Variational multiscale finite element formulation for computational fluid dynamics

Dr.-Ing. Andreas Apostolatos Dipl.-Ing. Matthew Keller
Dipl.-Ing. Marko Leskovar

May 24, 2020

1 Theory

1.1 Fluid mechanics

Given is a domain Ω with a piecewise continuous boundary $\Gamma = \partial\Omega$ within which a Newtonian incompressible fluid flow is found. The unknown fields are the time dependent velocity and pressure fields denoted as $\mathbf{u}$ and p, respectively. The fluid's density ρ and kinematic viscosity μ are assumed to be constant. The so-called dynamic viscosity of the flow is defined as the ratio of the kinematic viscosity divided by the fluid density $\nu = \mu/\rho$. Moreover, assumed is also that the fluid flow is subject to body forces (e.g. gravitational forces) $\mathbf{f}$. The velocity and/or pressure field are prescribed to a fixed value $\bar{\mathbf{u}}$ and $\bar{p}$, respectively, along a portion of the domain's boundary known as the inlet (Dirichlet) boundary $\Gamma_{\mathrm{d}} \subset \Gamma$. The fluid traction vector along a cut $\gamma \subset \overline{\Omega}$ with outward normal $\mathbf{n}$ is given by,

$$\mathbf{t} = -p\,\mathbf{n} + \mathbf{n}\cdot\nu\,\boldsymbol{\nabla}^{\mathrm{s}}\mathbf{u}\,, \tag{1}$$

where $\boldsymbol{\nabla}^{\mathrm{s}}$ stands for the symmetric gradient operator namely, $\boldsymbol{\nabla}^{\mathrm{s}} = 1/2\,\left(\boldsymbol{\nabla} + \boldsymbol{\nabla}^{\mathrm{t}}\right)$. Moreover p is expressed in Nm/Kg since it is normalized by the fluid density, meaning that the actual pressure is obtained by $\tilde{p} = \rho\,p$. The domain Ω is also assumed to be moving with a given velocity $\mathbf{u}_{\mathrm{d}}$ independently of the fluid flow. Then, accordingly to the *Arbitrary Lagrangian-Eulerian* description of motion, the inertial forces are given by means of the material derivative $D(\bullet) = \dot{(\bullet)} + (\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}(\bullet)$, where $\dot{(\bullet)} = \partial(\bullet)/\partial t$ stands for the time derivative. Accordingly, the strong formulation of the transient incompressible Navier-Stokes problem writes; Find the velocity and pressure field $\mathbf{u} \in \mathbf{C}^2(\Omega)$ and $p \in C^1(\Omega)$, respectively, such that,

$$\dot{\mathbf{u}} - \nu\Delta\mathbf{u} + (\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}p = \mathbf{f} \quad \text{in } \Omega\,, \tag{2a}$$

$$\boldsymbol{\nabla}\cdot\mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2b}$$

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on } \subset \Gamma_{\mathrm{d}}\,, \tag{2c}$$

$$p = \bar{p} \quad \text{on } \subset \Gamma_{\mathrm{d}}\,, \tag{2d}$$

$$\mathbf{t} = \bar{\mathbf{t}} \quad \text{on } \Gamma_{\mathrm{n}}\,, \tag{2e}$$

where $\bar{\mathbf{t}}$ stands for the applied traction vector on the outlet (Neumann) boundary $\Gamma_{\mathrm{n}} \subset \Gamma$. Regarding the symbols in Eq. (2), $\Delta(\bullet) = \partial^2(\bullet)/\partial X_1^2 + \partial^2(\bullet)/\partial X_2^2$ stands for the Laplace second-order operator with respect to $\mathbf{X} = X_\alpha\mathbf{e}_\alpha \in \Omega$ where the Einstein's summation convention over repeated indices is assumed, whenever not otherwise stated, and where Greek and Latin indices range from 1 to 2 and from 1 to 3, respectively.

1.2 Variational multiscale stabilization of the weak formulation

Multiplying Eqs. (2a) and (2b) by test functions $\delta\mathbf{u}$ and δp, respectively, integrating over Ω, performing integration by parts and applying the boundary conditions yields the following variational formulation; Find $\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega)$ and $p \in \boldsymbol{\mathcal{L}}^2(\Omega)$ such that,

$$\langle \delta\mathbf{u}, \dot{\mathbf{u}} \rangle_{0,\Omega} + \langle \nu \boldsymbol{\nabla} \delta\mathbf{u}, \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} - \langle \delta\mathbf{u}, p \rangle_{0,\Omega} - \langle \delta p, \boldsymbol{\nabla} \cdot \mathbf{u} \rangle_{0,\Omega} = \langle \delta\mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, \bar{\mathbf{t}} \rangle_{0,\Gamma_\mathrm{c}} \ , \tag{3}$$

for all $\delta\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega)$ and all $\delta p \in \boldsymbol{\mathcal{L}}^2(\Omega)$, where $\left\langle \mathfrak{T}, \mathfrak{T}' \right\rangle_{0,\Omega} = \int_\Omega \mathfrak{T}_{\alpha_1 \ldots \alpha_m} \mathfrak{T}'_{\alpha_1 \ldots \alpha_m} \, \mathrm{d}\Omega$ stands for the $\boldsymbol{\mathcal{L}}^2$-norm in Ω for any pair of equal order tensors $\mathfrak{T}, \mathfrak{T}' \in \mathfrak{S}^m$. The discrete form of variational formulation in Eq. (3) has a unique solution given that the so-called *Ladyzhenskaya-Babuška-Brezzi* (LBB) or *inf-sup* condition is satisfied by the discrete spaces. It has been shown that for specific pairs of the $\mathbf{u}$-p discretization, such as p1-p0, this is satisfied but for equal order interpolations of the velocity and pressure fields this is not the case. This in turn leads into nearly singular matrices and oscillatory (checkerboard patter) pressure fields.

Herein the so-called *Variational Multi-Scale* (VMS) stabilization [HFMQ98] with the *Algebraic Sub-Grid Scales* (ASGS) projection [Cod01] of variational formulation in Eq. (3) is employed. Accordingly, the unknown fields are comprised by two parts, the coarse or resolvable scale fields and the fine or subscale fields. The former is resolvable within the finite element discretization whereas the latter is not. Therefore, the subscales are modeled by projecting the residuals of the Navier-Stokes Eqs. (2a) and (2b) onto the subscales. Let $\tilde{\boldsymbol{\mathcal{L}}} : \mathbf{C}^2(\Omega) \times C^1(\Omega) \to \mathbb{R}^3$ be the differential operator of the Navier-Stokes Eqs.(2a), (2b), namely,

$$\tilde{\boldsymbol{\mathcal{L}}} \left[\begin{array}{c} \mathbf{u} \\ p \end{array} \right] = \left[\begin{array}{c} \dot{\mathbf{u}} - \nu \Delta\mathbf{u} + (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla} p \\ \boldsymbol{\nabla} \cdot \mathbf{u} \end{array} \right] \ , \tag{4}$$

then the residual form of the Navier-Stokes equations can be written as,

$$\tilde{\boldsymbol{\mathcal{R}}}(\mathbf{u}, p) = \tilde{\boldsymbol{\mathcal{L}}} \left[\begin{array}{c} \mathbf{u} \\ p \end{array} \right] - \left[\begin{array}{c} \mathbf{f} \\ 0 \end{array} \right] \ . \tag{5}$$

The Picard-linearized steady-state adjoint operator of the Navier-Stokes equations $\tilde{\boldsymbol{\mathcal{L}}}^* : \mathbf{C}^2(\Omega) \times C^1(\Omega) \to \mathbb{R}^3$ writes,

$$\tilde{\boldsymbol{\mathcal{L}}}^* \left[\mathbf{u}; \ \begin{array}{c} \delta\mathbf{u} \\ \delta p \end{array} \right] = \left[\begin{array}{c} -\nu \Delta\delta\mathbf{u} - (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\delta\mathbf{u} - \boldsymbol{\nabla}\delta p \\ -\boldsymbol{\nabla} \cdot \delta\mathbf{u} \end{array} \right] \ , \tag{6}$$

for a given velocity field $\mathbf{u}$ of the primal solution. Given also a stabilization second order tensor $\boldsymbol{\tau} \in \mathfrak{S}^2$, the VMS-stabilized version of variational formulation in Eq. (3) becomes,

$$\begin{aligned} \left\langle \delta\mathbf{u}, \frac{\partial \mathbf{u}}{\partial t} \right\rangle_{0,\Omega} + \langle \nu \boldsymbol{\nabla}\delta\mathbf{u}, \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} - \langle \delta\mathbf{u}, \boldsymbol{\nabla} p \rangle_{0,\Omega} - \langle \delta p, \boldsymbol{\nabla} \cdot \mathbf{u} \rangle_{0,\Omega} - \\ - \left\langle \tilde{\boldsymbol{\mathcal{L}}}^* \left[\mathbf{u}; \ \begin{array}{c} \delta\mathbf{u} \\ \delta p \end{array} \right], \boldsymbol{\tau} \cdot \tilde{\boldsymbol{\mathcal{R}}}(\mathbf{u}, p) \right\rangle_{0,\Omega} = \langle \delta\mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, \bar{\mathbf{t}} \rangle_{0,\Gamma_\mathrm{c}} \ . \end{aligned} \tag{7}$$

The stabilization tensor $\boldsymbol{\tau} = \tau_{\alpha\beta} \, \mathbf{e}_1 \otimes \mathbf{e}_2$ is chosen diagonal, that is, $\tau_{12} = \tau_{21} = 0$, $\tau_{11} = \tau_\mathrm{m}$ and $\tau_{22} = \tau_\mathrm{c}$. In this way, the VMS-stabilized variational formulation in Eq. (7) becomes,

$$\begin{aligned} \langle \delta\mathbf{u}, \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \Delta\delta\mathbf{u}, \nu \, \tau_\mathrm{m} \, \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\delta\mathbf{u}, \tau_\mathrm{m} \, \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \boldsymbol{\nabla}\delta p, \tau_\mathrm{m} \, \dot{\mathbf{u}} \rangle_{0,\Omega} + \langle \boldsymbol{\nabla}\delta\mathbf{u}, \nu \, \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \\ \langle \delta\mathbf{u}, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} - \langle \delta\mathbf{u}, \boldsymbol{\nabla} p \rangle_{0,\Omega} - \langle \delta p, \boldsymbol{\nabla} \cdot \mathbf{u} \rangle_{0,\Omega} - \left\langle \Delta\delta\mathbf{u}, \nu^2 \, \tau_\mathrm{m} \, \Delta\mathbf{u} \right\rangle_{0,\Omega} - \\ \langle (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\delta\mathbf{u}, \nu \, \tau_\mathrm{m} \, \Delta\mathbf{u} \rangle_{0,\Omega} - \langle \boldsymbol{\nabla}\delta p, \nu \, \tau_\mathrm{m} \, \Delta\mathbf{u} \rangle_{0,\Omega} + \langle \Delta\delta\mathbf{u}, \nu \, \tau_\mathrm{m} \, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \\ \langle \boldsymbol{\nabla}\delta p, \tau_\mathrm{m} \, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \langle \Delta\delta\mathbf{u}, \nu \, \tau_\mathrm{m} \, \boldsymbol{\nabla} p \rangle_{0,\Omega} + \langle (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\delta\mathbf{u}, \tau_\mathrm{m} \, \boldsymbol{\nabla} p \rangle_{0,\Omega} + \\ \langle \boldsymbol{\nabla}\delta p, \tau_\mathrm{m} \, \boldsymbol{\nabla} p \rangle_{0,\Omega} + \langle (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\delta\mathbf{u}, \tau_\mathrm{m} \, (\mathbf{u} - \mathbf{u}_\mathrm{d}) \cdot \boldsymbol{\nabla}\mathbf{u} \rangle_{0,\Omega} + \langle \boldsymbol{\nabla} \cdot \delta\mathbf{u}, \tau_\mathrm{c} \, \boldsymbol{\nabla} \cdot \mathbf{u} \rangle_{0,\Omega} = \\ \langle \Delta\delta\mathbf{u}, \nu \, \tau_\mathrm{m} \, \mathbf{f} \rangle_{0,\Omega} + \langle \mathbf{u} \cdot \boldsymbol{\nabla}\delta\mathbf{u}, \tau_\mathrm{m} \, \mathbf{f} \rangle_{0,\Omega} + \langle \boldsymbol{\nabla}\delta p, \tau_\mathrm{m} \, \mathbf{f} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta\mathbf{u}, \bar{\mathbf{t}} \rangle_{0,\Gamma_\mathrm{c}} \ . \end{aligned} \tag{8}$$

2 Spatial discretization

Assumed is that the velocity and pressure fields is discretized with the standard linear *Finite Element Method* (FEM) on triangles. Employing the Buvnon-Galerkin FEM, the unknown velocity/pressures fields $(\mathbf{u}, p)$ and their variation $(\delta\mathbf{u}, \delta p)$ are discretized using the same basis functions $(\boldsymbol{\varphi}_i, \phi_i)$, that is,

$$\left[\begin{array}{c} \mathbf{u}_h \\ p_h \end{array}\right] = \sum_{i=1}^{n} \left[\begin{array}{ccc} \varphi_i & 0 & 0 \\ 0 & \varphi_i & 0 \\ 0 & 0 & \varphi_i \end{array}\right] \left[\begin{array}{c} \hat{u}_{2i-1} \\ \hat{u}_{2i} \\ \hat{p}_i \end{array}\right] , \tag{9a}$$

$$\left[\begin{array}{c} \delta\mathbf{u}_h \\ \delta p_h \end{array}\right] = \sum_{i=1}^{n} \left[\begin{array}{ccc} \varphi_i & 0 & 0 \\ 0 & \varphi_i & 0 \\ 0 & 0 & \varphi_i \end{array}\right] \left[\begin{array}{c} \delta\hat{u}_{2i-1} \\ \delta\hat{u}_{2i} \\ \delta\hat{p}_i \end{array}\right] , \tag{9b}$$

where φ_i are the linear basis functions at the element's parametric space. Let,

$$\boldsymbol{\varphi}_i = \varphi_{\ \mathrm{mod}\,\frac{i}{2}}\ \mathbf{e}_{\ \mathrm{mod}\,\frac{i+1}{2}+1} , \tag{10}$$

be the vector-valued basis function for the discretization of the velocity field along $\mathrm{mod}\,((i+1)/2)+1$-Cartesian direction, where $i = 1, \ldots, 2n$ and $\mathbf{e}_\alpha$ stands for the Cartesian base vector along the α-Cartesian direction. Accordingly, $(\hat{u}_i, \hat{p}_i)$ and $(\delta\hat{u}_i, \delta\hat{p}_i)$ stand for the *Degrees of Freedom* (DOFs) of the unknown and the test fields, respectively, and $n \in \mathbb{N}$ is the number of nodes in the mesh. Subscript h indicates then the smallest element length size within the computational mesh.

Substituting Eq. (9b) in Eq. (8) and taking the variation with respect to DOFs $(\delta\hat{u}_i, \delta\hat{p}_i)$ the following discrete residual systems of equations $\hat{\mathbf{R}}(\mathbf{u}, \mathbf{p})$ and $\hat{\boldsymbol{\mathcal{R}}}(\mathbf{u}, \mathbf{p})$, respectively, are obtained,

$$\begin{aligned} \hat{R}_i(\mathbf{u}, \mathbf{p}) = \frac{\partial(8)}{\partial\delta\hat{u}_i} = {} & \langle\boldsymbol{\varphi}_i, \dot{\mathbf{u}}\rangle_{0,\Omega} - \langle\Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\dot{\mathbf{u}}\rangle_{0,\Omega} - \langle(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\dot{\mathbf{u}}\rangle_{0,\Omega} + \\ & \langle\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \nu\,\boldsymbol{\nabla}\mathbf{u}\rangle_{0,\Omega} + \langle\boldsymbol{\varphi}_i\cdot(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\mathbf{u}\rangle_{0,\Omega} - \langle\boldsymbol{\varphi}_i, \boldsymbol{\nabla}p\rangle_{0,\Omega} - \big\langle\Delta\boldsymbol{\varphi}_i, \nu^2\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}\big\rangle_{0,\Omega} - \\ & \langle(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}\rangle_{0,\Omega} + \langle\Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\mathbf{u}\rangle_{0,\Omega} + \langle\Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\boldsymbol{\nabla}p\rangle_{0,\Omega} + \\ & \langle(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\boldsymbol{\nabla}p\rangle_{0,\Omega} + \langle(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\mathbf{u}\rangle_{0,\Omega} + \\ & \langle\boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_i, \tau_{\mathrm{c}}\,\boldsymbol{\nabla}\cdot\mathbf{u}\rangle_{0,\Omega} - \langle\Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\mathbf{f}\rangle_{0,\Omega} - \langle(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\mathbf{f}\rangle_{0,\Omega} - \\ & \langle\boldsymbol{\varphi}_i, \mathbf{f}\rangle_{0,\Omega} - \langle\boldsymbol{\varphi}_i, \bar{\mathbf{t}}\rangle_{0,\Gamma_{\mathrm{c}}} , \end{aligned} \tag{11a}$$

$$\begin{aligned} \hat{\mathcal{R}}_i(\mathbf{u}, \mathbf{p}) = \frac{\partial(8)}{\partial\delta\hat{p}_i} = {} & -\langle\boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,\dot{\mathbf{u}}\rangle_{0,\Omega} - \langle\varphi_i, \boldsymbol{\nabla}\cdot\mathbf{u}\rangle_{0,\Omega} - \langle\boldsymbol{\nabla}\varphi_i, \nu\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}\rangle_{0,\Omega} + \\ & \langle\boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,(\mathbf{u}-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\mathbf{u}\rangle_{0,\Omega} + \langle\boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,\boldsymbol{\nabla}p\rangle_{0,\Omega} - \langle\boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,\mathbf{f}\rangle_{0,\Omega} . \end{aligned} \tag{11b}$$

Applying the approximations in Eq. (9a) into residual Eqs. (11a) and (11b) one obtains the following matrix-vector expressions for the dynamic residual equations,

$$\left[\begin{array}{c} \hat{\mathbf{R}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \\ \hat{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \end{array}\right] = \left[\begin{array}{cc} \mathbf{M}(\hat{\mathbf{u}}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \dot{\hat{\mathbf{u}}} \\ \dot{\hat{\mathbf{p}}} \end{array}\right] + \left[\begin{array}{c} \bar{\mathbf{R}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \\ \bar{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \end{array}\right] - \left[\begin{array}{c} \hat{\mathbf{F}}(\hat{\mathbf{u}}) \\ \hat{\boldsymbol{\mathcal{F}}} \end{array}\right] , \tag{12}$$

where $\hat{\mathbf{u}}$ and $\hat{\mathbf{p}}$ stand for the complete vectors of velocity and pressure DOFs, that is,

$$\hat{\mathbf{u}} = \left[\begin{array}{ccc} \hat{u}_1 & \cdots & \hat{u}_{2n} \end{array}\right] , \tag{13a}$$

$$\hat{\mathbf{p}} = \left[\begin{array}{ccc} \hat{p}_1 & \cdots & \hat{p}_n \end{array}\right] , \tag{13b}$$

and $\dot{\hat{\mathbf{u}}}$ and $\dot{\hat{\mathbf{p}}}$ their time derivatives. Mass matrices $\mathbf{M}(\hat{\mathbf{u}})$, $\boldsymbol{\mathcal{M}}$, steady-state residual vectors $\bar{\mathbf{R}}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$, $\bar{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ and force vectors $\hat{\mathbf{F}}(\hat{\mathbf{u}})$, $\hat{\boldsymbol{\mathcal{F}}}$ in Eq. (12) are defined as follows,

$$M_{ij}(\hat{\mathbf{u}}) = \left\langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j \right\rangle_{0,\Omega} - \left\langle \Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\boldsymbol{\varphi}_j \right\rangle_{0,\Omega} - \left\langle (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\boldsymbol{\varphi}_j \right\rangle_{0,\Omega} \ , \tag{14a}$$

$$\mathcal{M}_{ij} = -\left\langle \boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,\boldsymbol{\varphi}_j \right\rangle_{0,\Omega} \ , \tag{14b}$$

$$\begin{aligned} \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}}) = {} & \langle \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \nu\,\boldsymbol{\nabla}\mathbf{u}_h \rangle_{0,\Omega} + \langle \boldsymbol{\varphi}_i, (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\mathbf{u}_h \rangle_{0,\Omega} - \langle \boldsymbol{\varphi}_i, \boldsymbol{\nabla} p_h \rangle_{0,\Omega} - \\ & \left\langle \Delta\boldsymbol{\varphi}_i, \nu^2\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}_h \right\rangle_{0,\Omega} - \langle (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}_h \rangle_{0,\Omega} + \\ & \langle \Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,(\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\mathbf{u}_h \rangle_{0,\Omega} + \langle \Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\boldsymbol{\nabla} p_h \rangle_{0,\Omega} + \\ & \langle (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\boldsymbol{\nabla} p_h \rangle_{0,\Omega} + \langle (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,(\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\mathbf{u}_h \rangle_{0,\Omega} + \\ & \langle \boldsymbol{\nabla} \cdot \boldsymbol{\varphi}_i, \tau_{\mathrm{c}}\,\boldsymbol{\nabla} \cdot \mathbf{u}_h \rangle_{0,\Omega} \end{aligned} \tag{14c}$$

$$\begin{aligned} \bar{\mathcal{R}}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}}) = {} & -\langle \varphi_i, \boldsymbol{\nabla} \cdot \mathbf{u}_h \rangle_{0,\Omega} - \langle \boldsymbol{\nabla}\varphi_i, \nu\,\tau_{\mathrm{m}}\,\Delta\mathbf{u}_h \rangle_{0,\Omega} + \langle \boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,(\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\mathbf{u}_h \rangle_{0,\Omega} + \\ & \langle \boldsymbol{\nabla}\varphi_i, \tau_{\mathrm{m}}\,\boldsymbol{\nabla} p_h \rangle_{0,\Omega} \ , \end{aligned} \tag{14d}$$

$$\hat{F}_i(\hat{\mathbf{u}}) = \langle \Delta\boldsymbol{\varphi}_i, \nu\,\tau_{\mathrm{m}}\,\mathbf{f} \rangle_{0,\Omega} + \langle (\mathbf{u}_h - \mathbf{u}_{\mathrm{d}}) \cdot \boldsymbol{\nabla}\boldsymbol{\varphi}_i, \tau_{\mathrm{m}}\,\mathbf{f} \rangle_{0,\Omega} + \langle \boldsymbol{\varphi}_i, \mathbf{f} \rangle_{0,\Omega} + \langle \boldsymbol{\varphi}_i, \bar{\mathbf{t}} \rangle_{0,\Gamma_{\mathrm{c}}} \ , \tag{14e}$$

$$\hat{\mathcal{F}}_i = \langle \tau_{\mathrm{m}}\,\boldsymbol{\nabla}\varphi_i, \mathbf{f} \rangle_{0,\Omega} \ , \tag{14f}$$

where $\mathbf{u}_h$ and p_h is the numerical approximation using the finite element method as per Eqs. (13).

3 Time discretization

3.1 Bossak time integration scheme

Within the so-called Bossak time integration scheme [WBZ80], given two reals $\alpha_{\mathrm{b}}, \gamma_{\mathrm{b}} \in \mathbb{R}$ and a time discretization with time step $dt = t_{\hat{n}+1} - t_{\hat{n}}$, the following approximation of residual equations in Eq. (12) are made,

$$\begin{aligned} \begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \hat{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} = {} & (1-\alpha_{\mathrm{b}}) \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}+1} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}+1} \end{bmatrix} + \alpha_{\mathrm{b}} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} + \\ & \begin{bmatrix} \bar{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \bar{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{F}}_{\hat{n}+1}(\hat{\mathbf{u}}_{\hat{n}+1}) \\ \hat{\boldsymbol{\mathcal{F}}}_{\hat{n}+1} \end{bmatrix} , \end{aligned} \tag{15a}$$

$$\begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1} \\ \hat{\mathbf{p}}_{\hat{n}+1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}} \\ \hat{\mathbf{p}}_{\hat{n}} \end{bmatrix} + dt\,(1-\gamma_{\mathrm{b}}) \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} + dt\,\gamma_{\mathrm{b}} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}+1} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}+1} \end{bmatrix} . \tag{15b}$$

The subscript $\hat{n}$ indicates the time instance at which the quantities are evaluated, that is for example $\hat{\mathbf{u}}_{\hat{n}} = \hat{\mathbf{u}}(t_{\hat{n}})$. Solving Eqs. (15b) for the time derivatives of the unknown fields at the current time instance $t_{\hat{n}+1}$ and substituting them into (15a) one arrives at a nonlinear residual equation which is to be solved for the pair $(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1})$ at the current time instance, namely,

$$\begin{aligned} \begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \hat{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} = {} & \frac{1-\alpha_{\mathrm{b}}}{\gamma_{\mathrm{b}}} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1} \\ \hat{\mathbf{p}}_{\hat{n}+1} \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \bar{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} - \\ & \frac{1-\alpha_{\mathrm{b}}}{dt\,\gamma_{\mathrm{b}}} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}} \\ \hat{\mathbf{p}}_{\hat{n}} \end{bmatrix} - \frac{1-\alpha_{\mathrm{b}}-\gamma_{\mathrm{b}}}{\gamma_{\mathrm{b}}} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \boldsymbol{\mathcal{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} - \\ & \begin{bmatrix} \hat{\mathbf{F}}_{\hat{n}+1}(\hat{\mathbf{u}}_{\hat{n}+1}) \\ \hat{\boldsymbol{\mathcal{F}}}_{\hat{n}+1} \end{bmatrix} . \end{aligned} \tag{16}$$

As equation system in Eq. (16) is highly nonlinear on $(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1})$, it is not fully linearized but only the actual convection term in Eq. (7) is exactly linearized. For all other terms a Picard linearization is assumed, see also in [Cod01]. Let $(\hat{\mathbf{u}}_{\hat{n},\hat{i}}, \hat{\mathbf{p}}_{\hat{n},\hat{i}})$ be the solution at time step $\hat{n}$ and nonlinear iteration $\hat{i}$. Within the employed quasi-Newton linearization, for all the terms which depend on $(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{i}})$ but the actual convection term, the Picard linearization reads,

$$\left[\begin{array}{c}\hat{\mathbf{u}}_{\hat{n}+1,0}\\ \hat{\mathbf{p}}_{\hat{n}+1,0}\end{array}\right]=\left[\begin{array}{c}\hat{\mathbf{u}}_{\hat{n}}\\ \hat{\mathbf{p}}_{\hat{n}}\end{array}\right] \tag{17a}$$

$$\left[\begin{array}{c}\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}}\\ \hat{\mathbf{p}}_{\hat{n}+1,\hat{i}}\end{array}\right]=\left[\begin{array}{c}\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}-1}\\ \hat{\mathbf{p}}_{\hat{n}+1,\hat{i}-1}\end{array}\right],\quad \forall \hat{i}\geq 1 \tag{17b}$$

where $(\hat{\mathbf{u}}_{\hat{n}},\hat{\mathbf{p}}_{\hat{n}})$ stands for the converged solution at time instance $t_{\hat{n}}$ and $(\hat{\mathbf{u}}_{\hat{n}+1,0},\hat{\mathbf{p}}_{\hat{n}+1,0})$ stands for the initial guess of the quasi-Newton iterative procedure at time instance $t_{\hat{n}+1}$. Therefore, the quasi-Newton linearization of residual equations in Eq. (16) leads to the following set of iterations,

$$\begin{aligned}\left(\frac{1-\alpha_{\mathrm{b}}}{\gamma_{\mathrm{b}}}\mathbf{M}(\hat{\mathbf{u}}_{\mathrm{n}+1,\hat{i}})+\left[\begin{array}{cc}\mathbf{K}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}}) & \boldsymbol{\mathcal{C}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}})\\ \boldsymbol{\mathfrak{C}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}}) & \boldsymbol{\mathcal{K}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}})\end{array}\right]\right)\left[\begin{array}{c}\Delta_{\hat{n}+1,\hat{i}}\hat{\mathbf{u}}\\ \Delta_{\hat{n}+1,\hat{i}}\hat{\mathbf{p}}\end{array}\right]=\\ -\left[\begin{array}{c}\hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}})\\ \hat{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}},\hat{\mathbf{p}}_{\hat{n}+1,\hat{i}})\end{array}\right],\end{aligned} \tag{18}$$

where $\Delta_{\hat{n}+1,\hat{i}}\hat{\mathbf{u}}=\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}+1}-\hat{\mathbf{u}}_{\hat{n}+1,\hat{i}}$. Matrices $\mathbf{K}(\hat{\mathbf{u}},\hat{\mathbf{p}})$, $\boldsymbol{\mathcal{C}}(\hat{\mathbf{u}},\hat{\mathbf{p}})$, $\boldsymbol{\mathfrak{C}}(\hat{\mathbf{u}},\hat{\mathbf{p}})$ and $\boldsymbol{\mathcal{K}}(\hat{\mathbf{u}},\hat{\mathbf{p}})$ are defined as follows,

$$\begin{aligned}K_{ij}(\hat{\mathbf{u}},\hat{\mathbf{p}})=\frac{\partial\bar{R}_i(\hat{\mathbf{u}},\hat{\mathbf{p}})}{\partial\hat{u}_j}&=\left\langle\boldsymbol{\nabla}\boldsymbol{\varphi}_i,\nu\,\boldsymbol{\nabla}\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}+\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j\cdot\boldsymbol{\nabla}\mathbf{u}_h\right\rangle_{0,\Omega}+\\ &\left\langle\boldsymbol{\varphi}_i,(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}-\left\langle\Delta\boldsymbol{\varphi}_i,\nu^2\,\tau_{\mathrm{m}}\,\Delta\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}-\\ &\left\langle(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i,\nu\,\tau_{\mathrm{m}}\,\Delta\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}+\left\langle\Delta\boldsymbol{\varphi}_i,\nu\,\tau_{\mathrm{m}}\,(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}+\\ &\left\langle(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i,\tau_{\mathrm{m}}\,(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}+\left\langle\boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_i,\tau_{\mathrm{c}}\,\boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_j\right\rangle_{0,\Omega},\end{aligned} \tag{19a}$$

$$\begin{aligned}\mathcal{C}_{ij}(\hat{\mathbf{u}},\hat{\mathbf{p}})=\frac{\partial\bar{R}_i(\hat{\mathbf{u}},\hat{\mathbf{p}})}{\partial\hat{p}_j}&=\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\nabla}\varphi_j\right\rangle_{0,\Omega}+\left\langle\Delta\boldsymbol{\varphi}_i,\nu\,\tau_{\mathrm{m}}\,\boldsymbol{\nabla}\varphi_j\right\rangle_{0,\Omega}+\\ &\left\langle(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_i,\tau_{\mathrm{m}}\,\boldsymbol{\nabla}\varphi_j\right\rangle_{0,\Omega},\end{aligned} \tag{19b}$$

$$\begin{aligned}\mathfrak{C}_{ij}(\hat{\mathbf{u}},\hat{\mathbf{p}})=\frac{\partial\bar{\mathcal{R}}_i(\hat{\mathbf{u}},\hat{\mathbf{p}})}{\partial\hat{u}_j}&=-\left\langle\varphi_i,\boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}-\left\langle\boldsymbol{\nabla}\varphi_i,\nu\,\tau_{\mathrm{m}}\,\Delta\boldsymbol{\varphi}_j\right\rangle_{0,\Omega}+\\ &\left\langle\boldsymbol{\nabla}\varphi_i,\tau_{\mathrm{m}}\,(\mathbf{u}_h-\mathbf{u}_{\mathrm{d}})\cdot\boldsymbol{\nabla}\boldsymbol{\varphi}_j\right\rangle_{0,\Omega},\end{aligned} \tag{19c}$$

$$\mathcal{K}_{ij}(\hat{\mathbf{u}},\hat{\mathbf{p}})=\frac{\partial\bar{\mathcal{R}}_i(\hat{\mathbf{u}},\hat{\mathbf{p}})}{\partial\hat{p}_j}=\left\langle\boldsymbol{\nabla}\varphi_i,\tau_{\mathrm{m}}\,\boldsymbol{\nabla}\varphi_j\right\rangle_{0,\Omega}. \tag{19d}$$

Unconditional stability for the Bossak time integration algorithm can be ensured by choosing $\gamma_{\mathrm{b}}=1/2-\alpha_{\mathrm{b}}$. In the case of isogeometric finite elements where virtually all derivatives defined in Eqs. (14) are available, the following choice of the stabilization parameters is made [BMCH10],

$$\tau_{\mathrm{m}}=\left(C_{\mathrm{t}}\,dt^{-2}+\frac{\partial X_\alpha}{\partial\theta_\beta}u_{\mathrm{h},\alpha}u_{\mathrm{h},\beta}+C_{\mathrm{i}}\,\nu^2\,\frac{\partial X_\alpha}{\partial\theta_\beta}\frac{\partial X_\alpha}{\partial\theta_\beta}\right)^{-\frac{1}{2}}, \tag{20a}$$

$$\tau_{\mathrm{c}}=\left(\tau_{\mathrm{m}}\left\|\sum_{\alpha=1}^{2}\frac{\partial\mathbf{X}}{\partial\theta_\alpha}\right\|_2^2\right)^{-1}, \tag{20b}$$

where $C_{\mathrm{t}}=C_{\mathrm{i}}=4$ are empirical constants, $\|\bullet\|$ is the Frobenious norm of a matrix, $[\partial X_\alpha/\partial\theta_\beta]$ is the Jacobian matrix of the geometric transformation to the element's parametric space θ_1-θ_2, $\partial\mathbf{X}/\partial\theta_\alpha$ is the base vector along the θ_α-parametric direction and $\|\bullet\|_2$ stands for the 2-norm of a vector. On the other hand, in the case of linear finite elements, such as the *Constant Strain Triangle* (CST), where higher than first order derivatives in Eqs. (14) the stabilization parameters are chosen as [Cod01],

$$\tau_{\mathrm{m}}=\left(C_{\mathrm{t}}\,dt^{-1}+2\,h^{-1}\,\|\mathbf{u}_{\mathrm{h}}\|_2+C_{\mathrm{i}}\,\nu\,h^{-2}\right)^{-1}, \tag{21a}$$

$$\tau_{\mathrm{c}}=\left(\nu+0.5\,h\,\|\mathbf{u}_{\mathrm{h}}\|_2\right)^{-1}, \tag{21b}$$

where $\mathbf{u}_\mathrm{h}$ stands for the numerical velocity field at the integration point from the previous quasi-Newton iteration.

4 Steady-state formulation

For the steady-state formulation of the incompressible Navier-Stokes Eqs. (2) using the VMS stabilization method and the corresponding finite element discretization, the inertial terms are neglected. In this way, the $\hat{i}$-th quasi-Newton iteration becomes,

$$\begin{bmatrix} \mathbf{K}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) & \boldsymbol{\mathcal{C}}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) \\ \boldsymbol{\mathfrak{C}}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) & \boldsymbol{\mathcal{K}}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) \end{bmatrix} \begin{bmatrix} \Delta_{\hat{i}}\hat{\mathbf{u}} \\ \Delta_{\hat{i}}\hat{\mathbf{p}} \end{bmatrix} = - \begin{bmatrix} \bar{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) - \hat{\mathbf{F}}(\hat{\mathbf{u}}_{\hat{i}}) \\ \bar{\boldsymbol{\mathcal{R}}}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\mathbf{p}}_{\hat{i}}) - \hat{\boldsymbol{\mathcal{F}}} \end{bmatrix} . \tag{22}$$

Regarding the estimation of the stabilization parameters, the same formulas as in Eqs. (20) and (21) for the isogeometric and the low-order finite elements, respectively, hold also for the steady-state case with the only difference that $C_\mathrm{t} = 0$.

Equation systems in Eqs. (18) and (22) are obviously unsymmetric and not necessarily well-conditioned. Preconditioners have been studied in the literature, see for example in [BMCH10], but in this work no preconditioners are used and the linear equation systems are solved as presented herein in 2D.

5 Steady-state drag optimization using finite differencing

Given is a portion of the domain's boundary $\gamma_\mathrm{D} \subset \Gamma$ over which the drag force f_D is sought. If $\gamma_{\mathrm{D},h}$ stands for the geometric discretization of γ_D due to the finite element approximation of the primal fields, then

$$f_\mathrm{D}(\mathbf{u}, p) = \frac{1}{\rho} \sum_{i \in \mathcal{I}_\mathrm{D}} \bar{R}_{2i-1}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \, , \tag{23}$$

where $\mathcal{I}_\mathrm{D}$ stands for the set of indices corresponding to the finite element nodes on $\gamma_{\mathrm{D},h}$. Eq. (23) is nothing else but the sum of all forces by means of the momentum residuals in Eq. (14c) acting on $\gamma_{\mathrm{D},h}$ along the X_1-Cartesian direction depending on the unknown fields $\mathbf{u}$ and p, which are in turn defined by their nodal values $\hat{\mathbf{u}}$ and $\hat{\mathbf{p}}$ within the finite element approximation in Eqs. (9).

Herein the unconstrained shape optimization of a cylinder in flow is considered. The only design variable is the radius of the cylinder r and the objective/cost function $J(\mathbf{u}(r), p(r), r) = f_\mathrm{D}(\mathbf{u}(r), p(r), r)$ is the drag force. It is expected that the cylinder collapses into a point since no other constraint is assumed. The optimization problem can then be formulated as follows,

$$r = \arg\min_{r'} J(\mathbf{u}(r'), p(r'), r') \tag{24a}$$

$$\text{s.t.} \quad \begin{bmatrix} \bar{\mathbf{R}}(\mathbf{u}(r), p(r)) \\ \bar{\boldsymbol{\mathcal{R}}}(\mathbf{u}(r), p(r)) \end{bmatrix} = \mathbf{0} \text{ in } \Omega(r) \, . \tag{24b}$$

Eqs. (24) demonstrate that there is an implicit dependence of the state to the design variable, that is, $\mathbf{u} = \mathbf{u}(r)$ and $p = p(r)$. Within the Finite Differencing (FD) approach, problem in Eq. (24) is solved in a staggered iterative manner, see also in [Ble14]. Accordingly, at each optimization iteration/design update, the state variables $\mathbf{u}$ and p are computed such that Eq. (24b) is satisfied, see Sec. 4. To minimize objective/cost function in Eq. (24a), its total variation should be equal to zero, that is, $\delta f_\mathrm{D}(\mathbf{u}(r), p(r)) = 0$, which results in the following residual form,

$$s(\mathbf{u}(r), p(r), r) = \frac{\partial J}{\partial u_\alpha} \frac{\partial u_\alpha}{\partial r} + \frac{\partial J}{\partial p} \frac{\partial p}{\partial r} + \frac{\partial J}{\partial r} \, , \tag{25}$$

where $\mathbf{u} = u_\alpha \mathbf{e}_\alpha$. Eq. (25) is known as the sensitivity equation and is to be solved for the design update. Linearization of Eq. (25) results in,

$$H(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}})\, \Delta_{\tilde{i}} r = -s(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}}) \;, \tag{26}$$

where $\Delta_{\tilde{i}} r = r_{\tilde{i}+1} - r_{\tilde{i}}$ and $\tilde{i}$ stands for the index of the optimization iteration. Hessian matrix H is given by,

$$H(\mathbf{u}(r), p(r), r) = \frac{\partial s(\mathbf{u}(r), p(r), r)}{\partial r} \;. \tag{27}$$

Herein, the Hessian matrix in Eq. (27) is approximated with the identity matrix, also because its computation is very costly. Therefore Eq. (26) becomes,

$$r_{\tilde{i}+1} = r_{\tilde{i}} - s(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}}) \;, \tag{28}$$

which indicates the the shape update must be in the direction opposite to the sensitivity. Another direct implication of choosing the identity matrix in the place of the Hessian is that no quadratic convergence in the neighborhood of the solution must be expected. The Hessian on the other hand might be ill-conditioned dependent on the primal problem, the design variables and the objection function which often requires the use of preconditioners.

As aforementioned, for each design update $r_{\tilde{i}}$ the primal variables $\mathbf{u}(r_{\tilde{i}})$ and $p(r_{\tilde{i}})$ are updated such that Eq. (24b) is satisfied. Then, within FD given a small enough perturbation parameter $eps\tilde{i}lon > 0$, sensitivity $s(\mathbf{u}_{\tilde{i}}(r_{\tilde{i}}), p_{\tilde{i}}(r_{\tilde{i}}), r_{\tilde{i}})$ in Eq. (25) is computed as,

$$s(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}}) \approx \frac{J(\mathbf{u}(r_{\tilde{i}} + \tilde{\epsilon}), p(r_{\tilde{i}} + \tilde{\epsilon}), r_{\tilde{i}} + \tilde{\epsilon}) - J(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}})}{\tilde{\epsilon}} \;. \tag{29}$$

1. Initialize optimization counter $\tilde{i} = 1$ and design parameter $r_1 = r_0$;

while $J(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}}) > \epsilon \wedge \; \tilde{i} \leq n_{\max}$ **do**

1i. Solve the primal problem in Eq. (22) for the design $r_{\tilde{i}}$ to obtain objective $J(\mathbf{u}(r_{\tilde{i}}), p(r_{\tilde{i}}), r_{\tilde{i}})$;

1ii. Evaluate convergence conditions and if they are fulfilled break the loop;

1iii. Solve the perturbed system in Eq. (22) for the design $r_{\tilde{i}} + \tilde{\epsilon}$ to obtain objective $J(\mathbf{u}(r_{\tilde{i}} + \tilde{\epsilon}), p(r_{\tilde{i}} + \tilde{\epsilon}), r_{\tilde{i}} + \tilde{\epsilon})$;

1iv. Compute the sensitivity using Eq. (29);

1v. Update the design using Eq. (28) ;

1vi. Update the optimization iteration $\tilde{i}++$;

end

Algorithm 1: Unconstrained optimization using finite differencing

6 Preprocessing using GiD®

Herein it is introduced a tutorial for the demonstration of all the necessary steps in the setup of a case in GiD® by generating a corresponding input file. In particular, the simple flow around a cylinder case in 2D is employed which is often used for the verification fluid solvers. GiD® only acts as a preprocessor while the actual analysis takes place within *cane* MATLAB® framework. Setting up a computational fluid dynamics (CFD) problem is similar to setting up a standard FEM-based structural analysis, the difference here lying in the naming of the functions and time integration options.

6.1 Problem Type

The user must specify the MATLAB® GiD® problem type which can be found under `./cane/matlab.gid` (Fig. 1). Select *Data* → *Problem Type* → *Load...*, locate the folder where

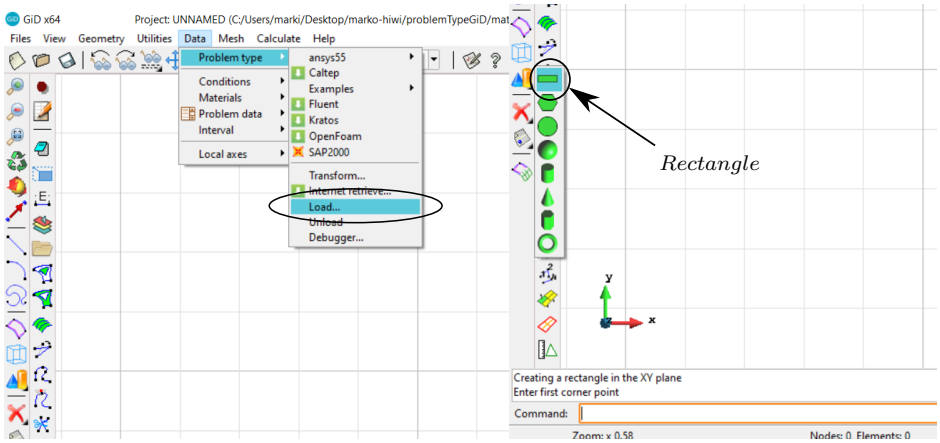

Figure 1: Problem type and geometry selection.

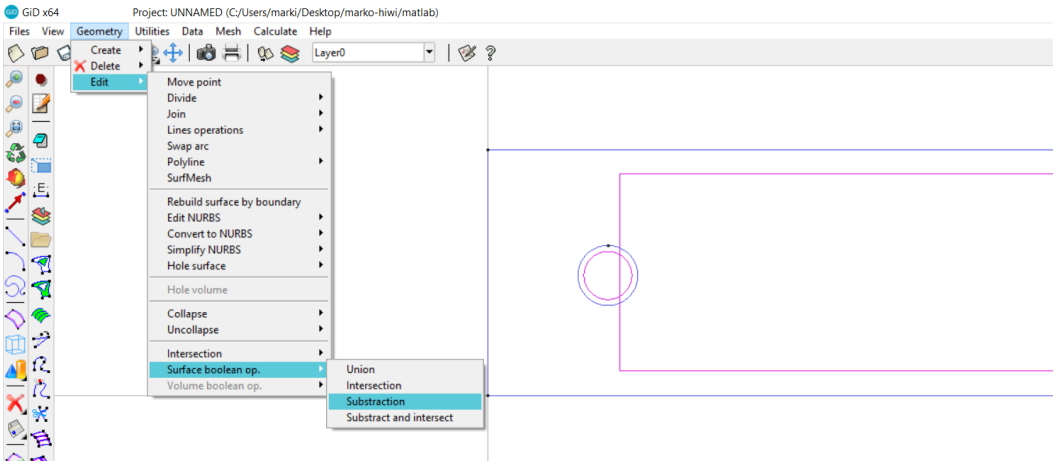

Figure 2: Subtracting circle from rectangle.

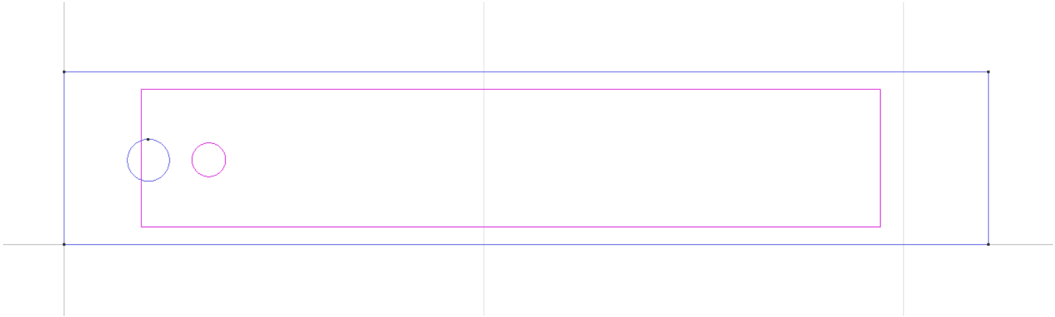

Figure 3: Final geometry.

the MATLAB® problem type is saved and then select `problemTypeGiD/matlab`. This is a custom problem type made specifically for the interface between GiD® and *cane* MATLAB® framework. This problem type can be expanded along with the parser of the corresponding analysis (e.g. CFD analysis, etc.) depending on the user needs.

6.2 Geometry setup

To create a geometry, select *create object* → *rectangle*. The rectangle can be drawn by clicking on the drawing plane or by specifying the coordinates of its corner edges (Fig. 1). The coordinates must be typed in the format `x y z`. They must contain white space between each coordinate, whereas omitting coordinates are assumed by default to be zero. Enter the first point `0 0` in the command

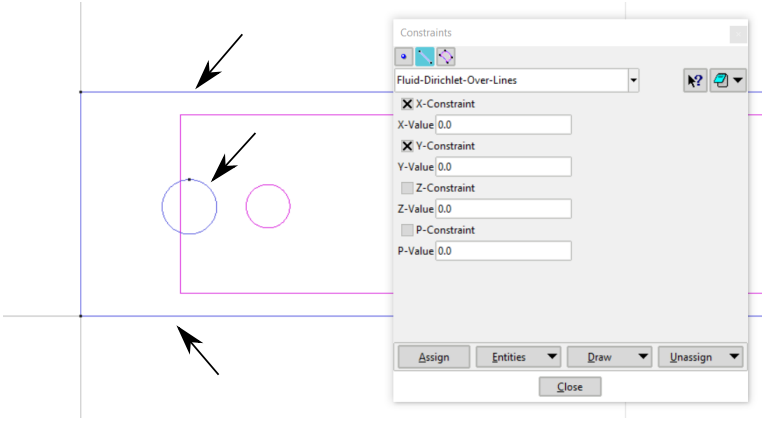

Figure 4: Applied no-slip boundary conditions.

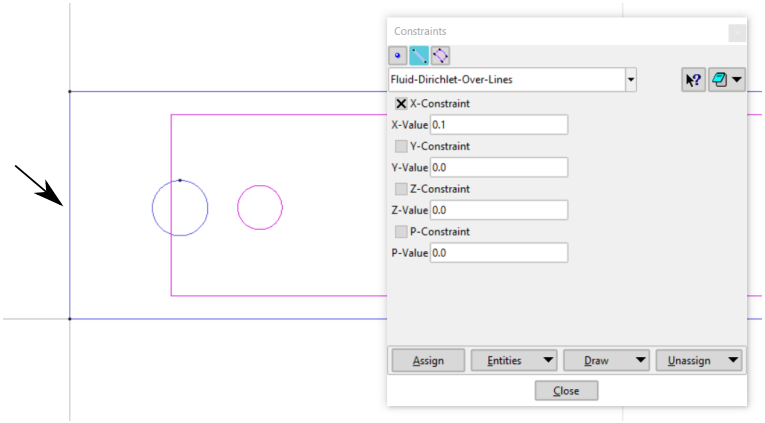

Figure 5: Applied inlet boundary conditions.

line and confirm with *esc*. Now enter the second point `2.2 0.41` and again confirm with *esc*. Next the creation of a circle with coordinates `0.2 0.2` with normal in positive `z` direction and radius `0.05` is demonstrated.

Two independent surface geometries are generated as shown in Fig. 2. Next step is to subtract the circle from the rectangle to get the desirable shape. Select *Geometry* → *Edit* → *Surface Boolean op.* → *Substraction*. Firstly, the surface to be subtracted (rectangle) needs to be selected, then the action needs to be confirmed using *esc* and lastly the subtracting surface (circle) needs to be selected. The resulting geometry is now complete (Fig. 3). At this point it is recommended to save the project.

The console output of GiD® can be used to identify which commands have already be selected and which action is expected to be taken for given commands.

6.3 Dirichlet boundary conditions

To specify the Dirichlet boundary conditions select *Data* → *Conditions* → *Constraints*. Select *lines* (line icon) as selection type and select *Fluid-Dirichlet-Over-Lines*. To apply the no-slip boundary conditions set the `x` and `y` component of the velocity to `0.0` on top and bottom line and on the circle as shown in Fig. 4. Be aware that if you again select *Fluid-Dirichlet-Over-Lines* boundary conditions on the same line or node, the new condition will overwrite the previous one.

To specify the inlet boundary conditions set the `x` component of the velocity to `0.1` as shown in Fig. 5. Note that this kind of inlet boundary condition just specifies the `x` component of the velocity field whereas the `y` component of the velocity is free. Typically the full velocity vector needs to be specified at the inlet but herein only one component is used for the sake of simplicity. As

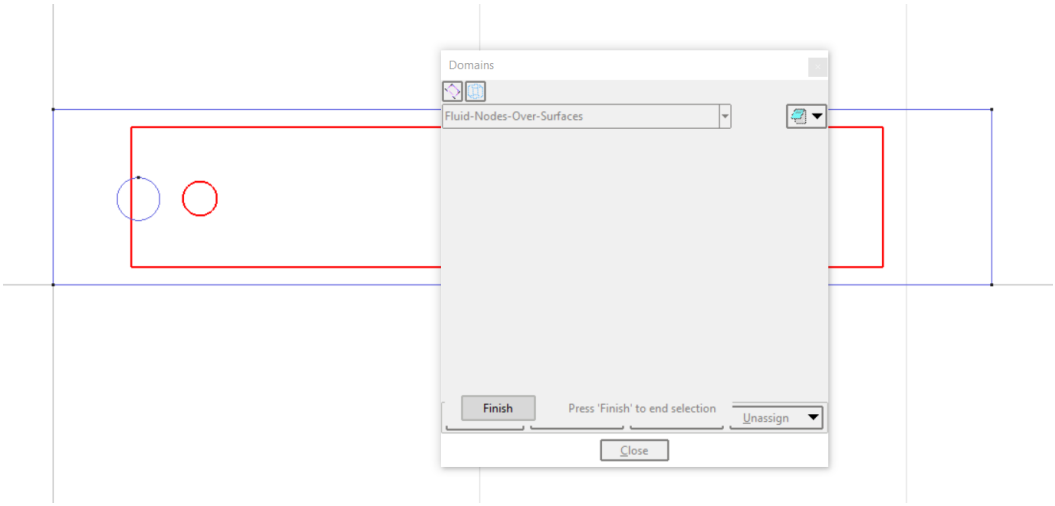

Figure 6: Defined domain for the mesh elements and nodes.

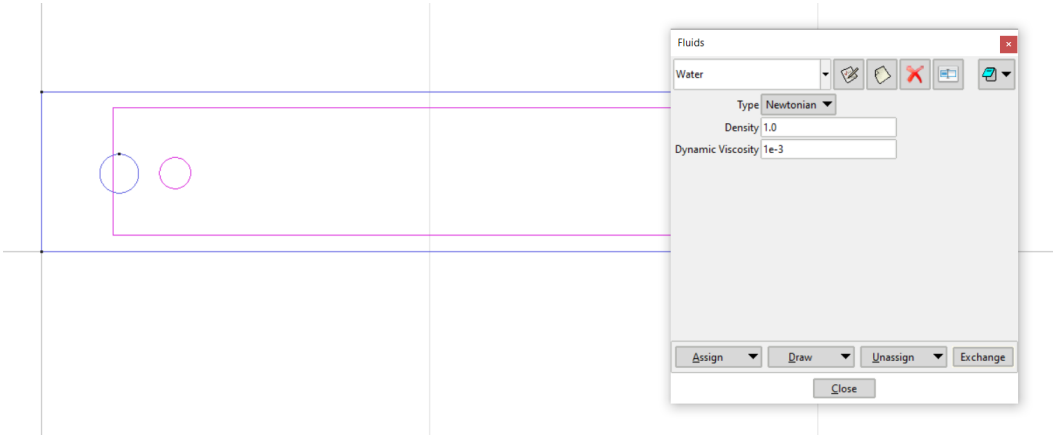

Figure 7: Define material over surfaces.

mentioned earlier, this overwrites previously specified no-slip conditions on top-left and bottom-left node (corner). The order of specifying the Dirichlet boundary conditions is therefore important and dependent on the application of the desirable conditions in each case.

No Dirichlet boundary condition is specified at the outlet in this case, but theoretically the pressure may well be prescribed in the same manner as the velocity field in the inlet boundary.

6.4 Definition of the computational domain

The computational mesh needs then to be assigned to a domain, so that the nodes and the elements for the chosen domain are written out to the desirable input file. Select *Data* → *Conditions* → *Domains*, choose *Fluid-Nodes-Over-Surfaces* from the drop-down menu and select the whole surface according to Fig. 6. Confirm with *esc*. Then, select *Fluid-Elements-Over-Surfaces* and repeat the previous step to assign the elements to the computational domain.

6.5 Selection of the material properties

In order to select the material properties, select *Data* → *Materials* → *Fluids*. There one can select the default material *Water* from the drop-down menu (Fig. 7) or just change the given parameters to adjust material properties. Apply the material to the geometry of the problem by selecting *Assign* → *Surfaces* and choose the surface of defining the problem's domain. Save the changes if asked so. The user may also expand the materials selection by editing the corresponding files under the folder *matlab.gid*.

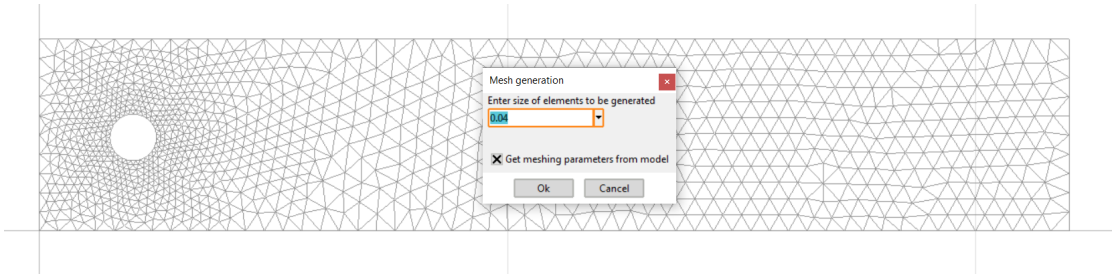

Figure 8: Mesh generation with mesh size.

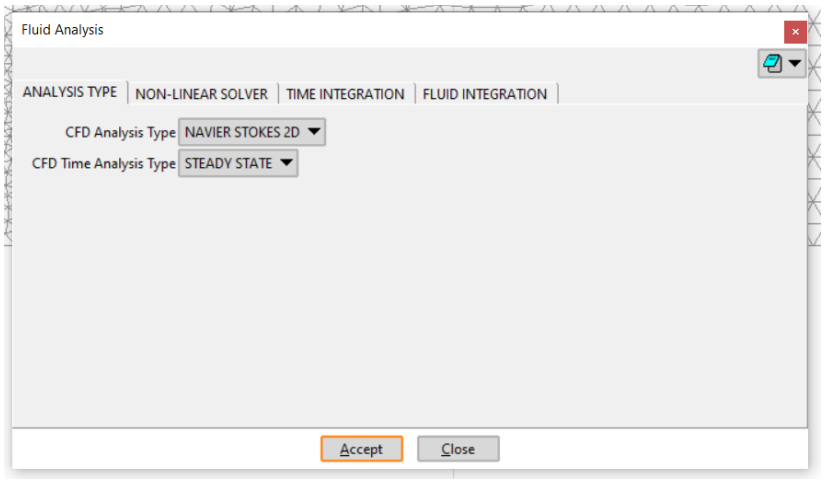

Figure 9: Analysis type and solver setup.

6.6 Generation of the computational mesh

Lastly, the finite element mesh needs to be generated. The simplest way is to go to *Mesh* → *Generate Mesh...* and specify the element size (Fig. 8). Be aware that only triangular elements are implemented in *cane* MATLAB® framework for the CFD analysis. So make sure to select those if using a structured mesh. For more details about the mesh generation please look at the official GiD® documentation.

6.7 Selection of the analysis type and solver setup

To select the analysis type and setup the solver, select *Data* → *Problem Data* → *Fluid Analysis*. In this window the user can select 2D or 3D analysis and furthermore choose between a steady state or a transient analysis. Many more settings are possible, specifically on the time integration schemes and Gauss integration, but they are not needed for this simple case in which the default options are sufficient.

6.8 Generation of input file for analysis in MATLAB®

After the setup is complete, select *Calculation* → *Calculation (F5)* or just select *F5* to write out the input file (Fig. 10) which will be later on parsed within *cane* MATLAB® framework. This file has the same name as our project whereas its extension is `.dat`. The user can also open it with any text editor to check or adjust the data. This file needs then to be placed under folder `./cane/inputGiD/FEMComputationalFluidDynamicsAnalysis` and then a new `caseName` with the same name needs to be defined in the MATLAB® main driver script located under `./cane/man/main_steadyStateIncompressibleNavierStokesFlow`. Furthermore, the user must specify the initial conditions within the *cane* MATLAB® framework.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%   Fluid Boundary Value Problem                                                %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FLUID_ANALYSIS
 ANALYSIS_TYPE,NAVIER_STOKES_2D

FLUID_MATERIAL_PROPERTIES
 DENSITY,1.0
 DYNAMIC_VISCOSITY,1e-3

FLUID_NLINEAR_SCHEME
 NLINEAR_SCHEME,newtonRapshon
 NO_LOAD_STEPS,1
 TOLERANCE,1e-5
 MAX_ITERATIONS,10

FLUID_TRANSIENT_ANALYSIS
 SOLVER STEADY_STATE
 TIME_INTEGRATION BOSSAK
 ALPHA_BETA -0.1
 GAMMA 0.6
 START_TIME 0
 END_TIME 10
 NUMBER_OF_TIME_STEPS 1e3
 ADAPTIVE_TIME_STEPPING false

FLUID_INTEGRATION
 DOMAIN default
 domainNoGP 1
 boundaryNoGP 1
```

Figure 10: GiD® input file in detail.

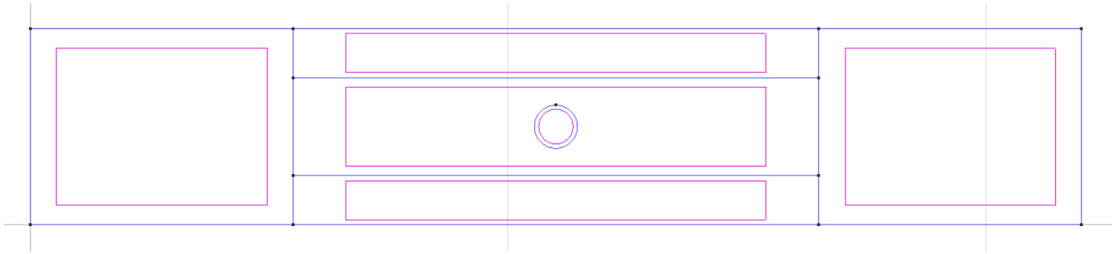

Figure 11: Geometry for ALE mesh motion.

7 Arbitrary Lagrangian-Eulerian description of the motion

The *cane* MATLAB® framework also allows to have a mesh motion imposed on a part of the mesh. This feature is necessary for imposing a desirable mesh motion within *Fluid-Structure Interaction* or *Shape Optimization* as explained in the theory. The underlying algorithm is that of the pseudostructural solver, where the mesh motion is simply treated as structural deformation with a given material properties (Young's modulus equal to 1E+3 and Poisson ratio equal to zero). Herein a short tutorial demonstrating the use of the mesh motion algorithm within `cane` is provided. All the necessary steps to appropriately setup the case up to this point are provided in Sec. 6.

The case used herein as a demonstration is similar to the one used in Sec. 6. It is a typical flow around a cylinder problem, but in this case the cylinder and the surrounding mesh moves back and forth (as a function of time) during the transient analysis (Fig. 11). The corresponding GiD® problem can be found under `./cane/gid_cases/cylinder2D_backAndForth_ALE.gid` and can be directly loaded to GiD®.

The implemented functions for ALE mesh motion are problem specific. The user needs expand the functions found under `./cane/FEMComputationalFluidDynamicsAnalysis/ALEMotion` to account for the desirable mesh motions for more different cases. As a general rule, the mesh motion along

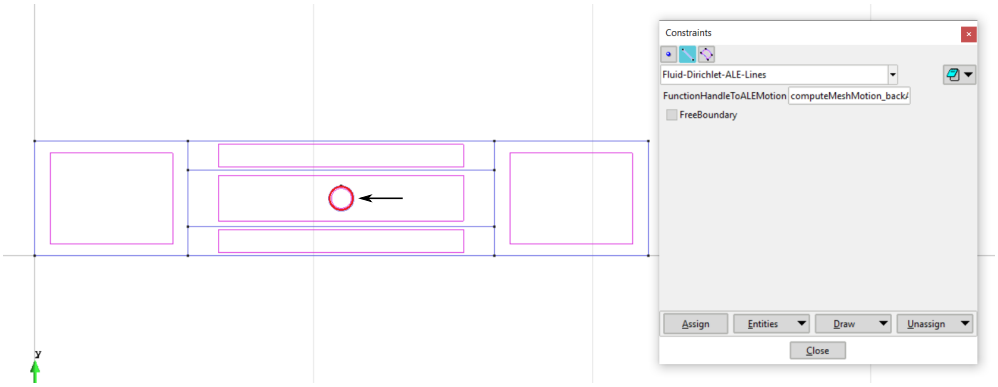

Figure 12: Fix boundary selection.

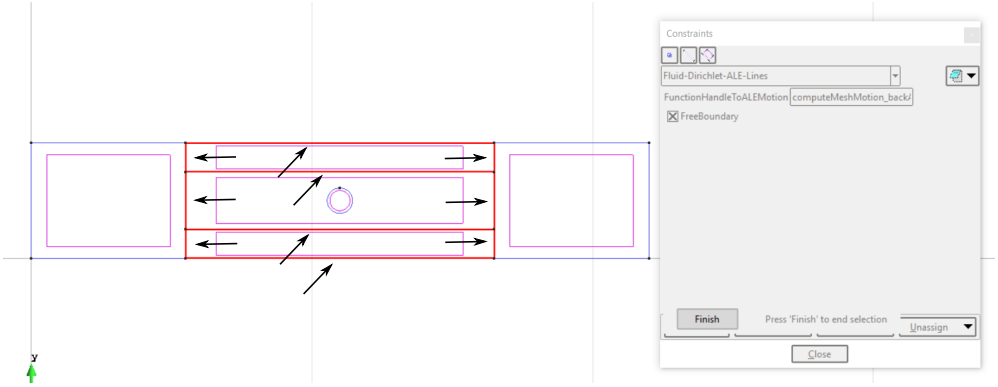

Figure 13: Free boundary selection of lines.

a portion of the computational domain may be specified with a function within `cane` MATLAB® framework.

7.1 Specifying an arbitrary Lagrangian-Eulerian mesh motion

To specify which parts of the geometry will move during the simulation select *Data* → *Conditions* → *Constraints*. Select *lines* (line icon) as selection type and choose *Fluid-Dirichlet-ALE-Lines*. From here on all the options are defined as function handles. The user can also specify which lines or points belong to the free boundary. This is a boundary that lies inside the computational mesh and should not impose the solution vector at that location (velocity and pressure field remains unaffected by these boundaries meaning that they are not prescribed by the mesh motion).

Firstly, specify the `computeMeshMotion_backAndForth` as a `functionHandle` and assign it to the circle in the middle as shown in Fig. 12. This should not be selected as a free boundary as the velocity of the moving cylinder should be imposed into the fluid.

Then keep the same `functionHandle` and check the *FreeBoundary* box. Apply this selection to the lines shown on Fig. 13 so that the mesh motion at these boundaries is not imposed into the flow variables.

Next the selection from the drop-down menu needs to be changed to *points* (point icon) and the selection *Fluid-Dirichlet-ALE-Points* needs to be chosen. Once more the `computeMeshMotion_backAndForth` needs to be specified as a `functionHandle` together with the *FreeBoundary* box. Apply this selection to all the points that connect the lines from previous selection according to Fig. 14.

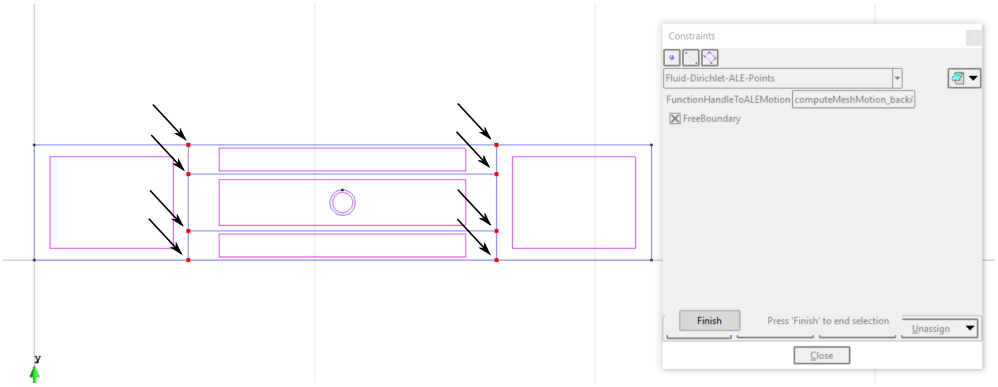

Figure 14: Free boundary selection of points.

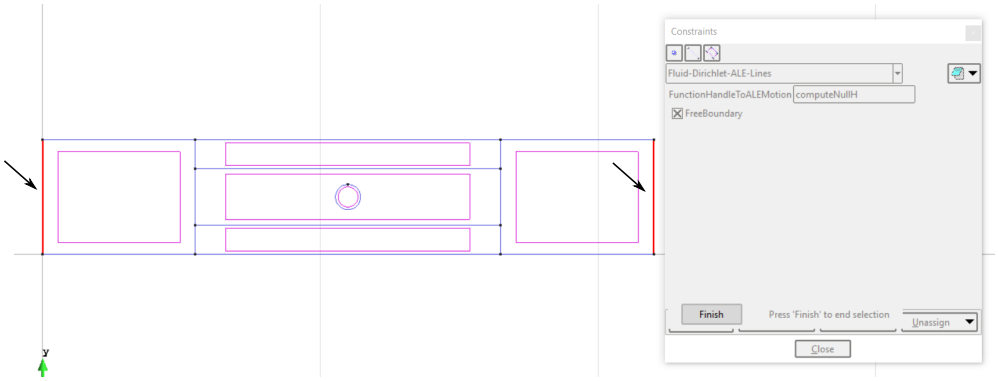

Figure 15: Fixed mesh motion for vertical lines.

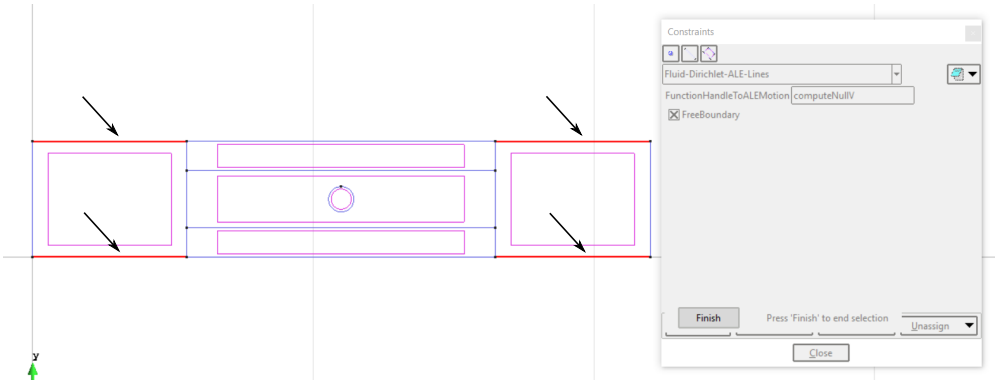

Figure 16: Fixed mesh motion for horizontal lines.

7.2 Specifying a fixed boundary for the arbitrary Lagrangian-Eulerian mesh motion

For the mesh motion algorithm to appropriately work, fixed conditions need to also be accordingly imposed. To do so, selection *Data* → *Conditions* → *Constraints* needs to be firstly chosen. Select *lines* (line icon) as selection type and choose *Fluid-Dirichlet-ALE-Lines*. Specify the `computeNullH` as a `functionHandle` together with the *FreeBoundary* box. This already implemented function in MATLAB® returns a zero horizontal motion at the selected boundary. Apply this selection to the vertical lines as shown on Fig. 15. Next specify the `computeNullV` as a `functionHandle` together with the *FreeBoundary* box and apply this selection to the horizontal lines according to Fig. 16. This already implemented function in MATLAB® returns a zero vertical motion at the horizontal lines. These conditions are applied so that the outer domain does not change though the mesh motion.

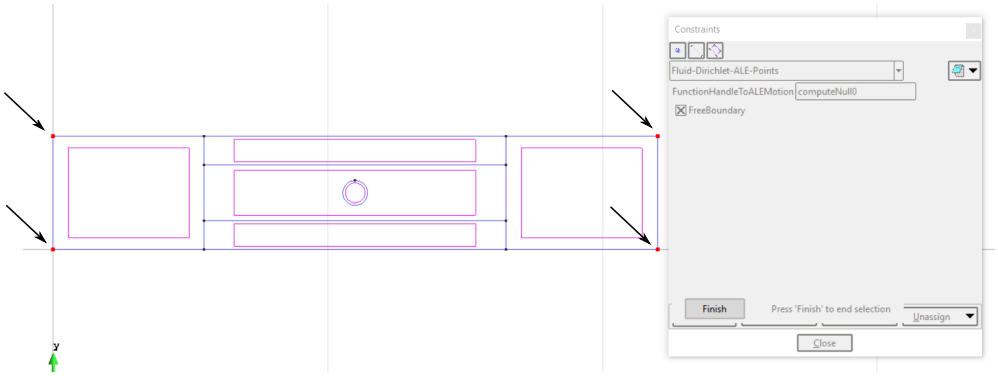

Figure 17: Fixed mesh motion for points.

Lastly the remaining outer corners of the domain need to be specified as fixed. Change the selection from the drop-down menu to *points* (point icon) and choose *Fluid-Dirichlet-ALE-Points*. Now specify the `computeNull0` as a `functionHandle` together with the *FreeBoundary* box. This already implemented function in MATLAB® returns zero displacement at the corner points. Apply this selection to the outer-most points of the domain as shown on Fig. 17. At this point the mesh and subsequently the GiD® input file can be generated and the ALE conditions will be written out along with the rest of the CFD attributes.

References

[Ble14] Kai-Uwe Bletzinger. A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Structural and Multidisciplinary Optimization*, 49(6):873–895, 2014.

[BMCH10] Y. Bazilevs, C. Michler, V.M. Calo, and T.J.R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(13):780 – 790, 2010. Turbulence Modeling for Large Eddy Simulations.

[Cod01] Ramon Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190(20):2681 – 2706, 2001.

[HFMQ98] Thomas J.R. Hughes, Gonzalo R. Feijóo, Luca Mazzei, and Jean-Baptiste Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1):3 – 24, 1998. Advances in Stabilized Methods in Computational Mechanics.

[WBZ80] W. L. Wood, M. Bossak, and O. C. Zienkiewicz. An alpha modification of newmark's method. *International Journal for Numerical Methods in Engineering*, 15(10):1562–1566, 1980.