Solving the steady Navier-Stokes equations with Finite Element Methods

Eike Köhn and Milan Klöwer

January 7, 2016

Abstract

The present study introduces the Finite Element Methods in theory and describes its implementation for an idealized two dimensional test case. The equations of interest are the Navier-Stokes equations, describing the conservation of momentum for fluid flows. Although the here presented test case is idealized, the Finite Element Method is capable of adapting to complex domains due to its ability to use irregularly triangulated meshes. We solve the steady Navier-Stokes equations for an incompressible fluid with homogeneous density but variable viscosity. Doing so, the model is able to simulate fluid flows up to Reynolds numbers of order 10^2. We use a regular triangulation of a square domain with linear finite elements as well as the *bubble* element for velocity in order to keep the solution stable. The model converges with increasing resolution n_x as $\mathcal{O}(n_x^{-2})$ for pressure and velocity and has a runtime that increases with $\mathcal{O}(n_x^3)$. It is therefore concluded that the Finite Element Method provides a promising framework for solving partial differential equations. Although harder to implement, it comes with several advantages in comparison to the widely used Finite Difference Method and, depending on the problem, should be considered as a serious alternative.

1 Theoretical Background

1.1 The idea of the Finite Element Method

Solving partial differential equations (PDEs) numerically always goes hand in hand with errors due to the required discretization. There exists a variety of methods to solve these mathematical problems and the accuracy of a solution can strongly depend on the chosen method. The best known and widely used scheme in climate science is the Finite Difference Method (FDM) but there are also others such as the Finite Volume Method (FVM) and the Finite Element Method (FEM). While the FDM is based on differential quotients between individual grid points, the FEM uses the Galerkin ansatz, where we can find the solution function of a PDE, by constructing a linear combination of some given basis functions. Most commonly, these basis functions are continuous, piecewise polynomials, which are referred to as *Finite Elements*. Today, the FEM mostly plays a central role in modern engineering. However, in this report we attempt to solve the fundamental equation in Fluid Dynamics, the Navier-Stokes equation, by using the FEM method.

1.2 Finite Element Methods versus Finite Difference Methods

The choice of the method is a key factor in determining the characteristics of a model. The advantages and disadvantages of FEM and FDM with respect to implementation, performance and analysis are discussed below. The simple idea behind the FDM, based on a linear Taylor Expansion, translates into a generally simple model set up with a more regular grid structure. This set up thus requires simple boundary conditions. Different schemes concerning the temporal evolution and spatial connection influence the stability and accuracy of the solutions. However, the convergence and stability analysis of a model using the FDM is rather limited. In some cases, a solution of a PDE could be strongly improved if a fraction of the domain was resolved to a higher degree. This required refinement of the grid is not common using FDM as irregular grids are difficult to deal with. However, this flexibility and high adaptivity is one of the main strengths of the FEM. Grids can be easily refined. Also, the error and stability analysis is easier to perform in the FEM case. These advantages however come at the price of generally slower models which are more difficult to implement.

For each individual application, a choice between both methods has to be made. In this case, where we solve the Navier-Stokes equations on a regular grid with *simple* boundaries, the FDM seems to be the better choice. However, for educational purposes we solve this test case using FEM.

1.3 Discretization - Translating a partial differential equation into a linear equation system

Let u be the solution function of a partial differential equation. The Galerkin ansatz approximates this solution by a linear combination of some basis functions ϕ_i, which span the function space in which we seek the solution u. Thus

$$u = \sum_j u_j \phi_j \tag{1}$$

For the following partial differential equation $\mathcal{P}u = f$, where $\mathcal{P}$ is a linear differential operator and forcing f is a function in $\Omega \subseteq \mathbb{R}^n$, every solution in u also fulfills

$$\langle \mathcal{P}u, \phi \rangle = \langle f, \phi \rangle, \tag{2}$$

with appropriate $\phi's$. This so called *variational* or *weak* formulation of the partial differential equation allows a transformation into a linear equation system, which will be briefly demonstrated in the example case of the Poisson problem. There, the linear differential operator is the negative Laplacian, i.e.

$$-\Delta u = f \tag{3}$$

Testing this equation with the basis functions, we obtain

$$\langle -\Delta u, \phi_i \rangle = \sum_j u_j \langle -\Delta \phi_j, \phi_i \rangle = \langle f, \phi_i \rangle. \tag{4}$$

The introduction of the *stiffness* matrix A with $A_{ij} = \langle -\Delta \phi_i, \phi_j \rangle$ and the forcing vector $\mathbf{f}_i = \langle f, \phi_i \rangle$ eventually yields the linear equation system $A\mathbf{u} = \mathbf{f}$, which can be solved numerically. To demonstrate this conversion, we solve the 1D-Poisson problem on the unit interval (0,1) with homogeneous boundary conditions. We divide the interval in n equally long subintervals. As finite elements, we choose the *hat* function

$$\phi_k(x) = \mathbf{1}_{I_k}(x) \frac{x - x_{k-1}}{x_k - x_{k-1}} + \mathbf{1}_{I_{k+1}}(x) \frac{x_{k+1} - x}{x_{k+1} - x_k}, \tag{5}$$

where $k = 1, ..., n$ and

$$\mathbf{1}_I(x) = \begin{cases} 1 & \text{if } x \in I \\ 0 & \text{if } x \notin I. \end{cases} \tag{6}$$

The hat function has a small support. Using integration by parts, the variational formulation of the PDE can be transformed to

$$\sum_j u_j \langle -\Delta \phi_j, \phi_i \rangle = \sum_j u_j \langle -\nabla \phi_j, \nabla \phi_i \rangle = \langle f, \phi_i \rangle. \tag{7}$$

Thus the entries of the stiffness matrix can be computed through

$$A_{ij} = \int_0^1 \frac{\partial \phi_i(x)}{\partial x} \frac{\partial \phi_j(x)}{\partial x} dx \tag{8}$$

Due to the equidistant subintervals with length h the resulting matrix has the following symmetric shape.

$$A_{ij} = \frac{1}{h} \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \tag{9}$$

As the right hand side can almost never be computed analytically, the forcing f is approximated by a function of the ansatz space, again through linear combination.

$$f \approx \sum_j f_j \phi_j \tag{10}$$

Thus, for the right hand side it follows

$$\langle f, \phi_i \rangle = \sum_j f_j \langle \phi_j, \phi_k \rangle = M \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} \tag{11}$$

and analog ous to the stiffness matrix a *mass* matrix M can be established, such that

$$M_{ij} = \int_\Omega \phi_i \phi_j d\mathbf{x} \tag{12}$$

In the 1D-Poisson problem we can approximate the result by using the simpler trapezoidal rule such that $f_k = \langle f, \phi_k \rangle \approx h f(x_k)$ in this equidistant case. However, in general, knowing the analytic f requires solving equation 10 to obtain the discrete values of f_j at the nodes in the finite element space (see section 2.5).

Subsequently, the linear equation system $A\mathbf{u} = \mathbf{f}$ can be solved.

1.4 Linear Finite Elements in 2D

In the example above, we only employed linear finite elements to solve the Poisson problem. An option to improve the result is by using finite elements of higher orders. In this study, we only use linear finite elements for simplicity. However, in contrast to the 1D problem from above, we solve the Navier-Stokes Problem in a 2D domain. Similar to the equidistant intervals in the 1D problem, we employ triangulation to identify nodes on which the equation is evaluated. The general choice of finite

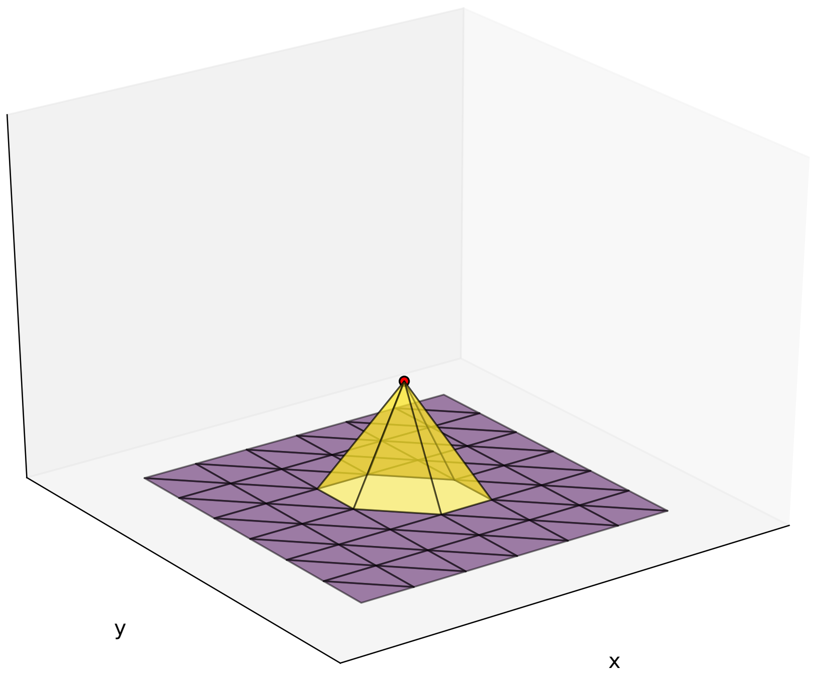

Figure 1: Example for a basis function sitting on the node i with the support of the surrounding triangles.

elements for a triangulated field are polynomials of degree $\leq r$, which follow the formula

$$P_r = \{u(x,y) = \sum_{\substack{0\leq i+j\leq r\\ 0\leq i,j}} a_{ij}x^i y^j\} \tag{13}$$

For linear finite elements ($r = 1$) we obtain $u(x,y) = a_0 + a_1x + a_2y$. The three degrees of freedom can then be linked to the corner nodes of every triangle. To reduce the number of finite elements to only three, we project each individual triangle to a unit triangle with three nodes $(0,0),(0,1)$ and $(1,0)$. The finite elements are then defined as the following three functions on the reference triangle

$$\phi_0^1 = 1 - x - y \tag{14a}$$

$$\phi_0^2 = x \tag{14b}$$

$$\phi_0^3 = y \tag{14c}$$

The support is in fact so small, that the finite element shows the value one on its associated node, while it vanishes on the other two nodes. The projection function from the reference triangle to an arbitrary triangle on the domain with nodes $(p_1,p_2),(q_1,q_2)$ and (r_1,r_2) is given by

$$\Phi_T(\mathbf{x}) = P\mathbf{x} + \mathbf{p}, \tag{15}$$

with

$$P = \begin{pmatrix} q_1 - p_1 & r_1 - p_1 \\ q_2 - p_2 & r_2 - p_2 \end{pmatrix}, \quad \mathbf{p} = (p_1, p_2), \quad \mathbf{x} = (x, y).$$

1.5 The Navier-Stokes equations

The Navier-Stokes equations of a stationary, incompressible fluid are the following

$$\begin{aligned} (v \cdot \nabla)v - \nu \Delta v + \nabla p &= f && \text{in } \Omega && (16a) \\ \nabla \cdot v &= 0 && \text{in } \Omega && (16b) \\ v &= 0 && \text{on } \partial\Omega, && (16c) \end{aligned}$$

where v, p and ν are the 2D velocity field, pressure and kinematic viscosity, respectively. Equation 16c states, that the velocity vanishes on the boundary of the chosen domain. In contrast to the simple Stokes problem equation 16a is nonlinear due to the $(v \cdot \nabla)v$ term. However, this set of equations is already a simplified version as we only consider one layer of fluid with constant density.

To transfer this set of equations into a linear equation system, we again use test functions and thus convert the differential problem into a problem of integration. As both, pressure and velocity are unknown , we create two ansatz spaces for the discrete solution. The space V_h is set up by the basis $\{\phi_1, \phi_2, \ldots, \phi_{N_v}\}$ and contains the velocity solution. The pressure solution is contained in Q_h which is spanned by the basis $\{\xi_1, \xi_2, \ldots, \xi_{N_p}\}$. Testing equation 16 then leads to the variational formulation

$$\begin{aligned} \langle (v \cdot \nabla)v, \phi \rangle + \nu \langle \nabla v, \nabla \phi \rangle - \langle p, \nabla \cdot \phi \rangle &= \langle f, \phi \rangle && (17a) \\ \langle \nabla \cdot v, \xi \rangle &= 0 && (17b) \end{aligned}$$

Similar to the Poisson problem, the second and fourth term of equation 17a are transformed using a stiffness A and mass matrix M upon discretization, respectively. The nonlinear term (first term in equation 17a), pressure gradient term (third term in equation 17a) as well as the divergence term in equation 17b are discretized by the use of the nonlinear matrix $N(\mathbf{v})$ and the so called *divergence* matrix B. The resulting linear equation system has the following structure.

$$\begin{aligned} N(\mathbf{v})\mathbf{v} + A\mathbf{v} + B^T\mathbf{p} &= \mathbf{f} && (18a) \\ B\mathbf{v} &= 0, && (18b) \end{aligned}$$

where $A_{ij} = \nu \langle \nabla \phi_i, \nabla \phi_j \rangle$ and $B_{ij} = -\langle \nabla \cdot \phi_j, \xi_i \rangle$. In matrix notation, the linear equation system can be expressed as

$$\begin{pmatrix} A + N(\mathbf{v}) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \qquad (19)$$

The divergence matrix is again computed by solving the integrals on the standard triangle. The computation of the nonlinear matrix is more complicated, as the matrix itself is dependent on the velocity. Here, the theoretical idea behind the nonlinear matrix will be explained briefly.

The nonlinear term $(v \cdot \nabla) \, v$ can be translated into the matrix $N(\mathbf{v})$, using the trilinear form $n(\phi_k, \phi_i, \phi_j) = \langle (\phi_k \cdot \nabla) \, \phi_i, \phi_j \rangle$. The matrix is then set up by

$$N(v)_{ji} = \sum_k v_k n(\phi_k, \phi_i, \phi_j). \tag{20}$$

Thus, for the first velocity component we obtain

$$\begin{aligned} N(v)_{ji} = \sum_T |\det P| \Bigg(& \sum_{k=1}^{N_v} v_k \left(P_{11}^{-1} \int_{\hat{T}} \phi_r^1 \partial_x \phi_s^1 \phi_t^1 + P_{21}^{-1} \int_{\hat{T}} \phi_r^1 \partial_y \phi_s^1 \phi_t^1 \right) \\ & + \sum_{k=N_v+1}^{2N_v} v_k \left(P_{12}^{-1} \int_{\hat{T}} \phi_r^2 \partial_x \phi_s^1 \phi_t^1 + P_{22}^{-1} \int_{\hat{T}} \phi_r^2 \partial_y \phi_s^1 \phi_t^1 \right) \Bigg), \end{aligned} \tag{21}$$

where r, s and t are the corresponding indices for k, i, j on the reference triangle.

The exact setup of all matrices $(A, B, M, N(\mathbf{v}))$ will be explained related to the 2D-Navier-Stokes problem in the implementation section.

1.6 The Use of Bubble Nodes - Stability

In some cases the discretized formulation of the problem might not have a definite solution, even though the partial differential equation has a unique solution. In these cases, where the so called *LBB-condition* (Ladyschenskaja-Babuška-Brezzi) is not fulfilled, the linear equation system is considered to be instable. There are at least two ways to deal with this instability issue, which is also present in the Navier-Stokes problem. One of them is the PSPG-method (Pressure-Stabilization-Petrov-Galerkin method), in which an additional stabilization term is included in equation 18b. However, in this case we follow the other conventional method, in which we make use of an additional finite element, the so called *bubble* element for the velocity discretization. On the standard triangle it is defined as

$$\phi_0^4 = xy(1 - x - y) \tag{22}$$

and has its associated *bubble node* at midpoint $m = (\frac{1}{3}, \frac{1}{3})$ of the reference triangle. Thus, the velocity field is discretized to a higher order than the pressure field, which makes the solution stable. The bubble element is actually cubic and part of P_3 (equation 13). In contrast to the linear finite elements, this element only reaches the value $1/27$ on its associated node, i.e. $\phi_0^4(m) = \frac{1}{27}$. It still reaches the value zero at the corner nodes, however, the corner node finite elements do not vanish at the bubble node.

These issues have to be taken care of for the transformation between the normal space and the finite element space when implementing the Navier-Stokes solver in the following section.

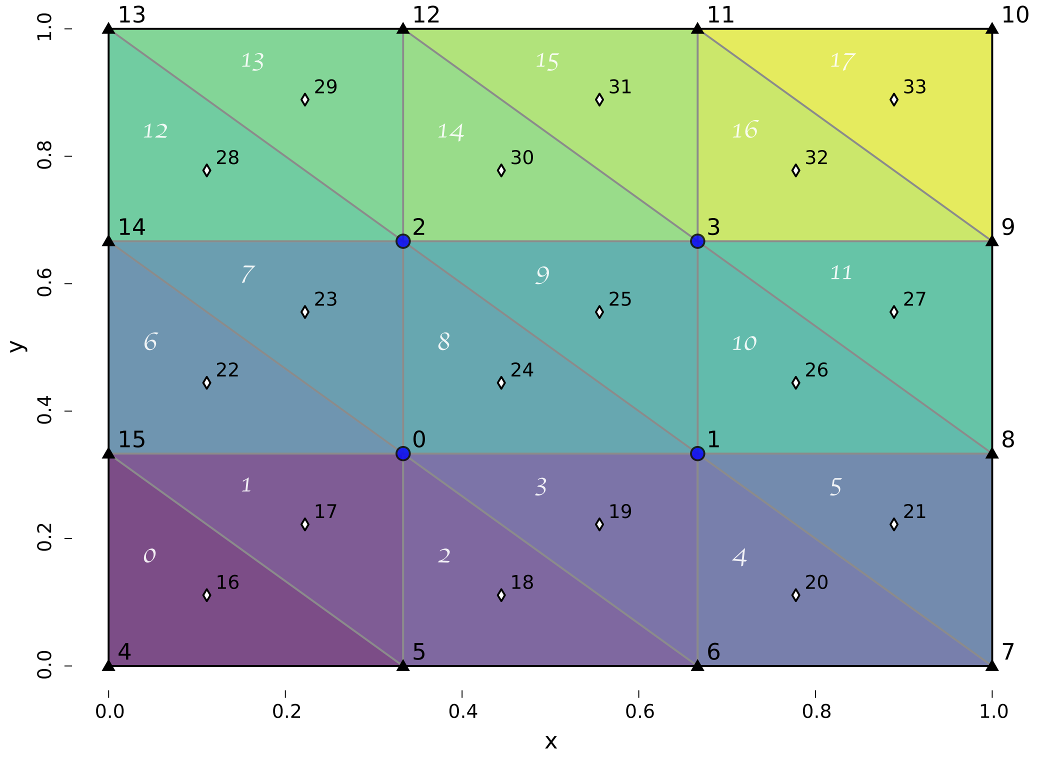

Figure 2: Numbering for $n_x = n_y = 4$ in the domain $\Omega = [0,1] \times [0,1]$. Interior nodes (blue circles) are numbered first, row-wise from bottom left to top right. Boundary nodes (black triangles) are numbered secondly, anti-clockwise starting from bottom left; bubble nodes (white diamonds) are numbered thirdly, row-wise from bottom left to top right. This numbering allows for partitioned matrices.

2 Implementing the Navier-Stokes Solver

The stationary Navier-Stokes problem (introduced in section 1.5) is implemented for the 2D domain $\Omega = [0,1] \times [0,1]$ using the linear finite elements ϕ_0^i, $i = 1,2,3$ (section 1.4) and the bubble function ϕ_0^4 (section 1.6).

2.1 The triangulated grid and data structure

The domain Ω is discretized using a triangulation with regularly arranged right-angled triangles of the same size. The triangulation and the corresponding numbering of nodes and triangles is shown in Figure 2 for an example case in which the number of corner nodes in x and y direction, i.e. the *resolution* n_x and n_y, equals four.

We differentiate between different types of nodes: The *corner* nodes sit on the corner of all triangles. The *bubble* nodes sit in the center of all triangles. The *corner* nodes are further subdivided into *boundary* nodes, which sit on the boundary of the domain and *inner* or *interior* nodes that sit within the domain.

Independent of the resolution, the node numbering always follows the same pattern: *a*) the interior nodes row-wise from bottom left to top right, *b*) the boundary nodes anti-clockwise beginning in the bottom left corner and *c*) the bubble nodes row-wise from bottom left to top right. In contrast, the triangles are just numbered row-wise from bottom left to top right. The used node numbering comes with the advantage of providing partitioned operator matrices as discussed in section 2.3.

The operator matrices are set up by multiplication and integration of overlapping finite elements and their derivatives. Due to the small support of the finite elements, there are only few that overlap. Defining neighbouring nodes as those whose finite elements provide some overlap, finite elements that sit on boundary nodes have 3 to 7 neighbours (2 to 4 corner nodes and 1 to 3 bubble nodes). Interior nodes have always 12 neighbours (6 corner nodes and 6 bubble nodes) and bubble nodes have always 3 neighbours (the adjacent corners of the associated triangle).

Thus, there is a maximum of 12 entries per row or column of the operator matrices and many entries are therefore zero, so that the use of sparse matrices proves to be computationally significantly more efficient.

2.2 Ansatz space

Using the finite elements $\phi_0^i,\ i \in \{1,2,3,4\}$ (as introduced in section 1.4), we search for a solution for the velocity v in the space V_h, spanned by the two-dimensional basis made up of

$$\begin{aligned} \phi_1 &= \begin{pmatrix} \phi_0^1 \\ 0 \end{pmatrix}, \quad \phi_2 = \begin{pmatrix} \phi_0^2 \\ 0 \end{pmatrix}, \quad \phi_3 = \begin{pmatrix} \phi_0^3 \\ 0 \end{pmatrix}, \quad \phi_4 = \begin{pmatrix} \phi_0^4 \\ 0 \end{pmatrix}, \\ \phi_5 &= \begin{pmatrix} 0 \\ \phi_0^1 \end{pmatrix}, \quad \phi_6 = \begin{pmatrix} 0 \\ \phi_0^2 \end{pmatrix}, \quad \phi_7 = \begin{pmatrix} 0 \\ \phi_0^3 \end{pmatrix}, \quad \phi_8 = \begin{pmatrix} 0 \\ \phi_0^4 \end{pmatrix}. \end{aligned} \tag{23}$$

The ansatz space Q_h for the pressure p is spanned by the one-dimensional basis made up of

$$\xi_1 = \phi_0^1, \quad \xi_2 = \phi_0^2, \quad \xi_3 = \phi_0^3, \quad \xi_4 = \phi_0^4. \tag{24}$$

2.3 Stiffness, Divergence and Mass Matrix

The integrals for the different terms in the Navier-Stokes equations (see section 1.5, especially equation 17) are computed using the finite elements $\phi_i,\ \xi_i$ (section 2.2) on the reference triangle. By using the transformation function Φ_T (equation 15) the integral values can be projected onto any triangle in the domain. This way, we obtain the stiffness, divergence and mass matrix, which will be briefly presented below.

The effect of the node numbering is, that the stiffness matrix S is partitioned with the following block structure

$$S = \begin{pmatrix} S_{0,0} & S_{0,b} & S_{0,\beta} \\ S_{b,0} & S_{b,b} & S_{b,\beta} \\ S_{\beta,0} & S_{\beta,b} & S_{\beta,\beta} \end{pmatrix} \tag{25}$$

where the indices 0, b and β refer to inner, boundary and bubble nodes, respectively. The double index implies the interaction between two groups of nodes. Due to the homogeneous boundary conditions in v, the velocity is zero on the boundary and thus the stiffness matrix reduces to

$$S = \begin{pmatrix} S_{0,0} & S_{0,\beta} \\ S_{\beta,0} & S_{\beta,\beta} \end{pmatrix} \tag{26}$$

To apply the stiffness matrix S to both velocity components, we arrange matrix A from the Navier Stokes equation as

$$A = \begin{pmatrix} S & 0 \\ 0 & S \end{pmatrix}. \tag{27}$$

The divergence matrix B also has the block structure with the general appearance

$$B = \begin{pmatrix} B^1_{0,0} & B^1_{0,b} & B^1_{0,\beta} & B^2_{0,0} & B^2_{0,b} & B^2_{0,\beta} \\ B^1_{b,0} & B^1_{b,b} & B^1_{b,\beta} & B^2_{b,0} & B^2_{b,b} & B^2_{b,\beta} \end{pmatrix} \tag{28}$$

The superscripts 1 and 2 denote the association with the first and the second velocity component, respectively. A third row is not existing, as the pressure field is not defined on bubble nodes. Analogously to S, i.e. by taking the homogeneous boundary conditions into account, B can be reduced to

$$B = \begin{pmatrix} B^1_{0,0} & B^1_{0,\beta} & B^2_{0,0} & B^2_{0,\beta} \end{pmatrix} \tag{29}$$

The mass matrix weights the forcing nodes with the size of the the respective triangle and the choice of the finite element within.As the forcing consists of two components the mass matrix M has the partitioned structure

$$M = \begin{pmatrix} M_0 & 0 \\ 0 & M_0 \end{pmatrix} \tag{30}$$

The M_0 block itself is structured as

$$M_0 = \begin{pmatrix} M_{0,0} & M_{b,0} & M_{0,\beta} \\ M_{\beta,0} & M_{\beta,b} & M_{\beta,\beta} \end{pmatrix} \tag{31}$$

M_0 is not quadratic, as the forcing value from the boundary nodes is projected onto the bubble and interior nodes, whereas vice versa they are not, due to the homogeneous boundary conditions in p and v.

2.4 Nonlinear Matrix

The nonlinear matrix $N(v)_{ji}$ depends on the velocity v. Only the integrals in equation 21 are therefore precomputed. As we chose the same one-dimensional basis ϕ_0^i for each of the two components of the two-dimensional basis for v (see equation 23 there are only two different tensors N^1_{kji} and N^2_{kji} with $i, j, k = 1, 2, 3, 4$ to store. The matrix N that results for a given velocity v from equation 21 is again simplified due to the homogeneous boundary conditions as

$$N = \begin{pmatrix} N_{0,0} & N_{0,\beta} \\ N_{\beta,0} & N_{\beta,\beta} \end{pmatrix}. \tag{32}$$

In fact, N can be included in the stiffness matrix A, so that

$$A = \begin{pmatrix} S+N & 0 \\ 0 & S+N \end{pmatrix}. \tag{33}$$

2.5 Projection and Reprojection Matrix

The projection from the (x, y) space, i.e. the *normal* space into the finite element space and back is obtained from the ansatz equation

$$f = \sum_j f_j \phi_j \tag{34}$$

with a projection matrix Q and a reprojection matrix R so that

$$Qf(x_j) = f_j, \qquad Rf_j = f(x_j) \tag{35}$$

As the finite elements sitting on one corner node are zero at all other corner nodes Q, R are identity matrices without bubble nodes. However, as the finite elements on corner nodes are non-zero on bubble nodes their values have to be taken into account. For the midpoint m of the triangle (i.e. the location of the bubble node) we follow from equation 34

$$f(m) = f_1\phi_0^1(m) + f_2\phi_0^2(m) + f_3\phi_0^3(m) + f_4\phi_0^4(m) \tag{36}$$

where $f_i,\ i \in \{1, 2, 3\}$ represent the value of the three corner nodes and f_4 represents the value of the bubble node. For the ϕ_i as given in section 1.4 and 1.6 this is

$$f(m) = \frac{1}{3}(f_1 + f_2 + f_3) + \frac{1}{27} f_4. \tag{37}$$

This linear combination is then written in form of a matrix multiplication with R. Note, that this reprojection could be done onto arbitrary points in the domain Ω. R is therefore not necessarily quadratic.

In the case of transforming the analytic form of the forcing $f(x)$ into the finite element space to obtain the values f_j, we have for all corner nodes x_j

$$f(x_j) = f_j. \tag{38}$$

However, for the bubble node m we have to rearrange equation 36, this time with f_4 as unknown, i.e.

$$f_4 = 27(f(m) - \frac{1}{3}(f_1 + f_2 + f_3)). \tag{39}$$

Again, we can write this as a linear operation in terms of a matrix multiplication with Q.

2.6 Solving the linear equation system

2.6.1 The Uzawa Algorithm (for the linear case)

Ignoring the nonlinear $\mathbf{N}(\mathbf{v})$ term, the linear equation system in equation 19 describes a saddle point problem, where A is a positive definite matrix. To solve such a problem, the *Schur Element* $S = BA^{-1}B^T$ is introduced, which itself is positive definite, if the LBB condition is fulfilled, by multiplying the first equation of the linear equation system by BA^{-1}. The resulting equation $S\mathbf{p} = BA^{-1}\mathbf{f}$ can be used to calculate the pressure $\mathbf{p}$ and subsequently the velocity $\mathbf{v}$ by solving $A\mathbf{v} = \mathbf{f} - B^T\mathbf{p}$. As the computation of the Schur element S is computationally too expensive, the iterative *Uzawa Algorithm* is used instead. This so called preconditioned gradient method requires an initial guess of the pressure and velocity field $(\mathbf{p}^(0), \mathbf{v}^(0))$, updates these over and over again and converges to the solutions for $\mathbf{p}$ and $\mathbf{v}$. The algorithm for this computation is stated below. Along the way, it calculates an optimal step width, in which the solutions are updated. This allows for a relatively fast convergence, compared to algorithms with prescribed and constant step widths. The algorithm stops iterating, once the updates of the solutions have dropped below a certain tolerance level (TOL).

Uzawa Algorithm with optimal step width

```
1: k = 0, choose start pressure $p^{(0)}$
2: Solve $Av^{(0)} = \mathbf{f} - B^T p^{(0)}$
3: $k \leftarrow k + 1$, compute defect $d^{(k)} = Bv^{(k)}$
4: Solve $Me^{(k)} = d^{(k)}$
5: compute $b^{(k)} = B^T e^{(k)}$
6: solve Poisson problem $Aw^{(k)} = -b^{(k)}$
7: compute step witdh $\alpha_k = \frac{\langle d^{(k)}, e^{(k)} \rangle}{\langle -w^{(k)}, b^{(k)} \rangle}$
8: do the corrections $p^{(k)} = p^{(k-1)} + \alpha_k e^{(k)}, v^{(k)} = v^{(k-1)} + \alpha_k w^{(k)}$
9: If $\alpha_k ||e^{(k)}|| \geq$ TOL, goto 3.
```

However, the Uzawa algorithm only solves the linear problem. Thus, we need to implement a scheme that also accounts for the nonlinear terms.

2.6.2 (Quasi-) Newton scheme

The entire nonlinear stationary Navier-Stokes problem can be written as

$$J(\mathbf{v},\mathbf{p}) = \begin{pmatrix} A\mathbf{v} + N(\mathbf{v})\mathbf{v} + B^T\mathbf{p} - \mathbf{f} \\ B\mathbf{v} \end{pmatrix} = 0. \tag{40}$$

The solution $x^* = (\mathbf{v}^*, \mathbf{p}^*)$ can then be approximated by a linear expansion following Taylor's theorem around an original guess x^n. Thus, we can iteratively update the solution by solving the equation

$$x^{n+1} = x^n - J'(x^n)^{-1}(J(x^n)), \tag{41}$$

which brings us closer and closer to x^*, given that the original guess is close enough to the solution. However, as inverting the matrix $J'(x^n)$ is again, computationally too expensive, we will pursue a different scheme where we at first explicitly solve the equation $J'(x^n)d^n = J(x^n)$ and subsequently update the solution $x^{n+1} = x^n - d^n$ with a defect d^n. The derivative of J is given by

$$J'(\mathbf{v},\mathbf{p})(\mathbf{w},\mathbf{q}) = \begin{pmatrix} A\mathbf{w} + N(\mathbf{w})\mathbf{v} + N(\mathbf{v})\mathbf{w} + B^T\mathbf{q} - f \\ B\mathbf{w} \end{pmatrix}. \tag{42}$$

Ignoring the unstable term $N(\mathbf{w})\mathbf{v}$ we end up with the linear equation system

$$A\mathbf{d}^n + N(\mathbf{v}^n)\mathbf{d}^n + B^T\mathbf{e}^n = A\mathbf{v}^n + N(\mathbf{v}^n)\mathbf{v}^n + B^T\mathbf{p}^n - \mathbf{f} \tag{43}$$

$$B\mathbf{d}^n = B\mathbf{v}^n, \tag{44}$$

where $\mathbf{d}^n$ and $\mathbf{e}^n$ represent the defects by which the velocity and pressure fields are updated: $\mathbf{v}^{n+1} = \mathbf{v}^n - \mathbf{d}^n$ and $\mathbf{p}^{n+1} = \mathbf{p}^n - \mathbf{e}^n$. As the linear equation system has a saddle point structure, one can use the Uzawa algorithm within each Newton iteration to solve for the defects $\mathbf{d}^n$ and $\mathbf{e}^n$. Again, if the defects drop below a chosen level of tolerance, the Newton scheme is stopped.

3 Testing the code

3.1 Test functions

To analyse the numerical solution of the finite element solver, we construct the forcing f from the following pressure p and velocity $v = (v_1, v_2)$ that fulfill equations 16b and 16c

$$v_1 = \sin^2(\pi x)\sin(\pi y)\cos(\pi y) \tag{45a}$$

$$v_2 = -\sin^2(\pi y)\sin(\pi x)\cos(\pi x) \tag{45b}$$

$$p = xy(1 - x - y). \tag{45c}$$

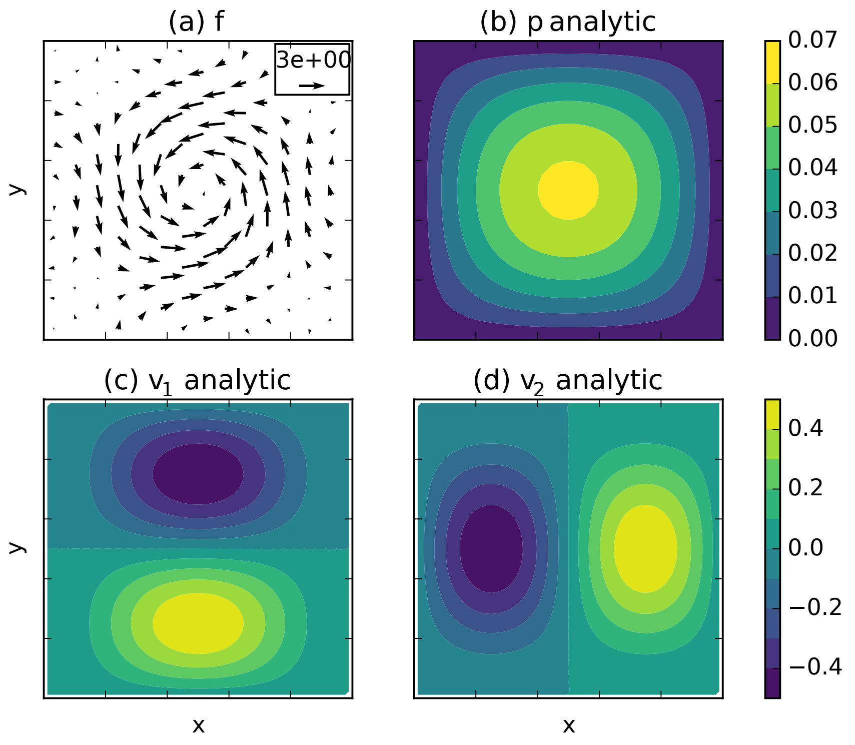

Figure 3: Analytic solution for pressure p and velocity $v = (v_1, v_2)$ of the steady Navier-Stokes equations with forcing f and viscosity $\nu = 1\text{e-}1$ (see equations 45 and 46). The displayed domain is $x = y = [0, 1]$ in all subplots. The quiverkey for (a) is given in the upper right corner.

Equation 16a then yields the forcing $f = (f_1, f_2)$ as

$$
\begin{aligned}
f_1 = {} & (1-2x)y(1-y) \\
& - \nu 2\pi^2 \left(\cos^2(\pi x) - 3\sin^2(\pi x)\right)\sin(\pi y)\cos(\pi y) \\
& + \pi \sin^3(\pi x)\sin^2(\pi y)\cos(\pi x) \qquad (46a) \\
f_2 = {} & (1-2y)x(1-x) \\
& + \nu 2\pi^2 \left(\cos^2(\pi y) - 3\sin^2(\pi y)\right)\sin(\pi x)\cos(\pi x) \\
& + \pi \sin^3(\pi y)\sin^2(\pi x)\cos(\pi y), \qquad (46b)
\end{aligned}
$$

where each component consists of a term resulting from the pressure gradient (first line), one from diffusion (second line) and one from advection (third line). Hence, we have an analytic solution (equation 45) for the steady Navier-Stokes equations (equation 16) if forced with f (equation 46).

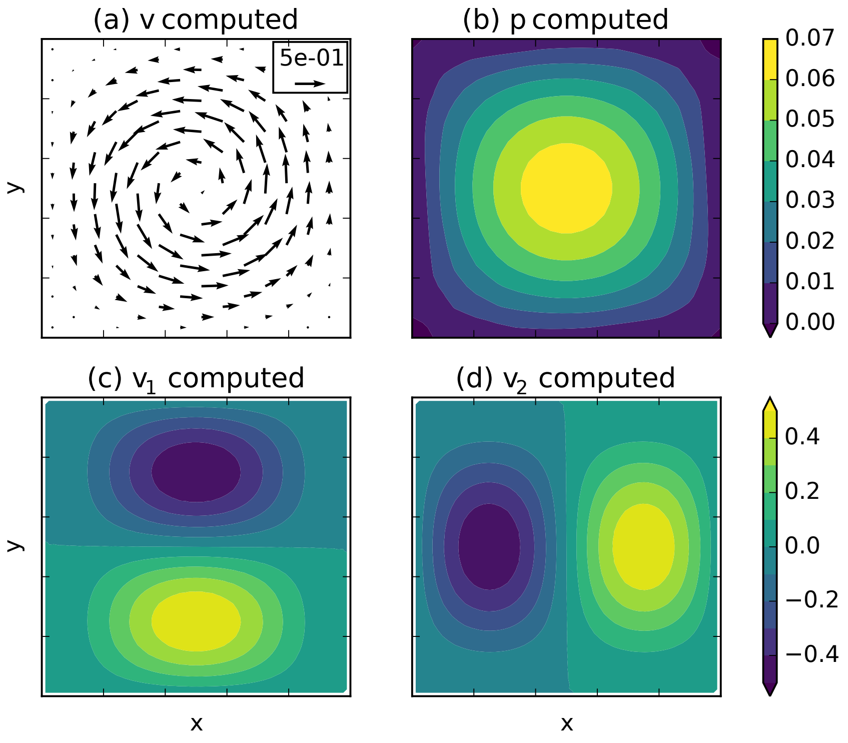

Figure 4: Same as Fig. 3 but for the numerical solution with finite elements using $n_x = n_y = 30$.

3.2 Results

The numerical solution of the steady Navier-Stokes equations is obtained from the finite element solver with a grid resolution of $n_x = n_y = 30$ and viscosity $\nu =$1e-1 (Fig. 4). Tolerances τ for the Uzawa and Quasi Newton algorithms are set to $\tau_{\text{uz}} =$ 1e-5 and $\tau_{\text{qn}} =$ 1e-3. The numerical solution and compares well to the analytic solution (Fig. 3). The applied forcing f is mainly the superposition of an anti-clockwise rotating whirl and a convergence in the center of the domain. The pressure p is larger in the center of the domain and vanishes at the boundary, the pressure gradient force therefore counteracts the convergence of the forcing. The velocity v follows the whirl of f producing an anti-clockwise rotating eddy.

In order to analyse the error we define the absolute error ε_{abs} and the relative error ε_{rel} of a variable ζ and its numerical approximation $\tilde{\zeta}$ as

$$\varepsilon_{\text{abs}} = \tilde{\zeta} - \zeta, \qquad \varepsilon_{\text{rel}} = \frac{\tilde{\zeta} - \zeta}{\zeta}. \tag{47}$$

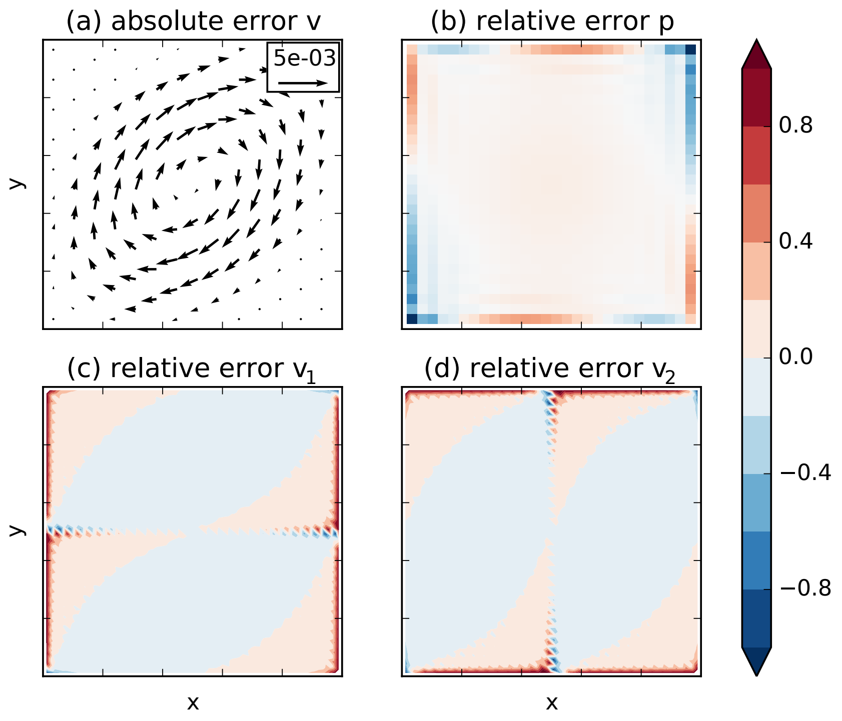

Figure 5: The error between analytic solution (Fig. 3) and the numerical solution (Fig. 4) with finite elements. Absolute and relative error are defined as in section 3.2. Boundary nodes are omitted for p, as $p = 0$ at the boundary for the analytic solution. Black arrows in (a) correspond to interior nodes, red arrows to bubble nodes.

Accordingly, we define a mean relative error $\bar{\varepsilon}_{\text{rel}}$ by summing over all gridpoints $i = 1...n$

$$\bar{\varepsilon}_{\text{rel}} = \frac{\sqrt{\frac{1}{n}\sum_i \left(\tilde{\zeta}_i - \zeta_i\right)^2}}{\sqrt{\frac{1}{n}\sum_i \zeta_i^2}} \tag{48}$$

The pattern of p is in the numerical solution slightly tilted in a anti-clockwise direction as coinciding with the sign of the relative error ε_{rel} close to the boundaries (Fig. 5). The largest relative errors of pressure p are confined to the boundaries, and reach further into the interior for larger viscosity ν (not shown). The absolute error of velocity v reveals that the eddy is spinning slower in the numerical solution compared to the analytic. The relative error of v_1, v_2 is larger for bubble nodes (Fig. 6b) and mainly confined to the boundaries and at $x = 0.5$ as well as $y = 0.5$. This follows as these are the regions where the analytic solution vanishes.

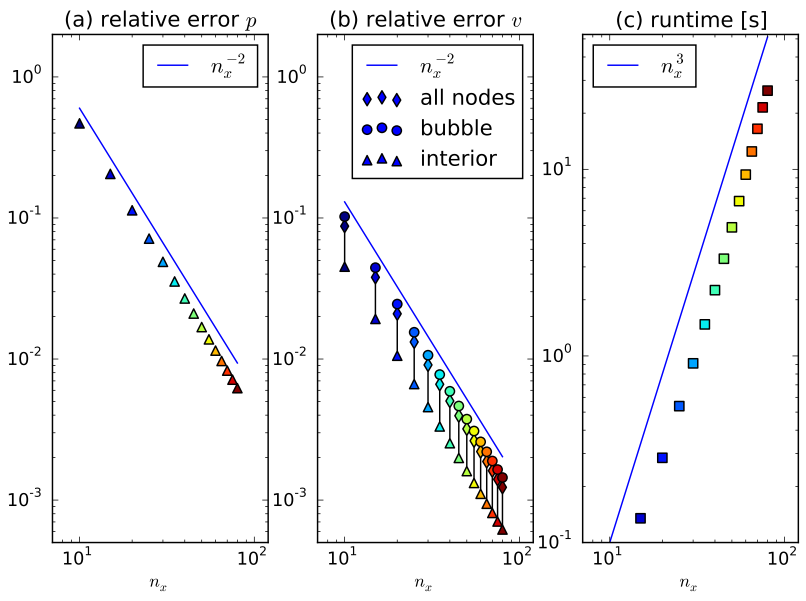

Figure 6: Error and runtime dependence on the resolution n_x. Solid lines represent (a) n_x^{-2} power law; (b) n_x^{-2} power law; and (c) n_x^3 power law. The power laws are chosen so that they resemble the slope of the data points. Symbols are coloured according to n_x.

3.3 Dependency of the error and of runtime on the grid resolution

In order to investigate how the error and the runtime changes with changing grid resolution we use

$$n_x \in \{10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80\}$$

and $n_y = n_x$. The mean relative error $\bar{\varepsilon}_{\text{rel}}$ for p decreases like $\mathcal{O}(n_x^{-2})$. The mean relative error $\bar{\varepsilon}_{\text{rel}}$ for v shows a similar behaviour (Fig. 6b). However, the error is larger for bubble nodes than for interior nodes. The runtime is observed to behave like $\mathcal{O}(n_x^3)$ (Fig. 6c), which is due to an increased runtime of the Uzawa algorithm (Fig. 7). The Uzawa iterations increase linearly with n_x. However, there is no increase in the Quasi Newton iterations, which is always 4.

The error decay of $\mathcal{O}(n_x^{-2})$ is expected in a result from linear finite elements that were used. Since the Uzawa iterations increase linearly with n_x, and the runtime of each iteration is based on solving a linear equation system, which in turn has a runtime of $\mathcal{O}(n_x^2)$, due to sparse matrices, the overall runtime is expected to increase like $\mathcal{O}(n_x^3)$. However, we only reach this behaviour using an LU-decomposition. The advantage hereby is, that the computationally expensive calculation (i.e. the decomposition itself) is only done once per each Quasi Newton step. Solving the

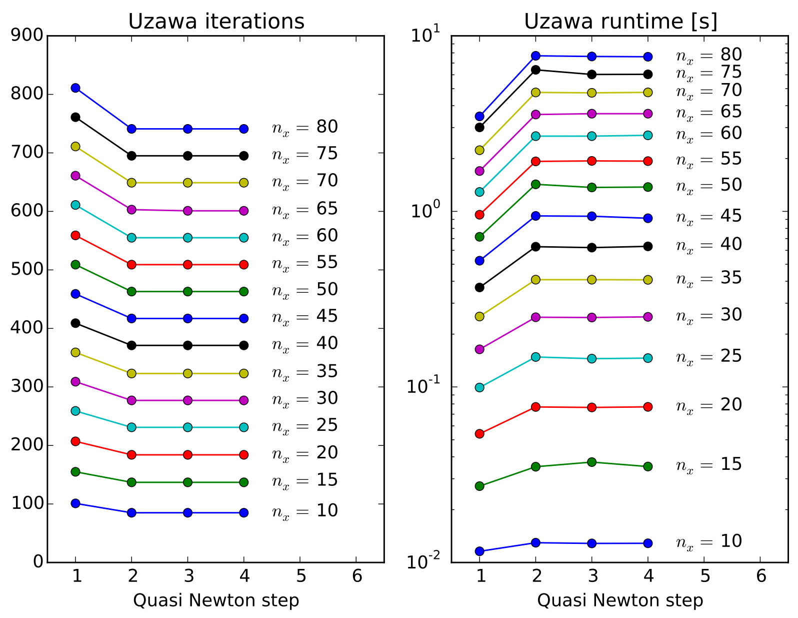

Figure 7: Uzawa iterations and runtime for each Quasi Newton step for different resolutions n_x. For all resolutions the Quasi Newton scheme converges after 4 iterations. The viscosity is for all resolutions kept fixed at $\nu = 1\text{e-}1$.

linear equation system is then a trivial operation only involving substitution.

3.4 Dependency on Reynolds Number

In order to investigate how the error and the runtime changes with viscosity ν we use

$$\begin{aligned}\nu \in \ &\{1, .9, .8, .7, .6, .5, .4, .3, .2, .1,\\ &.09, .08, .07, .06, .05, .04, .03, .02, .01,\\ &.009, .008, .007, .006, .005\}. \end{aligned} \tag{49}$$

and keep the resolution $n_x = 30$ to be constant.

For different ranges of ν we observe qualitatively different behaviour. At least two regimes can be identified. The Reynolds number is estimated as $\text{Re} = \frac{UL}{\nu} = \frac{1}{2\nu}$ with $L = 1$ and $U = 0.5$. For $\nu > 0.1$, which implies $\text{Re} < 5$ this is the *small* Reynolds number case and for $\nu \leq 0.1$ this is the *large* Reynolds number case, with $5 \leq \text{Re} \leq 100$. Note, however, that *large* is meant in a relative sense. In general, some turbulent flows can reach Reynolds numbers larger than 10^5 which is beyond the scope of this study.

For *small* Reynolds numbers the relative error in p increases linearly with increasing ν (Fig. 8a), whereas the relative error in v is constant (Fig. 8b). For

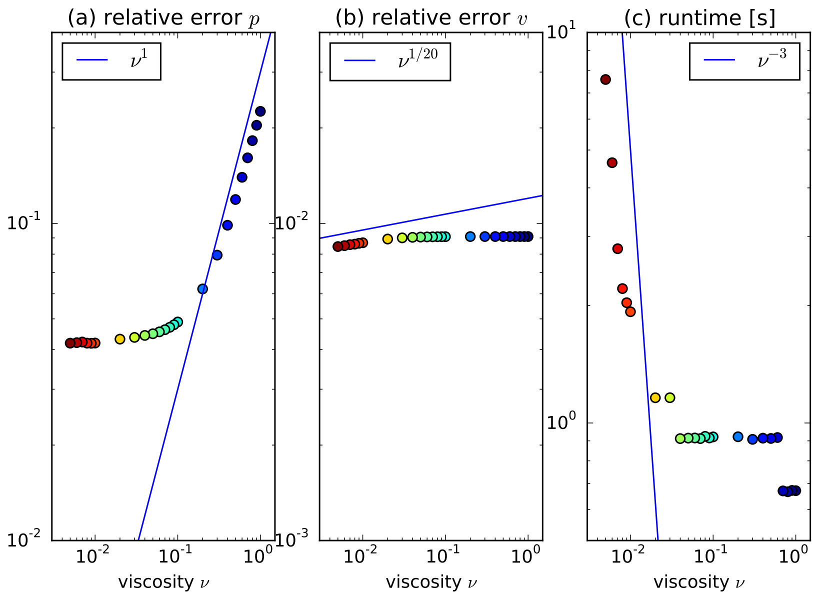

Figure 8: Error and runtime dependence on the viscosity ν. Solid lines represent (a) ν^1 power law, i.e. linear; (b) $\nu^{1/20}$ power law; and (c) ν^{-3} power law. The power laws are chosen so that they resemble the slope of data points in certain ranges of ν. Circles are coloured according to ν.

large Reynolds numbers, however, this behaviour changes: the relative error in p approaches some limit around 4% and is rather constant compared to the linear dependence on ν in the *small* Reynolds number case. On the contrary, the relative error in v is decreasing with increasing Re with very weak power law (on the order of 1/20), which is still comparably constant. The runtime is fairly constant for *small* Reynolds numbers with the model taking less than a second to converge, although showing some step function-like behaviour (Fig. 8c). For *large* Reynolds numbers the runtime increases with a power law of $\mathcal{O}(\mathrm{Re}^3)$.

Analyzing the number of iterations and runtime for each Uzawa algorithm (Fig. 9) reveals that this increase in runtime has two reasons: The number of Uzawa iterations increases drastically for the largest considered Reynolds numbers so that also the runtime for each of the Uzawa convergences increases. The runtime of each Uzawa algorithm is observed to be linearly dependent on the number of iterations, as in the previous subsection for changing resolution. Additionally, the runtime for the largest considered Reynolds numbers also increases as the number of Quasi Newton steps is increasing from previously only 4 up to 8.

Hence, we conclude that for changing the resolution n_x changes in errors and runtime statistics show a behaviour that is explainable with a power law. However, changing the Reynolds numbers, which increases the relative importance of the non-

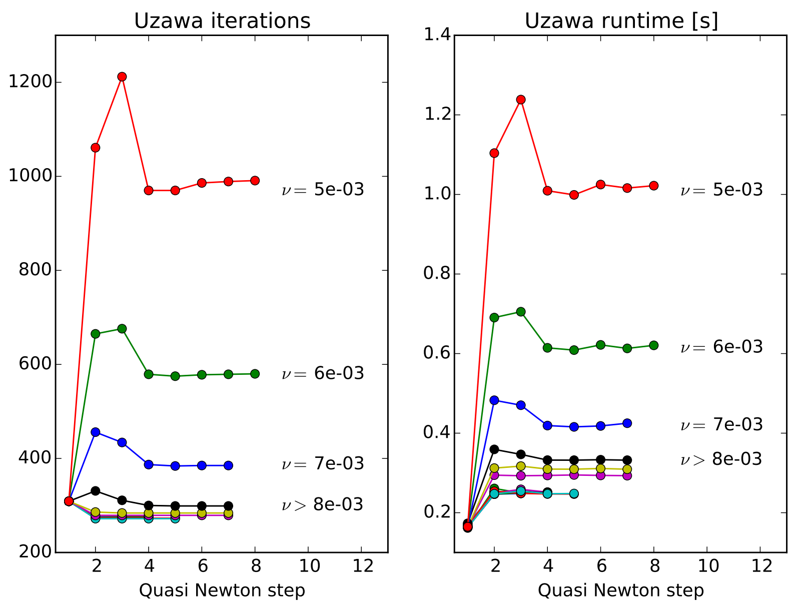

Figure 9: Uzawa iterations and runtime for each Quasi Newton step for different viscosities ν. The resolution was kept fixed at $n_x = 30$.

linear terms causes some behaviour that is far more complicated and can only be described regime-wise, if at all, with power laws.

4 Concluding discussions

The Finite Element Method presented in this study is tested solving the steady Navier-Stokes equations in a rectangular two dimensional domain with a regular triangulation. The linear finite elements are chosen as basis functions as well as the bubble element for velocity in order to not violate the LBB-condition. The model is observed to have an error convergence that behaves like $\mathcal{O}(n_x^{-2})$ with changing resolution which agrees with the theory due to the use of linear finite elements. The runtime increases as a function of n_x^3, a behaviour that is only reached with sparse matrices and LU-decomposition. Changing the viscosity of the fluid, allows to study the model's behaviour in regimes with Reynolds numbers up to order 10^2. Convergence and runtime analysis reveals that the increasing nonlinearity of the system also projects onto error, runtime and iteration, which are only regime-wise explainable with power laws. Although the present study was more for the sake of educational purposes, it is concluded that the Finite Element Method provides a promising framework in order to solve the Navier-Stokes equations numerically in complex domains for the cost of a harder implementation in comparison to the Finite

Difference Method. The present model could be further improved by implementing algorithms for grid refinement, complex boundaries, higher order finite elements and a temporal derivative in the underlying equations.

5 Reference/Acknowledgements

Most information about theory and implementation of the Finite Element Method is taken from Fritz Krügers Lecture

Krüger, F., 2015: *Numerical and Finite Element Methods*, lecture notes.

We hereby acknowledge the lecture notes and refer to these for broad background on Finite Element Methods.