

Fall 11-2014

## A Continuous/Discontinuous FE Method for the 3D Incompressible Flow Equations

Nikolaos Kyriazis  
*Embry-Riddle Aeronautical University*

Follow this and additional works at: <https://commons.erau.edu/edt>



Part of the [Aerospace Engineering Commons](#)

---

### Scholarly Commons Citation

Kyriazis, Nikolaos, "A Continuous/Discontinuous FE Method for the 3D Incompressible Flow Equations" (2014). *Doctoral Dissertations and Master's Theses*. 273.  
<https://commons.erau.edu/edt/273>

This Thesis - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in Doctoral Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact [commons@erau.edu](mailto:commons@erau.edu).

**A CONTINUOUS/DISCONTINUOUS FE METHOD FOR THE 3D  
INCOMPRESSIBLE FLOW EQUATIONS**

**Nikolaos Kyriazis**

A Thesis Submitted to the  
Graduate Studies Office  
in Partial Fulfillment of the Requirements for the Degree of  
Master of Science in Aerospace Engineering

Embry-Riddle Aeronautical University  
Daytona Beach, Florida  
November 2014



---

**A CONTINUOUS/DISCONTINUOUS FE METHOD FOR THE 3D  
INCOMPRESSIBLE FLOW EQUATIONS**

**Copyright 2014  
by Nikolaos Kyriazis**



---

**A continuous/discontinuous FE method for the 3D incompressible  
flow equations**

By

**Nikolaos Kyriazis**

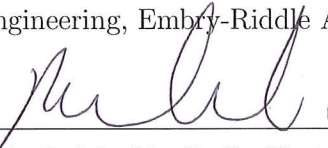
This thesis was prepared under the direction of the candidate's thesis committee chairman, Dr. John A. Ekaterinaris, Professor of Aerospace Engineering, and has been approved by the members of his thesis committee. It was submitted to the Aerospace Engineering department and was accepted in partial fulfillment of the requirements for the degree of Masters of Science in Aerospace Engineering.

THESIS COMMITTEE



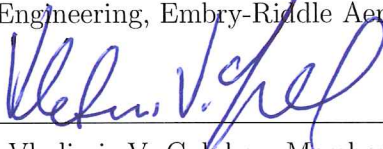
---

Dr. John A. Ekaterinaris, Chairman  
Professor of Aerospace Engineering, Embry-Riddle Aeronautical University



---

Dr. Reda R. Mankbadi, Co-Chairman  
Professor of Aerospace Engineering, Embry-Riddle Aeronautical University



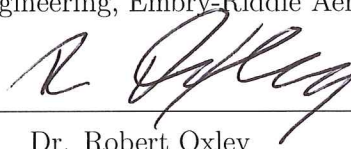
---

Dr. Vladimir V. Golubev, Member  
Professor of Aerospace Engineering, Embry-Riddle Aeronautical University



---

Dr. Yi Zhao, Graduate Program Coordinator  
Professor of Aerospace Engineering, Embry-Riddle Aeronautical University



---

Dr. Robert Oxley  
Associate Vice President for Academics, Embry-Riddle Aeronautical University

11/12/14  
Date



# Abstract

A projection scheme for the numerical solution of the incompressible Navier-Stokes equations is presented. Finite element discontinuous Galerkin (dG) discretization for the velocity in the momentum equations is employed. The incompressibility constraint is enforced by numerically solving the Poisson equation for pressure using a continuous Galerkin (cG) discretization. The main advantage of the method is that it does not require the velocity and pressure approximation spaces to satisfy the usual inf-sup condition, thus equal order finite element approximations for both velocity and pressure can be used. Furthermore, by using cG discretization for the Poisson equation, no auxiliary equations are needed as it is required for dG approximations of second order derivatives. In order to enable large time steps for time marching to steady-state and time evolving problems, implicit scheme is used in connection with high order implicit RK methods. Numerical tests demonstrate that the overall scheme is accurate and computationally efficient.





In memory of my best friend Dimitris, who left life so early.

---

# Acknowledgements

First of all, I would like to thank my parents, Sotiria and Thanasis, for emotional and financial support throughout years. Special thanks should be given to my sister Eleftheria, friends and my girlfriend Aggeliki for encouraging me to pursuit my dreams. I am also grateful to my advisor, Dr. Ekaterinaris for being such a great mentor. Finally, I acknowledge the Research Funding Program THALES, as well as Embry-Riddle Aeronautical University for funding and providing me with access to Zeus and Rigel clusters.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Historical Perspective . . . . .	3
1.2.1	Historical review of the discontinuous Galerkin method . . .	7
1.3	Discretization Methods in CFD . . . . .	9
1.3.1	Finite Difference Method . . . . .	9
1.3.2	Finite Volume Method . . . . .	10
1.3.3	Finite Element Method . . . . .	11
1.4	Solution Algorithms for Incompressible Flow . . . . .	15
1.5	Objective . . . . .	16
<b>2</b>	<b>Governing Equations</b>	<b>19</b>
2.1	Governing Equations . . . . .	19
2.2	Analytical Solutions . . . . .	20
2.2.1	Inviscid Flow over a Cylinder . . . . .	21
2.2.2	Kovasznay Flow . . . . .	21
<b>3</b>	<b>Finite Element Framework</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	Expansion Bases . . . . .	23
3.3	Elemental Operations . . . . .	26
3.3.1	Numerical Integration . . . . .	26
3.3.2	Differentiation . . . . .	29
3.4	Global Operations . . . . .	29
3.4.1	Mapping Process . . . . .	30
3.4.2	Modal Connectivity . . . . .	31
3.4.3	Dirichlet Boundary Condition Enforcement . . . . .	34
3.4.4	Parallelization . . . . .	34

---

<b>4</b>	<b>Discontinuous/Continuous Finite Element Discretization of the Governing Equations</b>	<b>37</b>
4.1	Introduction . . . . .	37
4.2	Projection Method . . . . .	38
4.3	dG discretization for the momentum equations . . . . .	40
4.3.1	Inviscid Numerical Flux . . . . .	41
4.3.2	Viscous Numerical Flux . . . . .	44
4.4	cG discretization for the pressure Poisson equation . . . . .	45
4.4.1	Poisson solver Validation . . . . .	46
4.5	Time Marching Scheme . . . . .	47
4.5.1	Explicit Algorithm . . . . .	48
4.5.2	Implicit Algorithm . . . . .	50
4.5.3	The GMRES Method . . . . .	57
<b>5</b>	<b>Numerical Examples</b>	<b>59</b>
5.1	Introduction . . . . .	59
5.2	Inviscid Flow . . . . .	59
5.2.1	Inviscid Cylinder . . . . .	59
5.2.2	Inviscid NACA 0012 . . . . .	60
5.3	Viscous Flow . . . . .	61
5.3.1	Kovaszny Flow . . . . .	61
5.3.2	2D Lid Driven Cavity . . . . .	63
5.3.3	Flow over a flat plate . . . . .	64
5.3.4	Viscous Cylinder . . . . .	65
5.3.5	Viscous NACA 0012 . . . . .	66
5.3.6	3D Lid Driven Cavity . . . . .	68
<b>A</b>	<b>Linear Elasticity</b>	<b>73</b>

# List of Figures

3.1	Indices for the modes in the three dimensions . . . . .	25
3.2	Numbering of global and local hexahedral nodes . . . . .	27
3.3	Local numbering of vertices, edges and faces . . . . .	31
3.4	Index numbering in the quadrature points . . . . .	32
3.5	Modes $C^0$ continuity . . . . .	33
3.6	Ghost element for no slip wall boundary condition. . . . .	33
3.7	Communication pattern in the dG method . . . . .	36
3.8	Communication pattern in the cG method . . . . .	36
4.1	2D Poisson verification . . . . .	47
4.2	3D Poisson verification . . . . .	47
4.3	Jacobian-free Newton Iteration for implicit time marching . . . . .	54
4.4	Jacobian matrix and block Jacobi preconditioner pattern . . . . .	54
5.1	Inviscid cylinder simulation . . . . .	60
5.2	Inviscid NACA 0012 simulation . . . . .	61
5.3	Velocity and pressure comparison in Kovasznay Flow . . . . .	62
5.4	Velocity and Pressure fields in Kovasznay Flow . . . . .	62
5.5	Spatial accuracy for Kovasznay flow . . . . .	63
5.6	Steady state u and v velocity profiles in 2D Cavity . . . . .	64
5.7	Velocity fields in 2D Cavity . . . . .	64
5.8	Boundary layer profile . . . . .	65
5.9	Viscous cylinder flow visualization . . . . .	66
5.10	NACA 0012 contours . . . . .	67
5.11	Pressure coefficient comparison for NACA 0012 . . . . .	67
5.12	Steady state u and v velocity profiles in 3D Cavity . . . . .	68
5.13	Velocity streamlines in 3D Cavity . . . . .	69
A.1	Verification for cantilever beam with uniform load . . . . .	76
A.2	Comparison between analytical and computational solutions in the string vibration case . . . . .	77





# List of Tables

4.1	The Butcher tableau for the explicit Runge-Kutta method. . . . .	49
4.2	The Butcher tableau for the explicit $RK(2,2)$ method. . . . .	50
4.3	The Butcher tableau for the implicit Runge-Kutta method. . . . .	56
4.4	The Butcher tableau for the $DIRK_2$ method. . . . .	56



# Chapter 1

## Introduction

### 1.1 Motivation

The incompressible Navier-Stokes equations have a wide range of applications, some of the most important are horizontal and vertical axis wind turbines, biomedical flows such as in the cardiovascular system, vehicle and building aerodynamics and hydrodynamics for swallow water and oceanography. The final objective of this work is to develop a strongly coupled method for fluid structure interaction (FSI) of low speed flows in horizontal axis wind turbines. Among others, Bazilevs et al. [10], Galdi and Rannacher [41], as well as Duarte, Gormaz and Natesan [37] have recently presented developments in this field.

Fossil fuels are responsible for the fast global climate change, emitting tremendous amounts of harmful gases in the atmosphere. Therefore a need for clean and renewable sources of energy has been created. Wind turbines are an increasingly important source of renewable energy and nowadays have become very popular. Large amounts of research and resources are being spent in order to harness wind energy effectively. Designing a wind turbine is a complex issue not only because the blades must have an aerodynamic design, but also a complete wind power system must be constructed taking into account aeroelasticity effects. For this reason, three dimensional simulations must be performed. Furthermore, for the large scale wind turbines, accurate prediction of aeroelastic coupling is necessary to avoid serious accidents during operation.

In the wind turbine, the flow depends on the shape and the motion of the blade, whereas the motion and the deformation of the blade depend on the fluid

mechanics forces applied on it. Computational methods have been used in FSI due to the non linearity and time dependence of the coupled system of differential equations. The fluid and structure domains do not overlap, but in their interface the coupling is achieved by satisfying physically interface conditions. Another important issue in FSI is the discretization of the fluid-structure interface. If different meshes are used in the two domains, correct coupling between the tractions and the displacements must be made. Although this choice is challenging, it has many advantages such as refinement techniques can be applied either in fluid, or in the structured part. Furthermore, different discretization methods can be used for solving the elasticity and the flow equations. The mesh for the fluid domain must be finer in critical areas of the interface, for example in the leading and the trailing edge in order to capture the velocity and pressure gradients. On the other hand, this is not the case in the structural domain, where for example finer mesh around holes will be necessary. Due to these conflicting requirements the use of the same mesh in both sets of equations becomes computationally inefficient, especially for complex geometries. As a result, the mesh at the interface of the fluid and the solid is different.

The Eulerian frame of reference is usually used for the fluid mechanics equations, whereas Lagrangian framework is used for the non linear elasticity equations. In the Eulerian description the grid is fixed in space and the conservation equations are developed for a fixed in space control volume. In Lagrangian description, the mesh points are actually the material particles, and a new mesh is created after the displacements are found. Arbitrary Lagrangian Eulerian (ALE) description is combining the advantages of the two previous methodologies, where the mesh can be moved in an arbitrary way. This approximation has the features of Eulerian approximation when the mesh cannot follow the fluid motion, and the features of Lagrangian approximation when relatively small motion takes place.

The most important benefits of the ALE method are:

- Interfaces and solid boundaries keep their boundary conditions by moving the boundary and interior nodes of the grid.
- CPU time can be reduced because no re-meshing is needed.
- The so called projection error can be avoided. Projection error is the error

which is created when the numerical solution is projected from the old mesh, to the new mesh.

- Time accuracy must be retained.

In order to combine all the previous attributes, discontinuous Galerkin (dG) and continuous Galerkin (cG) approximations with modal shape functions will be used for the formulation. Modal bases are ideal for different mesh in the interface because the solution doesn't actually refer to a specific point and higher order accuracy polynomials and schemes can be created. The Legendre polynomials are used for the construction of the modal bases which can facilitate transfer of information between the fluid and structural solvers through the arbitrary Lagrangian Eulerian (ALE) coupling approach. Mixed dG-cG formulation was used for the numerical solution of the incompressible Navier-Stokes equations combining this way their main attributes. The dG methods are suitable for the momentum discretization because of their upwinding formulation. The cG method was used for the pressure Poisson equation avoiding this way to solve the auxiliary equations for the pressure gradients. Furthermore, using cG formulation, the discretized Poisson equation is solved only once for each time step.

In the next section, a historical review of the most important accomplishments in CFD is made. Since the dG method is going to be implemented, a historical review of this method is presented as well. Because there are many different discretization methods and solution algorithms for the incompressible flows, a review in the most important of them is made, stating their main attributes and drawbacks.

## 1.2 Historical Perspective

Recently, Shang [80] presented a historical review of CFD mainly for compressible flow. Some historical perspective of numerical methods for incompressible flow can be found in Cockburn, Karniadakis and Shu [22], as well as in [33] and [3].

Richardson [72], Courant, Friedrichs and Lewy [27], Southwell [83], von Neumann [93], Lax [55] and Godunov [43] are considered to be the initiators of computational fluid dynamics (CFD), traced back in the early 1900s. They focused on

solving discontinuous fluid phenomena, the famous Riemann problem [74]. The first numerical solution for viscous flow around a circular cylinder was obtained by Thom [88] in the early 1930s. Harlow [47] proposed the particle in cell (PIC) method in 1957 which is a combination of Eulerian and Lagrangian description of the fluid motion. The PIC method was able to simulate multi-dimensional and time dependent fluid problems, which is why it was the first breakthrough in the field of CFD for incompressible flow and it was used for the development of future algorithms and schemes.

Flow separation in a boundary layer is a problem of great interest until nowadays. Davis [29] solved accurately the compressible boundary layer equation by using a combined implicit-explicit finite difference scheme. Stewartson [84] indicated that the flow separation singular point can move and provided a scaling law for the interacting boundary layers.

For inviscid three dimensional supersonic flow the method of characteristics was developed. Rakich [70] created a three dimensional mesh in order to solve the flow field around Mach cone shock fitting. Moretti and Abbett [63] solved time dependent Euler equations using finite difference method and obtained a steady state solution. Their main breakthrough was that they successfully incorporated Rankine-Hugoniot conditions into their scheme. Furthermore, the Euler equations maintained their hyperbolic type even in subsonic flow due to the time discretization. Small disturbances theory was used in inviscid subsonic flows around aircraft [76] and it has been used for airplane design even nowadays.

Dean Chapman, Director of Aeronautical Science Directorate of the NASA-Ames Research Center, had the vision to create an organization specializing in fluid dynamics for Aerospace and Aeronautic applications. For that purpose, talented scientists in the field of fluid dynamics and computer design were recruited. Their combined efforts and collaboration between different NASA research centers, resulted in great development around computational aerodynamics.

In 1969, McCormack published his well-known explicit predictor-corrector algorithm [59]. The MacCormack algorithm is a second order accurate scheme for resolving aerodynamic problems, even when strong flow field gradients are present, due to a damping term which adds artificial viscosity which is proportional to the

local pressure gradient. His numerical scheme has been successfully used for more than 30 years. In the 1970s, a three dimensional hypersonic compression corner simulation was performed using McCormack's scheme and giving great agreement with the experimental results [81].

Transonic flow in the Euler equations are of elliptic, parabolic or hyperbolic type depending on the existence of the flow field in the subsonic, transonic or supersonic domain respectively. Murman and Cole [64] used a combination of central, backward and forward differences for the simulation of the transonic flow field.

Jameson [48] had numerous contributions in CFD, among them are his numerical scheme for transonic flow, the multigrid algorithm and aerodynamics optimizing techniques. Multigrid is a technique for solving large systems and achieving faster convergence by solving a smaller system instead of the original one. The multigrid methods are widely used for incompressible flows and they are generally classified into the geometric and the algebraic multigrid. Brandt [15] introduced the geometric multigrid algorithm. In this method, the low frequency numerical error is being filtered out by interpolating the finer grid result to the coarser grid, and the opposite process for constructing the corrected solution in the finer mesh is followed. Use of multigrid can dramatically accelerate the numerical solution of the Poisson equation for pressure. In general, some type of multigrid acceleration, geometric or algebraic must be employed for Poisson solvers in large scale computations because the Poisson equation is of elliptic type and domain decomposition methods do not help much.

Thompson's technique for grid generation [90] is the reason that CFD became vastly used in practical applications, where differential equations of various types have been solved in order to construct the mesh. MacCormack's implicit scheme was used for the aerodynamic performance of the X-24C vehicle while it was entering the atmosphere [82]. The simulation was performed on the Cray 1 computer and the mesh was constructed from Steger's GRAPE grid generation software.

Briley and McDonald used the alternating direction implicit scheme (ADI) for implicitly advancing in time the Navier-Stokes equations in the 1970s [16], [17]. The scheme was unconditionally stable but matrix inversion was needed,



resulting in larger memory requirements. Later, Beam and Warming solved the compressible Navier-Stokes equations by using factorized implicit algorithms [11]. ADI scheme was further developed by Pulliam and others and it became the most famous algorithm in CFD for the next few decades.

In the 1980s the finite volume schemes became widely used. The first numerical solution using finite volume discretization was from MacCormack and Paullay [61] in 1972. However, Rizzi and Inouye were the first who used the term finite volume in 1973 [75]. Some other famous contributions to finite volume method are the implicit Gauss-Seidel relaxation algorithm implemented by Thomas and Walters [89] and MacCormack [60], as well as the van Leer upwind flux [92]. Ideas from finite volume method are used in the present work and more information about the method is given in subsection 1.3.2 .

Complex geometry representation requires the use of unstructured meshes because the excessive number of cells increases the computational time and could make the analysis inefficient. Therefore an unstructured mesh framework was created. Instead of using hexahedrals in three dimensions, prisms, pyramids and tetrahedrals have been introduced. The first scheme used for unstructured grid was Delaunay's scheme [31] for triangular and tetrahedral elements in two and three dimensions, respectively. Later, more advanced grid generation methods enabled multi type elements in the domain of interest. Unstructured grid is considered to have better scalability than the structured, making parallel computing more efficient, as well as it is suitable for adaptive mesh refinement. It must be emphasized that most discretization methods have better efficiency for Cartesian meshes. Therefore, the overset grid concept with Cartesian-type meshes in the far field and body conforming grids for the near flow, interfaced with the Cartesian off body grids with the Chimera approach, gained significant popularity.

Advancement in computer science enabled large scale simulation using parallel systems. Strang used a parallel network of computers and provided the numerical solution of the Euler and Navier-Stokes equations [85]. As a result, aerodynamic simulations of aircrafts became possible. In this work the massively parallel environment for many cores was adopted through domain decomposition using *METIS* library [50] and MPI prototype.

Modeling of turbulence has always been a challenging issue for the CFD due to its stochastic character. The approaches used for numerical solution of turbulent flows are direct numerical simulation (DNS), large eddy simulation (LES), and the so-called Reynolds-averaged Navier-Stokes (RANS) method. In DNS, the Navier-Stokes equations are solved without any modification or assumption for finding the time dependent velocity and pressure. In order to resolve the small scales the grid points must be proportional to  $Re^{9/4}$  for a three dimensional problem, thus making the method computationally inefficient. In the LES approach, velocity and pressure are computed exactly like in DNS found for the moderate and large scales, whereas the small scale effects are captured by using model approximations. Finally in the RANS approximation, mean velocity, pressure and eddy viscosity or Reynolds stresses are computed. In order to close the system of equations, the eddy viscosity hypothesis is often employed. Depending on the turbulence model being used, the eddy viscosity and the Reynolds stresses can be found from one or two equation models.

Nowadays the fundamental equations of linear or non-linear structure dynamics or electromagnetics are solved in a coupled fashion with the Navier-Stokes equations. Furthermore numerical schemes for the aerothermodynamics of the hypersonic flow, species reaction and computational magneto-fluid dynamics have also been developed.

### 1.2.1 Historical review of the discontinuous Galerkin method

In this work, the dG method is used as basic discretization scheme for the momentum equation. Therefore a brief historical review of the developments in the dG method is given. The dG finite element method was introduced for solving linear hyperbolic systems. More specifically, Reed and Hill [71] proposed for the first time the dG method in order to solve the neutron transport equation in 1973. One year later, LeSaint and Raviart [56] made the first analysis of the dG method and showed that the method is strongly A-stable of order  $2k + 1$  at mesh points, when polynomials of degree  $k$  are used. Some early applications of the dG method in the late 1970s are wave propagation analysis in elastic media by Oden and Wellford [96], [98] and [97] and to optimal control by Delfour and Trochu [32].

In the 1989, Fortin [40] used the method for numerical computation of viscoelastic flows, by expressing the extra-stress tensor in terms of the velocity. Ten years later, Warburton and Karniadakis [95] discretized the magneto-hydrodynamics equations using dG method.

Since linear hyperbolic equations have been successfully treated by dG method, the method was expanded to non-linear hyperbolic systems. Implicit time discretization had been used because of the nonlinearities in the system of differential equations. Lowie and Morel [58] among others used space-time elements in order to avoid solving the system of the implicit scheme and keep its local character. Explicit schemes had been used to avoid the difficult implicit treatment, first by using the Euler method [19] and later by using the Runge Kutta Discontinuous Galerkin method (*RKDG*). The first *RKDG* method was introduced by Cockburn and Shu [24] and it was second order accurate in time, stable for CFL number less than  $1/3$  and total variation bounded in the means (TVBM). In 1989, Cockburn and Shu [23] generalized the method and created high-order accurate *RKDG* methods for the scalar hyperbolic conservation law. The success of this scheme was that the accuracy wasn't destroyed by the slope limiter and no oscillations were noticed in the numerical solutions.

The dG method does not impose continuity constraints for the solution at the element interfaces. This gives local character to the dG but introduces difficulties for second order derivative discretization. In 1991, Dawson [30] used for first time upwind-mixed methods (UMM) for advection-diffusion equations by upwinding the convective terms. One year later, convection-diffusion discretization was provided by Richter [73]. One of the first attempts in dG framework for treating the viscous terms (second order derivatives) in the Navier Stokes equations is the local discontinuous Galerkin (LDG) method by Cockburn and Shu [25] which was a generalization of Bassi's and Rebay's [5] approach for the compressible Navier-Stokes equations. The basic idea is to write the second order system in a bigger first order system and then carefully choose the numerical fluxes. So in this approach the velocity vector field and the velocity gradient tensor are treated as independent unknowns and thus auxiliary equations for computing the velocity gradients are derived.

Some recent developments in the dG method and turbulence modeling are RANS equations coupled with either k-omega or Spalart-Allmaras turbulence models by Bassi [7], Oliver [67] and others. Arbitrary Lagrangian-Eulerian (ALE) formulation has been incorporated into the dG method [69], allowing the interior computational mesh to move, while the boundary mesh moves along the materials. This formulation is mostly used in fluid structure interaction problems (FSI).

## 1.3 Discretization Methods in CFD

In this section the most common discretization methods in CFD are presented and their main features are briefly outlined in order to point out advantages of the Finite Element method that was chosen in this work. Finite differences and finite volumes, are the most common low order (second order) methods in CFD. High order methods are finite difference explicit and compact schemes, as well as the ENO and WENO schemes [3]. These methods are quite efficient computationally, they produce the design order of accuracy for moderately stretched structured meshes but they involve wide stencils. Finite element approximation, such as continuous Galerkin (cG), discontinuous Galerkin (dG) and Hybridizable Discontinuous Galerkin (HDG) are well suited for high order accurate discretizations with a compact stencil in unstructured meshes. The main consequence of low or high order of approximation, is that in the former only mesh refinement (h-refinement) can be applied. On the other hand, in high order methods both mesh and polynomial refinement (p-refinement) can be applied. References used for this part are Ferrer [38], Shahbazi [79] and Karniadakis [49].

### 1.3.1 Finite Difference Method

Finite difference method is the first discretization technique used in numerical methods and it was introduced by Euler in the 18th century. The physical domain is discretized by a set of grid points, where the variable values are unknown. In this method the differential form of the equations is transformed into a system of algebraic equations usually by using Taylor series expansions in order to approximate the derivatives. Another equivalent way to approximate the derivatives is

by using polynomial fitting.

The main advantages of the method are ease of implementation and reduced computational cost. The main drawback in that method is that it cannot be applied in complex geometries. Metrics need to be computed if the grid points don't form canonical quadrilateral or hexahedral in the two and three dimensions respectively and the transformed equations must be solved. Finally, depending on the accuracy which is needed, more terms in the series expansion are taken in order to approximate the first or the second derivative. That way, imposing the boundary conditions becomes a complicated issue and parallel implementation with domain decomposition becomes less efficient as the stencil width increases.

### **1.3.2 Finite Volume Method**

The finite volume method became very popular in the 1980s and has been applied for conservation laws in the Eulerian reference frame. It is self explanatory that the physical domain is discretized in finite volumes. The integral form of the Navier-Stokes is used where mass, momentum and energy are conserved both in the element and in the total domain. The Gauss' divergence theorem is applied to transform the viscous and inviscid terms from volume integrals to surface integrals in a three dimensional mesh. Finally, the fluxes from the neighbor volumes are used in order to calculate the surface integrals.

The most important feature of this approximation is easy implementation of the numerical scheme for the conservation laws, making the method appropriate for complex geometries, as a result the FV method was used in many industrial applications. One unknown for each variable per element has to be computed. Therefore, the total number of unknown degrees of freedom is relatively small, making large mesh simulations feasible. Furthermore, no metrics need to be found, the only geometrical operations needed are calculation of the area and the volume of each face and cell respectively. In addition to this, the flux vector which is normal to each cell surface has to be calculated for the diffusive and convective terms.

Although finite volumes have been successfully used for more than 30 years , they have specific disadvantages. High order accuracy cannot be easily achieved

and it is extremely computationally inefficient, because the stencil is becoming very large. Coupling with other discretization schemes is difficult to be achieved, restraining the choices of the numerical method, for example in FSI simulations .

### 1.3.3 Finite Element Method

In the finite element method (FEM) the variables are approximated by the product of the transposed shape function vector and the solution vector. For the finite element discretization, the weak form of the equations is used, which is created by multiplying the partial differential equations with a weight function and integrating by parts, using Gauss divergence theorem. The values of the surface and volume integrals are calculated using numerical integration at specific quadrature points, whose the number and the location depend on the order of the accuracy used.

If the test function is the same as the shape function, the FEM is called Galerkin approximation. A well known method where the weight and the shape functions are not the same, is the Streamline Upwind Petrov Galerkin (SUPG) method, due to the upwinding of the weight functions in the streamwise direction. Nodal (Lagrange polynomials), modal (Legendre or Jacobi polynomials), even nurbs that could achieve higher order continuity and geometric representation have been employed as basis functions. The most widely used finite element techniques are continuous Galerkin (cG), discontinuous Galerkin (dG) and Hybridizable Discontinuous Galerkin (HDG), which is actually in the family of dG schemes.

In finite element schemes, the polynomial basis and the numerical integration are responsible for the order of accuracy, so the latter can be increased simply by changing the order in the input file. As a result, higher order polynomials will be used for the basis construction and more quadrature points at new locations will be taken. This is of major importance, because no sophisticated interpolations and fluxes from far away elements are needed.

## Continuous Galerkin Method

In the cG method, there is at least  $C^0$  continuity, which means the approximated variable has to be continuous in the face between the elements. If  $k$  order continuity is achieved, it means that the variable and all the derivatives up to  $k$  order, are continuous. Since the basis and the variable are continuous, there is no need for numerical fluxes and all the interior surface integrals are eliminated. Surface integrals are only used for the Neumann type boundary conditions. The Dirichlet type boundary conditions are enforced either by the row and column elimination, or by the penalty approach.

The main attributes of the cG method is that high order of accuracy can be guaranteed, while fast convergence with low diffusion and dispersion errors is achieved. The number of the equations to be solved is smaller than those in dG, because no auxiliary equations for the gradients of the unknown variables have to be derived, which is the case in the dG method. In addition to this, the total degrees of freedom (DOFs) for the unknown variable in the system are much more less than those in the dG method, especially for polynomials of lower order. Although only implicit schemes are derived, due to the continuity of the variable, only one iteration is required for steady state problems, whereas using dG methods, multiple iterations will be necessary until convergence is achieved. Finally, by using a specific numbering for the global degrees of freedom (more details in chapter 3), the matrix has most of his elements in or close to the diagonal, so it has a better matrix condition, thus it can be easier solved.

However the previous approximation has the following drawbacks. The continuous framework has to be constructed, for example algorithms for mapping the local DOFs into the global DOFs, the edge connectivity between neighbor elements sharing the same edge, probably remapping in order to eliminate the variables which are known from the Dirichlet type boundary conditions etc. All these are not needed in the dG method. Finally, the parallelization in cG schemes is more difficult than using dG schemes, because of the mapping. Even if an element belongs to a specific rank, the correspondent global DOFs maybe belong to another rank, making the communication between ranks necessary. The num-

ber of the unknown coefficients per element for each variable is much larger than in the finite volumes discretization, but this is the case in every finite element method. In most cG formulations, local conservation cannot be achieved due to the continuity of the bases, and global conservation can only be achieved if there are no Dirichlet type boundary conditions.

### **Discontinuous Galerkin Method**

In the dG method, there is no continuity across the element boundaries and the conservation of the flow properties is ensured by the numerical fluxes used in the surface integrals. The numerical fluxes must be carefully chosen in order for the scheme to be stable and consistent. Boundary conditions can be easily treated using ghost elements approach, so they are weakly enforced in comparison with the cG method, where the boundary conditions are usually strongly enforced.

One of the most important advantages of the dG method is that the mass, momentum and energy are preserved in a local manner and global manner (locally and globally conservative method) due to the finite volume character achieved through discontinuous test functions. Physical discontinuities can be captured without any oscillations around them, for example shock waves in compressible flow problems. High order accuracy can easily be achieved using higher order polynomials for the shape functions. The minimum order of accuracy is  $k + 1/2$  and usually for smooth flow problems the accuracy is  $k + 1$ , when polynomials of  $k$  order are used. The bases which are often used are hierarchical, thus no actual change is needed in the numerical scheme in order to obtain a higher order approximation. Both p-type and h-type refinement can be achieved using dG discretization. In the non-conforming h-type adaptivity, the refined mesh has hanging nodes, which can be easily handled if modal basis have been used. In the p-type refinement, the order of the polynomial used for the dG approximation can be different from element to element, thus better resolution in elements of interest is achieved. Discontinuous Galerkin is a compact method, thus communication only with the immediate neighbors is needed, regardless of the order of the scheme, making the method easily parallelizable. Complicated geometries can be captured by using unstructured or mixed type mesh. Boundary conditions can be easily



satisfied in the dG formulation and due to the discontinuity of the approximation variables, no special treatment in higher order scheme is needed, which was the case in finite volume and difference.

However the main disadvantage of the method is that it is more computationally expensive compared to the FD and FV methods. For example, if first order polynomials have been used in order to give second order of accuracy in space, the number of the bases used for hexahedral elements is eight, which means that for each element 8 unknown coefficients must be found for approximating one variable. The implementation of the scheme is much more sophisticated compared to low order methods. Moreover, auxiliary equations for the variable gradients are needed, the only case to avoid that is by using Interior Penalty Method (IPM) [1]. When solving numerically the INS equations, nine auxiliary equations are required which have to be solved at each time step in order to find all the necessary velocity gradients, needed in the viscous flux computation.

### **Hybridizable discontinuous Galerkin Method**

The HDG method is actually a dG scheme with special choice of the numerical traces and fully implicit formulation [66]. Unique value for the trace along the element boundaries is derived by enforcing the continuity of the normal component of the flux across the element boundary. The global equation system is only in terms of the approximate trace, whereas all the other variables are treated explicitly.

The most important feature of the method, is that the unknowns from the implicit scheme are the trace of the velocity and the mean of the pressure on element boundaries, thereby there is reduced number of degrees of freedom. The HDG method has superconvergence properties for the velocity field, and after elemental post-processing, the velocity field is exactly divergence free and converges with order  $k + 2$ , if  $k$  is the degree of the polynomial used in the expansion. Finally, unified treatment is used for both the viscous and the inviscid numerical fluxes by using a single numerical flux.

On the contrary, HDG suffers from serious disadvantages, difficulty in parallelization among others. The main reason for that is that there are several matrix

multiplications and other operators in order to decouple the traces from the other unknowns. Moreover, there is high computational cost in memory, due to the global matrices storage which is required.

## 1.4 Solution Algorithms for Incompressible Flow

The compressible Navier-Stokes equations describe the flow when the flow speed is significant ( $M = \frac{u}{a} > 0.3$ ) relative to the speed of sound. If the speed of sound of the medium becomes infinity, the compressible flow equations become singular. This singularity is the main difficulty in solving incompressible flow using a compressible flow formulation. The challenge in incompressible flow is to satisfy the continuity equation, which is often used as a constraint rather than a new equation to be solved. Many different techniques have been used for surmounting this problem and for that reason there is an ambiguity in the literature as far as the different solution algorithms are concerned. The most popular techniques for solving INS are summarized by Karniadakis and Sherwin [49], as well as Deville, Fischer and Mund [33]. The two different methodologies for solving the INS are fully coupled and uncoupled methods for the velocity and pressure.

In coupled algorithms, the velocity field and the pressure are solved simultaneously. The first uncoupled method for the linear Stokes equations was probably the Uzawa algorithm [62] which converges slowly and it is computationally expensive. Later coupled techniques, regardless of the discretization method, incorporate the unknown pressure variable into the continuity equation, such as the artificial compressibility method. In this method, a pseudo-time derivative of pressure, multiplied by an artificial compressibility parameter, is added in the continuity equation [46]. Now the system is hyperbolic-parabolic type and thus implicit schemes and discretization methods designed for compressible flow can be applied. A time marching scheme is used until the divergence of the velocity converges to a specific tolerance. The incompressibility parameter has to be chosen to its highest possible value, so as the incompressibility is quickly recovered. Other methods treating continuity equation in terms of velocity and pressure are Hybridizable Discontinuous Galerkin method (HDG) [66] and discontinuous

Galerkin method where the incompressible numerical flux of the velocity in the divergence free condition is a function of both velocity and pressure [57]. Coupled algorithms are generally considered to give exactly divergence free velocity field.

The uncoupled methods which are usually much more efficient, since two smaller systems for the velocity field and the pressure are solved, are divided into velocity correction [36] and projection methods [33]. In these methods, the velocity field at the new step is found without calculating the pressure field. On the one hand, in the velocity correction schemes, the viscous term is treated explicitly or ignored in the first substep and a correction to the velocity is made in the second substep. More specifically, in the rotational velocity correction formulation [36], the rotational form of the viscous term is used in order to obtain a high-order splitting scheme. On the other hand, in the projection methods, pressure is treated explicitly or ignored in the first substep, and an intermediate velocity field which is not divergence free is calculated. This field is projected on the solenoidal space. Projection methods are also divided into fractional step methods and pressure correction methods. The former, is a splitting technique for decoupling nonlinear convection terms and the diffusion terms in the INS equations into separate problems. In the beginning, the INS equations are solved without the pressure gradient term at all and next the pressure Poisson equation is solved. In the later, the INS are solved using an old time level value for the pressure in order to find a velocity field which is not divergence free. In the second step, pressure correction is performed, by subtracting the old from the new time level pressure and it is added to the velocity field in order to satisfy the divergence free condition. Although split algorithms don't give exactly divergence free velocity field, they are robust and computationally efficient.

## 1.5 Objective

A numerical algorithm for the numerical solution of the unsteady three dimensional incompressible Navier-Stokes equations has been developed using finite element discretization techniques. The main objectives of the thesis are the following:

1. To develop high order accurate method in space using a hybrid dG and

cG discretization approach for the velocity vector and the pressure fields respectively. The h/p Spectral Element framework is employed to obtain unified discretization of the fluid and solid domains.

2. To ensure time accuracy and overcome CFL stability limitation, implicit Runge-Kutta methods are used for time marching.
3. To implement parallelism using Message Passing Interface (MPI) in order to make possible large scale simulation in massively parallel computer architectures.
4. To perform verification and validation of the solution algorithm for its accuracy and stability by solving two and three dimensional problems. The algorithm is verified by comparing with the analytical solution and validation is performed by comparing with the experimental results.

Although the long term application of the numerical method is accurate prediction of the flow around wind turbine blades, there are many more applications in three dimensional incompressible flows. The rest of the thesis is organized as follows. In chapter 2, the non-dimensional form of the governing equations in three dimensions is presented. Some exact solutions for simple problems are also given and later used for verification. In chapter 3, the continuous and discontinuous finite element framework is described by employing the three dimensional expansion bases, local and global operations. The numerical scheme used and the pressure-velocity decoupling using the projection method combined with time marching techniques are introduced in chapter 4. Numerical examples and verification of the method by performing comparison with analytical solutions has been shown in chapter 5. Finally, all the work is summarized and future directions are given.



# Chapter 2

## Governing Equations

### 2.1 Governing Equations

We seek the numerical solution for the unsteady three dimensional Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega, \quad (2.1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \mathbf{f} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \quad \text{in } \Omega, \quad (2.2)$$

where Eq. (2.1) is the continuity equation and Eq. (2.2) are the momentum equations, in the physical domain  $\Omega$ . The boundary faces  $\partial\Omega$  are divided into Dirichlet  $\partial\Omega_D$  and Neumann  $\partial\Omega_N$  type, where  $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$  and  $\partial\Omega_D \cap \partial\Omega_N = \emptyset$ . In order to close the system, the following initial and boundary conditions are specified.

$$\mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}_0, \quad \text{in } \Omega, \quad (2.3)$$

$$\mathbf{u} = \mathbf{u}_D, \quad \text{on } \partial\Omega_D, \quad (2.4)$$

$$\mu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} = p \mathbf{n}, \quad \text{on } \partial\Omega_N, \quad (2.5)$$

where  $\mathbf{u}$  is the velocity vector field  $\mathbf{u} = [u \ v \ w]^\top$ ,  $p$  is the static pressure,  $\rho$  is the density,  $\mathbf{f}$  is the body force vector,  $t$  is the time and  $\nu$  is the kinematic viscosity given by:  $\nu = \frac{\mu}{\rho}$ . The initial velocity is  $\mathbf{u}_0$ , whereas  $\mathbf{u}_D$  is the specified velocity at the Dirichlet type boundaries  $\partial\Omega_D$  in Eq. (2.4). The gradient of the

velocity at each direction in the Neumann type boundaries  $\partial\Omega_N$  is given by the condition in Eq. (2.5).

The non-dimensional form of the governing equations is given by normalizing the static pressure by the upstream dynamic pressure:  $\hat{p} = \frac{P}{\rho U_\infty^2}$ , while the velocity is non dimensionalized by dividing by the free stream velocity  $\hat{\mathbf{u}} = \frac{\mathbf{u}}{U_\infty}$ , the non dimensional time is  $\hat{t} = \frac{tU_\infty}{L}$ , where  $L$  is the characteristic flow length scale.

$$\hat{\nabla} \cdot \hat{\mathbf{u}} = 0, \quad \text{in } \Omega, \quad (2.6)$$

$$\frac{\partial \hat{\mathbf{u}}}{\partial \hat{t}} + (\hat{\mathbf{u}} \cdot \hat{\nabla}) \hat{\mathbf{u}} = \hat{\mathbf{f}} - \hat{\nabla} \hat{p} + \frac{1}{Re} \hat{\nabla}^2 \hat{\mathbf{u}}, \quad \text{in } \Omega, \quad (2.7)$$

$$\hat{\mathbf{u}}(\hat{\mathbf{x}}, \hat{t}) = \hat{\mathbf{u}}_0, \quad \text{in } \Omega, \quad (2.8)$$

$$\hat{\mathbf{u}} = 1, \quad \text{on } \partial\Omega_D, \quad (2.9)$$

$$\frac{1}{Re} \frac{\partial \hat{\mathbf{u}}}{\partial \mathbf{n}} = \hat{p} \mathbf{n}, \quad \text{on } \partial\Omega_N, \quad (2.10)$$

where the Reynolds number is defined as the ratio of inertial to viscous forces and it is given by:  $Re = \frac{U_\infty L}{\nu}$ . For low Reynolds number the viscous effects dominate, whereas in high Reynolds numbers the convective terms are the most important. The interesting fact here is that the flow is independent of the actual value of the inflow velocity, since in Eq. (2.9) the value is unity. As a result, the Reynolds number is the non dimensional parameter that affects the speed of the fluid in the incompressible flow. For simplicity in the rest of this work, the hat symbol is omitted for all the variables, with the understanding that all variables are non dimensional.

## 2.2 Analytical Solutions

In this section analytical solutions for the incompressible Navier-Stokes equations are presented such as the inviscid cylinder case and the Kovasznay flow. These exact results will be used for verification of the numerical algorithm.

---

### 2.2.1 Inviscid Flow over a Cylinder

The analytical velocities and pressure for the Euler equations describing the inviscid and irrotational flow around a cylinder are given by:

$$u = U_\infty + \frac{U_\infty \rho^2 (y^2 - x^2)}{x^4 + y^4 + 2x^2 y^2}, \quad (2.11)$$

$$v = \frac{-2U_\infty \rho^2 xy}{x^4 + y^4 + 2x^2 y^2}, \quad (2.12)$$

$$p = p_\infty + \frac{U_\infty^2 \rho r^2 (x^2 - y^2)}{x^4 + y^4 + 2x^2 y^2}, \quad (2.13)$$

where  $r$  is the cylinder radius. The pressure coefficient for inviscid flow is given by:

$$C_p = 1 - 4\sin^2\theta. \quad (2.14)$$

### 2.2.2 Kovasznay Flow

The Kovasznay flow is one of the few exact steady solutions for viscous incompressible flow. The Kovasznay flow is defined in a square domain  $(-0.5, 1.5) \times (0, 2)$  with boundary conditions for the velocity and pressure taken from the analytical solution in Eq. (2.15)-(2.17). The analytical solution describing the field is given by the following equations [53]:

$$u = 1 - e^{\lambda x} \cos(2\pi y), \quad (2.15)$$

$$v = \frac{\lambda}{2\pi} e^{\lambda x} \sin(2\pi y), \quad (2.16)$$

$$p = 0.5(1 - e^{2\lambda x}), \quad (2.17)$$

where  $\lambda$  is defined as:

$$\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}. \quad (2.18)$$





# Chapter 3

## Finite Element Framework

### 3.1 Introduction

Before deriving the finite element discretization with the Galerkin approximation, the expansion bases will be introduced following the h/p Spectral element framework [49]. In addition, the elemental and global operations needed in dG and cG formulations for calculating the integrals and assembling the global matrix will be outlined. Numerical integration and differentiation are considered to be the local operations, whereas the local to global mapping, modal face connectivity and enforcement of the Dirichlet boundary conditions are referred as global operations.

Let  $\Omega$  denote the physical domain and  $\Omega_j$  the elements belong in:  $\Omega_j \in \Omega$ . All the operations, either the expansion bases or the numerical integration and differentiation, are performed in the computational domain. The standard elements  $K_j$  in the computational domain  $K$  are actually a transformation of the physical elements  $\Omega_j$  and they are normal hexahedrals defined in the space:  $K_j = \{-1 \leq \xi_1, \xi_2, \xi_3 \leq +1\}$ .

### 3.2 Expansion Bases

The modal discontinuous and continuous bases are constructed in the computational space  $K$  using tensor product and then they are transformed back to the physical space  $\Omega$  using a collapsed coordinate system. The shape functions used in velocity approximations are discontinuous, whereas the shape functions used in pressure approximation are  $C^0$  continuous.

The bases are constructed using the Jacobi polynomials  $P_p^{\alpha,\beta}$  which are solutions of the following ordinary differential equation:

$$\frac{d}{dx}[(1-x)^{1+\alpha}(1+x)^{1+\beta} \frac{d}{dx} u_p(x)] = \lambda_p (1-x)^\alpha (1+x)^\beta u_p(x), \quad (3.1)$$

$$u_p(x) = P_p^{\alpha,\beta}, \quad (3.2)$$

$$\lambda_p = -p(\alpha + \beta + p + 1). \quad (3.3)$$

The one-dimensional dG bases used in the present work are:

$$\phi_d^p(\xi) = P_p^{0,0}(\xi), \quad 0 \leq p \leq P, \quad (3.4)$$

where  $d$  subscript stands for discontinuous. The one-dimensional modal  $C^0$ -continuity expansion bases are introduced next:

$$\phi_c^p(\xi) = \begin{cases} \psi_0^\alpha(\xi) = \frac{1-\xi}{2}, & p = 0 \\ \psi_p^\alpha(\xi) = \frac{1+\xi}{2} \frac{1+\xi}{2} P_{p-1}^{1,1}(\xi), & 0 < p < P \\ \psi_P^\alpha(\xi) = \frac{1+\xi}{2}, & p = P \end{cases} \quad (3.5)$$

where  $c$  subscript stands for continuous. In the one-dimensional bases, the boundary modes are the first and last modes, that is for  $p = 0$  and  $p = P$ , whereas the rest of them are the interior modes. The three dimensional bases can be constructed by tensor product of the one-dimensional bases in each of the Cartesian coordinate directions either for the continuous or the discontinuous shape functions:

$$\phi^{pqr}(\xi_1, \xi_2, \xi_3) = \phi_p(\xi_1) \phi_q(\xi_2) \phi_r(\xi_3), \quad 0 \leq p, q, r; \quad p \leq P_1, \quad q \leq P_2, \quad r \leq P_3 \quad (3.6)$$

The shape functions used in pressure approximation are  $C^0$  continuous and they are decomposed into boundary and interior modes. Boundary modes are all the modes with non-zero support on the boundary and interior modes are zero on all boundaries. For three dimensional bases employed in this work, the boundary modes are decomposed in vertex, edge, and face modes. Vertex modes have a unit

magnitude at one vertex and zero value at all the other vertices. Similarly, edge modes have support at one edge and zero value at all the other edges and vertices. Finally, face modes have support at one face and zero support at all the other faces, edges and vertices. Following the indexing shown in Fig. (3.1) [49], several examples of modes are given next.

The vertex mode A is:

$$\phi^{000}(\xi_1, \xi_2, \xi_3) = \psi_0^\alpha(\xi_1)\psi_0^\alpha(\xi_2)\psi_0^\alpha(\xi_3), \quad (3.7)$$

the edges mode AB are:

$$\phi^{p00}(\xi_1, \xi_2, \xi_3) = \psi_p^\alpha(\xi_1)\psi_0^\alpha(\xi_2)\psi_0^\alpha(\xi_3), 0 < p < P_1, \quad (3.8)$$

the face modes ABFE are:

$$\phi^{p0r}(\xi_1, \xi_2, \xi_3) = \psi_p^\alpha(\xi_1)\psi_0^\alpha(\xi_2)\psi_r^\alpha(\xi_3), 0 < p < P_1, 0 < r < P_3, \quad (3.9)$$

and the interior modes are:

$$\phi^{pqr}(\xi_1, \xi_2, \xi_3) = \psi_p^\alpha(\xi_1)\psi_q^\alpha(\xi_2)\psi_r^\alpha(\xi_3), 0 < p, q, r; p < P_1, q < P_2, r < P_3. \quad (3.10)$$

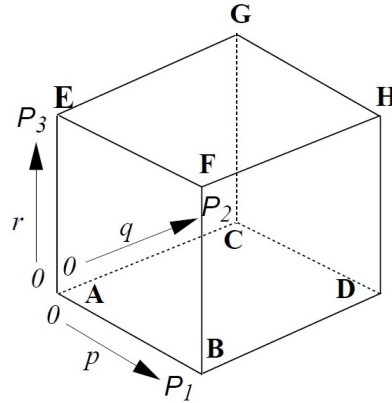


Figure 3.1: Indices for the modes in the three dimensions.

For dG finite element discretization, within each element the number of the discontinuous bases used for a polynomial of degree  $k$  (so order of accuracy  $k + 1$  can be achieved) is:

$$n_{vb_d} = (k + 1)^3, \quad 0 \leq k, \quad (3.11)$$

and the number of the continuous bases used for a polynomial of  $k$  order is:

$$nvc = vertices + edges(k - 1) + faces(k - 1)^2 + (k - 1)^3, \quad 1 \leq k. \quad (3.12)$$

Where *vertices*, *edges* and *faces* is the number of the vertices, edges and faces respectively. In the previous equations  $nvc_d$  and  $nvc_c$  are actually the same for a given polynomial order  $k$ . The bases used are hierarchical, because higher order expansions are built from lower order expansions. Thus p-refinement implementation is straightforward and efficient. In addition, the bases are orthogonal polynomials in the Legendre inner product. This type of orthogonality offers better matrix conditioning and is more appropriate for using an explicit time step [49].

### 3.3 Elemental Operations

#### 3.3.1 Numerical Integration

The transformation relation between computational and physical space for a hexahedron shown in Fig. (3.2) is given by:

$$\begin{aligned} \mathbf{X}(\xi_1, \xi_2, \xi_3) = & \frac{(1 - \xi_1)(1 - \xi_2)(1 - \xi_3)}{8} \mathbf{X}_A + \frac{(1 - \xi_1)(1 + \xi_2)(1 - \xi_3)}{8} \mathbf{X}_B \\ & \frac{(1 + \xi_1)(1 - \xi_2)(1 - \xi_3)}{8} \mathbf{X}_C + \frac{(1 + \xi_1)(1 + \xi_2)(1 - \xi_3)}{8} \mathbf{X}_D \\ & \frac{(1 - \xi_1)(1 - \xi_2)(1 + \xi_3)}{8} \mathbf{X}_E + \frac{(1 - \xi_1)(1 + \xi_2)(1 + \xi_3)}{8} \mathbf{X}_F \\ & \frac{(1 + \xi_1)(1 - \xi_2)(1 + \xi_3)}{8} \mathbf{X}_G + \frac{(1 + \xi_1)(1 + \xi_2)(1 + \xi_3)}{8} \mathbf{X}_H, \end{aligned} \quad (3.13)$$

where  $\mathbf{X} = [x \quad y \quad z]^T$ . The differential change in the physical coordinates is:

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi_1} d\xi_1 + \frac{\partial x}{\partial \xi_2} d\xi_2 + \frac{\partial x}{\partial \xi_3} d\xi_3 \\ \frac{\partial y}{\partial \xi_1} d\xi_1 + \frac{\partial y}{\partial \xi_2} d\xi_2 + \frac{\partial y}{\partial \xi_3} d\xi_3 \\ \frac{\partial z}{\partial \xi_1} d\xi_1 + \frac{\partial z}{\partial \xi_2} d\xi_2 + \frac{\partial z}{\partial \xi_3} d\xi_3 \end{bmatrix} \implies \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = [J] \begin{bmatrix} d\xi_1 \\ d\xi_2 \\ d\xi_3 \end{bmatrix}, \quad (3.14)$$

where  $[J]$  is the volume Jacobian defined as:

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ \frac{\partial z}{\partial \xi_1} & \frac{\partial z}{\partial \xi_2} & \frac{\partial z}{\partial \xi_3} \end{bmatrix}. \quad (3.15)$$

The Jacobian of the transformation given by Eq. (3.16) must be positive, otherwise self-intersecting elements are detected which cannot be used for the computations.

$$|J| = \frac{\partial x}{\partial \xi_1} \left( \frac{\partial y}{\partial \xi_2} \frac{\partial z}{\partial \xi_3} - \frac{\partial z}{\partial \xi_2} \frac{\partial y}{\partial \xi_3} \right) - \frac{\partial x}{\partial \xi_2} \left( \frac{\partial y}{\partial \xi_1} \frac{\partial z}{\partial \xi_3} - \frac{\partial z}{\partial \xi_1} \frac{\partial y}{\partial \xi_3} \right) + \frac{\partial x}{\partial \xi_3} \left( \frac{\partial y}{\partial \xi_1} \frac{\partial z}{\partial \xi_2} - \frac{\partial z}{\partial \xi_1} \frac{\partial y}{\partial \xi_2} \right). \quad (3.16)$$

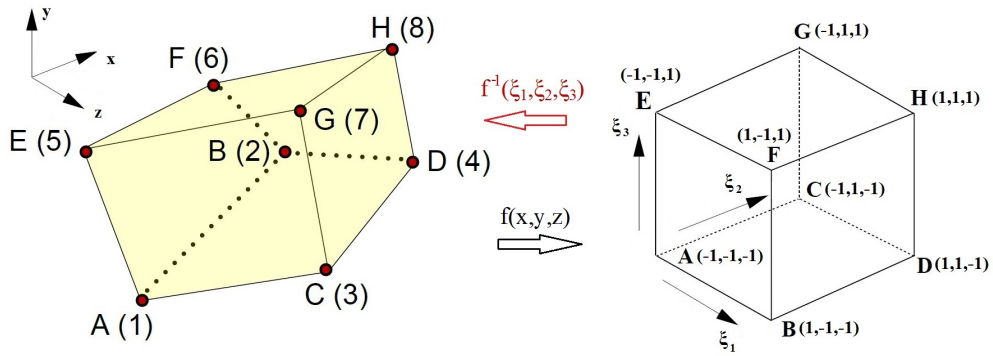


Figure 3.2: Numbering of global and local hexahedral nodes.

The bases are employed in the computational domain  $K$  where numerical integration can also be performed, therefore the following volume integral transformation is made from the physical space elements to the canonical elements of the computational domain:

$$\int_{\Omega_j} u(x_1, x_2, x_3) dx_1 dx_2 dx_3 = \int_{K_j} u(\xi_1, \xi_2, \xi_3) |J| d\xi_1 d\xi_2 d\xi_3. \quad (3.17)$$

The transformation for a surface integral is similar, taking into consideration that  $\partial\Omega_j$  is a specific local face of the element  $\Omega_j$  in the physical domain. For demonstration purposes and without loss of the generality it is assumed that  $\partial K_j$  is the corresponding local face where  $\xi_3 = -1$  of the element  $K_j$  in the computational domain:

$$\int_{\partial\Omega_j} u(x_1, x_2, x_3) dS = \int_{\partial K_j} u(\xi_1, \xi_2, -1) \left| \frac{\partial \mathbf{x}}{\partial \xi_1} \times \frac{\partial \mathbf{x}}{\partial \xi_2} \right| d\xi_1 d\xi_2, \quad (3.18)$$

where

$$\frac{\partial \mathbf{x}}{\partial \xi_i} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi_i} \\ \frac{\partial x_2}{\partial \xi_i} \\ \frac{\partial x_3}{\partial \xi_i} \end{bmatrix}. \quad (3.19)$$

The volume and surface integrals derived from the weak formulation are calculated using Gaussian quadrature which is exact for polynomials of degree  $k$  once  $\frac{3k+1}{2}$  quadrature points are used in Legendre type of integration. The value of the integrated function is required only at the specific quadrature points. The numerical integration for a volume integral is presented:

$$\int_{K_j} u(\xi_1, \xi_2, \xi_3) d\xi_1 d\xi_2 d\xi_3 = \sum_{i=1}^{Q_1} w_i \left\{ \sum_{j=1}^{Q_2} w_j \left\{ \sum_{k=1}^{Q_3} w_k u(\xi_{1i}, \xi_{2j}, \xi_{3k}) \right\} \right\}, \quad (3.20)$$

where  $w$  are the weights and  $Q_1, Q_2, Q_3$  is the number of quadrature points in each direction. A similar to Eq. (3.20) formula can be employed for the surface integrals. In the current work, Gauss-Legendre and Gauss-Lobatto-Legendre type of integration will be used. The difference is the location and the number of the quadrature points and the weights used. In Gauss-Legendre type the quadrature points are inside the interval, whereas in Gauss-Lobatto type the end-points,  $\xi = \pm 1$ , are quadrature points as well.

Gauss-Legendre type:

$$\xi_i = \xi_{i-1, Q+1}^{0,0}, \quad i = 1, \dots, Q \quad (3.21)$$

Gauss-Lobatto type:

$$\xi_i = \begin{cases} 1, & i = 1 \\ \xi_{i-2, Q-3}^{1,1}, & i = 2, \dots, Q-1 \\ -1, & i = Q \end{cases} \quad (3.22)$$

The number of the quadrature points taken depending on the polynomial order is the smallest integer value greater than or equal to:  $\frac{3k+1}{2}$  and  $\frac{3k+3}{2}$  for Legendre and Lobatto integration type respectively.

### 3.3.2 Differentiation

Differentiation is performed at the quadrature points as well, by using chain rule, since the shape functions are in terms of the canonical coordinates  $\xi_1, \xi_2, \xi_3$ . Therefore if the gradient of a scalar value  $q$  is required, the gradient of the basis  $\Phi$  (continuous or discontinuous) must be found  $\nabla\Phi$ . Applying chain rule, the derivatives can be found:

$$\nabla\Phi = \begin{bmatrix} \frac{\partial\Phi}{\partial x_1} \\ \frac{\partial\Phi}{\partial x_2} \\ \frac{\partial\Phi}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial\xi_1}{\partial x_1} \frac{\partial\Phi}{\partial\xi_1} + \frac{\partial\xi_2}{\partial x_1} \frac{\partial\Phi}{\partial\xi_2} + \frac{\partial\xi_3}{\partial x_1} \frac{\partial\Phi}{\partial\xi_3} \\ \frac{\partial\xi_1}{\partial x_2} \frac{\partial\Phi}{\partial\xi_1} + \frac{\partial\xi_2}{\partial x_2} \frac{\partial\Phi}{\partial\xi_2} + \frac{\partial\xi_3}{\partial x_2} \frac{\partial\Phi}{\partial\xi_3} \\ \frac{\partial\xi_1}{\partial x_3} \frac{\partial\Phi}{\partial\xi_1} + \frac{\partial\xi_2}{\partial x_3} \frac{\partial\Phi}{\partial\xi_2} + \frac{\partial\xi_3}{\partial x_3} \frac{\partial\Phi}{\partial\xi_3} \end{bmatrix}. \quad (3.23)$$

Inverting the Jacobian matrix in Eq. (3.15), the following derivatives are computed:

$$\begin{aligned} \frac{\partial\xi_1}{\partial x} &= \frac{1}{|J|} \left( \frac{\partial y}{\partial\xi_2} \frac{\partial z}{\partial\xi_3} - \frac{\partial y}{\partial\xi_3} \frac{\partial z}{\partial\xi_2} \right), & \frac{\partial\xi_1}{\partial y} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_2} \frac{\partial z}{\partial\xi_3} - \frac{\partial x}{\partial\xi_3} \frac{\partial z}{\partial\xi_2} \right), \\ \frac{\partial\xi_1}{\partial z} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_2} \frac{\partial y}{\partial\xi_3} - \frac{\partial x}{\partial\xi_3} \frac{\partial y}{\partial\xi_2} \right), \\ \\ \frac{\partial\xi_2}{\partial x} &= \frac{1}{|J|} \left( \frac{\partial y}{\partial\xi_1} \frac{\partial z}{\partial\xi_3} - \frac{\partial y}{\partial\xi_3} \frac{\partial z}{\partial\xi_1} \right), & \frac{\partial\xi_2}{\partial y} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_1} \frac{\partial z}{\partial\xi_3} - \frac{\partial x}{\partial\xi_3} \frac{\partial z}{\partial\xi_1} \right), \\ \frac{\partial\xi_2}{\partial z} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_1} \frac{\partial y}{\partial\xi_3} - \frac{\partial x}{\partial\xi_3} \frac{\partial y}{\partial\xi_1} \right), \\ \\ \frac{\partial\xi_3}{\partial x} &= \frac{1}{|J|} \left( \frac{\partial y}{\partial\xi_1} \frac{\partial z}{\partial\xi_2} - \frac{\partial y}{\partial\xi_2} \frac{\partial z}{\partial\xi_1} \right), & \frac{\partial\xi_3}{\partial y} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_1} \frac{\partial z}{\partial\xi_2} - \frac{\partial x}{\partial\xi_2} \frac{\partial z}{\partial\xi_1} \right), \\ \frac{\partial\xi_3}{\partial z} &= \frac{1}{|J|} \left( \frac{\partial x}{\partial\xi_1} \frac{\partial y}{\partial\xi_2} - \frac{\partial x}{\partial\xi_2} \frac{\partial y}{\partial\xi_1} \right). \end{aligned} \quad (3.24)$$

## 3.4 Global Operations

In this section, the global operations performed in continuous and discontinuous finite element approximation are described. These operations are the mapping process from local DOFs to global DOFs in order to create the global matrix and vector, modal face connectivity in order to apply continuity, and the Dirichlet boundary condition enforcement.



### 3.4.1 Mapping Process

#### dG formulation

In the dG approximation where no continuity is required, the process for assembling the global system is straightforward, since all the local DOFs are and global DOFs as well. The first  $nvb$  rows and columns of the global matrix contain the DOFs of the first element, the next  $nvb$  rows and columns are the DOFs of the second element, etc. However, in a specific row where the equations of a certain element are placed, there is also contribution from other elements, so columns of neighbor element DOFs are non zero, due to the numerical flux in the surface integrals.

#### cG formulation

In the cG approximation where  $C^0$  continuity is required, the appropriate mapping must be made in order to assemble the global matrix and vector from the local matrices and vectors respectively. For each element, a local index vector is created connecting the local DOFs with the global ones. The numbering in the local DOFs is the following. The vertex modes come first, then the  $k - 1$  edge modes for each edge follow, then the  $(k-1)^2$  face modes for each face and finally the  $(k-1)^3$  interior modes. The numbering between the local vertices, edges and faces is consistent with the computational canonical element, as it is shown in Fig. (3.3). The global numbering is accomplished with a similar procedure as follows. First, all the global vertex modes are assigned, defined by their global coordinates, then the edge modes are assigned, defined by the global edge numbering. Next, the global face modes are assigned, defined by the global face numbering, and finally the interior modes are independently assigned, by looping over all elements. Although the algorithm defining the unique matching between local and global edges or faces appears complicated, this type of global numbering is adopted because it yields to minimal bandwidth, therefore offering a better matrix conditioning.

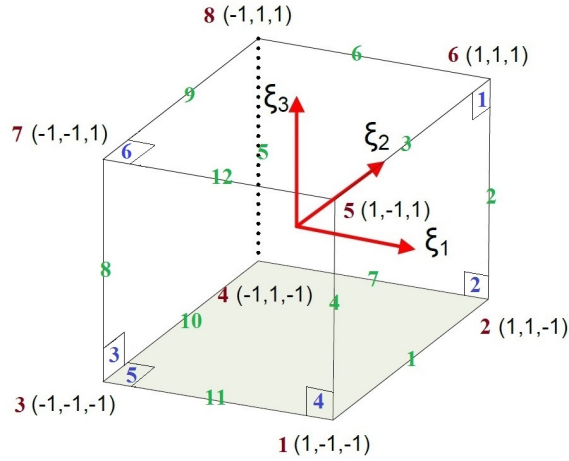


Figure 3.3: Local numbering of vertices (red), edges (green) and faces (blue) in the canonical element.

### 3.4.2 Modal Connectivity

#### dG formulation

For the numerical flux calculation in the dG approximation, information from elements neighboring the element under consideration is needed, therefore quadrature points connectivity must be established. For that purpose all possible orientations between two neighbor faces are examined. When calculating the value of a surface integral in a specific quadrature point, the corresponding quadrature point at the same location from the neighboring face must be taken. If the faces have different orientation, the quadrature points will not have the same index, thus proper modifications must be made. This is clearly shown in Fig. (3.4), where the  $\xi_1$  local axis is reversed, so for example the quadrature point (3,1) of the upper element should match with the quadrature point (1,1) of the lower element.

#### cG formulation

The situation is much more complicated in the cG approximation but the boundary/interior decomposition which was made, allows to match boundary modes of similar shape. For shape functions with polynomial order higher than two, where more than one edge modes exist, the local orientation of the element must also be taken into account, as it is shown in Fig. (3.5). Depending on the orientation of the local coordinate system, the sign of odd-ordered modes may need to be

reversed. This happens if the local system of the two neighboring elements is in opposite direction. In order to avoid re-ordering the edge modes, their numbering is based on their polynomial order, therefore the lowest edge mode number is the lowest polynomial order. Following the same procedure in global numbering, modes of similar polynomial order from two neighbor elements, have the same global number, thus mode matching is achieved. In a three dimensional formulation, the same procedure applies for the face orientation but now more orientation possibilities exist. One or more of the local coordinate systems may be transposed, reversed or a combination of those two, therefore a new mapping must be constructed transposing and changing the sign whenever needed in order that the boundary modes between the two elements match. Since the interior modes are different for each element, no re-ordering is needed for them.

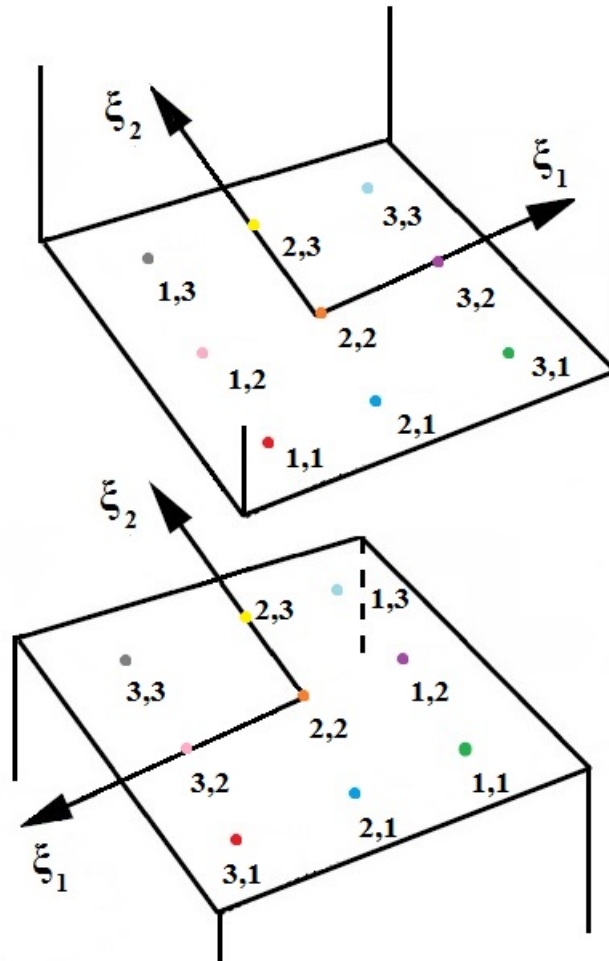


Figure 3.4: Quadrature points of the same color should match in the neighbor faces, although they have different index numbering due to reversed  $\xi_1$  axis.

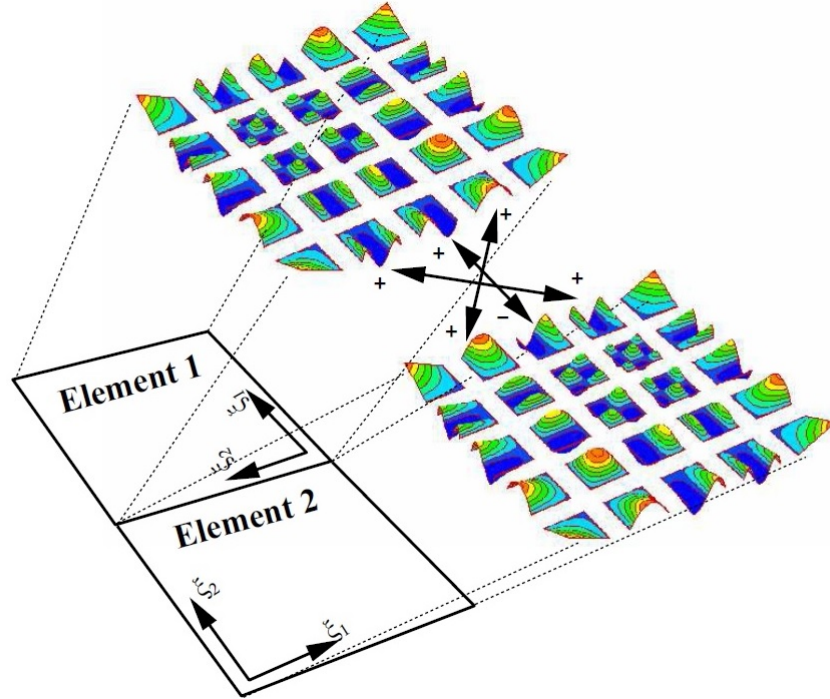


Figure 3.5:  $C^0$  expansion bases of order  $P_1 = P_2 = 4$  in two dimensions. Vertex and edge modes of similar shape should match. Expanding in  $3D$ , face modes should match as well. The figure was taken from [49].

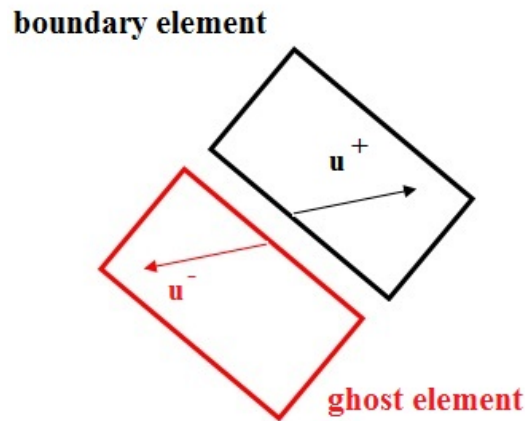


Figure 3.6: Ghost element for no slip wall boundary condition. The velocity in the ghost element is the opposite than the value in the element, thus resulting in zero velocity on the no slip boundary.

### 3.4.3 Dirichlet Boundary Condition Enforcement

In the dG discretization the boundary conditions are easily enforced using ghost elements. In the boundary faces, the value of the neighbor element is the boundary condition for the inflow and outflow. In order to satisfy the no slip condition for the stationary wall, the value of the velocity in the neighbor element is taken to be the opposite than the value calculated in the element, as it is shown in Fig. (3.6). Similar process is followed for the symmetry condition, where the velocity components normal to the boundary must vanish.

Although boundary condition implementation is easy in dG, this is not the case for the cG schemes. There are two different ways for enforcing the Dirichlet boundary conditions, either row and column elimination, or penalty approach. In the row and column elimination, the Dirichlet boundary modes have to be excluded from the total unknown DOFs. For that reason a new re-ordering mapping process is employed by checking all local DOFs. This process is strongly enforcing the boundary conditions and reducing the dimension of the global matrix but the implementation is rather difficult in comparison with the penalty approach. In the latter, for each Dirichlet mode, the diagonal entry of the global matrix is multiplied by a penalty parameter which is usually in the range  $10^6 - 10^8$ , whereas the corresponding DOF in the right hand side vector is replaced by the updated diagonal entry multiplied by the Dirichlet value. This way, all the other matrix entries in this specific row are very small relatively to the updated, thus this equations is actually giving the value specified for the Dirichlet mode. The disadvantages of this method is that it is inexact, ill-conditioned and the value of the parameter plays a significant role in the convergence of the GMRES algorithm used.

### 3.4.4 Parallelization

Parallelization is performed using the Message Passing Interface (MPI) prototype in C programming language and for the mesh decomposition *METIS* library [50] was used. In each processor (or rank), a specific amount of calculations and data storage are performed. The objective is to equally divide the workload among all ranks, therefore information must be sent through the partition boundaries. The

idea is to try to minimize the number of the messages sent and received due to the high start up cost. Furthermore, an element re-ordering algorithm was created in order to obtain an element numbering which follows the rank numbering. More specifically, the elements of the first rank are numbered first, then the numbering continues to the elements of the second rank and this procedure continues until the last rank.

Since the stencil of the dG methods is compact, messages containing information only from the immediate neighbor elements must be exchanged. Two ranks communicate with each other only if they share a global face in three dimensions, therefore each rank must send one message to the other with the solution vector needed for numerical fluxes calculations. A simplified two dimensional example is shown in Fig. (3.7) for four processors. Efforts to minimize the message cost were made and each rank sends a single message to each other, containing all the solution vectors from elements in their interface.

The volume of information exchange is larger in the cG formulation, because not only the face modes, but also the vertex and the edge modes belonging in the partition boundary and needed for superpositioning in the global stiffness matrix must be exchanged. As it is shown in the mesh decomposition Fig. (3.8) for a two dimensional grid, vertex modes communication is necessary, in case they belong to different ranks. Expanding to a three dimensional mesh, edge modes can belong to more than two ranks, thus communication between them is needed. In the cG method, the communication is more computationally expensive than in the dG method, due to more messages needed to be sent, although they are smaller. The assembly of the global matrix is getting even more complicated due to the PETSc [2] parallel linear system solver which was used. Each rank stores only a specific number of rows of the global matrix, but this way the global DOFs of the elements in this rank may belong to rows stored in another rank, making communication unavoidable.

Despite the difficulties encountered in global matrix assembly and element communication, the cG method was selected for the numerical solution of the Poisson equation because the discretized Poisson equation is solved only once for each time step and no auxiliary equations are needed for the pressure gradients.

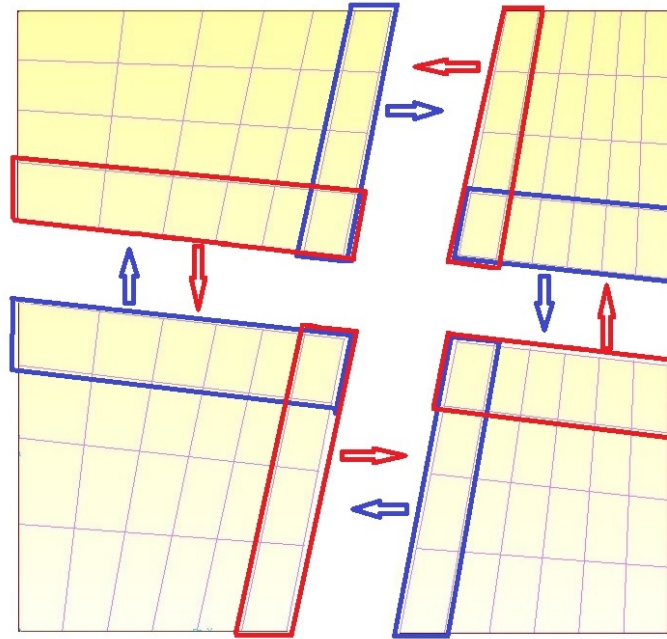


Figure 3.7: Communication pattern in the dG method for a two dimensional grid. Each process is sending two messages to its neighbor ranks, shown with the red and blue arrows.

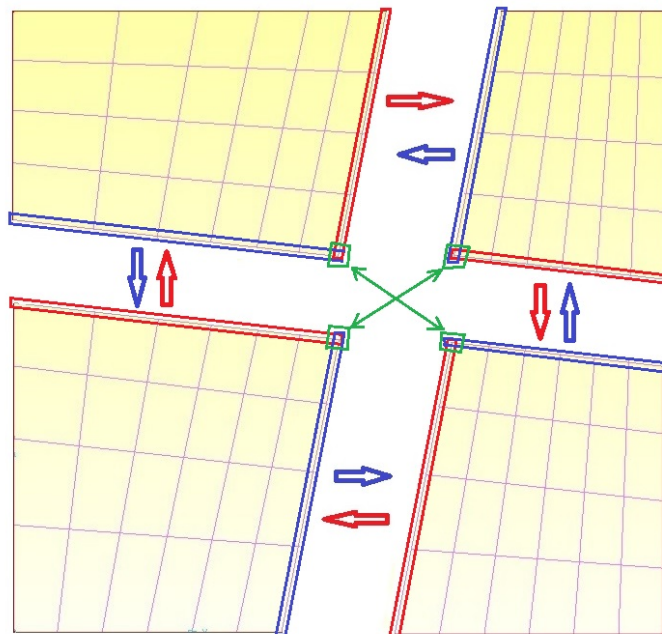


Figure 3.8: Communication pattern in the cG method for a two dimensional grid. Each process is sending three messages to its neighbor ranks, shown with the red, blue and green arrows.

# Chapter 4

## Discontinuous/Continuous Finite Element Discretization of the Governing Equations

### 4.1 Introduction

For the numerical implementation a Helmholtz splitting is adopted and the velocity vector field is considered as a sum of a divergence free vector field plus a rotation free vector field. The momentum equations for the rotation free component of the velocity vector (without the pressure gradient term) are discretized using dG formulation while cG formulation is used for the numerical solution of the pressure Poisson equation employed to enforce the incompressibility constraint [13], [35]. Then the velocity vector field is corrected with the pressure gradient term in order to ensure divergence free velocity field. Following this approach for momentum conservation, first advection-diffusion type equations are solved and then the pressure is corrected. This approach is less computationally intensive compared to other approaches for the Navier Stokes equations due to the fact that the weak formulation and the implementation is simpler compared with other unsplit methods.

Time marching is performed with explicit and diagonally implicit Runge-Kutta methods [78]. Upwind flux discretization is used for the non linear convective terms. However, these terms could also be linearized as suggested by Temam using the so called trilinear form [21], [87]. In the former case a non linear system



is solved using Jacobian-free Newton Krylov methods, whereas in the linearized trilinear form, iterative solution methods such as GMRES are used.

The following space discretization is element-wise, for that purpose let  $\Omega_j$  be an element:  $\Omega_j \in \Omega$ , where  $\Omega$  is the physical space and  $\partial\Omega_j$  are its local faces which can be either interior ( $\partial\Omega_{j,I}$ ) or boundary faces ( $\partial\Omega_{j,B}$ ). Boundary faces are divided into Neumann and Dirichlet type:  $\partial\Omega_{j,B} = \partial\Omega_{j,N} \cup \partial\Omega_{j,D}$ . The jumps  $[[\mathbf{u}]]$  and averages  $\{\mathbf{u}\}$  of  $\mathbf{u}$  between the local element and its neighbors are defined as:

$$\{\mathbf{u}\} = \frac{\mathbf{u}^+ + \mathbf{u}^-}{2}, \quad (4.1)$$

where  $\mathbf{u}^+$ ,  $\mathbf{u}^-$  denote  $\mathbf{u}$  at the local face of the elements  $j^+$  and its neighbors  $j^-$ ,

$$[[\mathbf{u} \cdot \mathbf{n}]] = \mathbf{u}^+ \cdot \mathbf{n}^+ + \mathbf{u}^- \cdot \mathbf{n}^-, \quad (4.2)$$

and  $\mathbf{n}^+$ ,  $\mathbf{n}^-$  denote the boundary outnormal vectors to the elements  $j^+$  and its neighbors  $j^-$ .

## 4.2 Projection Method

In the current projection scheme, Helmholtz splitting is used and the velocity vector is split into divergence free  $\mathbf{u}^{**}$  and rotation free  $\mathbf{u}^*$  components:

$$\mathbf{u} = \mathbf{u}^{**} + \mathbf{u}^*, \quad (4.3)$$

where

$$\nabla \times \mathbf{u}^* = 0, \quad (4.4)$$

$$\nabla \cdot \mathbf{u}^{**} = 0, \quad (4.5)$$

the momentum equation for the rotation free component of the velocity is:

$$\frac{\partial \mathbf{u}^*}{\partial t} + (\mathbf{u}^* \cdot \nabla) \mathbf{u}^* = \frac{1}{Re} \Delta \mathbf{u}^*, \quad (4.6)$$

and for the divergence free component is:

$$\frac{\partial \mathbf{u}^{**}}{\partial t} + (\mathbf{u}^{**} \cdot \nabla) \mathbf{u}^{**} = -\nabla p + \frac{1}{Re} \Delta \mathbf{u}^{**}. \quad (4.7)$$

Taking the divergence of Eq. (2.7) and splitting the velocity vector field into divergence free and rotation free components, the following Poisson equation for the pressure is obtained:

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^*, \quad (4.8)$$

where the viscous and inviscid terms vanished due to Eq. (2.6). Without loss of the generality, for deriving Eq. (4.8), Euler time integration has been used for approximating the time derivative in Eq. (4.6):  $\frac{\partial \mathbf{u}^*}{\partial t} = \frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t}$ . The divergence of the previous time-step term vanishes, since the velocity is divergence free at time-step  $n$ . In section 4.5 the corresponding coefficients in the momentum and Poisson equations are derived, depending on the order of accuracy in time. Equation (4.8) is a second-order elliptic equation which can be solved subject to the correct boundary conditions. For inflow and outflow the pressure is specified (Dirichlet boundary condition):

$$p = p_D, \quad \text{on } \partial\Omega_D. \quad (4.9)$$

For no-slip walls, Neumann-type boundary conditions can be derived by taking the dot product of the outward normal vector at the boundaries  $\mathbf{n}$  with the momentum equation (2.7):

$$\frac{\partial p}{\partial \mathbf{n}} = \left( -\frac{\partial \mathbf{u}}{\partial t} - (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{1}{Re} \Delta \mathbf{u} \right) \cdot \mathbf{n}, \quad \text{on } \partial\Omega_N. \quad (4.10)$$

The first two terms in the right hand side (rhs) of Eq. (4.10) vanish for fixed and no-slip boundaries. Using the identity for incompressible flow:

$$\Delta \mathbf{u} = -\nabla \times \nabla \times \mathbf{u}, \quad (4.11)$$

the Neumann-type boundary condition (4.10) for viscous flows becomes:

$$\frac{\partial p}{\partial \mathbf{n}} = \left( -\frac{1}{Re} \nabla \times \boldsymbol{\omega} \right) \cdot \mathbf{n}, \quad \text{on } \partial\Omega_N, \quad (4.12)$$

where  $\boldsymbol{\omega}$  is the vorticity vector field. If this non homogeneous Neumann-type condition is neglected and  $\frac{\partial p}{\partial \mathbf{n}}$  is set to zero, the scheme is prone to time-splitting errors [33]. Finally, the divergence free velocity at the new time step is:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p^{n+1}. \quad (4.13)$$

If Eq. (4.6) and (4.13) are added, they give the momentum Eq. (2.7).

### 4.3 dG discretization for the momentum equations

The velocity vector field is discretized using dG formulation, due to its better behavior when convection terms are dominant, for example in high Reynolds number flows. In addition, for the dG discretization, an upwind numerical flux can be defined [79],[57]. The velocity approximation for the component along the  $x_i$  direction, for each element  $\Omega_j$  is of the form:

$$u_i = \sum_{k=1}^{nb_d} \Phi_d^k \underline{u}_i^k, \quad \Omega_j \in \Omega, \quad i = 1, 2, 3, \quad (4.14)$$

where  $\Phi_d^k$  are the discontinuous bases, shown as  $\Phi_d$  from now on and  $\underline{u}_i^k$  are the  $nb_d$  coefficients of the  $x_i$  velocity component to be calculated. Starting from Eq. (4.6), integrating on elements and multiplying both parts by the weight function  $\Phi_d$ , the momentum equation in the  $x_i$ -direction is:

$$\frac{d}{dt} \int_{\Omega_j} \Phi_d u_i^* dV = - \int_{\Omega_j} u_k^* \frac{\partial u_i^*}{\partial x_k} \Phi_d dV + \frac{1}{Re} \int_{\Omega_j} \frac{\partial s_{ik}}{\partial x_k} \Phi_d dV, \quad (4.15)$$

where the velocity gradients are the tensor  $\bar{\bar{s}}$ :

$$\bar{\bar{s}} = \begin{bmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} + \frac{\partial w}{\partial y} + \frac{\partial w}{\partial z} \end{bmatrix}. \quad (4.16)$$

Applying Gauss' divergence theorem for the viscous and inviscid part in the Eq. (4.15), the weak formulation is derived:

$$\begin{aligned} \frac{d}{dt} \int_{\Omega_j} \Phi_d u_i^* dV = & + \int_{\Omega_j} u_i^* \frac{\partial u_k^* \Phi_d}{\partial x_k} dV - \int_{\partial\Omega_j} \hat{u}_i^* u_k^* n_k \Phi_d dS \\ & + \frac{1}{Re} \left( - \int_{\Omega_j} \frac{\partial \Phi_d}{\partial x_k} s_{ik} dV + \int_{\partial\Omega_j} \hat{s}_{ik} n_k \Phi_d dS \right). \end{aligned} \quad (4.17)$$

Proper numerical flux choice must be made for the convective  $\hat{u}_i^*$  and the diffusive  $\hat{s}_{ik}$  parts, in order to achieve stability. More discussion about numerical fluxes is made in subsections 4.3.1 and 4.3.2. Finally, Eq. (4.13) is integrated on elements and multiplied by the discontinuous weight function in order to obtain the divergence free velocity at time  $n + 1$ :

$$\int_{\Omega_j} \mathbf{u}^{n+1} \Phi_d dV = \int_{\Omega_j} \mathbf{u}^* \Phi_d dV - \Delta t \int_{\Omega_j} \nabla p^{n+1} \Phi_d dV. \quad (4.18)$$

Applying Gauss' theorem, the weak formulation for the  $\mathbf{u}^{n+1}$  :

$$\int_{\Omega_j} \mathbf{u}^{n+1} \Phi_d dV = \int_{\Omega_j} \mathbf{u}^* \Phi_d dV + \Delta t \left( \int_{\Omega_j} \nabla \Phi_d p^{n+1} dV - \int_{\partial\Omega_j} p^{n+1} \mathbf{n} \Phi_d dS \right). \quad (4.19)$$

The latter equation can be solved in an explicit manner, so it is robust and efficient.

The outline of the algorithm is:

- Find the velocity gradients  $\bar{s}$  from the auxiliary Eq. (4.35) by using the velocity from the previous time step  $\mathbf{u}^n$ .
- Solve implicitly or explicitly Eq. (4.17) to find the rotation free velocity component  $\mathbf{u}^*$ .
- Calculate the pressure at time step  $n + 1$  from Eq. (4.41).
- Project the velocities into the divergence free space using Eq. (4.19) and obtain the velocity at time step  $n + 1$ .
- Update solution and go back to the first step. If higher order of accuracy in time is used, more cycles are required before obtaining the solution at time step  $n + 1$ .

### 4.3.1 Inviscid Numerical Flux

One of the problems in the incompressible flow is the non-linear inviscid term treatment. In this subsection, different numerical fluxes for the inviscid part will

---

be examined. The advection term has three different forms, the convective, the conservative and the skew-symmetric form:

$$\text{convective} = \mathbf{u} \cdot \nabla \mathbf{u}, \quad (4.20)$$

$$\text{conservative} = \nabla \cdot \mathbf{u}\mathbf{u}, \quad (4.21)$$

$$\text{skew Symmetric} = \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{2} \nabla \cdot \mathbf{u}\mathbf{u}. \quad (4.22)$$

The previous terms are equivalent due to the incompressibility condition.

### Upwind Flux

The convective form is used in the INS and it was discretized using the upwind numerical flux [57] which is given by:

$$\hat{u}_i^* = \theta u_i^{*,+} + (1 - \theta) u_i^{*,-} \quad \text{where} \quad \begin{cases} \theta = 1, & \text{if } \mathbf{u}^+ \cdot \mathbf{n}^+ \geq 0 \\ \theta = -1, & \text{if } \mathbf{u}^- \cdot \mathbf{n}^- < 0 \end{cases} \quad (4.23)$$

Applying this to Eq.(4.17) the full discretized form using upwind flux for the inviscid term becomes:

$$\begin{aligned} \frac{d}{dt} \int_{\Omega_j} \Phi_d u_i^* dV &= + \int_{\Omega_j} u_i^* \frac{\partial u_k^* \Phi_d}{\partial x_k} dV - \int_{\partial\Omega_j} \theta u_i^{*,+} u_k^* n_k \Phi_d dS \\ &- \int_{\partial\Omega_j} (1 - \theta) u_i^{*,-} u_k^* n_k \Phi_d dS + \frac{1}{Re} \left( - \int_{\Omega_j} \frac{\partial \Phi_d}{\partial x_k} s_{ik} dV + \int_{\partial\Omega_j} \hat{s}_{ik} n_k \Phi_d dS \right). \end{aligned} \quad (4.24)$$

Because of the non-linear nature of the Eq. (4.24), Newton-like method has been used for solving the non linear system of the discretized equations as it is further explained in section 4.5.

### Trilinear Form

The skew symmetric form has been used for deriving the trilinear form. Starting from  $(\mathbf{u}^{n+1} \cdot \nabla) \mathbf{u}^{n+1} + \frac{1}{2} (\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^{n+1}$ , linearization is made using the Taylor expansion series for 2 variables,  $\mathbf{u}^{n+1}$ ,  $\nabla \mathbf{u}^{n+1}$  and  $\mathbf{u}^{n+1}$ ,  $\nabla \cdot \mathbf{u}^{n+1}$  respectively:

The first term becomes:

$$(\mathbf{u}^{n+1} \cdot \nabla) \mathbf{u}^{n+1} = \mathbf{u}^n \cdot \nabla \mathbf{u}^n + (\mathbf{u}^{n+1} - \mathbf{u}^n) \cdot \nabla \mathbf{u}^n + (\nabla \mathbf{u}^{n+1} - \nabla \mathbf{u}^n) \cdot \mathbf{u}^n, \quad (4.25)$$

$$(\mathbf{u}^{n+1} \cdot \nabla) \mathbf{u}^{n+1} = \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^n + \mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1} - \mathbf{u}^n \cdot \nabla \mathbf{u}^n. \quad (4.26)$$

And the second term is:

$$\begin{aligned} \frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^{n+1} &= \frac{1}{2}(\nabla \cdot \mathbf{u}^n) \mathbf{u}^n + \frac{1}{2}(\mathbf{u}^{n+1} - \mathbf{u}^n) \nabla \cdot \mathbf{u}^n \\ &+ \frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1} - \nabla \cdot \mathbf{u}^n) \mathbf{u}^n, \end{aligned} \quad (4.27)$$

$$\frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^{n+1} = \frac{1}{2}(\nabla \cdot \mathbf{u}^n) \mathbf{u}^{n+1} + \frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^n - \frac{1}{2}(\nabla \cdot \mathbf{u}^n) \mathbf{u}^n. \quad (4.28)$$

Adding Eq. (4.26) and (4.28) the final linearized form is:

$$\begin{aligned} (\mathbf{u}^{n+1} \cdot \nabla) \mathbf{u}^{n+1} + \frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^{n+1} &= \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^n + \frac{1}{2}(\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^n \\ &+ \mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1} + \frac{1}{2}(\nabla \cdot \mathbf{u}^n) \mathbf{u}^{n+1} + \mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{2}(\nabla \cdot \mathbf{u}^n) \mathbf{u}^n. \end{aligned} \quad (4.29)$$

or in a more compact notation:

$$\mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \frac{1}{2} \nabla \cdot \mathbf{u}^{n+1} \mathbf{u}^{n+1} = T(\mathbf{u}^{n+1}, \mathbf{u}^n) + T(\mathbf{u}^n, \mathbf{u}^{n+1}) + T(\mathbf{u}^n, \mathbf{u}^n), \quad (4.30)$$

where

$$T(\mathbf{w}, \mathbf{u}) = \mathbf{w} \cdot \nabla \mathbf{u} + \frac{1}{2}(\nabla \cdot \mathbf{w}) \mathbf{u}. \quad (4.31)$$

Following Di-Pietro, Ern [35] and Temam [87], the discretized form of  $T(\mathbf{w}, \mathbf{u})$  is  $t(\mathbf{w}, \mathbf{u}, \Phi_d)$ , where:

$$\begin{aligned} t(\mathbf{u}, \mathbf{w}, \Phi_d) &= \int_{\Omega_j} (\mathbf{u} \cdot \nabla) \mathbf{w} \Phi_d dV - \frac{1}{2} \int_{\partial\Omega_j / \partial\Omega_B} (\mathbf{w}^+ - \mathbf{w}^-) \Phi_d \mathbf{n} \cdot \{\mathbf{u}\} dS + \\ &\frac{1}{2} \int_{\Omega_j} (\nabla \cdot \mathbf{u}) \mathbf{w} \Phi_d dV - \frac{1}{4} \int_{\partial\Omega_j} \mathbf{n} \cdot (\mathbf{u}^+ - \mathbf{u}^-) \Phi_d \mathbf{w} dS. \end{aligned} \quad (4.32)$$

As a result, the inviscid part in Eq.(4.17) becomes:

$$\text{Convective Term} = t(\mathbf{u}^{n+1}, \mathbf{u}^n, \Phi_d) + t(\mathbf{u}^n, \mathbf{u}^{n+1}, \Phi_d) + t(\mathbf{u}^n, \mathbf{u}^n, \Phi_d). \quad (4.33)$$

The expanded form of Eq.(4.17) using the trilinear form is :

$$\begin{aligned}
 & \frac{d}{dt} \int_{\Omega_j} \Phi_d u_i^* dV = - \int_{\Omega_j} (\mathbf{u}^n \cdot \nabla) \mathbf{u}^{n+1} \Phi_d dV + \\
 & + \frac{1}{2} \int_{\partial\Omega_j/\partial\Omega_B} (\mathbf{u}^{n+1,+} - \mathbf{u}^{n+1,-}) \Phi_d \mathbf{n} \cdot \{\mathbf{u}^n\} dS - \frac{1}{2} \int_{\Omega_j} (\nabla \cdot \mathbf{u}^n) \mathbf{u}^{n+1} \Phi_d dV + \\
 & + \frac{1}{4} \int_{\partial\Omega_j} \mathbf{n} \cdot (\mathbf{u}^{n,+} - \mathbf{u}^{n,-}) \Phi_d \mathbf{u}^{n+1} dS - \int_{\Omega_j} (\mathbf{u}^{n+1} \cdot \nabla) \mathbf{u}^n \Phi_d dV + \\
 & + \frac{1}{2} \int_{\partial\Omega_j/\partial\Omega_B} (\mathbf{u}^{n,+} - \mathbf{u}^{n,-}) \Phi_d \mathbf{n} \cdot \{\mathbf{u}^{n+1}\} dS - \frac{1}{2} \int_{\Omega_j} (\nabla \cdot \mathbf{u}^{n+1}) \mathbf{u}^n \Phi_d dV \\
 & + \frac{1}{4} \int_{\partial\Omega_j} \mathbf{n} \cdot (\mathbf{u}^{n+1,+} - \mathbf{u}^{n+1,-}) \Phi_d \mathbf{u}^n dS - \int_{\Omega_j} (\mathbf{u}^n \cdot \nabla) \mathbf{u}^n \Phi_d dV \\
 & + \frac{1}{2} \int_{\partial\Omega_j/\partial\Omega_B} (\mathbf{u}^{n,+} - \mathbf{u}^{n,-}) \Phi_d \mathbf{n} \cdot \{\mathbf{u}^n\} dS - \frac{1}{2} \int_{\Omega_j} (\nabla \cdot \mathbf{u}^n) \mathbf{u}^n \Phi_d dV \\
 & + \frac{1}{4} \int_{\partial\Omega_j} \mathbf{n} \cdot (\mathbf{u}^{n,+} - \mathbf{u}^{n,-}) \Phi_d \mathbf{w} dS \\
 & + \frac{1}{Re} \left( - \int_{\Omega_j} \nabla \Phi_d \cdot \bar{\bar{s}} dV + \int_{\partial\Omega_j} \hat{\hat{s}} \cdot \mathbf{n} \Phi_d dS \right).
 \end{aligned} \tag{4.34}$$

Once the advection term is linearized, linear solver techniques can be used due to the linearity of the rest of the terms in the discretized momentum equation.

### 4.3.2 Viscous Numerical Flux

The viscous term is linear, since it is the Laplacian of each velocity component. As it is already mentioned, Bassi, Rebay [5] and later Cockburn, Shu [25] introduced auxiliary variables in the dG formulation, for second order differential equations such as diffusion terms. Since the velocity gradients are defined as new independent variables, additional equations are needed so as the system to be closed. The BR1 numerical flux [6] is an example of the latter case, whereas in the BR2 numerical flux [8], lifting operators are needed, making the method even more computationally expensive but reducing the stencil at the same time. However, there is no need for auxiliary equations in the so called Interior Penalty Method (IPM) [1] and only the value of arbitrary parameters must be specified.

#### BR1 Scheme

The velocity gradients are assigned from Eq. (4.16):  $\bar{\bar{s}} = \nabla \mathbf{u}$ . Integrating element wise, multiplying by the  $\Phi_d$  weight function and applying Gauss theorem, the

discretized equation for the auxiliary variables is derived:

$$\int_{\Omega_j} s_{ik} \Phi_d dV = - \int_{\Omega_j} \frac{\partial \Phi_d}{\partial x_k} u_i dV - \int_{\partial \Omega_j} \hat{u}_i \Phi_d n_k dS. \quad (4.35)$$

In the beginning of time step  $n+1$ , Eq. (4.35) is solved from  $\mathbf{u}^n$  in order to calculate  $\bar{s}$  which is needed in Eq. (4.17). In the BR1 scheme, the numerical fluxes for the velocity and the velocity gradients are defined as:

$$\hat{\mathbf{u}} = \{\mathbf{u}\}, \quad (4.36)$$

$$\hat{\hat{s}} = \{\bar{s}\}. \quad (4.37)$$

Using this formulation,  $9 \times DOFs$  unknowns are calculated explicitly for each element before solving the split form of the momentum equations.

## 4.4 cG discretization for the pressure Poisson equation

The local discontinuous Galerkin (LDG) has been used for solving convection-diffusion problems by Cockburn and Shu [25] but this way four equivalent equations are solved instead of solving only the Poisson equation. Another way of calculating the pressure is the HDG method for linear convection-diffusion equations [65] which is difficult to parallelize and memory demanding. Therefore, the space discretization for the pressure is taken to be continuous because this is an effective way of reducing the coupling between the momentum and the mass conservation equations [34]. Moreover, the cG framework is effective and efficient, since no auxiliary equations are needed for the pressure gradients and the Poisson equation is solved implicitly one time per time step. The pressure approximation for each element  $\Omega_j$  is of the form:

$$p = \sum_{k=1}^{nb_c} \Phi_c^k \mathbf{p}^k, \quad \Omega_j \in \Omega, \quad (4.38)$$

where  $\Phi_c^k$  are the  $k$  continuous bases and  $\mathbf{p}^k$  the unknown coefficients to be found. Since  $\Phi_c^k$  are in terms of  $\xi_1$ ,  $\xi_2$  and  $\xi_3$  from the computational space, proper transformations to the physical space  $\Omega$  must be made. For the rest of this work,



the subscript  $k$  will be skipped, so the notation of the continuous basis vector will be  $\Phi_c$ . Starting from Eq. (4.8), integrating element-wise and multiplying by the weight function  $\Phi_c$ :

$$\int_{\Omega_j} \nabla \cdot \nabla p^{n+1} \Phi_c dV = \frac{1}{\Delta t} \int_{\Omega_j} \nabla \cdot \mathbf{u}^* \Phi_c dV. \quad (4.39)$$

In order to derive the weak form of the Poisson equation, Gauss' theorem is applied on both rhs and left hand side (lhs) of Eq. (4.39):

$$-\int_{\Omega_j} \nabla \Phi_c \cdot \nabla p^{n+1} dV + \int_{\partial\Omega_j} \nabla p^{n+1} \cdot \mathbf{n} \Phi_c dS = \frac{1}{\Delta t} \left( -\int_{\Omega_j} \nabla \Phi_c \cdot \mathbf{u}^* dV + \int_{\partial\Omega_j} \hat{\mathbf{u}}^* \cdot \mathbf{n} \Phi_c dS \right), \quad (4.40)$$

where the surface integral from the rhs is transformed into the Neumann boundary condition, since the pressure is continuous and there is no effect from the interior faces:

$$\int_{\Omega_j} \nabla \Phi_c \cdot \nabla p^{n+1} dV = \frac{1}{\Delta t} \left( \int_{\Omega_j} \nabla \Phi_c \cdot \mathbf{u}^* dV - \int_{\partial\Omega_j} \hat{\mathbf{u}}^* \cdot \mathbf{n} \Phi_c dS \right) + \int_{\partial\Omega_{j,N}} \nabla p^{n+1} \cdot \mathbf{n} \Phi_c dS. \quad (4.41)$$

A numerical flux for the velocity approximation must be derived, since it is discontinuous. Following [34] and [13], the approximated velocity in the local faces is:

$$\hat{\mathbf{u}}^* = \{\mathbf{u}^*\}, \quad (4.42)$$

#### 4.4.1 Poisson solver Validation

Preliminary results for Poisson equation using cG discretization are shown in this subsection. In the first case, a 2D Poisson problem is solved, having as source term:  $f(x, y) = 6xy(1 - y) - 2x^3$  in a rectangular plate domain of unit length and boundary conditions  $u(x, 0) = 0$ ,  $u(x, 1) = 0$ ,  $u(0, y) = 0$  and  $u(1, y) = y(1 - y)$ . The exact solution is  $u(x, y) = y(1 - y)x^3$  and comparison with the computed is shown in Fig. (4.1). A grid of  $100 \times 100$  equally spaced elements was created. In the second case, a 3D Poisson problem is examined, having as source term  $f(x, y, z) = 14\sin(\pi x)\sin(2\pi y)\sin(3\pi z)$  in a cube with unit length and boundary

conditions  $u(1, y, z) = u(0, y, z) = u(x, 1, z) = u(x, 0, z) = u(x, y, 1) = u(x, y, 0) = 0$ . The analytical solution is given by  $u(x, y, z) = \sin(\pi x)\sin(2\pi y)\sin(3\pi z)$  and it is compared with the computed in Fig. (4.2). A mesh of  $40 \times 40 \times 40$  equally spaced elements was created for this case.

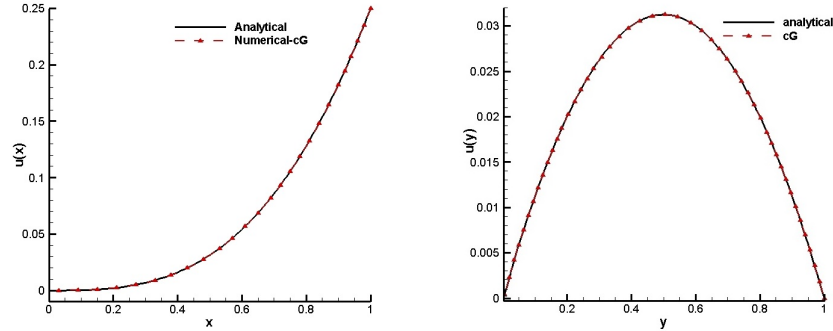


Figure 4.1: Comparison of the exact and numerical solutions (cG) for the plate, cut at  $y=0.5$  (left) and cut at  $x=0.5$  (right).

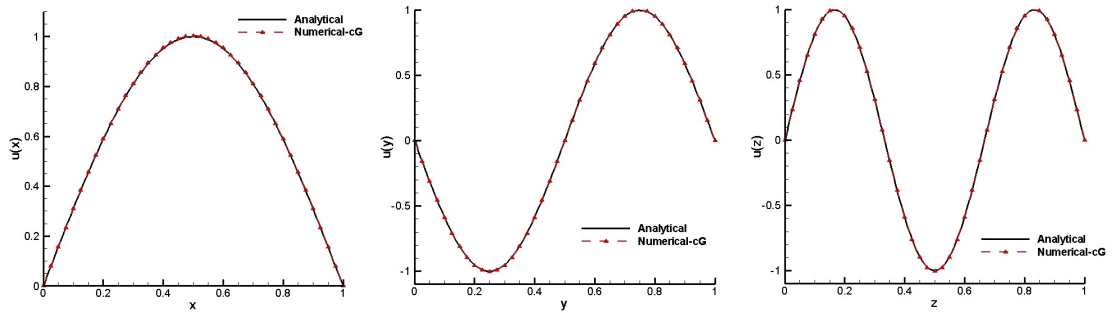


Figure 4.2: Comparison of the exact and numerical solution (cG) for the cube, cut at  $y=0.75$  and  $z=0.5$  (left), cut at  $x=0.5$  and  $z=0.5$  (middle) and cut at  $x=0.5$  and  $y=0.25$  (right).

Furthermore cG discretization was used for the numerical solution of the system of equations for linear elasticity. The method is described in detail in Appendix A and verification for steady and time dependent problems with exact solution is presented.

## 4.5 Time Marching Scheme

In order to advance in time the incompressible flow equations, high order explicit and implicit schemes can be used. Runge-Kutta (*RK*) methods for both schemes

demonstrate accuracy and strong stability properties [44]. Increasing the order of polynomial expansion  $P$ , which means the order of spatial accuracy is  $P + 1$ , the maximum  $CFL$  number in the explicit time marching methods reduces approximately to  $CFL \sim \frac{1}{P^2}$  [94]. Therefore for inviscid flow computations, with relatively large cells and relatively low order of accuracy  $P$ , explicit time advancement in the dG discretization can be used.

However, in viscous flow simulations, the near wall region where steep flow gradients are encountered must be discretized with small size elements or larger size elements but using higher order approximation [28]. In both cases, the resulting small time step for the explicit time marching scheme makes the scheme inefficient in both steady state and time-accurate computations. The large time step restriction imposed by the explicit scheme, necessitates the use of high order implicit scheme in order to allow large enough time steps and small temporal errors when time-accurate problems are solved. Furthermore, it has been demonstrated that the efficiency of the high-order dG scheme for both steady and unsteady flow computations, can be greatly enhanced with the use of implicit time marching methods [28], [4], [20], [12].

Therefore, in the present work explicit  $RK2$  and Diagonally Implicit Runge-Kutta ( $DIRK$ ) schemes have been developed, suitable for large scale parallel computations. The latter is based on the Jacobian free Newton-Krylov method [52] with the use of a suitable preconditioning applied on the linear systems produced by the Newton's method linearization. Each linear system between the Newton's iterations is solved by the Generalized Minimum Residuals method (GMRES) provided by the Portable Extensible Toolkit for Scientific computation (PETSc) [2].

#### 4.5.1 Explicit Algorithm

First order backward Euler and second order Runge-Kutta explicit methods have been used in the inviscid simulations. Multistage Runge-Kutta methods  $RK(s, p)$ , where  $s$  denotes the number of the stages and  $p$  its order, are a class of numerical methods for computing numerical solutions to the initial value problems. They are applied on time dependent form of Eq. (4.17):

$$\frac{\partial \mathbf{c}}{\partial t} = \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}, t), \quad (4.43)$$

using as initial condition:  $\mathbf{c}(t_0) = \mathbf{c}^0$ . Where  $\mathcal{M}$  is the mass matrix defined as:

$$\mathcal{M} = \int_{\Omega_j} \Phi_d \Phi_d^\top dV, \quad (4.44)$$

$\mathbf{c}$  is the solution vector of the rotation free velocity coefficients:

$$\mathbf{c} = [\underline{\mathbf{u}}^1 \quad \underline{\mathbf{u}}^2 \quad \dots \quad \underline{\mathbf{u}}^{nvd} \quad \underline{\mathbf{v}}^1 \quad \underline{\mathbf{v}}^2 \quad \dots \quad \underline{\mathbf{v}}^{nvd} \quad \underline{\mathbf{w}}^1 \quad \underline{\mathbf{w}}^2 \quad \dots \quad \underline{\mathbf{w}}^{nvd}]^\top, \quad (4.45)$$

and  $\mathcal{R}(\mathbf{c}^{n+1})$  is the rhs of the equation (4.17) at time  $n + 1$ .

Explicit *RK* methods compute a solution  $\mathbf{c}^{n+1}$  at time  $t^{n+1} = t_0 + (n+1)\Delta t$ , by only using the known solution in the previous time step  $n$  at time  $t^n = t_0 + (n)\Delta t$  and the time step  $\Delta t$  is restricted by the *CFL* condition. Therefore, in Table (4.1)  $s$  stages are recursively build:

$$\begin{aligned} \mathbf{c}^1 &= \mathbf{c}^n \\ \mathbf{c}^2 &= \mathbf{c}^n + \alpha_{21} \Delta t \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}_0) \\ &\vdots \\ \mathbf{c}^s &= \mathbf{c}^n + \sum_{j=1}^s \alpha_{sj} \Delta t \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}^j). \end{aligned} \quad (4.46)$$

0					
$c_2$	$a_{21}$				
$c_3$	$a_{31}$	$a_{32}$			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
$c_s$	$a_{s1}$	$a_{s2}$	$\dots$	$a_{s,s-1}$	
	$b_1$	$b_2$	$\dots$	$b_{s-1}$	$b_s$

Table 4.1: The Butcher tableau for the explicit Runge-Kutta method.

The *RK*(2, 2) is applied to the differential-algebraic system of INS and Poisson equations by following [78]:

$$\mathbf{c}^1 = \mathbf{c}^n + \alpha_{21} \Delta t \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}^n), \quad (4.47)$$

$$\mathbf{p}^1 = \frac{1}{c_2 \Delta t} \mathcal{L}^{-1} \mathcal{D}(\mathbf{c}^1), \quad (4.48)$$

$$\mathbf{c}_f^1 = \mathbf{c}^1 - c_2 \Delta t \mathcal{M}^{-1} \mathcal{G}(\mathbf{p}^1). \quad (4.49)$$

Where  $\mathbf{c}^{1f}$  is the divergence free velocity coefficients after the first stage and  $\mathcal{D}$ ,  $\mathcal{G}$  are the rhs of Eq. (4.41), (4.19) respectively. The second *RK* stage is:

$$\mathbf{c}^2 = \mathbf{c}_f^1 + b_2 \Delta t \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}^n), \quad (4.50)$$

$$\mathbf{p}^{n+1} = \frac{1}{\Delta t} \mathcal{L}^{-1} \mathcal{D}(\mathbf{c}^2), \quad (4.51)$$

$$\mathbf{c}^{n+1} = \mathbf{c}^2 - \Delta t \mathcal{M}^{-1} \mathcal{G}(\mathbf{p}^{n+1}), \quad (4.52)$$

finding the divergence free velocity coefficients  $\mathbf{c}^{n+1}$  at time step  $n + 1$ , whereas the  $\alpha$ ,  $b$ ,  $c$  values are taken from Table (4.2).

$c_2$	$\alpha_{21}$	$\frac{1}{2}$	$\frac{1}{2}$
	$b_1$ $b_2$		0    1

Table 4.2: The Butcher tableau for the explicit *RK*(2, 2) method.

## 4.5.2 Implicit Algorithm

Implicit time marching methods ensure high-order time accuracy and lead to substantially increased time steps thus avoiding the severe time step limitations of explicit schemes imposed by the dG discretization. The Diagonally Implicit Runge-Kutta (*DIRK*<sub>2</sub>) and the first order backward Euler scheme have been used and implemented providing second and first order time accuracy in the solution.

The *DIRK* methods have an advantage over other implicit methods *RK*. At each *RK* stage  $s$ , only the solution of this stage  $\mathbf{c}^s$  is solved implicitly, as it is shown in Table (4.3), without resulting into extremely large non-linear system (having as unknown variables the solution of all stages) which must be solved in order to obtain the solution of the next stage. In addition, because of the same diagonal coefficients  $\alpha_{ii}$ , the linear systems which arise by the Newton's method at each *RK* stage, have the same matrix (linear operator), so each time is changing

only the right hand side of the system. However, the *DIRK*<sub>2</sub> scheme developed in the current work, needs only one implicit step.

First, for the sake of simplicity, the Implicit Euler scheme is described. Applying the Euler scheme, the discrete form of flow equations (4.24) become:

$$\mathcal{M} \frac{\mathbf{c}^{n+1} - \mathbf{c}^n}{\Delta t} = \mathcal{R}(\mathbf{c}^{n+1}) \quad (4.53)$$

To solve the above system of equations, Newton's method linearizes the non-linear function,

$$\mathcal{F}(\mathbf{c}^{n+1}) = \mathcal{M} \frac{\mathbf{c}^{n+1} - \mathbf{c}^n}{\Delta t} - \mathcal{R}(\mathbf{c}^{n+1}) \quad (4.54)$$

and  $k$  Newton's steps are produced:

$$\frac{\partial \mathcal{F}(\mathbf{c}^k)}{\partial \mathbf{c}^k} \delta^k \mathbf{c} = -\mathcal{F}(\mathbf{c}^k) \Rightarrow \overbrace{\left[ \frac{\mathcal{M}}{\Delta t} - \frac{\partial \mathcal{R}(\mathbf{c})}{\partial \mathbf{c}} \right]_k}^{J_N} \delta^k \mathbf{c} = \mathcal{R}(\mathbf{c}^k) - \frac{\mathcal{M}}{\Delta t} (\mathbf{c}^k - \mathbf{c}^n) \quad (4.55)$$

$$\mathbf{c}^{k+1} = \mathbf{c}^k + \delta^k \mathbf{c} \quad (4.56)$$

In order to get the solution of the next time  $n + 1$ , the linear system (4.55) must be solved, updating at each Newton's step the solution  $\mathbf{c}$  until the  $L_2$  norm of the non-linear function  $\mathcal{F}$  (Eq. 4.54) becomes smaller than a constant tolerance which in this case has been set to  $10^{-8}$ .

The linear system of each Newton's step is not solved exactly but iteratively until the relative residual

$$\frac{\left. \frac{\partial \mathcal{F}(\mathbf{c})}{\partial \mathbf{c}} \right|_k \delta^k \mathbf{c} + \mathcal{F}(\mathbf{c}^k)}{\mathcal{F}(\mathbf{c}^k)} \leq \eta = 10^{-3} \quad (4.57)$$

becomes smaller than a parameter  $\eta \in [0, 1)$  called forcing term. In this inexact Newton's method and for all problems the forcing term is set  $\eta = 10^{-3}$  so that the quadratic convergence of the Newton's method is retained.

To solve each linear system at each Newton's step the iterative generalized minimal residual method (GMRES) method in conjunction with a preconditioner, in order to achieve better convergence of the iterative procedure, is used. PETSc

offers a GMRES solver for parallel solution of linear systems and preconditioning techniques for convergence acceleration.

In order to apply to each linear system a preconditioning technique, the preconditioning matrix  $P$  must be constructed and used to accelerate the convergence of the GMRES. For all problems, a right preconditioner can be applied which is defined as:

$$J_N P^{-1} \mathbf{y} = \mathbf{b} \quad \mathbf{y} = P^{-1} \mathbf{x} \quad (4.58)$$

where  $\mathbf{x}$  is the a vector containing the unknown degrees of freedom and  $\mathbf{b}$  the residual.

Despite the fact that the operator  $J_N$  (Jacobian matrix) of each linear system (4.55) is sparse following the pattern of Fig. (4.4), for large or even moderate problems it contains extremely large number of entries which are the derivatives  $\frac{\partial \mathcal{R}(\mathbf{c})}{\partial \mathbf{c}}$ . For that reason, the storage amount required for this operator is quite large, especially when high order of dG approximation is used e.g  $P^3$  or  $P^4$ . In this case the operator is not possible to be stored in the memory of the massive distributed systems. Moreover, the analytic or even the numerical differentiation for all the aforementioned derivatives is computational expensive. Some graph coloring techniques [26, 42] can exploit the known sparsity of the Jacobian  $J_N$  and accelerate the numerical computation of the jacobian entries but the computational cost still remains high. Note that a  $P^1$  expansion in a hexahedral element has 8 DOFs and 27 DOFs for a  $P^2$  expansion. For the system of flow equations, we have  $8 \times 3 = 24$  (81 for  $P^2$ ) DOFs per element. Therefore, for three dimensional meshes of real large-scale problems which contains very large number of elements  $\approx 10^7$ , the number of required derivatives increases very much and the cost in terms of memory and CPU time is rather large. The Jacobian-free idea [52, 18] overcomes this difficulty by using numerical differentiation when it is needed as it is shown next and makes possible the use of the inexact Newton's approach for very large-scale problems with millions of degrees of freedom in the domain.

The Jacobian-free Newton Krylov method (*JFNK*) [52] as a combination of a Newton's like method with a Krylov subspace iterative method which in this case is the GMRES for the non-exact solution of the linear systems, computes numerically only the derivatives needed for the Jacobian-vector product used by

the Krylov method without storing the entire Jacobian (matrix-free method). The Jacobian-free algorithm is summarized in Fig. (4.3).

The matrix-vector products required by the GMRES solver with the Jacobian-free method are computed numerically using first-order finite differences:

$$\frac{\partial \mathcal{F}(\mathbf{c})}{\partial \mathbf{c}} P^{-1} \mathbf{v}_n \approx \frac{\mathcal{F}(\mathbf{c} + \varepsilon P^{-1} \mathbf{v}_n) - \mathcal{F}(\mathbf{c})}{\varepsilon} \quad (4.59)$$

where  $\varepsilon$  is a small scalar parameter. A suitable choice of  $\varepsilon$  is important because very small values can lead to round off errors whereas large values can introduce truncation error. A proper formula suggested by Brown and Saad [18] for the definition of  $\varepsilon$  is:

$$\varepsilon = \frac{b}{\|\mathbf{v}_n\|_2} \max [|\mathbf{c}^T \mathbf{v}_n|, \text{typ}(\mathbf{c}^T | \mathbf{v}_n|)] \text{sign}(\mathbf{c}^T \mathbf{v}_n) \quad (4.60)$$

where  $\text{typ}(\mathbf{c}^T | \mathbf{v}_n|)$  is a user supplied typical size of  $\mathbf{c}$ ,  $\mathbf{v}_n$  denotes a unit vector, and  $b$  is small number proportional to the square root of the machine round off error, and it is taken equal to  $b = 10^{-8}$ . Even though in the Jacobian-free method, the sparse matrix (Jacobian) of the linear system is not stored, however, the preconditioning matrix  $P$  must be formed and stored. A suitable preconditioning matrix is often the same Jacobian matrix  $J_N$  and a fit preconditioner method tries to find a approximate inverse  $P^{-1}$  of this preconditioning matrix so that when it is multiplied by the Jacobian matrix in Eq. (4.58), the final operator of the linear system will be well conditioned and the convergence of GMRES faster. The choice of the most appropriate preconditioner is not a straightforward task due to the fact that different numerical problems require different preconditioning techniques in order to achieve a reasonable convergence rate of the GMRES solver with the less effort in terms of storage and numerical operations. As long as a the matrix-free method is used, the preconditioning matrix which must be constructed and stored, has to contain as less as possible entries so that the advantages of the matrix free method for little use of memory and fast computation can be exploited. Especially in the case of a dG method with high-order of accuracy ( $P^2$  or higher) where many DOFs are introduced, the preconditioning matrix becomes extremely large, and its construction is expensive in terms of computational effort and storage. For that reason, in order to take full advantage of matrix-free method, only the diagonal

---



blocks of the preconditioning matrix which are the diagonal blocks of the Jacobian  $J_N$  are computed and stored as it is shown in Fig. (4.4).

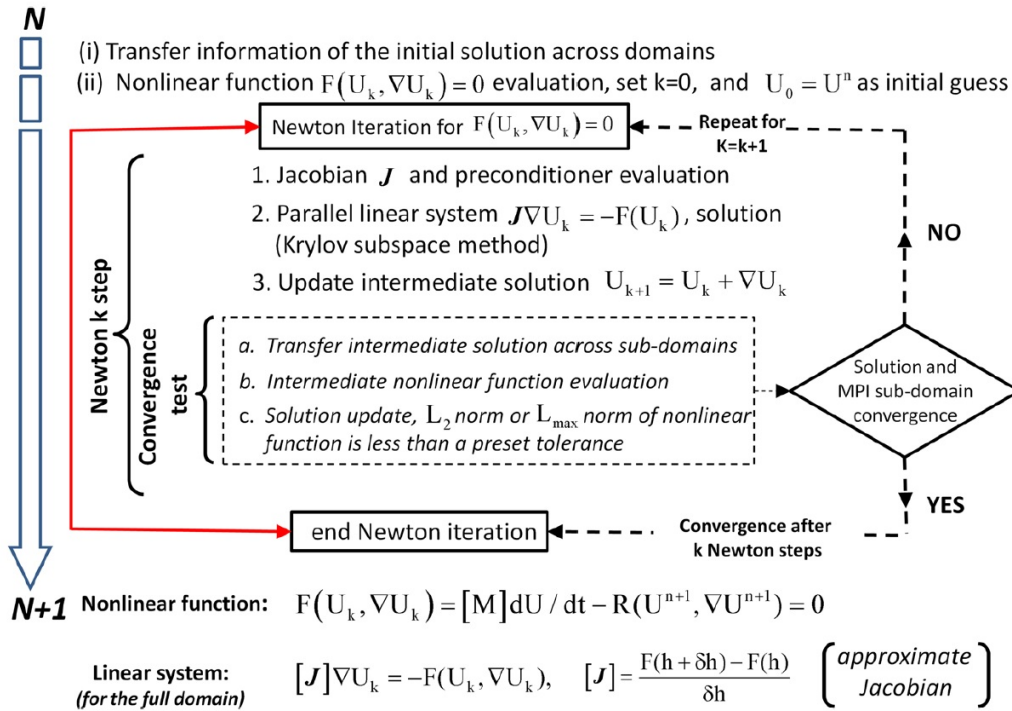


Figure 4.3: Jacobian-free Newton Iteration for implicit time marching.

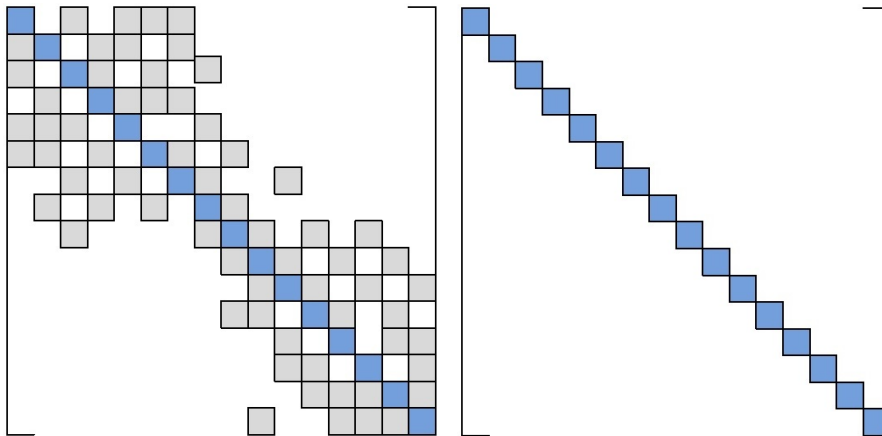


Figure 4.4: Jacobian matrix (left) and block Jacobi preconditioner (right) patterns.

Therefore, the most suitable preconditioner which inverts approximately the Jacobian  $J_N$  or inverts exactly the constructed preconditioning matrix  $P$  is the block-Jacobi preconditioner defined as:

$$P^{-1} = \begin{bmatrix} B_{1,1}^{-1} & 0 & 0 & \cdots & 0 \\ 0 & B_{2,2}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & B_{3,3}^{-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & B_{N_e, N_e}^{-1} \end{bmatrix}, J_N = \begin{bmatrix} B_{1,1} & B_{1,2} & B_{1,3} & \cdots & B_{1, N_e} \\ B_{2,1} & B_{2,2} & B_{2,3} & \cdots & B_{2, N_e} \\ B_{3,1} & B_{3,2} & B_{3,3} & \cdots & B_{3, N_e} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{N_e, 1} & B_{N_e, 2} & B_{N_e, 3} & \cdots & B_{N_e, N_e} \end{bmatrix} \quad (4.61)$$

where  $B$  denotes the non-zero blocks.

Apart from the block Jacobi preconditioner which is easy invertible and mainly used in this work, one can use especially for small problems other preconditioner such as ILU(0), ILU(1), SOR computing more blocks per block line for the construction of the preconditioning matrix. This can lead even better convergence of the GMRES method but the cost of the preconditioning matrix computation and the inversion is much larger than that of the block Jacobi. Moreover, as it is shown in [12], the block Jacobi preconditioner is quite efficient compared with other preconditioning techniques such as ILU.

In order to make the implicit scheme more efficient, a new preconditioning matrix has been constructed and stored every 50 time steps. Since between 50 time steps the solution of a time accurate or steady problem doesn't vary much, the preconditioning matrix which depends on the solution at each Newton iteration  $k$  and time step, behaves quite well.

The implicit  $RK$  methods can be constructed in a way that the kinetic energy in incompressible flows is conserved, making the scheme free of numerical diffusion [78]. This can become very useful in DNS or LES simulation and it also enables stability for even coarse meshes and large time steps [78]. The energy-conservation condition can only be satisfied in implicit  $RK$  methods, and more specifically, Gauss methods have the highest possible order for a given number of stages. The same procedure applied to the first order backward Euler scheme, is performed for the  $DIRK_2$ . Hence, the linearization with Newton's method of the two stages of  $DIRK_2$  method is written as:

$$\left[ \frac{\mathcal{M}}{\Delta t} - \alpha_{11} \frac{\partial \mathcal{R}(\mathbf{c})}{\partial \mathbf{c}} \Big|_k \right] \delta^k \mathbf{c} = \alpha_{11} \mathcal{R}(\mathbf{c}^k) - \frac{\mathcal{M}}{\Delta t} (\mathbf{c}^k - \mathbf{c}^n) \quad (4.62)$$

$$\mathbf{c}^{k+1} = \mathbf{c}^k + \delta^k \mathbf{c},$$

$$\mathbf{p}^1 = \frac{1}{c_1 \Delta t} \mathcal{L}^{-1} \mathcal{D}(\mathbf{c}^1), \quad (4.63)$$

$$\mathbf{c}_f^1 = \mathbf{c}^1 - c_1 \Delta t \mathcal{M}^{-1} \mathcal{G}(\mathbf{p}^1). \quad (4.64)$$

The second *RK* stage is explicit:

$$\mathbf{c}^2 = \mathbf{c}_f^1 + b_1 \Delta t \mathcal{M}^{-1} \mathcal{R}(\mathbf{c}^n), \quad (4.65)$$

$$\mathbf{p}^{n+1} = \frac{1}{\Delta t} \mathcal{L}^{-1} \mathcal{D}(\mathbf{c}^2), \quad (4.66)$$

$$\mathbf{c}^{n+1} = \mathbf{c}^2 - \Delta t \mathcal{M}^{-1} \mathcal{G}(\mathbf{p}^{n+1}), \quad (4.67)$$

finding the divergence free velocity coefficients  $\mathbf{c}^{n+1}$  at time step  $n + 1$ , whereas the  $\alpha$ ,  $b$ ,  $c$  values are taken from Table (4.4).

$c_1$	$a_{11}$				
$c_2$	$a_{21}$	$a_{22}$			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
$c_s$	$a_{s1}$	$a_{s2}$	$\cdots$	$a_{s,s-1}$	$a_{s,s}$
	$b_1$	$b_2$	$\cdots$	$b_{s-1}$	$b_s$

Table 4.3: The Butcher tableau for the implicit Runge-Kutta method.

$c_1$	$\alpha_{11}$	$\frac{1}{2}$	$\frac{1}{2}$
	$b_1$		1

Table 4.4: The Butcher tableau for the *DIRK*<sub>2</sub> method.

The Jacobian Newton Krylov method in conjunction with the block Jacobi preconditioner for the implicit scheme were provided by the PETSc package. The user must supply a routine which computes the non-linear function (4.54) and the parameters of the KSP (linear system solver) and SNES (non-linear system solver) objects. The systems are solved in parallel, by storing at each processor's local memory only the part of the entire data that are involved in the procedure of the implicit scheme. Although numerical differentiation with first order finite

differences is provided from PETSc in order to construct in parallel the preconditioning matrix, the differentiation is carried out by user defined routine for memory efficiency.

### 4.5.3 The GMRES Method

The method of Generalized Minimum Residuals (GMRES) used in this work solves iteratively any non-singular large sparse linear system:

$$[A]\underline{\mathbf{x}} = \underline{\mathbf{b}}, \quad (4.68)$$

where  $[A] \in \mathcal{R}^n$  and  $\underline{\mathbf{x}}, \underline{\mathbf{b}} \in \mathcal{R}^n$ . Starting with an initial solution  $\underline{\mathbf{x}}_0$ , the basic idea of GMRES method is to minimize the residual of the initial solution  $\underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - [A]\underline{\mathbf{x}}_0$  projecting at each  $m$  iteration the solution into the  $m$  dimension Krylov subspace as:

$$\mathcal{K}_n = \text{span}_{\underline{\mathbf{r}}_0}, [A]\underline{\mathbf{r}}_0, [A]^2\underline{\mathbf{r}}_0, \dots, [A]^{n-1}\underline{\mathbf{r}}_0. \quad (4.69)$$

The initial solution  $\underline{\mathbf{x}}_0$  is converging to the exact solution in no more than  $n$  iterations, which is the dimension of the solution vector. Due to the large requirements in storage and computational time which is needed to built up the Krylov subspace with dimension  $n$ , the method is restarted after a fixed and small number of iterations  $m$ . The resulted method is called restarted GMRES( $m$ ). In the restarted GMRES( $m$ ) method, the approximate solution  $x_m$  is used as initial solution in the next restart for the construction of the  $m$  dimension subspace. This approximate solution has been computed by minimizing the residual of the system 4.69 using the  $m$  dimension Krylov subspace after  $m$  iterations. The method is terminated when the residual becomes smaller than a user specified value or using the relative residual relation 4.57 for the inexact Newton's method. In the present work applications, the dimension of Krylov subspace has been set to  $m_{max} = 50$  or  $m_{max} = 30$ . For larger Krylov subspace dimension, the method becomes computationally expensive, whereas for much smaller dimension, slow solution convergence is noticed.

The vector  $\underline{\mathbf{r}}_0, [A]\underline{\mathbf{r}}_0, [A]^2\underline{\mathbf{r}}_0, \dots, [A]^{n-1}\underline{\mathbf{r}}_0$  of the Krylov subspace is almost linearly independent and thus the Arnoldi method with Gram-Schmidt process in order to produce a sequence of orthogonal vectors is used to project the solution.

The main feature of the GMRES method is that requires only products of the form  $[A]\underline{y}$ . Consequently, no direct access or manipulation of the  $[A]$  entries is required. For the GMRES and the Jacobian free methods, PETSc [2] toolkit has been used.

# Chapter 5

## Numerical Examples

### 5.1 Introduction

The method was verified for several test cases and good agreement with analytical results was obtained even for polynomial order  $p = 1$  (second order space accurate solution). In the inviscid cases where steady state solution is achieved after a short number of iterations, explicit scheme was used, whereas implicit algorithm was used in the viscous flow simulations. Validation was carried out by comparing with experiments.

The grid generator used to construct the meshes is Pointwise [68]. Then the grid was imported to Gambit [39], in order to obtain the Neutral format, because a translator for Neutral files was created in the code. The capability for trivial automatic structured mesh is also enabled for square and cube domains. The post-processing is performed using Tecplot software [86].

### 5.2 Inviscid Flow

#### 5.2.1 Inviscid Cylinder

The first case examined is the inviscid cylinder. Due to symmetry, mesh was created around half of the cylinder by using one hundred eighty elements along the perimeter, for representing accurately the geometry. Comparison with analytical solutions given by Eq. (2.11)-(2.14) is demonstrated in Fig. (5.1) for the pressure coefficient (left) and the velocity field (right).

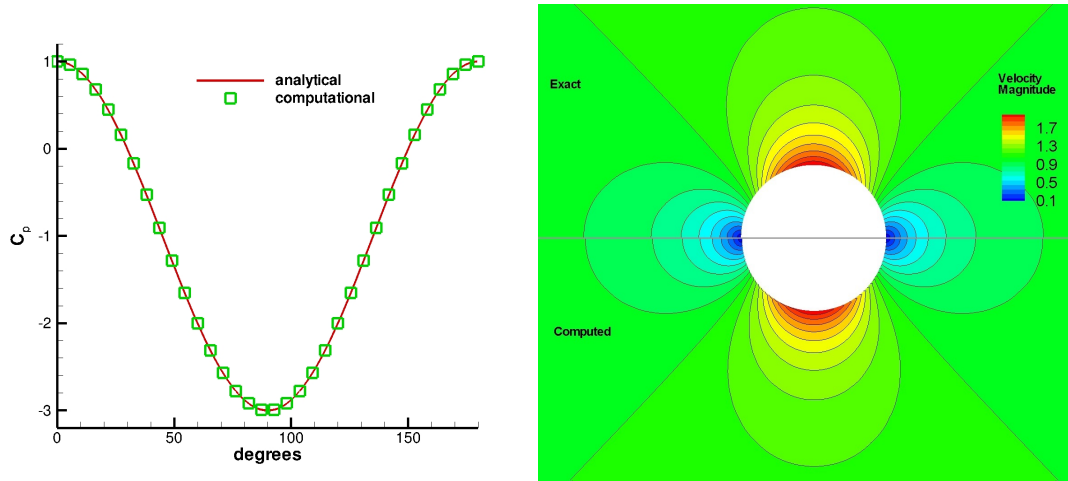


Figure 5.1: Inviscid cylinder simulation. Pressure coefficient (left) and velocity magnitude field comparison between analytical and computational solution (right).

### 5.2.2 Inviscid NACA 0012

Inviscid flow around NACA 0012 was obtained next. In high Reynolds number viscous terms become negligible, so the flow is almost inviscid. A C-type mesh was created having 200 elements along the airfoil where the leading and trailing edge areas were refined in order to capture the geometry and the velocity and pressure gradients. The pressure was specified using Dirichlet type boundary conditions in the inflow and outflow. The velocity boundary conditions were taken to be of Dirichlet type in the inflow and of homogeneous Neumann type in the outflow. Comparison with experimental pressure coefficient ( $C_p$ ) and the overall structure of the computed velocity field are shown in Fig. (5.2). It appears that a perfectly symmetric solution free of artificial surface entropy layers is obtained. Although there is good agreement between the experimental and computational data, the experimental pressure coefficient is slightly higher than the computed for a small region. This is happening because of the Gregory and O'Reilly [45] data that have been used for validation, in comparison with Ladson [54] data which give slightly lower pressure coefficient. Lift and Drag coefficients converge to their final value after 5 iterations, due to the inviscid nature of the simulation.

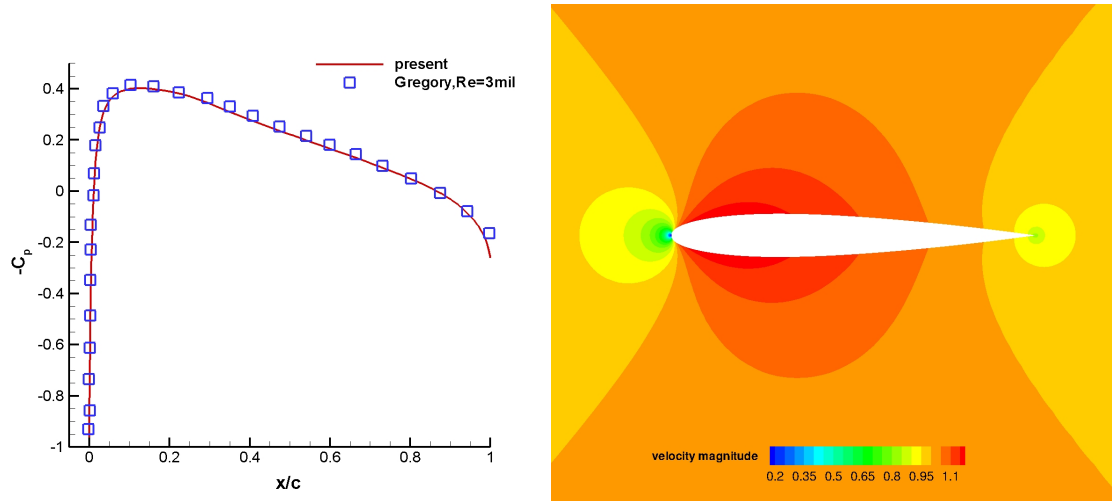


Figure 5.2: Inviscid NACA 0012 simulation. Pressure coefficient at angle of attack 0 degrees (left) and velocity magnitude field (right).

## 5.3 Viscous Flow

In this section, several problems were examined in order to prove the accuracy of the code. The implicit algorithm was used for the viscous flow simulations, enabling this way large time steps and achieving faster convergence.

### 5.3.1 Kovasznay Flow

Kovasznay flow was used for viscous flow verification of the code in a two dimensional domain at Reynolds number 40. The initial condition for the velocity and pressure was taken to be zero, whereas Dirichlet boundary conditions were weakly enforced from the analytical solution given by the Eq. (2.15)-(2.18). In Fig. (5.3) comparison with the analytical solution is shown and in Fig. (5.4) the velocity and pressure fields are demonstrated. For that simulation a uniform mesh of  $100 \times 100$  elements was used at the domain  $(-0.5, 1.0) \times (-0.5, 1.5)$ . For the spatial accuracy test of the dG discretization, a square domain  $(-0.5, 1.5) \times (0.0, 2.0)$  was used with 50, 80, 100 and 120 equally spaced elements in each dimension. In Fig. (5.5) the convergence rates are shown, where the slope between 3 successive points is almost the same.



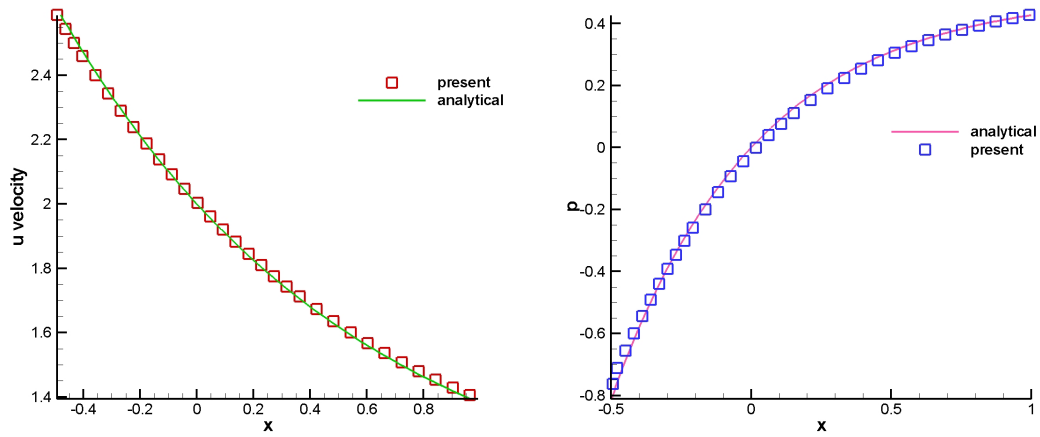


Figure 5.3: Kovaszny flow  $u$  velocity comparison (left) and pressure comparison (right) along  $y=0.5$  at Reynolds number 40. The velocity was discretized with discontinuous and the pressure with continuous Galerkin method.

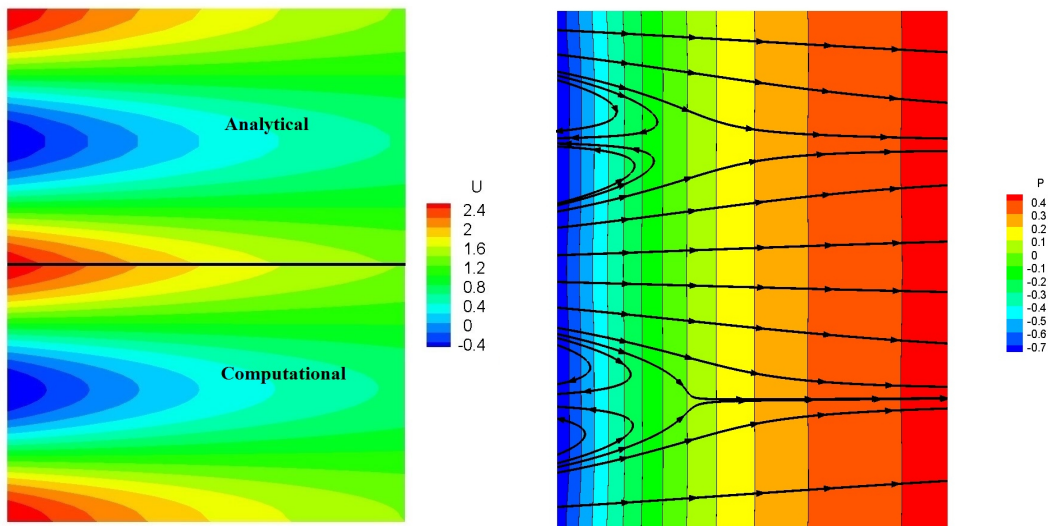


Figure 5.4:  $U$  velocity contours for analytical and computational solutions (left) and pressure field combined with velocity streamlines (right) for Kovaszny flow at Reynolds number 40.

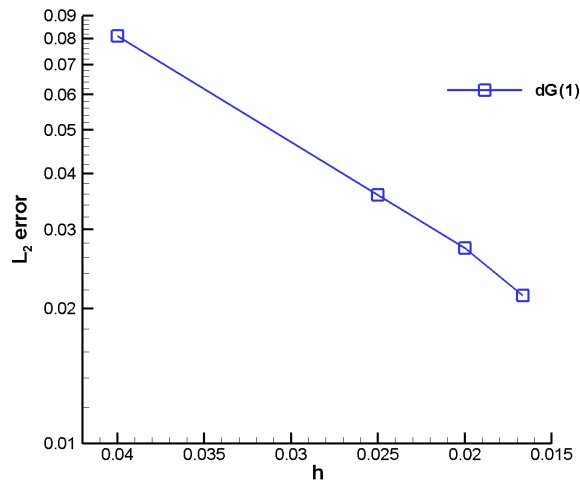


Figure 5.5: Spatial accuracy for Kovasznay flow.

### 5.3.2 2D Lid Driven Cavity

The steady-state lid driven cavity problem at Reynolds number 100 was examined next. A square box of unity length where no-slip conditions are imposed on all walls, except for the upper wall, where  $u = 1$  and  $v = 0$ . A grid of  $100 \times 100$  elements and first length 0.001 near the walls was used for this simulation. Homogeneous Neumann type boundary conditions are used for the Poisson equation for all the faces. The lid driven cavity is a benchmark test for the Incompressible flows since there is no ambiguity about the boundary conditions used for the pressure Poisson equation. Comparison of the velocity profiles with the solution given by Sahin and Owens [77] is shown in Fig. (5.6) whereas in Fig. (5.7) the computed velocity magnitude (left) and the  $v$  velocity component combined with streamlines (right) are shown. It appears that the upwind flux discretization of the convective term can better capture the  $v$  velocity component change along  $x$  direction, than the trilinear form, as it is shown in Fig. (5.6). The steady state solution is obtained after  $T=9$  (non dimensional time).

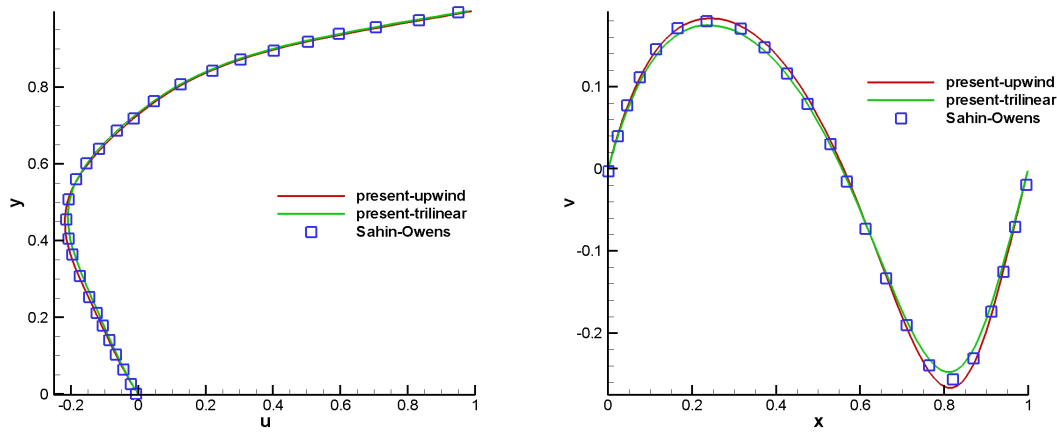


Figure 5.6: Velocity profiles comparison between Sahin-Owens [77] and present work, either using trilinear or upwind form at Reynolds number 100. Profiles of the  $u$  velocity component along  $x=0.5$  (left) and the  $v$  velocity component along  $y=0.5$  (right) are shown.

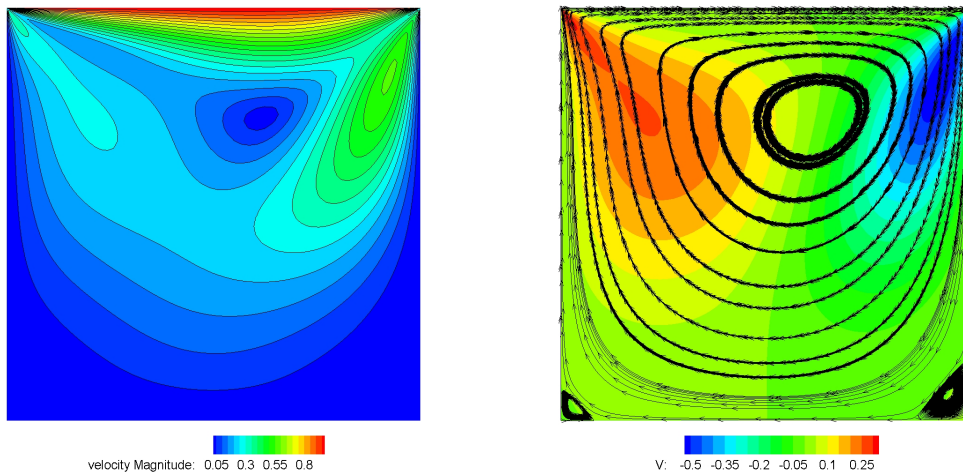


Figure 5.7: Velocity magnitude contours in 2D Cavity (left) and  $v$  velocity field combined with velocity streamlines (right).

### 5.3.3 Flow over a flat plate

The flow over a semi-infinite flat plate and comparison with Blasius solution for Reynolds number 1000 was performed next. Within the boundary layer approximately 30 elements were used in order to capture the velocity gradient even with  $p1$  (second order accurate in space) numerical approximation. In the inflow the velocity and the pressure were specified using Dirichlet type boundary conditions

and in the outlet homogeneous Neumann type boundary conditions were used for the velocity field and pressure. The velocity field in the fully developed area and the velocity profile in comparison with Blasius solution are demonstrated in Fig. (5.8). The Blasius numerical solution is given by solving the following differential equation:

$$f''' + \frac{1}{2}ff'' = 0, \quad (5.1)$$

$$f = f' = 0 \quad \text{at} \quad \eta = 0, \quad (5.2)$$

$$f' \rightarrow 1 \quad \text{as} \quad \eta \rightarrow \infty, \quad (5.3)$$

where  $\eta$  is the non dimension similarity variable,  $\psi$  is the stream function,  $\nu$  the kinematic viscosity and  $f(\eta)$  is the dimensionless function given by:

$$\eta = \sqrt{\frac{U_\infty}{\nu x}} y, \quad (5.4)$$

$$\psi = \sqrt{\nu U_\infty x} f(\eta). \quad (5.5)$$

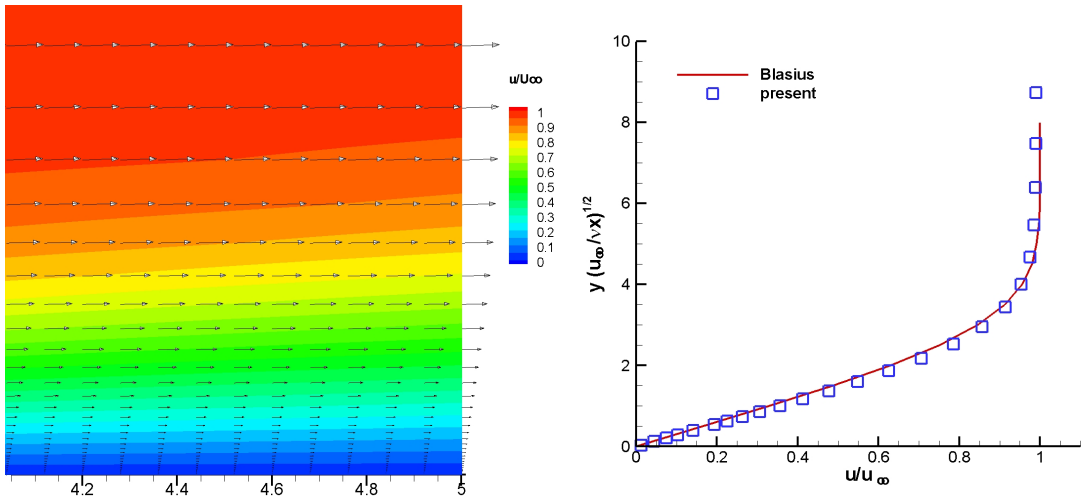


Figure 5.8: Velocity field (left) and velocity profile comparison with Blasius solution (right) for flow over a flat plate at Reynolds number 1000.

### 5.3.4 Viscous Cylinder

Steady state flow around viscous cylinder is also examined at Reynolds number 40. In the created mesh, three hundred sixty elements were used around the cylinder and equal size length was used for the normal to the wall mesh lines.

After the vortex is detached from the cylinder, a symmetric smooth solution is obtained. In higher Reynolds numbers, the vortices are getting detached creating the von Karman vortex street. Flow visualization taken from experiment [91] (up) and computed with the current scheme pressure contours combined with velocity streamlines (down) for a cylinder are shown in Fig. 5.9.

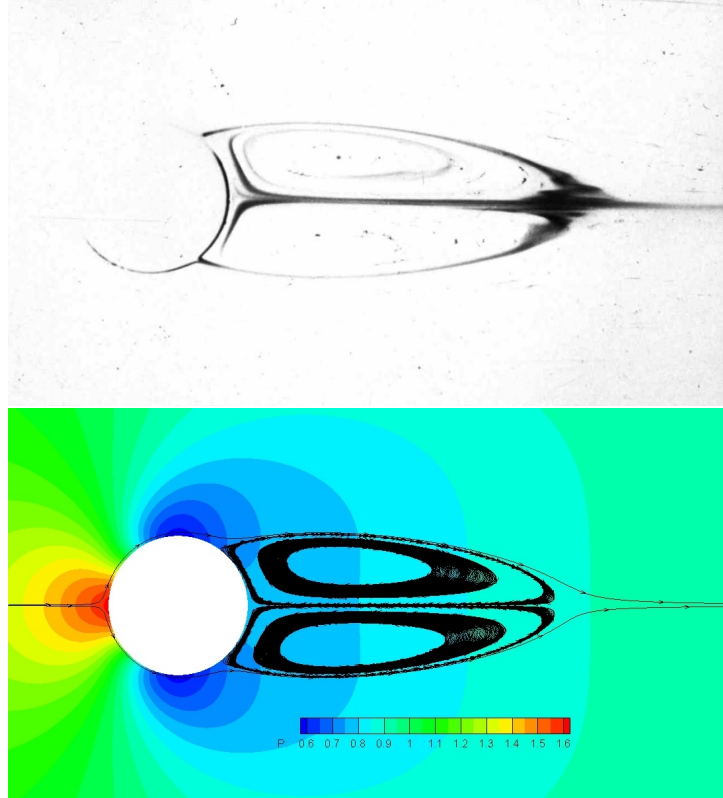


Figure 5.9: Viscous cylinder flow simulation. Flow visualization taken from experiment [91] (up) and pressure contours combined with velocity streamlines from the computational solution (down).

### 5.3.5 Viscous NACA 0012

The next test case is flow around NACA 0012 at Reynolds number 100 with angle of attack 5 degrees. The grid and the boundary conditions are the same as in subsection 5.2.2. The velocity and pressure contours are shown in Fig. (5.10). In order to validate the pressure coefficient, the very same viscous airfoil simulation was performed using Gnuid code [14], which is based on libMesh open source finite element library [51]. The reason for this was to avoid comparison with experimental data where compressibility effects are present and the flow

is turbulent due to the high Reynolds number that was measured during the experiment. The  $C_p$  comparison between Botti, Di-Pietro scheme [13] and present work is shown in Fig. (5.11).

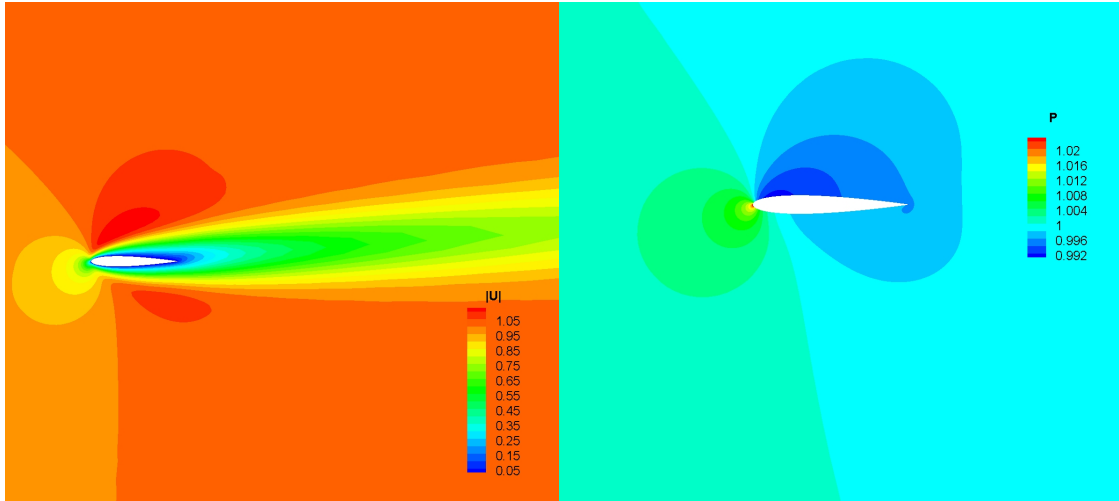


Figure 5.10: Velocity magnitude contour (left) and pressure contour (right) for NACA 0012 at Reynolds number 100 with angle of attack 5 degrees are shown.

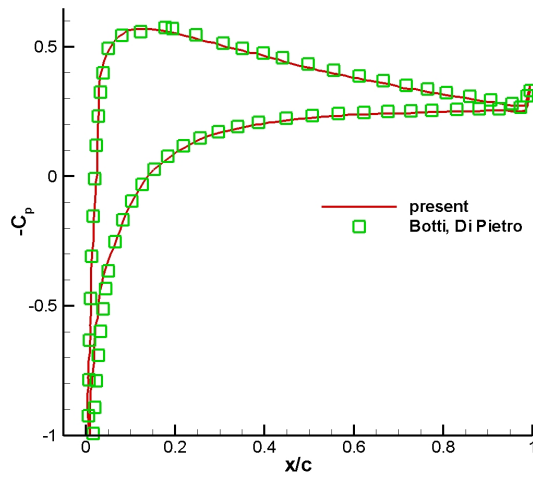


Figure 5.11: Pressure coefficient for NACA 0012 at Reynolds number 100 with angle of attack 5 degrees.

### 5.3.6 3D Lid Driven Cavity

Finally, the three dimensional steady-state lid driven cavity is examined at Reynolds number 1000. A cube of unity length with 50 elements equally spaced in each direction was created. No-slip conditions are imposed on all walls, apart from the upper moving wall, where  $u = 1$ ,  $v = 0$  and  $w = 0$ . Similar to the 2D case, homogeneous Neumann type boundary conditions are used for the Poisson equation. Comparison of the velocity profiles with the 3D solution given by Botti, Di-Pietro [13] is shown in Fig. (5.12) whereas in Fig. (5.13) the velocity streamlines are shown in the 2D plane (left) and in the 3D domain (right). It is worth pointing out that in the velocity profiles computed by Botti and Di-Pietro,  $p = 2$  approximation, thus 3rd order of spacial accuracy was used, whereas in the present work simulation, 2nd order of spacial accuracy was used for both the velocity and pressure variables. This can be seen in Fig. (5.12) where Botti's solution capture better the maximum value curve.

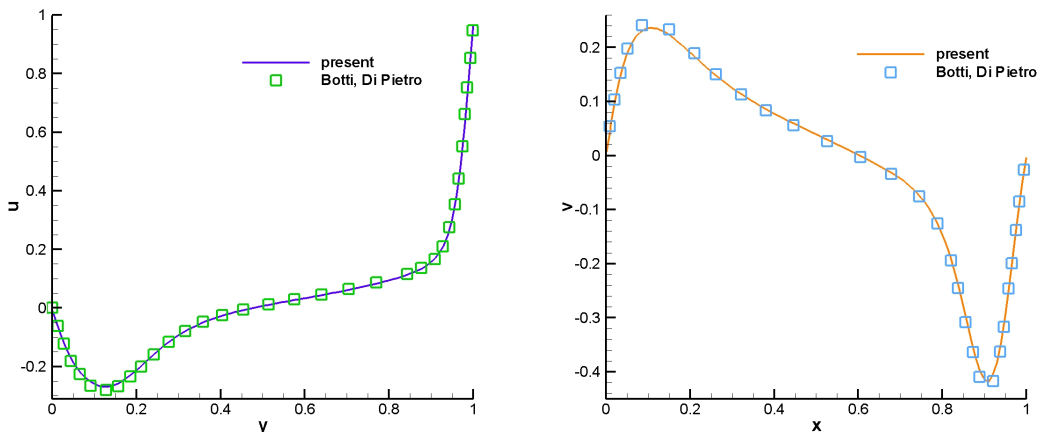


Figure 5.12: Velocity profiles comparison between Botti, Di-Pietro [13] and present work for the 3D Cavity. Profiles of the  $u$  velocity component along  $x=0.5$  (left) and the  $v$  velocity component along  $y=0.5$  (right) are shown.

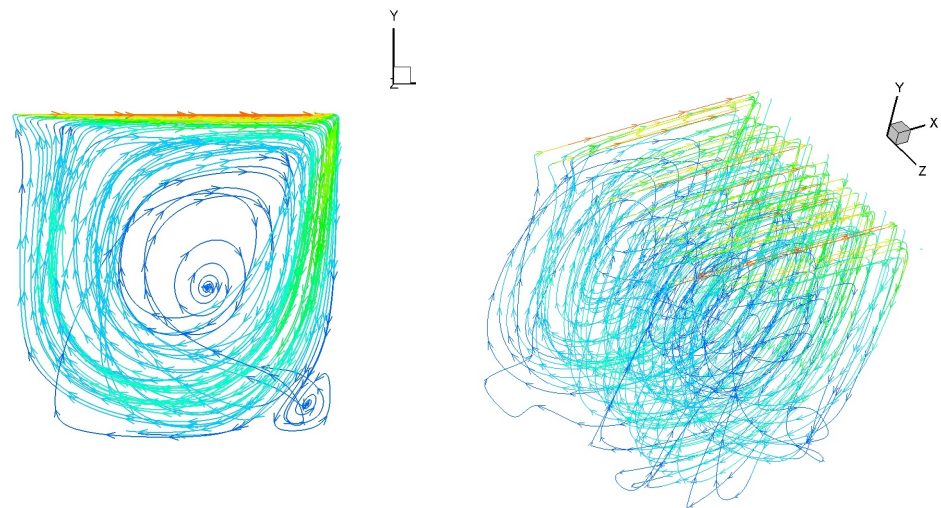


Figure 5.13: Velocity streamlines in 3D Cavity colored by velocity magnitude, in a 2D plane (left) and in the 3D domain (right).





# Conclusions

An efficient, high order accurate in space finite element method has been employed for incompressible flows. The projection method which was used, decouples the velocity with the pressure and satisfies the incompressibility constraint by solving the Poisson equation. The implicit algorithm developed, which is parallelizable, is using both continuous and discontinuous finite element approximation. Comparison with classical incompressible flow problems gave satisfactory results.

Use of multigrid for the Poisson equation can further enhance the efficiency of the method. The current method can be applied for fully unstructured, mixed-type meshes for simulations over complex geometries. Furthermore, the Spalart-Allmaras turbulence model must be incorporated to the present scheme in order to make possible RANS simulations at high Reynolds number external flows.



# Appendix A

## Linear Elasticity

Since the long term application of the scheme is FSI in wind turbine blades, the linear elasticity equations are solved next, using cG discretization. The displacement vector  $\mathcal{U} = [d_x \ d_y \ d_z]^\top$  is approximated by the following equation where  $\mathcal{U}_i$  is the displacement component along the  $x_i$  direction for each element  $\Omega_j$ .

$$\mathcal{U}_i = \sum_{k=1}^{nb_c} \Phi_c^k \underline{d}_i^k, \quad \Omega_j \in \Omega, \quad i = 1, 2, 3. \quad (\text{A.1})$$

The solution vector with the displacement coefficients is

$$\mathbf{U} = [\underline{d}_x^1 \ \underline{d}_x^2 \ \dots \ \underline{d}_x^{nb_c} \ \underline{d}_y^1 \ \underline{d}_y^2 \ \dots \ \underline{d}_y^{nb_c} \ \underline{d}_z^1 \ \underline{d}_z^2 \ \dots \ \underline{d}_z^{nb_c}]^\top. \quad (\text{A.2})$$

The strain in the specimen depends only on the stress applied on it and it doesn't depend on the rate or history of loading. The stress-strain and strain-displacement relations are linear and more specifically:

$$\sigma_{ij} = \frac{E}{1+\nu} \left( \varepsilon_{ij} + \frac{\nu}{1-2\nu} \varepsilon_{kk} \delta_{ij} \right), \quad (\text{A.3})$$

$$\varepsilon_{ij} = \frac{1}{2} \left( \frac{\partial d_i}{\partial x_j} + \frac{\partial d_j}{\partial x_i} \right). \quad (\text{A.4})$$

Using a vector form for the stress  $\boldsymbol{\sigma}$  and strain  $\boldsymbol{\varepsilon}$  tensors:

$$\boldsymbol{\sigma} = [C] \boldsymbol{\varepsilon}, \quad (\text{A.5})$$

$$[C] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix} \quad (\text{A.6})$$

$$\boldsymbol{\varepsilon} = [L]\boldsymbol{U} \rightarrow \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{yz} \\ \gamma_{zx} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \quad (\text{A.7})$$

where

$$\gamma_{ij} = 2\varepsilon_{ij}, \quad i \neq j. \quad (\text{A.8})$$

From Newton's second law

$$\nabla \bar{\boldsymbol{\sigma}} + \mathbf{b} = \rho \ddot{\boldsymbol{U}}, \quad (\text{A.9})$$

where  $\bar{\boldsymbol{\sigma}}$  is the stress in tensor form,  $\mathbf{b}$  is the field force vector,  $\rho$  is the density and  $\ddot{\boldsymbol{U}}$  is the acceleration. Using virtual work principle, Eq. (A.9) is multiplied by the virtual displacement  $\delta \boldsymbol{U}^\top$  and integrated on the domain.

$$\delta \Pi_{eq} = \int_{\Omega_j} \delta \boldsymbol{U}^\top (\nabla \bar{\boldsymbol{\sigma}} + \mathbf{b} - \rho \ddot{\boldsymbol{U}}) dV = 0, \quad (\text{A.10})$$

using Gauss' theorem:

$$- \int_{\Omega_j} \nabla \delta \boldsymbol{U}^\top \bar{\boldsymbol{\sigma}} dV + \int_{\partial \Omega_j} \delta \boldsymbol{U}^\top \bar{\boldsymbol{\sigma}} \cdot \mathbf{n} dS + \int_{\Omega_j} \delta \boldsymbol{U}^\top \mathbf{b} dV - \int_{\Omega_j} \delta \boldsymbol{U}^\top \rho \ddot{\boldsymbol{U}} dV = 0, \quad (\text{A.11})$$

replacing Eq. (A.5) into Eq. (A.11):

$$- \int_{\Omega_j} \delta \boldsymbol{\varepsilon}^\top \bar{\boldsymbol{\sigma}} dV + \int_{\partial \Omega_j} \delta \boldsymbol{U}^\top \mathbf{t} dS + \int_{\Omega_j} \delta \boldsymbol{U}^\top \mathbf{b} dV - \int_{\Omega_j} \delta \boldsymbol{U}^\top \rho \ddot{\boldsymbol{U}} dV = 0, \quad (\text{A.12})$$

---

where  $\mathbf{t} = \bar{\boldsymbol{\sigma}} \cdot \mathbf{n}$  is the traction vector. Using matrix notation:

$$-\int_{\Omega_j} [B][C][B]^\top dV \mathbf{U} + \int_{\partial\Omega_j} \Phi_c \mathbf{t} dS + \int_{\Omega_j} \Phi_c \mathbf{b} dV - \int_{\Omega_j} \Phi_c \Phi_c^\top \rho dV \ddot{\mathbf{U}} = 0, \quad (\text{A.13})$$

rearranging

$$\int_{\Omega_j} [B][C][B]^\top dV \mathbf{U} + \int_{\Omega_j} \Phi_c \Phi_c^\top \rho dV \ddot{\mathbf{U}} = \int_{\partial\Omega_j} \Phi_c \mathbf{t} dS + \int_{\Omega_j} \Phi_c \mathbf{b} dV, \quad (\text{A.14})$$

where

$$[B] = \begin{bmatrix} \frac{\partial \Phi_c}{\partial x} & 0 & 0 & 0 & \frac{\partial \Phi_c}{\partial z} & \frac{\partial \Phi_c}{\partial y} \\ 0 & \frac{\partial \Phi_c}{\partial y} & 0 & \frac{\partial \Phi_c}{\partial z} & 0 & \frac{\partial \Phi_c}{\partial x} \\ 0 & 0 & \frac{\partial \Phi_c}{\partial z} & \frac{\partial \Phi_c}{\partial y} & \frac{\partial \Phi_c}{\partial x} & 0 \end{bmatrix} \quad (\text{A.15})$$

Writing Eq. (A.14) in a more compact form:

$$\mathbf{K} \mathbf{U} + \mathbf{M} \ddot{\mathbf{U}} = \mathbf{R}(t), \quad (\text{A.16})$$

where  $\mathbf{M}$  is the mass matrix,  $\mathbf{K}$  is the stiffness matrix and  $\mathbf{R}$  is the load vector (field forces and tractions). For the static case the acceleration vanishes, thus  $\mathbf{K} \mathbf{U} = \mathbf{R}$ . Verification for the cantilever beam with uniform load is performed next and comparison between analytical and computational displacement is shown in Fig. (A.1). The analytical solution for the displacement in this case is given by

$$w(x) = \frac{Px^2(6L^2 - 4xL + x^2)}{24EI}, \quad (\text{A.17})$$

where  $P$  is the line pressure load on the beam,  $L$  is the beam length,  $I$  is the moment of inertia given by  $I = \frac{2}{3}c^3$  and  $c$  is the half of the height of the beam.

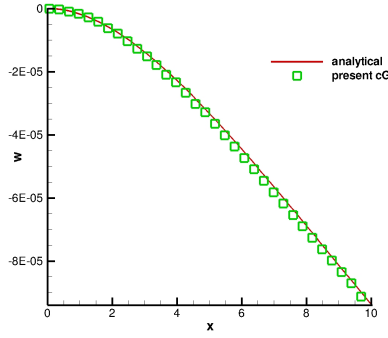


Figure A.1: Displacement comparison between analytical and computational solution in the cantilever beam case for  $E = 200GPa$ ,  $L = 10m$ ,  $c = 1m$ ,  $P = \frac{10^5}{L}N/m$ . 50 elements have been used in the  $x$  direction.

For unsteady simulations, Newmark's time integration [9] has been used, where the velocity and displacement vectors at time step  $t + \Delta t$  are calculated by:

$$\dot{\mathbf{U}}^{t+\Delta t} = \dot{\mathbf{U}}^t + \left[ (1 - \delta)\ddot{\mathbf{U}}^t + \delta\ddot{\mathbf{U}}^{t+\Delta t} \right] \Delta t, \quad (\text{A.18})$$

$$\mathbf{U}^{t+\Delta t} = \mathbf{U}^t + \dot{\mathbf{U}}^t \Delta t + \left[ (0.5 - \alpha)\ddot{\mathbf{U}}^t + \alpha\ddot{\mathbf{U}}^{t+\Delta t} \right] \Delta t^2. \quad (\text{A.19})$$

The parameters  $\alpha$  and  $\delta$  can be determined in order to obtain accuracy and stability:  $\delta \geq 0.50$  and  $\alpha \geq 0.25(0.5 + \delta)^2$ .

The algorithm for Newmark time integration is presented next:

- Initialize

1. Calculate the stiffness  $\mathbf{K}$  and mass  $\mathbf{M}$  matrices.
2. Initialize  $\mathbf{U}$ ,  $\dot{\mathbf{U}}$  and  $\ddot{\mathbf{U}}$ .
3. Select time step  $\Delta t$  and parameters  $\alpha$ ,  $\delta$ . Calculate the constants:

$$\alpha_1 = \frac{1}{\alpha\Delta t^2}; \quad \alpha_2 = \frac{1}{\alpha\Delta t}; \quad \alpha_3 = \frac{1}{2\alpha} - 1; \quad \alpha_4 = \Delta t(1 - \delta); \quad \alpha_5 = \delta\Delta t.$$

4. Form effective stiffness matrix  $\hat{\mathbf{K}} = \mathbf{K} + \alpha_1\mathbf{M}$ .
5. Inverse  $\hat{\mathbf{K}}$ .

- For each time step:

1. Calculate the effective loads at time  $t + \Delta t$ :

$$\hat{\mathbf{R}}^{t+\Delta t} = \mathbf{R}^{t+\Delta t} + \left( \alpha_1\mathbf{U}^t + \alpha_2\dot{\mathbf{U}}^t + \alpha_3\ddot{\mathbf{U}}^t \right) \mathbf{M}$$

- 
2. Find displacement vector at time  $t + \Delta t$  by solving the system:

$$\hat{\mathbf{K}}\mathbf{U}^{t+\Delta t} = \hat{\mathbf{R}}^{t+\Delta t}$$

3. Calculate the acceleration and the velocity vectors at time  $t + \Delta t$ :

$$\begin{aligned}\ddot{\mathbf{U}}^{t+\Delta t} &= \alpha_1(\mathbf{U}^{t+\Delta t} - \mathbf{U}^t) - \alpha_2\dot{\mathbf{U}}^t - \alpha_3\ddot{\mathbf{U}}^t, \\ \dot{\mathbf{U}}^{t+\Delta t} &= \mathbf{U}^t + \alpha_4\dot{\mathbf{U}}^t + \alpha_5\ddot{\mathbf{U}}^{t+\Delta t}.\end{aligned}$$

The unsteady algorithm has been tested on the string vibration case. The displacement  $u(x, t)$  of the string is described by the partial differential equation:

$$\lambda^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}, \quad (\text{A.20})$$

where  $u(0, t) = 0$  and  $u(L, t) = 0$  are the boundary conditions and  $\lambda = 3$  was chosen. The initial conditions for the wave propagation problem were chosen to be  $\dot{u}(x, 0) = 0$  and  $u(x, 0) = 4\sin(\pi x) - \sin(2\pi x) - 3\sin(5\pi x)$ , because this way the solution of Eq. (A.20) has finite number of modes, and is given by:

$$u(x, t) = 4\cos(3\pi t)\sin(\pi x) - \cos(6\pi t)\sin(2\pi x) - 3\cos(15\pi t)\sin(5\pi x). \quad (\text{A.21})$$

Differentiating Eq. (A.21) with respect to time one and two times, the velocity and acceleration are derived respectively. Comparison between analytical and computational solutions for the displacement, velocity and acceleration is shown in Fig. (A.2).

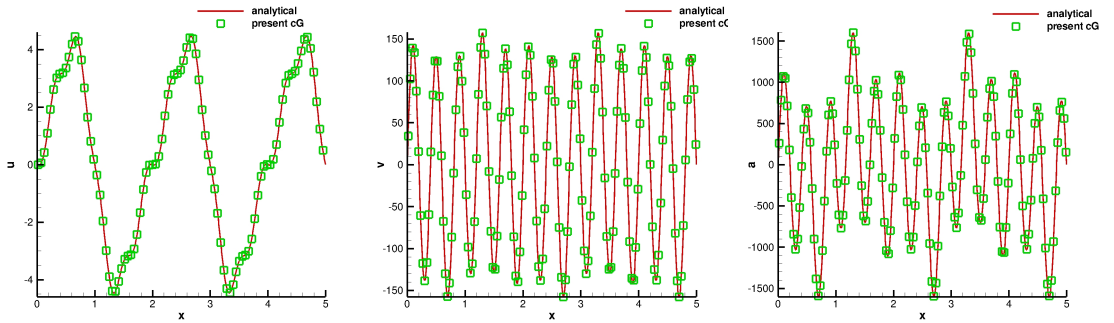


Figure A.2: Comparison between analytical and computational solutions for the displacement (left), velocity (middle) and acceleration (right) in the string vibration case at time  $t = 0.03$ . 500 elements have been used in order to capture the acceleration variations.





# Bibliography

- [1] D. N. Arnold. An Interior Penalty Finite Element Method with Discontinuous Elements. *SIAM Journal on Numerical Analysis*, 19:742–760, August 1982.
- [2] S. Balay, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B. Smith, and H. Zhang. *PETSc users manual, 2.3.3 edn.* Argonne National Laboratory, Mathematics and Computer Science Division.
- [3] T. J. Barth and H. Deconinck. *High-Order Methods for Computational Physics*. Springer, 1999.
- [4] F. Bassi, A. Crivellini, D. A. Di Pietro, and S. Rebay. An implicit high-order discontinuous Galerkin method for the steady and unsteady incompressible flows. *Computers & Fluids*, 36(10):1529–1546, 2007.
- [5] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 131:267–279, 1997.
- [6] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comp. Phys.*, 131:267–279, 1997.
- [7] F. Bassi and S. Rebay. A high-order discontinuous Galerkin method for compressible turbulent flow. *Lecture Notes in Computational Science and Engineering*, 11:77–88, 2000.
- [8] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. A high-Order Accurate Discontinuous Finite Element Method for Inviscid and Viscous Tur-

- bomachinery Flows. *2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, 5-7:99–108, 1997.
- [9] K. J. Bathe. *Finite Element Procedures*. PRENTICE HALL, 1996.
- [10] Y. Bazilevs, K. Takizawa, and T. E. Tezduyar. *Computational Fluid structure interaction (Methods and Applications)*. Wiley, 2013.
- [11] R. M. Beam and R. F. Warming. An implicit factored scheme for the compressible Navier-Stokes equations. *AIAA Journal*, 16:393–401, 1978.
- [12] P. Birken, G. Gassner, M. Haas, and C.-D. Munz. Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations. *J. Comp. Phys.*, 240(11):20–35, 2013.
- [13] L. Botti and D. A. Di Pietro. A pressure correction scheme for convection-dominated incompressible flows with discontinuous velocity and continuous pressure. *J. Comp. Phys.*, 230:572–585, 2011.
- [14] L. Botti and D. A. Di Pietro. dG-cG pressure-correction ins solver for hemodynamics, 2011.
- [15] A. Brandt. Multi-level adaptive technique (MALT) for fast numerical solution to boundary value problem. *Lecture notes in physics, Berlin: Springer*, 18:82–9, 1973.
- [16] W. R. Briley. A numerical study of laminar separation bubbles using the Navier-Stokes equations. *J Fluid Mech*, 47:713–36, 1971.
- [17] W. R. Briley and H. McDonald. Solution of the three-dimensional compressible Navier-Stokes equations by an implicit technique. *Lecture notes in Physics, New York: Springer*, 435:105–10, 1974.
- [18] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, 1990.
- [19] G. Chavent and Salzano. A finite element method for the 1D water flooding problem with gravity. *J. Comput. Phys.*, 45:307–344, 1982.

- [20] S. Chen and Y.-T. Zhang. Krylov implicit integration factor methods for spatial discretization on high dimensional unstructured meshes: Application to discontinuous Galerkin methods. *J. Comp. Phys.*, 230(11):4336–4352, 2011.
- [21] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math of Computation*, 22:745–762, 1968.
- [22] B. Cockburn, G. Karniadakis, and C. W. Shu. The development of discontinuous Galerkin methods. *Lecture Notes in Computational Science and Engineering*, 11:3–50, 2000.
- [23] B. Cockburn and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework. *Math. Comp.*, 52:411–435, 1989.
- [24] B. Cockburn and C. W. Shu. The Runge-Kutta local projection  $P^1$ -discontinuous Galerkin method for scalar conservation laws. *RAIRO Model. Math. Anal. Numer.*, 25:337–361, 1991.
- [25] B. Cockburn and C. W. Shu. The local discontinuous Galerkin finite element method for convection-diffusion systems. *SIAM J. Numer. Anal.*, 35, 1998.
- [26] T. F. Coleman and J. J. More. Estimation of sparse Jacobian matrices and graph coloring blems. *SIAM journal on Numerical Analysis*, 20(1):187–209, 1983.
- [27] R. Courant, K. O. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der Math. Physik. *Math Ann*, 100:32–74, 1928.
- [28] A. Crivellini and F. Bassi. An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations. *Computers & Fluids*, 50(1):81–93, 2011.
- [29] R. T. Davis and I. Flugge-Lotz. Second-order boundary effects in hypersonic flow past axisymmetric blunt bodies. *J Fluid Mech*, 20:593–623, 1964.
- [30] C. N. Dawson. Godunov-mixed methods for advection-diffusion equations in one space dimension. *SIAM J. Numer. Anal*, 28:1282–1309, 1991.

- [31] B. Delaunay. Sur la Sphère Vide. *Bull Acad Sci, USSR, VII, Class Sci Mat Nat*, pages 793–800, 1934.
- [32] M. Delfour and F. Trochu. Discontinuous Galerkin methods for the approximation of optimal control problems governed by hereditary differential systems. *Distributed Parameter Systems: Modelling and Identification (A. Ruberti, ed.)*, Springer Verlag, pages 256–271, 1978.
- [33] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press, 2002.
- [34] D. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer, 2010.
- [35] D. A. Di Pietro and A. Ern. Discrete functional analysis tools for discontinuous Galerkin methods with application to the incompressible Navier-Stokes equations. *Math. Comput.*, 79:1303–1330, 2010.
- [36] S. Dong and J. Shen. An unconditionally stable rotational velocity-correction scheme for incompressible flows. *J. Comp. Phys.*, 229:7013–7029, 2010.
- [37] F. Duarte, R. Gormaz, and S. Natesan. Arbitrary Lagrangian-Eulerian method for the Navier-Stokes equations with moving boundaries. *Computer Methods in Applied Mechanics and Engineering*, 193, November 2004.
- [38] E. Ferrer. *A high order Discontinuous Galerkin-Fourier incompressible 3D Navier-Stokes solver with rotating sliding meshes for simulating cross-flow turbines*. PhD thesis, Brasenose College, Department of Engineering Science, University of Oxford, 2012.
- [39] Fluent, Inc., Centerra Resource Park, 10 Cavendish Court, Lebanon, NH 03766. *GAMBIT Tutorial Guide*, 2000.
- [40] M. Fortin and A. Fortin. New approach for the finite element method simulation of viscoelastic flows. *J. Non-Newt. Fluid Mech.*, 32:295–310, 1989.
- [41] G. Galdi and Rannacher. *Fundamental Trends in Fluid-Structure Interaction*. World Scientific, 2010.

- [42] A. H. Gebremedhin, F. Manne, and A. Pothen. What Color Is Your Jacobian? Graph Coloring for Computing Derivatives. *SIAM Review*, 47:629–705, 2005.
- [43] S. K. Godunov. Finite-difference method for numerical computational of discontinuous solution of the equations of fluid dynamics. *Mat Sb*, 47:271–306, 1959.
- [44] S. Gottlieb and L.-A. J. Gottlieb. Strong stability preserving properties of Runge-Kutta time discretization methods for linear constant coefficient operators. *Journal of Scientific Computing*, 18(1):83–109, 2003.
- [45] N. Gregory and C. L. O’Reilly. Low-Speed Aerodynamic Characteristics of NACA 0012 Aerofoil Sections, including the Effects of Upper-Surface Roughness Simulation Hoar Frost. Technical report, NASA R&M 3726, 1970.
- [46] M. M. Hafez. *Numerical Simulations of Incompressible Flows*. World Scientific, 2003.
- [47] F. H. Harlow. The particle-in-cell computing method for fluid dynamics. *Methods in computational physics, New York: Academic Press*, 3:319–43, 1964.
- [48] A. Jameson. Iterative solution of transonic flows over airfoils and wings including flows at mach 1. *Commun Pure Appl Math*, 17:283–309, 1974.
- [49] G. E. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, 2003.
- [50] G. Karypis and V. Kumar. Metis: Unstructured graph partitioning and sparse matrix ordering system, version 5.1, 2013.
- [51] B. Kirk, J. W. Peterson, R.H. Stogner, and G. F. Carey. libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations Boundary Conditions for incompressible flows. *Eng. Comput.*, 22:237–254, 2006.
- [52] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comp. Phys.*, 193(2):357–397, 2004.

- [53] L. Kovasznay. Laminar flow behind a two dimensional grid. *Proc. Camb. Phil. Soc.*, 44:58–62, 1948.
- [54] C. L. Ladson. Effects of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section. Technical report, NASA TM 4074, 1988.
- [55] P. D. Lax. Weak solution of nonlinear hyperbolic equations and their numerical computation. *Commun Pure Appl Math*, 7:159–63, 1954.
- [56] P. LeSaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. *Mathematical aspects of finite elements in partial differential equations (C. de Boor, ed.)*, Academic Press, pages 89–145, 1974.
- [57] B. Q. Li. *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Springer, London, 2006.
- [58] R. B. Lowrie and J. Morel. Discontinuous Galerkin for the hyperbolic systems with stiff relaxation. *American Institute of Aeronautics and Astronautics*, 761235, Norfolk, VA, 1999.
- [59] R. W. MacCormack. The effect of viscosity in hyper velocity impact cratering. *AIAA paper*, 69-354, 1969.
- [60] R. W. MacCormack. Current status of numerical solutions of the Navier-Stokes equations. *AIAA 85-0032*, 1985.
- [61] R. W. MacCormack and A. J. Paullay. Computational efficiency achieved by time splitting of finite difference operators. *AIAA 72-154*, 1972.
- [62] Y. Maday, A. T. Patera, and E. M. Ronquist. An Operator-integration-factor splitting method for time-dependent problems: Application to incompressible fluid flow. *SIAM Review*, 5:263–292, 1990.
- [63] G. Moretti and M. Abbett. A time-dependent computational method for blunt-body flows. *AIAA Journal*, 4(12):2136–41, 1966.
- [64] E. M. Murman and J. D. Cole. Calculation of plane steady transonic flows. *AIAA Journal*, 9:114–21, 1971.

- [65] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations. *J. Comp. Phys.*, 228:3232–3254, 2009.
- [66] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations. *J. Comp. Phys.*, 230:1147–1170, 2011.
- [67] T. A. Oliver. *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*. PhD thesis, Massachusetts Institute of Technology, 2008.
- [68] Pointwise Inc., Forth Worth, Texas. *Pointwise User Manual*, 2013.
- [69] M. K. Prasad, J.L. Milovich, A. I. Shestahov, D.S. Kershaw, and J. J. Shaw. 3D unstructured mesh ALE hydrodynamics with the upwind discontinuous Galerkin method. *International Symposium on Discontinuous Galerkin Methods, Newport, RI*, 1999.
- [70] J. V. Rakich. A method of characteristics for steady three dimensional supersonic flow with application to inclined bodies of revolution. *NASA TN D-5341*, 1969.
- [71] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. *Tech Report LA-UR-73-379, Los Alamos Scientific Laboratory*, 1973.
- [72] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Phil Trans R Soc London, Series A*, 210:307–357, 1910.
- [73] G. R. Richter. The discontinuous Galerkin method with diffusion. *Math. Comp.*, 58:631–643, 1992.
- [74] G. F. B. Riemann. Über die Fortpflanzung Ebener Luftwellen von Endlicher Schwingungsweite. *Abh Konigl Ges Wiss Gottingen*, 8:449, 1860.



- [75] A. Rizzi and M. Inouye. Time-splitting finite-volume method for three-dimensional blunt-body flow. *AIAA Journal*, 11:1478–85, 1973.
- [76] P. Rubbert. On the continuing evolution of CFD for airplane design. *Conference text book, supercomputing, Japan*, April 1991.
- [77] M. Sahin and R. Owens. A novel fully implicit finite volume method applied to the lid-driven cavity-Part I: High Reynolds number flow calculations. *Int. J. Numer. Meth. Fluids*, 42:57–77, 2003.
- [78] B. Sande and B. Korren. Runge-Kutta methods for the incompressible Navier-Stokes equations. *AIAA Paper, 21st CFD Conference*, 3085, 2013.
- [79] K. Shahbazi. *A Parallel High-Order Discontinuous Galerkin Solver for the Unsteady Incompressible Navier-Stokes Equations in Complex Domains*. PhD thesis, Department of Mechanical and Industrial Engineering, University of Toronto, 2007.
- [80] J. S. Shang. Three decades of accomplishments in computational fluid dynamics. *Elsevier*, November 2004.
- [81] J. S. Shang and W. L. Hankey. Numerical solution of the compressible Navier-Stokes equations for a three-dimensional corner. *AIAA*, 15:1575–82, 1977.
- [82] J. S. Shang and S. J. Scherr. Navier-Stokes solution for a complete reentry configuration. *AIAA 85-1509; J Aircraft 1986*, 23:881–8.
- [83] R. V. Southwell. *Relaxation methods in engineering science*. London, UK: Oxford University Press, November 1940.
- [84] K. Stewartson and P. G. Williams. Self-induced separation. *Proc R Soc London, Series A*, 312:181–206, 1969.
- [85] W. Z. Strang, R. F. Tomaro, and M. J. Grismer. The defining methods of cobalt: a parallel, implicit, unstructured Euler/Navier-Stokes flow solver. *AIAA 99-0786*, January 1999.

- [86] Tecplot, Inc., Bellevue, WA. *Tecplot 360 User's Manual*, 2013.
- [87] R. Temam. Une méthode d'approximation des solutions des équations Navier-Stokes. *Bull. Soc. Math. France*, 98:115–152, 1968.
- [88] A. Thom. The flow past circular cylinder at low speeds. *Proc R Soc London, Series A*, 141:651–66, 1933.
- [89] J. L. Thomas and R. W. Walters. Upwind relaxation algorithms for the Navier-Stokes equations. *AIAA 85-1501*, 1985.
- [90] J. F. Thompson, F.C. Thames, and C. W. Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *J Comput Phys*, 15:299–319, 1974.
- [91] M. Van Dyke. *An Album of Fluid Motion*. THE PARABOLIC PRESS, 1982.
- [92] B. van Leer. On the relation between the upwind differencing schemes of Godunov, Engquist-Osher, and Roe. *SIAM J Sci Stat Comput*, 5:1–20, 1984.
- [93] J. von Neumann and R. D. Richtmeyer. A method for the numerical calculation on the hydrodynamic shocks. *J. Appl. Phys.*, 21:232–7, 1950.
- [94] L. Wang and D. J. Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. *J. Comp. Phys.*, 225(2):1994–2015, 2007.
- [95] T. C. Warburton and G. E. Karniadakis. A discontinuous Galerkin method for the viscous MHD equations. *J. Comput. Phys.*, 152:1–34, 1999.
- [96] L. C. Wellford and J. T. Oden. A theory of discontinuous finite element approximations for the analysis of shock waves in nonlinear elastic materials. *J. Comput. Phys.* 19, 19:178–210, 1975.
- [97] L. C. Wellford and J. T. Oden. A theory of discontinuous finite element approximations for the analysis of shock waves in nonlinear elastic solids: Accuracy and convergence. *Comput. Methods Appl. Mech. Engrg.*, 8:17–36, 1976.

- [98] L. C. Wellford and J. T. Oden. A theory of discontinuous finite element approximations for the analysis of shock waves in nonlinear elastic solids: Variational theory. *Comput. Methods Appl. Mech. Engrg*, 8:1–16, 1976.