Tutorial 8
Unsteady Incompressible Navier-Stokes Equations

Peter Bastian
Universität Heidelberg
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Im Neuenheimer Feld 368, D-69120 Heidelberg
mail: Peter.Bastian@iwr.uni-heidelberg.de

August 31, 2021

Abstract

This tutorial provides a vanilla implementation of an unsteady incompressible Navier-Stokes solver based on i) conforming Taylor-Hood finite elements, ii) grad-div stabilization to improve satisfaction of the divergence constraint, iii) directional do-nothing condition for improvement of the outflow boundary conditions and iv) Navier-slip condition for modelling rough domains. In addition the transport of a dissolved substance is computed which possibly interacts with the flow.

Contents

1 Incompressible Navier-Stokes Equations

1.1 Strong Form of the Equations

We consider the incompressible Navier-Stokes equations in a d-dimensional domain Ω and time interval Σ which read in strong form [3]:

$$\partial_t u + \nabla \cdot (uu^T) = \nabla \cdot \mathbb{S} + f \qquad \text{in } \Omega \times \Sigma, \tag{1a}$$

$$\mathbb{S} = \nu \left(\nabla u + (\nabla u)^T \right) - pI, \tag{1b}$$

$$\nabla \cdot u = 0. \tag{1c}$$

Here $u : \Omega \times \Sigma \to \mathbb{R}^d$ is the unknown velocity field, $p : \Omega \times \Sigma \to \mathbb{R}$ is the unknown pressure (rescaled by constant mass density), $\mathbb{S} = 2\nu\mathbb{D}(u) - pI$ is the stress tensor, ν is the kinematic viscosity, $\mathbb{D}(u) = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$ is the symmetric velocity gradient and f are external sources and sinks.

This model is equipped with the following boundary conditions:

$$u = g \qquad \text{on } \Gamma_D, \qquad \text{(Dirichlet)} \tag{2a}$$

$$\mathbb{S}n = \frac{1}{2}(u \cdot n)_- u \qquad \text{on } \Gamma_O, \qquad \text{(directional do-nothing)} \tag{2b}$$

$$u \cdot n = 0,\ n \times [\mathbb{S}n] \times n + \beta u = 0 \qquad \text{on } \Gamma_S. \qquad \text{(Navier slip)} \tag{2c}$$

As usual, n denotes the unit outer normal vector to the domain Ω. The directional do-nothing condition, with

$$(u \cdot n)_- = \begin{cases} 0 & u \cdot n \geq 0, \\ (u \cdot n) & u \cdot n < 0, \end{cases}$$

was introduced in [1] with $\mathbb{S}$ replaced by $\mathbb{T} = \nu\nabla u - pI$ and the form presented here is mentioned in [3]. The Navier slip condition is discussed in [4].

Remark 1. This formulation is based in the symmetric gradient in order to be able to incorporate the Navier slip condition. Moreover, in most formulations of the incompressible Navier-Stokes equations one exploits the divergence constraint in the momentum equation leading to $2\nabla\cdot\mathbb{D}(u) = \nabla\cdot\nabla u + \nabla\cdot(\nabla u)^T = \nabla\cdot\nabla u + \nabla(\nabla\cdot u) = \nabla \cdot \nabla u$ as well as $\nabla \cdot (uu^T) = (\nabla u)u + u(\nabla \cdot u)$. Since $\nabla \cdot u = 0$ does not hold in general in the discrete case we prefer to start from the formulation given here.

1.2 Weak Formulation

We recall the following compact notation for the L_2 inner product

$$(f, g)_{0,\Omega} = \begin{cases} \int_\Omega fg \; dx & f, g \text{ scalar valued} \\ \int_\Omega f \cdot g \; dx & f, g \text{ vector valued} \\ \int_\Omega f : g \; dx & f, g \text{ matrix valued} \end{cases},$$

and the integration by parts formula

$$(\nabla \cdot \mathbb{A}, v)_{0,\Omega} = -(\mathbb{A}, \nabla v)_{0,\Omega} + (\mathbb{A}n, v)_{0,\partial\Omega}$$

for any matrix valued function $\mathbb{A}(x)$ and vector valued function $v(x)$.

Then, for any test function with $v = 0$ on Γ_D we obtain the weak form of the momentum equation

$$\begin{aligned}(\partial_t u + \nabla\cdot(uu^T) - \nabla\cdot\mathbb{S}, v)_{0,\Omega} &= \partial_t(u,v)_{0,\Omega} + (\nabla\cdot(uu^T), v)_{0,\Omega} + (\mathbb{S}, \nabla v)_{0,\Omega} - (\mathbb{S}n, v)_{0,\partial\Omega} \\ &= \partial_t(u,v)_{0,\Omega} + (\nabla\cdot(uu^T), v)_{0,\Omega} + 2\nu(\mathbb{D}(u), \nabla v)_{0,\Omega} - (p, \nabla\cdot v)_{0,\Omega} \\ &\quad - \frac{1}{2}((u\cdot n)_- u, v)_{0,\Gamma_O} + \beta(u,v)_{0,\Gamma_S} \\ &= (f,v)_{0,\Omega}\end{aligned}$$

under the additional constraint $u \cdot n = 0$ on Γ_S. The weak form of the continuity equation reads for any test function q:

$$(\nabla\cdot u, q)_{0,\Omega} = 0.$$

Introducing the forms

$$a(u,v) = 2\nu(\mathbb{D}(u), \nabla v)_{0,\Omega} \tag{3}$$
$$b(u,q) = -(\nabla\cdot u, q)_{0,\Omega}, \tag{4}$$
$$c(u,w,v) = (\nabla\cdot(uw^T), v)_{0,\Omega} = ((\nabla u)w, v)_{0,\Omega} + (u\cdot v, \nabla\cdot w)_{0,\Omega}, \tag{5}$$
$$s(u,v) = \beta(u,v)_{0,\Gamma_S} - \frac{1}{2}((u\cdot n)_- u, v)_{0,\Gamma_O} \tag{6}$$

and the function spaces

$$V = \left\{v \in (H^1(\Omega))^d : v = 0 \text{ on } \Gamma_D \text{ and } v\cdot n = 0 \text{ on } \Gamma_S\right\}, \tag{7}$$
$$U = \left\{u \in (H^1(\Omega))^d : u = u_g + v,\ v \in V,\ u_g = g \text{ on } \Gamma_D,\ u_g \cdot n = 0 \text{ on } \Gamma_S\right\}, \tag{8}$$
$$Q = \left\{q \in L_2(\Omega) :\ (q,1)_{0,\Omega} = 0 \text{ if } \Gamma_D = \partial\Omega\right\}, \tag{9}$$

we obtain the following weak formulation. Find $(u,p) \in U \times Q$ such that

$$\partial_t(u,v)_{0,\Omega} + c(u,u,v) + a(u,v) + b(v,p) + s(u,v) = (f,v)_{0,\Omega} \qquad \forall v \in V, \tag{10}$$
$$b(u,q) = 0 \qquad \forall q \in Q. \tag{11}$$

Remark 2. This weak formulation has the following properties:

- The boundary terms give a positive contribution when inserting $v = u$, i.e. $s(u,u) > 0$ for $u \neq 0$.
- a is a symmetric bilinear form since $(\nabla u)^T : \nabla v = \nabla u : (\nabla v)^T$.

2 Finite Element Formulation

We employ a conforming finite element method here, the so-called Taylor-Hood element. It uses the function spaces

$$V_h = \left\{v \in (\mathbb{P}_{r+1})^d : v = 0 \text{ on } \Gamma_D \text{ and } v\cdot n = 0 \text{ on } \Gamma_S\right\}, \tag{12}$$
$$U_h = \left\{u \in (\mathbb{P}_{r+1})^d : u = u_g + v,\ v \in V_h,\ u_g = g \text{ on } \Gamma_D,\ u_g \cdot n = 0 \text{ on } \Gamma_S\right\}, \tag{13}$$
$$Q_h = \left\{q \in \mathbb{P}_r :\ (q,1)_{0,\Omega} = 0 \text{ if } \Gamma_D = \partial\Omega\right\}, \tag{14}$$

and is based on the following discrete weak formulation. Find $(u_h, p_h) \in U_h \times Q_h$ such that

$$\partial_t(u_h, v)_{0,\Omega} + c(u_h, u_h, v) + a(u_h, v) + b(v, p_h) + s(u_h, v) + \gamma(\nabla \cdot u_h, \nabla \cdot v) = (f, v)_{0,\Omega}, \tag{15}$$

$$b(u_h, q) = 0 \tag{16}$$

for all test functions $(v, q) \in V_h \times Q_h$.

Remark 3.

- Here, the grad-div stabilization was added with a factor $\gamma \geq 0$ in order to improve the pressure robustness of the method, see the review [2] on this topic.
- The slip boundary condition is most simple to implement when n points in the direction of a single coordinate.

Insertion of a basis representation results in a nonlinear algebraic system which is solved with Newton's method and the linearized sub problem is solved with a BiCG-Stab method preconditioned with incomplete factorization.

2.1 Preparations

In order to speed up the computations for the finite element method we aim at employing matrix-matrix computations as much as possible. The general strategy is

- Extract coefficients involved with one element from the global data structure.
- All element-wise sizes of arrays are known at compile-time.
- Compute quantities at quadrature points.
- Compute residual constributions.

We now collect some basic tools and introduce notation for element-wise computations. Then the individual terms of the weak formulation are treated.

Element Transformation

Given an element $T \in \mathcal{T}_h$ the transformation from the reference element to the real element is denoted by $T = \mu_T(\hat{T})$ with the Jacobian inverse transposed $J_T(\hat{x}) = \nabla \mu_T^{-T}(\hat{x})$.

Basis Functions and Gradients

For polynomial degree r introduce the global basis functions

$$\mathbb{P}_{r+1} = \text{span}\{\phi_1, \ldots, \phi_{N_u}\}, \qquad \mathbb{P}_r = \text{span}\{\psi_1, \ldots, \phi_{N_p}\} \tag{17}$$

and then set

$$(\mathbb{P}_{r+1})^d = \text{span}\left\{\Phi_{i,k} = \phi_k e_i, 1 \leq k \leq N_u, 1 \leq i \leq d\right\} \tag{18}$$

with the unit vectors $(e_i)_j = \delta_{ij}$. Using that notation we have

$$u_h(x) = \sum_{l=1}^{N_u} \sum_{j=1}^{d} z_{j,l} \phi_l e_j, \qquad p_h(x) = \sum_{m=1}^{N_p} y_m \psi_m(x) \tag{19}$$

The basis function ϕ_k at a point $T \ni x = \mu_T(\hat{x})$ is generated from a basis function $\hat{\phi}_l$ on the reference element via $\phi_k(\mu_T(\hat{x})) = \hat{\phi}_l(\hat{x})$ and we define the vector of local basis function evaluations

$$\hat{\phi}(\hat{x}) = \left(\hat{\phi}_1(\hat{x}), \ldots, \hat{\phi}_{n_u}(\hat{x})\right)^T, \qquad \hat{\psi}(\hat{x}) = \left(\hat{\psi}_1(\hat{x}), \ldots, \hat{\psi}_{n_p}(\hat{x})\right)^T$$

where $n_u = n_{r+1}$ is the number of basis functions of velocity space on the reference element and $n_p = n_r$ is the number of pressure basis functions on the reference element. Then we can evaluate

$$u_h(x) = \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} \phi_l(x) e_j = \sum_{l=1}^{n_u} \phi_l(x) \begin{pmatrix} z_{1,l} \\ \vdots \\ z_{d,l} \end{pmatrix} = Z_T \hat{\phi}(\hat{x}) \tag{20}$$

where $(Z_T)_{j,l} = z_{j,l}$ is the matrix of coefficients on element T. In a similar way the pressure on an element is evaluated as

$$p_h(x) = \sum_{m=1}^{n_p} y_m \psi_m(x) = Y_T \cdot \hat{\psi}(\hat{x}) \tag{21}$$

where Y_T is the vector of pressure degrees of freedom on element T.

For the gradients we define the matrices (no gradients of pressure basis functions is needed)

$$\hat{G}(\hat{x}) = \left[\hat{\nabla}\hat{\phi}_1, \ldots, \hat{\nabla}\hat{\phi}_{n_u}\right] \qquad \text{and} \qquad G_T(x) = J_T(\hat{x})\hat{G}(\hat{x})$$

and now seek to compute $\nabla u(x)$ for $T \ni x = \mu_T(\hat{x})$:

$$\begin{aligned} \nabla u_h(x) &= \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} \nabla(\phi_l(x) e_j) = \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} e_j (\nabla \phi_l(x))^T \\ &= \sum_{l=1}^{n_u} \left(\sum_{j=1}^{d} z_{j,l} e_j \right) (J_T(\hat{x}) \hat{\nabla} \hat{\phi}_l(\hat{x}))^T = \sum_{l=1}^{n_u} z_{*,l} (J_T(\hat{x}) \hat{\nabla} \hat{\phi}_l(\hat{x}))^T \\ &= Z_T G_T^T(x) = Z_T (\hat{G}(\hat{x}))^T J_T^T(\hat{x}). \end{aligned} \tag{22}$$

Now for the divergence:

$$\nabla \cdot u_h(x) = \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} \nabla \cdot (\phi_l(x) e_j) = \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} \partial_{x_j} \phi_l(x) = \sum_{l=1}^{n_u} \sum_{j=1}^{d} z_{j,l} (G_T(x))_{j,l} = Z_T : G_T(x). \tag{23}$$

Quadrature

The integral of a function $q : T \to \mathbb{R}$ is approximated by a generic quadrature rule

$$\int_T q(x)\,dx = \int_{\hat{T}} q(\mu_T(\hat{x}))\text{det}\nabla\mu_T(\hat{x})\,d\hat{x} = \sum_{\alpha=1}^{M_T} q(\mu_T(\hat{x}_\alpha))\text{det}\nabla\mu_T(\hat{x}_\alpha)\hat{w}_\alpha + \text{error}. \tag{24}$$

The weights are collected in the column vector $w_T \in \mathbb{R}^M$ with $(w_T)_\alpha = \text{det}\nabla\mu_T(\hat{x}_\alpha)\hat{w}_\alpha$.

Boundary integrals are treated in a similar way. Let F be a face of element $T \in \mathcal{T}_h$. F has the reference element $\hat{F}$ and there are two maps $\eta_F : \hat{F} \to F$ and $\xi_{F,T} : \hat{F} \to \hat{T}$ that satisfy $\eta_F(\hat{s}) = \mu_T(\xi_{F,T}(\hat{s}))$. Then the integral of a function $q : T \to \mathbb{R}$ over the face F of T can be computed by

$$\int_F q(s)\,ds = \int_{\hat{F}} q(\mu_T(\xi_{F,T}(\hat{s})))\Delta_{F,T}(\hat{s})\,d\hat{s} = \sum_{\alpha=1}^{M_F} q(\mu_T(\xi_{F,T}(\hat{s}_\alpha)))\Delta_{F,T}(\hat{s}_\alpha)\hat{w}_\alpha + \text{error} \tag{25}$$

with the integration element $\Delta_{F,T}(\hat{s}) = \sqrt{|\text{det}\left((\nabla\eta(\hat{s}))^T\nabla\eta(\hat{s})\right)|}$. We collect the weights in the vector $(w_F)_\alpha = \Delta_{F,T}(\hat{s}_\alpha)\hat{w}_\alpha$.

2.2 Element Integrals

Stress Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals

$$(R_T^{\mathbb{S}})_{i,k} = 2\nu(\mathbb{D}(u), \nabla\Phi_{i,k})_{0,T}$$

for all test functions $\Phi_{i,k}$ with support intersecting T. $R_T^{\mathbb{S}}$ is represented as a $d \times n_u$ matrix.

Using the representation of $\nabla u(x)$ from (22) we obtain

$$S_T(x) = 2\mathbb{D}(u)(x) = \nabla u(x) + (\nabla u(x))^T = Z_T G_T^T(x) + G_T(x) Z_T^T \tag{26}$$

and may compute

$$\begin{aligned}\left(R_T^{\mathbb{S}}\right)_{i,k} &= 2\nu \int_T \mathbb{D}(u) : \nabla\Phi_{i,k}\,dx = \nu \int_T (\nabla u + (\nabla u)^T) : (e_i(\nabla\phi_k)^T)\,dx \\ &= \nu \int_T \nabla u_i \cdot \nabla\phi_k + \partial_i u \cdot \nabla\phi_k\,dx\end{aligned}$$

which gives for a column

$$\left(R_T^{\mathbb{S}}\right)_{*,k} = \nu \int_T \nabla u \cdot \nabla\phi_k + (\nabla u)^T\nabla\phi_k\,dx = 2\nu \int_T \mathbb{D}(u)\nabla\phi_k\,dx,$$

and results in

$$R_T^{\mathbb{S}} = 2\nu \int_T \mathbb{D}(u)G_T(x)\,dx = \nu \int_T S_T(x)G_T(x)\,dx \approx \nu \sum_\alpha S_T(x_\alpha)G_T(x_\alpha)(w_T)_\alpha \tag{27}$$

after inserting quadrature.

Grad-Div Stabilization Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^{\nabla\nabla\cdot})_{i,k} = (\nabla \cdot u, \nabla \cdot \Phi_{i,k})_{0,T}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^C is represented as a $d \times n_u$ matrix. Observe

$$(R_T^{\nabla\nabla\cdot})_{i,k} = (\nabla \cdot u, \Phi_{i,k})_{0,T} = (\nabla \cdot u, \nabla \cdot (\phi_k e_i))_{0,T} = (\nabla \cdot u, \partial_{x_i}\phi_k)_{0,T} = (\nabla \cdot u, (G_T)_{i,k})_{0,T}$$

and therefore

$$R_T^{\nabla\nabla\cdot} = \int_T \nabla \cdot u(x) G_T(x)\, dx \approx \sum_\alpha (Z_T : G_T(\mu_T(\hat{x}_\alpha)) G_T(\mu_T(\hat{x}_\alpha)) (w_T)_\alpha$$

Pressure Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^p)_{i,k} = (p, \nabla \cdot \Phi_{i,k})_{0,T}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^p is represented as a $d \times n_u$ matrix:

$$(R_T^p)_{i,k} = (p, \nabla \cdot \Phi_{i,k})_{0,T} = (p, \partial_{x_i}\phi_k)_{0,T} = (p, (G_T(x))_{i,k})_{0,T}$$

which gives in matrix form:

$$R_T^p = \int_{\hat{T}} p(\mu_T(\hat{x})) G_T(\mu_T(\hat{x})) \det \nabla \mu_T(\hat{x})\, d\hat{x} \approx \sum_\alpha (Y_T \cdot \hat{\psi}(\hat{x}_\alpha)) G_T(\mu_T(\hat{x}_\alpha)) (w_T)_\alpha \tag{28}$$

Convection Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^C)_{i,k} = (\nabla \cdot (uu^T), \Phi_{i,k})_{0,T}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^C is represented as a $d \times n_u$ matrix. Using the product rule $\nabla \cdot (uu^T)(x) = \nabla u(x) u(x) + u(x) \nabla \cdot u(x)$ we can reuse (20), (22), (23):

$$(R_T^C)_{i,k} = \left(\nabla \cdot (uu^T), \Phi_{i,k}\right)_{0,T} = \left(\nabla \cdot (uu^T), \phi_k e_i\right)_{0,T} = \left(\left[\nabla \cdot (uu^T)\right]_i, \phi_k\right)_{0,T}$$

and therefore

$$\begin{aligned} R_T^C &= \int_T \nabla \cdot (uu^T) \hat{\phi}^T(\hat{x}) \det \nabla \mu_T(\hat{x})\, dx \\ &\approx \sum_\alpha \left[\nabla u_h(\mu_T(\hat{x}_\alpha)) \hat{u}_h(\hat{x}_\alpha) + \hat{u}_h(\hat{x}_\alpha) \nabla \cdot u_h(\mu_T(\hat{x}_\alpha))\right] \hat{\phi}^T(\hat{x}_\alpha) (w_T)_\alpha. \end{aligned} \tag{29}$$

Remark 4. The efficient evaluation of this expression should follow the rules:

- Do a right to left evaluation to minimize flop count.
- Vectorize over quadrature points.

Mass Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^M)_{i,k} = (u, \Phi_{i,k})_{0,T}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^M is represented as a $d \times n_u$ matrix:

$$(R_T^M)_{i,k} = (u, \Phi_{i,k})_{0,T} = (u, \phi_k e_i)_{0,T} = (u \cdot e_i, \phi_k)_{0,T}$$

which gives in matrix form

$$R_T^M = \int_{\hat{T}} u(\mu_T(\hat{x}))\hat{\phi}^T(\hat{x}) \det\nabla\mu_T(\hat{x})\, d\hat{x} \approx \sum_\alpha (Z_T\hat{\phi}(\hat{x}_\alpha))\hat{\phi}^T(\hat{x}_\alpha)(w_T)_\alpha \tag{30}$$

External Force Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^f)_{i,k} = (f, \Phi_{i,k})_{0,T}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^f is represented as a $d \times n_u$ matrix:

$$(R_T^f)_{i,k} = (f, \Phi_{i,k})_{0,T} = (f, \phi_k e_i)_{0,T} = (f \cdot e_i, \phi_k)_{0,T}$$

which gives in matrix form

$$R_T^f = \int_{\hat{T}} f(\mu_T(\hat{x}))\hat{\phi}^T(\hat{x}) \det\nabla\mu_T(\hat{x})\, d\hat{x} \approx \sum_\alpha f(\mu_T(\hat{x}_\alpha))\hat{\phi}^T(\hat{x}_\alpha)(w_T)_\alpha \tag{31}$$

Divergence Term

For element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^{\nabla\cdot})_k = (\nabla \cdot u, \psi_k)_{0,T}$ for all pressure test functions ψ_k with support intersecting T. $R_T^{\nabla\cdot}$ is represented as a column vector with n_p components:

$$(R_T^{\nabla\cdot})_k = (\nabla \cdot u, \psi_k)_{0,T} = (Z_T : G_T, \psi_k)_{0,T}$$

which gives

$$R_T^{\nabla\cdot} = \int_{\hat{T}} (Z_T : G_T(\mu_T(\hat{x})))\hat{\psi}(\hat{x}) \det\nabla\mu_T(\hat{x})\, d\hat{x} \approx \sum_\alpha (Z_T : G_T(\mu_T(\hat{x}_\alpha)))\hat{\psi}(\hat{x}_\alpha)(w_T)_\alpha$$

Navier Slip Boundary Term

For boundary face F of element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^S)_{i,k} = \beta(u, \Phi_{i,k})_{0,F}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^S is represented as a $d \times n_u$ matrix:

$$(R_T^S)_{i,k} = \beta(u, \Phi_{i,k})_{0,F} = \beta(u, \phi_k e_i)_{0,F} = \beta(u \cdot e_i, \phi_k)_{0,F},$$

so

$$R_T^S = \beta \int_{\hat{F}} u(\mu_T(\xi_{F,T}(\hat{s})))\hat{\phi}^T(\xi_{F,T}(\hat{s}))\Delta_{F,T}(\hat{s})\, d\hat{s} \approx \sum_\alpha (Z_T\hat{\phi}(\xi_{F,T}(\hat{s}_\alpha)))\hat{\phi}^T(\xi_{F,T}(\hat{s}_\alpha))(w_F)_\alpha. \tag{32}$$

Directional Do-Nothing Boundary Term

For boundary face F of element $T \in \mathcal{T}_h$ we seek to compute the integrals $(R_T^O)_{i,k} = \frac{1}{2}((u \cdot n)_- u, \Phi_{i,k})_{0,F}$ for all test functions $\Phi_{i,k}$ with support intersecting T. R_T^O is represented as a $d \times n_u$ matrix:

$$(R_T^O)_{i,k} = \frac{1}{2}((u \cdot n)_- u, \Phi_{i,k})_{0,F} = \frac{1}{2}((u \cdot n)_-(u \cdot e_i), \phi_k)_{0,F},$$

so

$$\begin{aligned} R_T^O &= \frac{1}{2}\int_{\hat{F}} (u(\mu_T(\xi_{F,T}(\hat{s}))) \cdot n)_- u(\mu_T(\xi_{F,T}(\hat{s})))\hat{\phi}^T(\xi_{F,T}(\hat{s}))\Delta_{F,T}(\hat{s})\, d\hat{s} \\ &\approx \sum_\alpha ((Z_T\hat{\phi}(\xi_{F,T}(\hat{s}_\alpha))) \cdot n)_-(Z_T\hat{\phi}(\xi_{F,T}(\hat{s}_\alpha)))\hat{\phi}^T(\xi_{F,T}(\hat{s}_\alpha))(w_F)_\alpha. \end{aligned} \tag{33}$$

3 Realization in PDELab

The structure of the code is very similar to that of tutorial 00. It was the aim to implement all computations as efficient as possible while still being flexible in the choice of the degree of the basis functions and the quadrature points (this was not done in tutorial 00). There are actually four examples in tutorial 08 which are called

`navier-stokes-`*example*

with

example $\in$ { `driven-cavity`, `driven-cavity-3d`, `cylinder`, `rayleigh-benard`}.

All examples, except the second one, are two-dimensional. There are the following files:

1) For every example there is a `.cc` file setting up initial and boundary conditions for the specific example and calling the driver.

2) For every example there is a corresponding `.ini` file holding parameters read by various parts of the code.

3) File `driver_flow.hh` has a function `driver_flow` which instantiates the necessary PDELab classes for solving the nonlinear instationary problem and then solves the problem and outputs results.

4) File `driver_coupled.hh` contains a function `driver_coupled` that instantiates the necessary PDELab classes for solving the nonlinear instationary problem as well as the transport problem for a dissolved component. The transport problem is coupled to the flow problem via the velocity field while the flow problem may be coupled to the transport problem e.g. through a concentration-dependent density. The transport problem is discretized with a discontinuous Galerkin method implemented in `dune-pdelab` and the coupled problem is solved with first-order operator splitting.

5) File `navier-stokes-lop` contains the class `ConformingNavierStokesLOP` realizing a PDELab local operator implementing the conforming finite element method for a given order and element type. So far, the jacobians are computed via numerical differentiation. Since the run-time of the code is limited by the linear solvers there is not much to gain with analytical jacobians without implementing a faster solver.

6) File `schemes.hh` implements several classes that can be used to parametrize the driver for various orders and element types.

7) File `timecapsule.hh` implements a simple helper class required for time-dependent boundary conditions.

8) Finally, the tutorial provides some mesh files.

3.1 Compilation Control

All examples use the UG grid manager for their computations. In order to cut down on compilation time the executable is compiled for exactly one configuration of dimension, element / mesh type and polynomial degree. The configuration is controlled by three compile time constants:

```
#define STRUCTURED
#define CUBE
#define DEGREE 2
```

When the constant `STRUCTURED` is defined, an equidistant structured mesh is used with parameters set in the inifile (see below). When `CUBE` is defined then quadrilateral or hexahedral meshes are used, otherwise triangles or tetrahedra. Note that the dimension is fixed by the example itself. Only the `driven-cavity-3d` example works in three dimensions, all other examples are two-dimensional. If `STRUCTURED` is not defined a gmsh file is read which may contain a cuboid or a simplicial mesh. In that case be sure to still set (or undef) `CUBE` to chose the correct finite element space. Finally, the constant `DEGREE` selects the polynomial degree used for velocity. Pressure is always one degree less and concentration uses the same degree as velocity. Note that on cube meshes only degree 2 is valid and on simplicial meshes degrees 2 and 3 are supported. New schemes can be defined by the user, see the examples in the file `schemes.hh`. There the number of quadrature points has to be chosen appropriately with respect to the polynomial degree. See table 1 for a correspondance between quadrature order and number of quadrature points.

3.2 Ini-File

The ini-file allows the user to set various run-time parameters for the execution of the program. Here we skim briefly through the sections of the parameters of the coupled flow problem givewn in the file `navier-stokes-rayleigh-benard.ini`.

```
[grid]
extend = 1.0 0.23456
cells = 50 25
```

```
meshfile = unitsquare.msh
refinement = 0

[problem]
viscosity = 1e-3
gamma = 0.5e-3
beta = 0.0
dt = 0.2
T = 20.0
heatconductivity = 1e-3
rho_0 = 12.0
alpha = 10.0

[solver]
lineariterationsmax = 8000
linearsolververbosity = 0
linearreduction = 1e-10

[newton]
reassemble_treshhold = 0.0
verbosity = 2
reduction = 1e-10
min_linear_reduction = 1e-6

[newton.terminate]
max_iterations = 50
absolute_limit = 1e-13

[newton.line_search]
line_search_max_iterations = 40

[output]
basename = rayleigh_benard_Pr1_Ra100000
subsampling = 2
every = 1
```

The `[grid]` section controls the grid that is used for the computations. If the program is compiled for a structured mesh then the keys `extend` and `cells` are relevant. These give the extend of the domain in x and y direction as well as the number of cells in x and y direction. In the case of an unstructured mesh the name of the meshfile is given by the key `meshfile`. Then, `refinement` gives the number of uniform refinement steps and is applicable to both types of meshes.

The `[problem]` section determines the parameters of the PDE and its discrete weak formulation:

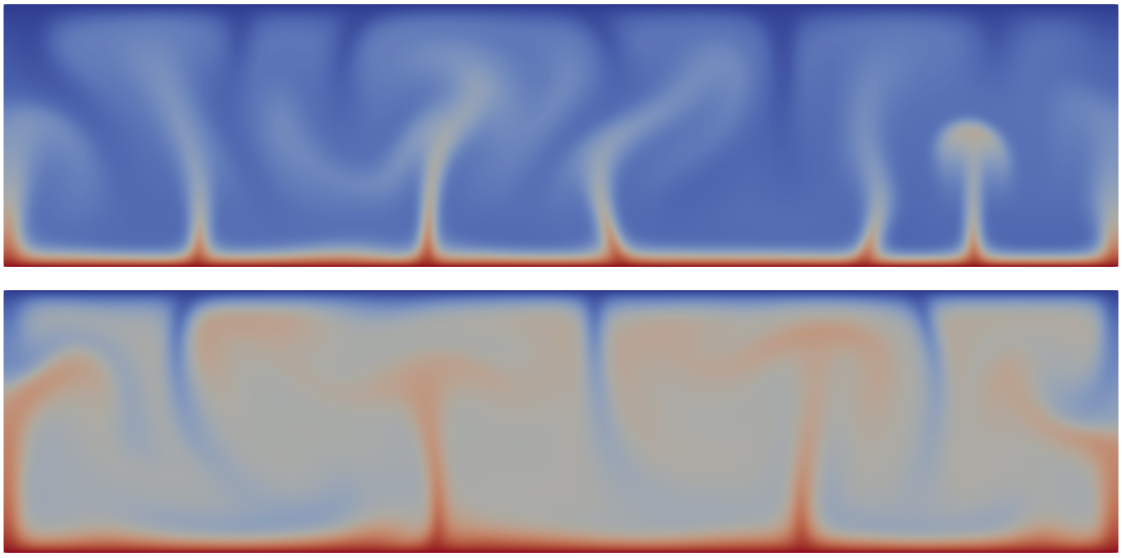

Figure 1: Rayleigh-Bénard flow for $Ra = 4 \cdot 10^7$ and $Pr = 40$. Top image is from the early phase and the bottom image is after establishment of convection rolls. The solution does not become stationary.

`viscosity`	dynamic viscosity ν in Navier-Stokes equations
`gamma`	γ in grad-div stabilization
`beta`	β in Navier slip condition (not used here)
`dt`	time step size
`T`	length of time interval
`heatconductivity`	diffusion parameter κ in transport problem
`rho_0`	density for concentration zero
`alpha`	expansion coefficient

The force density in the Navier-Stokes equation is $f = -\varrho(c)e_z$ where $e_z = (0, 1)^T$ points vertically up and $\varrho(c) = \varrho_0 - \alpha c$ describes the linear dependence of density on concentration (or rather temperature here). For this setup (with a domain of size 1) we have

$$Pr = \frac{\nu}{\kappa}, \qquad Ra = \frac{\alpha}{\nu\kappa},$$

where Pr is the Prandtl number and Ra is the Rayleigh number.

The `[solver]` section sets some parameters controlling the operation of the linear (ILU-preconditioned BiCGStab) solvers.

`lineariterationsmax`	max. BiCGStab iterations allowed
`linearsolververbosity`	amount of output of linear solver
`linearreduction`	relative reduction required for BiCGStab

Paremters for the nonlinear Newton solver can be set in the `[newton]` section of the ini file.

Finally, the `[output]` section controls the VTK output of the simulation by setting the name of the various output files, selecting the subsampling depth mesh refinement during output and restricting output to the timesteps numbers divisible by `every`.

Table 1: Number of quadrature points for a given order (polynomial degree that is integrated exactly) depending on the element type. In the implementation the number of quadrature points needs to be provided at compile-time.

order	line	tri	quad	tet	hex
1	1	1	1	1	1
2	2	3	4	4	8
3	2	4	4	8	8
4	3	6	9	15	27
5	3	7	9	15	27
6	4	12	16	60	64
7	4	12	16	60	64
8	5	16	25	96	125
9	5	19	25	114	125
10	6	25	36	175	216
11	6	28	36	196	216
12	7	33	49	264	343
13	7	56	49	448	343
14	8	64	64	576	512
15	8	72	64	648	512
16	9	81	81	810	729
17	9	90	81	900	729
18	10	100	100	1100	1000
19	10	110	100	1210	1000
20	11	121	121	1452	1331

References

[1] Malte Braack and Piotr Boguslaw Mucha. Directional do-nothing condition for the navier-stokes equations. *Journal of Computational Mathematics*, 32(5):507 – 521, 2014.

[2] V. John, A. Linke, C. Merdon, M. Neilan, and L. Rebholz. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM Review*, 59(3):492–544, 2017.

[3] Volker John. The Navier–Stokes equations as model for incompressible flows. `https://www.wias-berlin.de/people/john/LEHRE/NUM_NSE_14/num_nse_15_1.pdf` download September 10, 2018.

[4] Jiri Neustupa and Penel Patrick. The Navier-Stokes equation with slip boundary conditions. *RIMS Kôkyûroku*, 1536:46 – 57, 2007.