Mandatory assignment 1

Yngve Mardal Moe

March 2021

1 Exercise 1

1.1 Exercise 1a)

See hand written notes

1.2 Exercise 1b)-1d)

Since the equation is independent of y, we can solve it using a 1D domain. To solve the equation, I implemented a 1D FEniCS solver using both the Galerkin and streamwise upwind Petrov-Galerkin method using first order Lagrange elements.

For the Petrov-Galerkin method, I used the same variational formulation as for the standard Galerkin-method, but with test functions given by

$$v = v_g + \beta h \frac{\partial v}{\partial x}, \tag{1}$$

where h was the mesh resolution, $\beta = 0.5$ was the stabilisation coefficient and v_g was the standard first order Lagrangian test functions.

To estimate the parameters for the error estimate, I used the Nelder-Mead algorithm [3, 1] from SciPy [5] to minimise the loss function

$$\sum_{h \in \mathcal{H}} f\left(\|u_h - u\|, Ch^p\right)^2, \tag{2}$$

where $\mathcal{H} = \{1/8, 1/16, 1/32, 1/64\}$ is the set of all mesh resolutions and u and u_h are the true and estimated solution to the PDE, respectively. $C = C_\alpha$ and $p = \alpha$ for $\|\cdot\| = \|\cdot\|_1$ and $C = C_\beta$ and $p = \beta$ for $\|\cdot\| = \|\cdot\|_0$. f was a function used to penalise under-estimates of the error more severely than over-estimates, and were given by

$$f(a, b) = \begin{cases} a - b & a < b \\ \rho(a - b) & otherwise \end{cases}, \tag{3}$$

ρ was tuned manually to ensure that the error estimate was an upper bound for the true error, and was set to 10^8. The initial guess for the Nelder-Mead

Table 1: Estimated parameters

Scheme	$\boldsymbol{\mu}$	C_α	α	C_β	β
Galerkin	0.1	6.13	0.99	1.48	1.98
	0.3	1.88	1.09	0.30	2.00
	1.0	1.11	1.31	0.09	2.00
Petrov-Galerkin	0.1	5.94	0.87	1.26	1.12
	0.3	1.82	0.99	0.88	1.24
	1.0	0.90	1.24	0.32	1.48

algorithm was $C = 5$ and $p = 1$. The various estimated parameters are shown in Table 1.

From the error estimates in Table 1, we see that the traditional Galerkin approach has better convergence properties than the Petrov-Galerkin method. However, from the convergence plots in Figure 1, we see that the Petrov-Galerkin method shows promise for rough meshes.

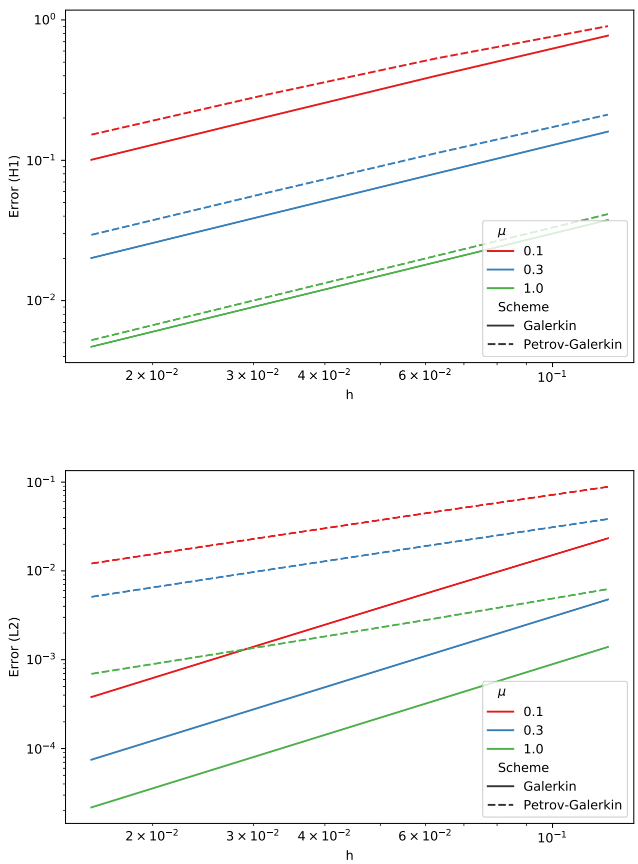

Figure 1: Error norm as a function of mesh resolution. Top: H1 norm, bottom: L2 norm

2 Exercise 2

2.1 Exercise 2a)

For both the explicit and the implicit scheme, based on the incremental pressure correction method described in [2]. The solver was made for benchmark task 2.2 b) in [4]. The code is available on the GitHub repository for the assignment.

To obtain the initial conditions for the solver, I solved Stokes equation with the same parameters as for the Navier-Stokes equation. By default, it uses the standard P2-P1 Taylor-Hood elements for the velocity-field and pressure, respectively, but with an option to select first order elements for both velocity and pressure. By using P1-P1 elements, we get NaN-values in the pressure field, illustrating the superiority of Taylor-Hood elements compared to all linear elements.

2.2 Exercise 2b)

I did not find a good way to solve this problem. However, I start with some arguments towards showing that the stability conditions for the explicit scheme is $C\Delta t \Delta x^{-2}$. We have the equation

$$\dot{u} = -u \cdot \nabla u - \nabla p + \nabla \cdot (\mu \nabla u) + f. \tag{4}$$

Consider a velocity field where ∇p is constant in time, then, we can combine f and ∇p into a single variable, $\omega = f - \nabla p$ and obtain the system

$$\dot{u} = -u \cdot \nabla u + \nabla \cdot (\mu \nabla u) + \omega. \tag{5}$$

Since we know that the stability requirements for

$$\dot{u} = \nabla \cdot (\mu \nabla u) + \omega. \tag{6}$$

solved with an explicit solver is that $\Delta t \leq C \Delta x^2$, and that $u \cdot \nabla u$ is a non-linear term that does not have stabilising properties, we can postulate that the same stability criterion is a best-case-scenario for solving the Navier-Stokes equation.

2.3 Exercise 2c)

To compute the drag coefficient, we first need to compute the drag force. To accomplish this, I used FEniCS to solve the integral

$$\int_{\Gamma_c} \left[(pI - \epsilon(u)) \cdot n \right]_x ds, \tag{7}$$

where Γ_c is the cylinder boundary, p is the pressure, $\epsilon(u)$ is the the strain tensor, n is the outwards facing unit normal of the cylinder and $[\cdot]_x$ denotes the x-component of a vector. Moreover, to improve the simulation accuracy, I start by running the simulation for 1.5 s with a low-resolution mesh ($h \approx 1/64$). The result of the final time-step was subsequently used as the initial condition for a simulation with a higher resolution mesh ($h \approx 1/256$) for which the simulation was run for 0.5 s. The maximal drag coefficient was estimated to be 3.64 and the maximal pressure difference was 3.05 Pa.

References

[1] Fuchang Gao and Lixing Han. "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". In: *Computational Optimization and Applications* 51.1 (2012), pp. 259–277.

[2] Hans Petter Langtangen, Kent-Andre Mardal, and Ragnar Winther. "Numerical methods for incompressible viscous flow". In: *Advances in water Resources* 25.8-12 (2002), pp. 1125–1146.

[3] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[4] Michael Schäfer et al. "Benchmark computations of laminar flow around a cylinder". In: *Flow simulation with high-performance computers II*. Springer, 1996, pp. 547–566.

[5] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature methods* 17.3 (2020), pp. 261–272.