# Locally chosen grad-div stabilization parameters for finite element discretizations of incompressible flow problems

Nathan D. Heavner * Sponsor: Leo G. Rebholz †

**Abstract**

Grad-div stabilization has recently been found to be an important tool for finite element method simulations of incompressible flow problems, acting to improve mass conservation in solutions and reducing the effect of the pressure error on the velocity error. Typically, the associated stabilization parameter is chosen globally, but herein we consider local choices. We show that, both for an analytic test problem and for lift and drag calculations of flow around a cylinder, local choices of the grad-div stabilization parameter can provide more accurate solutions.

## 1 Introduction

The finite element method is a highly successful tool for approximating solutions to a large class of partial differential equations. However, as with all numerical solution techniques, it is subject to many sources of error. Our interest herein is how the weak enforcement of mass conversation can cause catastrophic error in simulations of incompressible flow problems such as Stokes and Navier-Stokes equations. This topic has recently become a subject of significant interest, leading to studies of stabilization techniques [20, 21, 19, 1, 22, 4, 6, 2, 11, 8] and divergence free elements [28, 5, 23, 26, 7, 17, 25, 27, 18, 24]. Although divergence free elements are interesting and even ideal in some situations, our focus herein will be on grad-div stabilization, which acts to reduce the negative effects that arise from the weak enforcement of mass conservation. While divergence free elements provide pointwise mass conservation and eliminate the effect of pressure error on velocity error, their stability relies on special mesh constructions and choices of approximating polynomial degree. Grad-div stabilization, however, can be used with any element choice or mesh and can be

*Department of Mathematical Sciences, Clemson University, nheavne@clemson.edu, partially supported by NSF grant DMS1112593.

†Department of Mathematical Sciences, Clemson University, rebholz@clemson.edu.

easily added to an existing code.

Since grad-div stabilization is both a useful and relatively new method, there is much interest in developing an understanding of it to maximize its effectiveness, in particular with the choice of the associated stabilization parameter. Typically, this parameter is chosen to be a global constant, and recent work in [12] gives guidance on how to choose it. However, we propose to instead choose the parameter locally and will demonstrate that doing so can produce significantly more accurate solutions.

For the commonly used Taylor-Hood elements, where the velocity approximating polynomials will be one degree higher than for pressure, theoretical analysis and numerical simulations originally performed in [20] and later in [15] indicate that $\gamma = \mathcal{O}(1)$ is often a good choice. However, in [9] it was shown that in certain situations, an optimal $\gamma$ can be much larger than $\mathcal{O}(1)$, depending on the size of the pressure relative to the size of the velocity, i.e. the size of $\frac{|p|_{H^k}}{|\mathbf{u}|_{H^s}}$ (where $s-1$, $k$ are the finite element approximating polynomial degrees for velocity and pressure, and $|\cdot|_{H^k}$ denotes the $H^k(\Omega)$ seminorm, as well as kinematic viscosity and structure of the mesh. In [9], it was shown both analytically and with numerical studies that $\gamma = 10^4$ can yield far better results than $\gamma = \mathcal{O}(1)$ for a natural convection problem.

The next natural investigation regarding the grad-div stabilization parameter is determining whether it should be chosen locally, and if so, by what criteria? This question has been posed in [21], where the authors considered local polynomial expansion for a particular element choice. While this is a good start to our knowledge, however, no conclusive investigation has yet been done to determine whether choosing $\gamma$ locally rather than globally can significantly improve solution accuracy; this paper addresses that topic. It is the purpose of this paper to show test problems, including physically relevant ones, where choosing the parameter locally produces a significantly better solution than choosing it globally.

This paper is arranged as follows. First, we will discuss some of the analysis behind grad-div stabilization as well as a benchmark problem which clearly displays its effectiveness. Then, we will present two test problems where a locally chosen $\gamma$ provides more accurate results than any globally chosen one. Finally, we will analyze our results and discuss possible future work.

## 2 Preliminaries

We consider a domain $\Omega \subset \mathcal{R}^d$ ($d = 2$ or 3) to be a convex polygon. Define the natural velocity and pressure spaces to be the Hilbert spaces $X = H_0^1(\Omega)^d$ and $Q = L_0^2(\Omega)$, which are defined by

$$X = \{\mathbf{v} : \Omega \to \mathbb{R}^d : \mathbf{v} \in H^1(\Omega)^d \text{ and } \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega\},$$
$$Q = \{q : \Omega \to \mathbb{R} : \int_\Omega |q|^2 dx < \infty, \int_\Omega q\, dx = 0\}.$$

We will denote the $L^2$ inner product by $(\cdot, \cdot)$. Recall that this is defined by $(u, v) = \int_\Omega uv\ dx$. The $L^2$ norm, $||\cdot||$, is defined by $||u|| = \sqrt{(u, u)}$ for $u \in L^2(\Omega)$.

A regular, conforming triangulation of the domain $\Omega$ will be denoted by $\tau_h(\Omega)$. The finite element spaces $(X_h, Q_h) \subseteq (X, Q)$ are defined by

$$X_h = X_h(\tau_h(\Omega)) = \{\mathbf{v}_h \in X \,|\, \mathbf{v}_h \in C^0(\Omega) \cap P_k(e)\, \forall e \in \tau_h(\Omega)\},$$
$$Q_h = Q_h(\tau_h(\Omega)) = \{q_h \in Q \,|\, q_h \in C^0(\Omega) \cap P_s(e)\, \forall e \in \tau_h(\Omega)\},$$

where $P_k(e)$ denotes polynomials of degree $k$ on each element. The most common choice of velocity pressure elements is the $k = 2$, $s = 1$ case, which is called Taylor-Hood. We will refer to these types of elements by the notation $(P_k, P_s)$.

# 3 Grad-div Stabilization

To understand how grad-div stabilization works, consider the Stokes equations:

$$\begin{cases} -\nu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u}\big|_{\partial\Omega} = \mathbf{0}, \end{cases} \tag{3.1}$$

which has variational formulation: find $(\mathbf{u}, p) \subset (X, Q)$ such that

$$\begin{aligned} \nu(\nabla\mathbf{u}, \nabla\mathbf{v}) - (p, \nabla\cdot\mathbf{v}) &= (\mathbf{f}, \mathbf{v}) \quad \forall \mathbf{v} \in X, \\ (\nabla\cdot\mathbf{u}, q) &= 0 \qquad \forall q \in Q. \end{aligned} \tag{3.2}$$

Thus, the finite element formulation that arises is: find $(\mathbf{u}_h, p_h) \in (X_h, Q_h)$ satisfying

$$\begin{aligned} \nu(\nabla\mathbf{u}_h, \nabla\mathbf{v}_h) - (p_h, \nabla\cdot\mathbf{v}_h) &= -(\mathbf{f}, \mathbf{v}_h) \quad \forall \mathbf{v}_h \in X_h, \\ (\nabla\cdot\mathbf{u}_h, q_h) &= 0 \qquad \forall q_h \in Q_h. \end{aligned} \tag{3.3}$$

Note that (2.3.b) does not imply $\nabla \cdot \mathbf{u}_h = 0$. However, since $\mathbf{u}_h = \mathbf{0}$ on $\partial\Omega$, $\int_{\partial\Omega} \mathbf{u}_h \cdot \mathbf{n}\, ds = 0$, so by the divergence theorem, $\int_\Omega \nabla \cdot \mathbf{u}_h = 0$. Thus, mass is conserved in a global sense. Also, the standard error estimate for (2.3) is given by [14]

$$\nu||\nabla(\mathbf{u} - \mathbf{u}_h)||^2 \leq C\left(\nu \inf_{\mathbf{v}_h \in X_h} ||\nabla(\mathbf{u} - \mathbf{v}_h)||^2 + \nu^{-1} \inf_{q_h \in Q_h} ||p - q_h||^2\right), \tag{3.4}$$

and thus for $(P_2, P_1)$ Taylor-Hood elements,

$$\nu||\nabla(\mathbf{u} - \mathbf{u}_h)||^2 \leq Ch^4(\nu|\mathbf{u}|^2_{H^3} + \nu^{-1}|p|^2_{H^2}). \tag{3.5}$$

Since $||\nabla \cdot \mathbf{u}_h|| \leq ||\nabla \cdot (\mathbf{u} - \mathbf{u}_h)|| \leq ||\nabla(\mathbf{u} - \mathbf{u}_h)||$, it is true that the divergence error is optimal for $(P_k, P_{k-1})$ Taylor-Hood elements, in the sense of divergence error converging to zero with rate $\mathcal{O}(h^k)$ as the mesh width tends to zero. However, for practical computations, there is a finest mesh

on which one can compute, and so a small viscosity $\nu$ and large/complex pressure $p$ can lead to large divergence errors, despite it being optimal. Moreover, when $\nabla \cdot \mathbf{u}_h \neq 0$, the pressure discretization error can have a large input on the velocity error. Several examples of this phenomena are discussed in [6].

Now consider the same Stokes problem, but include grad-div stabilization by adding the term $-\gamma\nabla(\nabla \cdot \mathbf{u}) = \mathbf{0}$ to the momentum equation. This leads to the finite element formulation

$$\begin{aligned} \nu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) - (p_h, \nabla \cdot \mathbf{v}_h) + \gamma(\nabla \cdot \mathbf{u}_h, \nabla \cdot \mathbf{v}_h) &= (\mathbf{f}, \mathbf{v}_h) \quad \forall \mathbf{v}_h \in X_h, \\ (\nabla \cdot \mathbf{u}_h, q_h) &= 0 \qquad \forall q_h \in Q_h. \end{aligned} \tag{3.6}$$

By penalizing the divergence, $||\nabla \cdot \mathbf{u}||$ is decreased, and in turn decreases the error caused by weak mass conservation. This can be seen by choosing $\mathbf{v}_h = \mathbf{u}_h$ and $q_h = p_h$, which provides

$$\nu||\nabla \mathbf{u}_h||^2 + \gamma||\nabla \cdot \mathbf{u}_h||^2 = (\mathbf{f}, \mathbf{u}_h) = ||\nabla \mathbf{u}_h|| \frac{(\mathbf{f}, \mathbf{u}_h)}{||\nabla \mathbf{u}_h||} \leq ||\nabla \mathbf{u}_h|| ||\mathbf{f}||_{H^{-1}}. \tag{3.7}$$

Applying Young's inequality to the right hand side and then dropping the $\frac{\nu}{2}||\nabla \mathbf{u}_h||$ term gives

$$\gamma||\nabla \cdot \mathbf{u}_h||^2 \leq \frac{\nu^{-1}}{2}||\mathbf{f}||^2_{H^{-1}}. \tag{3.8}$$

Moreover, the error estimate becomes [22]:

$$\nu||\nabla(\mathbf{u} - \mathbf{u}_h)||^2 \leq C \left( \inf_{\mathbf{v}_h \in X_h} (\nu + \gamma)||\nabla(\mathbf{u} - \mathbf{v}_h)||^2 + \gamma^{-1} \inf_{q_h \in Q_h} ||p - q_h||^2 \right), \tag{3.9}$$

and thus for $(P_2, P_1)$ elements,

$$\nu||\nabla(\mathbf{u} - \mathbf{u}_h||^2 \leq Ch^4 \left( (\nu + \gamma)|\mathbf{u}|^2_{H^3} + \gamma^{-1}|p|^2_{H^2} \right). \tag{3.10}$$

Comparing to (2.5), observe that for complex pressure and small $\nu$, the velocity error will be reduced if $\gamma > \nu$, so long as the pressure error term in (2.7) is dominant.

To further illustrate the point that grad-div stabilization can increase accuracy, consider 2D

channel flow past a forward-backward facing step. In this benchmark test problem, the channel is given a length of 40 and height of 10. A square step of side length 1 is placed at the bottom of the channel with the left-most side of the step at $x = 5$. As described in [16], the boundary condition for the inflow and outflow is $\mathbf{u} = (y(10 - y)/25, 0)^T$. On the top and bottom walls, a no-slip boundary condition is enforced. In a correct solution with $\mathbf{f} = \mathbf{0}$ and $\nu = \frac{1}{600}$, the flow through the channel is smooth except for the occurrence of recirculating vortices that occur in the immediate wake of the step [16].

The Navier-Stokes equations are used to solve this time-dependent problem with a Crank-Nicolson temporal discretization and a $(P_2, P_1)$ finite element spatial discretization. We note that the analysis of error and effect of grad-div stabilization remains essentially the same in the Navier-Stokes case as in the Stokes case discussed above. Results are shown in Figure 1 as the velocity fields shown as streamlines over speed contours for the cases with ($\gamma = 1$) and without ($\gamma = 0$) grad-div stabilization at $t = 40$. In each case, a time step of $\Delta t = 0.025$ and a Delaunay mesh with 4,800 total degrees of freedom were used. When grad-div stabilization was not used, the weak mass conservation caused oscillations around the step. These oscillations compounded over time, quickly destroying the solution's accuracy. With the addition of the grad-div term, however, the oscillations were greatly reduced, and the solution matched the resolved solutions found in [16]. We note these computations were done using the software freefem++ [10].

## 4 Choosing the stabilization parameter locally

As seen in the error estimate (2.5), grad-div stabilization can reduce the velocity error by reducing the pressure discretization error term by scaling $\inf_{q_h \in Q_h} ||p - q_h||^2$ by $\gamma^{-1}$ instead of $\nu^{-1}$. As $\gamma$ is increased, the effect of this pressure term decreases, but the effect of the velocity term $\inf_{\mathbf{v}_h \in X_h} (\nu + \gamma)||\nabla(\mathbf{u} - \mathbf{v}_h)||^2$ increases, and hence $\gamma$ gives control over which error term is dominant. Thus, in cases where $p$ is large or complex and $\nu$ and $\mathbf{u}$ are small, a large global $\gamma$ is appropriate. However, there are situations where $p$ is small in some areas of a flow domain, but not others. In these situations, it seems intuitive that $\gamma$ should be large in some areas and small in others.

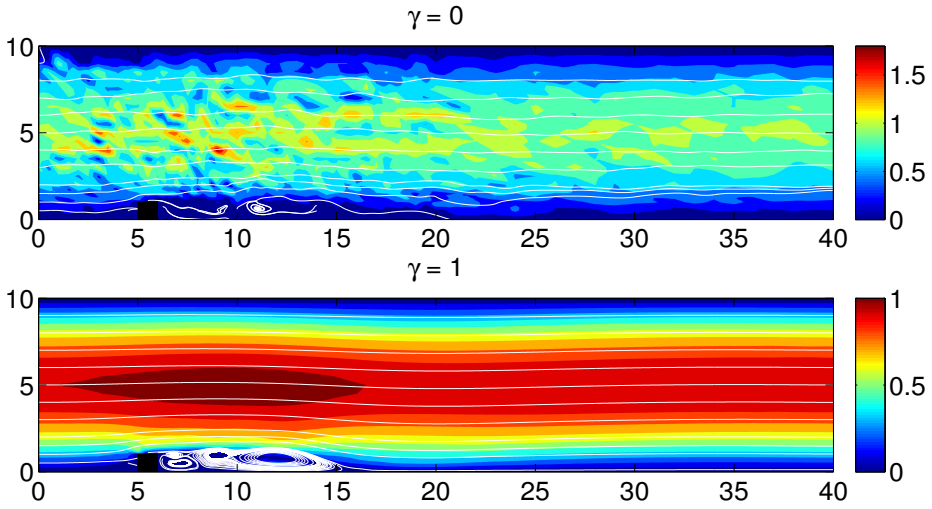



Figure 1: Fluid flow around a step using $(P_2, P_1)$ elements with $\nu = \frac{1}{600}$ at $t = 40$, with and without grad-div stabilization. These plots depict velocity streamlines over filled speed contours.

Consider again the error equation (2.7) and expand it as

$$\begin{aligned}
\nu||\nabla(\mathbf{u}-\mathbf{u}_h)||^2 &\leq C\left(\inf_{\mathbf{v}_h\in X_h}((\nu+\gamma)||\nabla(\mathbf{u}-\mathbf{v}_h)||^2)+\gamma^{-1}\inf_{q_h\in Q_h}||p-q_h||^2\right)\\
&\leq C\left(\inf_{\mathbf{v}_h\in X_h}\int_\Omega(\nu+\gamma)|\nabla(\mathbf{u}-\mathbf{v}_h)|^2dx+\inf_{q_h\in Q_h}\int_\Omega\gamma^{-1}|p-q_h|^2dx\right)\\
&\leq C\left(\inf_{\mathbf{v}_h\in X_h}\sum_{e\in\tau_h}\int_e(\nu+\gamma)|\nabla(\mathbf{u}-\mathbf{v}_h)|^2dx+\inf_{q_h\in Q_h}\sum_{e\in\tau_h}\int_e\gamma^{-1}|p-q_h|^2dx\right).
\end{aligned}$$

Suppose now that $\gamma$ is chosen element-wise: $\gamma|_e = \gamma_e$. Then we have

$$\nu||\nabla(\mathbf{u}-\mathbf{u}_h)||^2 \leq C\left(\inf_{\mathbf{v}_h\in X_h}\sum_{e\in\tau_h}\int_e(\nu+\gamma_e)|\nabla(\mathbf{u}-\mathbf{v}_h)|^2dx+\inf_{q_h\in Q_h}\sum_{e\in\tau_h}\int_e\gamma_e^{-1}|p-q_h|^2dx\right). \quad (4.1)$$

Since $\mathbf{v}_h$ and $q_h$ are continuous, the infimums will not directly allow for an element-wise comparison of velocity and pressure error. However, if we make the assumption that it can be well approximated

by the discontinuous polynomial functions, then

$$\begin{aligned}\nu||\nabla(\mathbf{u}-\mathbf{u}_h)||^2 &\leq C\left(\inf_{\mathbf{v}_h\in X_h}\sum_{e\in\tau_h}\int_e(\nu+\gamma_e)|\nabla(\mathbf{u}-\mathbf{v}_h)|^2dx+\inf_{q_h\in Q_h}\sum_{e\in\tau_h}\int_e\gamma_e^{-1}|p-q_h|^2dx\right)\\ &\approx C\left(\sum_{e\in\tau_h}\left[(\nu+\gamma_e)\inf_{\mathbf{v}_h\in P_k(e)}||\nabla(\mathbf{u}-\mathbf{v}_h)||_{L^2(e)}+\gamma^{-1}\inf_{q_h\in P_{k-1}(e)}||p-q_h||_{L^2(e)}\right]\right).\end{aligned}$$

For $(P_k, P_{k-1})$ elements, standard approximation theory (see, e.g. [3]) then gives

$$\nu||\nabla(\mathbf{u}-\mathbf{u}_h)||^2_{L^2(\Omega)}\lesssim C\sum_{e\in\tau_h}\left((\nu+\gamma_e)|\mathbf{u}|^2_{H^{k+1}(e)}+\gamma_e^{-1}|p|^2_{H^k(e)}\right), \tag{4.2}$$

which implies that when $|p|^2_{H^k(e)}\gg|\mathbf{u}|_{H^{k+1}(e)}$ and $\nu$ is small, $\gamma_e$ should be taken large, and vice versa.

For equal order elements, the estimate changes to

$$\nu||\nabla(\mathbf{u}-\mathbf{u}_h)||^2_{L^2(\Omega)}\lesssim C\sum_{e\in\tau_h}\left((\nu+\gamma_e)|\mathbf{u}|^2_{H^{k+1}(e)}+\gamma_e^{-1}h^2|p|^2_{H^{k+1}(e)}\right). \tag{4.3}$$

Although here the pressure term is scaled by $h^2$, the pressure seminorm is one degree higher. This estimate implies $\gamma_e$ should be taken large when $h|p|_{H^{k+1}(e)}\gg|\mathbf{u}|_{H^{k+1}(e)}$.

### 4.1 Analytic test problem with local grad-div stabilization

Our first test example is to consider (2.5) on $\Omega=(0,1)^2$ with $\gamma$ chosen locally. We choose an analytical solution

$$\begin{cases}\mathbf{u}_{true}=\begin{pmatrix}u_1^t\\ u_2^t\end{pmatrix}=\begin{pmatrix}\cos(\pi y)\\ \sin(\pi x)\end{pmatrix},\\ p_{true}=e^{4x}.\end{cases} \tag{4.4}$$

For the finite element problem, the boundary condition is enforced nodally as a Dirichlet condition using the true solution. The right hand side $\mathbf{f}$ is calculated using Stokes equations with $\nu=\frac{1}{4}$

and the true solution. For this analytical test problem, error can be computed to give an exact measurement of a solution's accuracy. We measured the $X_h$ error, which is equivalent to the $H^1(\Omega)$ error in this setting due to Poincaré's inequality [14]. We thus define error by $E = ||\nabla(\mathbf{u}_{true} - \mathbf{u})||$.

For this experiment, we calculated solutions using the scheme (2.6) with non-homogeneous Dirichlet boundary conditions, using the $(P_1^b, P_1)$ mini element (see, e.g. [14] for a detailed description of the mini element). A Delaunay mesh with $h = \frac{1}{64}$ was used, which provided 29,600 velocity degrees of freedom and 5,000 pressure degrees of freedom. First, we computed solutions using $\gamma$ chosen locally to be

$$\gamma|_e = 4e^{4x}.$$

This choice of $\gamma$ was made because the size of $|\mathbf{u}|_{H^{k+1}}$ is roughly constant across $\Omega$, but $|p|_{H^{k+1}}$ grows exponentially with $x$. Thus, this choice of $\gamma$ seems a reasonable first attempt at a local choice to reduce error over a constant choice. For comparison, we also calculated solutions using many choices of global $\gamma$'s, ranging from 0 to $10^4$, and recorded the error.

Results of these tests are shown in Figure 2 and Table 1. The optimal error when using a constant $\gamma$ was found to be $||\nabla(\mathbf{u} - \mathbf{u}_h)|| = 0.146$ when $\gamma = 100$. When the local $\gamma$ was used, the error was $||\nabla(\mathbf{u} - \mathbf{u}_h)|| = 0.100$. Observe that this local choice is about 33% better than the optimal global choice, and much better than when $\gamma = 1$ or $\gamma = \nu$.

Table 1: $H^1$ error for varying values of $\gamma$

| $\gamma$ | $H^1_{error}$ |
|---|---|
| 0 | 9.69 |
| 1 | 3.15 |
| 10 | 0.53 |
| 100 | 0.15 |
| 200 | 0.17 |
| 1000 | 0.32 |
| $4e^{4x}$ | 0.10 |

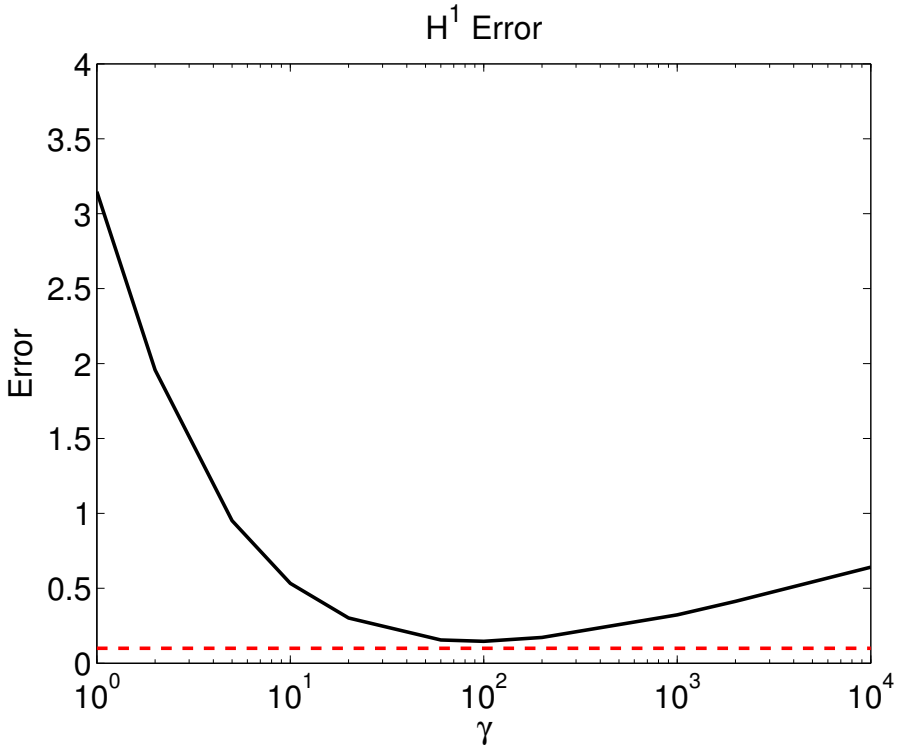



Figure 2: Shown above is the $H^1$ error of solutions generated from locally chosen and constant $\gamma$.

## 4.2 Flow around a cylinder

Our next test compares locally and globally chosen $\gamma$ in finite element computations for the benchmark problem of channel flow around a cylinder. The length and width of the channel are 2.2 and 0.41, respectively, and the center of the 0.1-diameter cylinder lies at the point $(0.2, 0.2)$. The boundary conditions for the inflow and outflow are set to be

$$\begin{cases} u_1(0, y, t) = u_1(2.2, y, t) = \frac{1}{0.41^2} \sin(\frac{3\pi t}{4} y(0.41 - y)) \\ u_2(0, y, t) = u_2(2.2, y, t) = 0. \end{cases} \tag{4.5}$$

In addition, no-slip boundary conditions were enforced along the top and bottom walls, and the initial condition is $\mathbf{u}(x, y, 0) = \mathbf{0}$. The viscosity is chosen to be $\nu = \frac{1}{1{,}000}$ and the external force $\mathbf{f} = \mathbf{0}$.

When a fluid flows past a surface such as a cylinder, it exerts a force on the surface. Lift is defined to be the component of that force that is exerted perpendicular to the oncoming flow direction, while drag is defined to be the parallel component of the force. This problem was chosen because it is of physical interest and the lift and drag around the cylinder can be computed, providing a quantitative measure of solution accuracy even though the problem seemingly has no analytical solution. As given in [13], lift ($c_l(t)$) and drag ($c_d(t)$) for this test problem are defined as follows:

$$c_d(t) = \frac{20}{U_{max}^2} \int_S \left( \rho\nu \frac{\partial \mathbf{u}_{t_S}(t)}{\partial n} n_y - p n_x \right) dS, \tag{4.6}$$

$$c_l(t) = -\frac{20}{U_{max}^2} \int_S \left( \rho\nu \frac{\partial \mathbf{u}_{t_S}(t)}{\partial n} n_x + p n_y \right) dS. \tag{4.7}$$

For a description of the parameters of this definition and reference values for $c_{d,max}$ and $c_{l,max}$, see [13].

A barycenter-refined mesh with $(P_2, P_1)$ elements was used, which gave 9,900 velocity degrees of freedom and 1,270 pressure degrees of freedom. The scheme used to compute the solution $\mathbf{u}$ is

given by

$$\begin{aligned} \tfrac{1}{\Delta t}(\mathbf{u}_h^{n+1} - \mathbf{u}_h^n, \mathbf{v}_h) + (\mathbf{u}_h^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}_h^{n+\frac{1}{2}}, \mathbf{v}_h) - (p_h^{n+\frac{1}{2}}, \nabla \cdot \mathbf{v}_h) + \nu(\nabla \mathbf{u}_h^{n+\frac{1}{2}}, \nabla \mathbf{v}_h) &= \mathbf{0} \quad \forall \mathbf{v}_h \in X_h, \\ (\nabla \cdot \mathbf{u}_h^{n+1}, q_h) &= 0 \quad \forall q_h \in Q_h. \end{aligned} \tag{4.8}$$

We computed solutions using many constant $\gamma$'s, calculated the lift and drag coefficients from each solution, and compared their maximum values to the reference values from [13]. Next, we repeated this process using several different choices of $\gamma$ chosen element-wise. Each choice consisted of choosing a disk of a certain thickness (indicated by $\rho$ in the results below) around the cylinder in which to apply stabilization with $\gamma = 1$. The optimal choice $\gamma_g$, in which $\gamma$ was chosen to provide stabilization only in the immediate wake of the cylinder, as well as another local choice, where $\gamma = 1$ only in a radius around the cylinder, are shown below in Figure 3.

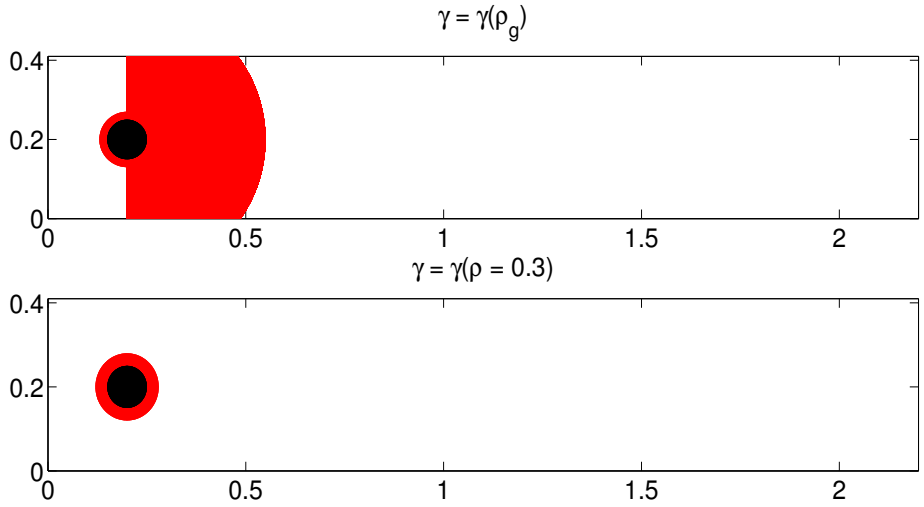
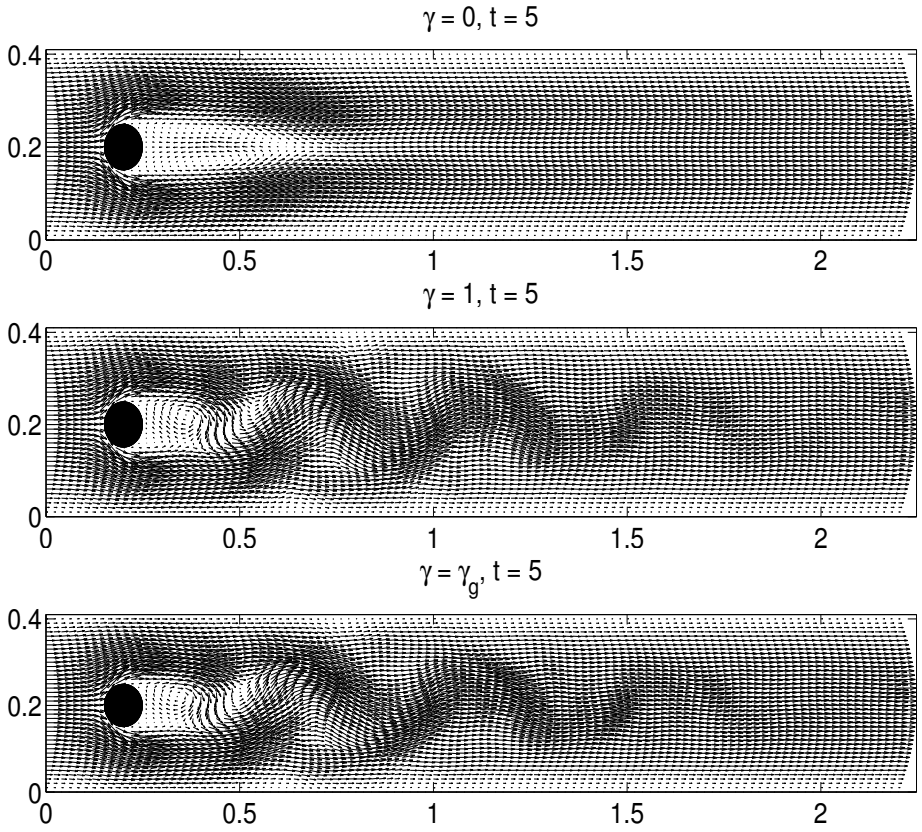



Figure 3: This figure depicts how $\gamma$ was chosen locally to obtain the results shown in Table 2. $\gamma = 1$ in the shaded areas and $\gamma = 0$ elsewhere.

The results of the tests are shown in Figures 4 and 5 and Table 2. It is evident from the plots of the solution that using no grad-div stabilization produces an incorrect result while using a constant or local $\gamma$ for stabilization yields solutions that look visually similar. When the more precise measure of solution accuracy, $c_{d,max}(t)$ and $c_{l,max}(t)$ are computed, however, we see that many choices for a local $\gamma$ yielded good or slightly better results than a constant $\gamma$. $\gamma_g$, on the

other hand, produced a solution that was significantly more accurate than any constant $\gamma$.



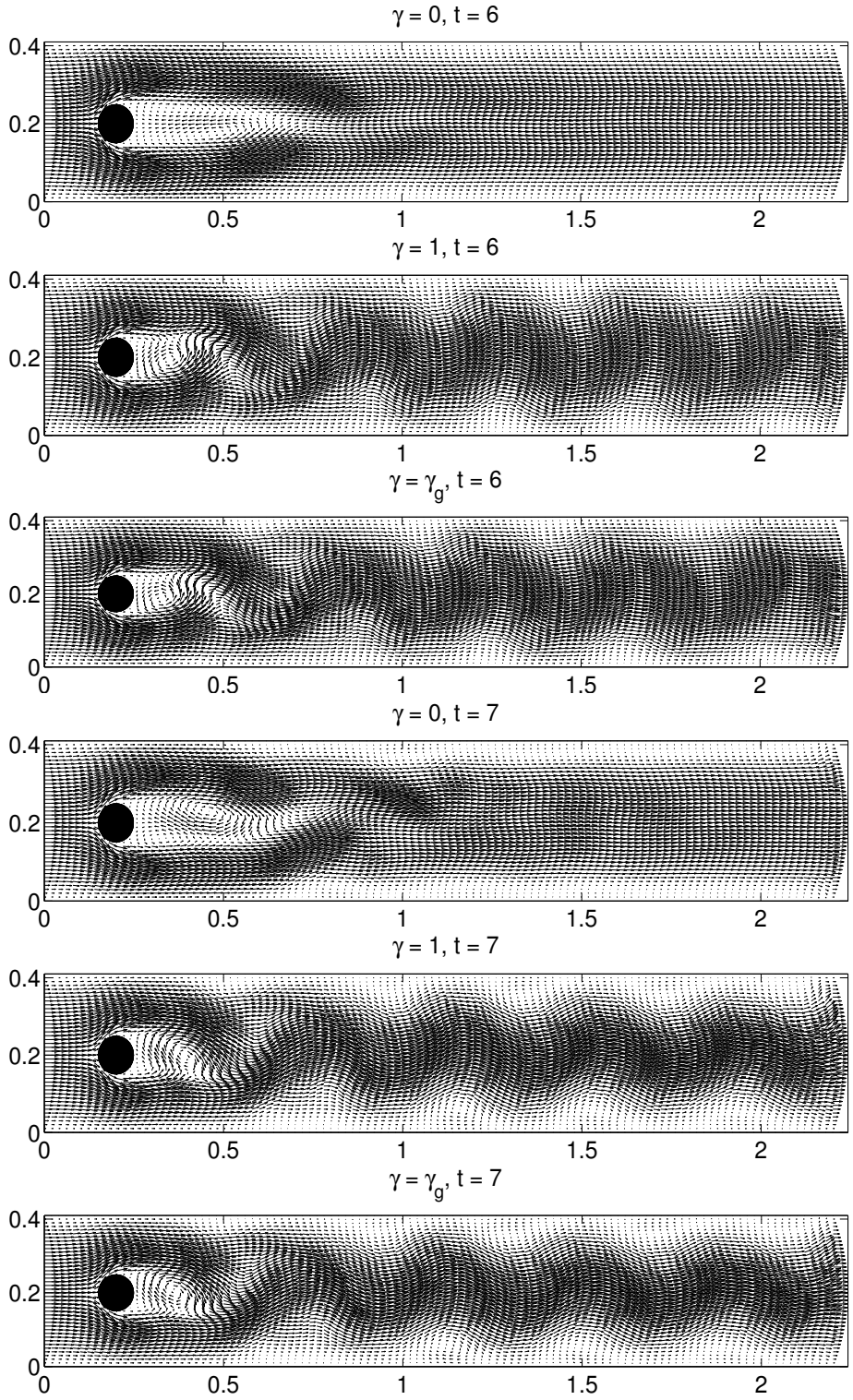



Figure 4: These plots depict the velocity $\mathbf{u}$ given by using $\gamma = 0, 1, \gamma_g$ at $t = 5, 6, 7$.

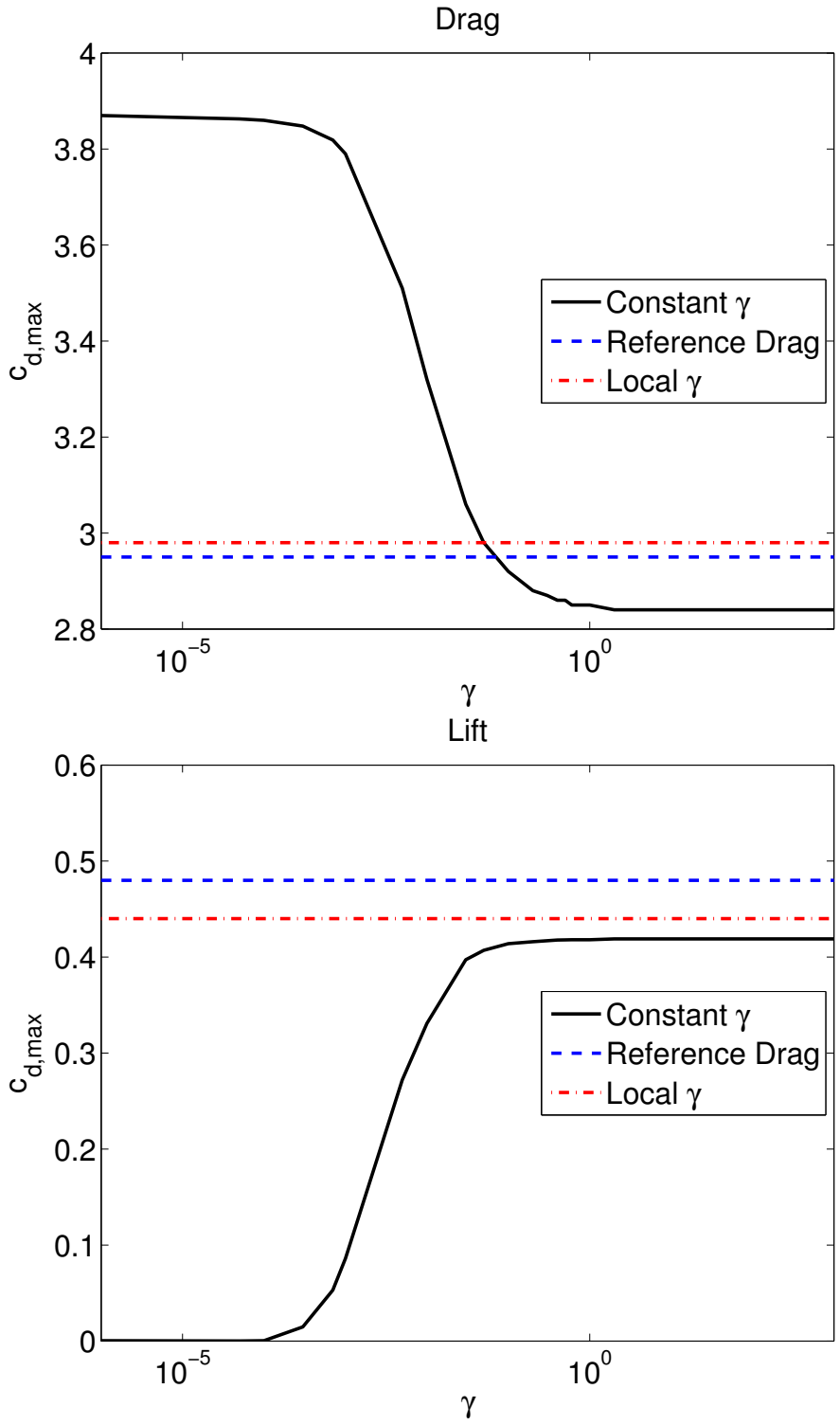



Figure 5: These plots compare locally chosen $\gamma$, constant $\gamma$, and the reference values used as the true solution.

Table 2: Lift and drag for varying values of $\gamma$. 2.94 and 0.48 are the reference values for drag and lift, respectively.

| $\gamma$ | drag | lift |
|---|---|---|
| 0 | 3.87 | 0.00 |
| 1 | 2.85 | 0.42 |
| $\gamma(\rho = 0.02)$ | 2.95 | 0.03 |
| $\gamma(\rho = 0.2)$ | 2.84 | 0.04 |
| $\gamma(\rho = 0.3)$ | 2.85 | 0.42 |
| $\gamma(\rho = \rho_g)$ | 2.98 | 0.44 |

# 5 Conclusion

We have shown that in certain cases, choosing the stabilization parameter $\gamma$ locally instead of globally can significantly improve results in solution accuracy for finite element computation of Stokes and Navier-Stokes equations. In particular, when pressure is large or complex only in parts of the domain, such as in our analytic test problem, choosing $\gamma$ to be large only in the area of large pressure can improve upon the solution produced by choosing a constant $\gamma$. In other cases, such as the cylinder test problem, providing stabilization only in critical areas can significantly improve results.

# References


[1] Bowers, A., Cousins, B., Linke, A., and Rebholz, L. New connections between finite element formulations of the Navier-Stokes equations. Journal of Computational Physics 229, 24 (2010), 9020–9025.

[2] Braack, M., Burman, E., John, V., and Lube, G. Stabilized finite element methods for the generalized Oseen problem. Computer Methods in Applied Mechanics and Engineering 196, 4–6 (2007), 853–866.

[3] Brenner, S., and Scott, L. R. The Mathematical Theory of Finite Element Methods. Springer-Verlag, 1994.

[4] Burman, E., and Hansbo, P. Edge stabilization for the generalized Stokes problem: a continuous interior penalty method. Comp. Methods Appl. Mech. Engrg. 195, 19–22 (2006), 2393–2410.

[5] Carrero, J., Cockburn, B., and Schötzau, D. Hybridized globally divergence-free LDG methods. I. The Stokes problem. Math. Comp. 75, 254 (2006), 533–563.

[6] Case, M., Ervin, V., Linke, A., and Rebholz, L. A connection between Scott-Vogelius elements and grad-div stabilization. SIAM Journal on Numerical Analysis 49, 4 (2011), 1461–1481.

[7] Cockburn, B., Kanschat, G., and Schötzau, D. A note on discontinuous Galerkin divergence-free solutions of the Navier-Stokes equations. J. Sci. Comput. 31, 1-2 (2007), 61–73.

[8] Franca, L., and Hughes, T. Two classes of mixed finite element methods. Computer Methods in Applied Mechanics and Engineering 69, 1 (1988), 89–129.

[9] Galvin, K., Linke, A., Rebholz, L., and Wilson, N. Stabilizing poor mass conservation in incompressible flow problems with large irrotational forcing and application to thermal convection. Computer Methods in Applied Mechanics and Engineering 237 (2012), 166–176.

[10] Hecht, F. New development in freefem++. J. Numer. Math. 20, 3-4 (2012), 251–265.

[11] Heister, T., and Rapin, G. Efficient augmented Lagrangian-type preconditioning for the Oseen problem using grad-div stabilization. Int. J. Numer. Meth. Fluids 71 (2013).

[12] Jenkins, E., John, V., Linke, A., and Rebholz, L. On the parameter choice in grad-div stabilization for incompressible flow problems. Advances in Computational Mathematics to appear (2013).

[13] John, V. Reference values for drag and lift of a two dimensional time-dependent flow around a cylinder. International Journal for Numerical Methods in Fluids 44 (2004), 777–788.

[14] Layton, W. An Introduction to the Numerical Analysis of Viscous Incompressible Flows. SIAM, Philadelphia, 2008.

[15] Layton, W., Manica, C., Neda, M., Olshanskii, M. A., and Rebholz, L. On the accuracy of the rotation form in simulations of the Navier-Stokes equations. Journal of Computational Physics 228, 9 (2009), 3433–3447.

[16] Layton, W., Manica, C., Neda, M., and Rebholz, L. Numerical analysis and computational testing of a high accuracy Leray-deconvolution model of turbulence. Numerical Methods for Partial Differential Equations 24, 2 (2008), 555–582.

[17] Linke, A. Divergence-free mixed finite elements for the incompressible Navier-Stokes Equation. PhD thesis, University of Erlangen, 2008.

[18] Linke, A., Matthies, G., and Tobiska, L. Non-nested multi-grid solvers for mixed divergence-free Scott-Vogelius discretizations. Computing 83, 2-3 (2008), 87–107.

[19] Linke, A., Rebholz, L., and Wilson, N. On the convergence rate of grad-div stabilized Taylor-Hood to Scott-Vogelius solutions for incompressible flow problems. Journal of Mathematical Analysis and Applications 381 (2011), 612–626.

[20] Olshanskii, M. A. A low order Galerkin finite element method for the Navier-Stokes equations of steady incompressible flow: a stabilization issue and iterative methods. Comput. Meth. Appl. Mech. Eng. 191 (2002), 55155536.

[21] Olshanskii, M. A., Lube, G., Heister, T., and Löwe, J. Grad-div stabilization and subgrid pressure models for the incompressible Navier-Stokes equations. Comput. Methods Appl. Mech. Engrg. 198, 49-52 (2009), 3975–3988.

[22] Olshanskii, M. A., and Reusken, A. Grad-div stabilization for Stokes equations. Math. Comp. 73, 248 (2004), 1699–1718.

[23] Wang, J., Wang, Y., and Ye, X. A robust numerical method for Stokes equations based on divergence-free $H(\mathrm{div})$ finite element methods. SIAM J. Sci. Comput. 31, 4 (2009), 2784–2802.

[24] Zhang, S. On the family of divergence-free finite elements on tetrahedral grids for the Stokes equations. Preprint University of Delaware (2007).

[25] Zhang, S. On the P1 Powell-Sabin divergence-free finite element for the Stokes equations. J. Comput. Math. 26, 3 (2008), 456–470.

[26] Zhang, S. A family of $Q_{k+1,k} \times Q_{k,k+1}$ divergence-free finite elements on rectangular grids. SIAM J. Numer. Anal. 47, 3 (2009), 2090–2107.

[27] Zhang, S. Quadratic divergence-free finite elements on Powell-Sabin tetrahedral grids. Calcolo 48, 3 (2011), 211–244.

[28] Zhang, S. Bases for C0-P1 divergence-free elements and for C1-P2 finite elements on union jack grids. Submitted (2012).

# 6 Appendix

## 6.1 Freefem++ Code for the Analytic Test Problem

```
// defining the boundary
// the triangulated domain Th is on the left side of its boundary

verbosity = 0;

int it;
int nt = 10; //number of iterations

//variables
real theta = 0;
real L = 1;

border A1(t=0,1){x=t; y=0; label=1;};
border A2(t=0,1){x=1; y=t; label=2;};
border A3(t=0,1){x=1-t; y=1; label=3;};
border A4(t=0,1){x=0; y=1-t; label=4;};

real mshsize = 80;

mesh Th = buildmesh(A1(mshsize)+A2(mshsize)+A3(mshsize)+A4(mshsize));

plot(Th, wait=1);

fespace Xh(Th,P1b);
```

```
fespace Qh(Th, P1);

// FEM variables
Xh u1, u2, v1, v2, gamma;
Qh p, q;

real nu = .25;
real pi = 3.14159265359;
real H1err;

func f1 = nu*pi^2*cos(pi*y) + 12*exp(12*x);
func f2 = nu*pi^2*sin(pi*x);

func utrue1 = cos(pi*y);
func utrue2 = sin(pi*x);
func utrue1x = 0.0;
func utrue1y = -pi*sin(pi*y);
func utrue2x = pi*cos(pi*x);
func utrue2y = 0.0;
Xh ptrue = exp(12*x);

real gammaval;
func gammafun = 500;//.02*exp(12*x);//100*sqrt(dx(ptrue)^2 + dy(ptrue)^2);

//create vectors to use in giving output later
int n=75,m=75;
real[int] xvec(n);
real[int] yvec(m);
```

```
  for (int ix = 0; ix < n ; ix++){
     for (int iy = 0; iy < m ; iy++){
          xvec[ix]=1.0*ix/(n-1);
          yvec[iy]=1.0*iy/(m-1);
     }
  }

/* Solve Stokes: -nu Laplacian(u) + grad(p)=f, div u=0 */

problem Stokes([u1, u2, p], [v1, v2, q], solver = UMFPACK) =
int2d(Th)
(
  // (grad u,grad v)
  nu*(dx(u1)*dx(v1) + dy(u1)*dy(v1) + dx(u2)*dx(v2) + dy(u2)*dy(v2))
  // -(p, div v)
    - p*(dx(v1) + dy(v2))
  // + (div u,q)
      + (dx(u1) + dy(u2))*q
  // + eps*(p,q)
       + p*q*0.0000000001
 // (div u,div v)
     + gammafun*(dx(u1)+dy(u2))*( dx(v1)+dy(v2) )
)
- int2d(Th)
(
  // (f,v)
  f1*v1 + f2*v2
)
```

```
/* bc for test problem */
+ on(1,2,3,4, u1 = utrue1, u2 = utrue2);

//algorithm starts here
Stokes;

H1err = sqrt(int2d(Th)( (utrue1x-dx(u1))^2 + (utrue2x-dx(u2))^2 +
(utrue1y-dy(u1))^2 + (utrue2y-dy(u2))^2));

cout << "H1err" << H1err << endl;

plot(u1, wait = 1);
plot(u2, wait = 1);
plot(p,wait=1);
plot([u1, u2], value=true, coef=0.1, wait = 1);
```

## 6.2 Freefem++ Code for the Flow Around a Cylinder Problem

```
/***********************************************/
/***** domain/mesh for cylinder problem  **************/
int nNS1=14,nNS=50,mNS=10,mNS1=20,nCNS=15; // Mesh 1 - 7600
//int nNS1=24,nNS=90,mNS=18,mNS1=28,nCNS=60; // Mesh 2 - 33000
//int nNS1=32,nNS=120,mNS=24,mNS1=32,nCNS=80; // Mesh 3 - 56000
//int nNS1=64,nNS=240,mNS=48,mNS1=64,nCNS=160; // Mesh 4 - 230000

border floorC(t=0,0.41){x=t;y=0;label=1;};
border floor(t=0.41,2.2){x=t;y=0;label=11;};
```

```
border right(t=0,0.41){x=2.2;y=t;label=4;};
border ceilingC(t=0.41,0){x=t;y=0.41;label=5;};
border ceiling(t=2.2,0.41){x=t;y=0.41;label=111;};
border left(t=0.41,0){x=0;y=t;label=8;};
border C(t=0,2*pi){ x = 0.2+0.05*cos(t); y = 0.2+0.05*sin(t); label=99;}
mesh Th = readmesh("Cylinder_mesh1_BC.msh");

plot(Th);

real dt = 0.001;
real nbiter = 6500;

real uval = 0,pval=0;
real vval = 0;
real divval = 0;
real pxval = 0, pyval = 0;
real u1xval = 0, u1yval = 0, u2xval = 0, u2yval = 0;

real deltap;
real cd, cdVisc, cdConvec, cdPressure;
real cl, clVisc, clConvec, clPressure;
real maxcd, timemaxcd, maxcl, timemaxcl;

//func gammafun = bool( (((Th[Th(x,y).nuTriangle][0].x+Th[Th(x,y).nuTriangle][1].x+
Th[Th(x,y).nuTriangle][2].x) / 3 <= sqrt(0.05^2 - ((Th[Th(x,y).nuTriangle][0].y+
Th[Th(x,y).nuTriangle][1].y+Th[Th(x,y).nuTriangle][2].y)/3 - 0.2)^2) + 0.2)
&& ((Th[Th(x,y).nuTriangle][0].x+Th[Th(x,y).nuTriangle][1].x+
Th[Th(x,y).nuTriangle][2].x) / 3) >= 0.2)   || (((Th[Th(x,y).nuTriangle][0].x+
```

```
Th[Th(x,y).nuTriangle][1].x+Th[Th(x,y).nuTriangle][2].x) / 3
>= -sqrt(0.08^2 - ((Th[Th(x,y).nuTriangle][0].y+Th[Th(x,y).nuTriangle][1].y+
Th[Th(x,y).nuTriangle][2].y)/3 - 0.2)^2) + 0.2)
&& ((Th[Th(x,y).nuTriangle][0].x+Th[Th(x,y).nuTriangle][1].x+
Th[Th(x,y).nuTriangle][2].x) / 3) <= 0.2) );

real gammaval = 0.0;



// Define the finite element spaces.
fespace Xh(Th,P2);
fespace Qh(Th,P1);

// Velocity variables
Xh u2,v2,un2,unn2,e2,wbar1,wbar2,wn1,wn2,w1,w2,
      wb1,wb2,temp1,temp2,v3,v4,vv1,vv2,phiTest1,phiTest2,gamma;
Xh u1,v1,un1,unn1,e1,phi1,phi2;
Qh pm=0, p=0, q, lambda;
Xh u1f,u2f,v1f,v2f,Ggw1,Ggw2, v1lift,v2lift,v1drag,v2drag;
Qh qf,pf;

func gammafun = 0.03;//abs(p / (dx(u1) + dy(u2) + .000001)) * (1/10000);

real Re=1000.0;
real nu=1.0/Re;
real PI = 3.141592653589793;
real t=0;
```

```
func g = (1./(0.41*0.41))*sin(pi*t/8.)*6.*y*(0.41-y);
u1 = 0; u2 = 0;
un1=u1; un2=u2;  // u^n
w1=0;
w2=0;
u1f=0;
u2f=0;
phi1=0;
phi2=0;

int itnl=0,iter=0,MaxNlIt=30;

int n=100,m=50;
real[int] xvec(n);
real[int] yvec(m);
  for (int ix = 0; ix < n ; ix++){
     for (int iy = 0; iy < m ; iy++){
          xvec[ix]=2.2*ix/(n-1);
          yvec[iy]=0.41*iy/(m-1);
     }
  }


problem cnle([u1,u2,p],[v1,v2,q],solver=UMFPACK) =
// use w1 and w2 as the filtered extrapolated term
    int2d(Th)(
               1.0/dt * ( u1*v1 + u2*v2 )
             - p * ( dx(v1) + dy(v2) )
```

```
             + 0.5* nu * ( dx(u1)*dx(v1) + dy(u1)*dy(v1)
 + dx(u2)*dx(v2) + dy(u2)*dy(v2) )
+ q * (dx(u1) + dy(u2))
+ gammafun*(dx(u1)*dx(v1) + dx(u1)*dy(v2) + dy(u2)*dy(v2) + dy(u2)*dx(v1))
+ p*q*(0.00000001)
// + 0.5*(v1 * (w1 * dx(u1) + w2 * dy(u1)) + v2 * (w1 * dx(u2) + w2 * dy(u2)))
+ 0.25*(v1 * (w1 * dx(u1) + w2 * dy(u1)) + v2 * (w1 * dx(u2) + w2 * dy(u2)))
- 0.25*(u1 * (w1 * dx(v1) + w2 * dy(v1)) + u2 * (w1 * dx(v2) + w2 * dy(v2)))


    )
    + int2d(Th)(
-1.0 / dt * (un1 * v1 + un2*v2)
  + 0.5 * nu * ( dx(un1)*dx(v1) + dy(un1)*dy(v1) + dx(un2)*dx(v2) + dy(un2)*dy(v2) )
// + 0.5*(v1 * (w1 * dx(un1) + w2 * dy(un1)) + v2 * (w1 * dx(un2) + w2 * dy(un2)))
+ 0.25*(v1 * (w1 * dx(un1) + w2 * dy(un1)) + v2 * (w1 * dx(un2) + w2 * dy(un2)))
- 0.25*(un1 * (w1 * dx(v1) + w2 * dy(v1)) + un2 * (w1 * dx(v2) + w2 * dy(v2)))
       )
+ on(4,8, u1 = g, u2 = 0)
+ on(1,11,5,111,99, u1=0, u2=0);



problem getPhi1([vv1,vv2,lambda], [phiTest1,phiTest2,q], solver = UMFPACK) =
//get Phi1 such that phi1 = 1 on 99 & 0 on others
int2d(Th)(
dx(vv1)*dx(phiTest1) + dy(vv1)*dy(phiTest1) +
dx(vv2)*dx(phiTest2) + dy(vv2)*dy(phiTest2)
+ lambda * (dx(phiTest1) + dy(phiTest2) )
```

```
+ ( dx(vv1) + dy(vv2) ) * q
+ 0.000001 * lambda * q
)
+ on(99, vv1 = 1, vv2 = 0)  //phi2 = 0 on all borders
+ on(4, 8, 1, 11, 5, 111, vv1 = 0, vv2 = 0);

problem getPhi2([vv1,vv2,lambda], [phiTest1,phiTest2,q], solver = UMFPACK) =
//get Phi1 such that phi1 = 1 on 99 & 0 on others
int2d(Th)(
dx(vv1)*dx(phiTest1) + dy(vv1)*dy(phiTest1)
 + dx(vv2)*dx(phiTest2) + dy(vv2)*dy(phiTest2)
+ lambda * (dx(phiTest1) + dy(phiTest2) )
+ ( dx(vv1) + dy(vv2) ) * q
+ 0.000001 * lambda * q
)
+ on(99, vv1 = 0, vv2 = 1)  //phi2 = 0 on all borders
+ on(4, 8, 1, 11, 5, 111, vv1 = 0, vv2 = 0);



vv1=0;
vv2=0;

getPhi1;
v1drag=vv1;
v2drag=vv2;

getPhi2;
v1lift=vv1;
```

```
v2lift=vv2;



u1=0;

u2=0;

un1=0;

un2=0;

for (iter=1;iter<=nbiter;iter++)

{

cout<< "*****************************************************"<<endl;

cout<< "Timestep = " << iter << " " << u1[].n << " " << p[].n <<endl;

t = dt*iter;

// extrapolate, and set previous=current

w1 = 1.5*u1 - 0.5*un1;

w2 = 1.5*u2 - 0.5*un2;

un1 = u1;

un2 = u2;

cnle;



// calculate lift/drag, errors, save data
```

```
  cdVisc = int2d(Th)(nu * ( dx(u1)*dx(v1drag) + dy(u1)*dy(v1drag)
            +           dx(u2)*dx(v2drag) + dy(u2)*dy(v2drag) ));

  cdConvec = int2d(Th)( u1*dx(u1)*v1drag + u2*dy(u1)*v1drag +
  u1*dx(u2)*v2drag + u2*dy(u2)*v2drag );

  cdPressure = int2d(Th)(p*dx(v1drag) + p*dy(v2drag));

  cd = -20.*(int2d(Th)((u1 - un1)*v1drag/dt + (u2 - un2)*v2drag/dt +
0.5*dx(u1)*dx(v1drag) + 0.5*dy(u2)*dy(v2drag) + dx(u1)*dy(v2drag) )
+cdVisc+cdConvec-cdPressure);


  clVisc = int2d(Th)(nu * ( dx(u1)*dx(v1lift) + dy(u1)*dy(v1lift)
            +           dx(u2)*dx(v2lift) + dy(u2)*dy(v2lift) ));

  clConvec = int2d(Th)( u1*dx(u1)*v1lift + u2*dy(u1)*v1lift
   + u1*dx(u2)*v2lift + u2*dy(u2)*v2lift );

  clPressure = int2d(Th)(p*dx(v1lift) + p*dy(v2lift));

  cl = -20.*(int2d(Th)((u1 - un1)*v1lift/dt + (u2 - un2)*v2lift/dt
+ 0.5*dx(u1)*dx(v1lift) + 0.5*dy(u2)*dy(v2lift) + dx(u1)*dy(v2lift)
)+clVisc+clConvec-clPressure);

  deltap = p(0.15,0.2)-p(0.25,0.2)+ 0.5 * ( dx(u1)(0.15,0.2) - dx(u1)(0.25,0.2) );

  cout << t << " " << cd << " " << cl << " " << deltap <<  " "
```

```
  << int2d(Th) (u1*u1 + u2*u2) << endl;



  if (cd > maxcd){
      maxcd = cd;
      timemaxcd = t;
  }

  if (cl > maxcl) {
      maxcl = cl;
      timemaxcl = t;
  }

  cout<< maxcd << " " <<timemaxcd << " " <<maxcl << " " <<timemaxcl<<endl;



  if (iter==1000 || iter==2000 || iter==3000 || iter==4000
|| iter==5000 || iter==6000 || iter==7000 || iter==8000 ){

ofstream myn("cyl_th_gdfn_gamma0.5_"+iter) ;

        for (int ix = 0; ix < xvec.n ; ix++){
        for (int iy = 0; iy < yvec.n ; iy++){
gamma = gammafun;
if ( (xvec[ix]-0.2)*(xvec[ix]-0.2)+(yvec[iy]-0.2)*(yvec[iy]-0.2)<0.05*0.05 ){
        uval = 0.0 ;
        vval = 0.0 ;
                        pval = 0.0 ;
```

```
divval=0.0;
gammaval = 0.0;
            }
else{
                  uval = u1(xvec[ix],yvec[iy]) ;
                  vval = u2(xvec[ix],yvec[iy]) ;
                 pval = p(xvec[ix],yvec[iy]) ;
divval = dx(u1)(xvec[ix],yvec[iy]) + dy(u2)(xvec[ix],yvec[iy]);
pxval = dx(p)(xvec[ix],yvec[iy]);
pyval = dy(p)(xvec[ix],yvec[iy]);
u1xval = dx(u1)(xvec[ix],yvec[iy]);
u1yval = dy(u1)(xvec[ix],yvec[iy]);
u2xval = dx(u2)(xvec[ix],yvec[iy]);
u2yval = dy(u2)(xvec[ix],yvec[iy]);
gammaval = gamma(xvec[ix],yvec[iy]);
                   }

                  myn << xvec[ix] << "  " << yvec[iy]  << "  " << uval <<
 "  " << vval << "  " << divval << " " << pval << " " <<
 gammaval << " " << pxval << " " << pyval << " " <<
  u1xval << " " << u1yval << " " << u2xval << " " <<
   u2yval << endl;

       }
       }
    }
 }
```