A numerical approximation for the Navier-Stokes equations using 
the Finite Element Method

João Francisco Marques 
joao.r.marques@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

June 2016

Abstract

We describe the numerical approximation of the Navier-Stokes equations, for incompressible New-
tonian fluids, using the Finite Element Method. In order to be able to deal with different typical
situations, a Matlab numerical implementation was done and tested for the linear case (Stokes equa-
tions), the stationary convection dominated Navier-Stokes equations and also for the evolutive case.
This included the possibility of dealing with both Dirichlet and Neumann boundary conditions. The
inf-sup stable P2-P1 FEM were used for the spatial discretization. In the case of convection dominated
problems, a stabilization technique was added at the linear iterative step of Newton’s method. The
fully implicit Euler finite difference scheme was used for the time discretization. The starting point
for the Matlab implementation was the Q2-Q1 implementation for the Stokes equations, with Dirichlet
boundary conditions, made available in [4], [3] and [2].

Keywords: Navier-Stokes equations, Numerical approximation, Newton-Raphson Method,

1 Introduction

The Navier-Stokes equations have been among the
most famous mathematical modelling tolls for a few
decades now. Its ability to describe many differ-
ent types of fluids allows its application to day
life type of problems that span a variety of re-
search and industrial fields such as weather fore-
cast, ocean stream modeling, airplane testing, au-
tomobile design, biodigestor prototyping or, more
recently, blood flow simulations. Although in the
past 50 years many advances have been made in
terms of the mathematical analysis of the Navier-
Stokes equations in its different versions (see, for
instance, [6] and [16]), for the type of applications
mentioned above, such thing as analytical solutions
must obviously be excluded. In this scenario, effi-
cient computational tools must be used instead, in
order to obtain reliable numerical approximations
of the real solution. Due to the nonlinear nature of
these equations, typical methods for the numerical
approximation of partial differential equations (8),
which perform well for linear problems, when kept
in their original form, fail to capture the complex-
ity of the Navier-Stokes equations ([14]) needing to
be modified in not so intuitive ways, in order to be
considered stable. This modifications, called stabi-
zation techniques (see [12]), must be chosen and
adjusted according to the specific type of fluid that
we are modelling.

At the present stage many comercial and also some
free packages can be found to compute numerical
solutions for the Navier-Stokes equations (Open
Foam, Comsol Multiphysics, Fenics Project, LifeV,
just to mention a few). However, a robust understanding of the inherent methods, as it is required for research purposes in fluid dynamics, requires a mastering of the implemented algorithm that can hardly be accomplished by just using such type of “ready-to-use” packages. In this sense, as part of a major strategy for future local research activities on this subject, a numerical algorithm to solve the time dependent Navier-Stokes equations for incompressible Newtonian fluids was implemented. The time discretization was done using an implicit Euler formula for finite differences, which allows to deal with time changing boundary conditions including pulsatile flow in a stable way ([14]). The space discretization approach is based on the Finite Element Method ([3], [5] and [12]). To allow the simulation of flow associated to high Reynolds number, stabilization terms were added to the weak formulation, corresponding to the so called Streamline Upwind Petrov-Galerkin stabilization (SUPG). Although different approaches can be used in the spirit of SUPG, here the approach of [5] was followed. Therefore, the Newton-Raphson (NR) method was used to deal with the nonlinear feature of the momentum equation, and the SUPG terms were added in each linear iteration of NR method. A separated code devoted to the stationary case was also prepared. It was drawn for both high Reynolds number (convection dominated problem) or low Reynolds number (Stokes flow). In Section 2, the case of the Stokes equations is presented. This includes the definition of its weak formulation and the application of Newton-Raphson method to the stationary case is described in addition with a short version of the associated stability result. The SUPG is introduced and described. The application of FEM is made and numerical results are shown using a generalization of the implementation used in the previous chapter. In Section 4 the time discretization is described and a stability estimate is given. The implementation is applied to two reference problems such as the Womersley flow and a obstacle based turbulent flow.

2 Stokes Equations

The following is the formulation for the Stokes problem adopted in my thesis:

\[ - \Delta \tilde{u} + \nabla p = \tilde{f} \]  \hspace{1cm} (2.1)
\[ \nabla \cdot \tilde{u} = 0 \]  \hspace{1cm} (2.2)

where \( \tilde{u} \in \mathbb{R}^2 \) is the fluid velocity, \( p \) is the pressure and \( \tilde{f} \) is an optional external force to be applied on the fluid particles. In addition we assume the equations of the flow are valid in a certain domain \( \Omega \). One way of solving the Stokes PDE, is to discretize it by using the Finite Element Method. In this section we derive the weak form. But before that, we first need to define a few functional spaces where we will be working on. We start with a very important, Sobolev Space, that we will be using from now on (with the norm defined as usual):

\[ H^1(\Omega) = \{ u : \Omega \rightarrow \mathbb{R}^2 | \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L^2(\Omega) \} \]

With \( L^2(\Omega) \) being the space with the functions regular enough so that they verify \( \int_\Omega |v|^2 < \infty \). Next we define the space where the solution will lie in:

\[ H^1_E(\Omega) = \{ \tilde{u} \mid \tilde{u} \in [H^1(\Omega)]^2, \tilde{u} = \tilde{g}, \text{in } \Gamma_{\Omega_D} \} \]

Additionally, we also need to make use of test functions, which are an essential ingredient of the weak formulation. These functions live in a different space:

\[ H^1_{E_0}(\Omega) = \{ \tilde{u} \in [H^1(\Omega)]^2 | \tilde{u} = 0, \text{in } \Gamma_{\Omega_D} \} \]

As far as the pressure is concerned, we consider the solution \( p \) to be in \( L^2_0 \) (which is the set of functions in \( L^2 \) such that \( \int_\Omega p = 0 \)) and the test functions \( q \) to be in \( L^2(\Omega) \). Now we have enough ingredients to proceed with the weak formulation process. The final form of the weak formulation is the following:
(for more details, see [1]): find $\bar{u} \in H^1_E(\Omega)$, $p \in L^2_0(\Omega)$ such that:

$$\int_{\Omega} \nabla \bar{u} : \nabla \bar{v} - \int_{\Omega} p (\nabla \cdot \bar{v}) = \int_{\Omega} f \cdot \bar{v} + \int_{\partial \Omega_N} \bar{s} \cdot \bar{v}$$

(2.3)

$$\int_{\Omega} q \cdot \nabla \cdot \bar{u} = 0$$

(2.4)

for all $\bar{v} \in H^1_E(\Omega)$, $q \in L^2(\Omega)$. To solve system (2.3) and (2.4) numerically, we need to discretize it. This is accomplished by considering a finite dimensional subspace $X^h$ of the original function space $H^1_E$. Generally one considers that such space, $X^h$, is generated by a base $\{\phi_i\}_{0 \leq i \leq n}$ for some $n$. Similarly one also considers a finite-basis subspace $M^h$ for the pressure whose basis is $\{\psi_i\}_{0 \leq i \leq m}$. The choices on the number of functions that form either the basis of $X^h$ or $M^h$ is directly related to the discretization of the domain $\Omega$. Naturally, these choices will have direct impact on the results of the computed solution. We omit here some of the details related to the discretization process and after some calculations we have the following discrete system (for the weak formulations at (2.3) and (2.4)):

$$\sum_{j=1}^{n_{in}} u_j \int_{\Omega} \nabla \phi_j : \nabla \phi_i - \int_{\Omega} \nabla \cdot \phi_i \psi_k = 0$$

(2.5)

We still need to take one additional step. Note that in equation (2.5) we have all $u$ nodes on the left hand side of the equation, which does not make sense since we are only solving for nodes whose value is unknown (we already know the Dirichlet boundary nodes with respect to $u$). So we do:

$$- \sum_{j=1}^{n_{in}} u_j \int_{\Omega} \psi_k \cdot \nabla \phi_j = \sum_{j=n_{in}+1}^{n_{in}+n_d} u_j \int_{\Omega} \psi_k \cdot \nabla \phi_j$$

(2.6)

which can be reformulated as:

$$\begin{pmatrix} A & B^T \\ B & O \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

(2.7)

where:

$$A_{ij} = \int_{\Omega} \nabla \phi_i : \nabla \phi_j$$

(2.8)

$$B_{kj} = - \int_{\Omega} \psi_k \cdot (\nabla \cdot \phi_i)$$

(2.9)

$$f_i = \int_{\Omega} f \cdot \phi_i + \int_{\Gamma_{\Omega_N}} \bar{s} \cdot \phi_i - \sum_{j=n_{in}+1}^{n_{in}+n_d} u_j \int_{\Omega} \nabla \phi_j : \nabla \phi_i$$

(2.10)

$$g_k = \sum_{j=n_{in}+1}^{n_{in}+n_d} u_j \int_{\Omega} \psi_k \cdot \nabla \phi_j$$

(2.11)

To solve system (2.7), we used P2-P1 finite element discretization. To test our algorithm, we applied it to several standard test cases. Here we present just a few of them. For the remaining, see [1]. Table 1 shows the relative errors we have obtained for a Poiseuille flow solution.

| DOF | $|e_{u_1}|$ | time (seconds) |
|-----|---------|---------------|
| 25  | $4.7 \times 10^{-10}$ | $1.32 \times 10^{-2}$ |
| 81  | $1.6 \times 10^{-10}$ | $2.01 \times 10^{-3}$ |
| 289 | $4.59 \times 10^{-11}$ | $8.9 \times 10^{-4}$ |
| 1089| $1.20 \times 10^{-11}$ | $4.04 \times 10^{-5}$ |
| 4225| $3.09 \times 10^{-12}$ | $2.28 \times 10^{-6}$ |
| 16641| $7.84 \times 10^{-13}$ | $1.39 \times 10^{-7}$ |
| 66049| $2.01 \times 10^{-13}$ | $7.67 \times 10^{-9}$ |

Figure 1: Log-Log graph with linear regression for example 1
see that visually, the solution is what we expect. In addition, figure 1 shows the evolution of the quadratic relative error (the blue dots represent the entries in table 1 in red we have a linear regression of the said data and in green we have a linear function with slope 2). Other test we did was to solve a cavity flow. Figure 2 is a streamline of the computed solution.

Since we don’t have the analytic expression for our solution this time, we can’t have an accurate way to test if our solution is correct. However, we can compare it to previous works such as [3], pag. 220. We verify that the solution has the expected appearance. In addition, we tested the algorithm for a problem with an analytic solution given by:

\[
\begin{align*}
    u_x &= 20xy^3, \quad u_y = 5x^4 - 5y^4, \quad p = 60x^2y - 20y^3 + c \\

\end{align*}
\]  
  \hspace{1cm} (2.12)

We considered Dirichlet boundary conditions on boundaries number 1 and 3 and the remaining will be Neumann boundary conditions. Using the definition of the Neumann boundary function \( \vec{s} \), we know that:

\[
\begin{align*}
    \vec{s}_2 &= (-60x^2y + 40y^3, 20x^3) \\
    \vec{s}_4 &= (60x^2y - 40y^3, -20x^3)
\end{align*}
\]  
  \hspace{1cm} (2.13)

\hspace{1cm} (2.14)

Table 2 shows the relative errors for our computed solution.

The error evolution decreases as the discretization rises as expected.

| DOF | \( |e_{u_1}| \) | \( |e_{u_2}| \) | time (seconds) |
|-----|----------------|----------------|----------------|
| 25  | \( 5.0 \times 10^{-2} \) | \( 1.2 \times 10^{-1} \) | \( 4.14 \times 10^{-2} \) |
| 81  | \( 6.7 \times 10^{-4} \) | \( 1.19 \times 10^{-2} \) | \( 2.29 \times 10^{-3} \) |
| 289 | \( 3.84 \times 10^{-4} \) | \( 1.0 \times 10^{-4} \) | \( 1.094 \times 10^{-2} \) |
| 1089| \( 4.45 \times 10^{-5} \) | \( 7.72 \times 10^{-5} \) | \( 4.13 \times 10^{-4} \) |
| 4225| \( 3.11 \times 10^{-6} \) | \( 5.40 \times 10^{-6} \) | \( 2.12 \times 10^{-5} \) |
| 16641| \( 2.04 \times 10^{-7} \) | \( 3.70 \times 10^{-7} \) | \( 1.249 \times 10^{0} \) |
| 66049| \( 1.36 \times 10^{-8} \) | \( 2.45 \times 10^{-8} \) | \( 7.35 \times 10^{0} \) |

3 Stationary Navier Stokes Equations

In this Section we describe our methods to solve the stationary case of the Navier Stokes Equations. The following is a possible formulation for the Navier-Stokes equations:

\[
\begin{align*}
- \nu \Delta \vec{u} + \left( \vec{u} \cdot \nabla \right) \vec{u} + \nabla p &= \vec{f}, \quad \text{in } \Omega \hspace{1cm} (3.1) \\
\nabla \cdot \vec{u} &= 0 \hspace{1cm} (3.2)
\end{align*}
\]

3.1 Application of Newton Method

To solve the Navier-Stokes Equations we proceed as we did before, that is, first we write the correspondent weak formulation of the Navier-Stokes problem. Details are omitted here, and the final form of the weak formulation is the following: find \( \vec{u} \in H^1_E \) and \( p \in L^2(\Omega) \):

\[
\begin{align*}

\nu \int_{\Omega} \nabla \vec{u} : \nabla \vec{v} + \int_{\Omega} (\vec{u} \cdot \nabla) \vec{u} \cdot \vec{v} + \int_{\Omega} p(\nabla \cdot \vec{v}) = \\
\int_{\Omega} \vec{f} \cdot \vec{v}, \forall \vec{v} \in H^1_E \\
\int_{\Omega} q(\nabla \cdot \vec{u}) = 0, \forall q \in L^2(\Omega)
\end{align*}
\]

Unfortunately we no longer obtain a linear system that can easily be solved, and this happens because of the term \( (\vec{u}, \nabla) \vec{u} \). In order to surpass this challenge, we have chosen to apply Newton’s method to the problem. For the linearization details see [11]. The following is the final linear problem:
where \( A \) and \( B \) are defined as in the Stokes discrete formulation (2.8) and (2.9) and:

\[
\begin{align*}
fi &= Re_i \\
gk &= \int_\Omega \psi_k(\nabla \cdot \bar{u}_k) \\
Ni_{ij} &= \int_\Omega (\bar{u}_k \nabla \phi_j) \cdot \bar{\phi}_i \\
Wi_{ij} &= \int_\Omega (\bar{\phi}_j \cdot \nabla \bar{u}_k) \phi_i
\end{align*}
\]

with \( u_{k+1} = \delta u_k + u_k \) and \( p_{k+1} = \delta p_k + p_k \). Naturally we need to agree on a stopping criteria for these methods as well. In our case we will simply control the quantity:

\[
||\delta u_k||_{H^1(\Omega)} = (||\nabla \delta u_k||_{L^2(\Omega)}^2 + ||\delta u_k||_{L^2(\Omega)}^2)^{1/2}
\]

(3.8)

and agree that our method has converged if \( ||\delta u_k||_{H^1(\Omega)} < \epsilon \), for a previously chosen \( \epsilon \).

### 3.2 SUPG Method

In order to reduce the interpolation errors, one often uses a stabilization term on the weak problem. More on this topic can be found in [9], chapter 2 and [14] sections 8.3.1 and 8.3.2. For stabilization on other problems, such as streamline-diffusion or advection-diffusion, see [10], [17], [15] or [1]. In our case what we will do is to consider first the weak formulation of the Newton method for the stationary Navier-Stokes equations: Find \( u^{k+1} \in H^1_{Ko}; p^{k+1} \in L^2 \):

\[
a(u^{k+1}, p^{k+1}, v) = F(v), \forall v \in H^1_{Ko}
\]

(3.9)

where:

\[
a(u^{k+1}, p^{k+1}, v) = -\nu(\Delta u^{k+1}, v) + (u^{k+1} \cdot p^{k+1}, v) + (u^{k+1} \cdot \nabla u^{k+1}, v) + (\nabla p^{k+1}, v)
\]

\[
F(v) = (f, v) + (u^k \cdot \nabla u^k, v)
\]

The SUPG method consists of adding a stabilization operator to formulation (3.9). \( \tilde{L}(w,p) \), which will be zero if we evaluate it on the actual solution. The following is the adopted formulation for our stabilization operator (SUPG case), for other possible stabilization formulas, see [9]:

\[
\tilde{L}(w,p) = \delta \sum_{k \in \tau_\alpha} (L(w,p) - b, h_k^2 \cdot L_{SS}(v,q))_K
\]

(3.10)

where:

\[
L(v,q) = \left( -\nu \Delta v + (u^k \cdot \nabla)v + (v \cdot \nabla)u^k + \nabla q \right) \nabla \cdot v
\]

(3.11)

\[
b = \left( f + (u^k \cdot \nabla)u^k \right)
\]

(3.12)

and \( L_{SS} \) is the skew-symmetric associate of the operator \( L \). The final stabilized method is the following:

\[
a(u^{k+1}, p^{k+1}, v) + \tilde{L}(u^{k+1}, p^{k+1}) = F(v) \forall v \in H^1_{Ko}
\]

(3.13)

The challenge is to find a good value for \( \delta \) in formula (3.10). The optimal value for the \( \delta \) seems to vary significantly according to the problem type and the ammount of discretization used. However if we choose a value for \( \delta \) that is too far from the optimal value, SUPG seems to take too long to converge (and does not converge in a realistic ammount of time). We have failed to come up with an heuristic that behaves optimally for all problems and all discretizations, however we did find an heuristic that seems to behave properly for our examples (but it has limitations) as we will see in the next chapter.
Our heuristic (found only via trial-and-error) is:

$$\delta = \frac{6.4 \times 10^{-4}}{\nu \cdot h^2_K}$$

(3.14)

however, other heuristics to calculate $\delta$ can be found in [13]. The following stability result is also proved on the extended version of this thesis:

**Theorem 3.1** Assuming that $u^T \cdot \nabla b \cdot u \geq 0$, where $b$ is the solution obtained in the previous iterate and that $(L_{SS}(v, q), L_{S}(v, q))_K$ is also 0, $\forall v \in H^1_K$, $\forall q \in L^2$. Furthermore assume that our norm $\|\cdot\|_{SUPG}$ is defined by the following:

$$\|u\|_{SUPG} = \nu \|\nabla u\|_{L^2}^2 + (b, \nabla u, u) + (u \nabla b, u) + \sum_{k \in \tau_h} (L_{SS}(u, p), h^2_K \cdot L_{SS}(u, p))_K + (\nabla p, u)$$

then,

$$\|u\|_{SUPG} \leq \frac{4}{3}(\epsilon_1 + 1)\|g\|_{L^2}^2$$

(3.15)

where $g = f + b \cdot \nabla b$.

In order to test our algorithm several cases were analyzed (see the full version of my thesis for the extended details). Table 3 shows the relative errors obtained when trying to compute a solution for the Poiseuille flow of the stationary Navier-Stokes equations case.

Table 3: Navier-Stokes Poiseuille Benchmark (Fully implicit Newton-SUPG). Relative errors and computational time for different degrees of freedom.

<table>
<thead>
<tr>
<th>DOF</th>
<th>$|\epsilon_{u_1}|$</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>$1.2 \times 10^{-1}$</td>
<td>2.00</td>
</tr>
<tr>
<td>81</td>
<td>$3.3 \times 10^{-1}$</td>
<td>1.07</td>
</tr>
<tr>
<td>289</td>
<td>$5.31 \times 10^{-4}$</td>
<td>1.66</td>
</tr>
<tr>
<td>1089</td>
<td>$6.24 \times 10^{-4}$</td>
<td>1.63</td>
</tr>
<tr>
<td>4225</td>
<td>$6.57 \times 10^{-4}$</td>
<td>2.9</td>
</tr>
<tr>
<td>16641</td>
<td>$6.65 \times 10^{-4}$</td>
<td>10.07</td>
</tr>
<tr>
<td>66049</td>
<td>$6.64 \times 10^{-4}$</td>
<td>51.7</td>
</tr>
</tbody>
</table>

The results didn’t appear to have the error decrease we were expecting, one cause for this might be the formula we have adopted to calculate the $\delta$ parameter of the SUPG method. Table 4 denotes the minimal conditions for which convergence was found (according to formula 3.8) after trying to compute a solution for the cavity flow of the stationary Navier-Stokes equations.

Table 4: Cavity Convergence Benchmark (SUPG fully implicit Newton) for several different values of viscosity ($\nu$). Minimum required discretization level and number of iterates to achieve convergence.

<table>
<thead>
<tr>
<th>$\nu (m^2/s)$</th>
<th>convergence (DOF, iterates)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 10^4$</td>
<td>(1089, 4)</td>
</tr>
<tr>
<td>$1 \times 10^7$</td>
<td>(1089, 5)</td>
</tr>
<tr>
<td>$1 \times 10^{-1}$</td>
<td>(1089, 6)</td>
</tr>
<tr>
<td>$1 \times 10^{-2}$</td>
<td>(1089, 9)</td>
</tr>
<tr>
<td>$1 \times 10^{-3}$</td>
<td>(1089, 13)</td>
</tr>
<tr>
<td>$1 \times 10^{-4}$</td>
<td>(1089, 52)</td>
</tr>
<tr>
<td>$1 \times 10^{-5}$</td>
<td>(1089, 46)</td>
</tr>
<tr>
<td>$1 \times 10^{-6}$</td>
<td>(1089, 47)</td>
</tr>
<tr>
<td>$1 \times 10^{-7}$</td>
<td>(1089, 45)</td>
</tr>
<tr>
<td>$1 \times 10^{-10}$</td>
<td>(1089, 48)</td>
</tr>
<tr>
<td>$1 \times 10^{-13}$</td>
<td>(1089, 47)</td>
</tr>
</tbody>
</table>

We must note that although the last table entry on table didn’t converge according to our convergence definition, we got an error margin of 0.99 on the last iteration and it had just been starting to fastly converge. The results seem very positive however for the table values of $\nu \leq 1/1000$ $m^2/s$ we only get convergence for DOF=1089. This may seem odd, but the reader should be remembered that our heuristic depends on the macroelements as well, which means that the chosen heuristic cannot handle all situations properly.

Figure 3: Stationary Navier-Stokes cavity flow solution with $\nu = 1 \times 10^{-3}$ $m^2/s$(solution calculated with Newton SUPG)

Figure 3 denotes one of the many images computed using our implementation of Newton SUPG stabilization method for a cavity problem with $\nu = 1/1000$ $m^2/s$.
4 Non-Stationary Navier-Stokes Equations

In this section, we just expose the standard non-stationary Navier-Stokes equations formulation, its weak form and how we have actually solved it. Further on, we will also present some examples and a stability result. The non-stationary form of the Navier-Stokes Equations is:

\[
\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} - \nu \Delta \vec{u} + \nabla p = \vec{f} \text{ in } \Omega \times (0, \infty) \tag{4.1}
\]

\[
\nabla \cdot \vec{u} = 0 \text{ in } \Omega \times (0, \infty) \tag{4.2}
\]

The way we adopted to solve this equation is by first doing time discretization and then in each time iterate solve a non-linear system using the aforementioned methods. The details are omitted, but the following is the final weak formulation for this problem: \(\forall n \geq 0, \text{ find } u^{n+1} \in H^1_0(\Omega), p^{n+1} \in L^2_0(\Omega), \) for any \(v \in H^1_0(\Omega) \) and \(q \in L^2(\Omega)\):

\[
\frac{1}{\Delta t}(u^{n+1}, v) + a(u^{n+1}, v) + b(v, p^{n+1}) + c(u^{n+1}, u^{n+1}, v) = (f^{n+1}, v) + \frac{1}{\Delta t}(u^n, v)
\]

\[
b(u, q) = 0
\]

The following is a stability estimate for the original time dependent Navier-Stokes problem (demonstration details are omitted).

\[
||u^N||^2_{L^2} + \alpha \Delta t \sum_{n=0}^{N-1} ||u^{n+1}||^2_V \leq
\]

\[
\frac{\Delta t}{\alpha} \sum_{n=0}^{N-1} ||F^{n+1}||^2_V + ||u_0||^2_{L^2}
\]

where \(F^n(v) = (f^n, v)\). As usual, we present also computed solutions to test our algorithms.

Figure (4) and (5) denote distinct frames of our computed solution to the Womersley flow. Our results look as expected, in fact, similar results can be found in [5]. Figures (6) and (7) denote the result of a computed solution for a turbulent flow with an obstacle.

Figures (6) and (7) show both components of the velocity solution \(u_1\) and \(u_2\) at two different time frames. Figures (8) and (9) denote the solution \(||u||^2\) at two different time frames. Our results seem very promising, and appear to be the expected ones when comparing with similar results in [13].

5 Conclusion

We have shown that the proposed numerical implementation can handle with different benchmark idealized problems in the two dimensional frame. It is, of course, far from being completed, if one aims to use it for realistic 3D simulations. Several improvements can be introduced such as adaptive meshes for both the domain level and the time discretization. Furthermore, parallelization tech-
Figure 7: Obstacle based flow solution with $t = 4.0$.

Figure 8: Obstacle based flow solution with $t = 1.50$.

Techniques should be considered in order to manage the tremendous computational effort required for realistic simulations. One interesting work to follow would be to port those parallelized algorithms to a platform such as CUDA (from NVIDIA), and take advantage of the huge parallelization power offered by the GPU.

6 Download Links

The following link contains an archive with all the code samples used throughout this thesis along with the associated algorithms.

https://drive.google.com/open?id=0B7jM6LDiQSvvNno0ZjhHcGVVWkE

A more detailed version of these links with description can be found in [11].

References


