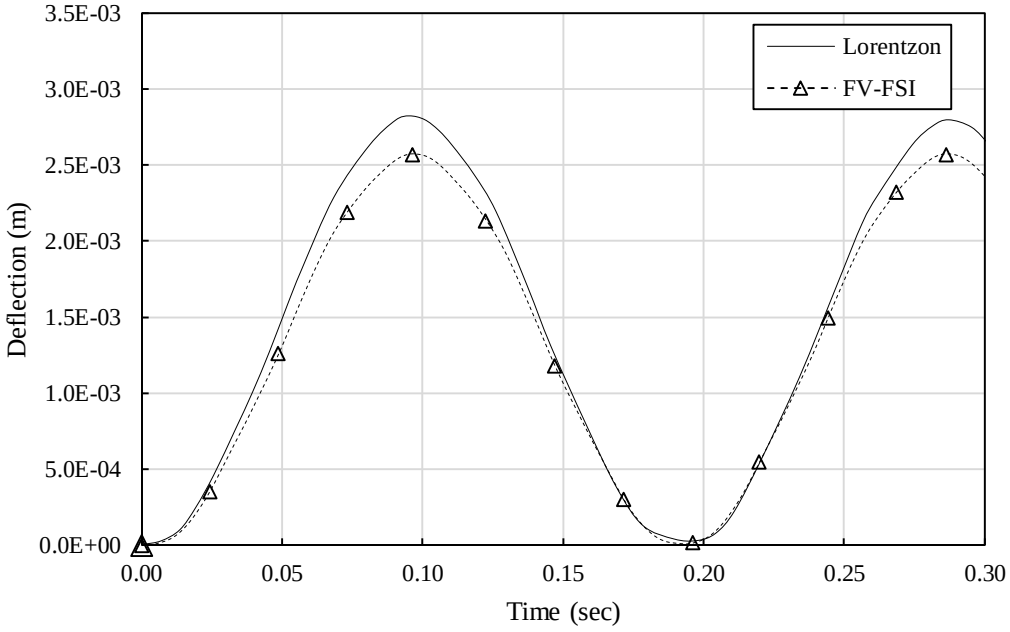


Figure 59 Tip deflection history comparison for ρ_s = 50 kg/m^3.

Table 13

Maximum deflection and frequency comparison

	ρ_s = 10 kg/m^3		ρ_s = 50 kg/m^3	
	U_{max} (m)	Frequency (Hz)	U_{max} (m)	Frequency (Hz)
Lorentzon	0.0028	0.0868	0.0028	0.1926
FV-FSI	0.0026	0.0866	0.0026	0.1929
Error	7.14%	0.23%	7.14%	-0.16%

CONCLUSIONS

This chapter has described the procedure used for coupling the fluid solver with the structural solver and the mesh deformation algorithm. The case of flow-induced cantilever beam vibration has been simulated. Four different flow inlet speeds have been analyzed, i.e. V = 0.25, 0.5, 0.75, and 1.0 m/s. The predicted beam vibrations have preserved the natural frequency for all cases with a margin of error below 2.7%. Furthermore, increasing the flow velocity increases the magnitude of the beam deflection, as expected.

The RBF based mesh deformation algorithm has shown a very robust and efficient performance. On average, only 10% of the total number of boundary nodes have been picked by the greedy algorithm. The interpolation error was limited below 0.1% of the maximum deflection. The use of incremental solver helped to reduce the CPU time significantly. On average, the CPU time of mesh deformation for all cases was below 1.4 seconds per time step.

The behavior of the flow-induced cantilever beam vibrations for two different structural densities were simulated. The results were validated against the FSI results produced by coupling a finite element structural solver with a finite volume fluid solver

provided by Lorentzon [137]. The predicted structural response was found to be in a good agreement for both density values. The proposed FV-FSI solver under-predicted the maximum deflection by 7% and well-predicted the vibration frequency with error margin less than 0.23%.

CHAPTER 7

OVERALL CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

This dissertation has described the development of a loosely coupled three-dimensional fluid-structure interaction methodology. The developed approach has several novel features to improve the overall efficiency, compatibility, and reduce the complexity of simulation process. This study is considered the first to implement the dual time-stepping discretization scheme to solve linear elasticity problems using a cell-centered FVM in three-dimension. The use of matching numerical approach for solving the governing equations for both mediums resulted in straight forward data transfer. In addition, an efficient and robust mesh deformation approach has been developed. The novel integration of incremental solver along with the use of greedy algorithm for mesh deformation process has shown to reduce the computational time dramatically.

Finite Volume Based Structural Solver

A second order cell-centered finite volume approach to model three-dimensional linear elastic structures was developed and validated. The case of a three-dimensional cantilever beam under three different loading conditions was tested. The obtained results were compared against the traditional finite element method and against analytical solutions. The presented finite volume methodology results show a good agreement when compared to analytical and finite element results. Furthermore, the developed numerical

method was tested for large amplitude deflection cases. The load was applied as a diagonal traction acting on the free-end of the beam. The finite volume results well-predicted the dynamic as well as the damped responses. Two different dimensionless load magnitudes were applied to test the developed approach capability of predicting large amplitudes of deflection. The error in the deflection amplitude for the lower load and higher load were 5.5% and 7.19%, respectively, with frequency shifts of 1.0% and 4.8%, respectively.

Based on the obtained results, the proposed finite volume approach can be used as an alternative for the well-established finite element method. The developed finite volume methodology adapts the exact numerical discretization scheme applied in state-of-the art generalized mesh based CFD solvers. This ensures same meshing requirements for the fluid and structural domains, which allows generating identical surface meshes at the interface. This leads to an efficient FSI coupling without the need for data interpolation at the interface between fluid and structural domains.

RBF Based Mesh Deformation

An efficient mesh deformation technique using radial basis functions, by combining the advantages of merging the use of both greedy algorithm and incremental approach, for fluid-structure interaction simulations is developed. A greedy algorithm is used to reduce the number of centers used for the RBF interpolation and an incremental approach is used for the inversion of the matrix system during each greedy iteration. Two different incremental approaches were implemented and tested: 1) Matrix inversion based, and 2) LU decomposition based. The use of the incremental approach decreased the computational complexity of solving the system of equations within each greedy algorithm's iteration from $O(n^3)$ to $O(n^2)$, where n is the number of selected interpolation

centers. This technique does not need any cell, face, or edge connectivity information and it depends only on the set of points and the boundary deformation. Therefore, it could be efficiently parallelized. However, in this study, the proposed approaches have not been implemented in a parallel framework.

Benchmark test cases with four different analytic deformations are used to evaluate the performance of the presented approach. The results from the numerical experiments showed that the number of centers required to perform the interpolation is independent of the total number of nodes and mainly depends on the deformation. This makes the technique optimal for fluid-structure interaction simulations where the meshes are very fine. Moreover, the algorithm's order of accuracy is also independent of the total number of nodes. Results are also presented for mesh deformations for deflections of a cantilever beam and a rectangular supercritical wing. These simulations showed that both proposed incremental approaches save up to 67% of CPU time as compared to the traditional full LU decomposition.

The presented results show that improvement in CPU time saving increases as the number of selected centers for RBF increases. Moreover, the matrix inversion based approach showed instability issues when error tolerance is less than $1x10^{-4}$. Therefore, it is recommended to use the LU decomposition based approach for solving the RBF system, even though the CPU time requirement is slightly higher as compared with the matrix inversion based approach.

Fluid-Structure Coupling

The procedure used for coupling the fluid solver with the structural solver and the mesh deformation algorithm has been described in Chapter 6. The case of flow-induced cantilever beam vibration has been used to validate the developed methodology. Four different flow inlet speeds have been analyzed, i.e. V = 0.25, 0.5, 0.75, and 1.0 m/s, to study the effect of flow velocity on structural deflection. The predicted beam vibrations have preserved the natural frequency for all cases with error below 2.7%. As expected, increasing the flow velocity increases the magnitude of the beam deflection.

Moreover, the RBF based mesh deformation algorithm has shown a very robust and efficient performance. On average, only 10% of the total number of boundary nodes has been picked by the greedy algorithm to perform the mesh deformation. The greedy algorithm's convergence criteria was set to limit the interpolation error below 0.1% of the maximum deflection. The use of incremental solver helped reduce the CPU time significantly. On average, the CPU time for mesh deformation for different inlet velocity cases was below 1.4 seconds per time step on a mesh consists of 19,164 nodes..

Furthermore, the structural dynamic response for the case of flow-induced cantilever beam vibration for two different structural densities was simulated. The results were compared against the FSI results produced by coupling a finite element structural solver with a finite volume fluid solver provided by Lorentzon [137]. The predicted structural response was found to be in a good agreement for both density values. The proposed FV-FSI solver under-predicted the maximum deflection by 7% and well-predicted the vibration frequency with error margin less than 0.23%.

SUGGESTIONS FOR FUTURE WORK

Alternative Approach for Gradients Calculation

In the current development of the structural solver, the Green-Gauss theory has been used for the calculation of displacement gradients. This approach requires the displacement values to be known at the center of each face. Since the displacement values are being calculated at the center of the control volume, it was essential to use a weighted averaging interpolation to transfer the displacement values to the nodes, then transfer the nodal values into the center of the face. This entire process is relatively more computationally expensive than other gradients calculation methods. In the future, it is of importance to replace the Green-Gauss method with an alternative less expensive method, e.g. least square fitting.

Code Parallelization

In the current FSI analysis framework, the most CPU time consuming module is the CSD solver. Parallelizing the developed CSD code will allow running of large complex FSI cases efficiently. However, adding parallelization capabilities to the solver will not be a trivial task. Another component that needs to be parallelized is the mesh deformation module. The developed RBF based mesh deformation module does not require connectivity information, which will make it easier to parallelize.

Validation against other Benchmark Cases

After implementing the previous two suggestions it will be feasible to test more complex FSI problems. Two among several interesting benchmark cases to test would be the Rectangular Supercritical Wing case (RSW) and the AGARD wing case. Experimental data for the RSW and the AGARD wing deformations are available for validation purposes [134-136].

Add Non-Linear Elasticity and Thermal Elasticity Capabilities

In the current implementation of the structural methodology, only linear elasticity constitutive relation has been included. A nominal effort would be needed in order to extend the current version of the solver to include non-linear elasticity and thermal elasticity constitutive relations. This will allow the FSI solver to cover a wide range of interesting applications.

LIST OF REFERENCES

1. Heil, M., *Stokes flow in an elastic tube—a large-displacement fluid-structure interaction problem.* International Journal for Numerical Methods in Fluids, 1998. **28**(2): p. 243-265.

2. Ishihara, D. and S. Yoshimura, *A monolithic approach for interaction of incompressible viscous fluid and an elastic body based on fluid pressure Poisson equation.* International Journal for Numerical Methods in Engineering, 2005. **64**(2): p. 167-203.

3. Slone, A.K., et al., *A finite volume unstructured mesh approach to dynamic fluid–structure interaction: an assessment of the challenge of predicting the onset of flutter.* 2004. **28**(2): p. 211–239.

4. Hubner, B., E. Walhorn, and D. Dinkler, *A monolithic approach to fluid–structure interaction using space–time finite elements.* 2004. **193**(Issues 23–26): p. 2087–2104.

5. Kaliske, M.R., H. , *Formulation and implementation of three-dimensional viscoelasticity at small and finite strains.* Computational Mechanics, 1997. **19**(3): p. 228-239.

6. Park, K.F., C. ; Ohayon, R. , *Partitioned formulation of internal fluid–structure interaction problems by localized Lagrange multipliers.* Computer Methods in Applied Mechanics and Engineering, 2001. **190**(24): p. 2989-3007.

7. Zhu, M., et al., *An unstructured finite volume time domain method for structural dynamics.* 2012. **36**(1): p. 183–192.

8. Suliman, R., et al., *An enhanced finite volume method to model 2D linear elastic structures.* 2014. **38**(Issues 7–8): p. 2265–2279.

9. Vaz Jr., M., P.A. Muñoz-Rojas, and G. Filippini, *On the accuracy of nodal stress computation in plane elasticity using finite volumes and finite elements.* Computers and Structures, 2009. **87**(17-18): p. 1044-1057.

10. Xia, G. and C.L. Lin, *An Unstructured Finite Volume Approach for Structural Dynamics in Response to Fluid Motions.* Comput Struct, 2008. **86**(7-8): p. 684-701.

11. Batina, J.T., *Unsteady Euler airfoil solutions using unstructured dynamic meshes.* AIAA Journal, 1990. **28**(8): p. 1381-1388.

12. Luke, E., E. Collins, and E. Blades, *A fast mesh deformation method using explicit interpolation.* J. Comput. Phys., 2012. **231**(2): p. 586-601.

13. Robert, B. and E. John, *An overview of recent developments in computational aeroelasticity*, in *29th AIAA, Fluid Dynamics Conference*. 1998, American Institute of Aeronautics and Astronautics.

14. Dowell, E.H. and K.C. Hall, *Modeling Of Fluid-Structure Interaction.* Annual Review of Fluid Mechanics, 2001. **33**(1): p. 445-490.

15. Tijsseling, A.S., *Fluid-Structure Interaction In Liquid-Filled Pipe Systems: A Review.* 1996. **10**(2): p. 109–146.

16. Felippa, C.P., K. , *Staggered transient analysis procedures for coupled mechanical systems: Formulation.* Computer Methods in Applied Mechanics and Engineering, 1980. **24**(1): p. 61-111.

17. Matthies, H.G. and J. Steindorf, *Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction.* Computers & Structures, 2002. **80**(27-30): p. 1991-1999.

18. Steindorf, J. and H.G. Matthies, *Numerical Efficiency of Different Partitioned Methods for Fluid-Structure Interaction.* Journal of Applied Mathematics and Mechanics, 2000. **80**(S2): p. 557-558.

19. Zhao, S.Z., X.Y. Xu, and M.W. Collins, *The numerical analysis of fluid-solid interactions for blood flow in arterial structures. Part 2: Development of coupled fluid-solid algorithms.* Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 1998. **212**(4): p. 241-252.

20. Felippa, C., K.C. Park, and C. Farhat, *Partitioned analysis of coupled mechanical systems.* Partitioned analysis of coupled mechanical systems, 2001. **190**(24-25): p. 3247-3270.

21. Hübner, B., E. Walhorn, and D. Dinkler, *A monolithic approach to fluid–structure interaction using space–time finite elements.* 2004. **193**(Issues 23–26): p. 2087–2104.

22. Behr, M., et al., *Computation of incompressible flows with implicit finite element implementations on the connection machine.* Computer Methods in Applied Mechanics and Engineering, 1993. **108**: p. 19.

23. Carstens, V., R. Kemme, and S. Schmitt, *Coupled simulation of flow-structure interaction in turbomachinery* 2003. **7**(4): p. 298–306.

24. Chiandussi, G., et al., *A simple method for automatic update of finite element meshes.* Communications in Numerical Methods in Engineering, 2016. **16**(1): p. 1-19.

25. Kevlahan, N.K.-R. and O.V. Vasilyev, *An Adaptive Wavelet Method for Fluid—Structure Interaction.* 2016: p. 253-260.

26. Sarrate, J., A. Huerta, and J. Donea, *Arbitrary Lagrangian–Eulerian formulation for fluid–rigid body interaction.* 2001. **190**(Issues 24–25): p. 3171–3188.

27. Gordnier, R.E. and M.R. Visbal, *Development of A Three-Dimensional Viscous Aeroelastic Solver for Nonlinear Panel Flutter.* 2002. **16**(4): p. 497–527.

28. Sussman, T. and J. Sundqvist, *Fluid–structure interaction analysis with a subsonic potential-based fluid formulation.* 2003. **81**(Issues 8–11): p. 949–962.

29. Demirdžić, I. and S. Muzaferija, *Finite volume method for stress analysis in complex domains.* International Journal for Numerical Methods in Engineering, 1994. **37**(21): p. 3751-3766.

30. Yates, M.N., *Application of the finite-volume method to fluid-structure interaction analysis*, in *Aerospace and Civil Engineering*. 2011, University of Manchester.

31. Patankar, S.V.S., D. B. , *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows.* International Journal of Heat and Mass Transfer, 1972. **15**: p. 1787-1806.

32. Leonard, B.P., *A stable and accurate convective modelling procedure based on quadratic upstream interpolation.* Computer Methods in Applied Mechanics and Engineering, 1979: p. 59-98.

33. Wang, X.Q., R.M.C. So, and Y. Liu, *Flow induced vibration of an Euler–Bernoulli beam.* Journal of Sound and Vibration, 2001. **243,**: p. 241-268.

34. Baudlille, R. and M.E. Biancolini, *FSI makes FLUENT more flexible*, in *Fluent News*. 2005, FLUENT Inc.: 10 Cavendish Court, Lebanon, NH 03766, USA.

35. Van de Vosse, J., et al., *Finite element-based computational methods for cardiovascular fluid–structure interaction.* Journal of Engineering Mathematics, 2003. **47**: p. 335-368.

36. Gupta, A.K., et al., *Anomalous resonance in a nanomechanical biosensor.* 2006. **103**(36): p. 13362–13367.

37. Williamson, J.F., *Richard courant and the finite element method: A further look.* 1980. **7**(4): p. 369–378.

38. Turner, M.J., et al., *Stiffness and Deflection Analysis of Complex Structures.* Journal of the Aeronautical Sciences, 1956. **23**(9): p. 805-823.

39. Argyris, J.H. and S. Kelsey, *Energy theorems and structural analysis.* Aircraft Engineering, 1955.

40. Bijelonja, I., I. Demirdžić, and S. Muzaferija, *A finite volume method for incompressible linear elasticity*. 2006. **195**(Issues 44–47): p. 6378–6390.

41. Filippini, G., C.R. Maliska, and J.M. Vaz, *An Element-Based Finite Volume Method For Solid Mechanics Problems*, in *European Conference on Computational Fluid Dynamics*. 2010: Lisbon, Portugal.

42. Demirdzic, I., D. Martinovic, and A. Ivankovic, *Numerical simulation of thermal deformation in welded specimens.* Zavarivanje 1988. **31**: p. 209-219.

43. Demirdžić, I. and D. Martinovic, *Finite volume method for thermo-elasto-plastic stress analysis.* 1993. **109**(Issues 3–4): p. 331–349.

44. Wheel, M.A., *A finite-volume approach to the stress analysis of pressurized axisymmetric structures.* 1996. **68**(3): p. 311–317.

45. Tukovic, Z. and H. Jasak, *Updated Lagrangian finite volume solver for large deformation dynamic response of elastic body.* TRANSACTIONS OF FAMENA, 2006. **31**(1).

46. Jasak, H. and H.G. Weller, *Application of the finite volume method and unstructured meshes to linear elasticity.* International Journal for Numerical Methods in Engineering, 2000. **48**(2): p. 267-287.

47. Pletcher, R.H., J.C. Tannehill, and D. Anderson, *Computational Fluid Mechanics and Heat Transfer*. second edition ed. 1997: Taylor & Francis.

48. Bram, V., *CFD education - Past, present, future*, in *37th Aerospace Sciences Meeting and Exhibit*. 1999, American Institute of Aeronautics and Astronautics.

49. Khalil, E., *CFD History and Applications.* CFD LETTERS, 2012. **4**(2).

50. Launder, B.E. and D.B. Spalding, *The Numerical Computation of Turbulent Flows.* Computer Methods App. Mech., 1974: p. 269-275.

51. Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*. 1980: CRC Press

52. Blazek, J., *Computational Fluid Dynamics: Principles and Applications (Second Edition)*. . 2005: ELSEVIER.

53. Courant, R., K. Friedrichs, and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik - Springer.* Mathematische Annalen, 1928. **100**(1): p. 32-74.

54. Lerat, A. and C. Corre, *A Residual-Based Compact Scheme for the Compressible Navier–Stokes Equations.* 2001. **170**(2): p. 642–675.

55. Conlon, J. *OVERFLOW code empowers Computational fluid Dynamics*. 1998 [cited 2016 March]; Available from: http://www.hq.nasa.gov/hpcc/insights/vol5/overflow.htm.

56. Thomas, P., *High Order Accurate Finite-Difference Methods: as seen in OVERFLOW*, in *20th AIAA Computational Fluid Dynamics Conference*. 2011, American Institute of Aeronautics and Astronautics.

57. Zienkiewicz, O.C.T., R. L. ; Zhu, J. Z. , *The Finite Element Method: Its Basis and Fundamentals*. 7th Edition ed. 2013: Elsevier Science & Technology

58. Koomullil, R., B. Soni, and R. Singh, *A comprehensive generalized mesh system for CFD applications.* Mathematics and Computers in Simulation, 2008. **78**(5-6): p. 605-617.

59. Koomullil, R.P. and B.K. Soni, *Flow Simulation Using Generalized Static and Dynamic Grids.* AIAA Journal, 1999. **37**(12): p. 1551-1557.

60. Farhat, C., et al., *Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes.* Computer Methods in Applied Mechanics and Engineering, 1998. **163**(1-4): p. 231-245.

61. Nielsen, E.J. and W.K. Anderson, *Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes.* AIAA Journal, 2002. **40**: p. 1155-1163.

62. Crumpton, P.I. and M.B. Giles, *Implicit Time-Accurate Solutions on Unstructured Dynamic Grids.* Int. Journal of Numerical Methods & Fluids, 1997. **25**: p. 1285-1300.

63. McDaniel, D.R. and S.A. Morton. *Efficient Mesh Deformation for Computational Stability and Control Analyses on Unstructured Viscous Meshes*. in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*. 2009. Orlando, Florida.

64. Boer, A.D., M.S. van der Schoot, and H. Bijl. *New Method for Mesh Moving Based on Radial Basis Function Interpolation*. in *European Conference on Computational Fluid Dynamics. ECCOMAS CFD*. 2006.

65. Boer, A.D., M.S. van der Schoot, and H. Bijl, *Mesh Deformation Based on Radial Basis Function Interpolation.* Journal of Computers and Structures, 2007: p. 784-795.

66. Rendall, T. and C. Allen, *Efficient Mesh Motion Using Radial Basis Functions With Data Reduction Algorithms.* Journal of Computational Physics, 2009. **228**: p. 6231–6249.

67. Demirdžić, I., S. Muzaferija, and M. Perić, *Benchmark Solutions of Some Structural Analysis Problems Using Finite-Volume Method and Multigrid Acceleration.*

International Journal for Numerical Methods in Engineering, 1997. **40**(10): p. 1893-1908.

68. Wheel, M.A., *A finite volume method for analysing the bending deformation of thick and thin plates.* 1997. **147**(Issues 1–2): p. 199–208.

69. Demirdzic, I. and D. Martinovic, *Finite volume method for thermo-elasto-plastic stress analysis.* 1993. **109**(Issues 3–4): p. 331–349.

70. Slone, A.K., C. Bailey, and M. Cross, *Dynamic solid mechanics using finite volume methods.* 2003. **27**(2): p. 69–87.

71. Taylor, G.A., C. Bailey, and M. Cross, *A vertex-based finite volume method applied to non-linear material problems in computational solid mechanics.* International Journal for Numerical Methods in Engineering, 2002. **56**(4): p. 507-529.

72. Zienkiewicz, O.C. and R.L. Taylor, *The Finite Element Method - (Second Edition)*. 2000: Butterworth-Heinemann.

73. Slaughter, W.S., *The Linearized Theory of Elasticity*. 2002: Birkhäuser Basel.

74. Meyers, M.A. and K.K. Chawla, *Mechanical Behavior of Materials*. 1999: Prentice Hall.

75. Zienkiewicz, O.C. and R.L. Taylor, *The Finite Element Method, The Basis*. fifth edition ed. 2000, Oxford: Butterworth-Heinemann.

76. Lv, X., et al., *An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3D unsteady compressible flows with moving objects.* Journal of Computational Physics, 2006. **215**: p. 661-690.

77. Ahmed, E. and W.H.W. Badaruzzaman, *Evaluation of natural frequency and damping of Profiled steel sheet dry board composite panel.* J. of Engineering Science and Technology, 2011. **6**: p. 695–708.

78. Mattiasson, K., *Numerical results from large deflection beam and frame problems analysed by means of elliptic integrals.* International Journal for Numerical Methods in Engineering, 1981. **17**(1): p. 145-153.

79. Hassan, O., E.J. Probert, and K. Morgan, *Unstructured mesh procedures for the simulation of three-dimensional transient compressible inviscid flows with moving boundary components.* International Journal for Numerical Methods in Fluids, 1998. **27**(1-4): p. 41-55.

80. Ding, Z., H. Zhu, and M.H. Friedman, *Coronary Artery Dynamics In Vivo.* Annals of Biomedical Engineering, 2002. **30**(4): p. 419-429.

81. Douglass, R.W., et al., *Current views on grid generation: summaries of a panel discussion.* Numer. Heat Transfer Fund., 2002. **41**(211 SRC - GoogleScholar).

82. Samareh, J.A., *Status and future of geometry modeling and grid generation for design and optimization.* Journal of Aircraft, 1999. **36**(97 SRC - GoogleScholar).

83. Teng, S.H. and C.W. Wong, *Unstructured mesh generation: theory, practice, and perspectives.* International Journal of Computational Geometry & Applications, 2000. **10**(227 SRC - GoogleScholar).

84. Batina, J.T., *Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes.* AIAA Journal, 1990. **28**(8 SRC - GoogleScholar): p. 1381-1388.

85. Frederic, B., et al., *Analysis of fluid-structure interaction on moving airfoils by means of an improved ALE-method*, in *28th Fluid Dynamics Conference*. 1997, American Institute of Aeronautics and Astronautics.

86. Farhat, C., M. Lesoinne, and N. Maman, *Mixed explicit:implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution.* International Journal for Numerical Methods in Fluids, 1995. **21 SRC - GoogleScholar**: p. 807-835.

87. Piperno, S., *Explicit:implicit fluid:structure staggered procedures with a structural prediction and fluid subcycling for 2D inviscid aeroelastic simulations.* International Journal for Numerical Methods in Fluids, 1997. **25 SRC - GoogleScholar**: p. 1207-1226.

88. Blom, F., *Considerations on the spring analogy.* International Journal for Numerical Methods in Fluids, 2000. **32 SRC - GoogleScholar**: p. 647-668.

89. Farhat, C., et al., *Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes.* Computer Methods in Applied Mechanics and Engineering, 1998. **163**(1-4 SRC - GoogleScholar): p. 231-245.

90. Burg, C.O.E. *A Unstructured Grid Movement Strategy using Three-Dimensional Torsional Springs*. 2004. Portland, Oregon SRC - GoogleScholar.

91. Bottasso, C.L., D. Detomi, and R. Serra, *The ball-vertex method: a new simple analogy method for unstructured dynamic meshes.* Comput. Methods Appl. Mech. Engrg., 2005. **194 SRC - GoogleScholar**: p. 4244-4264.

92. Markou, G.A., et al., *The ortho-semi-torsional (OST) spring analogy method for 3D mesh moving boundary problems.* Comput. Methods Appl. Mech. Engrg., 2007. **196 SRC - GoogleScholar**: p. 747-765.

93. Degand, C. and C. Farhat, *A Torsional Spring Analogy Method for Unstructured Dynamic Meshes.* Comput. Struct., 2002. **80**(305 SRC - GoogleScholar).

94. Zeng, D. and C.R. Ethier, *A semi-torsional analogy model for updating unstructured meshes in 3D moving domains.* Finite Elem.Anal Des, 2005. **41 SRC - GoogleScholar**: p. 1118-1139.

95. Biedron, R.T. and E.M. Lee-Rausch. *Rotor Airloads Prediction Using Unstructured Meshes and Loose CFD/CSD Coupling*. 2008. Honolulu, Hawaii SRC - GoogleScholar.

96. Zhi, Y. and M. Dimitri, *Unstructured Dynamic Meshes with Higher-order Time Integration Schemes for the Unsteady Navier-Stokes Equations*, in *43rd AIAA Aerospace Sciences Meeting and Exhibit*. 2005, American Institute of Aeronautics and Astronautics.

97. Johnson, A.A. and T.E. Tezduyar, *Simulation of multiple spheres falling in a liquid-filled tube.* 1996. **134**(Issues 3–4): p. 351–373.

98. Karman, S.L., W.K. Anderson, and M. Sahasrabudhe, *Mesh Generation Using Unstructured Computational Meshes and Elliptic Partial Differential Equation Smoothing.* AIAA Journal, 2006. **44**(6): p. 1277-1286.

99. Steve, K., Jr., *Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing*, in *44th AIAA Aerospace Sciences Meeting and Exhibit*. 2006, American Institute of Aeronautics and Astronautics.

100. Freitag, L.A. and P.M. Knupp. *Tetrahedral Element Shape Optimization via the Jacobian Determinant and Condition Number*. in *in Proceedings of the 8th International Meshing Roundtable*. 1999.

101. Knupp, P.M., *Matrix Norms and the Condition Number: A General Framework to Improve Mesh Quality via Node-Movement*, in *8th International Meshing Roundtable [online database]*. 2005.

102. Yang, Z. and D.J. Mavriplis, *Mesh Deformation Strategy Optimized by the Adjoint Method on Unstructured Meshes.* AIAA Journal, 2007. **45**(12): p. 2885-2896.

103. Su-Yuen, H., C. Chau-Lyan, and S. Jamshid, *A Simplified Mesh Deformation Method Using Commercial Structural Analysis Software*, in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2004, American Institute of Aeronautics and Astronautics.

104. Johnson, A.A. and T.E. Tezduyar, *Simulation of Multiple Spheres Falling in a Liquid-Filled Tube.* Computer Methods in Applied Mechanics and Engineering, 1996. **134 SRC - GoogleScholar**: p. 351-373.

105. Dwight, R.P., *Robust Mesh Deformation using the Linear Elasticity Equations*, in *Computational Fluid Dynamics 2006: Proceedings of the Fourth International Conference on Computational Fluid Dynamics, ICCFD, Ghent, Belgium, 10-14*

July 2006, H. Deconinck and E. Dick, Editors. 2009, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 401-406.

106. Jasak, H. and Z. Tuković, *Automatic Mesh Motion for the Unstructured Finite Volume Method*, in *Transactions of Faculty of Mechanical Engineering and Naval Architecture*. 2007: Univ. of Zagreb, Crotia. p. 1-18.

107. Lohner, R.Y., C.; Onate, E., *Viscous Free Surface Hydrodynamics Using Unstructured Grids*, in *Proceedings of Twenty-Second Symposium on Naval Hydrodynamics*. 1998.

108. Burg, C., *Analytic study of 2D and 3D grid motion using modified Laplacian.* International Journal for Numerical Methods in Fluids, 2006. **52**(2): p. 163-197.

109. Thompson, J.F., Z.U.A. Warsi, and C.W. Mastin, *Numerical grid generation: foundations and applications*. 1985: Elsevier North-Holland, Inc. 483.

110. Tsai, H.M., et al., *Unsteady Flow Calculations with a Parallel Multiblock Moving Mesh Algorithm.* AIAA Journal, 2001. **39**(6): p. 1021-1029.

111. Chansup, B. and G. Guru, *A parallel, multi-block, moving grid method for aeroelastic applications on full aircraft*, in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 1998, American Institute of Aeronautics and Astronautics.

112. Zhao, Y. and A. Forhad, *A general method for simulation of fluid flows with moving and compliant boundaries on unstructured grids.* Computer Methods in Applied Mechanics and Engineering, 2003. **192**: p. 4439-4466.

113. Bartier, P.M. and C.P. Keller, *Multivariate interpolation to incorporate thematic surface data using inverse distance weighting (IDW).* Comput. Geosci., 1996. **22**(7): p. 795-799.

114. Jeroen, W., *Explicit and Robust Inverse Distance Weighting Mesh Deformation for CFD*, in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. 2010, American Institute of Aeronautics and Astronautics.

115. Sheng, C. and C.B. Allen, *Efficient Mesh Deformation Using Radial Basis Functions on Unstructured Meshes.* AIAA Journal, 2012. **51**(3): p. 707-720.

116. Mario Botsch, L.K. *Real-time shape editing using radial basis functions*. in *Computer Graphics Forum*. 2005.

117. Michler, A.K., *Aircraft control surface deflection using RBF-based mesh deformation.* International Journal for Numerical Methods in Engineering, 2011. **88**(10): p. 986-1007.

118. Liu, X., N. Qin, and H. Xia, *Fast dynamic grid deformation based on Delaunay graph mapping.* J. Comput. Phys., 2006. **211**(2): p. 405-423.

119. George, P.L., F. Hecht, and E. Saltel, *Automatic mesh generator with specified boundary*. Comput. Methods Appl. Mech. Eng., 1991. **92**(3): p. 269-288.

120. Weatherill, N., O. Hassan, and D. Marcum, *Calculation of steady compressible flowfields with the finite element method*, in *31st Aerospace Sciences Meeting*. 1993, American Institute of Aeronautics and Astronautics.

121. Guibas, L.J., D.E. Knuth, and M. Sharir, *Randomized incremental construction of Delaunay and Voronoi diagrams.* Algorithmica, 1992. **7**(1): p. 381-413.

122. Devroye, L., P.E. Mücke, and B. Zhu, *A Note on Point Location in Delaunay Triangulations of Random Points.* Algorithmica, 2012. **22**(4): p. 477-482.

123. Yu, L., G. Zhenga, and L. Jun, *RBFs-MSA Hybrid Method for Mesh Deformation.* 2012. **25**(4): p. 500–507.

124. Samareh, J.A., *Application of Quaternions for Mesh Deformation*. 2002, NASA.

125. Maruyama, D., D. Bailly, and G. Carrier, *High-Quality Mesh Deformation Using Quaternions for Orthogonality Preservation.* AIAA Journal, 2014. **52**(12): p. 2712-2729.

126. Gong, D., et al., *Adaptive Methods for Center Choosing of Radial Basis Function Interpolation: A Review*, in *Information Computing and Applications: First International Conference, ICICA 2010, Tangshan, China, October 15-18, 2010. Proceedings*, R. Zhu, et al., Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 573-580.

127. Botsch, M. and L. Kobbelt, *Real-time shape editing using radial basis functions.* Journal of Eurographics, 2005. **24**(3).

128. Skala, V., *Radial Basis Functions Interpolation and Applications: An Incremental Approach*, in *ASM 2010 conference*. 2010: Corfu, Greece. p. 209-213.

129. Sung Eun, J., K. Sang Woo, and P. Tae Joon. *Equally constrained affine projection algorithm*. in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*. 2004.

130. Bretscher, O., *Linear Algebra with Applications*. 4th Edition ed. 2009: Pearson Prentice Hall.

131. Xingyuan, S., S. Chunhua, and A. Christian, *An Efficient Mesh Deformation Approach Based On Radial Basis Functions In Unstructured Flow Solver*, in *20th AIAA Computational Fluid Dynamics Conference*. 2011, American Institute of Aeronautics and Astronautics.

132. Jasak, H.J., A.; Tukovi, Z., *OpenFOAM: A C++ Library for Complex Physics Simulations,* in *International Workshop on Coupled Methods in Numerical Dynamics*. 2007: Dubrovnik, Croatia.

133. Ito, Y.S., A. M.; Soni, B. K. *Unstructured Mesh Generation Using MEGG3D - Mixed-Element Grid Generator in Three Dimensions,*. in *Proceedings of the International Conference on Numerical Geometry , Grid Generation and Scientific Computing (NUMGRID2008)*. 2008. Moscow, Russia.

134. Ricketts, R.H., et al., *Geometric and structural properties of a rectangular supercritical wing oscillated in pitch for measurement of unsteady transonic pressure distributions*. 1983, NASA Langley Research Center; Hampton, VA, United States.

135. Ricketts, R.H., et al., *Transonic pressure distributions on a rectangular supercritical wingoscillating in pitch.* Journal of Aircraft, 1984. **21**(8): p. 576-582.

136. Yates, E.C., Jr, *AGARD standard aeroelastic configurations for dynamic response. Candidate configuration I.-wing 445.6*. 1987, NASA Langley Research Center; Hampton, VA, United States.

137. Lorentzon, J., *Fluid-Structure Interaction (FSI) case study of a cantilever using OpenFOAM and DEAL.II with application to VIV,* in *Department of Energy Sciencies*. 2009, Lunds Institute of Technology, Sweden.