ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
Τμημα Μαθηματικων

Κυριακή Ν. Μπυράκη

Η Μεθοδος των Πεπερασμενων Στοιχειων με Εφαρμογες στη Ρευστομηχανικη

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

Ιωάννινα, 2022

UNIVERSITY OF IOANNINA

Department of Mathematics

Kyriaki N. Byraki

The Finite Element Method with Applications in Fluid Mechanics

Master's Thesis

Ioannina, 2022

Αφιερώνεται στον μπαμπά μου, Νίκο

Η παρούσα Μεταπτυχιακή Διατριβή εκπονήθηκε στο πλαίσιο των σπουδών για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στα Εφαρμοσμένα Μαθηματικά και την Πληροφορική που απονέμει το Τμήμα Μαθηματικών του Πανεπιστημίου Ιωαννίνων.

Εγκρίθηκε την 8 / 6 / 2022 από την εξεταστική επιτροπή:

Ονοματεπώνυμο	**Βαθμίδα**
Μιχαήλ Ξένος	Αναπληρωτής Καθηγητής (Επιβλέπων)
Θεόδωρος Χωρίκης	Καθηγητής
Φωτεινή Καρακατσάνη	Επίκουρη Καθηγήτρια

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

"Δηλώνω υπεύθυνα ότι η παρούσα διατριβή εκπονήθηκε κάτω από τους διεθνείς ηθικούς και ακαδημαϊκούς κανόνες δεοντολογίας και προστασίας της πνευματικής ιδιοκτησίας. Σύμφωνα με τους κανόνες αυτούς, δεν έχω προβεί σε ιδιοποίηση ξένου επιστημονικού έργου και έχω πλήρως αναφέρει τις πηγές που χρησιμοποίησα στην εργασία αυτή."

Κυριακή Ν. Μπυράκη

Ευχαριστιες

Η παρούσα διατριβή πραγματοποιήθηκε στο Τμήμα Μαθηματικών του Πανεπιστημίου Ιωαννίνων. Με την ολοκλήρωση της παρούσας διατριβής θα ήθελα να ευχαριστήσω όλους εκείνους, των οποίων η συμβολή ήταν καθοριστική σε αυτή τη μελέτη.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μιχαήλ Ξένο, για τη βοήθεια, τις πολύτιμες συμβουλές, την υποστήριξη και την καθοδήγηση που μου προσέφερε κατά τη διάρκεια της εκπόνησης της εργασίας.

Ιδιαίτερα θέλω να ευχαριστήσω και τα άλλα δύο μέλη της τριμελούς επιτροπής, κ. Θεόδωρο Χωρίκη και κα. Φωτεινή Καρακατσάνη για το ενδιαφέρον που έδειξαν, τον χρόνο που διέθεσαν αλλά και για τις εύστοχες και εποικοδομητικές παρατηρήσεις και υποδείξεις τους στην διατριβή αυτή με σκοπό την βελτίωση της.

Ακόμη θα ήθελα να ευχαριστήσω όλους του διδάσκοντες του Τμήματος Μαθηματικών για τις γνώσεις που μου προσέφεραν σε προπτυχιακό και μεταπτυχιακό επίπεδο και τη συνεργασία που είχαμε όλα αυτά τα χρόνια.

Το πιο μεγάλο ευχαριστώ ανήκει δικαιωματικά στην οικογένειά μου, που με στήριξε ώστε να ολοκληρώσω τις μεταπτυχιακές μου σπουδές.

Ιωάννινα 2022,
Κ. Μπυράκη

Περιληψη

Η μέθοδος των πεπερασμένων στοιχείων είναι μια ευρέως γνωστή αριθμητική μέθοδος για τον υπολογισμό προσεγγιστικών λύσεων συνήθων διαφορικών εξισώσεων (Σ.Δ.Ε.) και μερικών διαφορικών εξισώσεων (Μ.Δ.Ε.). Η μέθοδος είναι ένα πολύ ισχυρό εργαλείο στη μελέτη διαφόρων μαθηματικών μη-γραμμικών προβλημάτων και έχει πολλές εφαρμογές, όπως η δομική ανάλυση και η μηχανική των ρευστών. Σε αυτή τη διατριβή επικεντρωνόμαστε στην εφαρμογή της μεθόδου κυρίως σε προβλήματα Ρευστομηχανικής. Αρχικά παρουσιάζουμε τη μέθοδο μαζί με τα βασικά θεωρήματα και παραδείγματα. Αναλύουμε τα εκ των προτέρων (a priori) σφάλματα για γραμμικά προβλήματα και παρουσιάζουμε τις συναρτήσεις βάσης που εφαρμόζονται στα υπό εξέταση προβλήματα που μελετάμε.

Παρουσιάζουμε την αριθμητική λύση της μονοδιάστατης μη-γραμμικής εξίσωσης του Duffing. Επιπλέον, επικεντρωνόμαστε στο δισδιάστατο πρόβλημα του Stokes. Στην εργασία αυτή παρουσιάζουμε νεότερες παραλλαγές των μεθόδων πεπερασμένων στοιχείων, όπως είναι η ασυνεχής μέθοδος του Galerkin. Παρουσιάζεται επίσης η έννοια του τοπικά εκλεπτυσμένου πλέγματος (adaptive mesh). Τέλος, μελετάμε τις δισδιάστατες μη-γραμμικές εξισώσεις των Navier–Stokes εφαρμόζοντας τη κλασική μέθοδο του Galerkin. Αυτές οι προηγμένες μέθοδοι παρέχουν αξιόπιστα αριθμητικά αποτελέσματα σε όλες τις περιπτώσεις που μελετήθηκαν. Αυτό επιτυγχάνεται με την εφαρμογή των μεθόδων Πεπερασμένων Στοιχείων σε κατάλληλα ‘προβλήματα δοκιμής’, όπως είναι η οπίσθια κατάβαση (σκαλί) της ροής (backward facing step). Όλα τα αριθμητικά πειράματα έχουν πραγματοποιηθεί με κώδικά που αναπτύχθηκε στα προγράμματα Matlab και FEniCS.

Abstract

The finite element method is a widely known numerical method for calculating approximate solutions of ordinary differential equations (ODEs) and partial differential equations (PDEs). This method is a powerful tool in the study of various nonlinear problems and has many applications, such as structural analysis and fluid mechanics. In this thesis we concentrate on applying the method mainly to Fluid Mechanics problems. Initially, we present the method along with the basic theorems and examples. We analyze the *a priori* errors for linear problems and the base functions that distinguish the problem under consideration.

We further present the numerical solution of the one–dimensional nonlinear Duffing equation. Additionally, we concentrate on the two–dimensional Stokes problem. We focus on presenting novel finite element method variants, such as the Discontinuous Galerkin method. The notion of adaptive mesh is also discussed. Lastly, we study the two–dimensional Navier–Stokes equations. We present the formulation of the equations in the classical Galerkin method. These advanced methods provide reliable numerical results in all studied cases. This is achieved with the application of the Finite Element methods to appropriate "test problems", such as the backward facing step. We obtain all the numerical results utilizing the software programs Matlab and FEniCS.

Contents

CHAPTER 1

Introduction

Nonlinear differential equations govern a plethora of biological, mechanical and physical phenomena. In practice, all PDEs aren't analytically solvable, hence numerical methods are utilized. The finite element method (FEM) is a widely known numerical method providing approximate solutions to differential equations. A large domain divides into smaller discrete cells, called finite elements, being simple polygonal shapes, forming the computational mesh of this domain. The method excels in its accurate representation of complex geometries, the finite elements of which are approximated by polynomials. Nowadays, FEM is arguably one of the most well established and convenient computational techniques. There is a variety of applications in many fields, such as mechanical design, structural analysis, fluid flow, heat transfer and electromagnetism to computer programming aspects.

Origins of the FEM are found in the early approximation of π by considering a sequence of inscribed polygons, although the method was formally introduced in 1960 by Clough [28]. In terms of the present day notation, each side of the polygon represented an element and as their number increases, the approximate values converge to the true one. Solving complex elasticity and structural analysis problems in civil and aeronautical engineering, for example wings and fuselages are treated as assemblies of stringers, skins and sear panels, further developed the method.

In 1851 Schellback in order to obtain a differential equation of a surface of a minimum area bounded by a specific closed curve, divided the surface into several triangles and used a finite difference expression to find the total discretized area. The equation was replaced by a set of algebraic equations. Until the 1900's the behaviour of structural frameworks, composed of several bars arranged in a regular pattern, has been approximated by one of an Isotropic elastic body.

In 1909 Ritz developed an effective method to find the approximate solution to problems of deformable solid mechanics. His approach referred to an approximation of an energy functional by known functions multiplied with unknown coefficients. Minimizing the functional in relation to each unknown leads to a system determining those coefficients. The functions used are restricted in satisfying the given boundary conditions.

In 1915, Boris Grigoryevich Galerkin formulated a general method for solving differential equations. His method with piecewise polynomial spaces is known as the finite element method. Technological advancements, further developed Galerkin's method. The approach traces back to variational principles of Leibniz, Euler, Lagrange, Dirichlet, Hamilton, Castigliano, Rayleigh and Ritz. Hrenikoff, in 1941, introduced the framework method, replacing a plain elastic medium with an equivalent system of sticks and rods.

The FEM was introduced in the 50s by structural engineers, especially in the aircraft industry, predicting stresses induced in aircraft wings, despite being independently proposed by Courant in 1943 [24]. He introduced special linear functions over triangular regions, obtained by dividing the cross region and applying the method for the solution of torsional rigidity and hallow shaft. The latter introduced the Rayleigh–Ritz method. Ritz functions didn't need to satisfy the boundary conditions. Courant's theory could not be implemented due to the current absence of computers. More significant contributions to FEM, were made by Turner in 1956.

FEM obtained its real impetus in the 60s and 70s through developments of, among others, J. H. Argyris and collaborators. Clough in 1960 introduced the term "finite element" in [4]. The first book on FEM was published in 1967 by Zienkiewicz and Cheung [27]. The main motive behind the wide spread of FEM was the handling of big volume of numerical solution by computers [23].

In this dissertation, focus is set on the Galerkin FEM. As previously mentioned, any progress to FEM, regarding fluid mechanics applications, was significantly delayed due to nonlinear convection and solution instability originating from the element selection. In this section, we are analysing basic principals of FEM, where more details can be found in the textbooks by Brenner & Scott [6]. In the next section we discuss about the FEM for the Stokes and Navier–Stokes problems. Finally, we introduce the Discontinuous Galerkin (DG) method and the adaptive mesh refinement approach.

CHAPTER 2

Basic Principles of Finite Elements

2.1 Finite Element Theory

It is important to understand the basics of the finite element theory. Analyzing the following simple example helps the reader to understand the introduction of the weak form, as well as the path needed to be followed in order to create the weak form. Consider an one–dimensional boundary value problem,

$$\begin{cases} -\dfrac{d^2u}{dx^2} = f(x), \quad x \in (0,1) \\ u(0) = 0, \quad u(1) = 0 \end{cases} \tag{2.1}$$

Multiplying both parts of the equation with a function v with $v(0) = v(1) = 0$ and integrating by parts, we get,

$$\begin{aligned}(f,v) := \int_0^1 f(x)v(x)\ dx = \int_0^1 -u''(x)v(x)\ dx = -[\,u'(x)v(x)]\,_0^1 + \int_0^1 u'(x)v'(x)\ dx \\ = \int_0^1 u'(x)v'(x)\ dx := \alpha(u,v),\end{aligned}$$

where $\alpha(u,v)$ is a bilinear form.

Definition 2.1.1 (**Bilinear form**). A bilinear form is a function $B : V \times V \to K$ where V is a vector space and K is the field of scalars, that is linear in each argument separately,

1. $B(u+v,w) = B(u,w) + B(v,w)$ and $B(\lambda u, v) = \lambda B(u,v)$
2. $B(u,v+w) = B(u,w) + B(u,w)$ and $B(u,\lambda v) = \lambda B(u,v)$

Definition 2.1.2 (**Square–integrable function**)**.** A square–integrable function or a quadratically integrable function is denoted as L^2 function and is defined as: $f : [a, b] \to \mathbb{R}$, square–integrable on [a,b] $\iff \int_a^b |f(x)|^2 \, dx < \infty$

Taking into consideration the above informations a function space can be defined as a test space:

$$V = \{v \in L^2(0, 1) : \alpha(u, v) < \infty \text{ and } v(0) = v(1) = 0\} \tag{2.2}$$

and

$$u \in V \text{ such that } a(u, v) = (f, v), \; \forall v \in V, \tag{2.3}$$

where $L^2(0, 1)$ is the space of square integrable functions in $[0, 1]$.

In general, the function v which multiplies the PDE is referred as a test function. The unknown function u that needs to be approximated is called a trial function.

The function v is an arbitrary function. For the linear case it has been shown that: If the weak form is $a(u, v) = (f, v)$, where $a(\cdot, \cdot)$ is bilinear, and $u \in C^2[0, 1]$ and $f \in C^0[0, 1]$ satisfy the weak form, then u also satisfies the strong form with the appropriate initial conditions [6].

According to the Ritz–Galerkin approximation we have that if $S \subset V$ is any finite dimensional subspace and we consider that (2.3) with V is replaced by S, we get,

$$u_S \in S \text{ such that } a(u_S, v_S) = (f, v_S), \; \forall v_S \in S. \tag{2.4}$$

With the above we can define a discrete scheme for approximating (2.1) and it has been proven that given $f \in L^2(0, 1)$, the equation has a unique solution.

2.1.1 Error Estimates

Definition 2.1.3 ($L^2(0, 1)$ **norm**)**.**

$$\|v\| = (v, v)^{\frac{1}{2}} = \left(\int_0^1 v(x)^2 dx \right)^{\frac{1}{2}} \tag{2.5}$$

2.1.2 Piecewise Polynomial Spaces

We introduce linear polynomials to construct the Galerkin FEM and for that purpose we should introduce a partition of our domain. We consider $0 = x_0 < x_1 < ... < x_N = 1$, be a partition of $[0, 1]$, and S be the linear space of functions v such that,
$S = \{v : [0, 1] \to \mathbb{R} : v$ is continuous, $v \in [x_i, x_{i+1}]$ is linear polynomial for $i = 1, ..., N$ and $v(0) = v(1) = 0\}$

For each $i = 1, ..., N$ we can define ϕ_i and $\phi_i(x_j) = \delta_{ij}$ = the Kronecker delta i.e:

$$\begin{cases} \phi_i(x) = \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, xi] \\ \phi_i(x) = \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & x \in [x_i, x_{i+1}] \\ \phi_i(x) = 0, \quad \textit{otherwise} \end{cases} \tag{2.6}$$

Theorem 2.1.1. *The functions $\{\phi_i\}_{i=1}^N$ are the basis functions of S, [21].*

Proof. Showing that

1. $\{\phi_i\}_{i=1}^N$ are linear independent

2. they produce S

proves the above theorem. We consider a linear combination of the functions ϕ_i which is zero at $[0, 1]$.

$$\sum_{i=1}^{N} \lambda_i \phi_i(x) = 0 \ , \forall x \in [0, 1].$$

Then, because $\phi_i(x_i) = 0$ for $i \neq j$ and $\phi_i(x_i) = 1$ for $i = j$, we have that $\lambda_i = 0$, $i = 1, ..., N$. Therefore, $\{\phi_i\}_{i=1}^N$ are linearly independent. In addition, if $v \in S$, we see that $v(x) = \sum_{i=1}^N v(x_i)\phi_i(x)$ $, \forall x \in [0, 1]$, because v and $\sum_{i=1}^N v(x_i)\phi_i$, are linear polynomials in any interval $[x_i, x_{i+1}]$ and are identical at its edges. Since 1. and 2. hold, $\{\phi_i\}_{i=1}^N$ are the basis functions of S. □

- $\{\phi_i : 1\leq \text{i} \leq \text{N} \}$ is a nodal basis from S and it's called the **nodal basis** of S. The set $\{\phi_i\}$ is used in order to define the functions of a discrete space.
- $\{u(x_i)\}$ are the **nodal values** of function u
- $\{x_i\}$ are the **nodes**
- $u_h = \sum_{i=1}^{n} u(x_i)\phi_i$, for $u \in C^0([0,1])$ and $u_h \in S$, is the **approximate solution** of u

Remark. If $u \in S \implies u = u_h$ since $u - u_h$ is linear on each $[x_{i-1}, x_i]$ and zero at the endpoints, hence must be identically zero.

Writing (2.1) in terms of a basis $\{\phi_i : 1 \leq i \leq n\}$ of S with $u_S = \sum_{i=1}^{N} u_i\phi_j$, $K_{ij} = a\left(\phi_j, \phi_i\right), F_i = \left(f, \phi_i\right)$ for $i, j = 1, \ldots, N$. Setting $\mathbf{U} = \left(u_i\right), \mathbf{K} = \left(K_{ij}\right)$ and $\mathbf{F} = \left(F_i\right)$. Then, equation (2.1) is equivalent to solving the following algebraic system,

$$\mathbf{KU} = \mathbf{F} \tag{2.7}$$

The matrix $\mathbf{K}$ is symmetric and positive-definite. In addition, the matrix is sparse and tridiagonal, so the system in equation (2.7) can be solved.

2.1.3 An Application to the Duffing equation

In this subsection, we present an example of solving the Duffing equation using the Galerkin FEM method. The Duffing equation arguably serves as the simplest mathematical model for describing the chaotic behaviour of a system. It describes a periodically forced oscillator of a second order non-linear ODE with constant coefficients.

$$\begin{cases} \dfrac{d^2u}{dt^2} + \delta\dfrac{du}{dt} + \alpha u - \beta u^3 = \gamma\cos(\omega t), \quad u = u(t), \quad t \in [0, L] \\ u(0) = 0 \ \text{ and } \ u'(L) = 0. \end{cases} \tag{2.8}$$

The parameter α, is the linear stiffness coefficient, β controls the amount of nonlinearity in the restoring force (if $\beta = 0$, the equation describes simple harmonic oscillation). In physical terms the equation represents a nonlinear spring whose stiffness does not exactly obey Hooke's law. The term $\gamma\cos(\omega t)$

is the external force. At first, we will demonstrate the steps of utilizing FEM in this equation and then we will obtain numerical results [16, 20].

Initially, we discretize the domain of the equation into a number of elements to highlight the numerical approach. We start this procedure by diving the function into elements in the domain $[0, L]$, e.g. $0 = t_0 < t_1 < ... < t_N = L$, be a uniform partition. Let $\phi = \phi(x)$ be the basis or test function. For each element, we have two nodes t_i and t_{i+1}.

$$\int_{t_i}^{t_{i+1}} \left(u'' + \delta u' + \alpha u - \beta u^3\right) \phi \, dt - \int_{t_i}^{t_{i+1}} \gamma \phi \cos(\omega t) \, dt = 0, \tag{2.9}$$

where ϕ_i are given as,

$$\begin{cases} \phi_i(t) = \dfrac{t - t_{i-1}}{t_i - t_{i-1}}, & t \in [t_{i-1}, ti] \\ \phi_i(t) = \dfrac{t_{i+1} - t}{t_{i+1} - t_i}, & t \in [t_i, t_{i+1}] \\ \phi_i(t) = 0, \quad otherwise \end{cases}$$

Because of the Neumann boundary condition for $i = N$, we also introduce the following basis function,

$$\begin{cases} \phi_N(t) = \dfrac{t - t_{N-1}}{t_N - t_{N-1}}, & t \in [t_{N-1}, t_N] \\ \phi_N(t) = 0, \ otherwise \end{cases}$$

Substituting the basis functions into equation (2.9) where,

$$h = t_{i+1} - t_i, \tag{2.10}$$

we obtain k_{ij}^e, for each element and it is given as,

$$k_{ij}^e = \int_{t_i}^{t_{i+1}} \left(-\phi_i', \phi_j' + \delta\phi_i\phi_j + \alpha\phi_i\phi_j - \beta\phi_i^3\phi_j\right) \mathrm{dt}$$

and also g_i^e is given as,

$$g_i^e = \gamma \int_{t_i}^{t_{i+1}} \phi_j \cos(\omega t) dt - \phi_j u' |_{t_i}^{t_{i+1}}$$

We finally obtain the system,

$$\mathbf{K}\,\mathbf{U} = \mathbf{F} + \mathbf{G}, \text{ where } \mathbf{U} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \ldots \\ u_N \end{pmatrix}$$

Where $\mathbf{K}$ is the coefficient matrix, $\mathbf{U}$ is the vector of the unknowns, and $\mathbf{F} + \mathbf{G}$ is the right hand side of the system under consideration. Due to the essential boundary conditions, we know that $u(0) = u_1 = 0$. We should not forget that the approximate solution is, $u_h = \sum_{i=1}^{N} u(t_i)\phi_i$, where $u(t_i) = u_i$ and $\phi_i(t) = \phi_i$.

The corresponding numerical solution for $u(0) = u'(L) = 0$, $f(t) = 0.2\cos t$, $\delta = 0$, $\alpha = 0.06$, $\gamma = 0.2$, $\omega = 1$ and $\beta = 0.0001$ is shown for $N = 160$ nodes, and it is compared with the finite difference method (FDM) and the Runge–Kutta shooting methods until satisfy the boundary condition $u'(L) = 0$, where $L = 15$, as depicted in Figure 2.1 [16].

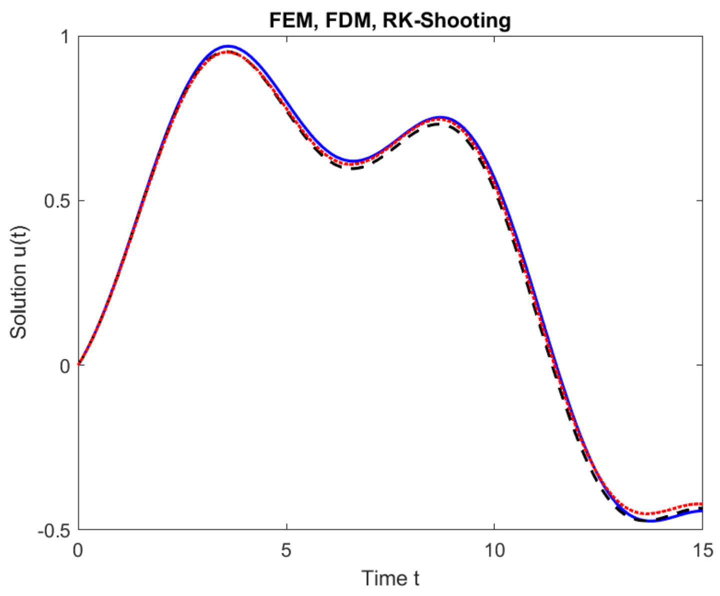

Figure 2.1: Duffing equation with three methods, FEM, FDM and Runge–Kutta shooting method, for $u(0) = 0, u'(L) = 0$, $f(t) = 0.2\cos(\omega t)$, $\delta = 0$, $\alpha = 0.06$, $\gamma = 0.2$, $\omega = 1$ and $\beta = 0.0001$.

Table 2.1: Local error estimates for the three methods, FEM, FDM and RK-shooting.

i	FEM	FDM	$\lvert\delta_{(FEM-FDM)}\rvert$	RK-s	$\lvert\delta_{(FEM-RK-s)}\rvert$
1	0	0	0	0	0
20	0.5888	0.5840	0.0048	0.5815	0.0073
40	0.9677	0.9502	0.0174	0.9492	0.0184
60	0.7029	0.6758	0.0271	0.6850	0.0179
80	0.6647	0.6464	0.0183	0.6608	0.0038
100	0.7094	0.6819	0.0276	0.6979	0.0115
120	0.1033	0.0625	0.0408	0.0854	0.0179
140	0.4427	0.4526	0.0098	0.4290	0.0138
160	0.4414	0.4339	0.0075	0.4214	0.02

Table 2.2: L^2 error estimates for the three methods, FEM, FDM and RK-shooting.

$\lvert\lvert u_{FEM} - u_{FDM}\rvert\rvert_{L^2}$	$\lvert\lvert u_{FEM} - u_{RK-s}\rvert\rvert_{L^2}$
0.0867	0.0563

Additionally, we compare the numerical solution obtained from the FEM with the analytical solution of the problem under consideration. More precisely, solving analytically equation (2.8) with boundary conditions, $u(0) = 1$, $u'(L) = -0.244$, zero forcing term, $f(t) = 0$ and all the other parameters as described above, we show the comparison between the obtained exact solution,

$$u'(t) = \pm\sqrt{E - \alpha u^2 - \frac{\beta}{2}u^4}, \tag{2.11}$$

and the corresponding numerical solution. Where E, is the energy of the system and we use the positive solution (positive velocity) of equation (2.11) for the comparison between the analytical and the numerical solution. The maximum error obtained from the comparison is, $||\delta||_\infty = 1.4 \times 10^{-3}$.

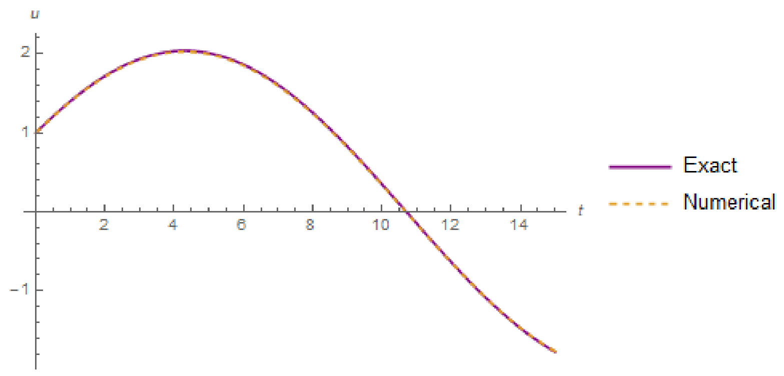

Figure 2.2: Comparing exact and numerical solution for the Duffing equation for $u(0) = 1, u'(L) = -0.244,\ f(t) = 0,\ \delta = 0,\ \alpha = 0.06$, and $\beta = 0.0001$.

All the numerical results of this section were obtained with the help of Matlab and Mathematica software packages. The Matlab code can be found in the Appendix.

2.1.4 Two–Dimensional finite elements

Let's consider a two–dimensional problem (Dirichlet problem) for the Poisson equation,

$$u_{xx}(x,y) + u_{yy}(x,y) = f(x,y),\ \ in\,\Omega, u = 0,\ \ on\,\partial\Omega \tag{2.12}$$

Where Ω is a connected open region in the (x,y), $\partial\Omega$ is the boundary of Ω. The weak form of this problem is,

$$(f,v) := \int_\Omega fv\ d\Omega = -\int_\Omega \nabla u \cdot \nabla v\ d\Omega := -\alpha(u,v), \tag{2.13}$$

where, ∇ denotes the gradient operator and $\cdot$ denotes the dot or inner product in the two–dimensional space.

The procedure to construct a basis function for two-dimensional finite elements has similarities with the one-dimensional case. We denote as $u(x,y)$ and $v(x,y)$ the components of the velocity. In two-dimensional, triangular and quadrilateral elements are the most commonly used ones.

- $N_i(x, y)$ are the **nodal basis** and $\sum_{i=1}^{n} N_i(x, y) = 1$
- (x_i, y_i) are the **nodes**
- $u_i(x, y)$ and $v_i(x, y)$ are the **nodal values**
- $u_h(x, y) = \sum_{i=1}^{n} u_i(x, y)N_i(x, y)$ and $v_h(x, y) = \sum_{i=1}^{n} v_h(x, y)N_i(x, y)$

2.1.5 Triangular elements

In determining the shape functions $N_i(i = 1, 2, 3)$ we assume that the shape functions are linear functions of x and y.

By mapping the physical coordinates (x, y) of triangular element to the local natural coordinates (ξ, η) i.e. $(0, 0)$, $(1, 0)$ and $(0, 1)$ we can use the above general form: $N = A + B\xi + C\eta$

$$\begin{aligned}N_1 &= 1 - \xi - \eta\\ N_2 &= \xi\\ N_3 &= \eta\end{aligned}$$

2.1.6 Quadrilateral elements

The dimension of the element is defined here as $2a \times 2b$. A local natural coordinate system (ξ, η) with its origin located at the centre of the rectangular element is defined. The relationship between the physical coordinate (x, y) and the local natural coordinate system (ξ, η) is given by

$$\xi = x/a, \quad \eta = y/b$$

Master element coordinates, ξ and η, vary between -1 and 1. Local node numbering starts from the lower left corner and goes CCW.
General form: $N = A + B\xi + C\eta + D\xi\eta$

$$\begin{aligned}N_1 &= \frac{1}{4}(1 - \xi)(1 - \eta)\\ N_2 &= \frac{1}{4}(1 + \xi)(1 - \eta)\\ N_3 &= \frac{1}{4}(1 + \xi)(1 + \eta)\\ N_4 &= \frac{1}{4}(1 - \xi)(1 + \eta)\end{aligned}$$

Shape functions can be determined by considering the general form and using the Kronecker–delta property. In two-dimensional (x, y) coordinates can be written in terms of (ξ, η) coordinates by using the previously defined 2D shape functions as follows:

$$x = \sum_{i=1}^{4} N_i(\xi, \eta) x_i$$

$$y = \sum_{i=1}^{4} N_i(\xi, \eta) y_i$$

We are now able to express ξ and η derivatives of the i^{th} shape function in terms of derivatives with respect to x and y as follows

$$\begin{bmatrix} \partial N_i/\partial \xi \\ \partial N_i/\partial \eta \end{bmatrix} = \mathbf{J} \begin{bmatrix} \partial N_i/\partial x \\ \partial N_i/\partial y \end{bmatrix}$$

where J is the Jacobian matrix defined by

$$\mathbf{J} = \begin{bmatrix} \partial x/\partial \xi & \partial y/\partial \xi \\ \partial x/\partial \eta & \partial y/\partial \eta \end{bmatrix}$$

Substituting the interpolation of the coordinates into the above equation, and after manipulations we conclude:

$$\begin{bmatrix} \partial N_i/\partial x \\ \partial N_i/\partial y \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \partial N_i/\partial \xi \\ \partial N_i/\partial \eta \end{bmatrix}$$

which gives the relationship between the differentials of the shape functions with respect to x and y with those with respect to ξ and η.

Next, we will apply the finite elements method to the **Poisson equation** (2.12) with quadrilateral elements. The procedure for trilingual elements is similar [13].

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \\ u &= u_D \text{ on } \Gamma_{\mathrm{D}} \\ \nabla u \cdot n &= g \text{ on } \Gamma_{\mathrm{N}} \end{aligned} \tag{2.14}$$

where, $n = (n_x, n_y)$ is the unit normal vector on the boundary Γ_{N}, and

$u_D = \begin{cases} 1, & x = 0 \\ 0, & x = 1 \end{cases}, \quad g = \sin(5x) \text{ for } y = 0, 1 \ ,$

$$f(x, y) = c \exp\left(-\frac{(x-\alpha)^2 + (y-b)^2}{d} \right), \tag{2.15}$$

where, $c = 10.$, $\alpha = b = 0.5$ and $d = 0.02$. We know that,

$$-\int_{\Omega} (\Delta u) v \ \mathrm{d}\Omega = \int_{\Omega} \nabla u \cdot \nabla v \ \mathrm{d}\Omega - \int_{\partial\Omega} (\nabla u \cdot n)\, v \,\mathrm{ds},$$

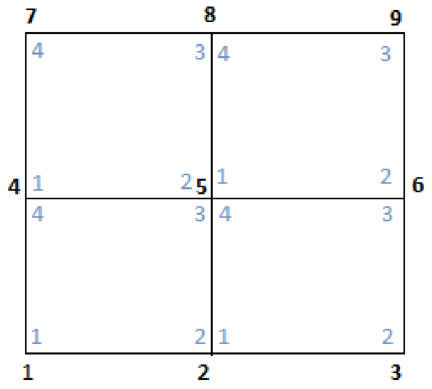

Figure 2.3: We visualize four quadrilateral elements on a simplified FEM grid.

Expressing equation (2.14) in the standard notation $a(u, v) = L(v)$, is relatively easy to do and is given as,

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \ \mathrm{d}\Omega$$
$$L(v) = \int_{\Omega} f v \ \mathrm{d}\Omega + \int_{\Gamma_N} (\nabla u \cdot n)\, v \ \mathrm{d}s,$$

where, $u_S = \sum_{j=1}^{N} u_j N_j(x, y)$, and N denotes the number of nodes, $v = N_i(x, y)$ where N_i denotes the nodal basis.

- $\int_{\Omega} \nabla u \cdot \nabla v \ \mathrm{d}\Omega = \int_{\Omega} \left(\frac{\partial v}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial u}{\partial y} \right) d\Omega$
- $\int_{\Gamma} v(n \cdot \nabla u) ds = \int_{\Omega} \left(n_x \frac{\partial u}{\partial x} + n_y \frac{\partial u}{\partial y} \right) N_i ds$
- $\int_{\Omega} v f \ \mathrm{d}\Omega = \int_{\Omega} N_i f \ \mathrm{d}\Omega$

Where, K^e_{ij} integral can be written in terms of the generalized element coordinates ξ and η as follows,

$$K^e_{ij} = \int_{\Omega^e} \left[\left(J^{-1}_{11} \frac{\partial N_i}{\partial \xi} + J^{-1}_{12} \frac{\partial N_i}{\partial \eta} \right) \left(J^{-1}_{11} \frac{\partial N_j}{\partial \xi} + J^{-1}_{12} \frac{\partial N_j}{\partial \eta} \right) \right.$$
$$\left. + \left(J^{-1}_{21} \frac{\partial N_i}{\partial \xi} + J^{-1}_{22} \frac{\partial N_i}{\partial \eta} \right) \left(J^{-1}_{21} \frac{\partial N_j}{\partial \xi} + J^{-1}_{22} \frac{\partial N_j}{\partial \eta} \right) \right] |J^e| \, d\xi d\eta,$$

where, $|J^e|$ is the determinant of the Jacobian of the transformation from physical space to the normalized one, and the terms J^{-1}_{ij}, $i, j = 1, 2$ are the metrics of the transformation. In Figure 2.4, we present the results obtained from the problem and boundary conditions discussed above, equation (2.14), with a number of cells 128 and 81 nodes, respectively.

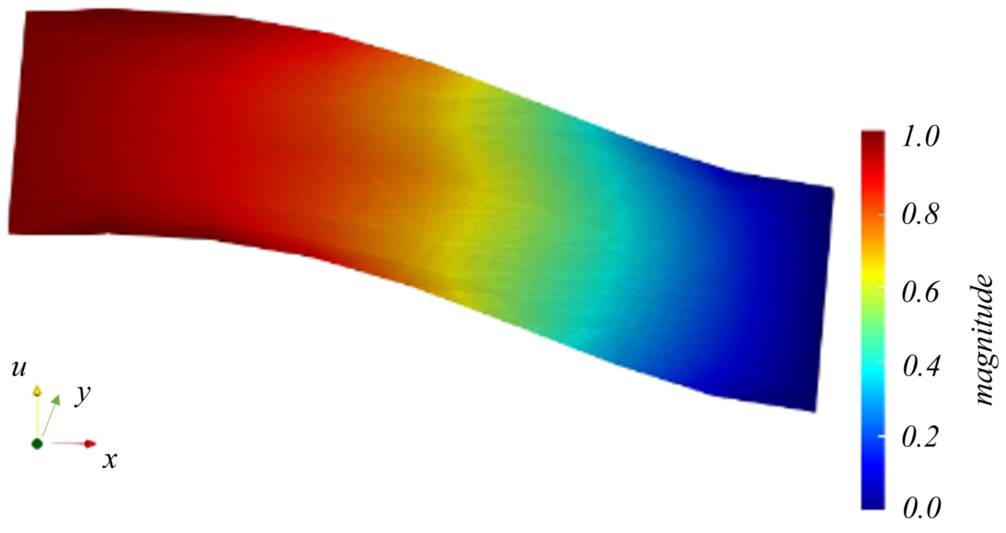

Figure 2.4: Numerical solution of the Poisson equation with FEM, solution of the problem under consideration, equation (2.14)

In next chapters, we are going to study the **Stokes** and **Navier-Stokes** equations with higher order elements with more nodes, as shown below. For quadrilateral elements we have a general form,

$$N = A + B\xi + C\eta + D\xi\eta + E\xi^2 + F\eta^2 + G\xi^2\eta + H\xi\eta^2 + I\xi^2\eta^2$$

For triangular elements we have a general form,

$$N = A + B\xi + C\eta + D\xi\eta + E\xi^2\eta + F\xi\eta^2$$

For the **triangular Taylor Hood** ($P_2 - P_1$) elements, that we are going to use later, it has been proven that [12],

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} = \begin{bmatrix} L_1 \left(2L_1 - 1\right) \\ L_2 \left(2L_2 - 1\right) \\ L_3 \left(2L_3 - 1\right) \\ 4L_1 L_2 \\ 4L_2 L_3 \\ 4L_3 L_1 \end{bmatrix}$$

for both of the components of the velocity and linear polynomials for the pressure,

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix},$$

where,

$$L_i = a_i + b_i x + c_i y, \quad i = 1, 2, 3$$

In this section, we are analysing basic FEM principals and discussing about shape functions for quadrilateral and triangular elements. In the next chapter, we discuss about the FEM for the Stokes and Navier–Stokes problems.

CHAPTER 3

The Stokes and Navier–Stokes Problem

3.1 The Stokes Problem

The linear Stokes equations are the limiting case of zero Reynolds number for the Navier–Stokes equations. The Stokes equations have attracted a substantial attention from researchers because of its close relation with the nonlinear Navier–Stokes equations. Initially, we consider the stationary Stokes problem for incompressible flow. Ω is a bounded open set of $\mathbb{R}^n$ (where $n = 2, 3$) with regular boundary and $\boldsymbol{f}$ is a square integrable function on Ω. We seek a solution $(\boldsymbol{u}, p) \in H_0^1(\Omega)^2 \times (L^2(\Omega)/\mathbb{R})$ of the problem [17, 23],

$$\begin{cases} -\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \;\; in \;\; \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 \;\; in \;\; \Omega, \\ \boldsymbol{u} = 0 \;\; on \;\; \partial\Omega. \end{cases} \tag{3.1}$$

Here, $\boldsymbol{u} = (u_x, u_y)$ denotes the velocity vector, whereas $\boldsymbol{f} = (f_x, f_y)$ stands for the body force vector and $\boldsymbol{v}$ =(v_x, v_y), finally ν is the fluid kinematic viscosity, considered constant in the domain Ω. Based on this problem, we will introduce error estimates and we will briefly discuss about the uniqueness of the solution for this problem [5]. According to the finite element analysis, we end up with the following weak form.

$$\begin{cases} a(\boldsymbol{u}, \boldsymbol{v}) + b(p, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), & \forall\, \boldsymbol{v} \in H_0^1(\Omega)^n,\ \boldsymbol{u} \in H_0^1(\Omega)^n, \\ b(\boldsymbol{u}, q) = \int_\Omega (\nabla \cdot \boldsymbol{u}) q \, \mathrm{d}\Omega = 0, & \forall\, q \in H^1(\Omega),\ p \in H^1(\Omega), \end{cases} \tag{3.2}$$

where, $a(\boldsymbol{u}, \boldsymbol{v}) = \nu \displaystyle\int_\Omega \nabla\boldsymbol{u} \cdot \nabla\boldsymbol{v} \, d\Omega$ and $b(p, \boldsymbol{v}) = \displaystyle\int_\Omega p\, (\nabla \cdot \boldsymbol{v}) \, d\Omega$. The kinematic viscosity $\nu = \dfrac{\mu}{\rho}$ is considered constant, where μ is the dynamic viscosity and ρ is the fluid density.

The trial and test spaces V and V_0 are defined as,

$$\begin{aligned} V &= \left\{ \boldsymbol{v} \in H^1(\Omega) : \boldsymbol{v} = \boldsymbol{u}_0 \text{ on } \partial\Omega \right\}, \\ V_0 &= \left\{ \boldsymbol{v} \in H^1_0(\Omega) : \boldsymbol{v} = \boldsymbol{0} \text{ on } \partial\Omega \right\}. \end{aligned}$$

where $H^1(\Omega)$ is a Hilbert space. More details can also be found in the Appendix.

Given two finite dimensional subspaces, $V_h \subset H^1(\Omega)^n$ and $Q_h \subset H^1(\Omega)$ the corresponding discrete form is,

$$\begin{cases} a\left(\boldsymbol{u}_h, \boldsymbol{v}_h\right) + b\left(p_h, \boldsymbol{v}_h\right) = \left(\boldsymbol{f}, \boldsymbol{v}_h\right), & \forall\, \boldsymbol{v}_h \in V_{0h},\ \boldsymbol{u}_h \in V_{0h}, \\ b\left(\boldsymbol{u}_h, q_h\right) = 0, & \forall\, q_h \in Q_h,\ p_h \in Q_h, \end{cases} \tag{3.3}$$

where, $V_{0h} = \{\boldsymbol{v}_h \in V_h : \boldsymbol{v}_h \mid_{\partial\Omega} = 0\}$.

Definition 3.1.1 (**Hilbert space**)**.** A Hilbert space is a vector space whose topology is defined using an inner–product. One example of a Hilbert space is $L^2(0,1)$ with inner-product $(\cdot,\cdot)$. Hilbert spaces are complete metric spaces.

Definition 3.1.2 (**Sobolev space**)**.** A Sobolev space is a vector space of functions equipped with a norm that is a combination of L^p–norms of the function itself and its derivatives up to a given order. We define the Sobolev spaces as,

$$H^k(\Omega) = W^{k,p}(\Omega) = \left\{ u \in L^p(\Omega) : D^\alpha u \in L^p(\Omega),\ \forall\ |\alpha| \leqslant k \right\},$$

where k is a non–negative integer and $D^\alpha u$ are the weak derivatives of the function u. We define the Sobolev norm as follows,

$$\|u\|_{W^{k,p}(\Omega)} := \begin{cases} \left(\sum_{|\alpha| \leqslant k} \|D^\alpha u\|^p_{L^p(\Omega)} \right)^{\frac{1}{p}} & 1 \leqslant p < \infty \\ \max_{|\alpha| \leqslant k} \|D^\alpha u\|_{L^\infty(\Omega)} & p = \infty \end{cases}$$

In the cases where $p = 2$ Sobolev space is a Hilbert space. The Sobolev space $W^{1,2}(\Omega)$ is also denoted by $H^1(\Omega)$ with norm

$$\|u\|_{H^1} = \left(\int_\Omega |u|^2 + |\nabla u|^2 \right)^{\frac{1}{2}}$$

Two cases are analyzed for both triangular and quadrilateral elements depending on the number of nodes on each element [5]. We focus only on the Taylor–Hood method (six node triangular elements), second order polynomials for the velocity and first order polynomials for the pressure at each element $(P_2 - P_1)$.

After finding a solution, for the problem under consideration, it is important to show that it is stable and how the input data affect it. This can be done using the inf–sup condition, the Ladyzhenskaya–Babuska–Brezzi (LBB) condition. This is a condition for saddle point problems i.e. problems arising in different types of discretization of equations. Convergence is ensured for most discretization schemes for positive definite problems but for saddle point problems there are still discretizations that are unstable, due to spurious oscillations [25]. In these cases a better approach is the adaptation of the computational grid [18]. We further discuss for the BB condition, introducing the following theorem.

Theorem 3.1.1. *If Ω is polygonal and $\Omega_h = \Omega$, $\Omega_h = \bigcup_i T_i$, where T_i are the triangles and h denotes the length of greatest triangle side, if all triangles have at least one vertex which is not on $\partial\Omega$, if V_h, Q_h are chosen as in the Taylor–Hood method, then there exists a constant C, independent of h, such that,*

$$sup_{\boldsymbol{v}_h \in V_{0h}} \frac{(\boldsymbol{v}_h, \nabla q_h)}{(\boldsymbol{v}_h, \boldsymbol{v}_h)^{\frac{1}{2}}} \geq C\left(\nabla q_h, \nabla q_h\right)^{\frac{1}{2}}, \quad \forall q_h \in Q_h. \tag{3.4}$$

Where the Sobolev spaces used in the theorem are defined above (3.3). This theorem follows the idea of the BB condition and the proof depends on the choice of the elements and can be found in [5]. One of the most important questions in solving such a problem is that of existence and uniqueness of the solution. In this case we focus on the discrete form of the problem under consideration, equation (3.3) where we can ensure the previous with the following theorem [5].

Theorem 3.1.2. *Under the conditions of theorem 3.1.1 the discrete form, equation 3.3, has a unique solution $(\boldsymbol{u}_h, p_h)$ in $V_{0h} \times (Q_h/\mathbb{R})$.*

Additionally, we are interested in error estimates of the Stokes problem as discussed in the next section.

3.1.1 A Priori Error Estimates

A priori error estimates express the error in terms of the regularity of the exact unknown solution and may give important information about the order of convergence of a finite element method. A posteriori error estimates express the error in terms of computable quantities such as the residual error and possibly the solution of an auxiliary dual problem and contribute to the grid adaptation, as described in next chapter [18]. A theorem that provides a priori error estimates for the discrete form of the stationary Stokes problem using Taylor-Hood elements ($P_2 - P_1$) is as follows.

Theorem 3.1.3. *Let Ω be a polygon and $\Omega_h = \Omega$ for all h. We assume that each element of $\mathcal{T}_h$ (set of triangles) has at least one vertex not on the boundary. Then the following inequalities are valid,*

$$\begin{aligned} \|\nabla(\boldsymbol{u} - \boldsymbol{u}_h)\| &\leqslant h^2 K \left(\|\boldsymbol{u}\|_{H^3(\Omega)^N} + \|p\|_{H^2(\Omega)/R}\right) \\ \|\nabla(p - p_h)\| &\leqslant hK \left(\|\boldsymbol{u}\|_{H^3(\Omega)^N} + \|p\|_{H^2(\Omega)/R}\right) \end{aligned}$$

Similar inequalities can be found in the case where we have quadrilaterals [5].

3.1.2 The Backward Facing Step Problem

The backward facing step (BFS) is a “test problem” widely known for its application on internal flows. In this problem, flow separation is caused due to sudden changes in the geometry. This creates a recirculation zone close to the step wall, and downstream a reattachment point. In a two–dimensional BFS geometry, the fluid flow can be distinguished into three regions, the shear layer, the separation bubble and the reattachment zone [1, 22].

Due to the adverse pressure gradient that develops in the thin shear layer, the characteristics of a BFS flow begin with an upstream boundary layer that separates at the edge of the backward facing step. The region where the shear layer develops is referred to as the shear layer region. This flow causes the formation of a recirculation zone, which is located between the shear layer and the adjacent wall. Eventually, the shear layer curves down towards the wall and reattaches at the so called reattachment point. The horizontal distance between the step and the reattachment point is defined as the “reattachment length”. Due to the oscillatory motion of the shear layer, the reattachment length is unsteady. Consequently, the reattachment point spreads within a zone, called reattachment zone [1, 22]. In this problem the flow parameters of interest are,

$$\begin{cases} u = \text{ horizontal velocity component, } v = \text{ vertical velocity component,} \\ L = \text{ length, } h = \text{ the step height, } H = \text{ the whole height,} \\ \mu = \text{viscosity}, \rho = \text{density}, \end{cases}$$

with a typical set of boundary conditions,

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \Gamma_D \\ &\nabla \boldsymbol{u} \cdot \boldsymbol{n} + p\,\boldsymbol{n} = \boldsymbol{g} \text{ on } \Gamma_N \end{aligned}$$

where,

- Γ_D is referred to the **Dirichlet** or essential boundary conditions

- Γ_N is referred to the **Neumann** or natural boundary conditions

where $\boldsymbol{n}$ is the associated normal vector. For further computations, it is easier to write equations (3.1) to the following form for finding $(\boldsymbol{u}, p) \in W$ such that,

$$a((\boldsymbol{u}, p), (\boldsymbol{v}, q)) = L(\boldsymbol{v}, q),$$

for all $(\boldsymbol{v}, q) \in W$, where

$$\begin{aligned} a((\boldsymbol{u}, p), (\boldsymbol{v}, q)) &= \int_\Omega [\nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} - (\nabla \cdot \boldsymbol{v})p + (\nabla \cdot \boldsymbol{u})\, q] \; d\Omega, \\ L(\boldsymbol{v}, q) &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \; d\Omega + \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{v} \; ds. \end{aligned}$$

The space W should be a mixed (product) function space $W = V \times Q$, such that $\boldsymbol{u} \in V$ and $q \in Q$. We will use the Galerkin FEM to analyse the velocity and pressure on this test problem.

The obtained numerical results are shown in the following figures. Figure 3.3 shows the velocity field in the domain for the backward step. We also present the streamlines in the domain in order to visualize the recirculation close to the step. It is observed that the maximum velocity is at the entrance of the channel and the velocity drops rapidly as the domain expands. In Figure 3.4 we visualize the pressure field with the classical Galerkin FEM. It is observed that the pressure field is smooth even with the Taylor-Hood elements.

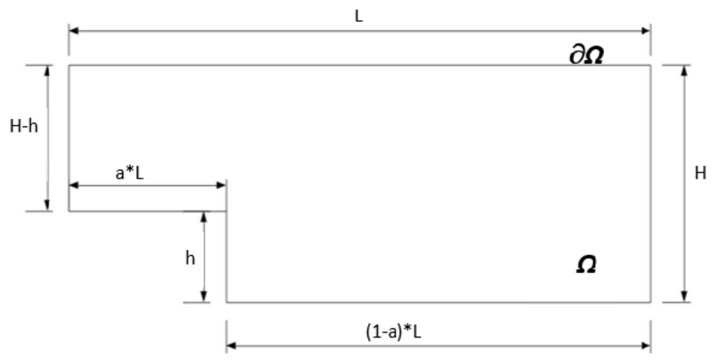

Figure 3.1: The domain Ω and the dimensions of the backward step.

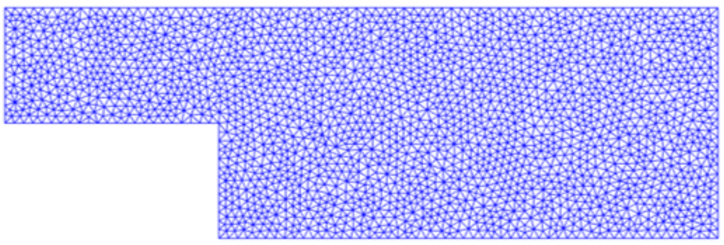

Figure 3.2: the computational mesh composed of approximately 6060 finite elements.

3.2 The Navier–Stokes Problem

Most of every real situation in fluid flows is characterized by the Navier–Stokes equations that are the model of nonlinear PDEs, so it is recognizable the importance to solve this particular equation. Because of the nonlinearity of the problems that are described by the Navier–Stokes equations, an exact solution is impossible to be obtained. However, it is very useful to describe and analyse the physics of fluid flow problems and also more complex materials if these equations can be solved. The finite element method constitutes an effective way to find an approximate solution to these equations.

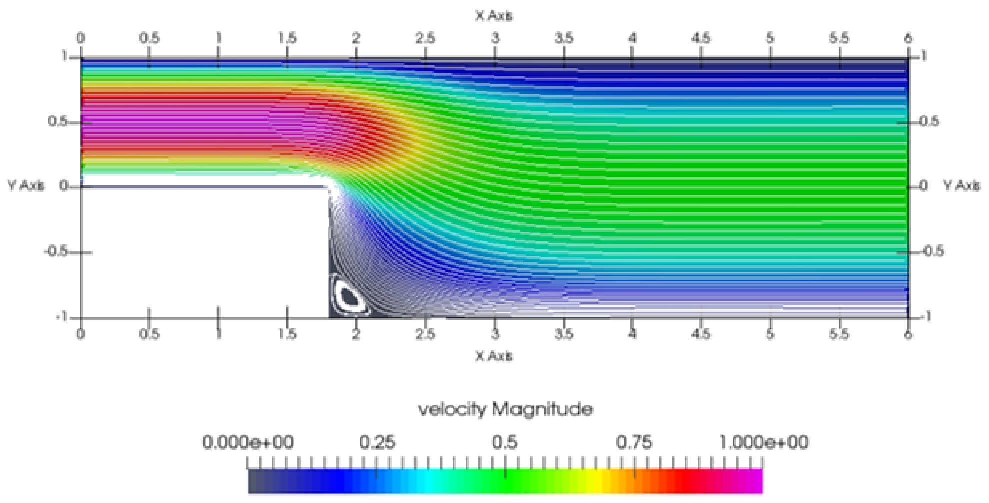

Figure 3.3: The velocity profile of the Stokes equation on the backward step problem.

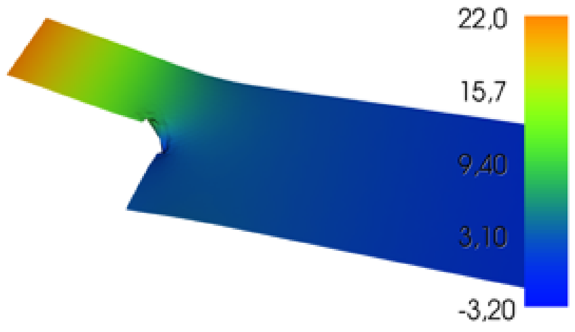

Figure 3.4: The pressure profile of the Stokes equation on the backward step problem.

In general the time–dependent, incompressible Navier–Stokes equations are:

$$
\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} &\quad \text{in } \Omega \times (0,T) \\
\nabla \cdot \boldsymbol{u} = 0 &\quad \text{in } \Omega \times (0,T) \\
\boldsymbol{u} = \boldsymbol{0} &\quad \text{on } \partial\Omega \times (0,T) \\
\boldsymbol{u}(\cdot,0) = \boldsymbol{u}_0 &\quad \text{in } \Omega
\end{aligned}
\tag{3.5}
$$

Here, $\boldsymbol{u}$ represents the velocity vector, p the zero–mean pressure, $\boldsymbol{f}$ an external force, and ν the kinematic viscosity (considered constant). These equations describe the motion of an incompressible fluid in the domain Ω. Compared to the Stokes equations we have to deal with an additional nonlinearity and a time derivative. The weak formulation of these equations are obtained by multiplying the momentum equation with a test function $\boldsymbol{v}$, defined in a suitable space V, and integrating both members with respect to the domain Ω.

Integrating by parts and using the Gauss' theorem, the formulation in equation (3.5) can be written as,

$$\begin{aligned}&\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t}\cdot \boldsymbol{v}\ \mathrm{d}\Omega + \nu\int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\ \mathrm{d}\Omega + \int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\Omega - \int_\Omega p\nabla\cdot\boldsymbol{v}\ \mathrm{d}\Omega = \\ &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\ \mathrm{d}\Omega + \int_{\partial\Omega}\left(\nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\right)\cdot\boldsymbol{v}\ \mathrm{d}s, \quad \forall\ \boldsymbol{v}\in V,\end{aligned} \tag{3.6}$$

where $\boldsymbol{n}$ is the unit normal vector. In a similar manner, the continuity equation is multiplied with a test function q belonging to a space Q and integrated in the domain Ω,

$$\int_\Omega q(\nabla\cdot\boldsymbol{u})\ \mathrm{d}\Omega = 0, \quad \forall\ q\in Q.$$

The space functions are chosen as follows,

$$\begin{aligned} V &= \left[H_0^1(\Omega)\right]^d = \left\{v\in\left[H^1(\Omega)\right]^d:\ v=0 \text{ on } \Gamma_D\right\} \\ Q &= L^2(\Omega)\end{aligned}$$

Because of the set of boundary conditions,

$$\begin{aligned}&\boldsymbol{u} = \boldsymbol{0} \text{ on } \Gamma_D \\ &\nabla\boldsymbol{u}\cdot\boldsymbol{n} + p\,\boldsymbol{n} = \boldsymbol{g} \text{ on } \Gamma_N\end{aligned}$$

the integral on the boundary can be written as,

$$\int_{\partial\Omega}\left(\nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\right)\cdot\boldsymbol{v}\ \mathrm{d}s = \int_{\Gamma_D}\left(\nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\right)\cdot\boldsymbol{v}\ \mathrm{d}s + \int_{\Gamma_N}\left(\nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\right)\cdot\boldsymbol{v}\ \mathrm{d}s = \int_{\Gamma_N}\boldsymbol{g}\cdot\boldsymbol{v}\ \mathrm{d}s$$

where,

- $\displaystyle\int_{\Gamma_D}\left(\nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\right)\cdot\boldsymbol{v}\ \mathrm{d}s = 0$

- $\displaystyle\int_{\Gamma_N}\left(\nu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n}\right)\cdot \boldsymbol{v}\ \mathrm{d}s = -\boldsymbol{g}\cdot\boldsymbol{v}$

Finally, the weak formulation of the Navier–Stokes equations is,

$$\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t}\cdot\boldsymbol{v}\ \mathrm{d}\Omega + \nu\int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\ \mathrm{d}\Omega + \int_\Omega(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\Omega - \int_\Omega p\nabla\cdot\boldsymbol{v}\ \mathrm{d}\Omega = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\ \mathrm{d}\Omega + \int_{\Gamma_N}\boldsymbol{g}\cdot\boldsymbol{v}\ \mathrm{d}s, \quad \forall\ \boldsymbol{v}\in V,$$

$$\int_\Omega q\nabla\cdot\boldsymbol{u}\ \mathrm{d}\Omega = 0, \quad \forall\ q\in Q.$$

The existence and uniqueness of the solution will be discussed by the following theorem,

Theorem 3.2.1. *[2] If $f\in\left[L^2\left(0,T;V'\right)\right]^d$ and $u_0\in H$, there exists a weak solution to the Navier–Stokes equations* (3.5) *that satisfies,*

$$\boldsymbol{u}\in L^2(0,T;\boldsymbol{V})\cap L^\infty(0,T;\boldsymbol{H})$$

where,

$$\boldsymbol{V} = \left\{\boldsymbol{v}\in\left[H_0^1(\Omega)\right]^d : \operatorname{div}\boldsymbol{v}=0\right\}$$
$$\boldsymbol{H} = \left\{\boldsymbol{v}\in\left[L_0^2(\Omega)\right]^d : \operatorname{div}\boldsymbol{v}=0\right\}$$

In the case of space dimension $d=2$ this solution is unique and

$$\boldsymbol{u}\in\mathcal{C}(0,T;\boldsymbol{H})$$
$$\boldsymbol{u}'\in L^2\left(0,T;\boldsymbol{V}'\right)$$

In three dimensions ($d=3$) uniqueness is an open question. In this section, we discussed about the time–dependent Navier–Stokes and continuity equations, where is the most general problem under consideration. In the next section, we focus on the steady–state form of the equations.

3.2.1 An Application to the Poiseuille flow

We study the two–dimensional incompressible flow between two long parallel plates (Poiseuille flow) with no–slip condition on both walls which are spaced apart in height $2h$, Figure 3.5. We assume constant flow, with constant density ρ and viscosity μ. Fluid is introduced with a parabolic velocity profile at the inlet of the domain, given by the expression $u_x(y) = (4\,y\,(y-1))$, for the u_x–velocity component, where the u_y–velocity is considered zero, at the inlet.

The two–dimensional Navier–Stokes and continuity equations in Cartesian coordinates, (x, y) are,

$$\begin{aligned}\rho\left(\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y}\right) &= -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2}\right) + F_x,\\ \rho\left(\frac{\partial u_y}{\partial t} + u_x\frac{\partial u_y}{\partial x} + u_y\frac{\partial u_y}{\partial y}\right) &= -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2}\right) + F_y,\\ \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} &= 0.\end{aligned} \tag{3.7}$$

Considering that there is no external force field, $\bar{F} = (F_x, F_y)$, the boundary conditions of this problem are written as,

$$\begin{cases}\text{If } y = -h \text{ or } y = +h, \text{ then } u_x(x,y) = 0, \text{ and } u_y(x,y) = 0,\\ F_x = F_y = 0.\end{cases}$$

Providing appropriate assumptions, steady–state flow $\left(\frac{\partial u_x}{\partial t} = \frac{\partial u_y}{\partial t} = 0\right)$, and utilizing the boundary conditions in (3.7), we get the following system of equations,

$$\frac{\partial u_x}{\partial x} = 0, \text{ i.e } u_x = u_x(y). \tag{3.8}$$

$$\rho u_x\frac{\partial u_x}{\partial x} = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2}\right) \tag{3.9}$$

$$0 = -\frac{\partial p}{\partial y}, \text{ i.e } p = p(x). \tag{3.10}$$

Thus, we obtain,

$$\frac{dp}{dx} = c = \mu\frac{d^2 u_x}{dy^2} \tag{3.11}$$

Solving the differential equation (3.9) and applying the boundary conditions yields to the analytical solution of the problem under consideration, that is a parabolic profile as expected,

$$u_x(y) = -\frac{c}{2\mu}(h^2 - y^2). \tag{3.12}$$

The velocity has a parabolic profile, as shown in equation (3.12). If $c > 0$ the maximum velocity value is at the centre of the domain, for $y = 0$ and $u_{x\,max} = -\frac{ch^2}{2\mu}$, that indicates a favorable pressure drop within this region, given by $\frac{dp}{dx} = c = -\frac{2\mu}{h^2}u_{x\,max}$.

So, we obtained the analytical solution for the Poiseuille problem. Additionally, solving the same problem with the help of FEM and the software package FEniCS we obtained the same parabolic profile, as shown in Figure 3.6. The FEM results were obtained for a computational mesh of 2754 triangular elements. The maximum velocity is at the centerline of the domain as discussed above with the analytical solution.

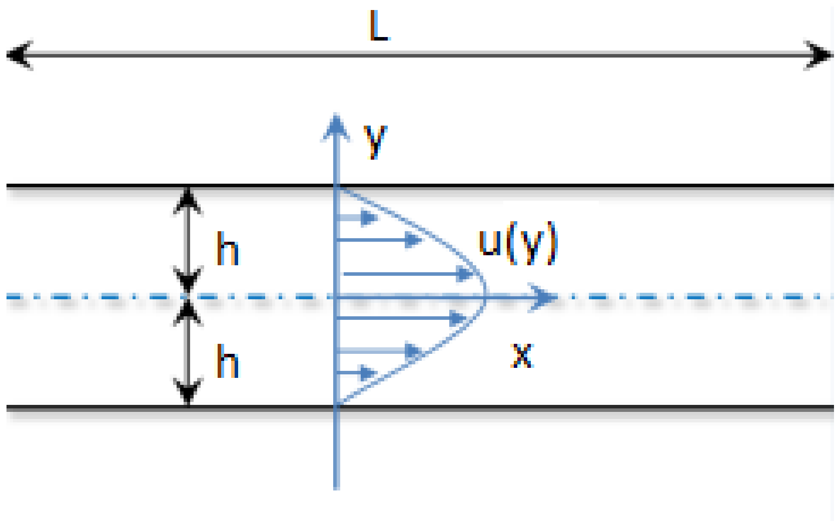

Figure 3.5: Poiseuille flow, the fluid flow between two parallel plates.

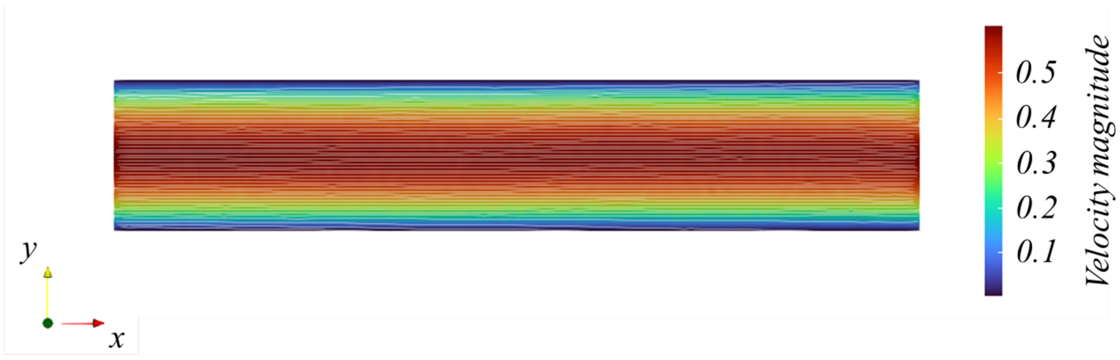

Figure 3.6: The numerical solution of the Poiseuille flow, using FEM and the software package FEniCS.

CHAPTER 4

Modifications of the classical Galerkin method

4.1 The Disconitinuous Galerkin Method

The discontinuous Galerkin (DG) methods are used in the numerical analysis of differential equations. It is a different method to both the finite element and finite volume methods and all these methods apply to a plethora of problems in fluid dynamics. The basis functions used, are discontinuous. These methods, allowing discontinuities, apply with great flexibility and benefits, handling complex geometries, irregular meshes, and polynomial approximations of different degree in each element [7]. The discontinuous methods are distinguished from the continuous ones in integrating flux terms over interior faces.

The DG methods first arose in solving PDEs in the early 70s, with continuous improvements on elliptic problems, through out the decade [3, 14]. Extensions of the methods, in the 90s, dealt with nonlinear hyperbolic conservation laws, as well as compressible flow. The analysis and development of such methods, is a topic of active research [8].

The following example, finds an approximate solution u_h of an ODE using the Discontinuous Galerkin (DG) method. Consider the initial–value problem [7]:

$$\begin{cases} \dfrac{d}{dt}u(t) = f(t)u(t), \;\; t \in (0,T), \\ u(0) = u_0. \end{cases} \tag{4.1}$$

Initially, we divide the interval $I := (0, T)$, into subintervals $I_i := (t_i, t_{i+1})$, for $i = 0, 1, ..., N-1$. Next, we seek for the approximate solution u_h, which on the interval I_i, is a polynomial of degree at most k^i, requiring that,

$$-\int_{I_i} u_h(s)\frac{d}{ds}v(s)\,ds + \hat{u_h}v\big|_{t_i}^{t_{i+1}} = \int_I u(s)f(s)v(s)\,ds, \tag{4.2}$$

for polynomials v of degree at most k^i, where the quantity $\hat{u_h}$ is ,

$$\hat{u_h} := \begin{cases} u_0, & \text{if } , t_i = 0 \\ \lim_{\varepsilon\to 0} u_h(t_i - \varepsilon), & \text{otherwise.} \end{cases}$$

The goal is to find a suitable definition of the *numerical trace*, $\hat{u_h}$, using discontinuous approximations, u_h and applying the Galerkin weak formulation. The DG FEMs are *consistent* methods. So, when we replace the approximate solution u_h with the exact solution u, in the weak formulation of the equation (4.2), the equation is satisfied. That can be applied if $\widehat{u} = u$.

Multiplying the ODE by u and integrating over $(0, T)$, we get,

$$\frac{1}{2}u^2(T) - \frac{1}{2}u_0^2 = \int_0^T f(s)u^2(s)ds.$$

Substituting $v = u_h$ in the weak formulation, equation (4.2), and integrating by parts we obtain the following,

$$\sum_{i=0}^{N-1}\left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h\right)\Bigg|_{t_i}^{t_{i+1}} = \frac{1}{2}u_h^2\left(T^-\right) + \Theta_h\left(T'\right) - \frac{1}{2}u_0^2 = \int_0^T f(s)u_h^2(s)ds,$$

where,

$$\Theta_h(T) = -\frac{1}{2}u_h^2\left(T^-\right) + \sum_{i=0}^{N-1}\left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h\right)\Bigg|_{t_i}^{t_{i+1}} + \frac{1}{2}u_0^2.$$

The stability is gained when $\Theta_h(T) \geq 0$, so we set,

$$u_h(t) = u_0, \quad t < 0$$

We further introduce the following definitions,

Definition 4.1.1. We define the average quantity of the discrete function u_h as, $\{u_h\} = \frac{1}{2}\left(u_h^- + u_h^+\right)$. Additionally, we define the difference of the discrete function u_h on the faces of each element as, $[u_h] = u_h^- - u_h^+$.

Definition 4.1.2. We define the limit of the discrete function u_h at the faces as, $u_h^{\pm}(t) = \lim_{\varepsilon\to 0} u_h(t \pm \varepsilon)$.

Remark. The above definitions yields to the following equation,

$$\left[u_h^2\right] = 2\left\{u_h\right\}\left[u_h\right].$$

Taking into consideration the above definitions,

$$\begin{aligned}\Theta_h(T) =& -\frac{1}{2}u_h^2\left(T^-\right) + \left(-\frac{1}{2}u_h^2\left(T^-\right) + \hat{u}_h(T)u_h\left(T^-\right)\right) + \sum_{i=1}^{N-1}\left(-\frac{1}{2}\left[u_h^2\right] + \hat{u}_h\left[u_h\right]\right)(t_i) \\ & - \left(-\frac{1}{2}u_h^2\left(0^+\right) + \widehat{u}_h(0)u_h\left(0^+\right)\right) + \frac{1}{2}u_0^2 \\ =& \left(\widehat{u}_h(T) - u_h\left(T^-\right)\right)u_h\left(T^-\right) + \sum_{i=1}^{N-1}\left(\left(\widehat{u}_h - \left\{u_h\right\}\right)\left[u_h\right]\right)(t_i) \\ & - \left(\widehat{u}_h(0) - u_0\right)u_h\left(0^+\right) + \frac{1}{2}\left[u_h\right]^2(0)\end{aligned}$$

Based on the above, we have for the $\widehat{u}_h$,

$$\widehat{u}_h\left(t_i\right) = \begin{cases} u_0, & \text{if } t_i = 0 \\ \left(\left\{u_h\right\} + C^i\left[u_h\right]\right)(t_i), & \text{if } t_i \in (0,T) \\ u_h(T-), & \text{if } t_i = T \end{cases}$$

$C^i \geq 0$ and $C^0 = 1/2$,

$$\Theta_h(T) = \sum_{i=0}^{N-1} C^i\left[u_h\right]^2(t_i) \geq 0.$$

The accuracy of the method depends on the choice of C^i. It can be proven that if we take $C^i = 1/2$, the order of the method at the points t_i is $2k+1$ and for $C^i = 0$, the order is $2k+2$ [10, 14]. However, for $C^i \equiv 1/2$ DG methods are consistent and stable. As a consequence, we can easily handle different types of approximations in different elements.

For higher order problems, the first step is the discretization of the domain of interest in triangles, denoting by $\mathcal{T}$ such triangulation. Then we seek a discontinuous approximate solution u_h, that in each element K of the triangulation $\mathcal{T}$, belongs to the space $V(K)$ [10, 14].

4.1.1 Application to the Poisson and Stokes Equations

Next, we show the methodology of applying the Discontinuous Galerkin to the Poisson and the Stokes equations [19]. We consider a DG method for the Poisson equation with Dirichlet boundary conditions for simplicity.

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= u_D \quad \text{on } \partial\Omega \end{aligned}$$

Next, we rewrite the Poisson equation to the weak form,

$$-\int_\Omega \Delta u\, v \; \mathrm{d}\Omega = \int_\Omega fv \; \mathrm{d}\Omega$$

Assume that we have a mesh $\mathcal{T}$ of Ω with cells $\{K\}$ and split the left integral into sum over cell integrals:

$$-\sum_{K\in\mathcal{T}} \int_K \Delta u\, v \; \mathrm{d}K = \int_\Omega fv \; \mathrm{d}\Omega$$

Integrating by parts,

$$\sum_{K\in\mathcal{T}} \int_K \nabla u \cdot \nabla v \; \mathrm{d}K - \sum_{K\in\mathcal{T}} \int_{\partial K} (\nabla u \cdot \vec{n}) v \; \mathrm{d}s = \int_\Omega fv \; \mathrm{d}\Omega.$$

where $\vec{n}$ is the outward unit normal vector. Before introducing the DG method, it is necessary to point out the definitions relevant to this method,

Definition 4.1.3 (Average and Jump operator)**.** We define the average quantity of the function v as, $\langle v \rangle = \frac{1}{2}(v^+ + v^-)$. We also define the difference (jump) of the function v on the element faces as, $[v\vec{n}] = v^+\vec{n} - v^-\vec{n}$ in Ω and $[v\vec{n}] = v\vec{n}$ on $\partial\Omega$, where $\vec{n}$ is the facet outward unit normal.

Definition 4.1.4 (Jump identity)**.** We define the jump identity of a functions u, v as, $[uv] = [u]\langle v \rangle + \langle u \rangle [v]$ in Ω.

We consider a DG formulation to approximate the problem. For this formulation, the approximation space is made of discontinuous piecewise polynomials, namely,

$$V = \left\{ v \in L^2(\Omega) : v|_K \in Q_p(K) \text{ for all } K \in \mathcal{T} \right\},$$

where $\mathcal{T}$ is the set of all cells K of the mesh, and $Q_p(K)$ is a polynomial space of degree–p defined on a cell K. In order to write the weak form of the problem, we need to introduce appropriate notation. The sets of interior and boundary facets associated with the mesh $\mathcal{T}$ are denoted here as $\mathcal{F}_i$ and $\mathcal{F}_e$, respectively. With v^+, and v^- being the restrictions of $v \in V$ to the cells K^+, K^- that share the same interior facet in $\mathcal{F}_i$ and $\vec{n}^+$, $\vec{n}^-$ the facet outward unit normals from either the perspective of K^+ and K^-, respectively.

With this notation, the weak form associated with the interior penalty formulation for the Poisson equation is presented as follows,

$$\begin{aligned}
a(v,u) &= \sum_{K\in\mathcal{T}} \int_K \nabla v \cdot \nabla u \, \mathrm{d}K \\
&- \sum_{K\in\mathcal{F}_e} \int_K v(\nabla u \cdot \vec{n})\mathrm{d}s - \sum_{K\in\mathcal{F}_e} \int_K (\nabla v \cdot \vec{n})u \, \mathrm{d}s + \sum_{K\in\mathcal{F}_e} \frac{\alpha}{h} \int_K v \cdot u \, \mathrm{d}s \\
&- \sum_{K\in\mathcal{F}_i} \int_K [v\vec{n}] \cdot \langle \nabla u \rangle \mathrm{d}s - \sum_{K\in\mathcal{F}_i} \int_K \langle \nabla v \rangle \cdot [u\vec{n}] \mathrm{d}s + \sum_{K\in\mathcal{F}_i} \frac{\alpha}{h} \int_K [v\vec{n}] \cdot [u\vec{n}] \mathrm{d}s,
\end{aligned}$$

and the right–hand side is presented as,

$$L(v) = \int_\Omega vf \, \mathrm{d}\Omega.$$

Remark. We provide the following equation for the element boundary,

$$\begin{aligned}
\sum_{K\in\mathcal{T}} \int_{\partial K} (\nabla u \cdot \vec{n})\, v \, \mathrm{d}s &= \sum_{K\in\mathcal{F}_e} \int_K (\nabla u \cdot \vec{n})\, v ds \\
+ \sum_{K\in\mathcal{F}_i} \int_K & \left[(\nabla u^+ \cdot \vec{n}^+)\, v^+ + (\nabla u^- \cdot \vec{n}^-)\, v^-\right] ds
\end{aligned} \tag{4.3}$$

The first line of the weak formulation for Poisson equation is similar with the classical Galerkin formulation. The next two lines are associated with the exterior an interior facets, respectively, combining (4.3) and the operators mentioned before to construct a DG formulation [11].

Remark. The terms, $\sum_{K\in\mathcal{F}_e} \frac{\alpha}{h} \int_K v \cdot u \, \mathrm{d}s$ and $\sum_{K\in\mathcal{F}_i} \frac{\alpha}{h} \int_K [v\vec{n}] \cdot [u\vec{n}] \mathrm{d}s$, are artificially added in the formulation. The constant α, is a stabilization parameter that should be chosen large enough such that the bilinear form $a(\cdot,\cdot)$ is stable and continuous. Where, h is a measure for the average of the mesh size defined as $h = (h^+ + h^-)/2$, for the two neighbouring cells K^+ and K^-, with the given interior facet.

When applying the above formulation, e.g. the DG FEM, using the software program FeniCS for the Poisson equation, we obtain the results shown in Figure 4.2. In the square domain we apply homogeneous Dirichlet conditions on the boundary and internally a Gaussian distribution is applied for the source term, defined by the function f on the right–hand side of the Poisson equation, given by the expression,

$$f(x, y) = c \exp\left(-\frac{(x-\alpha)^2 + (y-b)^2}{d}\right), \tag{4.4}$$

where, $c = 10.$, $\alpha = b = 0.5$ and $d = 0.02$.

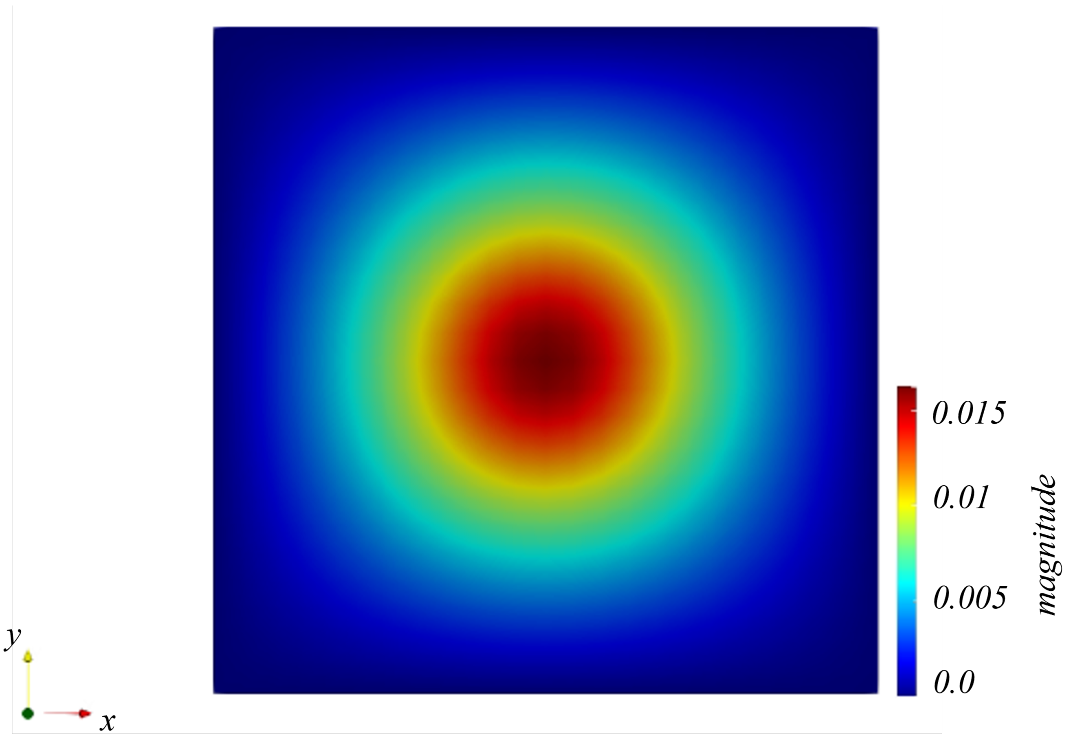

Figure 4.1: Poisson equation with DG FEM and homogeneous Dirichlet conditions.

Additionally, utilizing the DG method to the Poisson problem with a function, $f(x,y) = 2\pi^2 \sin(\pi x)\,\sin(\pi y)$, we can compare the numerical solution with the analytical solution of the problem under consideration, using the L^2 norm of the approximate and analytical solution which is, $u(x,y) = \sin(\pi x)\,\sin(\pi y)$.

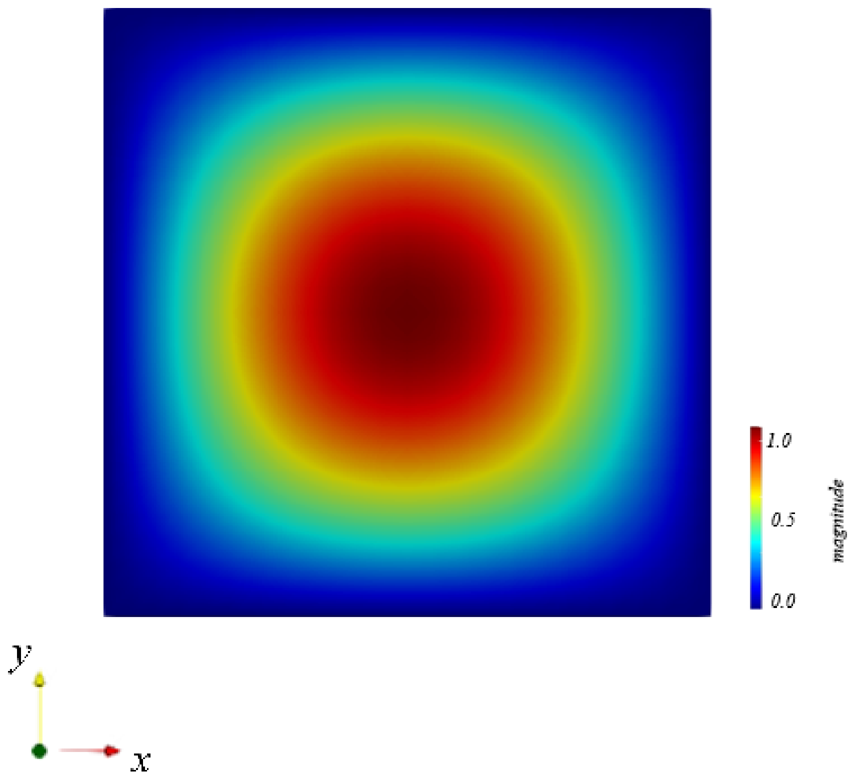

Figure 4.2: Poisson equation with DG FEM, the L^2 norm between the numerical and the analytical solution is, 2.4×10^{-3}.

Following the same methodology as before, we use DG methods for the Stokes problem,

$$\begin{cases} -\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \text{ in } \Omega, \\ \qquad\qquad \boldsymbol{u} = \boldsymbol{0} \text{ on } \partial\Omega. \end{cases}$$

Consider the function spaces V, equipped with discontinuous functions and

Q, with continuous ones,

$$\begin{aligned}
&V = \left\{ \boldsymbol{v} \in \left(L^2(\Omega)\right)^d : v_i \in P_k(K) \forall K \in \mathcal{T}, 1 \leq i \leq d \right\}, \\
&Q = \left\{ q \in H^1(\Omega) : q \in P_j(K) \forall K \in \mathcal{T} \right\}.
\end{aligned}$$

We presented earlier the weak formulation,

$$a((\boldsymbol{u}, p), (\boldsymbol{v}, q)) = L(\boldsymbol{v}, q),$$

for all $(\boldsymbol{v}, q) \in W$, where

$$\begin{aligned}
a((\boldsymbol{u}, p), (\boldsymbol{v}, q)) &= \int_\Omega (\nu \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} - \nabla p \cdot \boldsymbol{v} + \nabla q \cdot \boldsymbol{u}) \, \mathrm{d}x, \\
L((\boldsymbol{v}, q)) &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x + \int_{\partial\Omega_N} \boldsymbol{g} \cdot \boldsymbol{v} \, \mathrm{d}s.
\end{aligned}$$

The space W, should be a mixed (product) function space $W = V \times Q$, such that $u \in V$ and $q \in Q$.

We consider the Stokes equations with both discontinuous functions, as well as, basis functions with possibly varying polynomial orders. The particular bilinear and linear forms for the Stokes equation with DG method can be formulated as,

$$\begin{aligned}
&a(\boldsymbol{v}, q; \boldsymbol{u}, p) = \\
&\quad \sum_{K \in \mathcal{T}} \int_K \nu \nabla \boldsymbol{v} \cdot \nabla \boldsymbol{u} \, \mathrm{d}K + \sum_{K \in \mathcal{T}} \int_K \boldsymbol{v} \cdot \nabla p \, \mathrm{d}K - \sum_{K \in \mathcal{T}} \int_K \nabla q \cdot \boldsymbol{u} \, \mathrm{d}K \\
&\quad + \sum_{K \in \mathcal{F}_i} \int_K q[\boldsymbol{u} \cdot \vec{\boldsymbol{n}}] \, \mathrm{d}s - \sum_{K \in \mathcal{F}_i} \int_K \nu [\boldsymbol{v}] \cdot \langle \nabla \boldsymbol{u} \rangle \, \mathrm{d}s \\
&\quad - \sum_{K \in \mathcal{F}_i} \int_K \nu \langle \nabla \boldsymbol{v} \rangle \cdot [\boldsymbol{u}] \, \mathrm{d}s + \sum_{K \in \mathcal{F}_i} \int_K \nu \frac{\alpha}{h} [\boldsymbol{v}] \cdot [\boldsymbol{u}] \, \mathrm{d}s \\
&\quad + \sum_{K \in \mathcal{F}_e} \int_K q \boldsymbol{u} \cdot \vec{\boldsymbol{n}} \, \mathrm{d}s - \sum_{K \in \mathcal{F}_e} \int_K \nu \boldsymbol{v} \cdot \nabla \boldsymbol{u} \, \mathrm{d}s \\
&\quad - \sum_{K \in \mathcal{F}_e} \int_K \nu \nabla \boldsymbol{v} \cdot \boldsymbol{u} \, \mathrm{d}s + \sum_{K \in \mathcal{F}_e} \int_K \nu \frac{\alpha}{h} \boldsymbol{v} \cdot \boldsymbol{u} \, \mathrm{d}s
\end{aligned}$$

and the right–hand side of the Stokes problem is given as,

$$L(\boldsymbol{v}, q) = \int_\Omega \boldsymbol{v} \cdot \boldsymbol{f} \, \mathrm{d}\Omega.$$

Remark. The above formulation is constructed in a same manner as the Poisson equation with the terms $\sum_{K \in \mathcal{F}_i} \int_K \nu \frac{\alpha}{h} [\boldsymbol{v}] \cdot [\boldsymbol{u}] \, \mathrm{d}s$ and $\sum_{K \in \mathcal{F}_e} \int_K \nu \frac{\alpha}{h} \boldsymbol{v} \cdot \boldsymbol{u} \, \mathrm{d}s$ being artificially added for the stability of the equation [9].

4.2 Adaptive Mesh Refinement

The adaptive mesh refinement (AMR) is an approach for increasing the accuracy of the numerical solution in certain sensitive regions of the discretized domain. Numerical solutions, sometimes reveal accuracy problems to specific regions of the grid or mesh. However, some problems would be better suited if specific computational areas which needed precision could be refined only in the regions requiring the added precision rather than a uniform region. There are widely used methods that omit this problem, called Adaptive Finite Element Mesh Refinement methods (AFEM) with a range of applications to engineering problems. AFEM can be classified into three categories [26].

- In the h-refinement AFEM, we use the same type of finite elements, but their sizes are continuously divided according to a geometric parameter such as the element length or diameter. Among the three categories referred, this is the simplest and more common one to use.
- In the p-refinement AFEM, we increase the order of the polynomial basis functions, but the mesh element size is kept the same.
- In r-refinement AFEM, we keep the number of mesh nodes and elements the same, but the nodes are relocated to problematic areas needed to be optimized.

These methods can also be combined, such as hp–refinement method. We can use the AFEM to obtain a solution of the desired accuracy, but with less computing time, as relatively low degrees of freedom (DOFs) are needed. For the h–type AFEM, the key issue is to determine the regions, where needed, to insert the nodes to balance or evenly distribute the numerical errors of the

FEM solution through a local a posteriori error estimate procedure. The *a posteriori* error estimation is to obtain an estimated numerical error for each element, which plays an important role in guiding the refinement procedures for AFEM.

The process consists of calculating the error indicators for each element of the mesh. Then, a selected number of elements in the domain, e.g. the elements with the largest error indicators, are finally refined. This process is repeated several times until the termination conditions are satisfied. Such conditions could be the maximum refinement number or maximum nodes number in the mesh. The elements can be refined with several methods, such as by bisection, trisection, regular refinement or any combinations of these methods [26].

More precisely, let's consider an *a posteriori* estimator for the Stokes problem. It can be shown that the discrete solution coincides with the continuous one. *A posteriori* error estimates express the error in terms of important quantities, such as the residual error equations and the solution of an auxiliary dual problem [15]. By using the classical Galerkin method, the finite element approximation $u_h \in V^h$ is the solution of,

$$a\left(u_h, v\right) = L(v), \quad \forall v \in V^h.$$

The numerical error in the approximation u_h of u is naturally defined as the function $e \in V$ such that,

$$e = |u - u_h|.$$

The residual errors are denoted as $r(v)$ where,

$$\begin{aligned} r(v) = L(v) - a\left(u_h, v\right) = a(u, v) - a\left(u_h, v\right) = \\ a\left(u - u_h, v\right) \leqslant C \left\|u - u_h\right\|_V \|v\|_V, \quad v \in V. \end{aligned} \tag{4.5}$$

Furthermore,

$$\begin{aligned} \alpha \left\|u - u_h\right\|_V^2 \leqslant a\left(u - u_h, u - u_h\right) = \\ a\left(u, u - u_h\right) - a\left(u_h, u - u_h\right) = \\ L\left(u - u_h\right) - a\left(u_h, u - u_h\right) = r\left(u - u_h\right). \end{aligned} \tag{4.6}$$

Remark. We notice that the residual error $r(v)$ vanishes for all $v \in V^h$, i.e.,

$$r(v) = 0, \quad \forall v \in V^h.$$

This yields to the following orthogonality property,

$$a(e, v) = 0, \quad \forall v \in V^h$$

Combining (4.5), (4.6) yields to:

$$\alpha \left\| u - u_h \right\|_V \leqslant \left\| r \right\|_{V'} \leqslant C \left\| u - u_h \right\|_V \tag{4.7}$$

where $\|r\|_{V'} = \sup_{v \in V, v \neq 0} r(v)/\|v\|_V$.

The *a posteriori* error estimates, equation (4.7) relate the numerical error with the residuals.

The object of Goal–oriented error estimation [15] is to assess the accuracy of finite element solutions in measures other than the classical energy norm. In numerical applications, it is often necessary to control the error in a certain output functional $\mathcal{M} : V \to \mathbb{R}$ of the computed solution to within some given tolerance $\varepsilon > 0$. Typical functionals are the quantities we are intrested in. In these situations, one would ideally like to choose the finite element space $V_h \subset V$, such that, the finite element solution u_h satisfies,

$$\eta \equiv \left| \mathcal{M}(u) - \mathcal{M}\left(u_h\right) \right| \leqslant \varepsilon,$$

with minimal computational work. We assume that both the output functional and the variational problem are linear, but the analysis may be easily extended to the full nonlinear case. To estimate the error in the output functional $\mathcal{M}$, we introduce an auxiliary dual problem: find $z \in V^*$ such that,

$$a^*(z, v) = \mathcal{M}(v), \quad \forall v \in \hat{V}^*.$$

We note here that the functional $\mathcal{M}$ enters as data in the dual problem. The dual (adjoint) bilinear form, $a^* : V^* \times \hat{V}^* \to \mathbb{R}$ is defined by,

$$a^*(v, w) = a(w, v), \quad \forall (v, w) \in V^* \times \hat{V}^*.$$

The dual trial and test spaces are given by,

$$\begin{aligned} V^* &= \hat{V}, \\ \hat{V}^* &= V_0 = \{v - w : v, w \in V\}, \end{aligned}$$

The definition of the dual problem leads to the following error representation,

$$\begin{aligned} \mathcal{M}(u) - \mathcal{M}\left(u_h\right) &= \mathcal{M}\left(u - u_h\right) \\ &= a^*\left(z, u - u_h\right) \\ &= a\left(u - u_h, z\right) \\ &= L(z) - a\left(u_h, z\right) \\ &= r(z) = r(z - z_h). \end{aligned}$$

So, the error is exactly represented by the residual of the dual solution,

$$\mathcal{M}(u) - \mathcal{M}(u_h) = r(z).$$

An adaptive algorithm seeks to determine a mesh size, $h = h(x)$, in a specific tolerance starting from an initial coarse mesh, and refine in those cells where the error indicator remains large.

For the Poisson equation, we take the goal functional to be defined as,

$$\mathcal{M}(u) = \int_\Omega u \, dx. \tag{4.8}$$

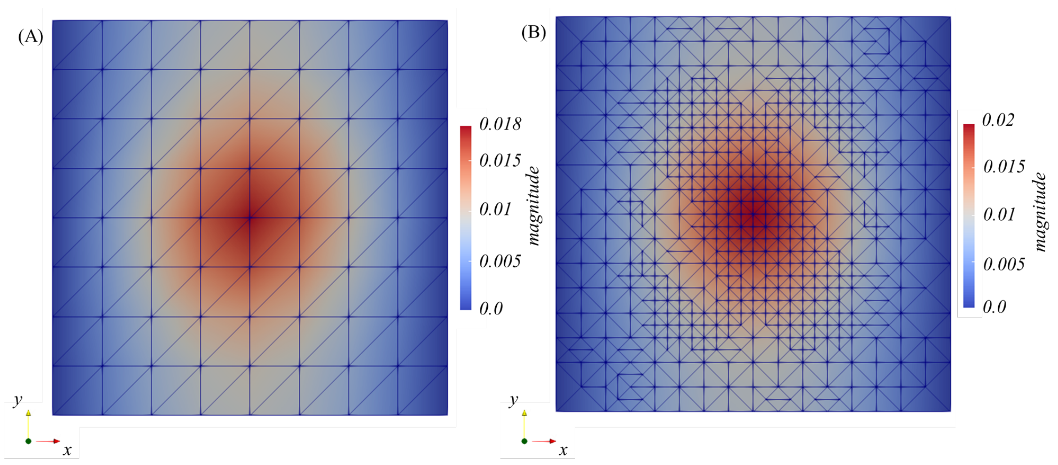

Figure 4.3: (A) Computational mesh and results for the Poisson equation, (B) Computational mesh and results for the Poisson equation after grid local refinement.

In Figure 4.3, we present the solution of Poisson equation with zero Dirichlet condition. At the middle of the domain, we apply a two–dimensional Gaussian distribution, given by the expression described in equation (4.4). We observe that locally refining the computational grid with the AFEM described above, and introducing the goal functional of equation (4.8), the results in the domain of interest are appropriate, capturing in detail the numerical solution of the Poisson equation. The given tolerance is 10^{-5} and the number of cells are increased to 1052 from the initial number of cells which was 128, provide with more accurate results for the problem under consideration.

For the Stokes problem, defining as goal functional the following,

$$\mathcal{M}(u) = \int_\Omega u^2\,dx, \tag{4.9}$$

and we obtain the results depicted in Figure 4.4.

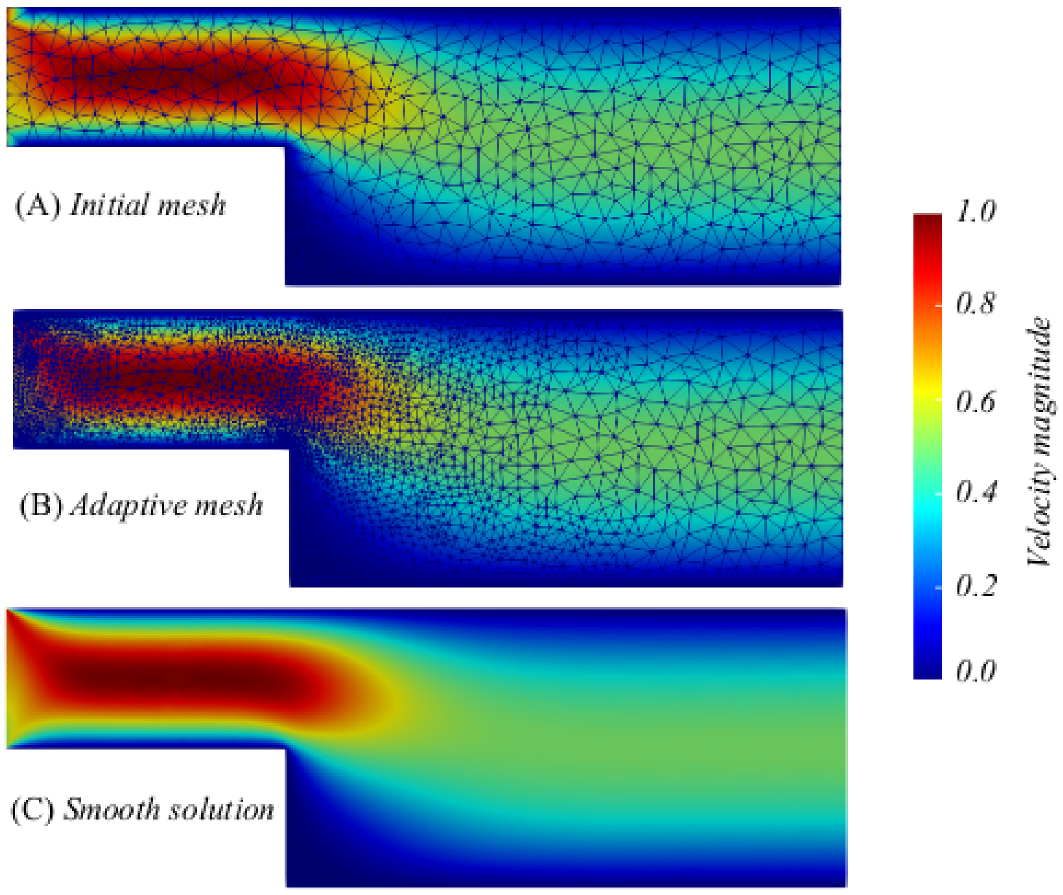

Figure 4.4: (A) Computational mesh and results for the Stokes equation, (B) Computational mesh and results for the Stokes equation after grid local refinement.

In this figure, Figure 4.4, we exhibit the solution of the Stokes equation in the backward facing step problem defining the goal function as in equation (4.9), providing more accurate results than the initial problem. The given tolerance is 10^{-7} and the number of cells are increased to 7464 from the initial number of cells, which was 890.

4.3 Conclusions

In this dissertation, the finite element method is utilized for ordinary differential equations (ODEs) and partial differential equations (PDEs). We mainly focus on applying the method to fluid mechanics problems. Initially, we present the method along with the basic theorems and examples. We analyse the error estimates for linear problems and the base functions to distinguish the problem under consideration. We further present the numerical solution of the one–dimensional Duffing equation and compare with the analytical solution. We further concentrate on the two–dimensional Stokes and Navier–Stokes problems. We finally focus on presenting novel finite element method variants such as the Discontinuous Galerkin (DG) method and adaptive methodologies (AFEM). These advanced methods provide reliable numerical results in all studied cases. This is achieved with the application of the FEM to "test problems", such as the backward facing step. We obtain all the numerical results utilizing the software programs MATLAB and FEniCS.

Appendix

In the following sections we present the main codes in Matlab For the duffing equation and FEniCS for the problems discussed in this thesis. The first problem is the numerical solution of the Duffing equation with three methods, FEM, FDM and RK-Shooting 4th order explicit. The second problem is the Poisson equation (the scalar problem) and the third one is the Stokes problem (the vector problem). The Poisson problem presented is for the Discintinuous Galerkin FEM (DG). The Stokes problem presented in this appendix is for the Adaptive mesh refinement FEM (AFEM).

Duffing equation with FEM in matlab:

```
%----------------------------------------------------------------
%  Integration of the equation with three methods,
%  FEM, FDM, RK-Shooting
%
%  u'' + a * u' + b * u + c * u^3 - g * cos(w * t) = 0
%  u = u(t)
%----------------------------------------------------------------

function DuffingFD
clear; clc;

a      =  0;
b      =  0.06;
```

```
c        =  0.0001;
g        =  0.2;
w        =  1.0;

N1          = 160;
timeend     = 15.0;

dt1 = timeend/(N1-1);

      t1(1)=0;
      for i = 2 : N1
          t1(i) = dt1 * (i-1);
          x1(i) = 0.1;
      end

options = optimoptions('fsolve','Display','testing','
    MaxIter', 1000,'MaxFunEvals',5000000)
x1 = fsolve(@testFD, x1, options, N1, a, b, c, g, w, dt1
    , t1);

% ------ Figures for u ----------------------
       for i = 2 : N1-1
           u(i) = x1(i-1);
       end

       u(1)     = 0.0;
       u(N1)     = u(N1-1);
size(u);
figure
plot(t1, u, '-*b')

i=1;k=1;
j1=0;
while i<=160
         newsol1(k)=u(i);
         j1=j1+20;
         i=j1;
         k=k+1;
end
```

```
    newsol1;

% ——————— FEM ——————————
N    = 161;
dt   = timeend/(N-1);
t    = 0 : dt : timeend;
td   = linspace(0, timeend, N);

% assign the shape functions
     Phi0  = f0(N, dt, t, td, 1);
     Phip0 = fp0(N, dt, t, td, 1);

     PhiN  = fN(N, dt, t, td, N-1);
     PhipN = fpN(N, dt, t, td, N-1);

     Phi_all(1,:)  = Phi0;
     Phip_all(1,:) = Phip0;

     Phi_all(N,:)  = PhiN;
     Phip_all(N,:) = PhipN;

for i = 2 : N-1
     Phi  = f(N, dt, t, td, i);
     Phip = fp(N, dt, t, td, i);

     Phi_all(i,:)  = Phi;
     Phip_all(i,:) = Phip;
end

% create the stiffness matrix and RHS vector

Int1 = zeros(N-1, N-1);

for i = 1 : N-1
     for j = 1 : N-1
          Int = int1(N, dt, t, Phi_all, Phip_all, i, j, a,
              b, c);
          Int1(i,j) = Int;
     end
```

```
end

Int1;

force  =  g * cos(w * t);

for i = 1 : N-1
        Int2 = int2(N, dt, t, Phi_all, force, i);
        Int3(i,1) = Int2;
end

Int3;

% solve the linear system

x = Int1 \ Int3;
size(x);

% plotting the solution
tplotnew  = linspace(0, timeend, (N-1));

xsol(1) = 0.0;

for i = 2 : N-1
 xsol(i) = x(i);
end

figure
plot(tplotnew, xsol, '-*r')
title('FEM Duffing');
xlabel('Time t');
ylabel('Solution u(t)');
size(xsol);

i=1;k=1;
j1=0;
while i<=160
       newsol(k)=xsol(i);
       j1=j1+20;
```

```
        i=j1;
        k=k+1;
end
    newsol;

% ------------ RK-shooting ----------
a1 = 0.0; b1 = 15.0;
h=(b1-a1)/(N-1);

y1 = zeros(1, length(t1));
y2 = zeros(1, length(t1));

y1(1) = 0.0;

Der = - 0.01;

F1 = @(t1,y1,y2)  y2;
F2 = @(t1,y1,y2) - b * y1 + c * y1^3 + g * cos(w*t1);

for j = 1 : 26

for i = 1 : (length(t1)-1)
     k1_1 = F1(t1(i), y1(i), y2(i));
     k2_1 = F2(t1(i), y1(i), y2(i));

     k1_2 = F1(t1(i)+0.5*h, y1(i)+0.5*h * k1_1, y2(i)+0.5*h * k2_1);
     k2_2 = F2(t1(i)+0.5*h, y1(i)+0.5*h * k1_1, y2(i)+0.5*h * k2_1);

     k1_3 = F1((t1(i)+0.5*h), (y1(i)+0.5*h * k1_2), (y2(i)+0.5*h * k2_2));
     k2_3 = F2((t1(i)+0.5*h), (y1(i)+0.5*h * k1_2), (y2(i)+0.5*h * k2_2));

     k1_4 = F1((t1(i)+h), (y1(i)+h * k1_3), (y2(i)+h * k2_3));
     k2_4 = F2((t1(i)+h), (y1(i)+h * k1_3), (y2(i)+h * k2_3));
```

```

      y1(i+1) = y1(i) + h * (1/6)*(k1_1 + 2 * k1_2 + 2 *
          k1_3 + k1_4);
      y2(i+1) = y2(i) + h * (1/6)*(k2_1 + 2 * k2_2 + 2 *
          k2_3 + k2_4);

end

s = length(t1);
Der = y2(s) - 0.0
y2(1) = y2(1) + 0.008;

end

figure
plot(t1,y1,'-*g')

y1;

i=1;k=1;
j1=0;
while i<=160
        newsol2(k)=y1(i);
        j1=j1+20;
        i=j1;
        k=k+1;
end
    newsol2;


%------------ Ploting the results ----------------
figure
plot(tplotnew, xsol, '-k', t1, u, '--k', t1, y1, ':k','
    LineWidth',1.5)
axis([0.0 15.0 -0.5 1.0])
title('FEM, FDM, RK-Shooting');
xlabel('Time t');
ylabel('Solution u(t)');

```

```
err1=abs(newsol-newsol1);
err2=abs(newsol-newsol2);

% ----- Table

figure('Name','Table of errors FEM-FDM','NumberTitle','
    off','Color',[.8 .8 .8])
tab=uitable('Data',err1');
tab.RowName={'i=1','i=20','i=40','i=60','i=80', 'i=100',
    'i=120','i=140','i=160'};
tab.ColumnName={'Values'};
tab.ColumnWidth={100};

tab.Position(3) = tab.Extent(3);
tab.Position(4) = tab.Extent(4);

% ----- Table

figure('Name','Table of errors FEM-RK','NumberTitle','
    off','Color',[.8 .8 .8])
tab=uitable('Data',err2');
tab.RowName={'i=1','i=20','i=40','i=60','i=80', 'i=100',
    'i=120','i=140','i=160'};
tab.ColumnName={'Values'};
tab.ColumnWidth={100};

tab.Position(3) = tab.Extent(3);
tab.Position(4) = tab.Extent(4);

trapz=0;
trapz1=0;
for i1 = 1 : N1-1
    trapz = trapz + dt1* ((xsol(i1)-u(i1))^2+(xsol(i1+1)
        -u(i1+1))^2)/2;
    trapz1 = trapz1 + dt1* ((xsol(i1)-y1(i1))^2+(xsol(i1
        +1)-y1(i1+1))^2)/2;
end

L2_FDM=sqrt(trapz)
```

```
L2_RK=sqrt(trapz1)

% ——— Functions for all codes ————————

% ——— Function test FD ————
function [f, x1] = testFD(x1, N1, a, b, c, g, w, dt1, t1
    )

      for i = 2 : N1-1
          u(i) = x1(i-1);
      end

% boundary conditions
      u(1) = 0.0;
      u(N1) = u(N1-1);  % for the derivative

      for i = 2 : N1-1
%————————————————————————
% Note: simplified version for a = 0
          coef1(i) =  (1/dt1^2);
          coef2(i) = -(2/dt1^2) + b + c * u(i)^2;
          coef3(i) =  (1/dt1^2);
          force(i)  =  g * cos(w * t1(i));

          f(i-1) = coef1(i) * u(i+1) + coef2(i) * u(i) +
              coef3(i) * u(i-1) - ...
                    force(i);
      end
% ——————— functions for fem ——————

function Phi = f(N, dt, t, td, i)
%———————————————————————
      Phi(t > td(i-1) & t <= td(i)) = ...
                        (t(t > td(i-1) & t <= td(i)) - td(
                            i-1)) / dt;
      Phi(t > td(i)   & t <= td(i+1)) = ...
                        ( td(i+1) - t(t > td(i) & t <= td(
                            i+1))) / dt;
      Phi(t > td(i+1))  = 0.0;
```

```
% ----------------------------------------------------------------

function Phip = fp(N, dt, t, td, i)
% ----------------------------------------------------------------
    Phip(t > td(i-1) & t <= td(i))   =   1 / dt;
    Phip(t > td(i)   & t <= td(i+1)) = - 1 / dt;
    Phip(t > td(i+1))                =   0.0;
% ----------------------------------------------------------------

function Phi0 = f0(N, dt, t, td, i)
% ----------------------------------------------------------------
    Phi0(t >= td(i) & t <= td(i+1)) = ...
                    0.0 * ( td(i+1) - t(t >= td(i) & t
                       <= td(i+1))) / dt;
    Phi0(t > td(i+1))               = 0.0;
% ----------------------------------------------------------------

function Phip0 = fp0(N, dt, t, td, i)
% ----------------------------------------------------------------
    Phip0(t >= td(i) & t <= td(i+1)) =   - 0.0 * 1 / dt
       ;
    Phip0(t > td(i+1))               =   0.0;

% ----------------------------------------------------------------

function PhiN = fN(N, dt, t, td, i)
% ----------------------------------------------------------------
    PhiN(t >= td(i) & t <= td(i+1)) = ...
                    ( t(t >= td(i) & t <= td(i+1)) -
                      td(i+1)) / dt;
    PhiN(t < td(i+1))               = 0.0;

% ----------------------------------------------------------------

function PhipN = fpN(N, dt, t, td, i)
% ----------------------------------------------------------------
    PhipN(t >= td(i) & t <= td(i+1)) =   0.0 * 1 / dt;
    PhipN(t < td(i+1))               =   0.0;

```

```
%------------------------------------------------------------

 function [Int] = int1(N, dt, t, Phi_all, Phip_all, i, j
     , a, b, c)
%------------------------------------------------------------
Int = 0.0;

for k = 1 : N-2
     Int = Int + 0.5 * (Phip_all(j,k)*Phip_all(i,k) + ...
                         Phip_all(j,k+1)*Phip_all(i,k+1) )
                              * (t(k+1)-t(k))...
                    - b * 0.5 * (Phi_all(j,k)*Phi_all(i,k) +
                         ...
                              Phi_all(j,k+1)*Phi_all(i,k+1)
                                  ) * (t(k+1)-t(k))...
                    + c * 0.5 * (Phi_all(j,k)^3 * Phi_all(i,k)
                         + ...
                              Phi_all(j,k+1)^3 * Phi_all(i,
                                  k+1) ) * (t(k+1)-t(k));
end

%------------------------------------------------------------

 function [Int2] = int2(N, dt, t, Phi_all, force, i)
%------------------------------------------------------------
Int2 = 0.0;

for k = 1 : N-2
     Int2 = Int2 - 0.5 * (Phi_all(i,k)*force(k) + ...
                           Phi_all(i,k+1)*force(k+1)) * (t
                               (k+1)-t(k));
end
%---------------- end of functions ----------------
```

Discontinuous Galerkin code in Fenics for the Poisson equation:

```
from dolfin import *

# Create mesh
mesh = UnitSquareMesh(24, 24)

plot(mesh)
interactive( )


#define function space
V = FunctionSpace(mesh, "DG", 1)


# Define boundary condition
# Boundaries
toll = 1E − 14  # tolerance for coordinate comparisons
def right(x, on-boundary): return near(x[0], 6, toll)
def left(x, on-boundary): return near(x[0], 0, toll)


# Define test and trial functions
u = TrialFunction(V)
v = TestFunction(V)


# Define normal component, mesh size
n = FacetNormal(mesh)
h = CellSize(mesh)


h_avg = (h('+') + h('-'))/2


f = Expression("exp( -(pow(x[0] - 0.5, 2) + pow(x[1] - 0.5, 2)) / 0.02)", de-
gree=1)
g = Expression("sin(5 * x[0])", degree = 1)


# Inflow boundary condition for velocity
inflow = Constant(1.0)
bc0 = DirichletBC(V, inflow, left)
```

```
#outflow boundary condition for velocity
outflow = Constant(0.0)
bc1 = DirichletBC(V, outflow, right)

# Collect boundary conditions
bcs = [bc0, bc1]

# Define parameters
alpha = 4.0

# Define bilinear form
a = dot(grad(v), grad(u))*dx
- dot(avg(grad(v)), jump(u, n))*dS
- dot(jump(v, n), avg(grad(u)))*dS
+ alpha/h_avg * dot(jump(v, n), jump(u, n)) * dS
- dot(grad(v), u*n) * ds
- dot(v * n, grad(u)) * ds
+ alpha / h * v * u * ds
# Define linear form
L = v * f * dx

# Compute solution
u = Function(V)
solve(a == L, u)

# Project solution to piecewise linears
P1 = FunctionSpace(mesh, "CG", 1)
u_proj = project(u, P1)

# Save solution to file
file = File("poissondg/solution.pvd")
file ¡¡ u-proj

# Plot solution
plot(u-proj, interactive = True)
```

Discontinuous Galerkin code 2 in Fenics for the Poisson equation:

# from dolfin import *

# Create mesh
mesh = UnitSquareMesh(24, 24)

plot(mesh)
interactive()

# define function space
V = FunctionSpace(mesh, ”DG”, 1)

# Define boundary condition

# Boundaries
toll = $1E - 14$ # tolerance for coordinate comparisons
def right(x, $on_boundary$): return near(x[0], 1, toll)
def left(x, $on_boundary$): return near(x[0], 0, toll)

# Define test and trial functions
u = TrialFunction(V)
v = TestFunction(V)

# Define normal component, mesh size
n = FacetNormal(mesh)
h = CellSize(mesh)

h_avg = (h(’+’) + h(’-’))/2

f = Expression(”2*pi*pi*sin(pi*x[0])*sin(pi*x[1])”, degree=1)
u_e = Expression(”sin(pi*x[0])*sin(pi*x[1])”, degree=2)

g = Expression(”sin(5*x[0])”, degree=1)

```
# Inflow boundary condition for velocity
inflow = Constant(1.0)
bc0 = DirichletBC(V, inflow, left)

# outflow boundary condition for velocity
outflow = Constant(0.0)
bc1 = DirichletBC(V, outflow, right)

# Collect boundary conditions
bcs = [bc0, bc1]

# Define parameters
alpha = 4

# Define bilinear form
a = dot(grad(v), grad(u))*dx
- dot(avg(grad(v)), jump(u, n))*dS
- dot(jump(v, n), avg(grad(u)))*dS
+ alpha/h_avg*dot(jump(v, n), jump(u, n))*dS
- dot(grad(v), u*n)*ds
- dot(v*n, grad(u))*ds
+ alpha/h*v*u*ds

# Define linear form
L = v*f*dx

# Compute solution
u=Function(V)
solve(a == L, u)

# Project solution to piecewise linears
P1 = FunctionSpace(mesh, "CG", 1)
u_proj = project(u, P1)
```

# Save solution to file
file = File(”poissondg1/solution.pvd”)
file ¡¡ u_proj

# Plot solution
plot(u_proj, interactive=True)

# Compute error in L2 norm
$error_L2$ = errornorm(u_e, u_proj, ’L2’)

# Print errors
print(’$error_L2$ =’, $error_L2$)

# Hold plot
interactive()

Adaptive mesh refinement in Fenics for the Stokes problem:

```
from dolfin import *
from mshr import *
import numpy as np
from dolfin.cpp import LinearVariationalSolver, NonlinearVariationalSolver
from dolfin.fem.adaptivesolving import AdaptiveLinearVariationalSolver
from dolfin.fem.adaptivesolving import AdaptiveNonlinearVariationalSolver
```

```
# Create mesh
channel = Rectangle(Point(0,0), Point(6,2))
step = Rectangle(Point(0,0), Point(2,1))
domain = channel - step
mesh = generate_mesh(domain, 24)
inlet_length = float(1)*float(6)
```

```
plot(mesh)
interactive( )
```

```
# Define function spaces (Taylor Hood element)
V = VectorElement("Lagrange", mesh.ufl_cell( ), 2)
Q = FiniteElement("Lagrange", mesh.ufl_cell( ), 1)
TH = V * Q
W = FunctionSpace(mesh, TH)
toll = 1.E − 14   # tolerance for coordinate comparisons
```

```
# Boundaries
def right(x, on-boundary): return near(x[0], 6, tol)
def left(x, on-boundary): return near(x[0], 0, tol)
def top-bottom(x, on-boundary): return near(x[1], 2, tol) or near(x[1], 0, tol)
def step(x, on-boundary): return near(x[0], 2, tol) or near(x[0], 2, tol) and
near(x[1], 1, tol) or near(x[1], 1, tol)
```

```
# No-slip boundary condition for velocity
noslip = Constant((0, 0))
bc0 = DirichletBC(W.sub(0), noslip, top_bottom)
bc1 = DirichletBC(W.sub(0), noslip, step)
```

```
# Inflow boundary condition for velocity
inflow = Expression((“- 0.1 * x[1] * (x[1]-)”.format(inlet_length), “0.0”), de-
gree = 2)
bc2 = DirichletBC(W.sub(0), inflow, left)

# Boundary condition for pressure at outflow
zero = Constant(0)
bc3 = DirichletBC(W.sub(1), zero, right)

# Collect boundary conditions
bcs = [bc0, bc1, bc2, bc3]

# Define variational problem
ni = Constant(0.2)
(u, p) = TrialFunctions(W)
(v, q) = TestFunctions(W)
f = Constant((0.0, 0.0))
a = ni * inner(grad(u), grad(v))*dx - div(v) * p * dx + q * div(u) * dx
L = inner(f, v) * dx

# Compute solution
w = Function(W)

# Define goal functional (quantity of interest)
(uu, pp) = (as_vector((w[0], w[1])), w[2])

M = inner(uu[0], uu[0]) * dx( )

# Define error tolerance
tol = 1.E-7

# Solve equation a = L with respect to u and the given boundary
# conditions, such that the estimated error (measured in M) is less
# than tol
```

```
solve(a == L, w, bcs, tol = tol, M = M)

# Split the mixed solution using deepcopy
# (needed for further computation on coefficient vector)

print('w1:', w.leaf_node().vector().array())
print('w:', w.vector().array())
R = w.leaf_node()#.function_space()

(u1, p1) = R.split(True)
(u, p) = w.split(True)

# Split the mixed solution using a shallow copy
(u1, p1) = R.split()
(u, p) = w.split()

# Save solution to file
ufile_pvd = File("adstokes/solution1.pvd")
ufile_pvd ¡¡ u1
ufile_pvd = File("adstokes/solution.pvd")
ufile_pvd ¡¡ u

# Plot solution(s)
plot(u1, title = "Solution on final mesh")
plot(u, title = "Solution on intial mesh")

interactive( )
```

Classical Galerkin FEM for the Stokes equation in Cartesian coordinates:

In this subsection, we present the classical Galerking FEM analysis of the Stokes problem in the two–dimensional Cartesian coordinates, $(x, y) = (x_1, x_2)$. This analysis follows the initial steps in Chapter 3 of this thesis. Where $\boldsymbol{u} = (u_x, u_y) = (u_1, u_2)$ and $\boldsymbol{v} = (v_x, v_y) = (v_1, v_2)$. We also consider that the kinematic viscosity, ν, of the fluid is constant.

Remark. We note that the notation in the term, $\displaystyle\int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\,d\Omega$, can be also found in the literature written as, $\displaystyle\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v}\,d\Omega$. Where, $\nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v} = \nabla\boldsymbol{u} : \nabla\boldsymbol{v}$ $\displaystyle= \sum_{i,j=1}^{n} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}$, where $n = 2$ for the two–dimensional case.

We obtain that,

$$
\begin{aligned}
a(\boldsymbol{u}, \mathbf{v}) = \nu \int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\,d\Omega = \nu \int_\Omega \sum_{i,j=1}^{n=2} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}\,d\Omega = \\
= \nu \iint_\Omega \sum_{i,j=1}^{n=2} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}\,dx_1\,dx_2 = \\
= \nu \sum_{K\in\mathcal{T}} \iint_K \sum_{i,j=1}^{n=2} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}\,dx_1\,dx_2,
\end{aligned}
$$

where K is each element of the triangulation, $\mathcal{T}$, of the Ω domain. Similarly we obtain,

$$
b(p, \boldsymbol{v}) = \int_\Omega p\,(\nabla\cdot\boldsymbol{v})\,d\Omega = \iint_\Omega p\left(\sum_{i=1}^{n=2}\frac{\partial u_i}{\partial x_i}\right)\,dx_1 dx_2 = \sum_{K\in\mathcal{T}} \iint_K p\left(\sum_{i=1}^{n=2}\frac{\partial u_i}{\partial x_i}\right)\,dx_1 dx_2.
$$

Bibliography

[1] Antil, H., Hoppe, R. H., and Linsenmann, C. Path-following primal-dual interior-point methods for shape optimization of stationary flow problems.

[2] Arndt, D. *Augmented Taylor-Hood elements for incompressible flow.* 2013.

[3] Baker, G. A. Finite element methods for elliptic equations using non-conforming elements. *Mathematics of Computation 31*, 137 (1977), 45–59.

[4] Barkanov, E. Introduction to the finite element method. *Institute of Materials and Structures Faculty of Civil Engineering Riga Technical University* (2001), 1–70.

[5] Bercovier, M., and Pironneau, O. Error estimates for finite element method solution of the stokes problem in the primitive variables. *Numerische Mathematik 33*, 2 (1979), 211–224.

[6] Brenner, S., and Scott, R. *The mathematical theory of finite element methods*, vol. 15. Springer Science & Business Media, 2007.

[7] Cockburn, B. Discontinuous galerkin methods. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics 83*, 11 (2003), 731–754.

[8] Cockburn, B., Karniadakis, G. E., and Shu, C.-W. The development of discontinuous galerkin methods. In *Discontinuous Galerkin Methods.* Springer, 2000, pp. 3–50.

[9] Cockburn, B., Karniadakis, G. E., and Shu, C.-W., Eds. *Discontinuous Galerkin Methods.* Springer Berlin Heidelberg, 2000.

[10] Delfour, M., Hager, W., and Trochu, F. Discontinuous galerkin methods for ordinary differential equations. *Mathematics of Computation 36*, 154 (1981), 455–473.

[11] Hartmann, R. Numerical analysis of higher order discontinuous galerkin finite element methods.

[12] Jiajan, W. *Solution to incompressible Navier stokes equations by using finite element method.* PhD thesis, The University of Texas at Arlington, 2010.

[13] Langtangen, H. P., and Logg, A. *Solving PDEs in python: the FEniCS tutorial I.* Springer Nature, 2017.

[14] Lesaint, P., and Raviart, P.-A. On a finite element method for solving the neutron transport equation. *Publications mathématiques et informatique de Rennes*, S4 (1974), 1–40.

[15] Logg, A., Mardal, K.-A., and Wells, G. *Automated solution of differential equations by the finite element method: The FEniCS book*, vol. 84. Springer Science & Business Media, 2012.

[16] Mbah, G. C., and Ibeh, K. K. Application of the finite element method to solving the duffing equation of ground motion. *Global Journal of Pure and Applied Sciences 26*, 1 (2020), 65–71.

[17] Mu, L., and Ye, X. A simple finite element method for the stokes equations. *Advances in Computational Mathematics 43*, 6 (2017), 1305–1324.

[18] Nochetto, R. H., Siebert, K. G., and Veeser, A. Theory of adaptive finite element methods: an introduction. In *Multiscale, nonlinear and adaptive approximation.* Springer, 2009, pp. 409–542.

[19] Ølgaard, K. B., Logg, A., and Wells, G. N. Automated code generation for discontinuous galerkin methods. *SIAM Journal on Scientific Computing 31*, 2 (2009), 849–864.

[20] Petropoulou, E. N., and Xenos, M. A. Qualitative, approximate and numerical approaches for the solution of nonlinear differential equations. In *Applications of Nonlinear Analysis.* Springer, 2018, pp. 611–664.

[21] Plexousakis, M., and Chatzipantelidis, P. *Numerical Solution of Partial Differential Equations (in Greek).* Kallipos, Open Academic Editions. http://hdl.handle.net/11419/665, 2015.

[22] Rajasekaran, J. *On the flow characteristics behind a backward-facing step and the design of a new axisymmetric model for their study.* University of Toronto Canada, 2011.

[23] Raptis, A., Kyriakoudi, K., and Xenos, M. A. Finite element analysis in fluid mechanics. In *Mathematical Analysis and Applications.* Springer, 2019, pp. 481–510.

[24] Reddy, J. *An introduction to the finite element method*, vol. 1221. McGraw-Hill New York, 2010.

[25] Tezduyar, T. E., Mittal, S., Ray, S., and Shih, R. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering 95*, 2 (1992), 221–242.

[26] Zhao, Y., Zhang, X., Ho, S. L., and Fu, W. An adaptive mesh method in transient finite element analysis of magnetic field using a novel error estimator. *IEEE transactions on magnetics 48*, 11 (2012), 4160–4163.

[27] Zienkiewicz, O. C., and Cheung, Y. K. *Finite Element Method in Structural & Continuum Mechanics.* McGraw Hill, 1967.

[28] Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. *The finite element method: its basis and fundamentals.* Elsevier, 2005.