Interdisziplinäres Zentrum
für Wissenschaftliches Rechnen (IWR)
Ruprecht–Karls–Universität Heidelberg

Prof. Dr. Vincent Heuveline
Dr. Philipp Gerstner

Exercise Sheet 9

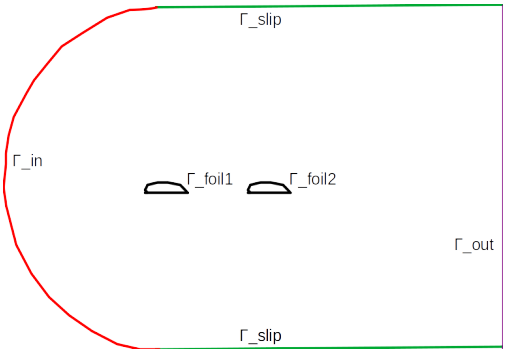
25.06.2021 – 02.07.2021

Computational Fluid Dynamics (Summer Term 2021)

Exercise 1 (Programming). The effect of dirty air (15 Points)
In this exercise you will solve the instationary Navier-Stokes equations to compute the flow around two airfoils:

$$\begin{aligned} \partial_t \mathbf{v} - \nu\Delta\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p &= \mathbf{f} \ \text{ in } [0,T]\times\Omega \\ \nabla\cdot\mathbf{v} &= 0 \ \text{ in } [0,T]\times\Omega \\ \mathbf{v} &= (v_x, 0) \ \text{ on } [0,T]\times\Gamma_{in}\cup\Gamma_{slip} \\ \mathbf{v} &= 0 \ \text{ on } [0,T]\times\Gamma_{foil} \\ (\nu\nabla\mathbf{v} - pI)\vec{n} &= 0 \ \text{ on } [0,T]\times\Gamma_{out} \\ \mathbf{v}(0) &= 0 \ \text{ on } \Omega \end{aligned}$$

with $\nu > 0$, $v_x > 0$, $\mathbf{f} = (0,-1)$ and domain Ω and boundaries given by the figure below.



In this setting, the foils are exposed to the force

$$\mathbf{f}_{foil} = \int_{\Gamma_{foil}} \sigma(\mathbf{v},p)\vec{n}_{foil}\,\mathrm{dS}$$

with Cauchy stress tensor

$$\sigma(\mathbf{v},p) = \mu(\nabla\mathbf{v} + \nabla\mathbf{v}^T) - pI, \ \text{ and } \mu = \nu \ \text{(unit density)} \ ,$$

and $\vec{n}_{foil}$ denotes the unit normal on Γ_{foil} directing into the fluid domain.

Using the backward Euler method, the time-discrete system reads: for $n = 1,\dots,N$:

$$\begin{aligned} \frac{1}{k}(\mathbf{v}_n - \mathbf{v}_{n-1}) - \nu\Delta\mathbf{v}_n + (\mathbf{v}_n\cdot\nabla)\mathbf{v}_n + \nabla p_n &= \mathbf{f}(t_n) \ \text{ in } \Omega \\ \nabla\cdot\mathbf{v}_n &= 0 \ \text{ in } \Omega \\ \mathbf{v}_n &= \mathbf{v}_D(t_n) \ \text{ on } \partial\Omega \\ \mathbf{v}_0 &= 0 \end{aligned} \tag{1}$$

Here, $(\mathbf{v}_n, p_n)$ denotes the approximation to $(\mathbf{v}, p)(t_n)$ at time $t_n = nk$ with time step size $k > 0$. The time-discretized system has to be solved for $(\mathbf{v}_n, p_n)$ while $\mathbf{v}_{n-1}$ is used from the previous time step. System (1) corresponds to the nonlinear, stationary Navier-Stokes equations (with additional term $\frac{1}{k}(\mathbf{v}_n - \mathbf{v}_{n-1})$) which can be solved by Newton's method, analogously to sheet 8.

a. Modify the matrix assembly routine of the class LocalFlowAssembler at the location marked by TODO exercise A in ex8_FoilNavierStokes.h to compute $D_{(\mathbf{v},p)}F_\nu((\mathbf{v}_l, p_l); \cdot, \cdot)$.

Hint: In this routine, the data fields sol_ns_[q], grad_sol_ns_[q] store the values of the previous Newton iterate at quadrature point q, i.e. $(\mathbf{v}^l, p^l)(x_q)$ and $(\nabla\mathbf{v}^l, \nabla p^l)(x_q)$, respectively.

b. Modify the vector assembly routine of the class LocalFlowAssembler at the location marked by TODO exercise B in ex8_FoilNavierStokes.h to compute $F_\nu((\mathbf{v}_l, p_l), \cdot)$.

Hint: In this routine, the data fields sol_ns_[q], grad_sol_ns_[q] store the values of the previous Newton iterate at quadrature point q, i.e. $(\mathbf{v}^l, p^l)(x_q)$ and $(\nabla\mathbf{v}^l, \nabla p^l)(x_q)$, respectively. Further, the data fields sol_ts_[q], grad_sol_ts_[q] store the values of the previous time step at quadrature point q, i.e. $(\mathbf{v}_n, p_n)(x_q)$ and $(\nabla\mathbf{v}_n, \nabla p_n)(x_q)$, respectively.

c. Run the code on up to 12 MPI processes for $\nu = 1e-1$, $f_z = -1$, $v_X = 1$, $T = 0.05$, $k = 0.005$. Note that all parameters are defined in the parameter file ex8_FoilNavierStokes.xml and can be changed without recompilation.

The program output is given by a series of pvtu files ex8_solution_n.pvtu with n denoting the time step and a csv file pp_values.csv. Each row in the csv file corresponds to one time step n and is of the form

$$\begin{pmatrix} t_n & \|\nabla\cdot\mathbf{v}(t_n)\|_2 & \mathbf{f}_{foil1}[0] & \mathbf{f}_{foil1}[1] & \mathbf{f}_{foil1}[1]\mathbf{f}_{foil1}[0]^{-1} & \mathbf{f}_{foil2}[0] & \mathbf{f}_{foil2}[1] & \mathbf{f}_{foil2}[1]\mathbf{f}_{foil2}[0]^{-1} \end{pmatrix}$$

Plot these quantities over time t_n and submit it.

Load the provided paraview state file. This will automatically perform the setup of visualization filters. See the pdf paraview_notes.pdf, provided by sheet 8, for further details. Save an animation for the loaded solution files.

d. The mesh file is created by the python script ex8_FoilStokes/mesh_generation/create_channel_mesh.py. You can run it by the terminal command python3 create_channel_mesh.py inside the directory mesh_generation. Afterwards, you have to run ccmake ../ inside the Hiflow build directory in order to copy the newly created mesh channel.inp into the example build directory.

In this script, the variables angle_foil_1, angle_foil_2 (in radians, $\in [-\pi, \pi]$) set the angles of attack for both foils. Moreover, the variable foil_dist determines the distance between both foils.

Vary the angles and the distance to see how these variables affect the drag and down force acting on foil2, which is located in the slipstream of foil1.

This short video shows what may happen in practice, if a foil is located in the dirty air, created by another foil: https://www.youtube.com/watch?v=e06bEuBOEG4

The corresponding code framework is ex8_FoilNavierStokes. Don't forget to the modify the file exercises/CMakeLists.txt accordingly, see the first exercise sheet for details.

Submission: until 02.07.2021, 11 am, moodle upload