Ambit – A FEniCS-based cardiovascular multi-physics solver

Dr.-Ing. Marc Hirschvogel

February 16, 2024

Contents

1 Preface

Ambit [10] is an open-source multi-physics finite element solver written in Python, supporting solid and fluid mechanics, fluid-structure interaction (FSI), and lumped-parameter models. It is tailored towards solving problems in cardiac mechanics, but may also be used for more general nonlinear finite element analysis. The code encompasses re-implementations and generalizations of methods developed by the author for his PhD thesis [9] and beyond. Ambit makes use of the open-source finite element library FEniCS/dolfinx (https://fenicsproject.org) [15] along with the linear algebra package PETSc (https://petsc.org) [2], hence guaranteeing a state-of-the-art finite element and linear algebra backend. It is constantly updated to ensure compatibility with a recent dolfinx development version. I/O routines are designed such that the user only needs to provide input files that define parameters through Python dictionaries, hence no programming or in-depth knowledge of any library-specific syntax is required.

Ambit provides general nonlinear (compressible or incompressible) finite strain solid dynamics [13], implementing a range of hyperelastic, viscous, and active material models. Specifically, the well-known anisotropic Holzapfel-Ogden [14] and Guccione models [8] for structural description of the myocardium are provided, along with a bunch of other models. It further implements strain- and stress-mediated volumetric growth models [7] that allow to model (maladaptive) ventricular shape and size changes. Inverse mechanics approaches to imprint loads into a reference state are implemented using the so-called prestressing method [5] in displacement formulation [17].

Furthermore, fluid dynamics in terms of incompressible Navier-Stokes/Stokes equations – either in Eulerian or Arbitrary Lagrangian-Eulerian (ALE) reference frames – are implemented. Taylor-Hood elements or equal-order approximations with SUPG/PSPG stabilization [18] can be used.

A variety of reduced 0D lumped models targeted at blood circulation modeling are implemented, including 3- and 4-element Windkessel models [20] as well as closed-loop full circulation [12] and coronary flow models [1].

Monolithic fluid-solid interaction (FSI) [16] in ALE formulation using a Lagrange multiplier field is supported, along with coupling of 3D and 0D models (solid or fluid with 0D lumped circulation systems) such that cardiovascular simulations with realistic boundary conditions can be performed.

Implementations for a recently proposed novel physics- and projection-based model reduc-

tion for FSI, denoted as fluid-reduced-solid interaction (FrSI) [11], are provided, along with POD-based Galerkin model reduction techniques [4] using full or boundary subspaces.

The nonlinear (single- or multi-field) problems are solved with a customized Newton solver with PTC [6] adaptivity in case of divergence, providing robustness for numerically challenging problems. Linear solvers and preconditioners can be chosen from the PETSc repertoire, and specific block preconditioners are made available for coupled problems. Avenues for future functionality include cardiac electrophysiology, scalar transport, or finite strain plasticity.

In the following, a brief description of the supported problem types is given, including the strong and weak form of the underlying equations as well as the discrete assembled systems that are solved.

Examples of input files for the respective problem types can be found in the folder `demos` (with detailed setup descriptions) or amogst the test cases in the folder `tests`.

This documentation is structured as follows. In sec. 2, instructions on how to install and use Ambit are given. The relevant supported physics models are described in sec. 4. Demos are presented in sec. 5.

2 Installation

In order to use Ambit, you need to install FEniCSx (https://github.com/FEniCS/dolfinx#installation) (latest Ambit-compatible dolfinx development version dates to 19 Aug 2023).

Ambit can then be installed using pip, either the current release

```
python3 -m pip install ambit-fe
```

or latest development version:

```
python3 -m pip install git+https://github.com/marchirschvogel/ambit.git
```

Alternatively, you can pull a pre-built Docker image with FEniCSx and Ambit installed:

```
docker pull ghcr.io/marchirschvogel/ambit:latest
```

If a Docker image for development is desired, the following image contains all dependencies needed to install and run Ambit (including the dolfinx mixed branch):

```
docker pull ghcr.io/marchirschvogel/ambit:devenv
```

3 Ambit input

Here, a minimal Ambit input file is shown, exemplarily for a single-field problem. The mandatory parameter dictionaries to provide are input parameters (IO), time parameters (TME), solver parameters (SOL), finite element parameters (FEM), and constitutive/material parameters (MAT). For multi-physics problems, each field needs individual time, finite element, and constitutive parameters.

```
#!/usr/bin/env python3

# Minimal input file for an elastodynamics problem

import ambit_fe
import numpy as np

def main():

    # input/output
    IO = {"problem_type"        : "solid",                    # type of
     ↪  physics to solve
          "mesh_domain"         : "/path/mesh_d.xdmf",        # path to
           ↪  domain mesh
          "mesh_boundary"       : "/path/mesh_b.xdmf",        # path to
           ↪  boundary mesh
          "meshfile_type"       : "HDF5",                     # encoding
           ↪  (HDF5 or ASCII)
          "write_results_every" : 1,                          # step
           ↪  frequency for output
          "output_path"         : "/path/...",                # path to
           ↪  output to
          "results_to_write"    : ["displacement",
                                   "vonmises_cauchystress"], # results to
                                    ↪  output
          "simname"             : "my_results_name"}          # midfix of
           ↪  output name

    # time discretization
    TME = {'maxtime'            : 1.0,        # maximum simulation time
           'dt'                 : 0.01,       # time step size
           'timint'             : 'genalpha', # time integration:
            ↪  Generalized-alpha
           'rho_inf_genalpha'   : 1.0}        # spectral radius of
            ↪  Gen-alpha scheme
```

```
# solver
SOL = {'solve_type'         : 'direct', # direct linear solver
       'tol_res'            : 1.0e-8,   # residual tolerance
       'tol_inc'            : 1.0e-8}   # increment tolerance

# finite element discretization
FEM = {'order_disp'         : 1, # FEM degree for displacement field
       'quad_degree'        : 2} # quadrature scheme degree

# time curves
class TC:
    # user defined load curves, to be used in boundary conditions (BC)
    def tc1(self, t):
        load_max = 5.0
        return load_max * np.sin(t)

# materials
MAT = {'MAT1' : {'neohooke_dev' : {'mu' : 10.0},      # isochoric
↪  NeoHookean material
                 'ogden_vol'    : {'kappa' : 1.0e3},  # volumetric
                 ↪  Ogden material
                 'inertia'      : {'rho0' : 1.0e-6}}} # density

# boundary conditions
BC = {'dirichlet' : [{'id' : [<SURF_IDs>],  # list of surfaces for
↪  Dirichlet BC
                      'dir' : 'all',        # all directions
                      'val' : 0.0}],        # set to zero
      'neumann'   : [{'id' : [<SURF_IDs>],  # list of surfaces for
      ↪  Neumann BC
                      'dir' : 'xyz_ref',    # in cartesian reference
                      ↪  directions
                      'curve' : [1,0,0]}]} # load in x-direction
                      ↪  controlled by curve #1 (see time curves)

# problem setup
problem = ambit.Ambit(io_params=IO, time_params=TP, solver_params=SP,
↪  fem_params=FP, constitutive_params=MAT, boundary_conditions=BC,
↪  time_curves=TC)

# run: solve the problem
problem.solve_problem()
```

```
if __name__ == "__main__":

    main()
```

4 Physics Models

4.1 Solid mechanics

– Example: Sec. 5.1 and `demos/solid`

– `problem_type : "solid"`

– Solid mechanics are formulated in a Total Lagrangian frame

4.1.1 Strong form

Displacement-based

– Primary variable: displacement $\boldsymbol{u}$

$$\begin{aligned} \boldsymbol{\nabla}_0 \cdot \boldsymbol{P}(\boldsymbol{u}, \boldsymbol{v}(\boldsymbol{u})) + \hat{\boldsymbol{b}}_0 &= \rho_0 \boldsymbol{a}(\boldsymbol{u}) && \text{in } \Omega_0 \times [0,T], \\ \boldsymbol{u} &= \hat{\boldsymbol{u}} && \text{on } \Gamma_0^{\mathrm{D}} \times [0,T], \\ \boldsymbol{t}_0 = \boldsymbol{P}\boldsymbol{n}_0 &= \hat{\boldsymbol{t}}_0 && \text{on } \Gamma_0^{\mathrm{N}} \times [0,T], \\ \boldsymbol{u}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{u}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, \\ \boldsymbol{v}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, \end{aligned} \tag{1}$$

Incompressible mechanics

– Primary variables: displacement $\boldsymbol{u}$ and pressure p

$$\begin{aligned} \boldsymbol{\nabla}_0 \cdot \boldsymbol{P}(\boldsymbol{u}, p, \boldsymbol{v}(\boldsymbol{u})) + \hat{\boldsymbol{b}}_0 &= \rho_0 \boldsymbol{a}(\boldsymbol{u}) && \text{in } \Omega_0 \times [0,T], \\ J(\boldsymbol{u}) - 1 &= 0 && \text{in } \Omega_0 \times [0,T], \\ \boldsymbol{u} &= \hat{\boldsymbol{u}} && \text{on } \Gamma_0^{\mathrm{D}} \times [0,T], \\ \boldsymbol{t}_0 = \boldsymbol{P}\boldsymbol{n}_0 &= \hat{\boldsymbol{t}}_0 && \text{on } \Gamma_0^{\mathrm{N}} \times [0,T], \\ \boldsymbol{u}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{u}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, \\ \boldsymbol{v}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, \end{aligned} \tag{2}$$

4.1.2 Weak form

Displacement-based

– Primary variable: displacement $\boldsymbol{u}$

– Principal of Virtual Work:

$$r(\boldsymbol{u};\delta\boldsymbol{u}) := \delta\mathcal{W}_{\text{kin}}(\boldsymbol{u};\delta\boldsymbol{u}) + \delta\mathcal{W}_{\text{int}}(\boldsymbol{u};\delta\boldsymbol{u}) - \delta\mathcal{W}_{\text{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = 0, \quad \forall\, \delta\boldsymbol{u} \tag{3}$$

– Kinetic virtual work:

$$\delta\mathcal{W}_{\text{kin}}(\boldsymbol{u};\delta\boldsymbol{u}) = \int_{\Omega_0} \rho_0\, \boldsymbol{a}(\boldsymbol{u}) \cdot \delta\boldsymbol{u}\, \mathrm{d}V \tag{4}$$

– Internal virtual work:

$$\delta\mathcal{W}_{\text{int}}(\boldsymbol{u};\delta\boldsymbol{u}) = \int_{\Omega_0} \boldsymbol{P}(\boldsymbol{u},\boldsymbol{v}(\boldsymbol{u})) : \boldsymbol{\nabla}_0 \delta\boldsymbol{u}\, \mathrm{d}V = \int_{\Omega_0} \boldsymbol{S}(\boldsymbol{u},\boldsymbol{v}(\boldsymbol{u})) : \frac{1}{2}\delta\boldsymbol{C}(\boldsymbol{u})\, \mathrm{d}V \tag{5}$$

– External virtual work:

- conservative Neumann load:

$$\delta\mathcal{W}_{\text{ext}}(\delta\boldsymbol{u}) = \int_{\Gamma_0^{\text{N}}} \hat{\boldsymbol{t}}_0(t) \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{6}$$

- Neumann pressure load in current normal direction:

$$\delta\mathcal{W}_{\text{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = -\int_{\Gamma_0^{\text{N}}} \hat{p}(t)\, J\boldsymbol{F}^{-\text{T}}\boldsymbol{n}_0 \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{7}$$

- general Neumann load in current direction:

$$\delta\mathcal{W}_{\text{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = \int_{\Gamma_0} J\boldsymbol{F}^{-\text{T}}\, \hat{\boldsymbol{t}}_0(t) \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{8}$$

- body force:

$$\delta\mathcal{W}_{\text{ext}}(\delta\boldsymbol{u}) = \int_{\Omega_0} \hat{\boldsymbol{b}}_0(t) \cdot \delta\boldsymbol{u}\, \mathrm{d}V \tag{9}$$

- generalized Robin condition:

$$\delta\mathcal{W}_{\text{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = -\int_{\Gamma_0^{\text{N}}} [k\,\boldsymbol{u} + c\,\boldsymbol{v}(\boldsymbol{u})] \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{10}$$

- generalized Robin condition in reference surface normal direction:

$$\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = -\int\limits_{\Gamma_0^{\mathrm{N}}} (\boldsymbol{n}_0 \otimes \boldsymbol{n}_0)\,[k\,\boldsymbol{u} + c\,\boldsymbol{v}(\boldsymbol{u})] \cdot \delta\boldsymbol{u}\,\mathrm{d}A \tag{11}$$

Incompressible mechanics: 2-field displacement and pressure variables

– Primary variables: displacement $\boldsymbol{u}$ and pressure p

$$\begin{aligned} r_u(\boldsymbol{u},p;\delta\boldsymbol{u}) &:= \delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{u};\delta\boldsymbol{u}) + \delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u},p;\delta\boldsymbol{u}) - \delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u};\delta\boldsymbol{u}) = 0, \quad \forall\,\delta\boldsymbol{u} \\ r_p(\boldsymbol{u};\delta p) &:= \delta\mathcal{W}_{\mathrm{pres}}(\boldsymbol{u};\delta p) = 0, \quad \forall\,\delta p \end{aligned} \tag{12}$$

– Kinetic virtual work: (4)
– Internal virtual work:

$$\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u},p;\delta\boldsymbol{u}) = \int\limits_{\Omega_0} \boldsymbol{P}(\boldsymbol{u},p,\boldsymbol{v}(\boldsymbol{u})) : \boldsymbol{\nabla}_0\delta\boldsymbol{u}\,\mathrm{d}V = \int\limits_{\Omega_0} \boldsymbol{S}(\boldsymbol{u},p,\boldsymbol{v}(\boldsymbol{u})) : \frac{1}{2}\delta\boldsymbol{C}(\boldsymbol{u})\,\mathrm{d}V \tag{13}$$

– Pressure virtual work:

$$\delta\mathcal{W}_{\mathrm{pres}}(\boldsymbol{u};\delta p) = \int\limits_{\Omega_0} (J(\boldsymbol{u}) - 1)\,\delta p\,\mathrm{d}V \tag{14}$$

Time integration

– time scheme `timint : "static"`

$$\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u}) - \delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u}) = 0, \quad \forall\,\delta\boldsymbol{u} \tag{15}$$

– Generalized-alpha time scheme `timint : "genalpha"`

$$\begin{aligned} \boldsymbol{v}_{n+1} &= \frac{\gamma}{\beta\Delta t}(\boldsymbol{u}_{n+1} - \boldsymbol{u}_n) - \frac{\gamma-\beta}{\beta}\boldsymbol{v}_n - \frac{\gamma-2\beta}{2\beta}\Delta t\,\boldsymbol{a}_n \\ \boldsymbol{a}_{n+1} &= \frac{1}{\beta\Delta t^2}(\boldsymbol{u}_{n+1} - \boldsymbol{u}_n) - \frac{1}{\beta\Delta t}\boldsymbol{v}_n - \frac{1-2\beta}{2\beta}\boldsymbol{a}_n \end{aligned} \tag{16}$$

- option `eval_nonlin_terms : "midpoint"`:

$$\begin{aligned} \boldsymbol{u}_{n+1-\alpha_{\mathrm{f}}} &= (1-\alpha_{\mathrm{f}})\boldsymbol{u}_{n+1} + \alpha_{\mathrm{f}}\boldsymbol{v}_n \\ \boldsymbol{v}_{n+1-\alpha_{\mathrm{f}}} &= (1-\alpha_{\mathrm{f}})\boldsymbol{v}_{n+1} + \alpha_{\mathrm{f}}\boldsymbol{v}_n \\ \boldsymbol{a}_{n+1-\alpha_{\mathrm{m}}} &= (1-\alpha_{\mathrm{m}})\boldsymbol{a}_{n+1} + \alpha_{\mathrm{m}}\boldsymbol{a}_n \end{aligned} \tag{17}$$

$$\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n+1-\alpha_m};\delta\boldsymbol{u}) + \delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n+1-\alpha_f};\delta\boldsymbol{u}) - \delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n+1-\alpha_f};\delta\boldsymbol{u}) = 0, \quad \forall\,\delta\boldsymbol{u} \tag{18}$$

- option `eval_nonlin_terms : "trapezoidal"`:

$$\begin{aligned}&(1-\alpha_{\mathrm{m}})\,\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n+1};\delta\boldsymbol{u})+\alpha_{\mathrm{m}}\,\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n};\delta\boldsymbol{u})+\\&(1-\alpha_{\mathrm{f}})\,\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u})+\alpha_{\mathrm{f}}\,\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n};\delta\boldsymbol{u})-\\&(1-\alpha_{f})\,\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u})-\alpha_{\mathrm{f}}\,\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n};\delta\boldsymbol{u})=0,\quad\forall\,\delta\boldsymbol{u}\end{aligned}\tag{19}$$

– One-Step-theta time scheme `timint : "ost"`

$$\begin{aligned}\boldsymbol{v}_{n+1}&=\frac{1}{\theta\Delta t}(\boldsymbol{u}_{n+1}-\boldsymbol{u}_{n})-\frac{1-\theta}{\theta}\boldsymbol{v}_{n}\\\boldsymbol{a}_{n+1}&=\frac{1}{\theta^{2}\Delta t^{2}}(\boldsymbol{u}_{n+1}-\boldsymbol{u}_{n})-\frac{1}{\theta^{2}\Delta t}\boldsymbol{v}_{n}-\frac{1-\theta}{\theta}\boldsymbol{a}_{n}\end{aligned}\tag{20}$$

- option `eval_nonlin_terms : "midpoint"`:

$$\begin{aligned}\boldsymbol{u}_{n+\theta}&=\theta\boldsymbol{u}_{n+1}+(1-\theta)\boldsymbol{u}_{n}\\\boldsymbol{v}_{n+\theta}&=\theta\boldsymbol{v}_{n+1}+(1-\theta)\boldsymbol{v}_{n}\\\boldsymbol{a}_{n+\theta}&=\theta\boldsymbol{a}_{n+1}+(1-\theta)\boldsymbol{a}_{n}\end{aligned}\tag{21}$$

$$\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n+\theta};\delta\boldsymbol{u})+\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n+\theta};\delta\boldsymbol{u})-\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n+\theta};\delta\boldsymbol{u})=0,\quad\forall\,\delta\boldsymbol{u}\tag{22}$$

- option `eval_nonlin_terms : "trapezoidal"`:

$$\begin{aligned}&\theta\,\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n+1};\delta\boldsymbol{u})+(1-\theta)\,\delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{a}_{n};\delta\boldsymbol{u})+\\&\theta\,\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u})+(1-\theta)\,\delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}_{n};\delta\boldsymbol{u})-\\&\theta\,\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n+1};\delta\boldsymbol{u})-(1-\theta)\,\delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}_{n};\delta\boldsymbol{u})=0,\quad\forall\,\delta\boldsymbol{u}\end{aligned}\tag{23}$$

Note the equivalence of `"midpoint"` and `"trapezoidal"` for all linear terms, e.g. $\delta\mathcal{W}_{\mathrm{kin}}$, or for no or only linear dependence of $\delta\mathcal{W}_{\mathrm{ext}}$ on the solution.

Note that, for incompressible mechanics, the pressure kinematic constraint is always evaluated at t_{n+1}:

$$\delta\mathcal{W}_{\mathrm{pres}}(\boldsymbol{u}_{n+1};\delta p)=0,\quad\forall\,\delta p,\tag{24}$$

and the pressure in $\delta\mathcal{W}_{\mathrm{int}}$ is set according to (17) or (21), respectively.

Spatial discretization and solution

– Discrete nonlinear system to solve in each time step n (displacement-based):

$$\mathbf{r}_{u}(\mathbf{u})\big|_{n+1}=\mathbf{0}\tag{25}$$

– Discrete linear system to solve in each Newton iteration k (displacement-based):

$$\mathbf{K}_{uu}\big|_{n+1}^{k}\,\Delta\mathbf{u}_{n+1}^{k+1}=-\,\mathbf{r}_{u}\big|_{n+1}^{k}\tag{26}$$

– Discrete nonlinear system to solve in each time step n (incompressible):

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u(\mathbf{u}, \mathbf{p}) \\ \mathbf{r}_p(\mathbf{u}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{27}$$

– Discrete linear system to solve in each Newton iteration k (incompressible):

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{0} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{p} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_p \end{bmatrix}_{n+1}^{k} \tag{28}$$

4.2 Fluid mechanics

4.2.1 Eulerian reference frame

– Example: Sec. 5.2 and `demos/fluid`

– Problem type: `fluid`

– Incompressible Navier-Stokes equations in Eulerian reference frame

Strong Form

– Primary variables: velocity $\boldsymbol{v}$ and pressure p

$$\begin{aligned} \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p) + \hat{\boldsymbol{b}} &= \rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla} \boldsymbol{v})\, \boldsymbol{v} \right) && \text{in } \Omega_t \times [0, T], \\ \boldsymbol{\nabla} \cdot \boldsymbol{v} &= 0 && \text{in } \Omega_t \times [0, T], \\ \boldsymbol{v} &= \hat{\boldsymbol{v}} && \text{on } \Gamma_t^{\mathrm{D}} \times [0, T], \\ \boldsymbol{t} = \boldsymbol{\sigma} \boldsymbol{n} &= \hat{\boldsymbol{t}} && \text{on } \Gamma_t^{\mathrm{N}} \times [0, T], \\ \boldsymbol{v}(\boldsymbol{x}, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}) && \text{in } \Omega_t, \end{aligned} \tag{29}$$

with a Newtonian fluid constitutive law

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\,\boldsymbol{\gamma} = -p\boldsymbol{I} + \mu \left(\boldsymbol{\nabla} \boldsymbol{v} + (\boldsymbol{\nabla} \boldsymbol{v})^{\mathrm{T}} \right) \tag{30}$$

Weak Form

– Primary variables: velocity $\boldsymbol{v}$ and pressure p

– Principle of Virtual Power

$$\begin{aligned} r_v(\boldsymbol{v}, p; \delta\boldsymbol{v}) &:= \delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v}; \delta\boldsymbol{v}) + \delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v}, p; \delta\boldsymbol{v}) - \delta\mathcal{P}_{\mathrm{ext}}(\boldsymbol{v}; \delta\boldsymbol{v}) = 0, \quad \forall\, \delta\boldsymbol{v} \\ r_p(\boldsymbol{v}; \delta p) &:= \delta\mathcal{P}_{\mathrm{pres}}(\boldsymbol{v}; \delta p), \quad \forall\, \delta p \end{aligned} \tag{31}$$

– Kinetic virtual power:

$$\delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v};\delta\boldsymbol{v}) = \int\limits_{\Omega_t} \rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}\boldsymbol{v})\,\boldsymbol{v}\right)\cdot\delta\boldsymbol{v}\,\mathrm{d}v \tag{32}$$

– Internal virtual power:

$$\delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v},p;\delta\boldsymbol{v}) = \int\limits_{\Omega_t} \boldsymbol{\sigma}(\boldsymbol{v},p) : \boldsymbol{\nabla}\delta\boldsymbol{v}\,\mathrm{d}v \tag{33}$$

– Pressure virtual power:

$$\delta\mathcal{P}_{\mathrm{pres}}(\boldsymbol{v};\delta p) = \int\limits_{\Omega_t} (\boldsymbol{\nabla}\cdot\boldsymbol{v})\,\delta p\,\mathrm{d}v \tag{34}$$

– External virtual power:

- conservative Neumann load:

$$\delta\mathcal{P}_{\mathrm{ext}}(\delta\boldsymbol{v}) = \int\limits_{\Gamma_t^{\mathrm{N}}} \hat{\boldsymbol{t}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}a \tag{35}$$

- pressure Neumann load:

$$\delta\mathcal{P}_{\mathrm{ext}}(\delta\boldsymbol{v}) = -\int\limits_{\Gamma_t^{\mathrm{N}}} \hat{p}(t)\,\boldsymbol{n}\cdot\delta\boldsymbol{v}\,\mathrm{d}a \tag{36}$$

- body force:

$$\delta\mathcal{P}_{\mathrm{ext}}(\delta\boldsymbol{v}) = \int\limits_{\Omega_t} \hat{\boldsymbol{b}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{37}$$

Stabilization

Streamline-upwind Petrov-Galerkin/pressure-stabilizing Petrov-Galerkin (SUPG/PSPG) methods are implemented, either using the full or a reduced scheme

– to the fluid FEM params, add the dict entry:

```
"stabilization" : {"scheme" : <SCHEME>, "vscale" : 1e3, "dscales" :
↪  [<d1>,<d2>,<d3>],
                   "symmetric" : False}
```

Full scheme according to [18]: `"supg_pspg"`:
– Velocity residual operator in (31) is augmented by the following terms:

$$\begin{aligned} r_v \leftarrow r_v &+ \frac{1}{\rho} \int\limits_{\Omega_t} \tau_{\mathrm{SUPG}} \left(\boldsymbol{\nabla}\delta\boldsymbol{v}\right) \boldsymbol{v} \cdot \left[\rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + \left(\boldsymbol{\nabla}\boldsymbol{v}\right) \boldsymbol{v}\right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p)\right] \mathrm{d}v \\ &+ \int\limits_{\Omega_t} \tau_{\mathrm{LSIC}}\, \rho \left(\boldsymbol{\nabla} \cdot \delta\boldsymbol{v}\right)\left(\boldsymbol{\nabla} \cdot \boldsymbol{v}\right) \mathrm{d}v \end{aligned} \tag{38}$$

– Pressure residual operator in (31) is augmented by the following terms:

$$r_p \leftarrow r_p + \frac{1}{\rho} \int\limits_{\Omega_t} \tau_{\mathrm{PSPG}} \left(\boldsymbol{\nabla}\delta p\right) \cdot \left[\rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + \left(\boldsymbol{\nabla}\boldsymbol{v}\right) \boldsymbol{v}\right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p)\right] \mathrm{d}v \tag{39}$$

Reduced scheme (optimized for first-order): `"supg_pspg2"`:
– Velocity residual operator in (31) is augmented by the following terms:

$$\begin{aligned} r_v \leftarrow r_v &+ \int\limits_{\Omega_t} d_1 \left(\left(\boldsymbol{\nabla}\boldsymbol{v}\right) \boldsymbol{v}\right) \cdot \left(\boldsymbol{\nabla}\delta\boldsymbol{v}\right) \boldsymbol{v}\, \mathrm{d}v \\ &+ \int\limits_{\Omega_t} d_2 \left(\boldsymbol{\nabla} \cdot \boldsymbol{v}\right)\left(\boldsymbol{\nabla} \cdot \delta\boldsymbol{v}\right) \mathrm{d}v \\ &+ \int\limits_{\Omega_t} d_3 \left(\boldsymbol{\nabla} p\right) \cdot \left(\boldsymbol{\nabla}\delta\boldsymbol{v}\right) \boldsymbol{v}\, \mathrm{d}v \end{aligned} \tag{40}$$

– Pressure residual operator in (31) is augmented by the following terms:

$$\begin{aligned} r_p \leftarrow r_p &+ \frac{1}{\rho} \int\limits_{\Omega_t} d_1 \left(\left(\boldsymbol{\nabla}\boldsymbol{v}\right) \boldsymbol{v}\right) \cdot \left(\boldsymbol{\nabla}\delta p\right) \mathrm{d}v \\ &+ \frac{1}{\rho} \int\limits_{\Omega_t} d_3 \left(\boldsymbol{\nabla} p\right) \cdot \left(\boldsymbol{\nabla}\delta p\right) \mathrm{d}v \end{aligned} \tag{41}$$

– Discrete nonlinear system to solve in each time step n:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{42}$$

– Discrete linear system to solve in each Newton iteration k:

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \end{bmatrix}_{n+1}^{k} \tag{43}$$

– Note that $\mathbf{K}_{pp}$ is zero for Taylor-Hood elements (without stabilization)

4.2.2 ALE reference frame

– Problem type: `fluid_ale`

– Incompressible Navier-Stokes equations in Arbitrary Lagrangian Eulerian (ALE) reference frame

– ALE domain problem deformation governed by linear-elastic, nonlinear hyperelastic solid, or a diffusion problem, displacement field $\boldsymbol{d}$

– Fluid mechanics formulated with respect to the reference frame, using ALE deformation gradient $\widetilde{\boldsymbol{F}}(\boldsymbol{d}) = \boldsymbol{I} + \boldsymbol{\nabla}_0 \boldsymbol{d}$ and its determinant, $\widetilde{J}(\boldsymbol{d}) = \det \widetilde{\boldsymbol{F}}(\boldsymbol{d})$

ALE problem

– Primary variable: domain displacement $\boldsymbol{d}$

– Strong form:

$$\begin{aligned} \boldsymbol{\nabla}_0 \cdot \boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) &= \boldsymbol{0} \quad \text{in } \Omega_0, \\ \boldsymbol{d} &= \hat{\boldsymbol{d}} \quad \text{on } \Gamma_0^{\mathrm{D}}, \end{aligned} \tag{44}$$

– ALE material `linelast`:

$$\boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) = 2\mu\,\boldsymbol{\varepsilon} + \lambda\,\mathrm{tr}\boldsymbol{\varepsilon}\,\boldsymbol{I}, \qquad \text{with} \quad \boldsymbol{\varepsilon} = \frac{1}{2}\left(\boldsymbol{\nabla}_0 \boldsymbol{d} + (\boldsymbol{\nabla}_0 \boldsymbol{d})^{\mathrm{T}}\right) \tag{45}$$

– ALE material `diffusion`:

$$\boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) = D\,\boldsymbol{\nabla}_0 \boldsymbol{d} \tag{46}$$

– ALE material `neohooke` (fully nonlinear model):

$$\boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) = \frac{\partial \Psi}{\partial \widetilde{\boldsymbol{F}}}, \qquad \text{with} \quad \Psi = \frac{\mu}{2}\left(\mathrm{tr}(\widetilde{\boldsymbol{F}}^{\mathrm{T}}\widetilde{\boldsymbol{F}}) - 3\right) + \frac{\mu}{2\beta}\left(\widetilde{J}^{-2\beta} - 1\right) \tag{47}$$

– weak form:

$$r_d(\boldsymbol{d}; \delta\boldsymbol{d}) := \int_{\Omega_0} \boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) : \boldsymbol{\nabla}_0 \delta\boldsymbol{d}\,\mathrm{d}V = 0, \quad \forall\, \delta\boldsymbol{d} \tag{48}$$

Strong form (ALE)

– Primary variables: velocity $\boldsymbol{v}$, pressure p, and domain displacement $\boldsymbol{d}$

$$\begin{aligned} \boldsymbol{\nabla}_0 \boldsymbol{\sigma}(\boldsymbol{v}, \boldsymbol{d}, p) : \widetilde{\boldsymbol{F}}^{-\mathrm{T}} + \hat{\boldsymbol{b}} &= \rho\left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0 \boldsymbol{v}\,\widetilde{\boldsymbol{F}}^{-1})\,\boldsymbol{v}\right) && \text{in } \Omega_0 \times [0,T], \\ \boldsymbol{\nabla}_0 \boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}} &= 0 && \text{in } \Omega_0 \times [0,T], \\ \boldsymbol{v} &= \hat{\boldsymbol{v}} && \text{on } \Gamma_0^{\mathrm{D}} \times [0,T], \\ \boldsymbol{t} = \boldsymbol{\sigma}\boldsymbol{n} &= \hat{\boldsymbol{t}} && \text{on } \Gamma_0^{\mathrm{N}} \times [0,T], \\ \boldsymbol{v}(\boldsymbol{x}, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}) && \text{in } \Omega_0, \end{aligned} \tag{49}$$

with a Newtonian fluid constitutive law

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{\gamma} = -p\boldsymbol{I} + \mu\left(\boldsymbol{\nabla}_0\boldsymbol{v}\,\widetilde{\boldsymbol{F}}^{-1} + \widetilde{\boldsymbol{F}}^{-\mathrm{T}}(\boldsymbol{\nabla}_0\boldsymbol{v})^{\mathrm{T}}\right) \tag{50}$$

Weak form (ALE)

– Primary variables: velocity $\boldsymbol{v}$, pressure p, and domain displacement $\boldsymbol{d}$

– Principle of Virtual Power

$$\begin{aligned} r_v(\boldsymbol{v}, p, \boldsymbol{d}; \delta\boldsymbol{v}) &:= \delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v}, \boldsymbol{d}; \delta\boldsymbol{v}) + \delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v}, p, \boldsymbol{d}; \delta\boldsymbol{v}) - \delta\mathcal{P}_{\mathrm{ext}}(\boldsymbol{v}, \boldsymbol{d}; \delta\boldsymbol{v}) = 0, \quad \forall\, \delta\boldsymbol{v} \\ r_p(\boldsymbol{v}, \boldsymbol{d}; \delta p) &:= \delta\mathcal{P}_{\mathrm{pres}}(\boldsymbol{v}, \boldsymbol{d}; \delta p), \quad \forall\, \delta p \end{aligned} \tag{51}$$

– Kinetic virtual power:

$$\delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v}, \boldsymbol{d}; \delta\boldsymbol{v}) = \int_{\Omega_0} \widetilde{J}\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0\boldsymbol{v}\,\widetilde{\boldsymbol{F}}^{-1})\,\boldsymbol{v}\right)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{52}$$

– Internal virtual power:

$$\delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v}, p, \boldsymbol{d}; \delta\boldsymbol{v}) = \int_{\Omega_0} \widetilde{J}\boldsymbol{\sigma}(\boldsymbol{v}, p, \boldsymbol{d}) : \boldsymbol{\nabla}_0\delta\boldsymbol{v}\,\widetilde{\boldsymbol{F}}^{-1}\,\mathrm{d}V \tag{53}$$

– Pressure virtual power:

$$\delta\mathcal{P}_{\mathrm{pres}}(\boldsymbol{v}, \boldsymbol{d}; \delta p) = \int_{\Omega_0} \widetilde{J}\,\boldsymbol{\nabla}_0\boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}}\delta p\,\mathrm{d}V \tag{54}$$

– External virtual power:

- conservative Neumann load:

$$\delta\mathcal{P}_{\mathrm{ext}}(\delta\boldsymbol{v}) = \int_{\Gamma_0^{\mathrm{N}}} \hat{\boldsymbol{t}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}A \tag{55}$$

- pressure Neumann load:

$$\delta\mathcal{P}_{\mathrm{ext}}(\boldsymbol{d}; \delta\boldsymbol{v}) = -\int_{\Gamma_0^{\mathrm{N}}} \hat{p}(t)\,\widetilde{J}\widetilde{\boldsymbol{F}}^{-\mathrm{T}}\boldsymbol{n}_0\cdot\delta\boldsymbol{v}\,\mathrm{d}A \tag{56}$$

- body force:

$$\delta\mathcal{P}_{\mathrm{ext}}(\boldsymbol{d}; \delta\boldsymbol{v}) = \int_{\Omega_0} \widetilde{J}\,\hat{\boldsymbol{b}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{57}$$

Stabilization (ALE)

`"supg_pspg"`:
– Velocity residual operator in (51) is augmented by the following terms:

$$\begin{aligned} r_v \leftarrow r_v + \frac{1}{\rho} \int\limits_{\Omega_0} & \widetilde{J}\, \tau_{\mathrm{SUPG}}\, (\boldsymbol{\nabla}_0 \delta\boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, \boldsymbol{v} \cdot \\ & \cdot \left[\rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0 \boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, (\boldsymbol{v} - \boldsymbol{w}) \right) - \boldsymbol{\nabla}_0 \boldsymbol{\sigma}(\boldsymbol{v}, \boldsymbol{d}, p) : \widetilde{\boldsymbol{F}}^{-\mathrm{T}} \right] \mathrm{d}V \\ + \int\limits_{\Omega_0} & \widetilde{J}\, \tau_{\mathrm{LSIC}}\, \rho\, (\boldsymbol{\nabla}_0 \delta\boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}}) (\boldsymbol{\nabla}_0 \boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}})\, \mathrm{d}V \end{aligned} \tag{58}$$

– Pressure residual operator in (51) is augmented by the following terms:

$$\begin{aligned} r_p \leftarrow r_p + \frac{1}{\rho} \int\limits_{\Omega_0} & \widetilde{J}\, \tau_{\mathrm{PSPG}}\, (\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{\nabla}_0 \delta p) \cdot \\ & \cdot \left[\rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0 \boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, (\boldsymbol{v} - \boldsymbol{w}) \right) - \boldsymbol{\nabla}_0 \boldsymbol{\sigma}(\boldsymbol{v}, \boldsymbol{d}, p) : \widetilde{\boldsymbol{F}}^{-\mathrm{T}} \right] \mathrm{d}V \end{aligned} \tag{59}$$

`"supg_pspg2"`:
– Velocity residual operator in (51) is augmented by the following terms:

$$\begin{aligned} r_v \leftarrow r_v + & \int\limits_{\Omega_0} \widetilde{J}\, d_1\, ((\boldsymbol{\nabla}_0 \boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, (\boldsymbol{v} - \boldsymbol{w})) \cdot (\boldsymbol{\nabla}_0 \delta\boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, \boldsymbol{v}\, \mathrm{d}V \\ & + \int\limits_{\Omega_0} \widetilde{J}\, d_2\, (\boldsymbol{\nabla}_0 \boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}}) (\boldsymbol{\nabla}_0 \delta\boldsymbol{v} : \widetilde{\boldsymbol{F}}^{-\mathrm{T}})\, \mathrm{d}V \\ & + \int\limits_{\Omega_0} \widetilde{J}\, d_3\, (\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{\nabla}_0 p) \cdot (\boldsymbol{\nabla}_0 \delta\boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, \boldsymbol{v}\, \mathrm{d}V \end{aligned} \tag{60}$$

– Pressure residual operator in (51) is augmented by the following terms:

$$\begin{aligned} r_p \leftarrow r_p + \frac{1}{\rho} & \int\limits_{\Omega_0} \widetilde{J}\, d_1\, ((\boldsymbol{\nabla}_0 \boldsymbol{v}\, \widetilde{\boldsymbol{F}}^{-1})\, (\boldsymbol{v} - \boldsymbol{w})) \cdot (\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{\nabla}_0 \delta p)\, \mathrm{d}V \\ & + \frac{1}{\rho} \int\limits_{\Omega_0} \widetilde{J}\, d_3\, (\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{\nabla}_0 p) \cdot (\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{\nabla}_0 \delta p)\, \mathrm{d}V \end{aligned} \tag{61}$$

– Discrete nonlinear system to solve in each time step n:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{d}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{62}$$

– Discrete linear system to solve in each Newton iteration k:

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{vd} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{K}_{pd} \\ \mathbf{K}_{dv} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}^{k}_{n+1} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{d} \end{bmatrix}^{k+1}_{n+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_d \end{bmatrix}^{k}_{n+1} \tag{63}$$

– note that $\mathbf{K}_{pp}$ is zero for Taylor-Hood elements (without stabilization)

4.3 0D flow: Lumped parameter models

– Example: Sec. 5.3 and `demos/flow0d`

– Problem type: `flow0d`

– 0D model concentrated elements are resistances (R), impedances (Z, technically are resistances), compliances (C), inertances (L or I), and elastances (E)

– 0D variables are pressures (p), fluxes (q or Q), or volumes (V)

4.3.1 2-element Windkessel

– Model type : `2elwindkessel`

4.3.2 4-element Windkessel (inertance parallel to impedance)

– Model type : `4elwindkesselLpZ`

4.3.3 4-element Windkessel (inertance serial to impedance)

– Model type : `4elwindkesselLsZ`

4.3.4 In-outflow CRL link

– Model type : `CRLinoutlink`

4.3.5 Systemic and pulmonary circulation

– Model type : `syspul`

– Allows to link in a coronary flow model

left heart and systemic circulation

$$-Q^{\ell}_{\mathrm{at}} = \sum_{i=1}^{n^{\mathrm{pul}}_{\mathrm{ven}}} q^{\mathrm{pul}}_{\mathrm{ven},i} - q^{\ell}_{\mathrm{v,in}} \qquad \text{left atrium flow balance}$$

$$q^{\ell}_{\mathrm{v,in}} = q_{\mathrm{mv}}(p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}}) \qquad \text{mitral valve momentum}$$

$$-Q^{\ell}_{\mathrm{v}} = q^{\ell}_{\mathrm{v,in}} - q^{\ell}_{\mathrm{v,out}} \qquad \text{left ventricle flow balance}$$

$$q^{\ell}_{\mathrm{v,out}} = q_{\mathrm{av}}(p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}}) \qquad \text{aortic valve momentum}$$

$$-Q^{\mathrm{sys}}_{\mathrm{aort}} = q^{\ell}_{\mathrm{v,out}} - q^{\mathrm{sys}}_{\mathrm{ar,p}} - \mathbb{I}^{\mathrm{cor}} \sum_{i=1}^{2} q^{\mathrm{sys}}_{\mathrm{ar,cor,in},i} \qquad \text{aortic root flow balance}$$

$$I^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar,p}}}{\mathrm{d}t} + Z^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar,p}} = p^{\mathrm{sys}}_{\mathrm{ar}} - p^{\mathrm{sys}}_{\mathrm{ar,d}} \qquad \text{aortic root inertia}$$

$$C^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar,d}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar,p}} - q^{\mathrm{sys}}_{\mathrm{ar}} \qquad \text{systemic arterial flow balance}$$

$$L^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar}} = p^{\mathrm{sys}}_{\mathrm{ar,d}} - p^{\mathrm{sys}}_{\mathrm{ven}} \qquad \text{systemic arterial momentum}$$

$$C^{\mathrm{sys}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar}} - \sum_{i=1}^{n^{\mathrm{sys}}_{\mathrm{ven}}} q^{\mathrm{sys}}_{\mathrm{ven},i} \qquad \text{systemic venous flow balance}$$

$$L^{\mathrm{sys}}_{\mathrm{ven},i} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ven},i}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ven},i}\, q^{\mathrm{sys}}_{\mathrm{ven},i} = p^{\mathrm{sys}}_{\mathrm{ven}} - p^{r}_{\mathrm{at},i} \qquad \text{systemic venous momentum}$$

$$i \in \{1, ..., n^{\mathrm{sys}}_{\mathrm{ven}}\}$$

right heart and pulmonary circulation

$$-Q^r_{\mathrm{at}} = \sum_{i=1}^{n^{\mathrm{sys}}_{\mathrm{ven}}} q^{\mathrm{sys}}_{\mathrm{ven},i} - \mathbb{I}^{\mathrm{cor}} q^{\mathrm{sys}}_{\mathrm{ven,cor,out}} - q^r_{\mathrm{v,in}} \quad \text{right atrium flow balance}$$

$$q^r_{\mathrm{v,in}} = q_{\mathrm{tv}}(p^r_{\mathrm{at}} - p^r_{\mathrm{v}}) \quad \text{tricuspid valve momentum}$$

$$-Q^r_{\mathrm{v}} = q^r_{\mathrm{v,in}} - q^r_{\mathrm{v,out}} \quad \text{right ventricle flow balance}$$

$$q^r_{\mathrm{v,out}} = q_{\mathrm{pv}}(p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}}) \quad \text{pulmonary valve momentum}$$

$$C^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} = q^r_{\mathrm{v,out}} - q^{\mathrm{pul}}_{\mathrm{ar}} \quad \text{pulmonary arterial flow balance}$$

$$L^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ar}}\, q^{\mathrm{pul}}_{\mathrm{ar}} = p^{\mathrm{pul}}_{\mathrm{ar}} - p^{\mathrm{pul}}_{\mathrm{ven}} \quad \text{pulmonary arterial momentum}$$

$$C^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} = q^{\mathrm{pul}}_{\mathrm{ar}} - \sum_{i=1}^{n^{\mathrm{pul}}_{\mathrm{ven}}} q^{\mathrm{pul}}_{\mathrm{ven},i} \quad \text{pulmonary venous flow balance}$$

$$L^{\mathrm{pul}}_{\mathrm{ven},i} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ven},i}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ven},i}\, q^{\mathrm{pul}}_{\mathrm{ven},i} = p^{\mathrm{pul}}_{\mathrm{ven}} - p^{\ell}_{\mathrm{at},i} \quad \text{pulmonary venous momentum}$$

$$i \in \{1, ..., n^{\mathrm{pul}}_{\mathrm{ven}}\}$$

with:

$$Q^{\ell}_{\mathrm{at}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{at}}}{\mathrm{d}t}, \quad Q^{\ell}_{\mathrm{v}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{v}}}{\mathrm{d}t}, \quad Q^{r}_{\mathrm{at}} := -\frac{\mathrm{d}V^{r}_{\mathrm{at}}}{\mathrm{d}t}, \quad Q^{r}_{\mathrm{v}} := -\frac{\mathrm{d}V^{r}_{\mathrm{v}}}{\mathrm{d}t}, \quad Q^{\mathrm{sys}}_{\mathrm{aort}} := -\frac{\mathrm{d}V^{\mathrm{sys}}_{\mathrm{aort}}}{\mathrm{d}t} \tag{64}$$

and:

$$\mathbb{I}^{\mathrm{cor}} = \begin{cases} 1, & \text{if CORONARYMODEL,} \\ 0, & \text{else} \end{cases} \tag{65}$$

The volume V of the heart chambers (0D) is modeled by the volume-pressure relationship

$$V(t) = \frac{p}{E(t)} + V_{\mathrm{u}}, \tag{66}$$

with the unstressed volume V_{u} and the time-varying elastance

$$E(t) = (E_{\max} - E_{\min}) \cdot \hat{y}(t) + E_{\min}, \tag{67}$$

where $E_{\max}$ and $E_{\min}$ denote the maximum and minimum elastance, respectively. The normalized activation function $\hat{y}(t)$ is input by the user.
Flow-pressure relations for the four valves, eq. (**??**), (**??**), (**??**), (**??**), are functions of the pressure difference $p - p_{\mathrm{open}}$ across the valve. The following valve models can be defined:
Valve model `pwlin_pres`:

$$q(p - p_{\mathrm{open}}) = \frac{p - p_{\mathrm{open}}}{\tilde{R}}, \quad \text{with } \tilde{R} = \begin{cases} R_{\max}, & p < p_{\mathrm{open}} \\ R_{\min}, & p \geq p_{\mathrm{open}} \end{cases} \tag{68}$$

Remark: Non-smooth flow-pressure relationship
Valve model `pwlin_time`:

$$q(p-p_{\text{open}}) = \frac{p-p_{\text{open}}}{\tilde{R}}, \quad \text{with } \tilde{R} = \begin{cases} \begin{cases} R_{\max}, & t < t_{\text{open}} \text{ and } t \geq t_{\text{close}} \\ R_{\min}, & t \geq t_{\text{open}} \text{ or } t < t_{\text{close}} \end{cases}, & t_{\text{open}} > t_{\text{close}} \\ \begin{cases} R_{\max}, & t < t_{\text{open}} \text{ or } t \geq t_{\text{close}} \\ R_{\min}, & t \geq t_{\text{open}} \text{ and } t < t_{\text{close}} \end{cases}, & \text{else} \end{cases} \tag{69}$$

Remark: Non-smooth flow-pressure relationship with resistance only dependent on timings, not the pressure difference!
Valve model `smooth_pres_resistance`:

$$q(p-p_{\text{open}}) = \frac{p-p_{\text{open}}}{\tilde{R}}, \quad \text{with } \tilde{R} = 0.5\,(R_{\max}-R_{\min})\left(\tanh\frac{p-p_{\text{open}}}{\epsilon}+1\right)+R_{\min} \tag{70}$$

Remark: Smooth but potentially non-convex flow-pressure relationship!
Valve model `smooth_pres_momentum`:

$$q(p-p_{\text{open}}) = \begin{cases} \frac{p-p_{\text{open}}}{R_{\max}}, & p < p_{\text{open}} - 0.5\epsilon \\ h_{00}p_0 + h_{10}m_0\epsilon + h_{01}p_1 + h_{11}m_1\epsilon, & p \geq p_{\text{open}} - 0.5\epsilon \text{ and } p < p_{\text{open}} + 0.5\epsilon \\ \frac{p-p_{\text{open}}}{R_{\min}}, & p \geq p_{\text{open}} + 0.5\epsilon \end{cases} \tag{71}$$

with

$$p_0 = \frac{-0.5\epsilon}{R_{\max}}, \qquad m_0 = \frac{1}{R_{\max}}, \qquad p_1 = \frac{0.5\epsilon}{R_{\min}}, \qquad m_1 = \frac{1}{R_{\min}} \tag{72}$$

and

$$\begin{aligned} h_{00} &= 2s^3 - 3s^2 + 1, & h_{01} &= -2s^3 + 3s^2, \\ h_{10} &= s^3 - 2s^2 + s, & h_{11} &= s^3 - s^2 \end{aligned} \tag{73}$$

with

$$s = \frac{p - p_{\text{open}} + 0.5\epsilon}{\epsilon} \tag{74}$$

Remarks:
– Collapses to valve model `pwlin_pres` for $\epsilon = 0$
– Smooth and convex flow-pressure relationship

Valve model `pw_pres_regurg`:

$$q(p-p_{\text{open}}) = \begin{cases} cA_{\text{o}}\sqrt{p-p_{\text{open}}}, & p < p_{\text{open}} \\ \frac{p-p_{\text{open}}}{R_{\min}}, & p \geq p_{\text{open}} \end{cases} \tag{75}$$

Remark: Model to allow a regurgitant valve in the closed state, degree of regurgitation can be varied by specifying the valve regurgitant area A_{o}

– Coronary circulation model: `ZCRp_CRd_lr`

$$C_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\left(\frac{\mathrm{d}p_{\mathrm{ar}}^{\mathrm{sys},\ell}}{\mathrm{d}t}-Z_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\frac{\mathrm{d}q_{\mathrm{cor,p,in}}^{\mathrm{sys},\ell}}{\mathrm{d}t}\right)=q_{\mathrm{cor,p,in}}^{\mathrm{sys},\ell}-q_{\mathrm{cor,p}}^{\mathrm{sys},\ell} \quad \text{left coronary proximal flow balance}$$

$$R_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,p}}^{\mathrm{sys},\ell}=p_{\mathrm{ar}}^{\mathrm{sys}}-p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-Z_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,p,in}}^{\mathrm{sys},\ell} \quad \text{left coronary proximal momentum}$$

$$C_{\mathrm{cor,d}}^{\mathrm{sys},\ell}\frac{\mathrm{d}(p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-p_{\mathrm{v}}^{\ell})}{\mathrm{d}t}=q_{\mathrm{cor,p}}^{\mathrm{sys},\ell}-q_{\mathrm{cor,d}}^{\mathrm{sys},\ell} \quad \text{left coronary distal flow balance}$$

$$R_{\mathrm{cor,d}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,d}}^{\mathrm{sys},\ell}=p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-p_{\mathrm{at}}^{r} \quad \text{left coronary distal momentum}$$

$$C_{\mathrm{cor,p}}^{\mathrm{sys},r}\left(\frac{\mathrm{d}p_{\mathrm{ar}}^{\mathrm{sys},r}}{\mathrm{d}t}-Z_{\mathrm{cor,p}}^{\mathrm{sys},r}\frac{\mathrm{d}q_{\mathrm{cor,p,in}}^{\mathrm{sys},r}}{\mathrm{d}t}\right)=q_{\mathrm{cor,p,in}}^{\mathrm{sys},r}-q_{\mathrm{cor,p}}^{\mathrm{sys},r} \quad \text{right coronary proximal flow balance}$$

$$R_{\mathrm{cor,p}}^{\mathrm{sys},r}\,q_{\mathrm{cor,p}}^{\mathrm{sys},r}=p_{\mathrm{ar}}^{\mathrm{sys}}-p_{\mathrm{cor,d}}^{\mathrm{sys},r}-Z_{\mathrm{cor,p}}^{\mathrm{sys},r}\,q_{\mathrm{cor,p,in}}^{\mathrm{sys},r} \quad \text{right coronary proximal momentum}$$

$$C_{\mathrm{cor,d}}^{\mathrm{sys},r}\frac{\mathrm{d}(p_{\mathrm{cor,d}}^{\mathrm{sys},r}-p_{\mathrm{v}}^{\ell})}{\mathrm{d}t}=q_{\mathrm{cor,p}}^{\mathrm{sys},r}-q_{\mathrm{cor,d}}^{\mathrm{sys},r} \quad \text{right coronary distal flow balance}$$

$$R_{\mathrm{cor,d}}^{\mathrm{sys},r}\,q_{\mathrm{cor,d}}^{\mathrm{sys},r}=p_{\mathrm{cor,d}}^{\mathrm{sys},r}-p_{\mathrm{at}}^{r} \quad \text{right coronary distal momentum}$$

$$0=q_{\mathrm{cor,d}}^{\mathrm{sys},\ell}+q_{\mathrm{cor,d}}^{\mathrm{sys},r}-q_{\mathrm{cor,d,out}}^{\mathrm{sys}} \quad \text{distal coronary junction flow balance}$$

– Coronary circulation model: `ZCRp_CRd`

$$C_{\mathrm{cor,p}}^{\mathrm{sys}}\left(\frac{\mathrm{d}p_{\mathrm{ar}}^{\mathrm{sys}}}{\mathrm{d}t}-Z_{\mathrm{cor,p}}^{\mathrm{sys}}\frac{\mathrm{d}q_{\mathrm{cor,p,in}}^{\mathrm{sys}}}{\mathrm{d}t}\right)=q_{\mathrm{cor,p,in}}^{\mathrm{sys}}-q_{\mathrm{cor,p}}^{\mathrm{sys}} \quad \text{coronary proximal flow balance}$$

$$R_{\mathrm{cor,p}}^{\mathrm{sys}}\,q_{\mathrm{cor,p}}^{\mathrm{sys}}=p_{\mathrm{ar}}^{\mathrm{sys}}-p_{\mathrm{cor,d}}^{\mathrm{sys}}-Z_{\mathrm{cor,p}}^{\mathrm{sys}}\,q_{\mathrm{cor,p,in}}^{\mathrm{sys}} \quad \text{coronary proximal momentum}$$

$$C_{\mathrm{cor,d}}^{\mathrm{sys}}\frac{\mathrm{d}(p_{\mathrm{cor,d}}^{\mathrm{sys}}-p_{\mathrm{v}}^{\ell})}{\mathrm{d}t}=q_{\mathrm{cor,p}}^{\mathrm{sys}}-q_{\mathrm{cor,d}}^{\mathrm{sys}} \quad \text{coronary distal flow balance}$$

$$R_{\mathrm{cor,d}}^{\mathrm{sys}}\,q_{\mathrm{cor,d}}^{\mathrm{sys}}=p_{\mathrm{cor,d}}^{\mathrm{sys}}-p_{\mathrm{at}}^{r} \quad \text{coronary distal momentum}$$

4.3.6 Systemic and pulmonary circulation, including capillary flow

– Model type : `syspulcap`, cf. [9], p. 51ff.

$-Q^{\ell}_{\mathrm{at}} = q^{\mathrm{pul}}_{\mathrm{ven}} - q^{\ell}_{\mathrm{v,in}}$	left atrium flow balance
$\tilde{R}^{\ell}_{\mathrm{v,in}}\, q^{\ell}_{\mathrm{v,in}} = p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}}$	mitral valve momentum
$-Q^{\ell}_{\mathrm{v}} = q^{\ell}_{\mathrm{v,in}} - q^{\ell}_{\mathrm{v,out}}$	left ventricle flow balance
$\tilde{R}^{\ell}_{\mathrm{v,out}}\, q^{\ell}_{\mathrm{v,out}} = p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}}$	aortic valve momentum
$0 = q^{\ell}_{\mathrm{v,out}} - q^{\mathrm{sys}}_{\mathrm{ar,p}}$	aortic root flow balance
$I^{\mathrm{sys}}_{\mathrm{ar}} \dfrac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar,p}}}{\mathrm{d}t} + Z^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar,p}} = p^{\mathrm{sys}}_{\mathrm{ar}} - p^{\mathrm{sys}}_{\mathrm{ar,d}}$	aortic root inertia
$C^{\mathrm{sys}}_{\mathrm{ar}} \dfrac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar,d}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar,p}} - q^{\mathrm{sys}}_{\mathrm{ar}}$	systemic arterial flow balance
$L^{\mathrm{sys}}_{\mathrm{ar}} \dfrac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar}} = p^{\mathrm{sys}}_{\mathrm{ar,d}} - p^{\mathrm{sys}}_{\mathrm{ar,peri}}$	systemic arterial momentum
$\left(\sum_{j \in \{ \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}} \}} C^{\mathrm{sys}}_{\mathrm{ar},j} \right) \dfrac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar,peri}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar}} - \sum_{j \in \{ \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}} \}} q^{\mathrm{sys}}_{\mathrm{ar},j}$	systemic capillary arterial flow balance
$R^{\mathrm{sys}}_{\mathrm{ar},i}\, q^{\mathrm{sys}}_{\mathrm{ar},i} = p^{\mathrm{sys}}_{\mathrm{ar,peri}} - p^{\mathrm{sys}}_{\mathrm{ven},i}, \quad i \in \{ \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}} \}$	systemic capillary arterial momentum
$C^{\mathrm{sys}}_{\mathrm{ven},i} \dfrac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven},i}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar},i} - q^{\mathrm{sys}}_{\mathrm{ven},i}, \quad i \in \{ \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}} \}$	systemic capillary venous flow balance
$R^{\mathrm{sys}}_{\mathrm{ven},i}\, q^{\mathrm{sys}}_{\mathrm{ven},i} = p^{\mathrm{sys}}_{\mathrm{ven},i} - p^{\mathrm{sys}}_{\mathrm{ven}}, \quad i \in \{ \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}} \}$	systemic capillary venous momentum
$C^{\mathrm{sys}}_{\mathrm{ven}} \dfrac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} = \sum_{j = \substack{\mathrm{spl,espl,} \\ \mathrm{msc,cer,cor}}} q^{\mathrm{sys}}_{\mathrm{ven},j} - q^{\mathrm{sys}}_{\mathrm{ven}}$	systemic venous flow balance
$L^{\mathrm{sys}}_{\mathrm{ven}} \dfrac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ven}}\, q^{\mathrm{sys}}_{\mathrm{ven}} = p^{\mathrm{sys}}_{\mathrm{ven}} - p^{r}_{\mathrm{at}}$	systemic venous momentum

$$
\begin{aligned}
-Q^r_{\mathrm{at}} &= q^{\mathrm{sys}}_{\mathrm{ven}} - q^r_{\mathrm{v,in}} && \text{right atrium flow balance}\\
\tilde{R}^r_{\mathrm{v,in}}\, q^r_{\mathrm{v,in}} &= p^r_{\mathrm{at}} - p^r_{\mathrm{v}} && \text{tricuspid valve momentum}\\
-Q^r_{\mathrm{v}} &= q^r_{\mathrm{v,in}} - q^r_{\mathrm{v,out}} && \text{right ventricle flow balance}\\
\tilde{R}^r_{\mathrm{v,out}}\, q^r_{\mathrm{v,out}} &= p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}} && \text{pulmonary valve momentum}\\
C^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} &= q^r_{\mathrm{v,out}} - q^{\mathrm{pul}}_{\mathrm{ar}} && \text{pulmonary arterial flow balance}\\
L^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ar}}\, q^{\mathrm{pul}}_{\mathrm{ar}} &= p^{\mathrm{pul}}_{\mathrm{ar}} - p^{\mathrm{pul}}_{\mathrm{cap}} && \text{pulmonary arterial momentum}\\
C^{\mathrm{pul}}_{\mathrm{cap}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{cap}}}{\mathrm{d}t} &= q^{\mathrm{pul}}_{\mathrm{ar}} - q^{\mathrm{pul}}_{\mathrm{cap}} && \text{pulmonary capillary flow balance}\\
R^{\mathrm{pul}}_{\mathrm{cap}}\, q^{\mathrm{pul}}_{\mathrm{cap}} &= p^{\mathrm{pul}}_{\mathrm{cap}} - p^{\mathrm{pul}}_{\mathrm{ven}} && \text{pulmonary capillary momentum}\\
C^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} &= q^{\mathrm{pul}}_{\mathrm{cap}} - q^{\mathrm{pul}}_{\mathrm{ven}} && \text{pulmonary venous flow balance}\\
L^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ven}}\, q^{\mathrm{pul}}_{\mathrm{ven}} &= p^{\mathrm{pul}}_{\mathrm{ven}} - p^{\ell}_{\mathrm{at}} && \text{pulmonary venous momentum}
\end{aligned}
$$

with:

$$
Q^{\ell}_{\mathrm{at}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{at}}}{\mathrm{d}t}, \qquad Q^{\ell}_{\mathrm{v}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{v}}}{\mathrm{d}t}, \qquad Q^{r}_{\mathrm{at}} := -\frac{\mathrm{d}V^{r}_{\mathrm{at}}}{\mathrm{d}t}, \qquad Q^{r}_{\mathrm{v}} := -\frac{\mathrm{d}V^{r}_{\mathrm{v}}}{\mathrm{d}t}
$$

4.3.7 Systemic and pulmonary circulation, including capillary and coronary flow

– Model type : `syspulcapcor`

– Variant of `syspulcap`, with coronaries branching off directly after aortic valve

$$-Q^{\ell}_{\mathrm{at}} = q^{\mathrm{pul}}_{\mathrm{ven}} - q^{\ell}_{\mathrm{v,in}} \qquad \text{left atrium flow balance}$$

$$\tilde{R}^{\ell}_{\mathrm{v,in}}\, q^{\ell}_{\mathrm{v,in}} = p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}} \qquad \text{mitral valve momentum}$$

$$-Q^{\ell}_{\mathrm{v}} = q^{\ell}_{\mathrm{v,in}} - q^{\ell}_{\mathrm{v,out}} \qquad \text{left ventricle flow balance}$$

$$\tilde{R}^{\ell}_{\mathrm{v,out}}\, q^{\ell}_{\mathrm{v,out}} = p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}} \qquad \text{aortic valve momentum}$$

$$0 = q^{\ell}_{\mathrm{v,out}} - q^{\mathrm{sys}}_{\mathrm{ar,p}} - q^{\mathrm{sys}}_{\mathrm{ar,cor,in}} \qquad \text{aortic root flow balance}$$

$$I^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar,p}}}{\mathrm{d}t} + Z^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar,p}} = p^{\mathrm{sys}}_{\mathrm{ar}} - p^{\mathrm{sys}}_{\mathrm{ar,d}} \qquad \text{aortic root inertia}$$

$$C^{\mathrm{sys}}_{\mathrm{ar,cor}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar,cor,in}} - q^{\mathrm{sys}}_{\mathrm{ar,cor}} \qquad \text{systemic arterial coronary flow balance}$$

$$R^{\mathrm{sys}}_{\mathrm{ar,cor}}\, q^{\mathrm{sys}}_{\mathrm{ar,cor}} = p^{\mathrm{sys}}_{\mathrm{ar}} - p^{\mathrm{sys}}_{\mathrm{ven,cor}} \qquad \text{systemic arterial coronary momentum}$$

$$C^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar,d}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar,p}} - q^{\mathrm{sys}}_{\mathrm{ar}} \qquad \text{systemic arterial flow balance}$$

$$L^{\mathrm{sys}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ar}}\, q^{\mathrm{sys}}_{\mathrm{ar}} = p^{\mathrm{sys}}_{\mathrm{ar,d}} - p^{\mathrm{sys}}_{\mathrm{ar,peri}} \qquad \text{systemic arterial flow balance}$$

$$\left(\sum_{j \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}} \}} C^{\mathrm{sys}}_{\mathrm{ar},j} \right) \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ar,peri}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar}} - \sum_{j \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}} \}} q^{\mathrm{sys}}_{\mathrm{ar},j} \qquad \text{systemic arterial capillary flow balance}$$

$$R^{\mathrm{sys}}_{\mathrm{ar},i}\, q^{\mathrm{sys}}_{\mathrm{ar},i} = p^{\mathrm{sys}}_{\mathrm{ar,peri}} - p^{\mathrm{sys}}_{\mathrm{ven},i}, \quad i \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}} \} \qquad \text{systemic arterial capillary momentum}$$

$$C^{\mathrm{sys}}_{\mathrm{ven},i} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven},i}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar},i} - q^{\mathrm{sys}}_{\mathrm{ven},i}, \quad i \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}} \} \qquad \text{systemic venous capillary flow balance}$$

$$R^{\mathrm{sys}}_{\mathrm{ven},i}\, q^{\mathrm{sys}}_{\mathrm{ven},i} = p^{\mathrm{sys}}_{\mathrm{ven},i} - p^{\mathrm{sys}}_{\mathrm{ven}}, \quad i \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}} \} \qquad \text{systemic venous capillary momentum}$$

$$C^{\mathrm{sys}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} = \sum_{j = {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer}}} q^{\mathrm{sys}}_{\mathrm{ven},j} - q^{\mathrm{sys}}_{\mathrm{ven}} \qquad \text{systemic venous flow balance}$$

$$L^{\mathrm{sys}}_{\mathrm{ven}} \frac{\mathrm{d}q^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{sys}}_{\mathrm{ven}}\, q^{\mathrm{sys}}_{\mathrm{ven}} = p^{\mathrm{sys}}_{\mathrm{ven}} - p^{r}_{\mathrm{at}} \qquad \text{systemic venous momentum}$$

$$C^{\mathrm{sys}}_{\mathrm{ven,cor}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven,cor}}}{\mathrm{d}t} = q^{\mathrm{sys}}_{\mathrm{ar,cor}} - q^{\mathrm{sys}}_{\mathrm{ven,cor}} \qquad \text{systemic venous coronary flow balance}$$

$$R^{\mathrm{sys}}_{\mathrm{ven,cor}}\, q^{\mathrm{sys}}_{\mathrm{ven,cor}} = p^{\mathrm{sys}}_{\mathrm{ven,cor}} - p^{r}_{\mathrm{at}} \qquad \text{systemic venous coronary momentum}$$

$$
\begin{aligned}
-Q^r_{\mathrm{at}} &= q^{\mathrm{sys}}_{\mathrm{ven}} + q^{\mathrm{sys}}_{\mathrm{ven,cor}} - q^r_{\mathrm{v,in}} && \text{right atrium flow balance}\\
\tilde{R}^r_{\mathrm{v,in}}\, q^r_{\mathrm{v,in}} &= p^r_{\mathrm{at}} - p^r_{\mathrm{v}} && \text{tricuspid valve momentum}\\
-Q^r_{\mathrm{v}} &= q^r_{\mathrm{v,in}} - q^r_{\mathrm{v,out}} && \text{right ventricle flow balance}\\
\tilde{R}^r_{\mathrm{v,out}}\, q^r_{\mathrm{v,out}} &= p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}} && \text{pulmonary valve momentum}\\
C^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} &= q^r_{\mathrm{v,out}} - q^{\mathrm{pul}}_{\mathrm{ar}} && \text{pulmonary arterial flow balance}\\
L^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ar}}\, q^{\mathrm{pul}}_{\mathrm{ar}} &= p^{\mathrm{pul}}_{\mathrm{ar}} - p^{\mathrm{pul}}_{\mathrm{cap}} && \text{pulmonary arterial momentum}\\
C^{\mathrm{pul}}_{\mathrm{cap}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{cap}}}{\mathrm{d}t} &= q^{\mathrm{pul}}_{\mathrm{ar}} - q^{\mathrm{pul}}_{\mathrm{cap}} && \text{pulmonary capillary flow balance}\\
R^{\mathrm{pul}}_{\mathrm{cap}}\, q^{\mathrm{pul}}_{\mathrm{cap}} &= p^{\mathrm{pul}}_{\mathrm{cap}} - p^{\mathrm{pul}}_{\mathrm{ven}} && \text{pulmonary capillary momentum}\\
C^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} &= q^{\mathrm{pul}}_{\mathrm{cap}} - q^{\mathrm{pul}}_{\mathrm{ven}} && \text{pulmonary venous flow balance}\\
L^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ven}}\, q^{\mathrm{pul}}_{\mathrm{ven}} &= p^{\mathrm{pul}}_{\mathrm{ven}} - p^{\ell}_{\mathrm{at}} && \text{pulmonary venous momentum}
\end{aligned}
$$

with:

$$
Q^{\ell}_{\mathrm{at}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{at}}}{\mathrm{d}t}, \qquad Q^{\ell}_{\mathrm{v}} := -\frac{\mathrm{d}V^{\ell}_{\mathrm{v}}}{\mathrm{d}t}, \qquad Q^{r}_{\mathrm{at}} := -\frac{\mathrm{d}V^{r}_{\mathrm{at}}}{\mathrm{d}t}, \qquad Q^{r}_{\mathrm{v}} := -\frac{\mathrm{d}V^{r}_{\mathrm{v}}}{\mathrm{d}t} \tag{76}
$$

4.3.8 Systemic and pulmonary circulation, capillary flow, and respiratory (gas transport + dissociation) model

– Model type : `syspulcaprespir`

– Model equations described in [9], p. 51ff., 58ff.

4.4 Multi-physics coupling

4.4.1 Solid + 0D flow

– Example: Sec. 4.4.1 and `demos/solid_flow0d`

– Problem type: `solid_flow0d`

– solid momentum in (3) or in (12) augmented by following term:

$$
r_u \leftarrow r_u + \int\limits_{\Gamma_0^{\text{s-0d}}} \Lambda\, J \boldsymbol{F}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{77}
$$

– Multiplier constraint

$$r_\Lambda(\Lambda, \boldsymbol{u}; \delta\Lambda) := \left(\int\limits_{\Gamma_0^{\text{s-0d}}} J\boldsymbol{F}^{-\mathrm{T}}\boldsymbol{n}_0 \cdot \boldsymbol{v}(\boldsymbol{u}) \,\mathrm{d}A - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall\, \delta\Lambda \tag{78}$$

– Discrete nonlinear system to solve in each time step n for displacement-based solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u(\mathbf{u}, \boldsymbol{\Lambda}) \\ \mathbf{r}_\Lambda(\boldsymbol{\Lambda}, \mathbf{u}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{79}$$

– Discrete linear system to solve in each Newton iteration k for displacement-based solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{u\Lambda} \\ \mathbf{K}_{\Lambda u} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\boldsymbol{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_\Lambda \end{bmatrix}_{n+1}^{k} \tag{80}$$

– Discrete nonlinear system to solve in each time step n for incompressible solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u(\mathbf{u}, \mathbf{p}, \boldsymbol{\Lambda}) \\ \mathbf{r}_p(\mathbf{u}) \\ \mathbf{r}_\Lambda(\boldsymbol{\Lambda}, \mathbf{u}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{81}$$

– Discrete linear system to solve in each Newton iteration k for incompressible solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} & \mathbf{K}_{u\Lambda} \\ \mathbf{K}_{pu} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\Lambda u} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_p \\ \mathbf{r}_\Lambda \end{bmatrix}_{n+1}^{k} \tag{82}$$

4.4.2 Fluid + 0D flow

– Example: Sec. 4.4.2 `demos/fluid_flow0d`

– Problem type: `fluid_flow0d`

– fluid momentum in (31) augmented by following term:

$$r_v \leftarrow r_v + \int\limits_{\Gamma_t^{\text{f-0d}}} \Lambda\, \boldsymbol{n} \cdot \delta\boldsymbol{v} \,\mathrm{d}a \tag{83}$$

– Multiplier constraint

$$r_\Lambda(\Lambda, \boldsymbol{v}; \delta\Lambda) := \left(\int\limits_{\Gamma_t^{\text{f-0d}}} \boldsymbol{n} \cdot \boldsymbol{v} \,\mathrm{d}a - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall\, \delta\Lambda \tag{84}$$

– Discrete nonlinear system to solve in each time step n:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{\Lambda}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}) \\ \mathbf{r}_\Lambda(\mathbf{\Lambda}, \mathbf{v}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{85}$$

– Discrete linear system to solve in each Newton iteration k:

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\Lambda} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} \\ \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_\Lambda \end{bmatrix}_{n+1}^{k} \tag{86}$$

4.4.3 ALE fluid + 0D flow

– Problem type: `fluid_ale_flow0d`

– fluid momentum in (51) augmented by following term:

$$r_v \leftarrow r_v + \int_{\Gamma_0^{\text{f-0d}}} \Lambda\, \widetilde{J}\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot \delta\boldsymbol{v}\, \mathrm{d}A \tag{87}$$

– Multiplier constraint

$$r_\Lambda(\Lambda, \boldsymbol{v}, \boldsymbol{d}; \delta\Lambda) := \left(\int_{\Gamma_0^{\text{f-0d}}} \widetilde{J}\widetilde{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot (\boldsymbol{v} - \boldsymbol{w}(\boldsymbol{d}))\, \mathrm{d}A - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall\, \delta\Lambda \tag{88}$$

with $\boldsymbol{w}(\boldsymbol{d}) = \frac{\mathrm{d}\boldsymbol{d}}{\mathrm{d}t}$

– Discrete nonlinear system to solve in each time step n:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{\Lambda}, \mathbf{d}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\Lambda(\mathbf{\Lambda}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{89}$$

– Discrete linear system to solve in each Newton iteration k:

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\Lambda} & \mathbf{K}_{vd} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} & \mathbf{K}_{\Lambda d} \\ \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{\Lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_\Lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{90}$$

4.4.4 Fluid-Solid Interaction (FSI)

– Example: Sec. 5.6 and `demos/fsi`

– Problem type: `fsi`

– Solid momentum in (3) or (12) augmented by following term:

$$r_u \leftarrow r_u + \int\limits_{\Gamma_0^{\text{f-s}}} \boldsymbol{\lambda} \cdot \delta\boldsymbol{u} \,\mathrm{d}A \tag{91}$$

– Fluid momentum in (51) is augmented by following term:

$$r_v \leftarrow r_v - \int\limits_{\Gamma_0^{\text{f-s}}} \boldsymbol{\lambda} \cdot \delta\boldsymbol{v} \,\mathrm{d}A \tag{92}$$

Note the different signs (actio=reactio!)

– Lagrange multiplier constraint:

- "solid-governed":

$$r_\lambda(\boldsymbol{u}, \boldsymbol{v}; \delta\boldsymbol{\lambda}) := \int\limits_{\Gamma_0^{\text{f-s}}} (\boldsymbol{u} - \boldsymbol{u}_{\text{f}}(\boldsymbol{v})) \cdot \delta\boldsymbol{\lambda} \,\mathrm{d}A, \quad \forall\, \delta\boldsymbol{\lambda} \tag{93}$$

- "fluid-governed":

$$r_\lambda(\boldsymbol{v}, \boldsymbol{u}; \delta\boldsymbol{\lambda}) := \int\limits_{\Gamma_0^{\text{f-s}}} \left(\boldsymbol{v} - \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}\right) \cdot \delta\boldsymbol{\lambda} \,\mathrm{d}A, \quad \forall\, \delta\boldsymbol{\lambda} \tag{94}$$

– Discrete nonlinear system to solve in each time step n for displacement-based solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u^{\text{s}}(\mathbf{u}, \boldsymbol{\lambda}) \\ \mathbf{r}_v^{\text{f}}(\mathbf{v}, \mathbf{p}, \boldsymbol{\lambda}, \mathbf{d}) \\ \mathbf{r}_p^{\text{f}}(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\lambda(\mathbf{u}, \mathbf{v}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{95}$$

– Discrete nonlinear system to solve in each time step n for incompressible solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u^{\text{s}}(\mathbf{u}, \mathbf{p}^{\text{s}}, \boldsymbol{\lambda}) \\ \mathbf{r}_p^{\text{s}}(\mathbf{u}) \\ \mathbf{r}_v^{\text{f}}(\mathbf{v}, \mathbf{p}, \boldsymbol{\lambda}, \mathbf{d}) \\ \mathbf{r}_p^{\text{f}}(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\lambda(\mathbf{u}, \mathbf{v}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{96}$$

– Discrete linear system to solve in each Newton iteration k for displacement-based solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{u\lambda} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\lambda} & \mathbf{K}_{vd} \\ \mathbf{0} & \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\lambda u} & \mathbf{K}_{\lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}} \\ \mathbf{r}_v^{\mathrm{f}} \\ \mathbf{r}_p^{\mathrm{f}} \\ \mathbf{r}_\lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{97}$$

– Discrete linear system to solve in each Newton iteration k for incompressible solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{u\lambda} & \mathbf{0} \\ \mathbf{K}_{pu} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\lambda} & \mathbf{K}_{vd} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\lambda u} & \mathbf{0} & \mathbf{K}_{\lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{p} \\ \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}} \\ \mathbf{r}_p^{\mathrm{s}} \\ \mathbf{r}_v^{\mathrm{f}} \\ \mathbf{r}_p^{\mathrm{f}} \\ \mathbf{r}_\lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{98}$$

4.4.5 Fluid-Solid Interaction (FSI) + 0D flow

– Problem type: `fsi_flow0d`

– Discrete nonlinear system to solve in each time step n for displacement-based solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}}(\mathbf{u}, \boldsymbol{\lambda}) \\ \mathbf{r}_v^{\mathrm{f}}(\mathbf{v}, \mathbf{p}, \boldsymbol{\lambda}, \boldsymbol{\Lambda}, \mathbf{d}) \\ \mathbf{r}_p^{\mathrm{f}}(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\lambda(\mathbf{u}, \mathbf{v}) \\ \mathbf{r}_\Lambda(\boldsymbol{\Lambda}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{99}$$

– Discrete nonlinear system to solve in each time step n for incompressible solid:

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}}(\mathbf{u}, \mathbf{p}^{\mathrm{s}}, \boldsymbol{\lambda}) \\ \mathbf{r}_p^{\mathrm{s}}(\mathbf{u}) \\ \mathbf{r}_v^{\mathrm{f}}(\mathbf{v}, \mathbf{p}, \boldsymbol{\lambda}, \boldsymbol{\Lambda}, \mathbf{d}) \\ \mathbf{r}_p^{\mathrm{f}}(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\lambda(\mathbf{u}, \mathbf{v}) \\ \mathbf{r}_\Lambda(\boldsymbol{\Lambda}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{100}$$

– Discrete linear system to solve in each Newton iteration k for displacement-based solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{u\lambda} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\lambda} & \mathbf{K}_{v\Lambda} & \mathbf{K}_{vd} \\ \mathbf{0} & \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\lambda u} & \mathbf{K}_{\lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} & \mathbf{K}_{\Lambda d} \\ \mathbf{0} & \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\Lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}} \\ \mathbf{r}_v^{\mathrm{f}} \\ \mathbf{r}_p^{\mathrm{f}} \\ \mathbf{r}_\lambda \\ \mathbf{r}_\Lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{101}$$

– Discrete linear system to solve in each Newton iteration k for incompressible solid:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{u\lambda} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{pu} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\lambda} & \mathbf{K}_{v\Lambda} & \mathbf{K}_{vd} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\lambda u} & \mathbf{0} & \mathbf{K}_{\lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} & \mathbf{K}_{\Lambda d} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{p} \\ \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\Lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u^{\mathrm{s}} \\ \mathbf{r}_p^{\mathrm{s}} \\ \mathbf{r}_v^{\mathrm{f}} \\ \mathbf{r}_p^{\mathrm{f}} \\ \mathbf{r}_\lambda \\ \mathbf{r}_\Lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{102}$$

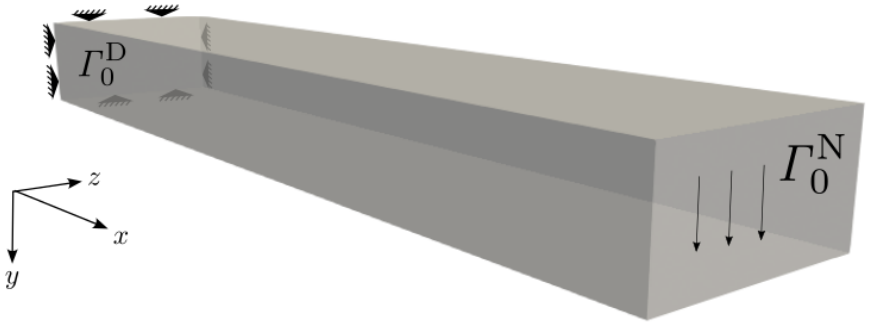

Figure 1: Cantilever, problem setup.

5 Demos

5.1 Demo: Solid

– Physics description given in sec. 4.1

– Input files: `demos/solid/`

Cantilever under tip load

This example demonstrates how to set up a quasi-static solid mechanics elasticity problem. The deformation of a steel cantilever under transverse conservative load is simulated. The structure is fixed on one end. Quadratic 27-node hexahedral finite elements are used for the discretization of the domain. The well-known St. Venant-Kirchhoff material is used as constitutive law, which is a generalization of Hooke's law to the nonlinear realm.

Study the setup shown in fig. 1 and the comments in the input file `solid_cantilever.py` Run the simulation, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
mpiexec -n 1 python3 solid_cantilever.py
```

It is fully sufficient to use one core (`mpiexec -n 1`) for the presented setup.
Open the results file `results_solid_cantilever_displacement.xdmf` in Paraview, and visualize the deformation over time.
Figure 2 shows the displacement magnitude at the end of the simulation.

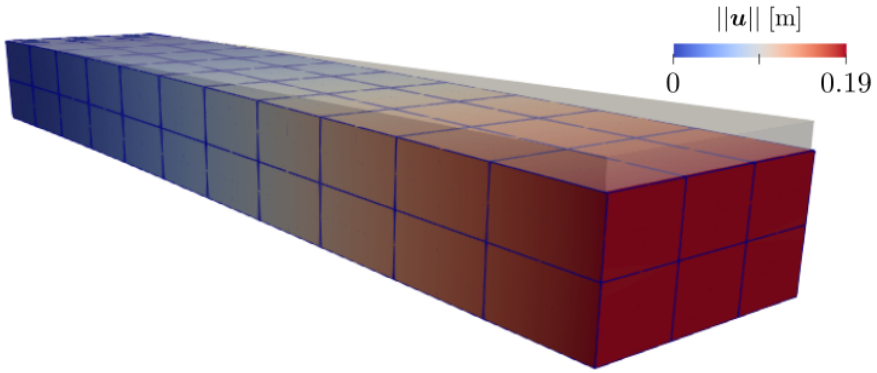

Figure 2: Cantilever, tip deformation. Color shows displacement magnitude.

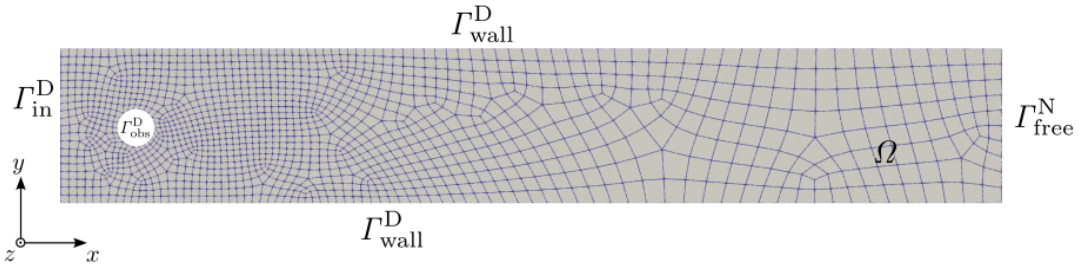

Figure 3: Channel flow, problem setup.

5.2 Demo: Fluid

– Physics description given in sec. 4.2

– Input files: `demos/fluid`

2D channel flow

This example shows how to set up 2D fluid flow in a channel around a rigid obstacle. Incompressible Navier-Stokes flow is solved using Taylor-Hood elements (9-node biquadratic quadrilaterals for the velocity, 4-node bilinear quadrilaterals for the pressure).
Study the setup and the comments in the input file `fluid_channel.py`. Run the simulation, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
mpiexec -n 1 python3 fluid_channel.py
```

It is fully sufficient to use one core (`mpiexec -n 1`) for the presented setup.
Open the results file `results_fluid_channel_velocity.xdmf` and `results_fluid_channel_pressure.xdmf` in Paraview, and visualize the velocity as well as the pressure over time.
Fig. 4 shows the velocity magnitude (top) as well as the pressure (bottom part) at the end of the simulation.

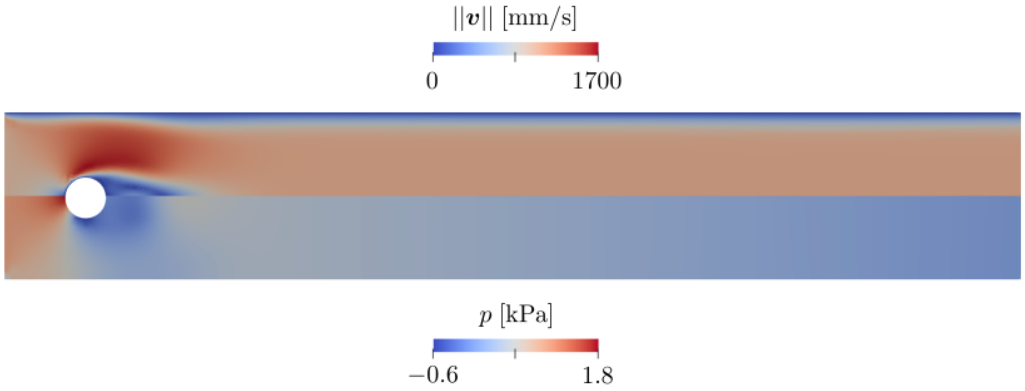

Figure 4: Velocity magnitude (top part) and pressure (bottom part) at end of simulation.

5.3 Demo: 0D flow

– Physics description given in sec. 4.3

– Input files: `demos/flow0d`

Systemic and pulmonary circulation

This example demonstrates how to simulate a cardiac cycle using a lumped-parameter (0D) model for the heart chambers and the entire circulation. Multiple heart beats are run until a periodic state criterion is met (which compares variable values at the beginning to those at the end of a cycle, and stops if the relative change is less than a specified value, here `` `eps_periodic' `` in the `TIME_PARAMS` dictionary). The problem is set up such that periodicity is reached after 5 heart cycles.
Study the setup in fig. 5 and the comments in the input file `flow0d_heart_cycle.py`. Run the simulation, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
python3 flow0d_heart_cycle.py
```

For postprocessing of the time courses of pressures, volumes, and fluxes of the 0D model, either use your own tools to plot the text output files (first column is time, second is the

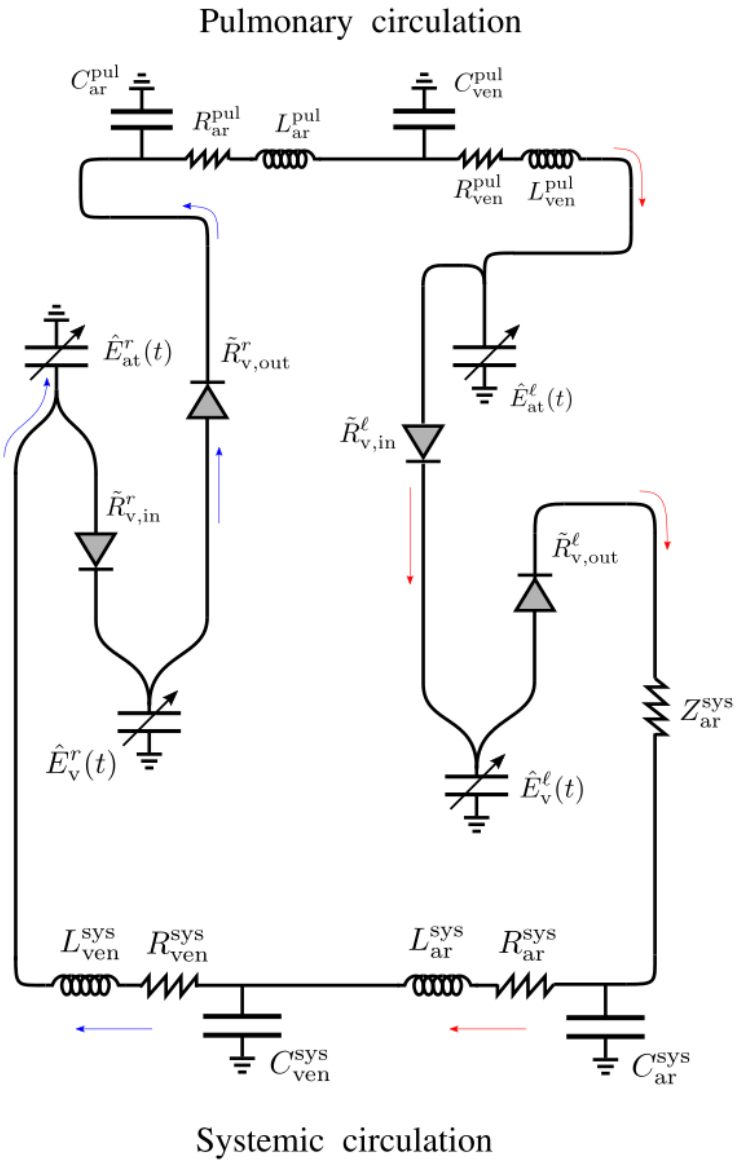

Figure 5: 0D heart, systemic and pulmonary circulation, problem setup.

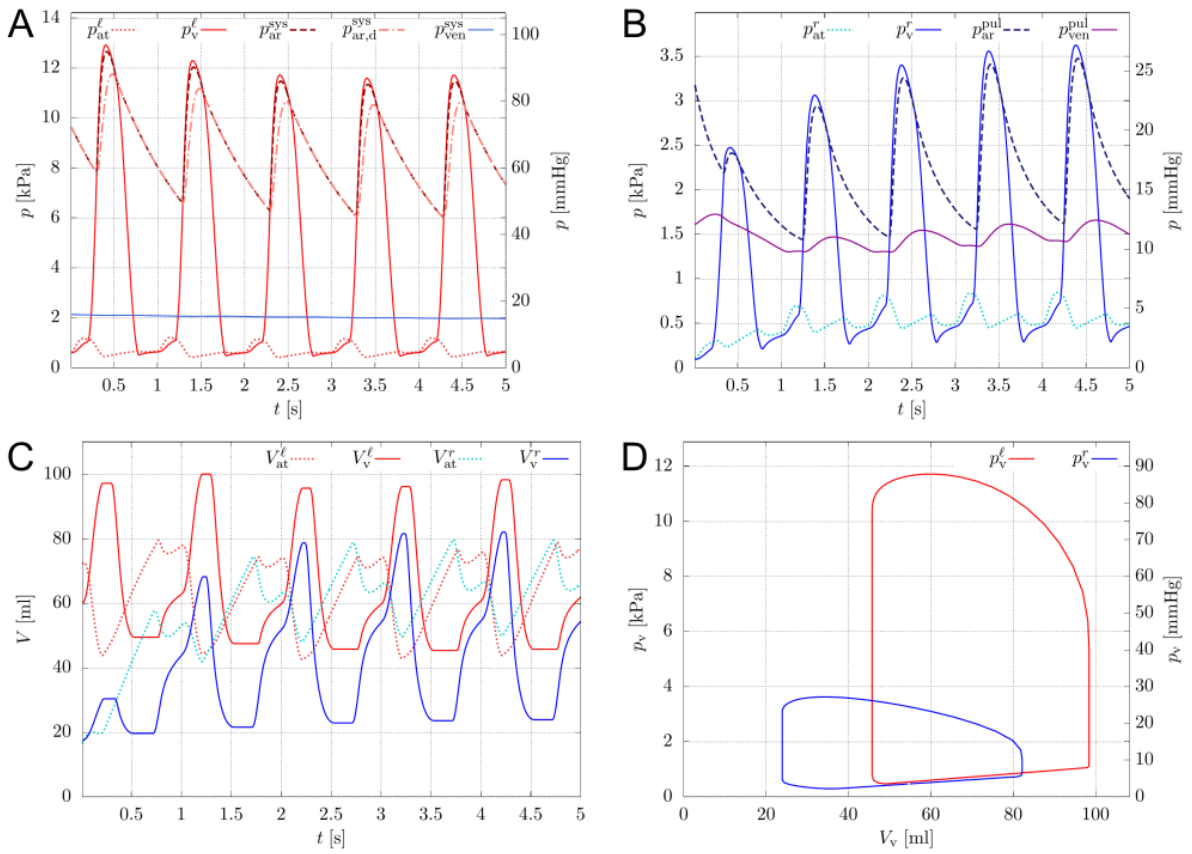

Figure 6: A. Left heart and systemic pressures over time. B. Right heart and pulmonary pressures over time. C. Left and right ventricular and atrial volumes over time. D. Left and right ventricular pressure-volume relationships of periodic (5th) cycle.

respective quantity), or make sure to have Gnuplot (and TeX) installed and navigate to the output folder (`tmp/`) in order to execute the script `flow0d_plot.py` (which lies in `ambit/src/ambit_fe/postprocess/`):

```
flow0d_plot.py -s flow0d_heart_cycle -n 100
```

A folder `plot_flow0d_heart_cycle` is created inside `tmp/`. Look at the results of pressures (p), volumes (V), and fluxes (q, Q) over time. Subscripts `v`, `at`, `ar`, `ven` refer to 'ventricular', 'atrial', 'arterial', and 'venous', respectively. Superscripts `l`, `r`, `sys`, `pul` refer to 'left', 'right', 'systemic', and 'pulmonary', respectively. Try to understand the time courses of the respective pressures, as well as the plots of ventricular pressure over volume. Check that the overall system volume is constant and around 4-5 liters.

The solution is depicted in fig. 6, showing the time course of volumes and pressures of the circulatory system.

5.4 Demo: Solid + 0D flow

– Physics description given in sec. 4.4.1

– Input files: `demos/solid_flow0d`

3D heart, coupled to systemic and pulmonary circulation

This example demonstrates how to set up and simulate a two-chamber (left and right ventricular) solid mechanics heart model coupled to a closed-loop 0D circulatory system. A full dynamic heart cycle of duration 1 s is simulated, where the active contraction is modeled by a prescribed active stress approach. Passive material behavior of the heart muscle is governed by the Holzapfel-Ogden anisotropic strain energy function [14] and a strain rate-dependent viscous model [3]. We start the simulation with "prestressing" using the MULF method [5, 17], which allows to imprint loads without changing the geometry, where the solid is loaded to the initial left and right ventricular pressures. Thereafter, we kickstart the dynamic simulation with passive ventricular filling by the systole of the atria (0D chamber models). Ventricular systole happens in $t \in [0.2\text{ s}, 0.53\text{ s}]$, hence lasting a third of the whole cycle time. After systole, the heart relaxes and eventually fills to about the same pressure as it has been initialized to.

NOTE: For demonstrative purposes, a fairly coarse finite element discretization is chosen here, which by no means yields a spatially converged solution and which may be prone to locking phenomena. The user may increse the parameter `` `order_disp' `` in the `FEM_PARAMS` section from `1` to `2` (and increase `` `quad_degree' `` to `6`) such that quadratic finite element ansatz functions (instead of linear ones) are used. While this will increase accuracy and mitigate locking, computation time will increase.
Study the setup shown in fig. 7 and the comments in the input file `solid_flow0d_heart_cycle.py`.
Run the simulation, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
mpiexec -n 1 python3 solid_flow0d_heart_cycle.py
```

It is fully sufficient to use one core (`mpiexec -n 1`) for the presented setup, while you might want to use more (e.g., `mpiexec -n 4`) if you increase `` `order_disp' `` to `2`.
Open the results file `results_solid_flow0d_heart_cycle_displacement.xdmf` in Paraview, and visualize the deformation over the heart cycle.
For postprocessing of the time courses of pressures, volumes, and fluxes of the 0D model, either use your own tools to plot the text output files (first column is time, second is the respective quantity), or make sure to have Gnuplot (and TeX) installed and navigate to the output folder (`tmp/`) in order to execute the script `flow0d_plot.py` (which lies in `ambit/src/ambit_fe/postprocess/`):

```
flow0d_plot.py -s solid_flow0d_heart_cycle -V0 117e3 93e3 0 0 0
```

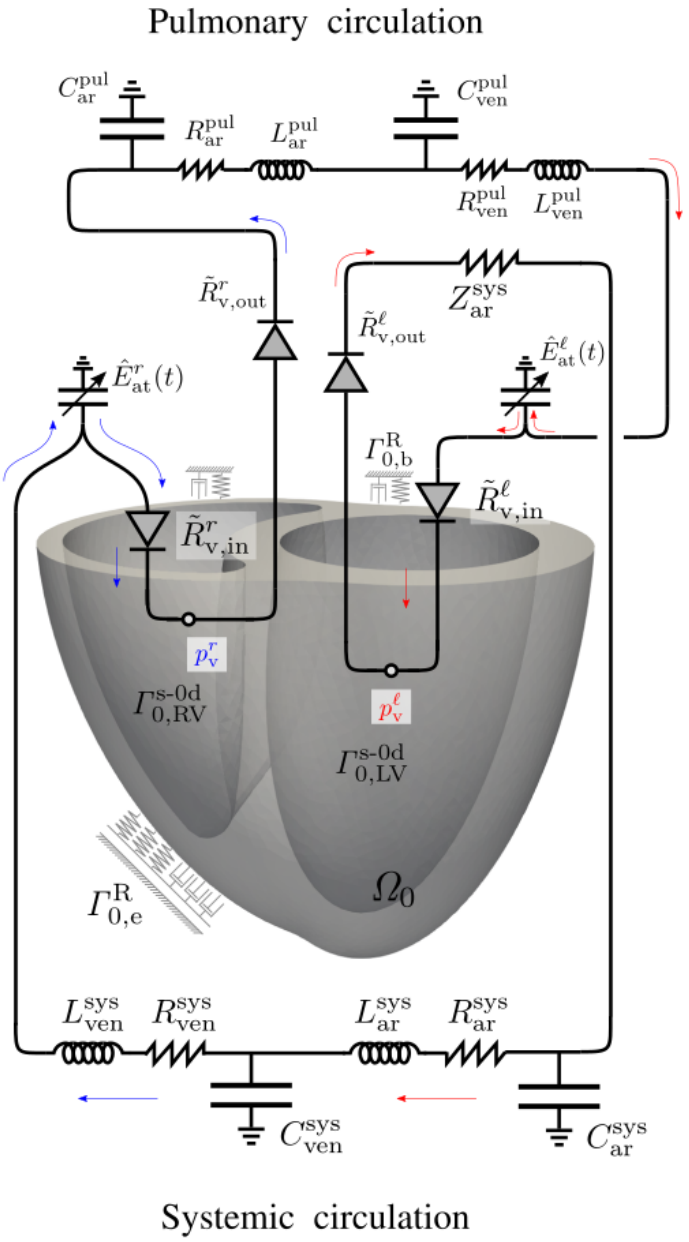

Figure 7: Generic 3D ventricular heart model coupled to a closed-loop systemic and pulmonary circulation model.

A folder `plot_solid_flow0d_heart_cycle` is created inside `tmp/`. Look at the results of pressures (p), volumes (V), and fluxes (q, Q) over time. Subscripts `v`, `at`, `ar`, `ven` refer to 'ventricular', 'atrial', 'arterial', and 'venous', respectively. Superscripts `l`, `r`, `sys`, `pul` refer to 'left', 'right', 'systemic', and 'pulmonary', respectively. Try to understand the time courses of the respective pressures, as well as the plots of ventricular pressure over volume. Check that the overall system volume is constant and around 4-5 liters.

NOTE: This setup computes only one cardiac cycle, which does not yield a periodic state solution (compare e.g. initial and end-cyclic right ventricular pressures and volumes, which do not coincide). Change the parameter `number_of_cycles` from `1` to `10` and re-run the simulation. The simulation will stop when the cycle error (relative change in 0D variable quantities from beginning to end of a cycle) falls below the value of `` `eps_periodic' `` (set to 5%). How many cycles are needed to reach periodicity?

Figure 8 shows a high-fidelity solution using a refined mesh and quadratic tetrahedral elements. Compare your solution from the coarser mesh. What is the deviation in ventricular volume?

5.5 Demo: Fluid + 0D flow

– Physics description given in sec. 4.4.2

– Input files: `demos/fluid_flow0d`

Blocked pipe flow with 0D model bypass

Note: *This demo only runs with the mixed dolfinx branch, which is pre-installed in the Ambit devenv Docker container. Pull this container and install Ambit in there according to the instructions in sec. 2.*

This example demonstrates how to couple 3D fluid flow to a 0D lumped-parameter model. Incompressible transient Navier-Stokes flow in a pipe with prescribed inflow is solved, with the special constraint that an internal boundary (all-time closed valve) separates region 1 and region 2 of the pipe. This internal Dirichlet condition can only be achieved by splitting the pressure space, hence having duplicate pressure nodes at the valve plane. Otherwise, fluid would experience deceleration towards the valve and unphysical acceleration behind it, since the pressure gradient drives fluid flow. To achieve this, the mixed Dolfinx branch instead of the main branch is used. It is installed inside the Ambit devenv Docker container. In the future, this functionality is expected to be merged into the Dolfinx main branch (at least it was announced...).

This example demonstrates how the closed valve can be bypassed by a 0D flow model that links the 3D fluid out-flow of one region to the in-flow of the other region. The 0D

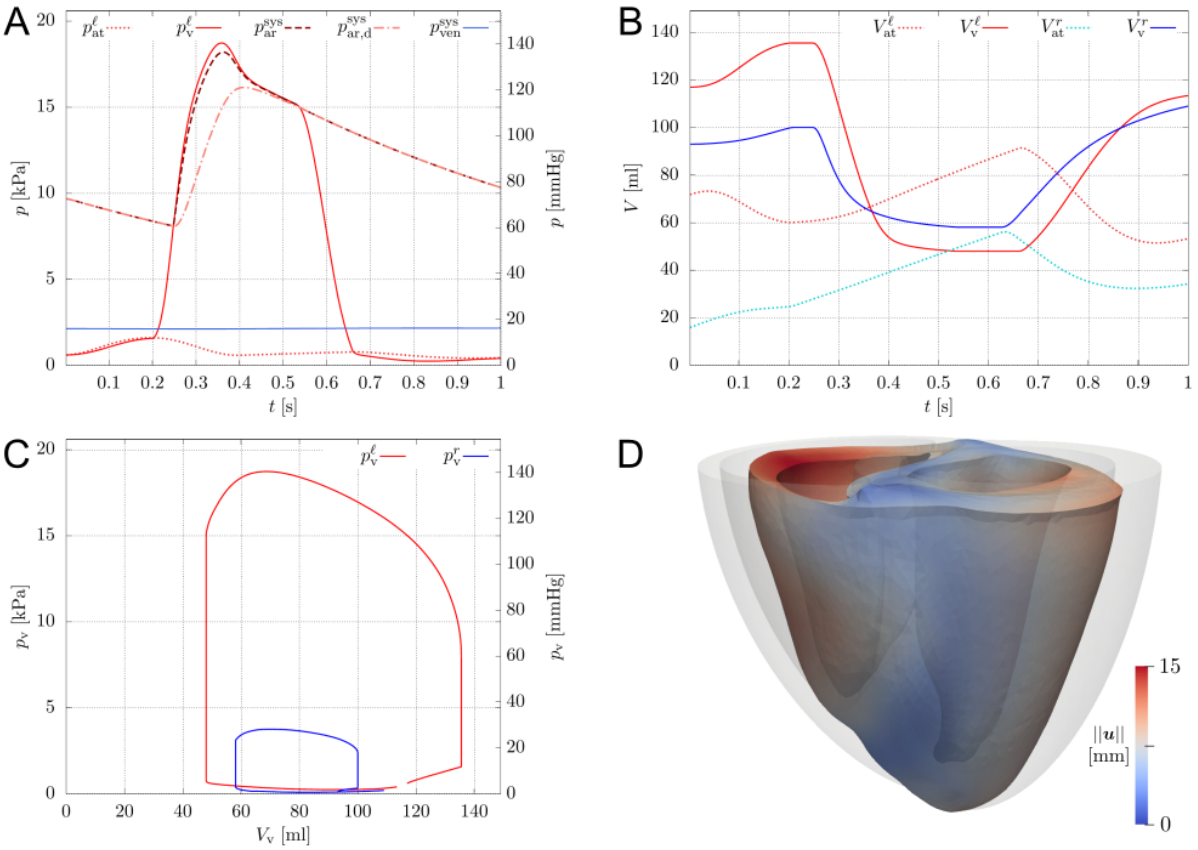

Figure 8: A. Left heart and systemic pressures over time. B. Left and right ventricular and atrial volumes over time. C. Left and right ventricular pressure-volume relationships. D. Snapshot of heart deformation at end-systole, color indicates displacement magnitude.

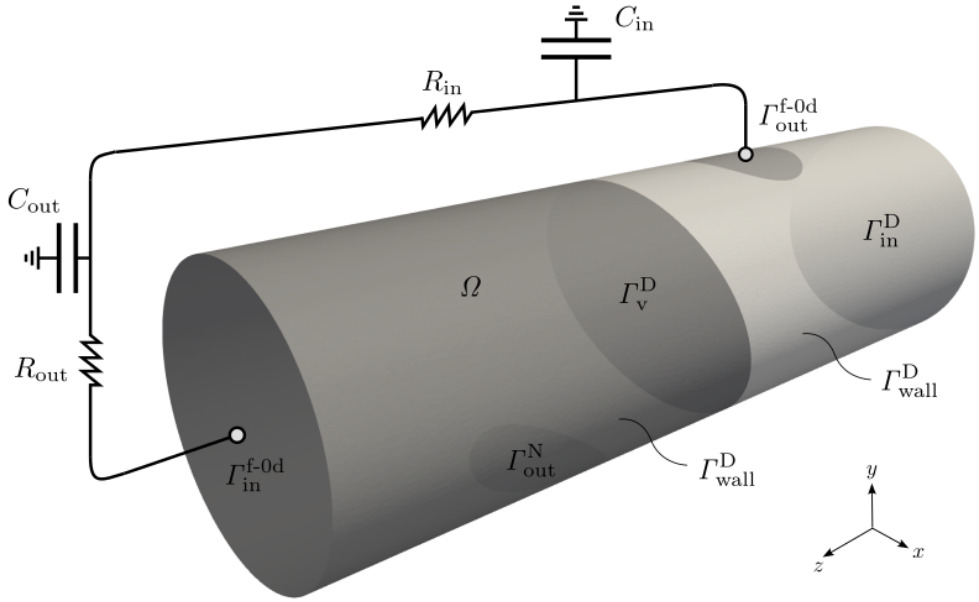

Figure 9: Blocked pipe with 0D model bypass, simulation setup.

model consists of two Windkessel models in series, each having compliance, resistance, and inertance elements.
Study the setup shown in fig. 9 and the comments in the input file `fluid_flow0d_pipe.py`. Run the simulation, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
mpiexec -n 1 python3 fluid_flow0d_pipe.py
```

It is fully sufficient to use one core (`mpiexec -n 1`) for the presented setup.
Open the results file `results_fluid_flow0d_pipe_velocity.xdmf` in Paraview, and visualize the velocity over time.
Think of which parameter(s) of the 0D model to tweak in order to achieve a) little to no fluid in-flow (into $\Gamma^{\mathrm{f\text{-}0d}}_{\mathrm{in}}$), b) almost the same flow across $\Gamma^{\mathrm{f\text{-}0d}}_{\mathrm{out}}$ and $\Gamma^{\mathrm{f\text{-}0d}}_{\mathrm{in}}$. Think of where the flow is going to in case of a).

Figure shows the velocity streamlines and magnitude at the end of the simulation.

5.6 Demo: FSI

– Physics description given in sec. 4.4.4

– Input files: `demos/fsi`

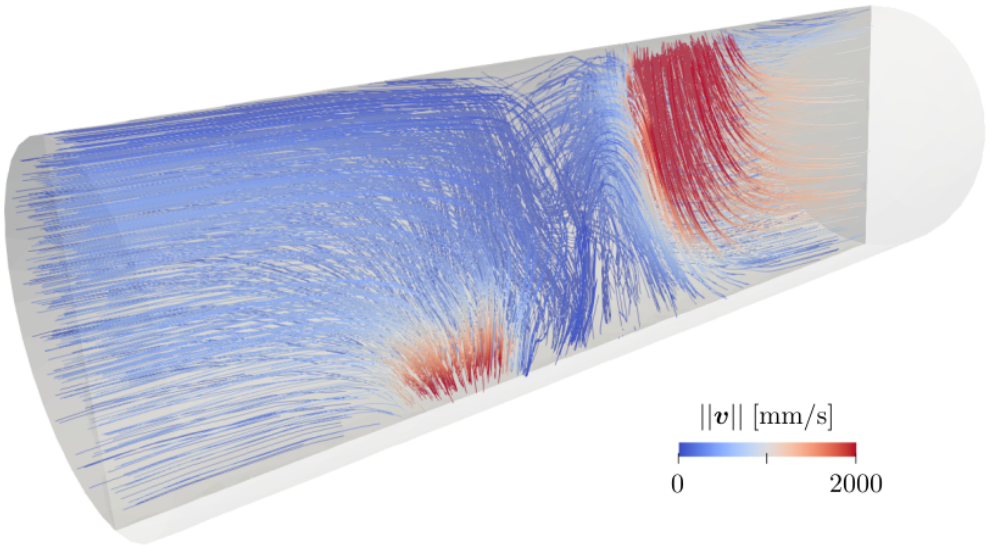

Figure 10: Streamlines of velocity at end of simulation, color indicates velcity magnitude.

Note: *FSI only runs with the mixed dolfinx branch, which is pre-installed in the Ambit devenv Docker container. Pull this container and install Ambit in there according to the instructions in sec. 2.*

Channel flow around elastic flag

Incompressible fluid flow in a 2D channel around an elastic flag is studied. The setup corresponds to the well-known Turek benchmark [19]. Here, the two cases FSI2 and FSI3 from the original setup are investigated. A prescribed inflow velocity with parabolic inflow profile is used:

$$\boldsymbol{v}_{\mathrm{f}} = \bar{v}(t,y)\boldsymbol{e}_x \quad \text{on } \Gamma^{D,\mathrm{f}}_{t,\mathrm{in}} \times [0,T], \tag{103}$$

with

$$\bar{v}(t,y) = \begin{cases} 1.5\,\bar{U}\,\frac{y(H-y)}{\left(\frac{H}{2}\right)^2}\frac{1-\cos\left(\frac{\pi}{2}t\right)}{2}, & \text{if } t < 2, \\ 1.5\,\bar{U}\,\frac{y(H-y)}{\left(\frac{H}{2}\right)^2}, & \text{else,} \end{cases} \tag{104}$$

with $\bar{U} = 10^3$ mm/s (FSI2) and $\bar{U} = 2 \cdot 10^3$ mm/s (FSI3).

Geometrical parameters, given in [mm], are:

L	H	r	l	h	d_x	d_y
2500	410	50	350	20	200	200

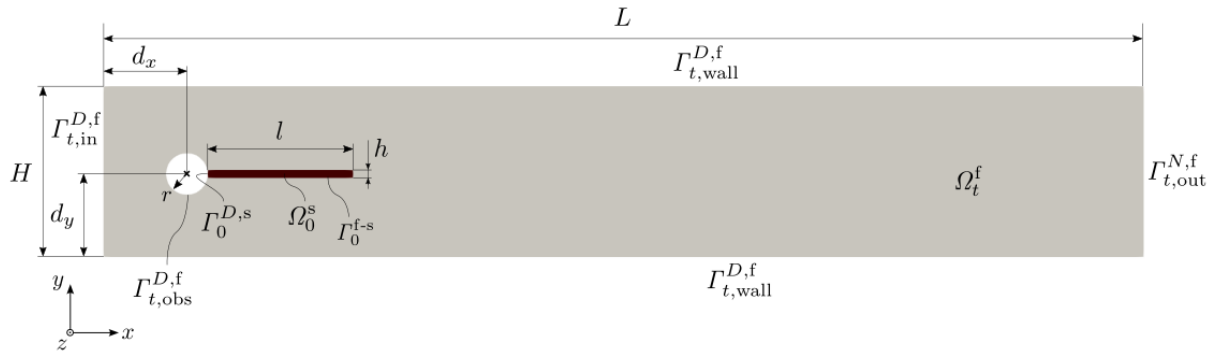

Figure 11: Channel flow around an elastic flag [19], problem setup.

Both solid and fluid are discretized with quadrilateral $\mathbb{Q}^2$-$\mathbb{Q}^1$ Taylor-Hood finite elements, hence no stabilization for the fluid problem is needed. Temporal discretization for both the solid and the fluid are carried out with a Generalized-α scheme with no numerical damping ($\rho_{\mathrm{inf}} = 1$).

Study the setup shown in fig. 11 together with the parameters in the table and the comments in the input file `fsi_channel_flag.py`. Run the simulation for FSI2 and FSI3 cases, either in one of the provided Docker containers or using your own FEniCSx/Ambit installation, using the command

```
mpiexec -n 1 python3 fsi_channel_flag.py
```

If your system allows, use more than one core (e.g. `mpiexec -n 4`) in order to speed up the simulation a bit.

The physics of the problem are strongly time-dependent, and a (near-)periodic oscillation of the flag only occurs after $t \approx 10$ s (FSI2) and $t \approx 5$ s (FSI3). Run the problem to the end ($t = 15$ s for FSI2, $t = 7.5$ s for FSI3), be patient, and monitor the flag tip displacement over time.

Figure 12 depicts the velocity at three instances in time towards the end of the simulation for the FSI2 case, and figure 13 shows the flag's tip displacement over time compared to the reference solution, over a time interval where the solution has become periodic. (Note that the reference solution from the official link shown in the input file needs to be time-adjusted, i.e. synchronized with the first peak in the interval of interest, since the time column of the reference data does not correspond to the physical time of the problem setup.)

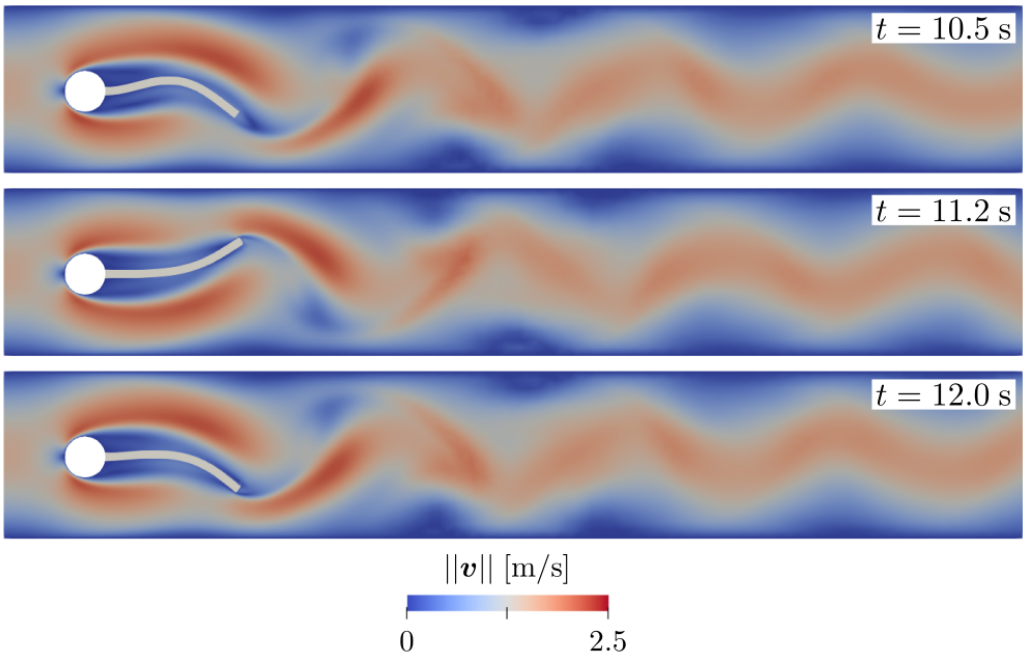

Figure 12: FSI2 case: Magnitude of fluid velocity at three instances in time ($t = 10.5$ s, $t = 11.2$ s, and $t = 12$ s) towards end of simulation, color indicates velcity magnitude.

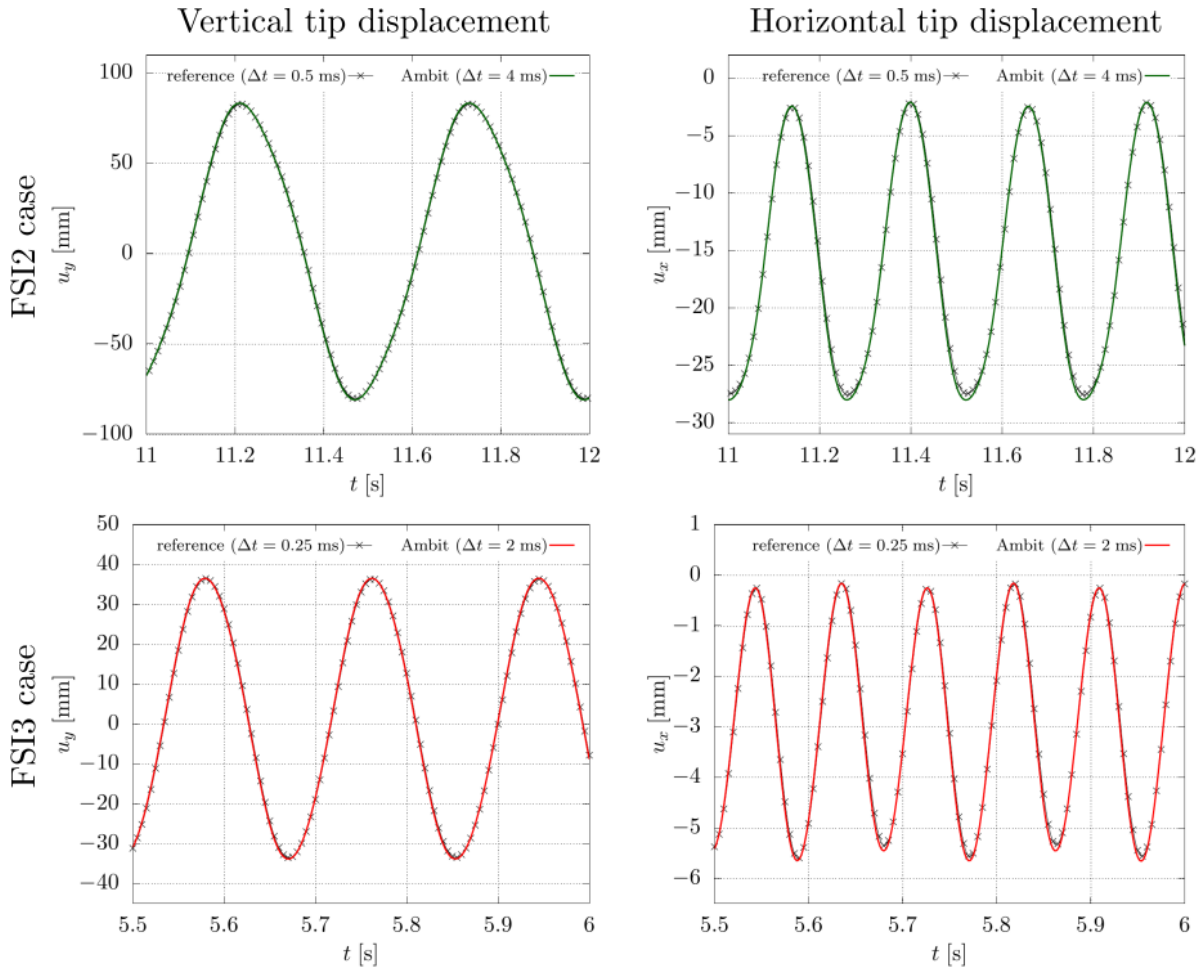

Figure 13: Comparison to benchmark reference solution for the time course of the flag's tip displacement for the two setups FSI2 and FSI3. A fairly coarse time step of $\Delta t = 4$ ms (FSI2) and $\Delta t = 2$ ms (FSI3) already allows a close match to the original results.

6 Table of symbols

Ω_0, Ω	: reference, current domain
Γ_0, Γ	: reference, current boundary
$\boldsymbol{x}_0, \boldsymbol{x}$	: coordinates of the reference, current frame
$\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$	: unit vectors of the cartesian reference frame
$\boldsymbol{n}_0, \boldsymbol{n}$	: unit outward normal defined in the reference, current frame
$\boldsymbol{\nabla}_0, \boldsymbol{\nabla}$	: Nabla operator with respect to the reference, current frame
t, T	: current, end time of an initial boundary value problem
$\boldsymbol{u}, \hat{\boldsymbol{u}}_0$	: solid mechanics displacement field, and prescribed initial value
$\delta\boldsymbol{u}, \Delta\boldsymbol{u}$	: solid mechanics displacement test, trial function
p	: solid mechanics hydrostatic pressure, or fluid mechanics pressure
$\delta p, \Delta p$	: solid or fluid mechanics pressure test, trial function
$\boldsymbol{v} = \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}, \hat{\boldsymbol{v}}_0$	: solid mechanics velocity, and prescribed initial value
$\boldsymbol{a} = \frac{\mathrm{d}^2\boldsymbol{u}}{\mathrm{d}t^2}$	: solid mechanics acceleration
$\boldsymbol{v}, \hat{\boldsymbol{v}}_0$	: fluid mechanics velocity, and prescribed initial value
$\delta\boldsymbol{v}, \Delta\boldsymbol{v}$	: fluid mechanics velocity test, trial function
$\boldsymbol{a} = \frac{\partial\boldsymbol{v}}{\partial t}$	: fluid mechanics acceleration
$\boldsymbol{d}, \hat{\boldsymbol{d}}_0$	: ALE displacement field, and prescribed initial value
$\delta\boldsymbol{d}, \Delta\boldsymbol{d}$	: ALE displacement test, trial function
$\hat{\boldsymbol{b}}_0, \hat{\boldsymbol{b}}$	: body force vector defined in the reference, current frame
$\boldsymbol{w} = \frac{\mathrm{d}\boldsymbol{d}}{\mathrm{d}t}, \hat{\boldsymbol{w}}_0$	: ALE velocity, and prescribed initial value
ρ_0, ρ	: reference, current density
$\boldsymbol{P} = \boldsymbol{F}\boldsymbol{S}$	: 1st Piola Kirchhoff stress tensor
$\boldsymbol{F} = \boldsymbol{I} + \boldsymbol{\nabla}_0\boldsymbol{u}$	: solid deformation gradient
$\widetilde{\boldsymbol{F}} = \boldsymbol{I} + \boldsymbol{\nabla}_0\boldsymbol{d}$	: ALE deformation gradient
$J = \det\boldsymbol{F}$	: determinant of solid deformation gradient
$\widetilde{J} = \det\widetilde{\boldsymbol{F}}$	: determinant of ALE deformation gradient
$\boldsymbol{S}$	: 2nd Piola-Kirchhoff stress tensor
$\boldsymbol{\sigma}$	: Cauchy stress tensor
$\boldsymbol{t}_0, \hat{\boldsymbol{t}}_0$	: 1st Piola-Kirchhoff traction, prescribed 1st Piola-Kirchhoff traction
$\boldsymbol{t}, \hat{\boldsymbol{t}}$	: Cauchy traction, prescribed Cauchy traction

Bibliography

[1] C. J. Arthurs, K. D. Lau, K. N. Asrress, S. R. Redwood, and C. A. Figueroa. A mathematical model of coronary blood flow control: simulation of patient-specific three-dimensional hemodynamics during exercise. *Am J Physiol Heart Circ Physiol*, 310(9):H1242–H1258, 2016.

[2] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.17, Argonne National Laboratory, 2022.

[3] D. Chapelle, P. L. Tallec, P. Moireau, and M. Sorine. Energy-preserving muscle tissue model: formulation and compatible discretizations. *Journal for Multiscale Computational Engineering*, 10(2):189–211, 2012.

[4] C. Farhat, P. Avery, T. Chapman, and J. Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *International Journal for Numerical Methods in Engineering*, 98(9):625–662, 2014.

[5] M. W. Gee, C. Förster, and W. A. Wall. A computational strategy for prestressing patient-specific biomechanical problems under finite deformation. *International Journal for Numerical Methods in Biomedical Engineering*, 26(1):52–72, 2010.

[6] M. W. Gee, C. T. Kelley, and R. B. Lehoucq. Pseudo-transient continuation for nonlinear transient elasticity. *International Journal for Numerical Methods in Engineering*, 78(10):1209–1219, 2009.

[7] S. Göktepe, O. J. Abilez, K. K. Parker, and E. Kuhl. A multiscale model for eccentric and concentric cardiac growth through sarcomerogenesis. *J Theor Biol*, 265(3):433–442, 2010.

[8] J. M. Guccione, K. D. Costa, and A. D. McCulloch. Finite element stress analysis of left ventricular mechanics in the beating dog heart. *J Biomech*, 28(10):1167–1177, 1995.

[9] M. Hirschvogel. *Computational modeling of patient-specific cardiac mechanics with model reduction-based parameter estimation and applications to novel heart assist technologies*. Verlag Dr. Hut, MediaTUM, 1 edition, 2019.

[10] M. Hirschvogel. Ambit – A FEniCS-based cardiovascular multi-physics solver. *Journal of Open Source Software*, 9(93):5744, 2024.

[11] M. Hirschvogel, M. Balmus, M. Bonini, and D. Nordsletten. Fluid-reduced-solid interaction (FrSI): Physics- and projection-based model reduction for cardiovascular applications. *Preprint, submitted to Elsevier*, 2022.

[12] M. Hirschvogel, M. Bassilious, L. Jagschies, S. M. Wildhirt, and M. W. Gee. A monolithic 3D-0D coupled closed-loop model of the heart and the vascular system: Experiment-based parameter estimation for patient-specific cardiac mechanics. *Int J Numer Method Biomed Eng*, 33(8):e2842, 2017.

[13] G. A. Holzapfel. *Nonlinear Solid Mechanics – A Continuum Approach for Engineering*. Wiley Press Chichester, 2000.

[14] G. A. Holzapfel and R. W. Ogden. Constitutive modelling of passive myocardium: A structurally based framework for material characterization. *Phil Trans R Soc A*, 367(1902):3445–3475, 2009.

[15] A. Logg, K.-A. Mardal, and G. N. Wells, editors. *Automated Solution of Differential Equations by the Finite Element Method – The FEniCS Book*. Springer, 2012.

[16] D. A. Nordsletten, M. McCormick, P. J. Kilner, P. Hunter, D. Kay, and N. P. Smith. Fluid-solid coupling for the investigation of diastolic and systolic human left ventricular function. *International Journal for Numerical Methods in Biomedical Engineering*, 27(7):1017–1039, 2011.

[17] A. Schein and M. W. Gee. Greedy maximin distance sampling based model order reduction of prestressed and parametrized abdominal aortic aneurysms. *Advanced Modeling and Simulation in Engineering Sciences*, 8(18), 2021.

[18] T. E. Tezduyar and Y. Osawa. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190(3–4):411–430, 2000.

[19] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In H.-J.Bungartz and M. Schäfer, editors, *Lecture Notes in Computational Science and Engineering*, volume 53, pages 371–385. Springer Berlin Heidelberg, 2006.

[20] N. Westerhof, J.-W. Lankhaar, and B. E. Westerhof. The arterial Windkessel. *Med Biol Eng Comput*, 47(2):H81–H88, 2009.