AMERICAN UNIVERSITY OF BEIRUT

THE STUDY OF DERIVING A HIGHER ORDER MOVING MESH SCHEME

by
HUSSEIN YOUSSEF MAANIEH

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Mechanical Engineering
to the Department of Mechanical Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
April 2016

AMERICAN UNIVERSITY OF BEIRUT

THE STUDY OF DERIVING A HIGHER ORDER MOVING MESH SCHEME

by

HUSSEIN YOUSSEF MAANIEH

Approved by:

Moukalled Fadl

Dr. Fadl Moukalled, Professor — Advisor
Department of Mechanical Engineering

[illegible]

Dr. Marwan Darwish, Professor — Member of Committee
Department of Mechanical Engineering

for Moukalled Fadl

Dr. Kamel Ghali, Professor — Member of Committee
Department of Mechanical Engineering

Date of thesis defense: April 22, 2016

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: ____Maanieh___________Hussein_______________Youssef________
Last First Middle

[✓ Master's Thesis] ◯ Master's Project ◯ Doctoral Dissertation

[✓ X] I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project,** to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

[illegible] ____________________________________ May 6th, 2016________
Signature Date

ACKNOWLEDGMENTS

I want to give more than thanks to my amazing family members and lovely fiancé for supporting and motivating me along these years to finish this accomplishment.

My recognition and gratitude are addressed to the duo, Professor Fadl Moukalled and Professor Marwan Darwish, for their substantial support, assistance and patience that led this work to be accomplished.

Special thanks are addressed to Professor Kamel Ghali for giving time to be a part of the committee to discuss this work, as well as to the C.F.D. group at the American University of Beirut and to all the graduate students who supported me throughout my work.

AN ABSTRACT OF THE THESIS OF

Hussein Youssef Maanieh for Master of Mechanical Engineering
Major: Mechanical Engineering

Title: The Study of Deriving a Higher Order Moving Mesh Scheme

The formulation and implementation of a higher-order accurate scheme for arbitrary moving mesh problems, is presented. The governing equations are formulated according to the Arbitrary Lagrangian-Eulerian (ALE) approach in a form that satisfies the Geometrical Conservation Laws (GCL) in an intrinsic manner. The contribution of this work is achieved by the technique used to deal with the introduced moving mesh term represented by face mesh velocities. The method replaces the aforementioned face mesh velocities by equivalent volumetric face increments on which the formulated explicit higher order scheme is applied. The scheme is implemented on the three dimensional MATLAB unstructured Finite Volume Method (u-FVM) solver where several test cases are conducted and simulated on static and arbitrary moving grids with their results being presented and analyzed. The results indicate that the set obtained from higher order mesh treatment tend to overlap with the results of the static mesh, hence, proving the effectiveness of the formulated scheme.

CONTENTS

ABSTRAC

LIST OF

LIST OF

Chapter

I. INTR

……

ILLUSTRATIONS

TABLES

CHAPTER I

INTRODUCTION

There are three important steps that have to be followed carefully in the computational modeling of any physical process; we list:

(i) Problem definition

(ii) Mathematical modeling

(iii) Computer simulation.

The most important prerequisite to the aforementioned steps is the correct treatment of the geometry of the problem. Meshing the geometry in an adequate framework is needed for the solution to be realistic, logical and as exact as possible to the experimental results.

For our interest, in Computational Fluid Dynamics (CFD), physical conservation laws represented in Partial Differential Equations (PDEs) are to be solved numerically on discrete grids using adequate algorithms. Such algorithms of numerical solving depend on the chosen frame of discretizing the partial differential equations, whether Finite Difference Method (FDM), Finite Element Method (FEM) or Finite Volume Method (FVM).

Finite Difference framework is the oldest discretization approach where it uses the differential form of the governing equations. The discretization procedure uses a topologically square network of lines over the computational domain under study, and at each grid point, the differential equations are approximated by terms from the neighboring grid points. For each grid point, a single algebraic equation is obtained. Taylor series expansion or polynomial fitting is used to obtain first or second order derivative terms of the governing equations.

The topology of discretization followed in the FDM acted as a potential bottleneck of the method when handling complex geometries in multiple dimensions. This issue motivated using the integral form of the PDEs and subsequently led to the development of the FEM and FVM approaches.

Finite Element framework is usually associated with solid mechanics, however it can be applied to fluid flows but with special care to ensure the continuity of mass (Ferziger and Peric 2002). This approach divides the domain into triangular or quadrilateral elements where the differential equation is multiplied by an arbitrary test function to later integrate both over the whole domain. Since this approach is predominantly used in unstructured meshes, it is well suited to deal with arbitrary geometries, as the grids are easily refined, however the resulted discretized system of equations is difficult to solve.

In the Finite Volume framework, the solution domain is filled with a mesh. This mesh is used to define the storage locations of each variable stating mainly the vertex-centered and cell-centered methods. Finite control volumes are constructed around each storage location, and the governing equations are integrated over each control volume in a conservative form. The volume integrals are converted to surface integrals by means of Gauss' divergence theorem, and the surface integrals are then approximated in terms of variables defined at the adjacent storage locations, depending on the order of the scheme selected. Hence, according to this process, the differential equations are replaced by algebraic equations: one for each conservation equation for each control volume.

Although FEM must be carefully formulated to be conservative, it is more stable than the FVM approach (Surana, et al. 2006) but the latter is conservative by phenomena.

However, FEM requires more memory and has slower solution times than the FVM (Huebner, Thornton and Byrom 1995). For time-dependent problems, the finite volume principle has traditionally been used to discretize the spatial dimensions and temporal dimensions. If, on another side, the mesh undergoes motion, these methods require the use of Leibniz rule (Spiegel 1997) (Kaplan 1973) (Hildebrand 1976) to account for mesh motion.

A. Thesis Objective:

In our presented work, the finite volume method is adopted and all the discretization of the partial differential equations will be casted according to this method. The main objective of the presented dissertation is the development of a high order scheme that will discretize the moving mesh term developed when allowing the mesh to move arbitrarily with time. The Geometric Conservation Laws that accompany the moving mesh phenomena are well treated and satisfied in the governing equations. Several test cases are simulated to ensure that the proposed scheme is valid and provides reliable results.

B. Thesis Significance:

Nowadays, fluid structure interaction problems are encountered in diverse scientific areas and the rapid progress in this field urged us to develop the capabilities of the unstructured Finite Volume Method (u-FVM) solver base on MATLAB software so as to introduce the mesh motion along with its relative changes and generated terms into the code and trying to deal effectively with them. The significance of this work, serves into the development of the computational field by introducing a scheme that will help researchers improve their computational codes.

C. Thesis Organization:

The remaining of this thesis is divided into chapters that cover the main boundaries of the research topic. The next chapter i.e. the second chapter reviews the literature associated with the topic of this work and presents work that is prerequisite or related to the topic. Following that, in the third chapter the governing equations are derived in an Arbitrary Lagrangian-Eulerian (ALE) frame after introducing and elaborating on the ALE approach. The fourth chapter briefly describes the u-FVM code by presenting its main functions and the borderline of how a simulation is performed in it. Chapter five describes how the mesh nodes are moved arbitrary in time and discuss the technique used to calculate the generated face volumetric increments satisfying the Geometric Conservation Laws (GCL). Chapter six presents and highlights the validity of the high order scheme formulated and implemented in the u-FVM code by applying and addressing the results of several test cases. The thesis is wrapped up in a conclusion presented in Chapter seven.

CHAPTER II

LITERATURE REVIEW

A. Background

Over the past decades, two categories of generated grids are differentiated in the literature of CFD as:

i) Domain-Conformal Grid Methods.

ii) Non Domain-Conformal Grid Methods.

The term “Immersed Boundary” was first introduced by Peskin in the aim of studying cardiac mechanics and blood flow patterns around heart valves (Peskin 1972). Peskin’s contribution was to carry out the entire simulation on a Cartesian grid that did not conform to the geometry of the heart. Moreover, the author formulated a procedure for imposing the effect of his new Immersed Boundary Method (IBM in short) on the flow. Since then, Peskin’s pioneering method evolved into a generally useful method for problems involving fluid-structure interaction. Another related class of methods, referred as “Cartesian Grid Methods”, were proposed and developed to simulate inviscid flows on Cartesian grids for the cases of complex embedded boundaries (Beyer 1992, Clarke, Hassan et al. 1986, De Zeeuw, Powell 1991). The aforementioned methods were further extended to cover unsteady viscous flows (Ye, Mittal et al. 1999, Udaykumar, Shyy et al. 1996) and by that they had similar capabilities as the IB methods. In order to avoid any conflict between the two set of methods, and to encompass a broader group which includes all the methods that simulate viscous flows with immersed or embedded boundaries on non-conformal boundary grids, we will refer to both set of methods by the term “Non Domain-Conformal Grid Methods”.

The major treatment that needs specific attention while developing Non Domain-Conformal Grid methods is how to impose the boundary conditions in the algorithm. The boundary conditions are indirectly enforced into the discretization process of flow equations (for example: Navier-Stokes in an incompressible flow) after undergoing certain modifications, of which, a source term (also called a forcing factor) that represents the effect of the boundary is added to the discretized equations.

There exist two approaches for implementing the forcing factor. In the first approach, called the *continuous forcing approach*, the forcing factor is enforced into the governing flow equations before discretizing them on the Cartesian grid. The equations are then discretized and solved over the entire domain of interest. Many applications have been reported in the literature in accordance to this approach with elastic and rigid boundaries in biological fluid dynamics (Beyer 1992, Peskin 1982, Peskin 2002), in studies on insects (Miller, Peskin 2004, Miller, Peskin 2005), and in multiphase flows (Unverdi, Tryggvason 1992, McIntyre 2011). However, this approach requires solving the governing equations inside the immersed body which will make the problem burdensome with increasing Reynolds number and on another side, the introduced forcing terms generally don't behave well in the rigid limit.

In the second approach, called the *discrete forcing approach*, the governing equations are first discretized on a Cartesian grid without regarding the immersed boundary. Then, boundary conditions are either imposed directly or indirectly on the immersed boundary and this opens up two distinguished categories in this approach. *Direct forcing* methods introduce a discrete momentum source that, when integrated over a time step, identically forces the velocity to the prescribed value on the immersed boundary. On the other hand, in the *Indirect Imposition* category, Mohd-Yosuf (Mohd-Yusof 1997) attempted to develop a method that extracts the forcing directly from the numerical solution so as an a priori prediction can be determined. Then, Verzicco et al. (Verzicco, Mohd-Yusof et al. 1998) applied the direct forcing method to a finite-difference code for large eddy simulation of flow in an engine cylinder. On the other hand, the need to retain the “sharp” interface of the immersed boundary especially at high Reynolds numbers with greater emphasis on the local accuracy near the immersed boundary has opened up a new category of direct discrete forcing. Two methods fit in this category, one suiting the finite difference frame of work called the *ghost-cell finite difference method* and the other suiting the finite volume frame being called the *cut-cell finite volume method*. The first method uses ghost-cells, which are defined as cells in the solid region having at least one neighbor in the fluid region. In this method, each ghost cell will have a special implicit interpolation scheme devised to it. Several authors proposed adequate interpolation schemes, we mention among (Ghias, Mittal et al. 2004, Majumdar, Iaccarino et al. 2001). Irrespective of the chosen interpolation scheme, the modified discrete equations of the ghost cells can be solved simultaneously with the discretized Navier-Stokes equations for fluid nodes. The second method in this category, called the *cut-cell method,* uses a finite volume work frame. Remarking that finite volume methods are the only to guarantee strict local and global conservation of mass and momentum, the cells of this method in the Cartesian grid that are cut by the immersed boundary are identified, and the intersection of the boundary with the sides of these cells is determined. Cells that are cut by the immersed boundary, and whose cells centers lie in the fluid domain, are reshaped by dumping the portion of these cells that lies in the solid domain. Neighboring cells grip portions of cut-cells, whose centers lie in the solid domain which results in formulating trapezoidal control volumes (Ye, Mittal et al. 1999).

The major advantage of the *discrete forcing approach* is the absence of the user-defined parameters in the forcing but the details of implementation still depend strongly on the numerical algorithm used to discretize the equations. Yet, there exists a major disadvantage in this approach which lies in the difficulty of including boundary motion and this makes the approach less desirable to use for problems involving moving boundaries.

B. Flow with Moving Immersed Boundaries

Most of the methods described in the Non Domain-Conformal Grid Methods category follow the Eulerian-Lagrangian frame, wherein the governing equations are represented in a Eulerian form and solved on a stationary grid while the moving boundaries are tracked following a Lagrangian treatment. Among these methods, we can distinguish the procedure adopted to track the immersed boundary as well as the methodology used to represent its influence on the underlying Eulerian flow-field. In this manner, Peskin's Immersed Boundary Method (Peskin 1982) tracks the boundary as a separate and sharp Lagrangian body while its influence is represented by a diffusing effect on the fluid field. However, for the methods of cut cell and ghost cell previously described, the immersed boundary is treated as a sharp Lagrangian body and at the same stage it is represented as such in the underlying Eulerian flow-field. One can also differentiate on another basis between the aforementioned Non Domain-Conformal Grid Methods and the Volume-of-Fluid methods (Hirt, Nichols 1981, Scardovelli, Zaleski 1999, Anderson, McFadden et al. 1998) where the last preserve the diffuse nature of the boundary in tracking along with representing its effect on the flow field.

Literature on sharp-interface methods such as cut-cell methods has pointed out an important issue to be realized and dealt with in order to enable boundary motion. As the immersed boundary moves across the fixed Cartesian grid, “freshly-established” cells, or in other words cells in the fluid field that were inside the solid field at the prior time step, are encountered. This is considered as a spatial discontinuity and is usually associated with sharp immersed boundaries, hence leading to a temporal discontinuity for the cells lying near the moving boundary (Mittal and Iaccarino 2005). Adding up, another disadvantage is associated with the Immersed Boundary Methods that is because the immersed boundary is smeared across few cell-widths where the point force is represented on a finite size mesh.

After all, the drawback of Non Domain-Conformal Grid Methods remains in its complicated treatment of the fluid structure interaction by accurately tracking the interface and accurately estimating the velocity field near it.

We turn our attention now to the Domain-Conformal Grid Methods. These methods are considered as a better choice than Non Domain-Conformal Grid Methods for Fluid-Structure Interaction (FSI) problems, with moderately complex interface and medium deformations or movements in the solid domain. We remark for instance that the generation of body conformal grids to the FSI interface permits the estimation of accurate solutions of the tractions and velocities near the FSI interface. In this category, the external mesh faces match up with the body interface and the external bounding faces of the domain. Simulating flows with moving boundaries using body-conformal grids requires the generation of a new grid at each time step as well as a technique to project the solution onto the new grid. When simulating moving bodies on body-conformal grids, the partial differential equations should be cast in Arbitrary

Lagrangian-Eulerian (ALE) frame of reference (Donea, Huerta et al. 2004), where the idea is to move the mesh with minimal distortion. Throughout the years, significant progress has been witnessed in simulating flows with moving boundaries on body-conformal grid methods (Baum, Luo et al. 1998, Ramamurti, Sandberg 2001, Tezduyar 2001).

C. Arbitrary Lagrangian-Eulerian Formulation

The ALE formulation which can be viewed as a mesh movement strategy is adopted in cases where the CFD equations are defined on domains with continuously moving or deforming grids with time due to boundary movement. The governing equations are enforced over the control volumes whose geometrical shape and position are neither constant in time, thereby the grid positioning geometric quantities and velocities of the moving or deforming grid points must be determined. Accurately determining these parameters is an essential factor to ensure the conservation of mass and momentum over the moving grid. For a general problem, the grid positioning geometric quantities are usually available as a function of time particularly if the grid deformation is governed by a set of partial differential equations. The ALE formulation is particularly useful for problems involving fluid-structure interaction based on the coupling between the fluid dynamics models and the finite element models of aircraft structure (Bendiksen 2011, Bennett, Edwards 1998, Schuster, Liu et al. 2003, Yurkovich 2003). The ALE formulation has been intensively employed in problems involving small and large structure displacements with no topological changes in the structure domain. A recent and interesting application of the ALE formulation occurs in the medical field, where undergoing experiments and extracting quantitative flow information is very difficult, time consuming and expensive (Taylor, Hughes et al. 1998). Among other applications, the ALE formulation was used as well for soil-structure interaction and the detailed analysis can be found in (Shahrour, Benchekh 1992). We focus our attention in this study on the implementation of the moving grid concept along with establishing the corresponding ALE flow equations in a finite volume discretization scheme for general boundary-fitted grids.

The major task while simulating the ALE equations is the determination of the variable geometric parameters, which is governed by the satisfaction of the so-called Geometric Conservation Laws (GCL in short).

D. Geometric Conservation Laws

While reviewing the literature on Computational Fluid Dynamics, it comes into notice that two additional equations that formulate, for static or moving meshes, the balance between the relevant geometric parameters have sometimes been ignored. This led to the misrepresentation of the convective velocities thus violating the conservation laws and producing extra sources or sinks in the physically conservative media. These errors have been witnessed and investigated by (Hindman 1982, Demirdžić, Perić 1988). Furthermore, violating these laws leads to the occurrence of severe restrictions on numerical solvers (Demirdžić, Perić 1988, Vinokur 1989). The importance of the GCL has long been ignored and the errors resulting from the non-satisfaction of the laws have been regarded to other sources (Amsden, Ruppel et al. 1980, Viviand, Ghazzi 1976). The two equations, called Geometric Conservation Laws helps establishing the conservative relations between the surfaces and volumes of the control volumes in the discretized domain. The first equation states that cell volumes must be enclosed by all its surfaces, and hence is referred as the Space Conservation Law (SCL in short). The second equation states that the volumetric increment of a moving control volume must be equal to the sum of the changes along the surfaces that enclose the volume of the control volume, and is called the Volumetric Conservation Law (VCL in short). Satisfying the two aforementioned equations is crucial in order to achieve the global conservation of the domain of interest.

Among several numerical solutions to problems with moving boundaries that are reported in literature (Gosman, Johns 1978, Krause 1979, Durst, Pereira et al. 1986), the SCL was first included into the set of mass, momentum and energy equations by (Trulio, Trigger 1961), but it wasn’t recognized until (Thomas, Lombard 1978, Thomas, Lombard 1979) rediscovered the equation and marked out the necessity of satisfying it numerically along with solving the set of governing equations.

In the study of (Demirdžić, Perić 1988), the authors defined the grid surface velocities in an explicit manner so that the rate of change of the cell volume obtained from the VCL exactly equalizes the actual geometrical rate of change. Care was not addressed to the path of the vertices or nodes of the moving surfaces of a control volume of the moving mesh unlike the study of (Trepanier, Reggio et al. 1991), where the latter figured out that for the 2-D and axisymmetric cases, the computed facial increments should be independent of the order of nodal motions, and the facial volumetric variations following any permutation must sum up to the same total volumetric variation of the moving element in the discretized domain. This was not the case for the 3-D problems where each permutation would define a different volumetric increment, in this case the investigators tend to average the value obtained from all the possible permutations to the movement of the nodes of an element. In the paper of Zhang et al., the authors implemented the approach of moving the grid onto Finite Volume framework and elaborated as well on the suggested modifications in the flux terms (Zhang, Reggio et al. 1993).

CHAPTER III

U-FVM DESCRIPTION

A. Introduction

u-FVM is a general MATLAB® code developed to solve over three dimensional unstructured as well as structured grids following the finite volume method approach.

The code has the capability of solving a wide range of flow problems with single or multi-fluids and transport phenomena problems. u-FVM which stems from unstructured Finite Volume Method is an academic fully accessible code that stresses on programming simplicity, organization and reliability over speed of performance. The organized structure of the code allows the user to develop it by embedding new functions within the main code to further enhance its capabilities.

B. The Basic Structure

1. Case Setup

u-FVM code is composed of several task oriented functions in the form of MATLAB® script files each devoted to a specific task. These set of functions that mimic the numeric of the Finite Volume Method, act all together in the aim of solving any specified case. The user can setup a case file by creating a script file in the MATLAB® editor and importing the appropriate functions in a wise order as illustrated in the Figure 3.1 below. The user should have the basic knowledge of the physics of his case problem so that to be able to determine which functions to include in the case problem script file.

```
    % Convection-Diffusion Problem Solved on Static Grid
    clear all;
          clc;
          global Domain
          cfdSetupDomain;
%Reading the Geometry from OpenFOAM®
          cfdReadOpenFOAM®Mesh('Domain25CV');
% setup Fluid
          cfdSetupFluid('water','MW',18);
          cfdSetIsTransient;
          cfdSetIsMoving(false);
%Creating the Property Fields
          cfdSetupProperty('Density:water','constant','1000');
    cfdSetupProperty('SpecificHeat:water','constant','4.186');
    cfdSetupProperty('conduction:water','constant','4.186');
    cfdSetupProperty('Velx:water','constant','10');
    cfdSetupProperty('Vely:water','constant','10');
    cfdSetupProperty('Velz:water','constant','0');
% Setting the equation:
%=======================
    cfdSetupEquation('T:water','ic','10','urf',1);
% Adding the terms constituting the equation with appropriate coefficients
    cfdAddTerm('T:water','Transient','coefficientName','Density:water',...
    'coeffiecientName','SpecificHeat:water');
    cfdAddTerm('T:water','Diffusion','coefficientName','conduction:water');
    cfdAddTerm('T:water','Convection','coefficientName','Density:water',...'scheme','UPWIND');
% Specifying the Boundary Conditions for the equation
    cfdSetBC('T:water',1,'type','Specified Value','value','10');%InletBC
    cfdSetBC('T:water',2,'type','Outlet'); % Outlet BC
    cfdSetBC('T:water',3,'type','Specified Value','value','0');%SideWalls
    cfdSetBC('T:water',4,'type','empty'); % Front & Back
% Creating an Mdot Field
    cfdSetupMdotFields;
% Initializing the special array for each field
    cfdInitializeFields;
    time_i=0;
    time_f=7;
    dt=1;
% Starting the Time loop
    for time=time_i:dt:time_f
    k=1;
    time_p=time;
    time_c=time_p+dt;
    cfdSetTime(time_c)
    cfdSetDt(dt)
    cfdTransientUpdate;
    fprintf('%s %d \n', 'Time:', time_c);
    disp('---------------------');
% Internal Iterations
    for iter=1:100
    cfdUpdateFields;
    fprintf('%s\n %d \n','Iteration: ',iter);
    cfdAssembleAndCorrectEquation('T:water');
    end
% Plotting the temperature field and the residuals of the equation
    cfdPlotField('T:water',k);
    colorbar;
    cfdPlotResiduals;
% A step to save the Phi Field at each time step
    cfdSavePhiAtEachTimeStep('T:water',k);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
```

Figure 3.1: The body of a test case in u-FVM MATLAB code

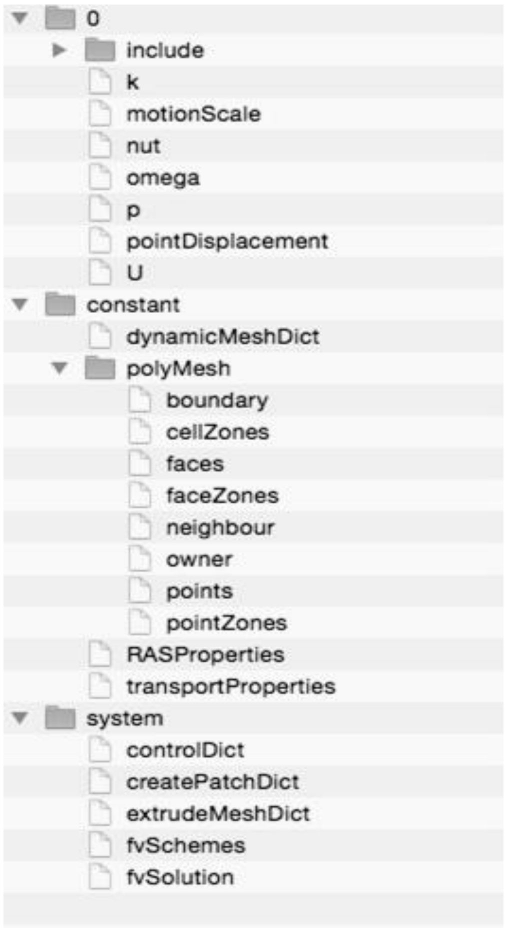
The illustrated case problem example above is governed by the simple temperature transport equation of the form presented in Equation 3.1:

Equation 3.1

$$\frac{\partial \rho T}{\partial t} + \nabla \cdot (\rho \vec{v} T) - \left(\nabla \cdot \Gamma \nabla T \right) = 0$$

2. *Setting the Geometry*

As seen in the above illustration (Figure 3.1), the first requirement when setting up a case problem is to import the geometry of the problem. uFVM has the ability to read the geometrical information necessary for building the geometry and the mesh generated from an OpenFOAM® case folder. OpenFOAM® (Jasak, Jemcov et al. 2013), which stands for Open Source Field Operation and Manipulation, is an open source object-oriented C++ continuum mechanics library that has the characteristic of converting geometries and meshes generated by any of the major mesh generators and CAD systems by the use of some specific commands (*ansysToFoam*, *cfxToFoam*, *fluent3DMeshToFoam*, *gambitToFoam*, *star3ToFoam* among others.)



```
0
    include
    k
    motionScale
    nut
    omega
    p
    pointDisplacement
    U
constant
    dynamicMeshDict
    polyMesh
        boundary
        cellZones
        faces
        faceZones
        neighbour
        owner
        points
        pointZones
    RASProperties
    transportProperties
system
    controlDict
    createPatchDict
    extrudeMeshDict
    fvSchemes
    fvSolution
```

Figure 3.2: The basic structure of an Open FOAM test case

The ‘ 0 ’ directory is the first time sub-folder that contains the initial conditions of each property or variable as shown in Figure 3.2. The ‘constant’ directory contains the dynamicMeshDict that is a dictionary only created for dynamic mesh, and a ‘polyMesh’ directory special for the description of the problem’s geometry. The ‘system’ directory contains dictionaries that define the case setup, the controlDict is concerned with the general control parameters of the test case, the *fvSchemes* defines the discretization schemes while the *fvSolution* contains information about the solution algorithms and relaxations to be used in the simulation.

After obtaining the mesh in an OpenFOAM® format, UFVM can easily read and recognize all the needed geometrical data through the use of a specialized function *cfdReadOpenFoamMesh*. This function starts by reading the points file from OpenFOAM® case folder (constant/polyMesh directory) and storing the x, y, z coordinates into a structure array called the *nodes* array, then the faces file is read and the nodes’ indices, that constitute each face, are stored into the *faces* structure array. Information about the patches and the associated patch faces is then read from the boundary file and stored in the *boundary* structure array. Finally, the *owners* and *neighbors* files are read and the *elements* structure array is composed. At this stage, the available data enables us to construct the elements and setup the mesh nodes’ connectivity. The mesh is then processed in UFVM using *cfdProcessOpenFoamMesh* where the volume and centroid coordinates of each element are calculated as well as the surface area, the interpolation factor and other properties for each face among other geometrical entities.

Now, the mesh structure array contains: the *nodes* array storing the centroid coordinates of each node, its index, the element/s and the faces for which it belongs; the *faces* array storing the index of each face as well as its centroid’s coordinates, area, interpolation factor, area vector, the owner and the neighbor, patch index and the nodes that construct it; the *elements* array storing the index, the neighboring elements' indexes, the faces' and the nodes' indexes constructing the element, volume, face sign and centroid's coordinates. Moreover, a *boundary* structure array is created that contains the boundary types included in the problem and read from the OpenFOAM® boundary file. The *boundary* array contains an index as well as the type of the boundary, the number of faces and the start face index. The previously described arrays are presented in Figure 3.3.

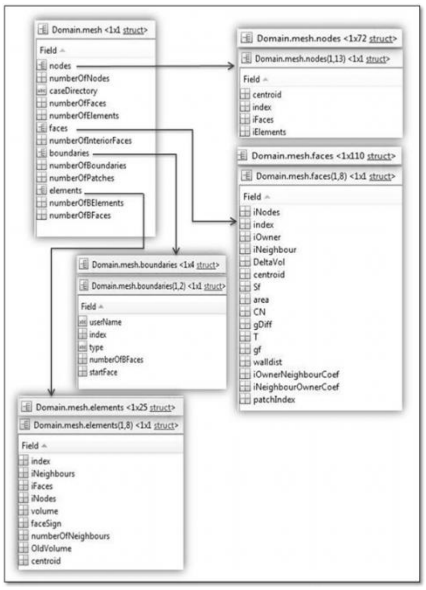


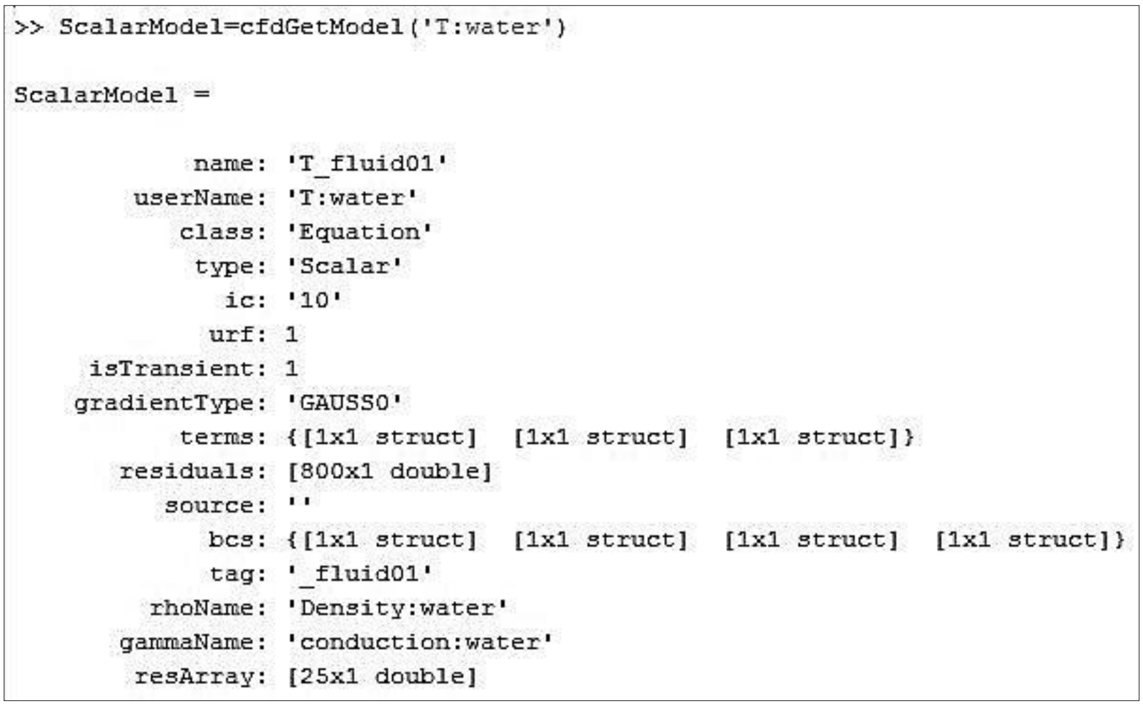
Figure 3.3: The basic structure of uFVM arrays.

3. *Setting up the Model*

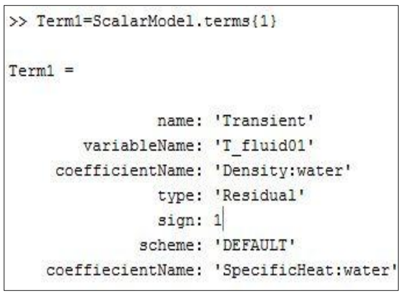
Before setting up the underlying physical equations that are the major constituent of any case problem, the fluid/s involved in the case problem should be defined by the function *cfdSetupFluid*. This function is responsible for defining the type of the fluid (continuous or disperse), its username, molecular weight, mode (compressible or incompressible) in addition to some other related information. It should be noted at this stage that all the data to be used by other functions are saved in a global structure array named as Domain. The thermo-physical variables appearing in the governing equations should be defined as well in a special function *cfdSetupProperty*. In this function, the user should specify the username, type and under relaxation factor among other input information that will be stored in an element mesh field.

Next, the setup of the governing equations of any case problem can proceed by using the function *cfdSetupEquation*. This is where we describe and define the type of the equation (scalar or vector), the initial conditions and the under-relaxation factor. The user can also choose which gradient type to be used for the derivatives appearing in the conservation equations. A field for each equation is stored on an element mesh size array in the global Domain structure. For each defined equation, the user can then add various associated terms (transient, convection, diffusion, pressure gradient, stress, source, electric potential, darcy, buoyancy, and drag among others) that will constitute the equation. Each term is defined in the code using the function *cfdAddTerm*. This function relates each term along with the coefficients (density, viscosity, diffusion coefficient among others) to its equation where the information will be stored for later assembly. The associated boundary conditions of each equation are then added using the function *cfdAddBC* where the type of the boundary (inlet, outlet, specified value, specified flux, slip, no slip among others) and its value, if needed, are defined. If one of the equations of any fluid defined in a case problem contains a convective term, then the user shall setup a 'mdot' (mass flow rate: density multiplied by the dot product of the velocity vector and the area vector) field that creates a mesh field covering the faces of the domain under study.

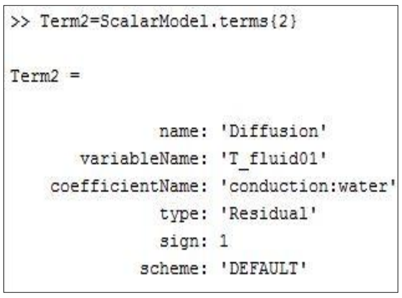
The internal structure of an equation ('T:water' for instance) is presented in Figure 3.4 below.



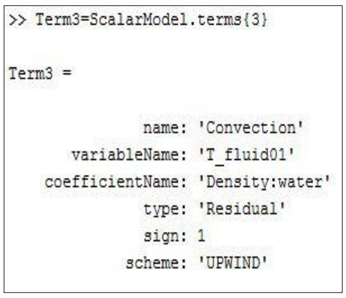
```
>> ScalarModel=cfdGetModel('T:water')

ScalarModel =

            name: 'T_fluid01'
        userName: 'T:water'
           class: 'Equation'
            type: 'Scalar'
              ic: '10'
             urf: 1
     isTransient: 1
    gradientType: 'GAUSS0'
           terms: {[1x1 struct]  [1x1 struct]  [1x1 struct]}
       residuals: [800x1 double]
          source: ''
             bcs: {[1x1 struct]  [1x1 struct]  [1x1 struct]  [1x1 struct]}
             tag: '_fluid01'
         rhoName: 'Density:water'
       gammaName: 'conduction:water'
        resArray: [25x1 double]
```



```
>> Term1=ScalarModel.terms{1}

Term1 =

                name: 'Transient'
        variableName: 'T_fluid01'
     coefficientName: 'Density:water'
                type: 'Residual'
                sign: 1
              scheme: 'DEFAULT'
    coeffiecientName: 'SpecificHeat:water'
```



```
>> Term2=ScalarModel.terms{2}

Term2 =

               name: 'Diffusion'
       variableName: 'T_fluid01'
    coefficientName: 'conduction:water'
               type: 'Residual'
               sign: 1
             scheme: 'DEFAULT'
```



```
>> Term3=ScalarModel.terms{3}

Term3 =

               name: 'Convection'
       variableName: 'T_fluid01'
    coefficientName: 'Density:water'
               type: 'Residual'
               sign: 1
             scheme: 'UPWIND'
```

Figure 3.3: The basic structure of an equation in a u-FVM test case

4. Setup of the Computational Field

In this section, we will describe how the fields are initialized in the code, each on its prescribed locale (elements, faces, nodes). Adding the function *cfdInitializeFields* into the case problem script file will automatically initialize the equation field, the property field and the ‘mdot’ field. An equation is initialized over the elements by computing and distributing the initial conditions onto each interior element along with the boundary conditions previously defined. After that, a property is initialized over the associated mesh field, whether calculated from a formula or a constant value, over elements or faces. The ‘mdot’ field is initialized as well by calling the density field (which is a property already defined and initialized) along with the velocity field to compute the value over each face of the mesh. Storing all the initialized fields in the appropriate arrays for later communication is done by each associated function (*cfdInitializeEquation*, *cfdInitiazeProperty*, *cfdInitializeMdotField*).

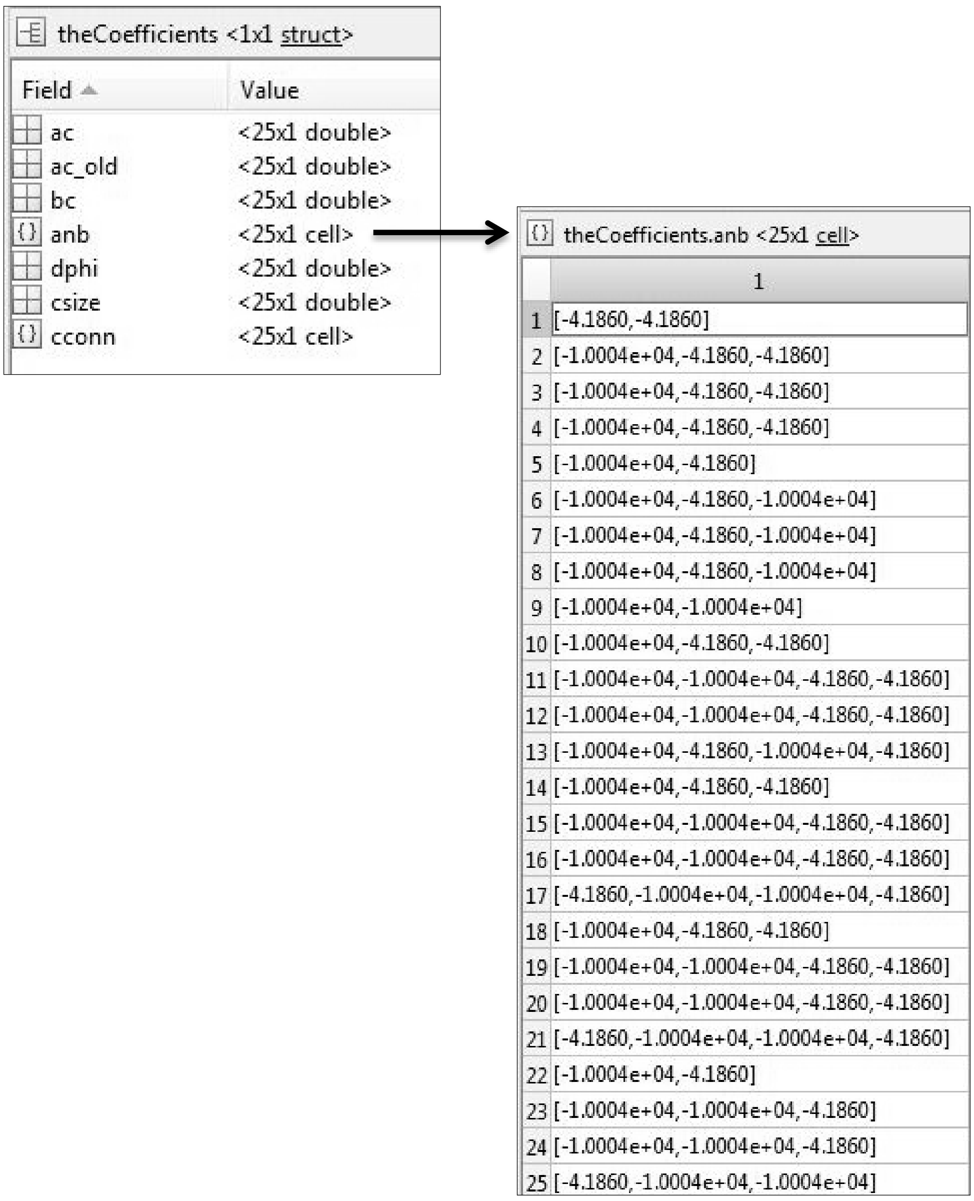
5. *Equation Discretization*

After the environment has been suited according to the chosen case and all the fields have been initialized and prepared for the solution procedure, the function *cfdAssembleAndCorrectEquation* is invoked in order to assemble, solve and correct the equations governing the modeled problem. This function should be stated precisely for each equation if several equations occur in the problem.

6. *Equation Assembly*

For each equation, the internal function *cfdAssembleEquation* is responsible for assembling the terms that have already been associated to the specified equation. A coefficient array containing the coefficient matrices of the a_c, b_c and a_{nb} with the size of the number of control volumes of the domain, is created using the *cfdSetupCoefficients* function.

Illustrations for the coefficients' and the neighbors array are presented in Figure 3.5.


theCoefficients <1x1 struct>

Field	Value
ac	<25x1 double>
ac_old	<25x1 double>
bc	<25x1 double>
anb	<25x1 cell>
dphi	<25x1 double>
csize	<25x1 double>
cconn	<25x1 cell>

theCoefficients.anb <25x1 cell>

	1
1	[-4.1860,-4.1860]
2	[-1.0004e+04,-4.1860,-4.1860]
3	[-1.0004e+04,-4.1860,-4.1860]
4	[-1.0004e+04,-4.1860,-4.1860]
5	[-1.0004e+04,-4.1860]
6	[-1.0004e+04,-4.1860,-1.0004e+04]
7	[-1.0004e+04,-4.1860,-1.0004e+04]
8	[-1.0004e+04,-4.1860,-1.0004e+04]
9	[-1.0004e+04,-1.0004e+04]
10	[-1.0004e+04,-4.1860,-4.1860]
11	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
12	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
13	[-1.0004e+04,-4.1860,-1.0004e+04,-4.1860]
14	[-1.0004e+04,-4.1860,-4.1860]
15	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
16	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
17	[-4.1860,-1.0004e+04,-1.0004e+04,-4.1860]
18	[-1.0004e+04,-4.1860,-4.1860]
19	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
20	[-1.0004e+04,-1.0004e+04,-4.1860,-4.1860]
21	[-4.1860,-1.0004e+04,-1.0004e+04,-4.1860]
22	[-1.0004e+04,-4.1860]
23	[-1.0004e+04,-1.0004e+04,-4.1860]
24	[-1.0004e+04,-1.0004e+04,-4.1860]
25	[-4.1860,-1.0004e+04,-1.0004e+04]

Figure 3.4: The coefficients and neighbors' arrays in a u-FVM test case

A process is initiated in *cfdAssembleEquationTerms* to loop over each term and calculate its fluxes according to a selected or default scheme. The calculated face fluxes and element fluxes are then assembled according to the discretized form into the global coefficient matrix by *cfdAssembleIntoGlobalMatrixFaceFluxes* or *cfdAssembleIntoGlobalMatrixElementFluxes* respectively. After assembling all the terms of a single equation and obtaining the complete coefficient array, the under-relaxation factor already specified for each equation is applied over the initial coefficient matrix a_c.

7. Solving the Equations

The solving procedure in the uFVM code is based on one of two implemented solvers, the successive over-relaxation method (SOR) and the Incomplete lower upper decomposition with no fill-in ILU(0) solver. The iterative SOR method is a generalization of and an improvement on the Gauss-Seidel Method for solving a linear system of equations. The coefficient array containing the coefficient matrices is imported to the solver that in turn loops over the elements of the domain to solve, get and update the $dphi$ in which the solver solves for according to Equation 3.2:

Equation 3.2

$$\delta\phi_c = \frac{b_c - \sum_{NB} a_{NB}\delta\phi_{NB}}{a_c}$$

On the other hand, the ILU(0) method follows the methodology of solving the system of equations in a residual form and has been used as a standard smoother with algebraic multi-grid solvers in many applications. ILU(0) is implemented in the uFVM code and can be selected by the user as the solver of the modeled problem.

8. Correcting the Equations

Whether the user is solving a unique simple scalar equation or a set of scalar and vector equations (Navier Stokes' Equations), the equations have to be corrected using the *cfdCorrectEquation* function along with several internal functions specific for the correction of each type of equation velocity, scalar or pressure (which has specific treatment). In the scalar type or vector type, that resembles the scalar equation yet have a three components to be considered, the value of phi is updated and corrected by adding the $dphi$ term previously computed from the solving part for the internal elements. Boundary conditions are corrected respectively each according to the specified type by the use of adequate functions (*cfdCorrectWallZeroFluxBC*, *cfdCorrectWallSpecifiedFluxBC*, *cfdCorrectInletInletBC*, *cfdCorrectOutletZeroFluxBC* among others). The pressure equation is corrected by the use of *cfdCorrectPPField* and *cfdCorrectPressureEquation* because we need to correct the pressure correction equation and then the pressure field. After that, the velocity and the 'mdot' fields need to be corrected at the interior elements-faces as well the boundaries.

9. *Computing the Residuals*

The residual of each equation or each component of an equation (in case the equation is of the vector type) is calculated using the *cfdComputeResidual* where the residual on each control volume is computed and averaged over the domain. At this stage one residual value for the whole equation or component of an equation is stored. The residual is needed as an indicator for the convergence of the solution and can plotted by a specific function to be mentioned later. First, a scaled ϕ for the equation to be solved is calculated as Equation 3.3:

Equation 3.3

$$\phi_{scale} = \max(\phi_{\max} - \phi_{\min}, abs(\phi_{\max}))$$

Then, the residuals are scaled using Equation 3.4:

Equation 3.4

$$R^{\phi}_{c,scaled} = \sum_{elements} \frac{b_c - a_c\phi_c - \sum_{NB} a_{NB}\phi_{NB}}{a_c\phi_{scale}}$$

And finally, the root-mean square residual is employed as in Equation 3.5:

Equation 3.5

$$R^{\phi}_{c,scaled} = \sqrt{\frac{\sum_{c,cells} (R^{\phi}_{c,scaled})^2}{number \quad of \quad cells}}$$

10. Plotting Utilities

Several plotting functions are available in uFVM code so that the results of any case problem can be visualized as well as the mesh and the geometry.

- *cfdPlotMesh* plots the domain under consideration of a case problem along with the mesh that covers it.
- *cfdPlotElements* plots any element the user choose or a set of specified elements using the index of each.
- *cfdPlotFaces* plots the faces of the domain using their index.
- *cfdPlotPatches* plots the full boundary patch that the user choose by using the index of the patch already defined in earlier stages of a case problem.
- *cfdPlotField* plots any field defined in the solver. Refer to Figure 3.6.

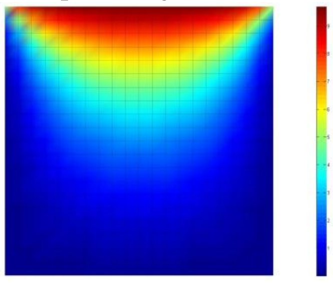


Figure 3.5: A sample plot of a chosen field in u-FVM code

- *cfdPlotResiduals* plots the residual value of each equation or component of an equation versus per iteration. Refer to Figure 3.7.

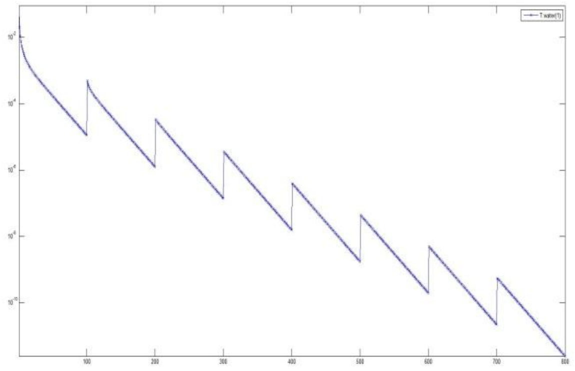


Figure 3.6: Residuals plot generated while simulating a u-FVM test case

- *cfdPlotVelocity* plots the mesh and the velocity vectors on the centroid of each control volumes and on boundary faces. Refer to Figure 3.8.

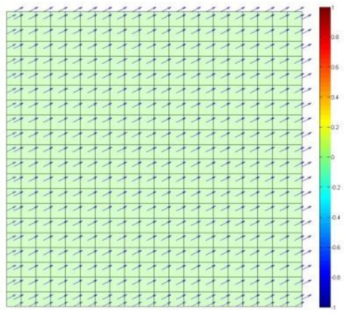


Figure 3.7: Velocity vectors plot of a u-FVM advection test case

C. Gradient Calculation

For structured orthogonal grids, the gradient of a scalar at a given element centroid can be easily computed using the definition of the derivatives. This phenomenon becomes more complicated when general unstructured grids are involved. The usual approach is to make use of Green-Gauss theorem, which states that the surface integral of a scalar function is equal to the volume integral (over the volume bound by the surface) of the gradient of the scalar function (Pfeffer 1991) as presented in Equation 3.6:

Equation 3.6

$$\int_{\Omega} \nabla\phi \, d\Omega = \int_{S} \phi \vec{n} \, dS$$

Where $\vec{n}$ is the surface normal pointing outwards from the control volume. Assuming that ∇f is constant over the control volume, the Green-Gauss equation can be written as Equation 3.7.

Equation 3.7

$$\int_{\Omega} \nabla\phi \, d\Omega = \nabla\phi_P \Omega = \int_{S} \phi \vec{n} dS$$

Next, the integral over the surface is approximated as a summation of the average scalar value in each face times the face's surface vector as in Equation 3.8.

Equation 3.8

$$\nabla\phi_P = \frac{1}{\Omega} \sum_{faces} \phi_f \overrightarrow{S_f}$$

The average face value f_f is computed following two approaches, the first is cell based in which the face value is computed using the values at its straddling cells using *cfdComputeGradientGauss* and the other approach is node based in which the face value is computed using the values at its straddling nodes using *cfdComputeGradientNodal*.

The use of any of the two mentioned methods is not specific; they can be used in any occasion once the face gradient is required.

1. ***Cell-Based Method***

In the cell-based method, we define g_f as the weighing geometric factor between cells P and N as in Equation 3.9.

Equation 3.9

$$g_f = \frac{\left|\overrightarrow{r_N} - \overrightarrow{r_f}\right|}{\left|\overrightarrow{r_N} - \overrightarrow{r_P}\right|}$$

Then, a simple approximation for the face value is defined by a compact stencil, where only the cells straddling the face are only involved in the interpolation. Such an approximation is also known as the weighted approximation and can be written as Equation 3.10.

Equation 3.10

$$f_f = g_f f_P + (1 - g_f) f_N$$

We sum the values of f_f for the faces constituting the element and divide by the volume of the control volume to obtain an average gradient.

2. Node-Based Method

In the node-based method, the value of f_f can be computed as the mean of the nodes defining the face. First, the values at the nodes are obtained by an interpolation from elements to nodes. Then, the values at the nodes are interpolated into faces using a specific interpolation scheme to be described in a later section. We the sum the values of f_f for the faces constituting the element and divide by the volume of the control volume to obtain an average gradient.

D. Interpolation Schemes

Several interpolation functions are included in the uFVM code each serves for a specific function. A summary of these interpolation functions is given below.

- *cfdInterpolateFromElementsToNodes* function is used to compute the gradient according to the node-based method and in the *cfdPlotField* function. A loop over all the nodes is performed and for each node an array stores the elements sharing the specified node. The value of each element is divided by the magnitude of the distance from the node to each element's centroid and we sum up to obtain the value at the node.
- *cfdInterpolateFromElementsToFaces* function is used to assemble the stress, diffusion and 'mdot' terms as well as in the initialization of the fields. Three interpolation schemes (Hyperbolic, Upwind and Average) are implemented in this function so that the user can choose which to use when computing any face value from elements values.
- *cfdInterpolateFromNodesTofaces* function is used to compute the gradient according to the node-based method. For a single face, the value at each node constituting the face is divided by the magnitude of the distance from the face centroid to the node centroid and a summation of these values gives the value at the selected face.
- *cfdInterpolateGradientsFromElementsToInteriorFaces* function is used to interpolate the gradients from elements to interior faces according to the selected interpolation scheme. Four interpolation schemes are implemented in this function (Average, Upwind, Downwind and Corrected Average). The Average schemes depends on the weighing geometric factor and includes the owner and neighbor of the interior face, while the Upwind scheme uses the value of the upwind element and the Downwind uses the value of the downwind element depending on the direction of the 'mdot' vector at the specified interior face.

The corrected Average scheme resembles the average scheme but with the introduction of a correction to the interpolated gradient following the below mathematical Equations 3.11, 3.12, 3.13 and 3.14.

Equation 3.11

$$\nabla f_f = \overline{\nabla f_f} + \left[\frac{f_F - f_C}{\left\| d_{CF} \right\|} - \left(\overline{\nabla f_f} \cdot e_{CF} \right) \right] e_{CF}$$

Where, Equations 3.12 through 3.14 define the parameters of Equation 3.11.

Equation 3.12

$$\overline{\nabla f_f} = g_C \nabla f_C + g_F \nabla f_F$$

Equation 3.13

$$e_{CF} = \frac{d}{\left\| d_{CF} \right\|}$$

Equation 3.14

$$d_{CF} = c - f$$

E. Solution Algorithm:

The solution algorithm in the UFVM-3D code for the calculation of unsteady flows with moving mesh arrangement is envisaged to be composed of the following sequence of steps:

1. Provide the initial grid and the values of the dependent variables (initial conditions).
2. Determine the location of each grid node after the time has advanced by Δt, which represents the new mesh configuration. Attention should be paid to the boundaries when moving with time. The number of control volumes is kept constant throughout the simulation.
3. Assemble and solve the equations for the velocity components, employing the currently available pressure and mass fluxes.
4. Calculate the new mass fluxes to update the field using the new velocity components.
5. Assemble and solve the pressure correction equation.
6. Correct the mass fluxes, velocity components and pressure by the calculated pressure correction.
7. Assemble and solve any other scalar equation which may be coupled with the momentum and update the fluid properties if necessary.
8. Return to step 3 and repeat until a converged solution is obtained.

Advance the time by the time increment Δt and return to step 2; repeat the process until the prescribed number of time steps is completed.

CHAPTER IV

GOVERNING EQUATIONS

A. Introduction:

In order to cover a wider variety of flow problems, the grid is enabled to move with time in a prescribed manner. This grid movement is introduced under the frame of the Arbitrary Lagrangian-Eulerian (ALE in short) approach that deals with the changes to be considered while formulating the conservation equations. In this chapter, we will formulate our equations of fluid flow in the ALE form which in turn will ensure that the geometric conservation laws (GCL in short) such as space conservation law (SCL in short) and Volumetric conservation law (VCL in short) are satisfied in an intrinsic manner. Further, we will elaborate on the approach of moving the mesh randomly in time the difficulties that rise accordingly.

B. The Arbitrary Lagrangian-Eulerian Approach:

1. Introduction:

There exists several approaches for solving fluid flow problems with moving boundaries and moving meshes, we mention of which the integrated space-time approach (Zwart, Raithby et al. 1998, Zwart, Raithby et al. 1999), the Arbitrary Lagrangian-Eulerian (ALE) approach (Demirdžić, Perić 1990), and the boundary-transformation approach (Ralph, Pedley 1989, Guilmineau, Queutey 2002). Among the mentioned approaches, the ALE approach appears more convenient to implement because of having high capabilities in mesh transformation. Thus, the ALE approach will be adopted in this work to solve problems involving moving grids.

The numerical simulation of flow problems may involve in certain situations strong distortions of the continuum under consideration especially in the case of free surface flows, fluid-fluid, or fluid-structure interaction. Hence, when developing a computer code, it is essential to choose the convenient kinematical description of the continuum to be studied. The algorithms of continuum mechanics basically follow one of two classical approaches for the continuum motion: The Lagrangian approach or the Eulerian approach. The Arbitrary Lagrangian-Eulerian approach (ALE in short) was originally developed to overcome the drawbacks of each of the Lagrangian and Eulerian approaches on one side and on the other side to benefit of the advantages of each approach when followed alone as best as possible. The Lagrangian approach or the method of characteristics, where each node of the computational mesh sticks and follows a corresponding material particle during motion, is usually suitable for the simulations of incompressible fluid dynamics problems but is mainly used in structural mechanics. This approach allows for easy tracking of the free surfaces and interfaces between fluids and/or structures. The weakness of this approach lies in its inability to follow large structural motions of the computational domain without requiring frequent re-meshing procedures. On the other hand, the Eulerian approach is mainly preferred to be used in fluid dynamics. It is in this approach that the continuum under consideration moves with respect to the fixed computational grid. Here, large structural motions in the continuum motion can be handled with relative ease comparable to that of the Lagrangian approach, but generally at the expense of interface precision and flow resolution.

In the ALE approach, computational nodes are allowed to move with the continuum with an arbitrary specified velocity that if exactly match the velocity of the material leads to the pure Lagrangian treatment, or if exactly equals to zero leads to the pure Eulerian treatment.

In this chapter, we will establish the conservation equations governing fluid flow according to the ALE approach in a finite volume frame of work for the incorporation of the ALE capabilities increases the strength. Two challenges will rise in the ALE approach are the time accuracy achievement and the satisfaction of the geometric conservation laws while marching in time.

2. The Lagrangian and Eulerian Descriptions:

In the continuum mechanics, two domains are generally used: the material domain $R_X \subset \mathbb{R}^n$ where n represents the spatial dimensions, and the spatial domain R_x. Noting that X and x represent the material points' coordinates and the spatial points' coordinates respectively, we consider that Ω is a continuous medium in the space $\mathbb{R}^n$ and that $t \in [0, \infty) \subset \mathbb{R}$ is the time variable.

The Lagrangian approach describes how the computational grid follows the material points in their motion. As previously stated, it is in this approach that the grid points are permanently attached to the same material points; hence, the motion can be related directly between the spatial coordinates and the material coordinates and is defined by the following mapping function φ :

Equation: 4-1

$$\varphi : R_{\vec{X}} \times \left[t_0, t_{final} \right[\rightarrow R_{\vec{x}} \times \left[t_0, t_{final} \right[$$
$$\left(\vec{X}, t\right) \rightarrow \varphi\left(\vec{X}, t\right) = \left(\vec{x}, t\right)$$

The above expression represents a change of coordinates and should verify the following condition:

Equation: 4-2

$$N \equiv \det\left(\frac{\partial x}{\partial \vec{X}}\right) = \det\left(\mathbb{N}\right) \neq 0$$

In order to have a reversible mapping function $\left(\vec{X}, t\right) = \varphi^{-1}\left(\vec{x}, t\right)$, so as to identify the initial position of the material particles at any instant of time t, the above expression should satisfy the condition (at: $t = 0, N \succ 0$).

At this stage, it is suitable to represent the gradient of φ in a matrix form:

Equation: 4-3

$$\nabla_{(\vec{X},t)}\varphi = \frac{\partial \varphi}{\partial\left(\vec{X}, t\right)} = \begin{pmatrix} \dfrac{\partial \vec{x}}{\partial \vec{X}} & \vec{v} \\ \vec{0}^T & 1 \end{pmatrix}$$

Where 0^T represents a null row vector and the material velocity $\vec{v}$ is:

Equation: 4-4

$$\vec{v}\left(\vec{X}, t\right) = \left.\frac{\partial \vec{x}}{\partial t}\right|_{\vec{X}}$$

With the symbol $(\ |_{\vec{X}})$ meaning that we hold the material coordinate X fixed with time.

It is to be pointed out that there occur no convective effects in the Lagrangian approach since the material points correspond to the same spatial points during the motion, i.e. the substantial (material) derivative simplifies to a time derivative.

On another hand, the Eulerian approach describes as time goes by, the physical properties of the fluid particles crossing a through a fixed region of space. In this approach, the continuum moves with respect to the fixed computational grid points and the conservation equations are derived in spatial coordinates x and time coordinates t. In this case, the material velocity v of any node on the mesh represents the velocity of the corresponding material point concurring at the same time t with the node under consideration. Therefore, the velocity v is expressed relative to the fixed computational grid points without any relation to the initial configuration and/or the material points' coordinates X. It is expressed according to the following form: $\vec{v} = \vec{v}\left(\vec{x}, t\right)$

Due to the fact that the Eulerian approach separates and distinguishes the material points from the computational points, convective effects come to appear in the flow equations to include the relative motion between the physical material and the computational grid.

3. The Arbitrary Lagrangian-Eulerian Approach:

a. <u>Kinematical Description</u>

The Lagrangian approach, of which each point of the computational mesh coincides with a corresponding material point during motion, is mainly used in structural mechanics. This approach eases up the procedure of tracking the free surfaces and interfaces between different materials. Its weakness is its inability to represent large distortions of the computational domain without the aid of frequent re-meshing techniques. On the Other hand, the Eulerian approach is usually followed in fluid dynamics. In this approach, large distortions of the continuum motion can be managed and followed, but with some compromise with the precision of the interface definition and the resolution of flow details (Donea, Huerta et al. 2004).

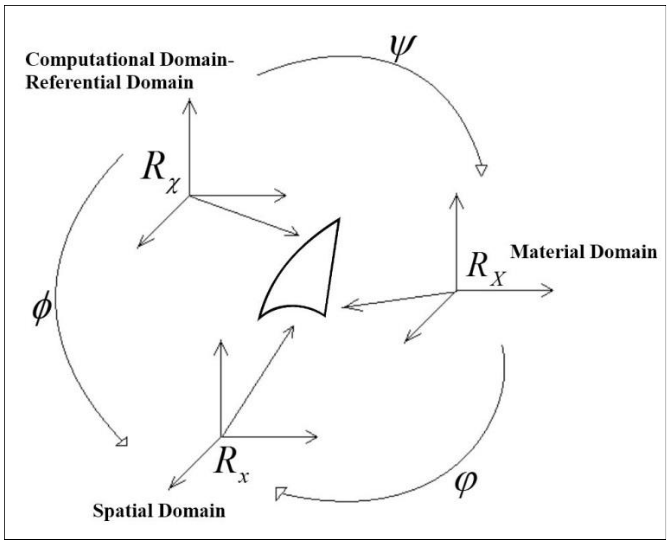
From these aforementioned drawbacks of the two approaches, the ALE approach emerges to combine the best aspects of the two approaches when followed each alone i.e. the large distortions are handled with more resolution.

In the ALE approach, the points of the computational mesh may be moved with the continuum configuration according to an arbitrary specified velocity, that if exactly match the velocity of the material leads to the pure Lagrangian approach treatment, or if exactly equals to zero leads to the pure Eulerian approach treatment. Hence, we can say that the ALE approach comes as a generalization technique where the pure Lagrangian and the pure Eulerian approaches would be special cases. In this chapter, we will establish the conservation equations governing fluid flow in the ALE form.

ALE methods were first presented in a finite difference and a finite volume frame of work. The original advancements on this approach were proposed by (Noh 1963), (Franck, Lazarus 1964), (Trulio 1966), and (Hirt, Amsden et al. 1974) among others. The approach was afterwards introduced in the finite element frame where early applications are included in the work of (Donéa, Fasoli-Stella et al. 1977), (Belytschko, Kennedy et al. 1980), (Belytschko, Kennedy 1978) and (Hughes, Liu et al. 1981). On another hand, some authors like (Naderi, Darbandi et al. 2010) attempted to incorporate the ALE approach in a mixed finite volume-element frame.

In the ALE kinematical description, neither the material domain R_X nor the spatial domain R_x is considered as the reference. For this reason, a third domain R_c is suggested and referred as the referential domain. Assuming that $\vec{\chi}$ represents the referential points' coordinates that track the grid points. The referential domain is transformed into the material and spatial domains by the transformation functions ψ and ϕ respectively. The particle motion φ is also expressed in terms of ϕ and ψ according to the following independent form: $\varphi = \phi \circ \psi^{-1}$. The transformations from/to the three domains are illustrated in Figure 4.1.

Figure: 4.1: The transformation functions between the Computational, Material and Spatial Domains



For further elaboration, the transformation of ϕ from the referential domain to the spatial domain, which represents the motion of the computational points in the spatial domain, is expressed by:

Equation: 4-5

$$\phi : R_\chi \times \left[t_0, t_{final} \right[\rightarrow R_x \times \left[t_0, t_{final} \right[$$

Equation: 4-6

$$(\vec{\chi}, t) \rightarrow \phi(\vec{\chi}, t) = (\vec{x}, t)$$

And the gradient of ϕ is represented as:

Equation: 4-7

$$\nabla_{(\vec{\chi},t)}\phi = \frac{\partial \phi}{\partial(\vec{\chi},t)} = \begin{pmatrix} \dfrac{\partial \vec{x}}{\partial \vec{\chi}} & \vec{v}\,' \\ \vec{0}^T & 1 \end{pmatrix}$$

In the above representation, $\vec{v}\,'$can be written as:

Equation: 4-8

$$\vec{v}\,'(\vec{\chi},t) = \left.\frac{\partial \vec{x}}{\partial t}\right|_{\vec{\chi}}$$

With the symbol $\left(\ |_{\vec{\chi}} \right)$ meaning that we hold the referential domain χ fixed in time.

At this point, we can note that the material and mesh velocities, $\vec{v}$ and $\vec{v}\,'$ respectively, are represented as a derivative with respect to time, that is because the material and the mesh are moving with respect to the observer.

It is adequate at this stage to represent the inverse of the transformation function ψ directly as follows:

Equation: 4-9

$$\psi^{-1} : R_{\vec{X}} \times \left[t_0, t_{final} \right[\rightarrow R_{\vec{\chi}} \times \left[t_0, t_{final} \right[$$

Equation: 4-10

$$\left(\vec{X},t\right) \rightarrow \psi^{-1}\left(\vec{X},t\right) = \left(\vec{\chi},t\right)$$

And the gradient of this transformation function is expressed in a matrix form as follows:

Equation: 4-11

$$\nabla_{(X,t)}\psi^{-1} = \frac{\partial \psi^{-1}}{\partial\left(X,t\right)} = \begin{pmatrix} \dfrac{\partial \vec{\chi}}{\partial \vec{X}} & \vec{w} \\ \vec{0}^T & 1 \end{pmatrix}$$

And the particle velocity in the referential domain $\vec{w}$ is defined as follows:

Equation: 4-12

$$\vec{w} = \left.\frac{\partial \vec{\chi}}{\partial t}\right|_{\vec{X}}$$

With the material domain $\vec{X}$ being held fixed.

After defining the three velocities $\vec{v}$, $\vec{v}\,'$ and $\vec{w}$, it is convenient to define a relation between them by differentiating equation ($\varphi = \phi \circ \psi^{-1}$) according to the following form:

Equation: 4-13

$$\frac{\partial \varphi}{\partial (X,t)}(X,t) = \frac{\partial \phi}{\partial (\chi,t)}\left(\psi^{-1}(X,t)\right)\frac{\partial \psi^{-1}}{\partial (X,t)}(X,t) = \frac{\partial \phi}{\partial (\chi,t)}(\chi,t)\frac{\partial \psi^{-1}}{\partial (X,t)}(X,t)$$

The velocities relation can also be represented in a matrix form:

Equation: 4-14

$$\begin{pmatrix} \dfrac{\partial x}{\partial X} & \vec{v} \\ \vec{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x}{\partial \chi} & \vec{v}\,' \\ \vec{0}^T & 1 \end{pmatrix}\begin{pmatrix} \dfrac{\partial \chi}{\partial X} & \vec{w} \\ \vec{0}^T & 1 \end{pmatrix}$$

Applying block multiplication to the matrix equation:

Equation: 4-15

$$\begin{pmatrix} \dfrac{\partial x}{\partial X} & \vec{v} \\ \vec{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x}{\partial \chi} \times \dfrac{\partial \chi}{\partial X} + \vec{v}\,' \times \vec{0}^T & \dfrac{\partial x}{\partial \chi} \times \vec{w} + \vec{v}\,' \\ \vec{0}^T \times \dfrac{\partial \chi}{\partial X} + \vec{0}^T & \vec{0}^T \times \vec{w} + 1 \end{pmatrix}$$

That returns for the material velocity as,

Equation: 4-16

$$\vec{v} = \vec{v}\,' + \frac{\partial x}{\partial \chi} \cdot \vec{w}$$

Or,

Equation: 4-17

$$\vec{c} := \vec{v} - \vec{v}\,' = \frac{\partial x}{\partial \chi} \cdot \vec{w}$$

The new velocity $\vec{c}$ is referred as the convective velocity and represents the relative velocity between the material and the computational mesh. The reader should not confuse between $\vec{c}$ and $\vec{w}$, as $\vec{w}$ is the particle velocity as seen from the referential domain, whereas $\vec{c}$ is the particle velocity relative to the mesh as seen from the spatial domain. We can notice that $\vec{c}$ is composed of $\vec{v}$ and $\vec{v}\,'$, and both are dependents on coordinate x.

b. The Equations Formulation

At this point, the conservation laws of mass, momentum and energy in the ALE framework are presented. One of the prerequisites to achieve the mentioned framework is to relate the material (total) time derivative to the referential time derivative which is described in the following subsection:

i. Material, Spatial and Referential Time Derivatives

Consider a scalar physical quantity referred as $f(x,t)$, $f^*(\chi,t)$ and $f^{**}(X,t)$ in the spatial, referential and material domains respectively. We shall note here that the star superscripts are used to differentiate between the functional forms for each domain. Referring to the Figure: 4-2-1, the spatial quantity $f(x,t)$ and the material quantity $f^{**}(X,t)$ are related by the particle motion φ, which has been previously introduced, according to the following form:

Equation: 4-18

$$f^{**}(X,t) = f\left(\varphi(X,t),t\right)$$

And for simplicity can be represented as:

Equation: 4-19

$$f^{**} = f \circ \varphi$$

The gradient of the above expression is presented as:

Equation: 4-20

$$\frac{\partial f^{**}}{\partial(X,t)}(X,t) = \frac{\partial f}{\partial(x,t)}(x,t)\frac{\partial \varphi}{\partial(X,t)}(X,t)$$

Or in a corresponding matrix form:

Equation: 4-21

$$\begin{pmatrix} \frac{\partial f^{**}}{\partial \vec{X}} & \frac{\partial f^{**}}{\partial t} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial \vec{x}} & \frac{\partial f}{\partial t} \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{x}}{\partial \vec{X}} & \vec{v} \\ \vec{0}^T & 1 \end{pmatrix}$$

The above form settles after block multiplication on two expressions:

First,

Equation: 4-22

$$\frac{\partial f^{**}}{\partial \vec{X}} = \frac{\partial f}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \vec{X}}$$

Second,

Equation: 4-23

$$\frac{\partial f^{**}}{\partial t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\vec{v}$$

The second equation relates the material time derivative to the spatial time derivative and can be cast in a better form to yield:

Equation: 4-24

$$\left.\frac{\partial f}{\partial t}\right|_X = \left.\frac{\partial f}{\partial t}\right|_x + \vec{v}.\nabla f$$

To ease the representation, the above form can be equivalently written as:

Equation: 4-25

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \vec{v}.\nabla f$$

Interpreting the previous form, we mention that the material (total) time derivative is the local variation plus the convective term representing the relative motion between the material and the spatial domain.

The transformation ψ will help us extend the previous relation between the material and spatial time derivatives to include now the referential time derivative and the following expression is obtained:

Equation: 4-26

$$f^{**} = f^{*} \circ \psi^{-1}$$

The gradient of the above expression is written as:

Equation: 4-27

$$\frac{\partial f^{**}}{\partial (X,t)}(X,t) = \frac{\partial f^{*}}{\partial (\chi,t)}(\chi,t)\frac{\partial \psi^{-1}}{\partial (X,t)}(X,t)$$

Or in an equivalent matrix form as follows:

Equation: 4-28

$$\begin{pmatrix} \frac{\partial f^{**}}{\partial X} & \frac{\partial f^{**}}{\partial t} \end{pmatrix} = \begin{pmatrix} \frac{\partial f^{*}}{\partial \chi} & \frac{\partial f^{*}}{\partial t} \end{pmatrix} \begin{pmatrix} \frac{\partial \chi}{\partial X} & \vec{w} \\ \vec{0}^{T} & 1 \end{pmatrix}$$

And after applying the block multiplication provides the new expression:

Equation: 4-29

$$\frac{\partial f^{**}}{\partial t} = \frac{\partial f^{*}}{\partial t} + \frac{\partial f}{\partial t} \cdot \vec{c}$$

At this stage, it is convenient to present the fundamental Arbitrary Lagrangian-Eulerian relation between the material time derivative, referential time derivative and the spatial gradient as follows:

Equation: 4-30

$$\left.\frac{\partial f}{\partial t}\right|_{X} = \left.\frac{\partial f}{\partial t}\right|_{\chi} + \frac{\partial f}{\partial t} \cdot \vec{c} = \left.\frac{\partial f}{\partial t}\right|_{\chi} + \vec{c} \cdot \nabla f$$

The above ALE relation presents the time derivative of the physical quantity f for a given particle X -the material derivative, which consists of the local derivative (holding the reference coordinate χ fixed in time) in addition to a convection term that accounts for the relative velocity $\vec{c}$.

Before establishing the integral form of the basic conservation laws for mass, momentum and energy, we are required to consider the rate of change of scalar and vector integrals over a moving volume occupied by fluid.

ii. Temporal Derivation of Integrals over Moving Control Volumes

Starting from a Lagrangian frame, consider a material volume represented by V_t in which it is bounded by a smooth closed surface S_t. The consisting points of the volume shall move with the material velocity $\vec{v} = \vec{v}(x,t)$ such that $x \in S_t$. A material volume in this frame is a volume that permanently consists of the same particles of fluid under consideration. The material time derivative of the integral of a scalar function $f(x,t)$ over the material volume V_t that is changing with time is represented by the following expression:

Equation: 4-31

$$\frac{d}{dt}\int_{V_t} f\left(x,t\right)dV = \int_{V_c=V_t} \frac{\partial f\left(x,t\right)}{\partial t}dV + \int_{S_c=S_t} f(x,t)\vec{v}.\vec{n}\,dS$$

The above expression is referred as the Reynolds Transport Theorem (RTT in short) and holds for the smooth function $f(x,t)$. The first term on the right side represents the volume integral of the rate of change of the function $f(x,t)$ over a space-fixed control volume that coincides with the material-moving volume at a considered time instant t. On the same side, the second term represents the flux of the scalar quantity f against a fixed boundary S_c that coincides with the closed surface S_t at the same instant of time that brings V_c with V_t. We note that $\vec{n}$ in the surface integral denotes the unit vector always pointing outwards in a normal direction to the surface, while $\vec{v}$ is the material velocity of the points constituting the boundary S_t.

Adding that the following expression is valid:

Equation: 4-32

$$\int_{S_c} f(x,t)\vec{v}.\vec{n}\,dS = \int_{V_c} \nabla.\left(f\vec{v}\right)dV$$

We hence obtain the alternative form of the RTT as follows:

Equation: 4-33

$$\frac{d}{dt}\int_{V_t} f(x,t)\,dV = \int_{V_c=V_t} \left(\frac{\partial f(x,t)}{\partial t} + \nabla.\left(f\vec{v}\right)\right)dV$$

iii. The Integral ALE form of the Conservation Equations

Consider an arbitrary control volume V_t with its boundary $S_t = \partial V_t$ moving with a mesh velocity $\vec{v}$,

Equation: 4-34

$$\left.\frac{d}{dt}\right|_{\chi} \int_{V_t} f(x,t)\,dV = \int_{V_t} \left(\left.\frac{\partial f(x,t)}{\partial t}\right|_{x} \right) dV + \int_{S_t} f(x,t)\vec{v}.\vec{n}\,dS$$

The function $f(x,t)$ can be replaced by density ρ, momentum $\rho\vec{v}$ or energy ρE respectively to start casting the integral ALE form of the mass, momentum and energy conservation equations.

We start the casting procedure from the well-known Eulerian forms by introducing the ALE differential form of the equations as follows:

Mass:

Equation: 4-35

$$\frac{d\rho}{dt} = \left.\frac{\partial \rho}{\partial t}\right|_{x} + \vec{v}.\nabla\rho = -\rho\nabla.\vec{v}$$

Momentum:

Equation: 4-36

$$\rho\frac{d\vec{v}}{dt} = \rho\left(\left.\frac{\partial \vec{v}}{\partial t}\right|_{x} + \left(\vec{v}.\nabla\right)\vec{v} \right) = \nabla.\sigma + \rho\vec{b}$$

Energy:

Equation: 4-37

$$\rho\frac{dE}{dt} = \rho\left(\left.\frac{\partial E}{\partial t}\right|_{x} + \vec{v}.\nabla E \right) = \nabla.\left(\sigma.\vec{v}\right) + \vec{v}.\rho\vec{b}$$

Noting that ρ is the mass density, $\vec{v}$ is the material velocity, σ denotes the Cauchy stress tensor, $\vec{b}$ is the specific body force vector and E is the specific total energy.

The differential form of the ALE conservation laws is presented as such:

Mass:

Equation: 4-38

$$\frac{\partial\rho}{\partial t}+\left(\vec{v}-\vec{v}\,'\right).\nabla\rho=-\rho\nabla.\vec{v}$$

Momentum:

Equation: 4-39

$$\rho\left\{\frac{\partial\vec{v}}{\partial t}+\left[\left(\vec{v}-\vec{v}\,'\right).\nabla\right]\vec{v}\right\}=\nabla.\sigma+\rho\vec{b}$$

Energy:

Equation: 4-40

$$\rho\left\{\frac{\partial E}{\partial t}+\left(\vec{v}-\vec{v}\,'\right).\nabla E\right\}=\nabla.\left(\sigma.\vec{v}\right)+\vec{v}.\left(\rho\vec{b}\right)$$

The differential conservation of mass equation is with the aid of the material derivative integrated over a control volume to yield:

Equation: 4-41

$$\int_{V_t}\left.\frac{d\rho}{dt}\right|_x dV=\int_{V_t}\vec{v}.\nabla\rho\,dV=-\int_{V_t}\rho\nabla.\vec{v}dV$$

Manipulating the above equality, we get:

Equation: 4-42

$$\int_{V_t}\left.\frac{d\rho}{dt}\right|_x dV=-\int_{V_t}\rho\nabla.\vec{v}dV-\int_{V_t}\vec{v}.\nabla\rho\,dV$$

Now, by the aid of the above expression, we successively replace the scalar function $f(x,t)$ in Equation: 4-34 by the respective fluid density, momentum and specific total energy, we get the following bunch of conservation equations in the ALE form:

Integral ALE Mass Conservation Equation:

Equation: 4-43

$$\left.\frac{\partial}{\partial t}\right|_{\chi} \int_{V_t} \rho \, dV + \int_{S_t} \rho \vec{c}.\vec{n} \, dS = 0$$

Integral ALE Momentum Conservation Equation:

Equation: 4-44

$$\left.\frac{\partial}{\partial t}\right|_{\chi} \int_{V_t} \rho \vec{v} \, dV + \int_{S_t} \rho \vec{c}.\vec{n} \, dS = \int_{V_t} \left(\nabla.\sigma + \rho \vec{b} \right) dV$$

Integral ALE Energy Conservation Equation:

Equation: 4-45

$$\left.\frac{\partial}{\partial t}\right|_{\chi} \int_{V_t} \rho E \, dV + \int_{S_t} \rho E \vec{c}.\vec{n} \, dS = \int_{V_t} \left(\vec{v}.\rho \vec{b} + \nabla.\left(\sigma.\vec{v} \right) \right) dV$$

To close up this section, we validate the equations formulation by ensuring that the integral forms for the Lagrangian and Eulerian approaches are special cases of the above ALE forms. The Lagrangian approach corresponds to the situation when $\vec{v}' = \vec{v}$ i.e. $\vec{c} = 0$, while the Eulerian approach corresponds to the situation of which $\vec{v}' = 0$ i.e. $\vec{c} = \vec{v}$.

iv. The Discretized ALE Conservation Equations

The discretization process of the momentum equation and the other conservation equations over moving grids resembles in its principle that followed for stationary grids. Yet, the transient term is discretized in a more general manner by taking into account the changes in the control volumes' volume that show up as the grid moves in time. Using the implicit Euler temporal scheme, we discretize the transient term as follows:

Equation: 4-46

$$\left.\frac{\partial}{\partial t}\right|_{\chi}\int_{V_t}\rho\alpha dV \approx \frac{\left(\rho\alpha\Delta V\right)^{t=n}-\left(\rho\alpha\Delta V\right)^{t=n-1}}{\Delta t}$$

Turning our attention into the continuity equation, we attempt to modify Equation: 4-2-42 by the aid of Equation: 5-2-16 and present it in the following form:

Equation: 4-47

$$\int_{S_t}\rho\vec{v}.\vec{n}\,dS=\int_{S_t}\rho\vec{v}\,'.\vec{n}\,dS-\left.\frac{\partial}{\partial t}\right|_{\chi}\int_{V_t}\rho\,dV$$

The term on the left-hand side represents physically the total mass flux crossing the surface of a control volume in a stationary grid. The right hand side of Equation: 4-2-46 constitutes the contribution of the grid movement. For a fully conservative moving grid procedure, the right hand side shall equate zero. This is achieved when the mass fluxes due to the grid movement cancel out the unsteady term of the equation. This procedure has to be accomplished according to the Volumetric Conservation Law to be presented in a following section. The first term of the right-hand side is added to the mass fluxes which are computed while solving the continuity (pressure-correction) equation, while the second term is added to the source term.

C. The Geometric Conservation Laws:

Upon designing any numerical code for solving the flow equations on grids moving with time, it is necessary to compute some geometric quantities that involve the grid velocity and grid points' positions. Two equations come to the foreground called the Geometric Conservation Laws (GCL in short) (Thomas, Lombard 1979), which form for static and moving grids, the balance between the relevant applicable geometric parameters, have long been ignored in the literature. The first law referred as the Space

Conservation Law (SCL in short) states that the cells' volume should be enclosed by its surfaces during the motion, while the second law referred as the Volume Conservation Law (VCL in short) states that the volumetric increment of a moving control volume should equalize the sum of the changes along its enclosing surfaces. Ensuring the satisfaction of the GCL into the numerical algorithm is advised to avoid undesirable errors in flow fields, moreover, violating these laws may misrepresent the convective fluxes and extra sources may be encountered (Vinokur 1989).

1. *Derivation of GCL Equations*

We start this section by presenting the general scalar transport equation in its simplified form representing the main four terms from left to right as: transient, convective, diffusive and source terms. The equation is as follows:

Equation: 4- 48

$$\frac{\partial(\rho\theta)}{\partial t}+\nabla.(\rho\vec{v}\theta)=\nabla.\Gamma\nabla\theta+Q$$

Upon integrating over the control volume, we get the following expression:

Equation: 4- 49

$$\int_V \frac{\partial(\rho\theta)}{\partial t}dV+\int_V \nabla.(\rho\vec{v}\theta).dV=\int_V \nabla.\Gamma\nabla\theta.dV+\int_V Q\,dV$$

Then with the aid of the Gauss divergence theorem that relates the volume integral to the surface integral (Pfeffer 1991), we get the following form in terms of the variable θ :

Equation: 4- 50

$$\int_V \frac{\partial(\rho\theta)}{\partial t}dV+\int_S (\rho\vec{v}\theta).dS=\int_S \Gamma\nabla\theta.dS+\int_V QdV$$

Generalizing the set of equations (Equation: 4-2-43, Equation: 4-2-44 and Equation: 4-2-45), we can write the general scalar transport equation in an ALE form as follows:

Equation: 4- 51

$$\frac{d}{dt}\int_{V(t)} \rho\theta \, dV + \int_{S(t)} \rho\theta\left(\vec{v}-\vec{v}'\right).dS = \int_{S(t)} \Gamma\nabla\theta.dS + \int_{V(t)} Q \, dV$$

The GCL requires that the state $q = constant$ be an exact solution of the above equation above and by that, the zero viscous fluxes resulted from the differentiation of a constant field are cancelled out. The source term is to be neglected, the density is considered unity and the flow velocity equalizes to zero all to yield

Equation: 4- 52

$$\frac{d}{dt}V(t) = \int_{S(t)} \vec{v}'.dS$$

Equation: 4-3-5 captures the change in time of the total volume of a control volume moving with time (referred as $V(t)$) to the net motion of the bounding surfaces. The equation is hence labeled as the Volumetric Conservation Law.

The second geometric conservation law is called the Space Conservation Law and is obtained by assuming a uniform flow field oriented in an arbitrary direction $\vec{b}$ on a non-moving grid. This will result in the following analytical definition:

Equation: 4- 53

$$\int_{S} \vec{b}.\vec{n} \, dS = 0$$

The above equations will be next discretized and transformed into algebraic equations connecting values at neighboring cells to each other. The discretization process involves approximating the transient, convection, diffusion, and source terms by equivalent algebraic relations through the use of interpolation profiles.

2. Discretization of Laws' Equations

Starting with the VCL equation (Equation: 4-3-8), we follow the first order fully implicit-scheme (Backward Euler) to integrate in time to get:

Equation: 4- 54

$$\int_{t}^{t+\Delta t} \frac{d}{dt} V(t) dt = \frac{V^{t+\Delta t} - V^{t}}{\Delta t}$$

Let $V^{t+\Delta t} = V^{t}$, and $V^{t} = V^{t-\Delta t}$ such that the following expression is obtained:

Equation: 4- 55

$$\int_{t}^{t+\Delta t} \frac{d}{dt} V(t)\, dt = \frac{V^{(t)} - V^{(t-\Delta t)}}{\Delta t} \cong \sum_{iface=1}^{Nf} \left[\vec{v}.n \right]_{iface}$$

But, we have to note that:

Equation: 4- 56

$$\sum_{iface=1}^{Nf} \left[\vec{v}.n \right]_{iface} = \sum_{iface=1}^{Nf} \left[\frac{\delta V}{\Delta t} \right]_{iface}$$

Hence, the discretized form of the VCL equation is presented as follows:

Equation: 4- 57

$$\frac{V^{(t)} - V^{(t-\Delta t)}}{\Delta t} = \sum_{iface=1}^{Nf} \left[\frac{\delta V}{\Delta t} \right]_{iface}$$

To clarify more, the time change of volume of a control volume must equate the sum of the volume increments of each enclosing face divided by the corresponding time step.

Moving to the SCL equation (Equation: 4-3-9), the discrete form is presented as follows:

Equation: 4- 58

$$\sum_{iface=1}^{Nf} \vec{b}.S_{iface} = 0$$

It is to be noticed that the SCL discrete equation emphasizes the necessity of evaluating the surface vectors of each control volume's surfaces exactly such that the summation of the term on the left hand side equates zero. The SCL has been ignored in the literature and some authors used the SCL term in representing the Volumetric Conservation Law without coming to mention the SCL as in (Demirdžić, Perić 1988, Demirdžić, Perić 1990).

A numerical algorithm is said to satisfy the Geometric Conservation Laws (VCL and SCL simultaneously) if the geometric parameters respect Equation: 4-3-5 and Equation: 4-3-6 respectively. Satisfying the two equations is crucial to achieve the global conservation in the domain of interest. The volumes and surfaces are the main fundamentally considered geometric parameters. Therefore, maintaining volumes and surfaces according to the prescribed equations is the key to satisfy the GCLs. However, many numerical algorithms may have other geometrical parameters and hence, computing these dependent parameters in terms of volumes and surfaces is a subsequent technique to get through the satisfaction of the GCLs.

3. Implementation of the GCL:

The integral form of the VCL for an arbitrary moving control volume was previously given by:

Equation: 4- 59

$$\frac{\delta V}{\Delta t} = \sum_{ifaces} \vec{v}\,'.S_{ifaces}^{new} \text{ for, } ifaces = e, w, n, s$$

Remembering that the total volumetric rate of change of the control volume was presented as:

Equation: 4- 60

$$\frac{\delta V}{\Delta t} = \frac{\sum_{ifaces} \delta V_{ifaces}}{\Delta t}$$

Where each component is decomposed as:

Equation: 4- 61

$$\frac{\delta V_{iface}}{\Delta t} = \vec{v}\,'_{iface} . S_{iface}^{new}$$

The convective term in any conservation equation for an arbitrary variable α (α may be 1, u , v , w or ϕ) is semi-discretized as follows:

Equation: 4- 62

$$\sum_{iface} \left(\rho\alpha\right)_{iface} \left(\vec{v} - \vec{v}\,'\right)_{iface} . S_{iface}^{new} = \sum_{iface} \left(\rho\alpha\right)_{iface} \left(\vec{v}_{iface} . S_{iface}^{new} - \vec{v}\,'_{iface} . S_{iface}^{new}\right) = \sum_{iface} \left(\rho\alpha\right)_{iface} \left(\vec{v}_{iface} . S_{iface}^{new} - \frac{\delta V_{iface}}{\Delta t}\right)$$

The total rate of change of the cell volume is hence calculated from the known grid point positions. This method does not require the definition of grid velocities rather the volumetric increment of each bounding surface is calculated at each time step.

This approach proposed by (Demirdžić, Perić 1988) is considered as an easier approach relative to the explicit one especially when applied to three dimensional algorithms, where at each control volume’s surface, three grid velocity components would have to be calculated. As opposed here, our adopted approach proposes the calculation of a single rate of change of the cell volume from a single volumetric increment of each bounding surface.

CHAPTER V

MESH MOVEMENT

When introducing the topic of mesh movement, one will expect that all the mesh nodes move randomly in time, which is a logical general expectation that was taken into consideration in our work. In this chapter, we will elaborate on the approach of moving the internal mesh nodes randomly in time and the arising consequences that were not resolved exactly. This fact has leaded us to the original assumption of moving the internal mesh nodes in constant magnitude and direction during each time step.

In the UFVM code, we already know the coordinates of the new locations of each grid point of the computational domain. Yet, we have to deal with the change in the volume of each control volume due to the grid movement. Each bounding surface of the control volume would move following a certain direction and owning a certain magnitude; the fact that will introduce a volumetric increment of each bounding surface that must sum up to the total volumetric change of the control volume so as to satisfy the Volumetric Conservation Law. Practically, we loop over the faces of the initial mesh configuration just before the mesh starts its movement, and as the nodes move in a prescribed manner, we have the information of the new location of each node forming a certain control volume. The initial configuration along with the information about the new locations of the grid nodes creates the volumetric increment of each surface enclosing a control volume that we aim on computing. First, we locate a center for the volumetric increment of each moving surface that is by summing up the coordinates of the nodes (initial nodes' coordinates along with the coordinates of the new location) and dividing by twice the number of initial nodes. Geometrically, the center will be located in a place bounded by the nodes and inside the virtually drawn volumetric increment. After that, we form vectors relative to each node (at its old and new location) having the center already spotted as the vectors' tip. An illustration of the described treatment is presented in the sketches of Figures 5.1 and 5.2 below, where a hexahedron shaped volumetric increment is created due to the motion of a face between two consecutive time intervals.

Figure: 5. 1: A sketch showing a surface movement with time of a hexahedral control volume

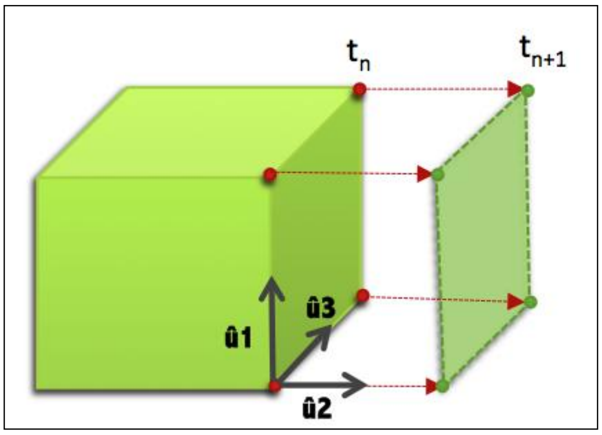
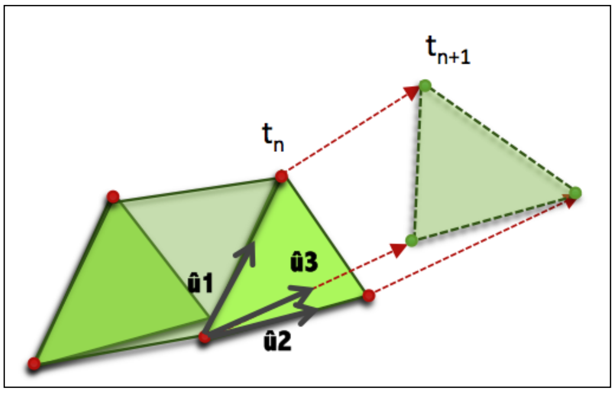


Figure: 5.2: A sketch showing the movement of a triangular face with time

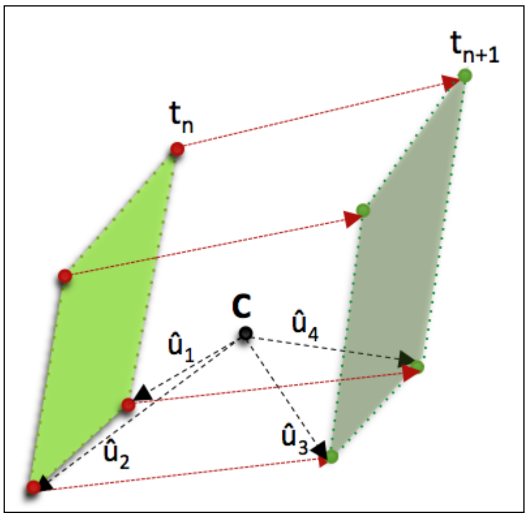


As shown in the (Figure 5-1), the volumetric increment of one of the surfaces of the hexahedral control volume has a cuboid shape with eight nodes; four of which are the surface's initial configuration at the initial time step and the other four represent the respective new locations of the nodes at the next time step. Hence, eight vectors are introduced in this situation for this single surface. These vectors will be used to compute the volume of the virtual pyramids formed by considering each face of the cuboid volumetric increment as the base of the pyramid, whereas the tip of the pyramids is the center previously located and computed. As illustrated in the Figure (5-3), six virtual pyramids appear (one of them is sketched) and the volume each virtual pyramid is calculated according to the following expression, refer to Equation 5.1:

Equation: 5.1

$$Volume_{pyramid} = \frac{1}{6}\left\{\left|\vec{u}_2 . \left(\vec{u}_1 \times \vec{u}_3\right)\right| + \left|\vec{u}_4 . \left(\vec{u}_1 \times \vec{u}_3\right)\right|\right\}$$

Figure: 5.3: A sketch showing the calculation method of the volume increment of a moving surface in 3-D (Pyramid Method)



For the case when the faces of a control volume are triangular, the face volumetric increment would take the shape of a triangular prism and we would compute the volume of 3 rectangular pyramids along with two triangular pyramids to get the total volumetric increment of a single triangular face of the control volume. (Figure 5.2)

We have to point out at this stage that throughout the work, we will assume that the nodal velocities are constant in magnitude and direction during each time step.

It is imperative to note that in our work, we intend to move the interior faces and keeping the boundary fixed in time, hence, due to the movement of the internal mesh at each time step, all the geometric parameters are reprocessed and recalculated according to the new prescribed positions of the nodes.

We recall that the main target in the moving mesh methods is to exactly satisfy the geometric conservation laws represented by the volumetric and space conservation laws. The volumetric conservation law needs more effort to ensure its satisfaction than the space conservation law due to the need to calculate exactly the volumetric increments resulting from the random movement of the interior nodes and hence interior faces. Hence, the target is to try to calculate the face volume increments as exact as possible so that the volumetric conservation equation is essentially satisfied exactly. We proposed an approach for calculating the face volumetric increments which constitutes of the following:

First, we focused on working on control volumes having quadrilateral faces so that we thought of dividing each face into four triangles simply by getting a rough face center. The center is obtained by summing the coordinates of the four face vertices that are already known and dividing by four afterwards. The obtained center will certainly be bounded by the four vertices and hence belongs to the face of the control volume. Once the center is obtained, four triangles can be constructed having the face center as a common vertex that belongs to the four triangles while the other two vertices represent respectively other vertices of the face.

Respectively from the formed four triangles on the original face configuration, we extrude four wedges to the four triangles formed on the face configuration at the new time step. As we already know the location of the four vertices of the face at the new time location, the mentioned four extruded wedges are simply formed by matching the vertices at the original location to those at the new location respectively including the center of the face. At this stage, each of the four wedges is divided into eight pyramids with triangular bases as an attempt to calculate the volume of the wedges and hence computing the volume of the volume increment relative to each face of the control volume as in Figure 5.4.

We notice here that we have missed an important point in our previous attempt to calculate the volume of the volumetric face increments that lies in our assumption about the center of the face. After applying the previous attempt, we faced inaccurate representation of the face volumetric increments because the face center that we averaged does not necessarily lie on the face. Random motion of the vertices of each face may form a skew face rather than forming a plane face and that makes it difficult to calculate a volume of a geometrical shape with skew faces. This fact leads us to our new attempt where we dispense the use of the face center on the original face and the new face at the new time level, and by that we don't need to ensure that a face center essentially lie on a skew face. Now, each quadrilateral face of the control volume is cut at the diagonal to form only two triangles. In a similar approach as the previous, the vertices of the two formed triangles are matched with their respective locations at the new time step to form two wedges (refer to Figure 5.5), which in turn are each divided into eight pyramids with triangular base and the volume increment is computed accordingly. We attempt to switch the diagonal that we used in the first stage of this approach and average the computed volume increment to get an approximation for the face volume increment. The results of all face increments of a single control volume were after all compared with the total change of volume of the whole control volume and the results were not as exact as needed to adopt the approach, that is because we couldn’t compute exactly the volume of geometry with faces that are not planar. At this point, we decided to turn our attention towards the constant movement of the mesh nodes so as to avoid random motion and its skew faces problem.

The discretized form of the geometric conservation equation is modified to the following form presented in Equation 5.2:

Equation 5.2

$$\frac{V^{(t)} - V^{(t-\Delta t)}}{\Delta t} - \sum_{iface=1}^{Nf} \left[\frac{dV}{\Delta t} \right]_{iface} = 0$$

Our task at this stage is to ensure that the left hand side of the equation approaches zero as much as possible.

We adopted a simple cubic geometry to elaborate on the mesh movement consequences and to check the VCL satisfaction, or in other form to check the satisfaction of Equation 5.2. The geometry consists of a cube (10 m in each dimension) divided into 3 control volumes in each dimension. The case consists of 27 elements, 54 internal faces and 54 boundary faces. The internal faces are allowed to move in a constant magnitude and direction by moving the nodes that constitute them by a factor of 0.1 m. in the x-direction and a factor of 0.2 m. in the y-direction at each time step. The test is done on three time steps, each of which represents 1 second in time. The initial configuration of the case is presented in Figure 5.6 in an isometric view, along with the respective configurations of the first, second and third time steps as top and isometric views respectively.

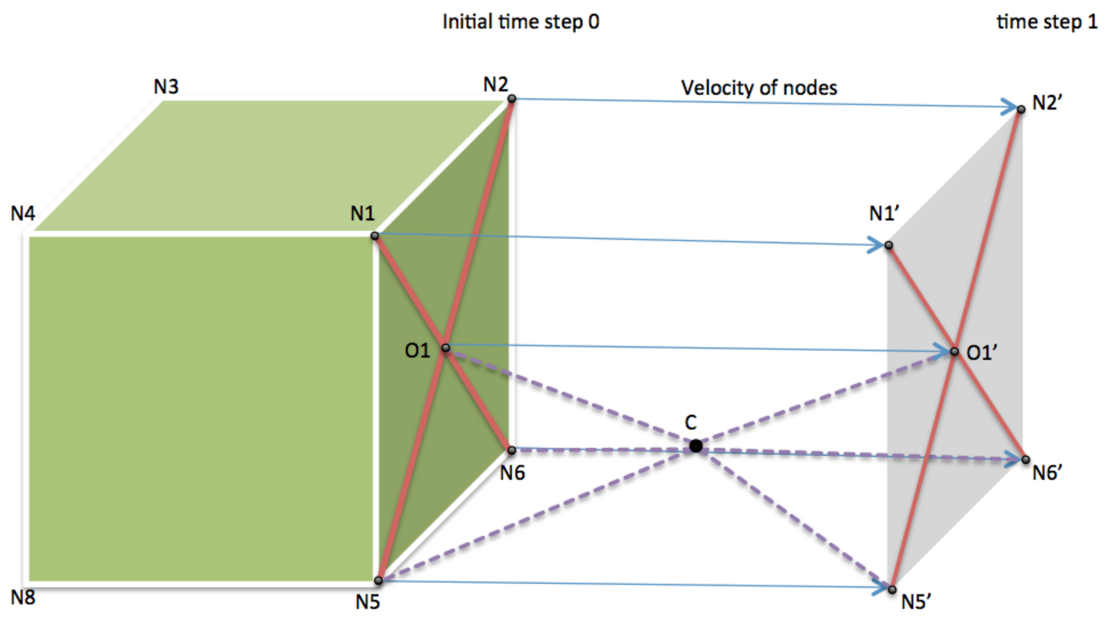


Figure 5.4: A sketch showing the technique of calculating volume of a generated volumetric face increment

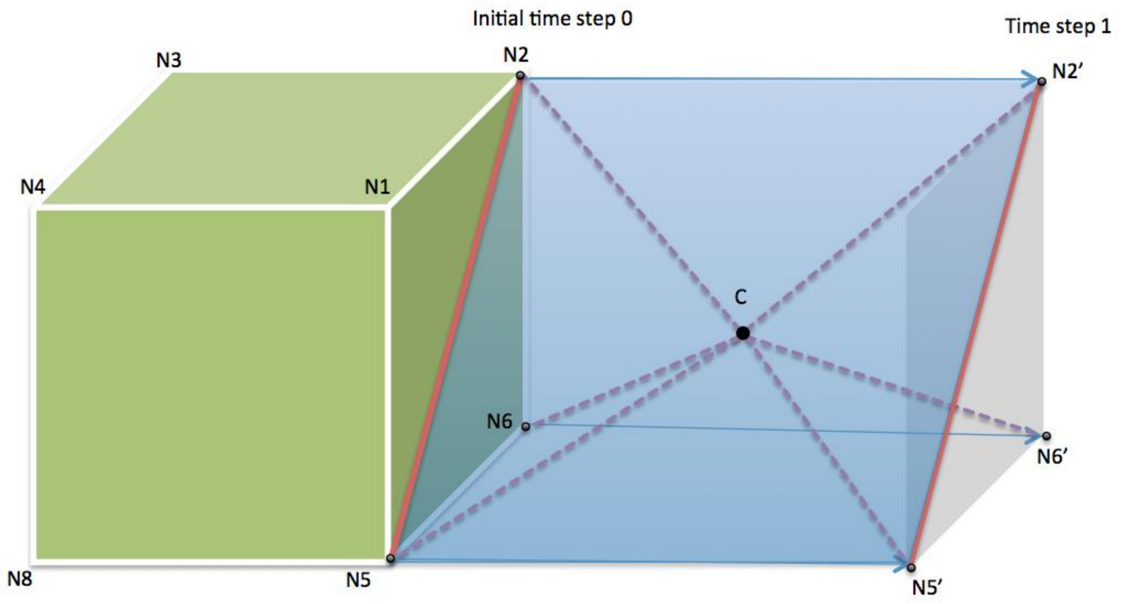


Figure 5.5: A sketch showing the case of splitting the generated volume of a face volumetric increment into two at the diagonal

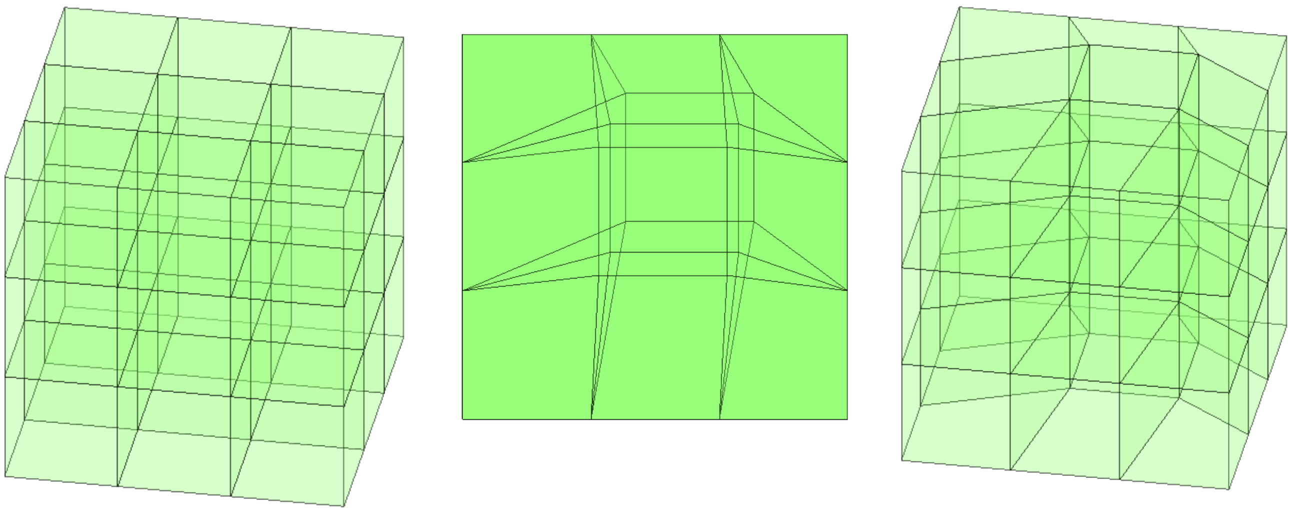


Figure 5.6: The initial and final moving mesh configuration. Left: to right: isometric vies of the initial configuration, top view of the final configuration, isometric vies of the final mesh configuration.

Equation 5.2 is checked on each of the 25 control volumes of the adopted geometry, where Equation 5.2 is coded and the results presented in Figure 5.7, which are practically the error of the equation, tend to the order of 10^{-13}. Such results are accepted as they approach zero and they signify that our approach of calculating the face volumetric increments is as accurate as the 10^{-13} degree.

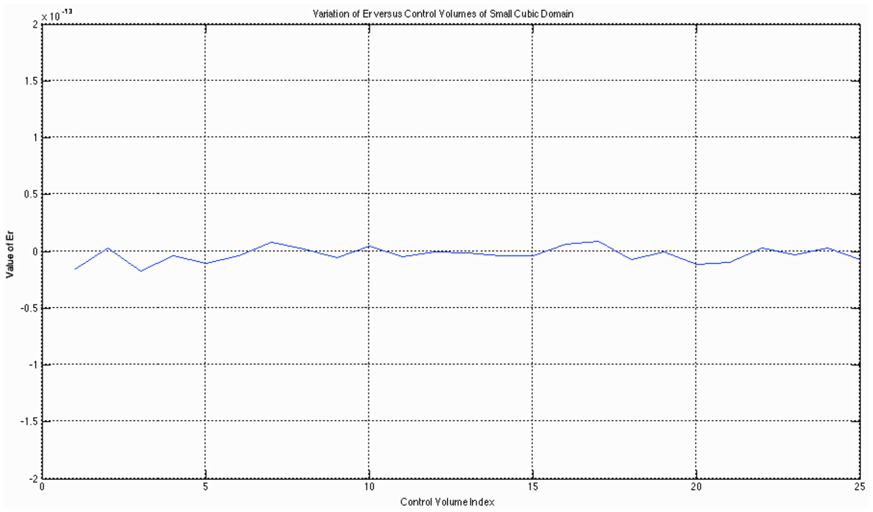


Figure 5.7: Results showing the left hand side of Equation 5.2

CHAPTER VI

HIGH ORDER MESH MOVEMENT SCHEME

This chapter is dedicated to discuss the implementation of a higher order scheme associated with the mesh movement that deals effectively with its resulting consequences. Several test cases are adopted and simulated for the aim of analyzing the effect of a first order then a second order scheme applied to discretize the mesh movement term; after which, the results are compared to the same test cases' static mesh results.

The mesh motion effect appears in our code as we update the mass flow rate ($\dot{m}$) after the mesh has moved to its new configuration directly at any second time step (the first time step will always constitute of the initial mesh configuration). At this point we have to note that in our study; only the internal faces of the mesh are allowed to move arbitrary in time keeping the boundaries fixed. This assumption essentially requires the satisfaction of the Volumetric Conservation Law i.e. the mesh nodes' movement must not affect the conservation laws. After the mesh faces move simply by moving the mesh nodes, we tend to recalculate all the geometrical parameters that are involved in the motion, and a new array storing the new information is generated. At this stage, the mass flow rate has to be updated as well and it is here where we devoted a scheme to handle the motion consequences.

Initially, the mass flow rate ($\dot{m}$) term is calculated in the u-FVM code according to Equation 6.1:

Equation 6.1

$$\dot{m} = \rho \times dot(\vec{A},\vec{V})$$

Where, ρ is the density, $\vec{A}$ is the face area and $\vec{V}$ is the fluid velocity.

The motion of the mesh introduces a modification to the mass flow rate described in Equation 6-1; this modification is in the form of a new velocity that is subtracted (exempting the discussion about the sign of the new velocity) from the fluid velocity. The new velocity is the velocity of the mesh is reshaped in our work and a new mass flow rate occurs according to Equation 6.2:

Equation 6.2

$$dot(\vec{A},\vec{V})_f = (\left.\frac{\Delta Vol}{\Delta t}\right|_f)$$

Where, ΔVol is the volumetric increment that is explicitly calculated for each mesh face, and Δt is the defined time step.

Following this technique of replacing the face velocities with face volumetric increments discretized by higher order schemes has not been spotted in the literature for Finite Volume framework. The technique treats the resulting face volumetric increments in an explicit way as we do not solve for the latter rather we calculate and store the associated increment at each time step.

Currently, the above consideration is a first order treatment to the mesh movement mass flow rate. The volumetric increment of each face is calculated according to the procedure explained in Chapter 5. Yet, our mission is to implement a higher order scheme to discretize the face volumetric increments resulting from the mesh faces' motion.

The introduced scheme for the mesh movement basically resembles the convective second order upwind scheme (SOU), in the sense that we store the volumetric increment of the specific face at an older time step and use it along with the currently calculated volumetric increment to produce the face mass flow rate modification as presented in Equation 6.3. The total mass flow rate now takes the mesh motion with its consequences into consideration and is presented in Equation 6-4.

Equation 6.3

$$\dot{m}\Big|_{mesh} = \rho \times \frac{\left(\tfrac{3}{2} \Delta Vol\Big|_{current} - \tfrac{1}{2} \Delta Vol\Big|_{old} \right)}{\Delta t}$$

Equation 6.4

$$\dot{m}\Big|_{total} = \dot{m} - \dot{m}\Big|_{mesh}$$

Using Taylor series expansion for the formulated scheme, the truncation error can be found to be as presented in Equation 6.5:

Equation 6.5

$$TE = -\frac{3}{8}\Delta x^2 \Delta t''' - \frac{1}{4}\Delta x^3 \Delta t^{iv} + \ldots$$

This clearly indicates that the formulated scheme is a second order accurate scheme. The derivation of the truncation error can be found in (Moukalled, Mangani et al. 2015).

A. Test Case 1: Transient Diffusion of a Scalar

The first test case adopted in our study is the case of solving and simulating a transient diffusion equation of a scalar (T). A brief description of the case geometry, the mesh information and the various conditions and parameters used in the solution procedure is tabulated in Table 6.1.

Table 6.1: Transient Diffusion of a scalar			
Geometry	Dimensions	1x1x0.1 m.	
Mesh	Number of Elements	400	
	Number of Faces	1640	
	Number of Nodes	882	
	Number of Interior Faces	760	
	Number of Patches	5	Patch 1: “left side”-type: wall Patch 2: “bottom side”- type: wall Patch 3: “right side”- type: wall Patch 4: “top side”- type: wall Patch 5: “front & Back sides”- type: empty
Fluid	Water	Molecular Weight = 18 Thermal expansion ratio = 10^{-4} Density= 1000 kg/m^3 Mode: Incompressible Viscous model: Laminar	
Conditions	Boundary Conditions	Scalar (T)	Patch 1: Specified value “2” Patch 2: Specified value “2” Patch 3: Specified value “2” Patch 4: Specified value “2” Patch 5: empty
	Initial Conditions	Constant “1” over all the domain	
	Under-Relaxation Factor	0.8	
	Number of Internal Iterations	50	
	Maximum Residuals Allowed	10^{-4}	
Parameters	Initial Time	1 second	
	Time Step	0.1 seconds	
	Final Time	10 seconds	
	Mesh Movement per time step	3%	Accumulative

According to the previously described boundary conditions, all the control volumes obtained a value of (2) after reaching the steady state at time=6 seconds. Figure 6.1 shows how the mesh is modified after moving for 10 seconds in time. The whole domain (i.e. all the control volumes) obtained a scalar value of (2), similar to the static mesh configuration. It can be noticed from the obtained set of results that the mesh movement did not affect the solution of the transient diffusion equation; this is because the mesh movement terms do not appear in the diffusion term. Ideally, when the mesh moves in time, the solution of any diffusion equation shall remain unaffected by the mesh motion and this was our case. Figure 6.2 presents the solution of the equation across section A-A as shown in Figure 6.1 and reveals a more clear comparison between the static mesh results and the results obtained from the applied first order and second order schemes. The three sets of results overlap at the same solution.

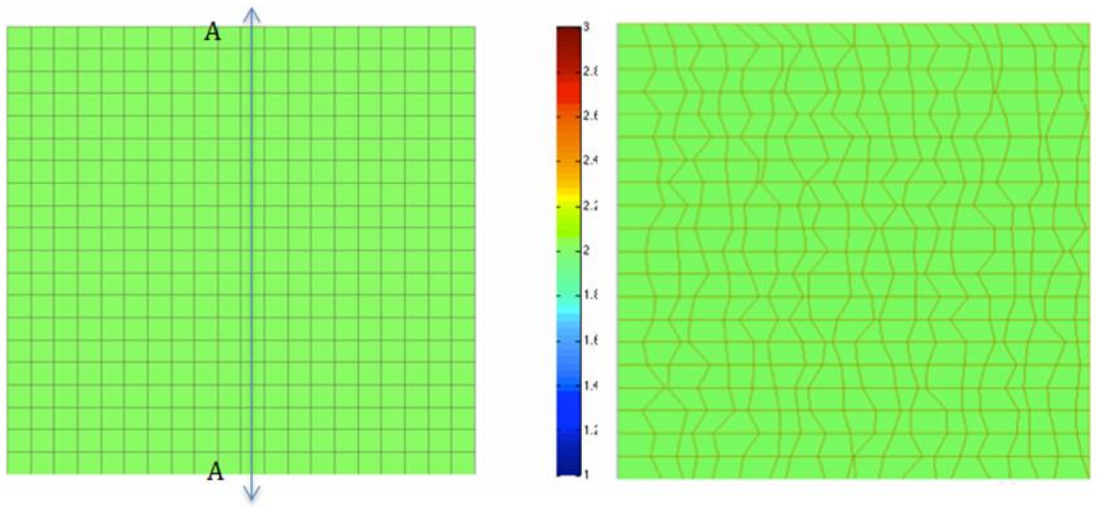


Figure 6.1 Initial and final mesh configurations of Transient Diffusion of a Scalar Test case

It is to be noted that using a first order scheme for the discretization of the mesh movement term or any higher order scheme will not show any effect on the solution.

This test case is conducted to make sure that no diffusive terms are generated or if so, are dealt with not to affect the final solution.

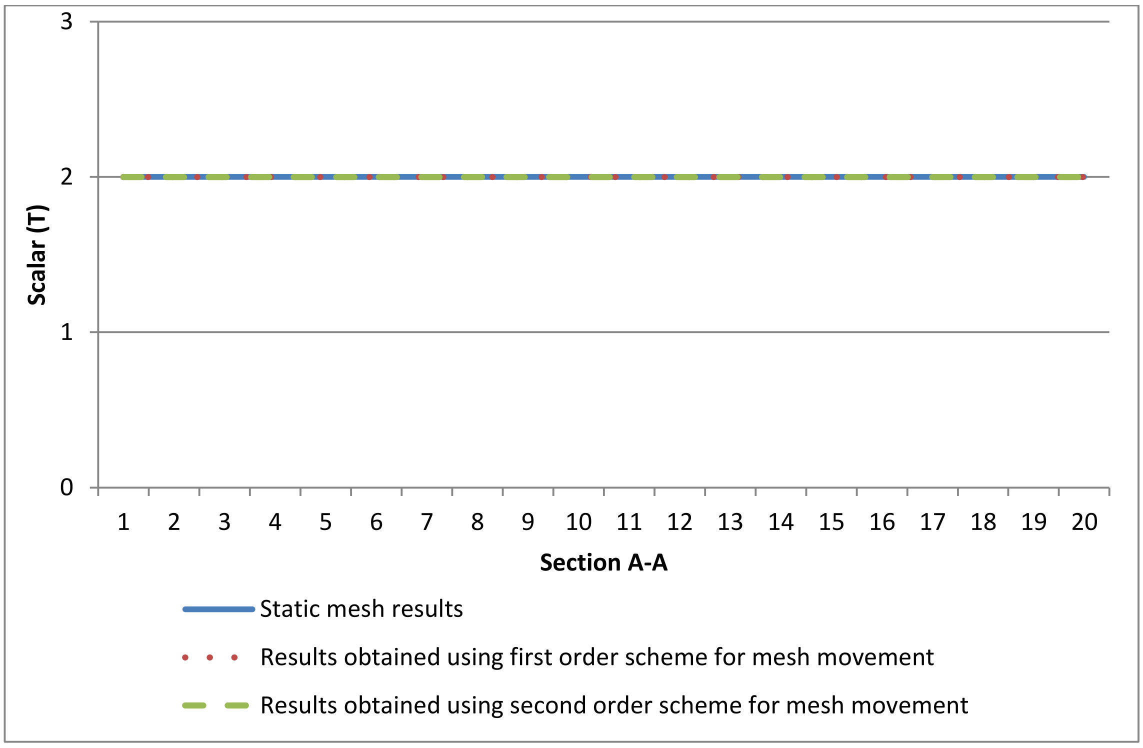


Figure 6.2 Chart presenting the results from static and moving mesh configurations

B. Test Case 2: Transient Advection of a Scalar with Inclined Velocity Field

The current test case is conducted to solve and simulate a transient advection equation of a scalar (T) over a geometry described in Table 6.2. We attempt in this case to simulate the same equation with the same parameters, initial and boundary conditions onto two mesh configurations. The first mesh configuration is composed of 400 control volumes whereas the second configuration is a denser mesh composed of 1600 control volumes. Each mesh configuration is simulated on a static mesh, then the mesh is allowed to move in a controlled random manner and results are obtained after implementing the first order and the second order schemes for mesh movement respectively. In both mesh configurations, we used two different schemes for the advection term: the Upwind scheme (first order) and the QUICK scheme (second order). The results are obtained and presented in the upcoming figures. Figure 6.3 presents the solution of scalar (T) on static mesh of the first configuration while Figure 6-4 presents the solution of scalar (T) on a moving mesh, both using an Upwind advection scheme. Figures 6.5 and 6.6 present the solution of the scalar (T) under the first mesh configuration case for a static mesh and a moving mesh respectively, both using a QUICK advection scheme. Figures 6.7, 6.8, 6.9 and 6.10 use the second mesh configuration (denser grid) to show the solution of scalar (T) on a static mesh (Figure 6.7) and a moving mesh (Figure 6.8) using the Upwind advection scheme. The QUICK scheme is used as the advection scheme on a static mesh (Figure 6.8) and a moving mesh (Figure 6.9) under the second mesh configuration.

Table 6.2: Transient Advection of a scalar – Inclined Velocity Vectors			
Geometry	Dimensions	1x1x0.1 m.	
	Number of Patches	5	Patch 1: “right and top sides”-type: outlet Patch 2: “left side”- type: inlet Patch 3: “bottom side”- type: inlet Patch 4: “front & Back sides”- type: empty
Fluid	Water	Molecular Weight = 18 Thermal expansion ratio = 10^{-7} Density= 1000 kg/m^3 Mode: Incompressible Viscous model: Laminar	
Conditions	Boundary Conditions	Scalar (T)	Patch 1: Outlet Patch 2: Specified value “2” Patch 3: Specified value “1” Patch 4: symmetry
	Initial Conditions	Constant “1” over all the domain	
	Under-Relaxation Factor	0.8	
	Number of Internal Iterations	50	
	Maximum Residuals Allowed	10^{-7}	
Parameters	Initial Time	1 second	
	Time Step	0.1 seconds	
	Final Time	10 seconds	
	Mesh Movement per time step	3%	Accumulative

Mesh Configurations		
	First Mesh Configuration	Second Mesh Configuration
Number of Elements	400	1600
Number of Faces	1640	6480
Number of Nodes	882	3362
Number of Interior Faces	760	3120

While analyzing the first mesh configuration represented by the set of Figures 6.3, 6.4, 6.5 and 6.6, we notice that the results obtained from the moved mesh cases follow the results of the static mesh and follow as well the order of the scheme employed for the convective term. Yet, we can notice the captured distraction in the plotted scalar field in Figure 6.6 due to the plotting function that assigns a respective color to each control volume. Because the control volumes have an irregular shape in the moved mesh configurations, these distractions are obviously realized. This is the reason why we introduced a denser mesh (second mesh configuration) in Figures 6.7, 6.8, 6.9 and 6.10 respectively as an attempt to show how the distractions in the plotted scalar field are reduced. Using an even denser mesh will decrease more the smearing of the plotted field.

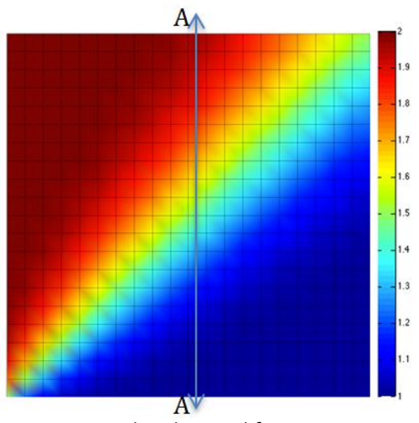


Figure 6.3 Results obtained from a static mesh using an Upwind advection scheme

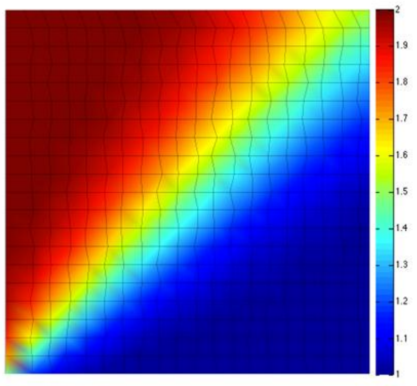


Figure 6.4 Results obtained from a moving mesh using an Upwind advection scheme

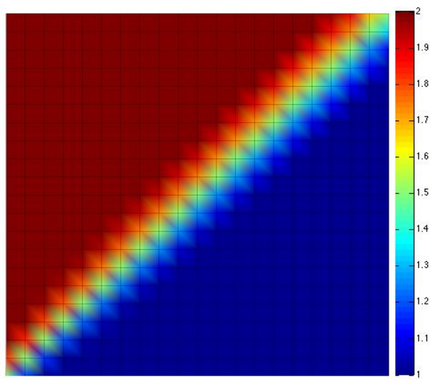


Figure 6.5 Results obtained from a static mesh using a QUICK advection scheme

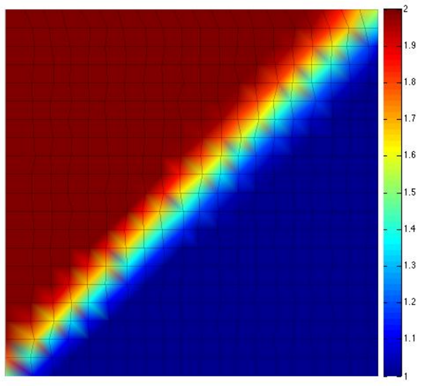


Figure 6.6 Results obtained from a moving mesh using a QUICK advection scheme

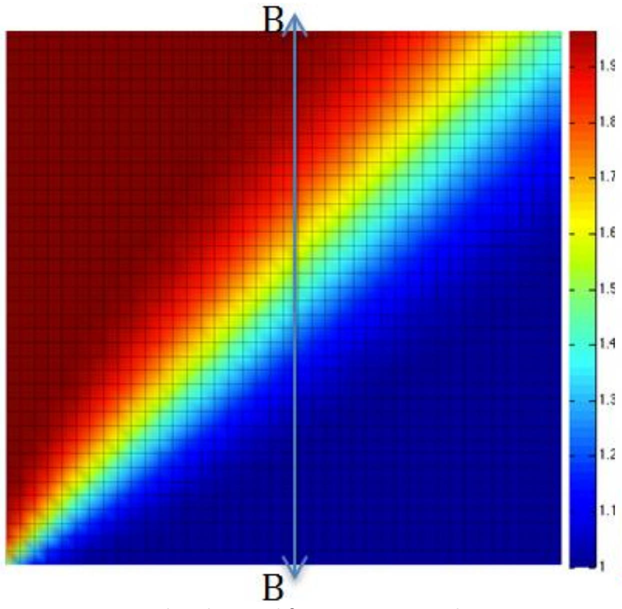


Figure 6.7 Results obtained from a static mesh using an Upwind advection scheme

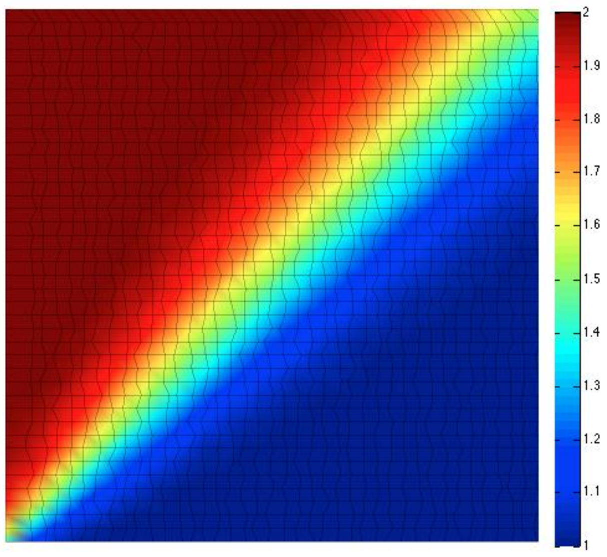


Figure 6.8 Results obtained from a moving mesh using an Upwing advection scheme

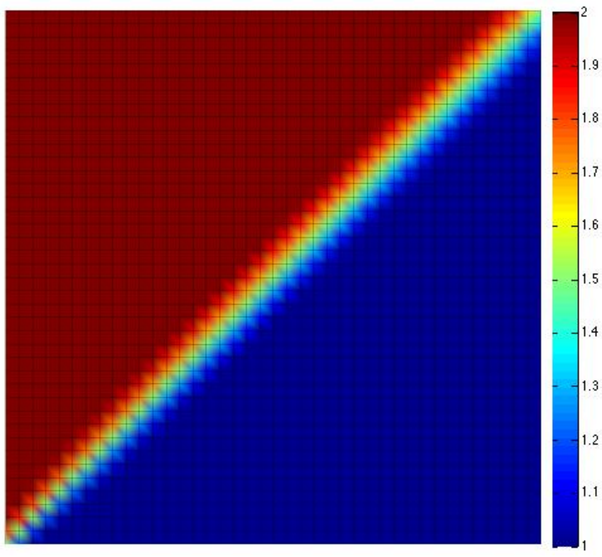


Figure 6.9 Results obtained from a static mesh using a QUICK advection scheme

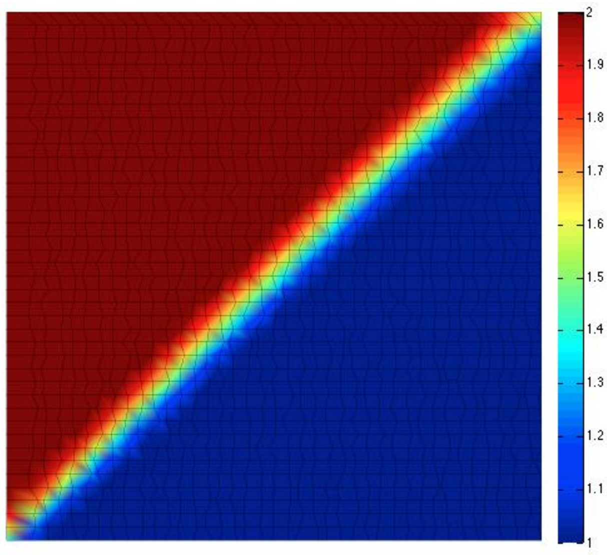


Figure 6.10 Results obtained from a moving mesh using a QUICK advection scheme

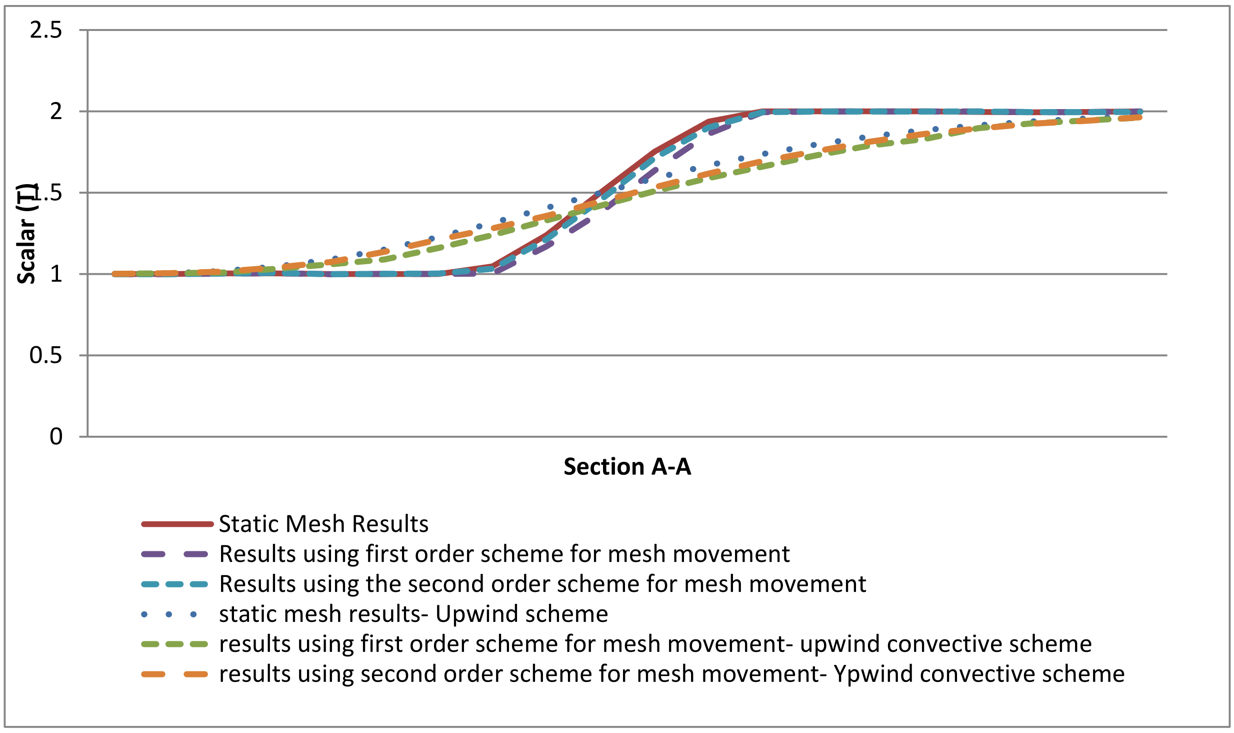


Figure 6.11 Chart showing the results of the first mesh configuration using an Upwind and a QUICK advection scheme on static and moving grids

The chart in Figure 6.11 presents the two sets of results assigned to the first mesh configuration. The first set of results in the above chart is obtained by using the upwind scheme to discretize the advection term of the transient advection equation simulated on a static mesh; then on a moving mesh employing the first order scheme for mesh movement and after that, on a moving mesh employing the second order mesh movement scheme. The results are plotted over the control volumes cut at the middle of the domain at section A-A as shown in Figure 6.3. The results of the second order mesh movement scheme are closer to the results of the static mesh than those of the first order mesh movement scheme. As decided, the upwind scheme is replaced by the QUICK scheme to deal with the discretization of the advection term of the equation. The second set of results is obtained also for a static mesh, and for a moved mesh employing the first order mesh movement scheme as well as a moved mesh employing the second order mesh movement scheme yet employing the QUICK scheme for the advection term. Similarly as before, the results obtained from the second order mesh movement scheme are closer to the static mesh results than those of the first order mesh movement scheme. Moreover, we notice that the set of results where the QUICK scheme is employed to discretize the advection term is more compact than the set of results with the Upwind scheme adopted. A more compact set of results is translated as a positive fact because the results would be closer to each other.

The chart in Figure 6.12, presents the two sets of results, along the middle of the domain at the section B-B shown in Figure 6.7, obtained when conducting the second denser mesh configuration. Basically, the two sets of results resemble those of the first mesh configuration; yet, the denser mesh affected the results by bringing each set of results closer to each other. While comparing the charts of Figure 6.11 and that of Figure 6.12, it is noticed how the two set of results (the first with the upwind advection scheme and the second with the QUICK advection scheme) on the second mesh configuration are closer to the static mesh results than on the first mesh configuration. This elaborates that using a denser mesh will improve the results obtained from the moved mesh and allow them to overlap with the results when the mesh is static.

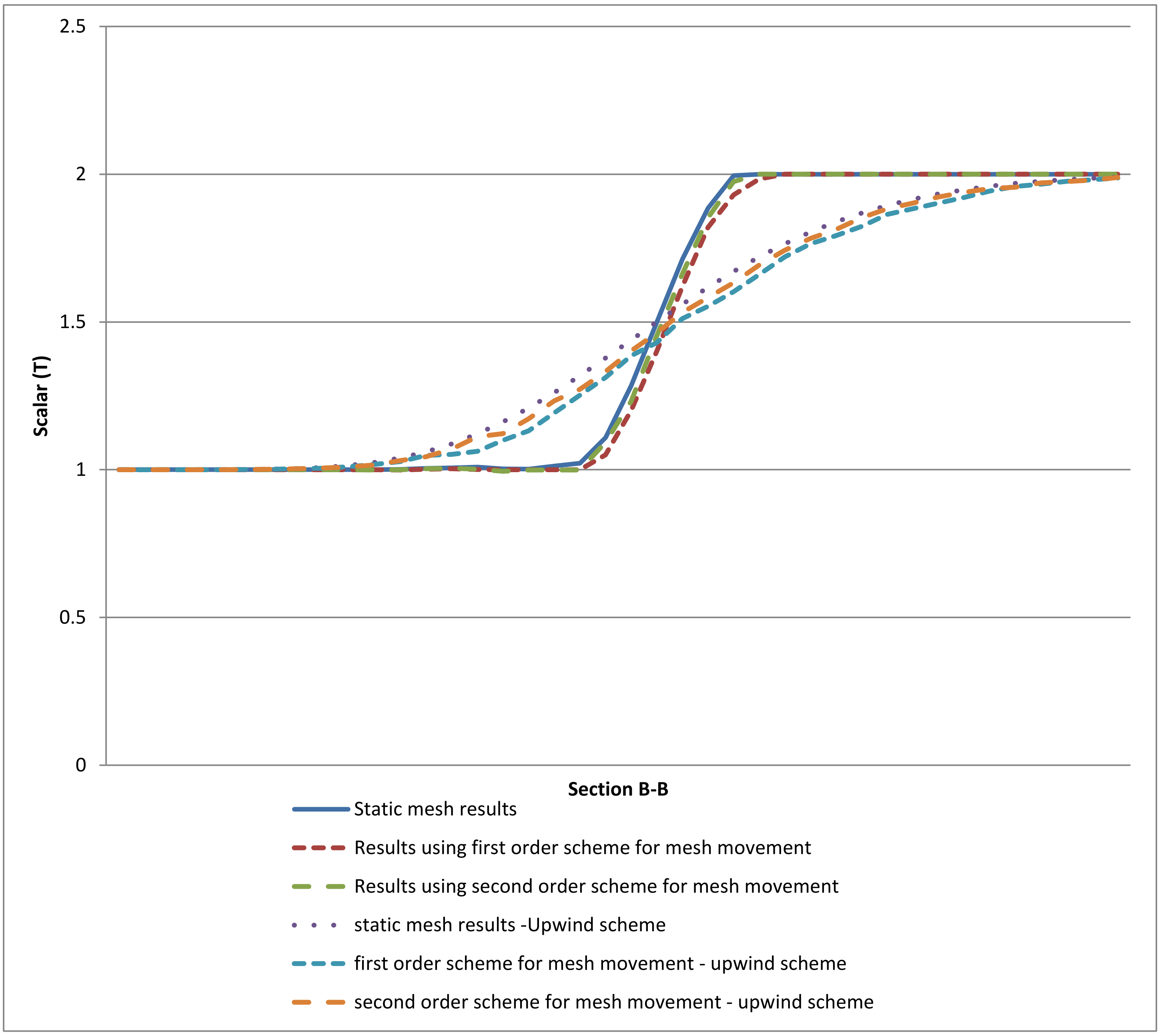


Figure 6.12 Chart showing the results of the second mesh configuration using an Upwind and a QUICK advection scheme on static and moving grids

C. Test Case 3: Transient Advection of a Scalar with Rotational Velocity Field

The following test case solves the same equation as the previous case which is the transient advection of a scalar. It is performed on the geometry described in the Table 6.3 along with the associated parameters and required conditions. Here, the velocity field is not an inclined field but rather is a rotational field distributed across the domain (refer to Figure 6.15). Figure 6.13 presents the initial mesh configuration that is the same mesh used in a static simulation. Figure 6.14 presents the final mesh configuration after moving in time for 10 seconds as stated in Table 6.3. The results are captured along section A-A shown in Figure 6.13 and plotted in the chart of Figure 6.16.
The orientation of the velocity vectors did not affect the work of the proposed higher order scheme for handling the mesh motion term. Figure 6.16 shows how the three set of results follow the same profile trend and shows as well that the results generated from the moving mesh case with the higher order scheme applied therein tend to overlap with the results obtained when simulating the same equation on a static mesh more than those results obtained from the moving mesh case with the first order scheme for mesh movement applied therein. This chart ensures that the formulated scheme is resolving the effects of the mesh motion properly enough to provide results close enough to the results of a static mesh simulation.

Table 6.3: Transient Advection of a scalar – Rotational Velocity Vectors			
Geometry	Dimensions	2x1x0.1 m.	
Mesh	Number of Elements	800	
	Number of Faces	3260	
	Number of Nodes	1722	
	Number of Interior Faces	1540	
		5	Patch 1: “left section of first half of bottom face”-type: inlet Patch 2: “right section of first half of bottom face”- type: inlet Patch 3: “second half of bottom face”-type: outlet Patch 4: “front & Back sides”- type: empty
Number of Patches	Water	Molecular Weight = 18 Thermal expansion ratio = 10^{-4} Density= 1000 kg/m^3 Mode: Incompressible Viscous model: Laminar	
Conditions	Boundary Conditions	Scalar (T)	Patch 1: Specified value “2” Patch 2: Specified value “1” Patch 3: Outlet Patch 4: Zero Flux
	Initial Conditions	Constant “0.5” over all the domain	
	Under-Relaxation Factor	0.8	
	Number of Internal Iterations	50	
	Maximum Residuals Allowed	10^{-4}	
Parameters	Initial Time	1 second	
	Time Step	0.1 seconds	
	Final Time	10 seconds	
	Mesh Movement per time step	3%	Accumulative
	Velocity vector formula	[2*y.*(1-x.^2);-2*x.*(1-y.^2);0]	

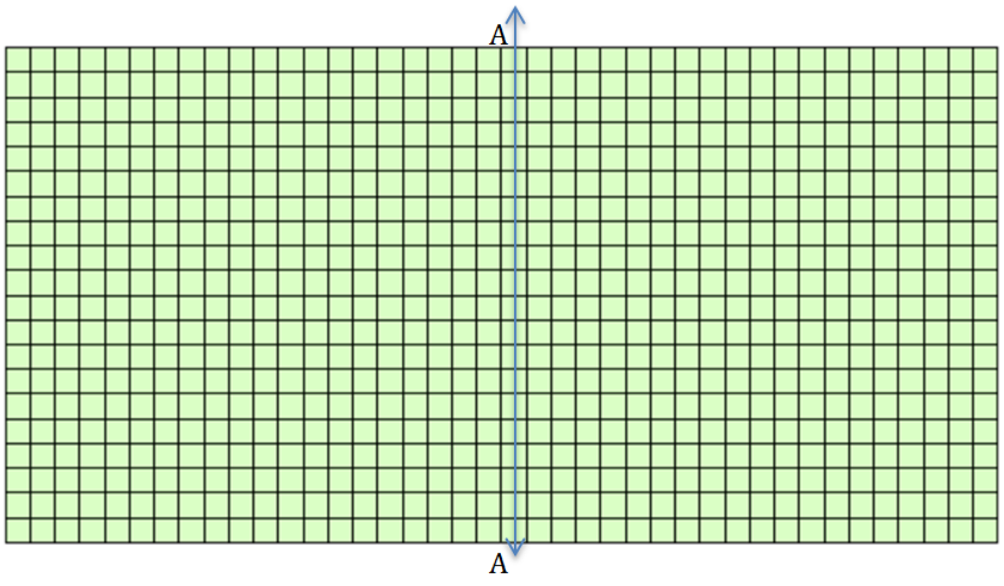


Figure 6.13 Static mesh configuration of the adopted domain.

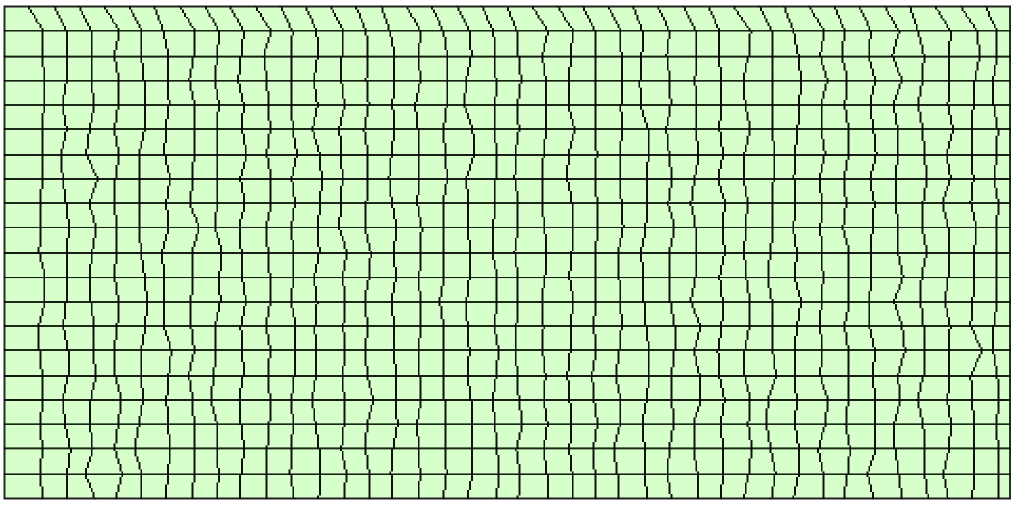


Figure 6.14 Moved mesh configuration of the adopted domain.

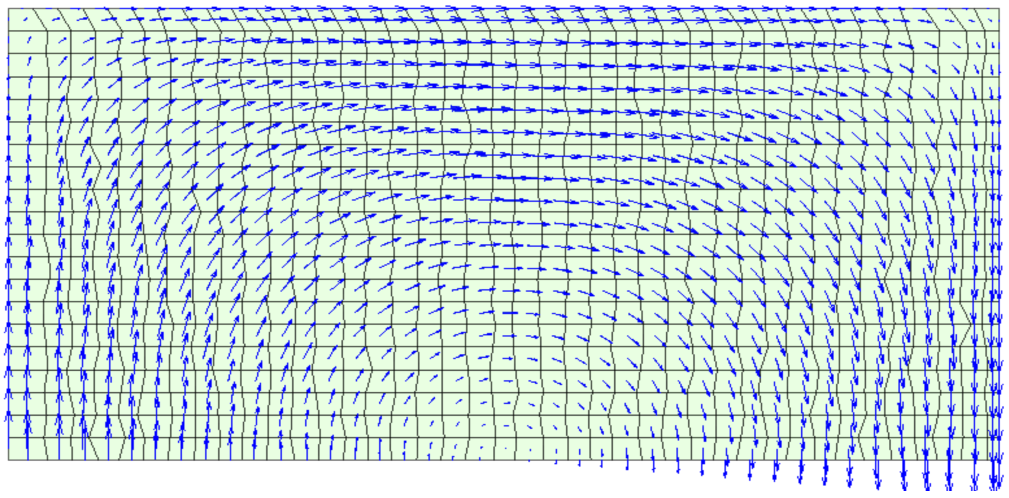


Figure 6.15 Velocity field across the adopted domain.

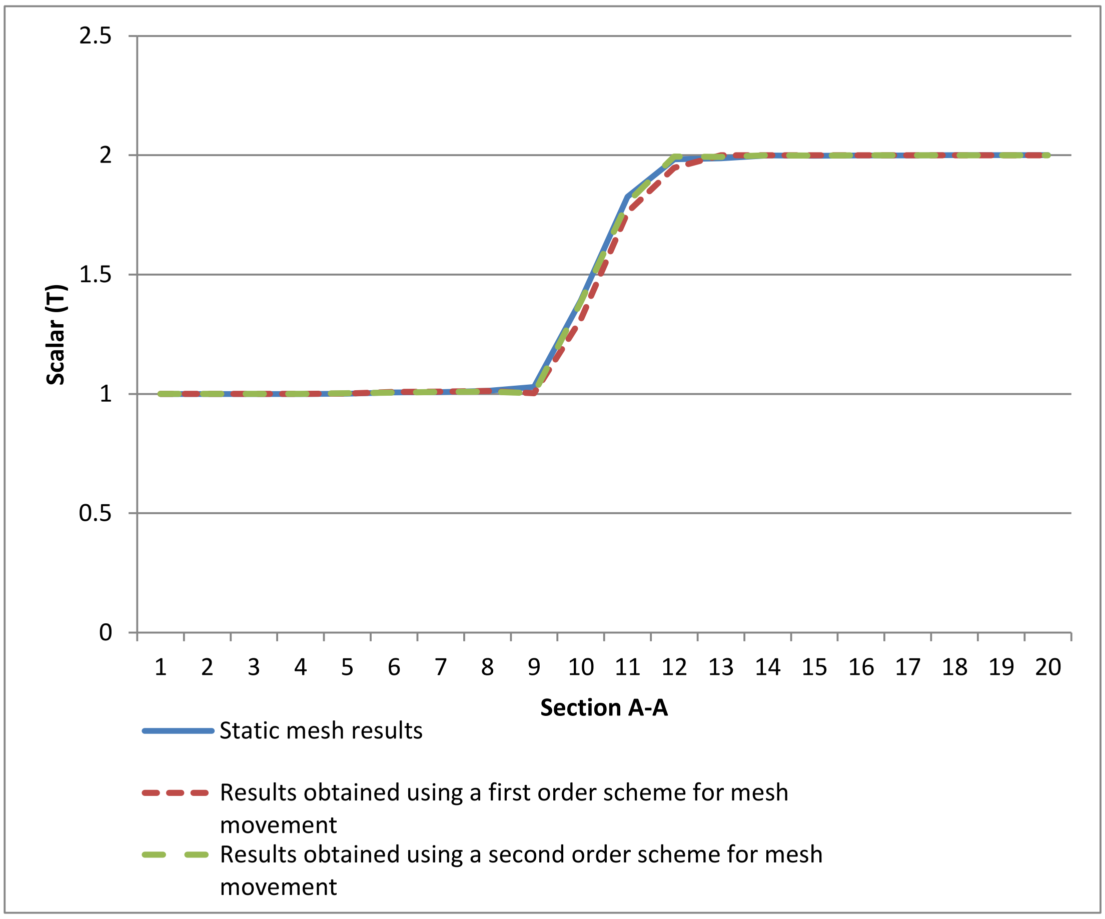


Figure 6.16 Chart presenting the results of Scalar (T) along section A-A for a static mesh as well as for a moving mesh using a first order and a second order scheme for mesh movement

D. Test Case 4: Navier Stokes' Equations in a Flow Problem

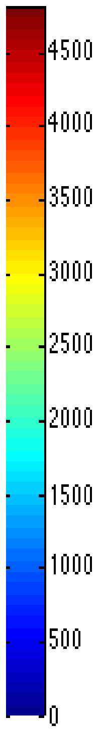
At this stage, we decided to solve the Navier Stokes' equations in a pressure-based solver as an attempt to test the validity of the mesh movement in such a situation. The adopted geometry that is described in Table 6.4 is allowed to move in an arbitrary manner each time step and the results of pressure were recorded and plotted over the domain. The patches of the mesh are presented in Table 6.4 and the outcome is three sets of generated results; the first is when the mesh was remained static within time, the second is when the mesh was allowed to move arbitrary with the application of the first order mesh movement scheme and the third is when the mesh is moved arbitrary with the second order accurate mesh movement scheme applied within.

The pressure filed is plotted in Figure 6.17 for the static mesh and in Figure 6.18 for the moved mesh.

The pressure field along section B-B shown in Figure 6.17 is plotted in chart of Figure 6.19. The same trend is respected for the three set of results as shown in Figure 6.19 while realizing that the generated results from the second order scheme are better than the results of the first order scheme for mesh motion in the sense that the first approach and tend to overlap clearly with the results obtained from the static mesh. Such an outcome along with the previous results obtained from the several presented test cases are sufficient to prove the validity of the higher order scheme adopted to discretize the moving mesh term.

Table 6.4: Flow Problem – Solving Navier-Stokes' Equations – Smith Hutton Case

Geometry	Dimensions	2x1x0.1 m.	
Mesh	Number of Elements	800	
	Number of Faces	3260	
	Number of Nodes	1722	
	Number of Interior Faces	1540	
	Number of Patches	5	Patch 1: "left section of first half of bottom face"-type: inlet Patch 2: "right section of first half of bottom face"- type: inlet Patch 3: "second half of bottom face"- type: outlet Patch 4: "front & Back sides"- type: empty
Fluid	Water	Molecular Weight = 18 Thermal expansion ratio = 10^{-4} Viscosity = 0.0008 Pa.s Density= 1000 kg/m^3 Mode: Incompressible Viscous model: Laminar	
Conditions	Boundary Conditions	Velocity eqn.	Patch 1: Specified value "[0,2,0]" Patch 2: Specified value "[0,1,0]" Patch 3: Outlet "[0,0,0]" Patch 4: Zero Flux
		Pressure eqn.	Patch 1: Inlet Patch 2: Inlet Patch 3: Specified Value "0" Patch 4: No Slip
	Initial Conditions	Velocity eqn.	Constant "[0.5,0.5,0.5]" over all the domain
		Pressure eqn.	Constant "0" over all the domain
	Under-Relaxation Factor	0.8	
	Number of Internal Iterations	50	
	Maximum Residuals Allowed	10^{-4}	
Parameters	Initial Time	1 second	
	Time Step	0.1 seconds	
	Final Time	75 seconds	
	Mesh Movement per time step	3%	Accumulative



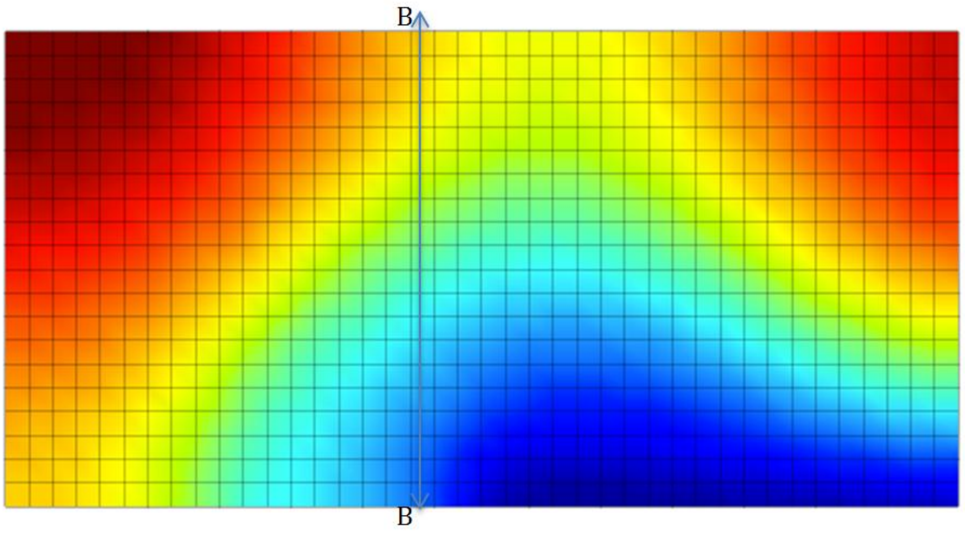


Figure 6.17 Pressure field plot on a static mesh configuration

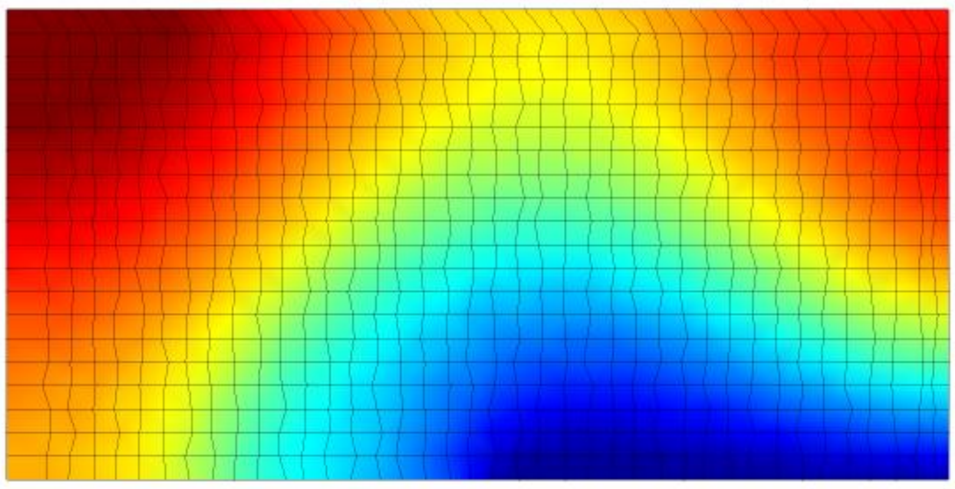


Figure 6.18 Pressure field plot on a moved mesh configuration

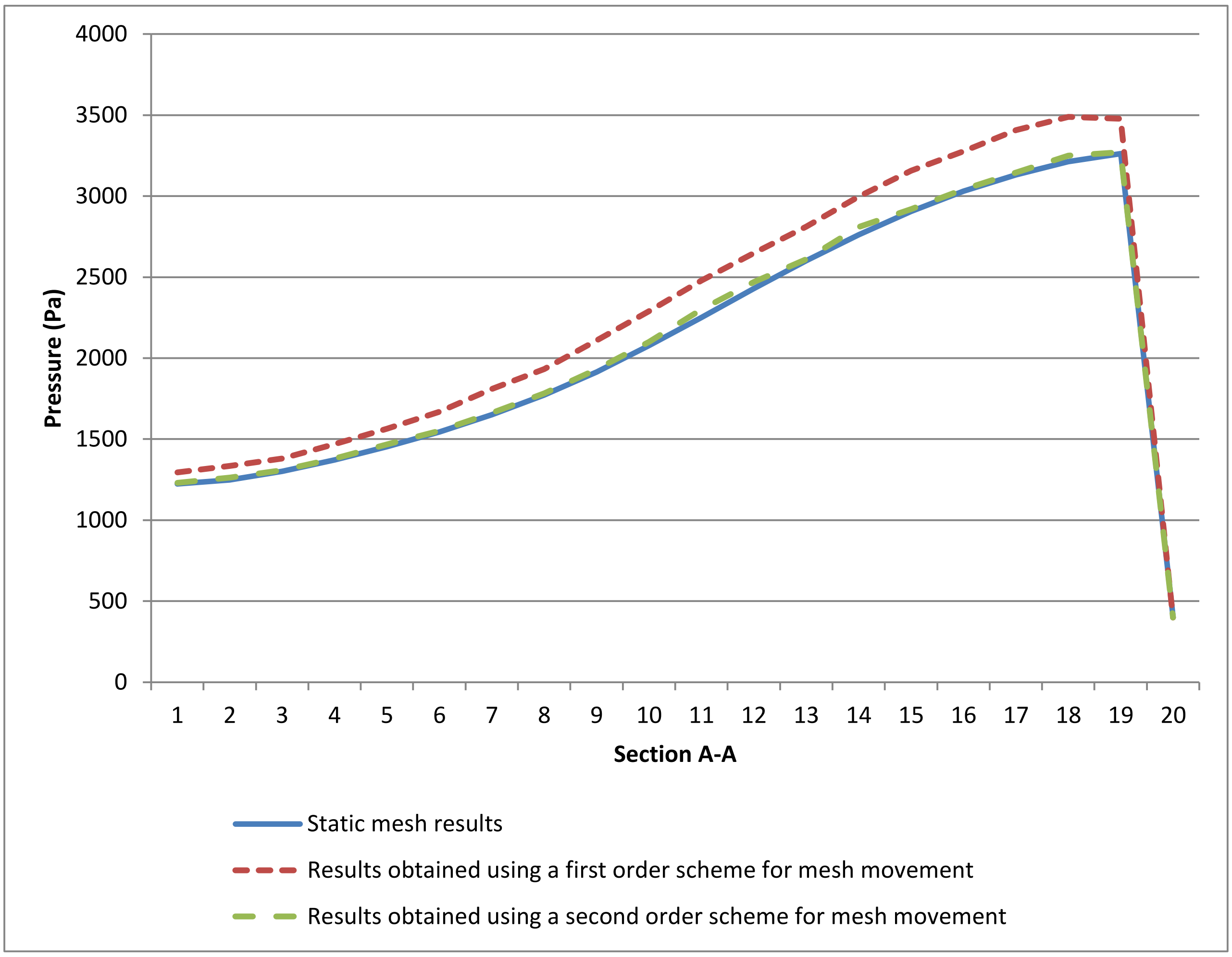


Figure 6.19 Chart presenting the pressure solution along section A-A for static mesh configuration and moved mesh configurations using a first order and a second order scheme for mesh movement

The concept of moving into higher order schemes for the discretization of the mesh motion term i.e. moving into third and even higher order schemes is promising, as long as the scheme is constructed in a way that satisfies the geometrical conservation laws.

CHAPTER VII

CONCLUSION

The main objective of this work was to develop the capabilities of the u-FVM solver to cover problems with moving grids. The contribution of applying a special technique of replacing the mesh movement term i.e. the generated mesh velocity in the governing equations by the volumetric face increments and explicitly calculating this term has shown its validity in Chapter 5. A high order scheme for the treatment of the aforementioned mesh movement term was addressed, discussed and implemented in Chapter 6, where the results have proved its improvement to the u-FVM results. The contribution can serve the field of fluid-structure interaction and free surface flow problems, as it is widely involved in the field of fluid-structure interaction problems. Further and future work can be proposed to apply higher order schemes for the mesh movement term discretization and adopt real life cases for simulation.

BIBLIOGRAPHY

Amsden A, Ruppel H, Hirt C. 1980. SALE: A simplified ALE computer program for fluid flow at all speeds. US Department of Commerce, National Technical Information Service.

Anderson D, McFadden GB, Wheeler A. 1998. Diffuse-interface methods in fluid mechanics. Annu Rev Fluid Mech 30(1):139-65.

Baum JD, Luo H, Loehner R. 1998. The numerical simulation of strongly unsteady flows with hundreds of moving bodies. AIAA Pap 788.

Belytschko T and Kennedy JM. 1978. Computer models for subassembly simulation. Nucl Eng Des 49(1):17-38.

Belytschko T, Kennedy JM, Schoeberle D. 1980. Quasi-eulerian finite element formulation for fluid-structure interaction. Journal of Pressure Vessel Technology 102(1):62-9.

Bendiksen OO. 2011. Review of unsteady transonic aerodynamics: Theory and applications. Prog Aerospace Sci 47(2):135-67.

Bennett RM and Edwards JW. 1998. An overview of recent developments in computational aeroelasticity. AIAA Paper 98:2421.

Beyer RP. 1992. A computational model of the cochlea using the immersed boundary method. Journal of Computational Physics 98(1):145-62.

Clarke DK, Hassan H, Salas M. 1986. Euler calculations for multielement airfoils using cartesian grids. Aiaa j 24(3):353-8.

De Zeeuw D and Powell KG. 1991. An adaptively refined cartesian mesh solver for the euler equations. AIAA Paper (91-1542).

Demirdžić I and Perić M. 1990. Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. Int J Numer Methods Fluids 10(7):771-90.

Demirdžić I and Perić M. 1988. Space conservation law in finite volume calculations of fluid flow. Int J Numer Methods Fluids 8(9):1037-50.

Donea J, Huerta A, Ponthot J-, Rodríguez-Ferran A. 2004. Arbitrary Lagrangian–Eulerian methods. Encyclopedia of Computational Mechanics 1(14).

Donéa J, Fasoli-Stella P, Giuliani S. 1977. Lagrangian and eulerian finite element techniques for transient fluid-structure interaction problems. In: Structural mechanics in reactor technology.

Durst F, Pereira J, Scheuerer G. 1986. Calculations and experimental investigations of the laminar unsteady flow in a pipe expansion. Finite Approximations in Fluid Mechanics: 43-55.

Franck R and Lazarus R. 1964. Mixed eulerian-lagrangian method. Methods in Computational Physics 3:47-67.

Ghias R, Mittal R, Lund TS. 2004. A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries. AIAA Paper 80:2004.

Gosman A and Johns R. 1978. Development of a predictive tool for in-cylinder gas motion in engines. Development of a Predictive Tool for in-Cylinder Gas Motion in Engines.

Guilmineau E and Queutey P. 2002. A numerical simulation of vortex shedding from an oscillating circular cylinder. J Fluids Struct 16(6):773-94.

Hindman RG. 1982. Generalized coordinate forms of governing fluid equations and associated geometrically induced errors. Aiaa j 20(10):1359-67.

Hirt C, Amsden AA, Cook J. 1974. An arbitrary lagrangian-eulerian computing method for all flow speeds. Journal of Computational Physics 14(3):227-53.

Hirt CW and Nichols BD. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of Computational Physics 39(1):201-25.

Hughes TJ, Liu WK, Zimmermann TK. 1981. Lagrangian-eulerian finite element formulation for incompressible viscous flows. Comput Methods Appl Mech Eng 29(3):329-49.

Jasak H, Jemcov A, Tukovic Z. 2013. OpenFOAM: A C library for complex physics simulations. .

Krause EGON. 1979. The computation of three dimensional viscous flows. In its computational fluid dyn. 26 p.

Majumdar S, Iaccarino G, Durbin P. 2001. RANS solvers with adaptive structured boundary non-conforming grids. Annual Research Briefs, Center for Turbulence Research, Stanford University :353-466.

McIntyre SM. 2011. An adaptive immersed boundary method for cfd simulation of multiphase flows with moving internal bodies. The Pennsylvania State University.

Miller LA and Peskin CS. 2005. A computational fluid dynamics of 'clap and fling' in the smallest insects. J Exp Biol 208(Pt 2):195-212.

Miller LA and Peskin CS. 2004. When vortices stick: An aerodynamic transition in tiny insect flight. J Exp Biol 207(Pt 17):3073-88.

Mittal R and Iaccarino G. 2005. Immersed boundary methods. Annu Rev Fluid Mech 37:239-61.

Mohd-Yusof J. 1997. Combined immersed-boundary/B-spline methods for simulations of ow in complex geometries. Annual Research Briefs.NASA Ames Research Center= Stanford University Center of Turbulence Research: Stanford :317-27.

Moukalled F, Mangani L, Darwish M. 2015. The finite volume method in computational fluid dynamics an advanced introduction with OpenFOAM® and matlab. 1st ed. Springer International Publishing.

Naderi A, Darbandi M, Taeibi-Rahni M. 2010. Developing a unified FVE-ALE approach to solve unsteady fluid flow with moving boundaries. Int J Numer Methods Fluids 63(1):40-68.

Noh W. 1963. A time-dependent two-space dimensional coupled eulerian-lagrangian code. Methods in Computational Physics 3:117-79.

Osler TJ. 1972. The integral analog of the leibniz rule. Mathematics of Computation :903-15.

Peskin CS. 2002. The immersed boundary method. Acta Numerica 11:479-517.

Peskin CS. 1982. The fluid dynamics of heart valves: Experimental, theoretical, and computational methods. Annu Rev Fluid Mech 14(1):235-59.

Peskin CS,. 1972. Flow patterns around heart valves : A digital computer method for solving the equations of motion. .

Pfeffer WF. 1991. The gauss-green theorem. Advances in Mathematics 87(1):93-147.

Ralph M and Pedley T. 1989. Viscous and inviscid flows in a channel with a moving indentation. J Fluid Mech 209:543-66.

Ramamurti R and Sandberg W. 2001. Simulation of flow about flapping airfoils using finite element incompressible flow solver. Aiaa 39(2):253-60.

Scardovelli R and Zaleski S. 1999. Direct numerical simulation of free-surface and interfacial flow. Annu Rev Fluid Mech 31(1):567-603.

Schuster DM, Liu DD, Huttsell LJ. 2003. Computational aeroelasticity: Success, progress, challenge. J Aircr 40(5):843-56.

Shahrour I. and Benchekh B. 1992. Analysis of the soil structure interaction under monotonic and cyclic loadings. Proceedings of the first european conference on numerical methods in engineering, bruxelles, elsevier, amsterdam.

Taylor CA, Hughes TJ, Zarins CK. 1998. Finite element modeling of blood flow in arteries. Comput Methods Appl Mech Eng 158(1):155-96.

Tezduyar TE. 2001. Finite element methods for flow problems with moving boundaries and interfaces. Archives of Computational Methods in Engineering 8(2):83-130.

Thomas P and Lombard C. 1979. Geometric conservation law and its application to flow computations on moving grids. Aiaa j 17(10):1030-7.

Thomas PD and Lombard CK. 1978. The geometric conservation law-A link between finite-difference and finite-volume methods of flow computation on moving grids. 11th fluid and plasma dynamics conference.

Trepanier J, Reggio M, Zhang H, Camarero R. 1991. A finite-volume method for the euler equations on arbitrary lagrangian-eulerian grids. Comput Fluids 20(4):399-409.

Trulio JG. 1966. Theory and structure of the AFTON codes. DTIC Document.

Trulio JG and Trigger KR. 1961. Numerical solution of the one-dimensional hydrodynamic equations in an arbitrary time-dependent coordinate system. University of California Lawrence Radiation Laboratory Report UCLR-6522 .

Udaykumar H, Shyy W, Rao M. 1996. Elafint: A mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries. Int J Numer Methods Fluids 22(8):691-712.

Unverdi SO and Tryggvason G. 1992. A front-tracking method for viscous, incompressible, multi-fluid flows. Journal of Computational Physics 100(1):25-37.

Verzicco R, Mohd-Yusof J, Orlandi P, Haworth D. 1998. LES in complex geometries using boundary body forces. Center for Turbulence Research Proceedings of the Summer Program, NASA Ames Stanford University :171-86.

Vinokur M. 1989. An analysis of finite-difference and finite-volume formulations of conservation laws. Journal of Computational Physics 81(1):1-52.

Viviand Henri and Ghazzi Walid. 1976. Numerical solution of the compressible navier-stokes equations at high reynolds numbers with applications to the blunt body problem. Proceedings of the fifth international conference on numerical methods in fluid dynamics june 28–July 2, 1976 twente university, enschedeSpringer. 434 p.

Ye T, Mittal R, Udaykumar H, Shyy W. 1999. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. Journal of Computational Physics 156(2):209-40.

Yurkovich R. 2003. Status of unsteady aerodynamic prediction for flutter of high-performance aircraft. J 40(5):832-42.

Zhang H, Reggio M, Trepanier J, Camarero R. 1993. Discrete form of the GCL for moving meshes and its implementation in CFD schemes. Comput Fluids 22(1):9-23.

Zwart P, Raithby G, Raw M. 1999. The integrated space-time finite volume method and its application to moving boundary problems. Journal of Computational Physics 154(2):497-519.