Ambit – A FEniCS-based cardiovascular physics solver

Dr.-Ing. Marc Hirschvogel

November 20, 2023

Contents

1 Preface

Ambit is an open-source software tool written in Python for parallel multi-physics simulations focusing on – but not limited to – cardiac mechanics. Amongst others, it contains re-implementations and generalizations of methods developed by the author for his PhD thesis [9]. Ambit makes use of the open-source finite element library FEniCS/dolfinx (https://fenicsproject.org) [14] along with the linear algebra package PETSc (https://petsc.org) [2]. It is constantly updated to ensure compatibility with a recent dolfinx development version, hence guaranteeing a state-of-the-art finite element and linear algebra backend.

Ambit is 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 [12], implementing a range of hyperelastic, viscous, and active material models. Specifically, the well-known anisotropic Holzapfel-Ogden [13] 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 [15].

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 [16] can be used.

A variety of reduced 0D lumped models targeted at blood circulation modeling are implemented, including 3- and 4-element Windkessel models [17] as well as closed-loop full circulation [11] and coronary flow models [1].

Monolithic multi-physics coupling of solid, fluid, and ALE-fluid with 0D lumped models is implemented such that cardiovascular simulations with realistic boundary conditions can be performed. Monolithic fluid-solid interaction (FSI) will be provided in the near future once mixed domain functionality is fully supported in the finite element backend dolfinx.

Implementations for a recently proposed novel physics- and projection-based model reduction for FSI, denoted as fluid-reduced-solid interaction (FrSI) [10], 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] adaptibity 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`.

2 Installation

In order to use Ambit, you need to install FEniCSx (https://github.com/FEniCS/dolfinxinstallation) (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 Solid mechanics

– Example: Sec. 7.1 and `demos/solid`

– Problem type: `solid`

– Solid mechanics are formulated in a Total Lagrangian frame

3.1 Strong form

3.1.1 Displacement-based

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

$$\begin{aligned}
\boldsymbol{\nabla}_0 \cdot \boldsymbol{P}(\boldsymbol{u}, \boldsymbol{v}(\boldsymbol{u})) + \hat{\boldsymbol{b}}_0 &= \rho_0 \boldsymbol{a}(\boldsymbol{u}) && \text{in } \Omega_0 \times [0,T], && (1)\\
\boldsymbol{u} &= \hat{\boldsymbol{u}} && \text{on } \Gamma_0^{\mathrm{D}} \times [0,T], && (2)\\
\boldsymbol{t}_0 = \boldsymbol{P}\boldsymbol{n}_0 &= \hat{\boldsymbol{t}}_0 && \text{on } \Gamma_0^{\mathrm{N}} \times [0,T], && (3)\\
\boldsymbol{u}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{u}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, && (4)\\
\boldsymbol{v}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, && (5)
\end{aligned}$$

3.1.2 Incompressible mechanics

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

$$\begin{aligned}
\boldsymbol{\nabla}_0 \cdot \boldsymbol{P}(\boldsymbol{u}, p, \boldsymbol{v}(\boldsymbol{u})) + \hat{\boldsymbol{b}}_0 &= \rho_0 \boldsymbol{a}(\boldsymbol{u}) && \text{in } \Omega_0 \times [0,T], && (6)\\
J(\boldsymbol{u}) - 1 &= 0 && \text{in } \Omega_0 \times [0,T], && (7)\\
\boldsymbol{u} &= \hat{\boldsymbol{u}} && \text{on } \Gamma_0^{\mathrm{D}} \times [0,T], && (8)\\
\boldsymbol{t}_0 = \boldsymbol{P}\boldsymbol{n}_0 &= \hat{\boldsymbol{t}}_0 && \text{on } \Gamma_0^{\mathrm{N}} \times [0,T], && (9)\\
\boldsymbol{u}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{u}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, && (10)\\
\boldsymbol{v}(\boldsymbol{x}_0, 0) &= \hat{\boldsymbol{v}}_0(\boldsymbol{x}_0) && \text{in } \Omega_0, && (11)
\end{aligned}$$

with velocity and acceleration $\boldsymbol{v} = \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}$ and $\boldsymbol{a} = \frac{\mathrm{d}^2\boldsymbol{u}}{\mathrm{d}t^2}$, respectively

3.2 Weak form

3.2.1 Displacement-based

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

– Principal of Virtual Work:

$$r(\boldsymbol{u}; \delta\boldsymbol{u}) := \delta\mathcal{W}_{\mathrm{kin}}(\boldsymbol{u}; \delta\boldsymbol{u}) + \delta\mathcal{W}_{\mathrm{int}}(\boldsymbol{u}; \delta\boldsymbol{u}) - \delta\mathcal{W}_{\mathrm{ext}}(\boldsymbol{u}; \delta\boldsymbol{u}) = 0, \quad \forall\, \delta\boldsymbol{u} \tag{12}$$

– Kinetic virtual work:

$$\delta \mathcal{W}_{\text{kin}}(\boldsymbol{u}; \delta \boldsymbol{u}) = \int\limits_{\Omega_0} \rho_0 \, \boldsymbol{a}(\boldsymbol{u}) \cdot \delta \boldsymbol{u} \, \mathrm{d}V \tag{13}$$

– Internal virtual work:

$$\delta \mathcal{W}_{\text{int}}(\boldsymbol{u}; \delta \boldsymbol{u}) = \int\limits_{\Omega_0} \boldsymbol{P}(\boldsymbol{u}, \boldsymbol{v}(\boldsymbol{u})) : \boldsymbol{\nabla}_0 \delta \boldsymbol{u} \, \mathrm{d}V \tag{14}$$

– External virtual work:

- conservative Neumann load:

$$\delta \mathcal{W}_{\text{ext}}(\delta \boldsymbol{u}) = \int\limits_{\Gamma_0^{\text{N}}} \hat{\boldsymbol{t}}_0(t) \cdot \delta \boldsymbol{u} \, \mathrm{d}A \tag{15}$$

- Neumann pressure load in current normal direction:

$$\delta \mathcal{W}_{\text{ext}}(\boldsymbol{u}; \delta \boldsymbol{u}) = - \int\limits_{\Gamma_0^{\text{N}}} \hat{p}(t) \, J \boldsymbol{F}^{-\text{T}} \boldsymbol{n}_0 \cdot \delta \boldsymbol{u} \, \mathrm{d}A \tag{16}$$

- general Neumann load in current direction:

$$\delta \mathcal{W}_{\text{ext}}(\boldsymbol{u}; \delta \boldsymbol{u}) = \int\limits_{\Gamma_0} J \boldsymbol{F}^{-\text{T}} \, \hat{\boldsymbol{t}}_0(t) \cdot \delta \boldsymbol{u} \, \mathrm{d}A \tag{17}$$

- body force:

$$\delta \mathcal{W}_{\text{ext}}(\delta \boldsymbol{u}) = \int\limits_{\Omega_0} \hat{\boldsymbol{b}}_0(t) \cdot \delta \boldsymbol{u} \, \mathrm{d}V \tag{18}$$

- generalized Robin condition:

$$\delta \mathcal{W}_{\text{ext}}(\boldsymbol{u}; \delta \boldsymbol{u}) = - \int\limits_{\Gamma_0^{\text{N}}} [k \, \boldsymbol{u} + c \, \boldsymbol{v}(\boldsymbol{u})] \cdot \delta \boldsymbol{u} \, \mathrm{d}A \tag{19}$$

- generalized Robin condition in reference surface normal direction:

$$\delta \mathcal{W}_{\text{ext}}(\boldsymbol{u}; \delta \boldsymbol{u}) = - \int\limits_{\Gamma_0^{\text{N}}} (\boldsymbol{n}_0 \otimes \boldsymbol{n}_0) \, [k \, \boldsymbol{u} + c \, \boldsymbol{v}(\boldsymbol{u})] \cdot \delta \boldsymbol{u} \, \mathrm{d}A \tag{20}$$

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

$$\mathbf{r}_u(\mathbf{u})|_{n+1} = \mathbf{0} \tag{21}$$

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

$$\mathbf{K}_{uu}|_{n+1}^{k} \, \Delta \mathbf{u}_{n+1}^{k+1} = - \, \mathbf{r}_u|_{n+1}^{k} \tag{22}$$

3.2.2 Incompressible mechanics: 2-field displacement and pressure variables

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

$$r_u(\boldsymbol{u}, p; \delta\boldsymbol{u}) := \delta\mathcal{W}_{\text{kin}}(\boldsymbol{u}; \delta\boldsymbol{u}) + \delta\mathcal{W}_{\text{int}}(\boldsymbol{u}, p; \delta\boldsymbol{u}) - \delta\mathcal{W}_{\text{ext}}(\boldsymbol{u}; \delta\boldsymbol{u}) = 0, \quad \forall\, \delta\boldsymbol{u} \tag{23}$$

$$r_p(\boldsymbol{u}; \delta p) := \delta\mathcal{W}_{\text{pres}}(\boldsymbol{u}; \delta p) = 0, \quad \forall\, \delta p \tag{24}$$

– Kinetic virtual work: (13)

– Internal virtual work:

$$\delta\mathcal{W}_{\text{int}}(\boldsymbol{u}, p; \delta\boldsymbol{u}) = \int_{\Omega_0} \boldsymbol{P}(\boldsymbol{u}, p, \boldsymbol{v}(\boldsymbol{u})) : \boldsymbol{\nabla}_0 \delta\boldsymbol{u} \,\mathrm{d}V \tag{25}$$

– Pressure virtual work:

$$\delta\mathcal{W}_{\text{pres}}(\boldsymbol{u}; \delta p) = \int_{\Omega_0} (J(\boldsymbol{u}) - 1)\, \delta p \,\mathrm{d}V \tag{26}$$

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

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

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

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

4 Fluid mechanics

4.1 Eulerian reference frame

– Example: Sec. 7.2 and `demos/fluid`

– Problem type: `fluid`

– Incompressible Navier-Stokes equations in Eulerian reference frame

4.1.1 Strong Form

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

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p) + \hat{\boldsymbol{b}} = \rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla} \boldsymbol{v})\, \boldsymbol{v} \right) \qquad \text{in } \Omega_t \times [0, T], \tag{29}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega_t \times [0, T], \tag{30}$$

$$\boldsymbol{v} = \hat{\boldsymbol{v}} \qquad \text{on } \Gamma_t^{\mathrm{D}} \times [0, T], \tag{31}$$

$$\boldsymbol{t} = \boldsymbol{\sigma} \boldsymbol{n} = \hat{\boldsymbol{t}} \qquad \text{on } \Gamma_t^{\mathrm{N}} \times [0, T], \tag{32}$$

$$\boldsymbol{v}(\boldsymbol{x}, 0) = \hat{\boldsymbol{v}}_0(\boldsymbol{x}) \qquad \text{in } \Omega_t, \tag{33}$$

with a Newtonian fluid constitutive law

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\, \boldsymbol{\gamma} = -p\boldsymbol{I} + \mu \left(\boldsymbol{\nabla} \boldsymbol{v} + (\boldsymbol{\nabla} \boldsymbol{v})^{\mathrm{T}} \right) \tag{34}$$

4.1.2 Weak Form

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

– Principle of Virtual Power

$$r_v(\boldsymbol{v}, p; \delta \boldsymbol{v}) := \delta \mathcal{P}_{\mathrm{kin}}(\boldsymbol{v}; \delta \boldsymbol{v}) + \delta \mathcal{P}_{\mathrm{int}}(\boldsymbol{v}, p; \delta \boldsymbol{v}) - \delta \mathcal{P}_{\mathrm{ext}}(\boldsymbol{v}; \delta \boldsymbol{v}) = 0, \quad \forall\, \delta \boldsymbol{v} \tag{35}$$

$$r_p(\boldsymbol{v}; \delta p) := \delta \mathcal{P}_{\mathrm{pres}}(\boldsymbol{v}; \delta p), \quad \forall\, \delta p \tag{36}$$

– Kinetic virtual power:

$$\delta \mathcal{P}_{\mathrm{kin}}(\boldsymbol{v}; \delta \boldsymbol{v}) = \int_{\Omega_t} \rho \left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla} \boldsymbol{v})\, \boldsymbol{v} \right) \cdot \delta \boldsymbol{v} \, \mathrm{d}v \tag{37}$$

– Internal virtual power:

$$\delta \mathcal{P}_{\mathrm{int}}(\boldsymbol{v}, p; \delta \boldsymbol{v}) = \int_{\Omega_t} \boldsymbol{\sigma}(\boldsymbol{v}, p) : \boldsymbol{\nabla} \delta \boldsymbol{v} \, \mathrm{d}v \tag{38}$$

– Pressure virtual power:

$$\delta \mathcal{P}_{\mathrm{pres}}(\boldsymbol{v}; \delta p) = \int_{\Omega_t} (\boldsymbol{\nabla} \cdot \boldsymbol{v})\, \delta p \, \mathrm{d}v \tag{39}$$

– External virtual power:

- conservative Neumann load:

$$\delta \mathcal{P}_{\mathrm{ext}}(\delta \boldsymbol{v}) = \int_{\Gamma_t^{\mathrm{N}}} \hat{\boldsymbol{t}}(t) \cdot \delta \boldsymbol{v} \, \mathrm{d}a \tag{40}$$

- pressure Neumann load:

$$\delta\mathcal{P}_{\text{ext}}(\delta\boldsymbol{v}) = -\int\limits_{\Gamma_t^{\text{N}}} \hat{p}(t)\,\boldsymbol{n}\cdot\delta\boldsymbol{v}\,\mathrm{d}a \tag{41}$$

- body force:

$$\delta\mathcal{P}_{\text{ext}}(\delta\boldsymbol{v}) = \int\limits_{\Omega_t} \hat{\boldsymbol{b}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{42}$$

4.1.3 Stabilization

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

Full scheme according to [16]: `supg_pspg`:
– Velocity residual operator (35) is augmented with the following terms:

$$r_v \leftarrow r_v + \frac{1}{\rho}\int\limits_{\Omega_t} \tau_{\text{SUPG}}\,(\boldsymbol{\nabla}\delta\boldsymbol{v})\,\boldsymbol{v}\cdot\left[\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}\boldsymbol{v})\,\boldsymbol{v}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}(\boldsymbol{v},p)\right]\mathrm{d}v \tag{43}$$

$$+\int\limits_{\Omega_t} \tau_{\text{LSIC}}\,\rho\,(\boldsymbol{\nabla}\cdot\delta\boldsymbol{v})(\boldsymbol{\nabla}\cdot\boldsymbol{v})\,\mathrm{d}v \tag{44}$$

– Pressure residual operator (36) is augmented with the following terms:

$$r_p \leftarrow r_p + \frac{1}{\rho}\int\limits_{\Omega_t} \tau_{\text{PSPG}}\,(\boldsymbol{\nabla}\delta p)\cdot\left[\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}\boldsymbol{v})\,\boldsymbol{v}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}(\boldsymbol{v},p)\right]\mathrm{d}v \tag{45}$$

Reduced scheme (optimized for first-order): `supg_pspg2`:
– Velocity residual operator (35) is augmented with the following terms:

$$r_v \leftarrow r_v + \int\limits_{\Omega_t} d_1\,((\boldsymbol{\nabla}\boldsymbol{v})\,\boldsymbol{v})\cdot(\boldsymbol{\nabla}\delta\boldsymbol{v})\,\boldsymbol{v}\,\mathrm{d}v \tag{46}$$

$$+\int\limits_{\Omega_t} d_2\,(\boldsymbol{\nabla}\cdot\boldsymbol{v})(\boldsymbol{\nabla}\cdot\delta\boldsymbol{v})\,\mathrm{d}v \tag{47}$$

$$+\int\limits_{\Omega_t} d_3\,(\boldsymbol{\nabla}p)\cdot(\boldsymbol{\nabla}\delta\boldsymbol{v})\,\boldsymbol{v}\,\mathrm{d}v \tag{48}$$

– Pressure residual operator (36) is augmented with the following terms:

$$r_p \leftarrow r_p + \frac{1}{\rho}\int_{\Omega_t} d_1 \left((\boldsymbol{\nabla}\boldsymbol{v})\,\boldsymbol{v}\right)\cdot(\boldsymbol{\nabla}\delta p)\,\mathrm{d}v \tag{49}$$

$$+\frac{1}{\rho}\int_{\Omega_t} d_3\,(\boldsymbol{\nabla}p)\cdot(\boldsymbol{\nabla}\delta p)\,\mathrm{d}v \tag{50}$$

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

$$\mathbf{r}_{n+1} = \begin{bmatrix}\mathbf{r}_v(\mathbf{v},\mathbf{p})\\ \mathbf{r}_p(\mathbf{p},\mathbf{v})\end{bmatrix}_{n+1} = \mathbf{0} \tag{51}$$

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

$$\begin{bmatrix}\mathbf{K}_{vv} & \mathbf{K}_{vp}\\ \mathbf{K}_{pv} & \mathbf{K}_{pp}\end{bmatrix}^{k}_{n+1}\begin{bmatrix}\Delta\mathbf{v}\\ \Delta\mathbf{p}\end{bmatrix}^{k+1}_{n+1} = -\begin{bmatrix}\mathbf{r}_v\\ \mathbf{r}_p\end{bmatrix}^{k}_{n+1} \tag{52}$$

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

4.2 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 or nonlinear hyperelastic solid, displacement field $\boldsymbol{d}$

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

4.2.1 ALE problem

– Displacement-based quasi-static solid

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

– Strong form:

$$\boldsymbol{\nabla}_0\cdot\boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) = \mathbf{0} \qquad \text{in } \Omega_0, \tag{53}$$

$$\boldsymbol{d} = \hat{\boldsymbol{d}} \qquad \text{on } \Gamma_0^{\mathrm{D}}, \tag{54}$$

with

$$\boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) = E\,\frac{1}{2}\left(\boldsymbol{\nabla}_0\boldsymbol{d} + (\boldsymbol{\nabla}_0\boldsymbol{d})^{\mathrm{T}}\right) + \kappa\,(\boldsymbol{\nabla}_0 \cdot \boldsymbol{d})\;\boldsymbol{I} \tag{55}$$

– weak form:

$$r_d(\boldsymbol{d};\delta\boldsymbol{d}) := \int_{\Omega_0} \boldsymbol{\sigma}^{\mathrm{G}}(\boldsymbol{d}) : \boldsymbol{\nabla}_0\delta\boldsymbol{d}\,\mathrm{d}V = 0, \quad \forall\;\delta\boldsymbol{d} \tag{56}$$

4.2.2 Strong form

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

$$\boldsymbol{\nabla}_0\boldsymbol{\sigma}(\boldsymbol{v},\boldsymbol{d},p) : \boldsymbol{F}^{-\mathrm{T}} + \hat{\boldsymbol{b}} = \rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0\boldsymbol{v}\,\boldsymbol{F}^{-1})\,\boldsymbol{v}\right) \qquad \text{in } \Omega_0\times[0,T], \tag{57}$$

$$\boldsymbol{\nabla}_0\boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}} = 0 \qquad \text{in } \Omega_0\times[0,T], \tag{58}$$

$$\boldsymbol{v} = \hat{\boldsymbol{v}} \qquad \text{on } \Gamma_0^{\mathrm{D}}\times[0,T], \tag{59}$$

$$\boldsymbol{t} = \boldsymbol{\sigma}\boldsymbol{n} = \hat{\boldsymbol{t}} \qquad \text{on } \Gamma_0^{\mathrm{N}}\times[0,T], \tag{60}$$

$$\boldsymbol{v}(\boldsymbol{x},0) = \hat{\boldsymbol{v}}_0(\boldsymbol{x}) \qquad \text{in } \Omega_0, \tag{61}$$

with a Newtonian fluid constitutive law

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{\gamma} = -p\boldsymbol{I} + \mu\left(\boldsymbol{\nabla}_0\boldsymbol{v}\,\boldsymbol{F}^{-1} + \boldsymbol{F}^{-\mathrm{T}}(\boldsymbol{\nabla}_0\boldsymbol{v})^{\mathrm{T}}\right) \tag{62}$$

4.2.3 Weak form

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

– Principle of Virtual Power

$$r_v(\boldsymbol{v},p,\boldsymbol{d};\delta\boldsymbol{v}) := \delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v},\boldsymbol{d};\delta\boldsymbol{v}) + \delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v},p,\boldsymbol{d};\delta\boldsymbol{v}) - \delta\mathcal{P}_{\mathrm{ext}}(\boldsymbol{v},\boldsymbol{d};\delta\boldsymbol{v}) = 0, \quad \forall\;\delta\boldsymbol{v} \tag{63}$$

$$r_p(\boldsymbol{v},\boldsymbol{d};\delta p) := \delta\mathcal{P}_{\mathrm{pres}}(\boldsymbol{v},\boldsymbol{d};\delta p), \quad \forall\;\delta p \tag{64}$$

– Kinetic virtual power:

$$\delta\mathcal{P}_{\mathrm{kin}}(\boldsymbol{v},\boldsymbol{d};\delta\boldsymbol{v}) = \int_{\Omega_0} J\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + (\boldsymbol{\nabla}_0\boldsymbol{v}\,\boldsymbol{F}^{-1})\,\boldsymbol{v}\right)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{65}$$

– Internal virtual power:

$$\delta\mathcal{P}_{\mathrm{int}}(\boldsymbol{v},p,\boldsymbol{d};\delta\boldsymbol{v}) = \int_{\Omega_0} J\boldsymbol{\sigma}(\boldsymbol{v},p,\boldsymbol{d}) : \boldsymbol{\nabla}_0\delta\boldsymbol{v}\,\boldsymbol{F}^{-1}\,\mathrm{d}V \tag{66}$$

– Pressure virtual power:

$$\delta\mathcal{P}_{\text{pres}}(\boldsymbol{v},\boldsymbol{d};\delta p) = \int\limits_{\Omega_0} J\,\boldsymbol{\nabla}_0\boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}}\delta p\,\mathrm{d}V \tag{67}$$

– External virtual power:

- conservative Neumann load:

$$\delta\mathcal{P}_{\text{ext}}(\delta\boldsymbol{v}) = \int\limits_{\Gamma_0^{\mathrm{N}}} \hat{\boldsymbol{t}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}A \tag{68}$$

- pressure Neumann load:

$$\delta\mathcal{P}_{\text{ext}}(\boldsymbol{d};\delta\boldsymbol{v}) = -\int\limits_{\Gamma_0^{\mathrm{N}}} \hat{p}(t)\,J\boldsymbol{F}^{-\mathrm{T}}\boldsymbol{n}_0\cdot\delta\boldsymbol{v}\,\mathrm{d}A \tag{69}$$

- body force:

$$\delta\mathcal{P}_{\text{ext}}(\boldsymbol{d};\delta\boldsymbol{v}) = \int\limits_{\Omega_0} J\,\hat{\boldsymbol{b}}(t)\cdot\delta\boldsymbol{v}\,\mathrm{d}V \tag{70}$$

4.2.4 Stabilization

ALE forms of stabilization introduced in sec. 4.1.3
`supg_pspg`:
– Velocity residual operator (63) is augmented with the following terms:

$$r_v \leftarrow r_v + \frac{1}{\rho}\int\limits_{\Omega_0} J\,\tau_{\text{SUPG}}\left(\boldsymbol{\nabla}_0\delta\boldsymbol{v}\,\boldsymbol{F}^{-1}\right)\boldsymbol{v}\,\cdot \tag{71}$$

$$\cdot\left[\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + \left(\boldsymbol{\nabla}_0\boldsymbol{v}\,\boldsymbol{F}^{-1}\right)(\boldsymbol{v}-\boldsymbol{w})\right) - \boldsymbol{\nabla}_0\boldsymbol{\sigma}(\boldsymbol{v},\boldsymbol{d},p) : \boldsymbol{F}^{-\mathrm{T}}\right]\mathrm{d}V \tag{72}$$

$$+\int\limits_{\Omega_0} J\,\tau_{\text{LSIC}}\,\rho\,(\boldsymbol{\nabla}_0\delta\boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}})(\boldsymbol{\nabla}_0\boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}})\,\mathrm{d}V \tag{73}$$

– Pressure residual operator (64) is augmented with the following terms:

$$r_p \leftarrow r_p + \frac{1}{\rho}\int\limits_{\Omega_0} J\,\tau_{\text{PSPG}}\left(\boldsymbol{F}^{-\mathrm{T}}\boldsymbol{\nabla}_0\delta p\right)\cdot \tag{74}$$

$$\cdot\left[\rho\left(\frac{\partial\boldsymbol{v}}{\partial t} + \left(\boldsymbol{\nabla}_0\boldsymbol{v}\,\boldsymbol{F}^{-1}\right)(\boldsymbol{v}-\boldsymbol{w})\right) - \boldsymbol{\nabla}_0\boldsymbol{\sigma}(\boldsymbol{v},\boldsymbol{d},p) : \boldsymbol{F}^{-\mathrm{T}}\right]\mathrm{d}V \tag{75}$$

`supg_pspg2`:

– Velocity residual operator (63) is augmented with the following terms:

$$r_v \leftarrow r_v + \int_{\Omega_0} J\, d_1\, ((\boldsymbol{\nabla}_0 \boldsymbol{v}\, \boldsymbol{F}^{-1})\, (\boldsymbol{v} - \boldsymbol{w})) \cdot (\boldsymbol{\nabla}_0 \delta\boldsymbol{v}\, \boldsymbol{F}^{-1})\, \boldsymbol{v}\, \mathrm{d}V \tag{76}$$

$$+ \int_{\Omega_0} J\, d_2\, (\boldsymbol{\nabla}_0 \boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}})(\boldsymbol{\nabla}_0 \delta\boldsymbol{v} : \boldsymbol{F}^{-\mathrm{T}})\, \mathrm{d}V \tag{77}$$

$$+ \int_{\Omega_0} J\, d_3\, (\boldsymbol{F}^{-\mathrm{T}} \boldsymbol{\nabla}_0 p) \cdot (\boldsymbol{\nabla}_0 \delta\boldsymbol{v}\, \boldsymbol{F}^{-1})\, \boldsymbol{v}\, \mathrm{d}V \tag{78}$$

– Pressure residual operator (64) is augmented with the following terms:

$$r_p \leftarrow r_p + \frac{1}{\rho} \int_{\Omega_0} J\, d_1\, ((\boldsymbol{\nabla}_0 \boldsymbol{v}\, \boldsymbol{F}^{-1})\, (\boldsymbol{v} - \boldsymbol{w})) \cdot (\boldsymbol{F}^{-\mathrm{T}} \boldsymbol{\nabla}_0 \delta p)\, \mathrm{d}V \tag{79}$$

$$+ \frac{1}{\rho} \int_{\Omega_0} J\, d_3\, (\boldsymbol{F}^{-\mathrm{T}} \boldsymbol{\nabla}_0 p) \cdot (\boldsymbol{F}^{-\mathrm{T}} \boldsymbol{\nabla}_0 \delta p)\, \mathrm{d}V \tag{80}$$

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

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{d}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{81}$$

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

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{vd} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{K}_{pd} \\ \mathbf{K}_{dv} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}^{k}_{n+1} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{d} \end{bmatrix}^{k+1}_{n+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_d \end{bmatrix}^{k}_{n+1} \tag{82}$$

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

5 0D flow: Lumped parameter models

– Example: Sec. 7.3 `demos/flow0d`

– Problem type: `flow0d`

5.1 2-element Windkessel

- Model type : `2elwindkessel`

5.2 4-element Windkessel (inertance parallel to impedance)

- Model type : `4elwindkesselLpZ`

5.3 4-element Windkessel (inertance serial to impedance)

- Model type : `4elwindkesselLsZ`

5.4 Systemic and pulmonary circulation

- Model type : `syspul`

- Allows to link in a coronary flow model

left heart and systemic circulation

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

$$q^{\ell}_{\mathrm{v,in}} = q_{\mathrm{mv}}(p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}}) \qquad \text{mitral valve momentum} \qquad (83)$$

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

$$q^{\ell}_{\mathrm{v,out}} = q_{\mathrm{av}}(p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}}) \qquad \text{aortic valve momentum} \qquad (84)$$

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

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

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

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

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

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

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

right heart and pulmonary circulation

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

$$q^r_{\mathrm{v,in}} = q_{\mathrm{tv}}(p^r_{\mathrm{at}} - p^r_{\mathrm{v}}) \qquad \text{tricuspid valve momentum} \tag{85}$$

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

$$q^r_{\mathrm{v,out}} = q_{\mathrm{pv}}(p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}}) \qquad \text{pulmonary valve momentum} \tag{86}$$

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

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

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

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

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

with:

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

and:

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

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

$$V(t) = \frac{p}{E(t)} + V_{\mathrm{u}}, \tag{87}$$

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

$$E(t) = (E_{\max} - E_{\min}) \cdot \hat{y}(t) + E_{\min}, \tag{88}$$

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. (83), (84), (85), (86), are functions of the pressure difference $p - p_{\mathrm{open}}$ across the valve. The following valve models can be defined:
Valve model `pwlin_pres`:

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

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

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

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

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

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

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

with

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

and

$$h_{00} = 2s^3 - 3s^2 + 1, \qquad h_{01} = -2s^3 + 3s^2,$$
$$h_{10} = s^3 - 2s^2 + s, \qquad h_{11} = s^3 - s^2$$

with

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

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

Valve model `pw_pres_regurg`:

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

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:

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

$$R_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,p}}^{\mathrm{sys},\ell}=p_{\mathrm{ar}}^{\mathrm{sys}}-p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-Z_{\mathrm{cor,p}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,p,in}}^{\mathrm{sys},\ell}$$

$$C_{\mathrm{cor,d}}^{\mathrm{sys},\ell}\frac{\mathrm{d}(p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-p_{\mathrm{v}}^{\ell})}{\mathrm{d}t}=q_{\mathrm{cor,p}}^{\mathrm{sys},\ell}-q_{\mathrm{cor,d}}^{\mathrm{sys},\ell}$$

$$R_{\mathrm{cor,d}}^{\mathrm{sys},\ell}\,q_{\mathrm{cor,d}}^{\mathrm{sys},\ell}=p_{\mathrm{cor,d}}^{\mathrm{sys},\ell}-p_{\mathrm{at}}^{r}$$

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

$$R_{\mathrm{cor,p}}^{\mathrm{sys},r}\,q_{\mathrm{cor,p}}^{\mathrm{sys},r}=p_{\mathrm{ar}}^{\mathrm{sys}}-p_{\mathrm{cor,d}}^{\mathrm{sys},r}-Z_{\mathrm{cor,p}}^{\mathrm{sys},r}\,q_{\mathrm{cor,p,in}}^{\mathrm{sys},r}$$

$$C_{\mathrm{cor,d}}^{\mathrm{sys},r}\frac{\mathrm{d}(p_{\mathrm{cor,d}}^{\mathrm{sys},r}-p_{\mathrm{v}}^{\ell})}{\mathrm{d}t}=q_{\mathrm{cor,p}}^{\mathrm{sys},r}-q_{\mathrm{cor,d}}^{\mathrm{sys},r}$$

$$R_{\mathrm{cor,d}}^{\mathrm{sys},r}\,q_{\mathrm{cor,d}}^{\mathrm{sys},r}=p_{\mathrm{cor,d}}^{\mathrm{sys},r}-p_{\mathrm{at}}^{r}$$

$$0=q_{\mathrm{cor,d}}^{\mathrm{sys},\ell}+q_{\mathrm{cor,d}}^{\mathrm{sys},r}-q_{\mathrm{cor,d,out}}^{\mathrm{sys}}$$

5.5 Systemic and pulmonary circulation, including capillary flow

- Model type : `syspulcap`

$$
\begin{aligned}
&- Q^{\ell}_{\mathrm{at}} = q^{\mathrm{pul}}_{\mathrm{ven}} - q^{\ell}_{\mathrm{v,in}} \\
&\tilde{R}^{\ell}_{\mathrm{v,in}}\, q^{\ell}_{\mathrm{v,in}} = p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}} \\
&- Q^{\ell}_{\mathrm{v}} = q^{\ell}_{\mathrm{v,in}} - q^{\ell}_{\mathrm{v,out}} \\
&\tilde{R}^{\ell}_{\mathrm{v,out}}\, q^{\ell}_{\mathrm{v,out}} = p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}} \\
&0 = q^{\ell}_{\mathrm{v,out}} - q^{\mathrm{sys}}_{\mathrm{ar,p}} \\
&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}} \\
&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}} \\
&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}} \\
&\left(\sum_{j \in \{ {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer,cor}} \}} 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,cor}} \}} q^{\mathrm{sys}}_{\mathrm{ar},j} \\
&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,cor}} \} \\
&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,cor}} \} \\
&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,cor}} \} \\
&C^{\mathrm{sys}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{sys}}_{\mathrm{ven}}}{\mathrm{d}t} = \sum_{j = {}^{\mathrm{spl,espl,}}_{\mathrm{msc,cer,cor}}} q^{\mathrm{sys}}_{\mathrm{ven},j} - q^{\mathrm{sys}}_{\mathrm{ven}} \\
&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}}
\end{aligned}
$$

$$
\begin{aligned}
&- Q^r_{\mathrm{at}} = q^{\mathrm{sys}}_{\mathrm{ven}} - q^r_{\mathrm{v,in}} \\
&\tilde{R}^r_{\mathrm{v,in}}\, q^r_{\mathrm{v,in}} = p^r_{\mathrm{at}} - p^r_{\mathrm{v}} \\
&- Q^r_{\mathrm{v}} = q^r_{\mathrm{v,in}} - q^r_{\mathrm{v,out}} \\
&\tilde{R}^r_{\mathrm{v,out}}\, q^r_{\mathrm{v,out}} = p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}} \\
&C^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} = q^r_{\mathrm{v,out}} - q^{\mathrm{pul}}_{\mathrm{ar}} \\
&L^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ar}}\, q^{\mathrm{pul}}_{\mathrm{ar}} = p^{\mathrm{pul}}_{\mathrm{ar}} - p^{\mathrm{pul}}_{\mathrm{cap}} \\
&C^{\mathrm{pul}}_{\mathrm{cap}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{cap}}}{\mathrm{d}t} = q^{\mathrm{pul}}_{\mathrm{ar}} - q^{\mathrm{pul}}_{\mathrm{cap}} \\
&R^{\mathrm{pul}}_{\mathrm{cap}}\, q^{\mathrm{pul}}_{\mathrm{cap}} = p^{\mathrm{pul}}_{\mathrm{cap}} - p^{\mathrm{pul}}_{\mathrm{ven}} \\
&C^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} = q^{\mathrm{pul}}_{\mathrm{cap}} - q^{\mathrm{pul}}_{\mathrm{ven}} \\
&L^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ven}}\, q^{\mathrm{pul}}_{\mathrm{ven}} = p^{\mathrm{pul}}_{\mathrm{ven}} - p^{\ell}_{\mathrm{at}}
\end{aligned}
$$

with:

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

5.6 Systemic and pulmonary circulation, including capillary and coronary flow

- Model type : `syspulcapcor`

$$
\begin{aligned}
&-Q^{\ell}_{\mathrm{at}} = q^{\mathrm{pul}}_{\mathrm{ven}} - q^{\ell}_{\mathrm{v,in}} \\
&\tilde{R}^{\ell}_{\mathrm{v,in}}\, q^{\ell}_{\mathrm{v,in}} = p^{\ell}_{\mathrm{at}} - p^{\ell}_{\mathrm{v}} \\
&-Q^{\ell}_{\mathrm{v}} = q^{\ell}_{\mathrm{v,in}} - q^{\ell}_{\mathrm{v,out}} \\
&\tilde{R}^{\ell}_{\mathrm{v,out}}\, q^{\ell}_{\mathrm{v,out}} = p^{\ell}_{\mathrm{v}} - p^{\mathrm{sys}}_{\mathrm{ar}} \\
&0 = q^{\ell}_{\mathrm{v,out}} - q^{\mathrm{sys}}_{\mathrm{ar,p}} - q^{\mathrm{sys}}_{\mathrm{ar,cor,in}} \\
&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}} \\
&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}} \\
&R^{\mathrm{sys}}_{\mathrm{ar,cor}}\, q^{\mathrm{sys}}_{\mathrm{ar,cor}} = p^{\mathrm{sys}}_{\mathrm{ar}} - p^{\mathrm{sys}}_{\mathrm{ven,cor}} \\
&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}} \\
&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}} \\
&\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} \\
&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}} \} \\
&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}} \} \\
&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}} \} \\
&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}} \\
&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}} \\
&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}} \\
&R^{\mathrm{sys}}_{\mathrm{ven,cor}}\, q^{\mathrm{sys}}_{\mathrm{ven,cor}} = p^{\mathrm{sys}}_{\mathrm{ven,cor}} - p^{r}_{\mathrm{at}}
\end{aligned}
$$

$$
\begin{aligned}
&-Q^r_{\mathrm{at}} = q^{\mathrm{sys}}_{\mathrm{ven}} + q^{\mathrm{sys}}_{\mathrm{ven,cor}} - q^r_{\mathrm{v,in}} \\
&\tilde{R}^r_{\mathrm{v,in}}\, q^r_{\mathrm{v,in}} = p^r_{\mathrm{at}} - p^r_{\mathrm{v}} \\
&-Q^r_{\mathrm{v}} = q^r_{\mathrm{v,in}} - q^r_{\mathrm{v,out}} \\
&\tilde{R}^r_{\mathrm{v,out}}\, q^r_{\mathrm{v,out}} = p^r_{\mathrm{v}} - p^{\mathrm{pul}}_{\mathrm{ar}} \\
&C^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} = q^r_{\mathrm{v,out}} - q^{\mathrm{pul}}_{\mathrm{ar}} \\
&L^{\mathrm{pul}}_{\mathrm{ar}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ar}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ar}}\, q^{\mathrm{pul}}_{\mathrm{ar}} = p^{\mathrm{pul}}_{\mathrm{ar}} - p^{\mathrm{pul}}_{\mathrm{cap}} \\
&C^{\mathrm{pul}}_{\mathrm{cap}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{cap}}}{\mathrm{d}t} = q^{\mathrm{pul}}_{\mathrm{ar}} - q^{\mathrm{pul}}_{\mathrm{cap}} \\
&R^{\mathrm{pul}}_{\mathrm{cap}}\, q^{\mathrm{pul}}_{\mathrm{cap}} = p^{\mathrm{pul}}_{\mathrm{cap}} - p^{\mathrm{pul}}_{\mathrm{ven}} \\
&C^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}p^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} = q^{\mathrm{pul}}_{\mathrm{cap}} - q^{\mathrm{pul}}_{\mathrm{ven}} \\
&L^{\mathrm{pul}}_{\mathrm{ven}} \frac{\mathrm{d}q^{\mathrm{pul}}_{\mathrm{ven}}}{\mathrm{d}t} + R^{\mathrm{pul}}_{\mathrm{ven}}\, q^{\mathrm{pul}}_{\mathrm{ven}} = p^{\mathrm{pul}}_{\mathrm{ven}} - p^{\ell}_{\mathrm{at}}
\end{aligned}
$$

with:

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

5.7 Systemic and pulmonary circulation, capillary flow, and respirory model

- Model type : `syspulcaprespir`

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

6 Multi-physics coupling

6.1 Solid + 0D flow

– Example: Sec. 7.4 and `demos/solid_flow0d`

– Problem type: `solid_flow0d`

– (12) or (23) augmented by following term:

$$
r_u \leftarrow r_u + \int\limits_{\Gamma_0^{\text{s-0d}}} \Lambda\, J \boldsymbol{F}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot \delta\boldsymbol{u}\, \mathrm{d}A \tag{89}
$$

– Multiplier constraint

$$r_{\Lambda}(\Lambda, \boldsymbol{u}; \delta\Lambda) := \left(\int\limits_{\Gamma_0^{\text{s-0d}}} J\boldsymbol{F}^{-\mathrm{T}}\boldsymbol{n}_0 \cdot \boldsymbol{v}(\boldsymbol{u}) \,\mathrm{d}A - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall\, \delta\Lambda \tag{90}$$

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

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u(\mathbf{u}, \boldsymbol{\Lambda}) \\ \mathbf{r}_{\Lambda}(\boldsymbol{\Lambda}, \mathbf{u}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{91}$$

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

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{u\Lambda} \\ \mathbf{K}_{\Lambda u} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\boldsymbol{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_{\Lambda} \end{bmatrix}_{n+1}^{k} \tag{92}$$

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

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_u(\mathbf{u}, \mathbf{p}, \boldsymbol{\Lambda}) \\ \mathbf{r}_p(\mathbf{u}) \\ \mathbf{r}_{\Lambda}(\boldsymbol{\Lambda}, \mathbf{u}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{93}$$

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

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} & \mathbf{K}_{u\Lambda} \\ \mathbf{K}_{pu} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\Lambda u} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{p} \\ \Delta\boldsymbol{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_p \\ \mathbf{r}_{\Lambda} \end{bmatrix}_{n+1}^{k} \tag{94}$$

6.2 Fluid + 0D flow

– Example: Sec. 7.5 `demos/fluid_flow0d`

– Problem type: `fluid_flow0d`

– (35) augmented by following term:

$$r_v \leftarrow r_v + \int\limits_{\Gamma_t^{\text{f-0d}}} \Lambda\, \boldsymbol{n} \cdot \delta\boldsymbol{v} \,\mathrm{d}a \tag{95}$$

– Multiplier constraint

$$r_\Lambda(\Lambda, \boldsymbol{v}; \delta\Lambda) := \left(\int_{\Gamma_t^{\text{f-0d}}} \boldsymbol{n} \cdot \boldsymbol{v} \, \mathrm{d}a - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall \, \delta\Lambda \tag{96}$$

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

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{\Lambda}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}) \\ \mathbf{r}_\Lambda(\mathbf{\Lambda}, \mathbf{v}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{97}$$

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

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\Lambda} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} \\ \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{\Lambda} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_\Lambda \end{bmatrix}_{n+1}^{k} \tag{98}$$

6.3 ALE fluid + 0D flow

– Problem type: `fluid_ale_flow0d`

– (63) augmented by following term:

$$r_v \leftarrow r_v + \int_{\Gamma_0^{\text{f-0d}}} \Lambda \, J \boldsymbol{F}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot \delta\boldsymbol{v} \, \mathrm{d}A \tag{99}$$

– Multiplier constraint

$$r_\lambda(\Lambda, \boldsymbol{v}, \boldsymbol{d}; \delta\Lambda) := \left(\int_{\Gamma_0^{\text{f-0d}}} J \boldsymbol{F}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot (\boldsymbol{v} - \boldsymbol{w}(\boldsymbol{d})) \, \mathrm{d}A - Q^{\text{0d}}(\Lambda) \right) \delta\Lambda, \quad \forall \, \delta\Lambda \tag{100}$$

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

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

$$\mathbf{r}_{n+1} = \begin{bmatrix} \mathbf{r}_v(\mathbf{v}, \mathbf{p}, \mathbf{\Lambda}, \mathbf{d}) \\ \mathbf{r}_p(\mathbf{p}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_\Lambda(\mathbf{\Lambda}, \mathbf{v}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}) \end{bmatrix}_{n+1} = \mathbf{0} \tag{101}$$

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

$$\begin{bmatrix} \mathbf{K}_{vv} & \mathbf{K}_{vp} & \mathbf{K}_{v\Lambda} & \mathbf{K}_{vd} \\ \mathbf{K}_{pv} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\Lambda v} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} & \mathbf{K}_{\Lambda d} \\ \mathbf{K}_{dv} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{p} \\ \Delta\mathbf{\Lambda} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_\Lambda \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k} \tag{102}$$

6.4 Fluid-Solid Interaction (FSI)

– Problem type: `fsi`

– Not yet fully implemented!

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

– Problem type: `fsi_flow0d`

– Not yet fully implemented!

7 Demos

7.1 Solid

– Physics description given in sec. 3

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
```

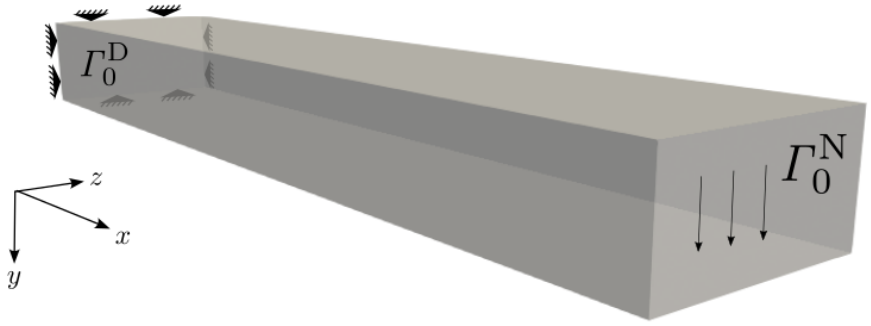

Figure 1: Cantilever, problem setup.

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.

7.2 Fluid

– Physics description given in sec. 4

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.

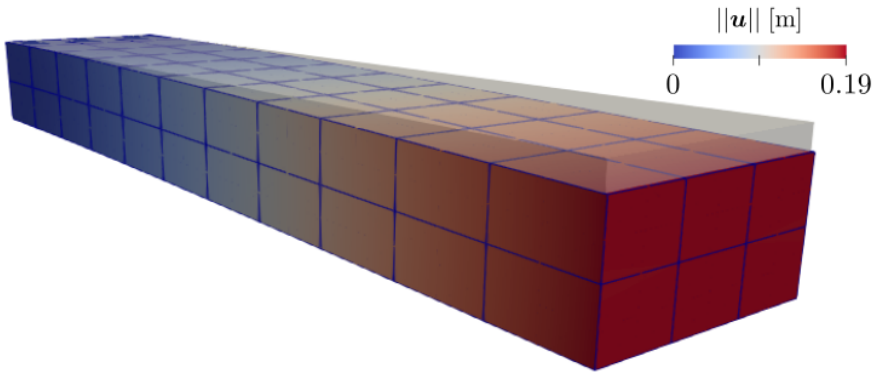

Figure 2: Cantilever, tip deformation. Color shows displacement magnitude.

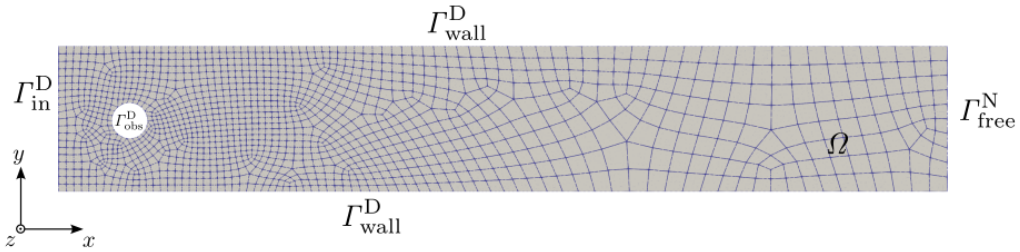

Figure 3: Channel flow, problem setup.

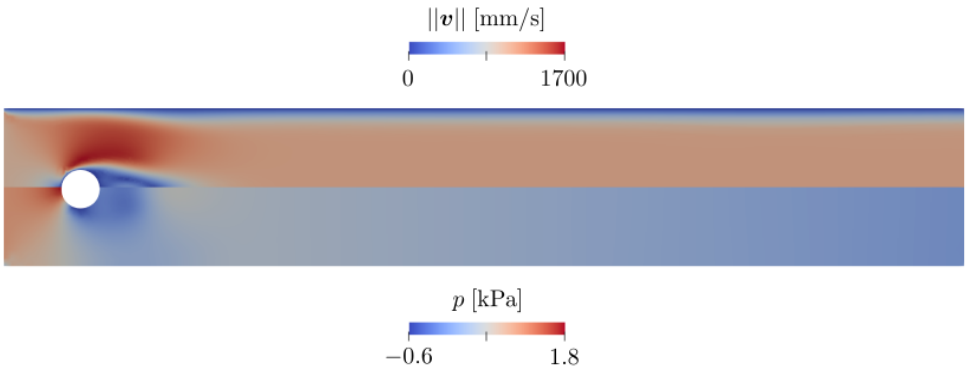

Figure 4: Velocity magnitude (top part) and pressure (bottom part) at end of simulation.

7.3 0D flow

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 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.

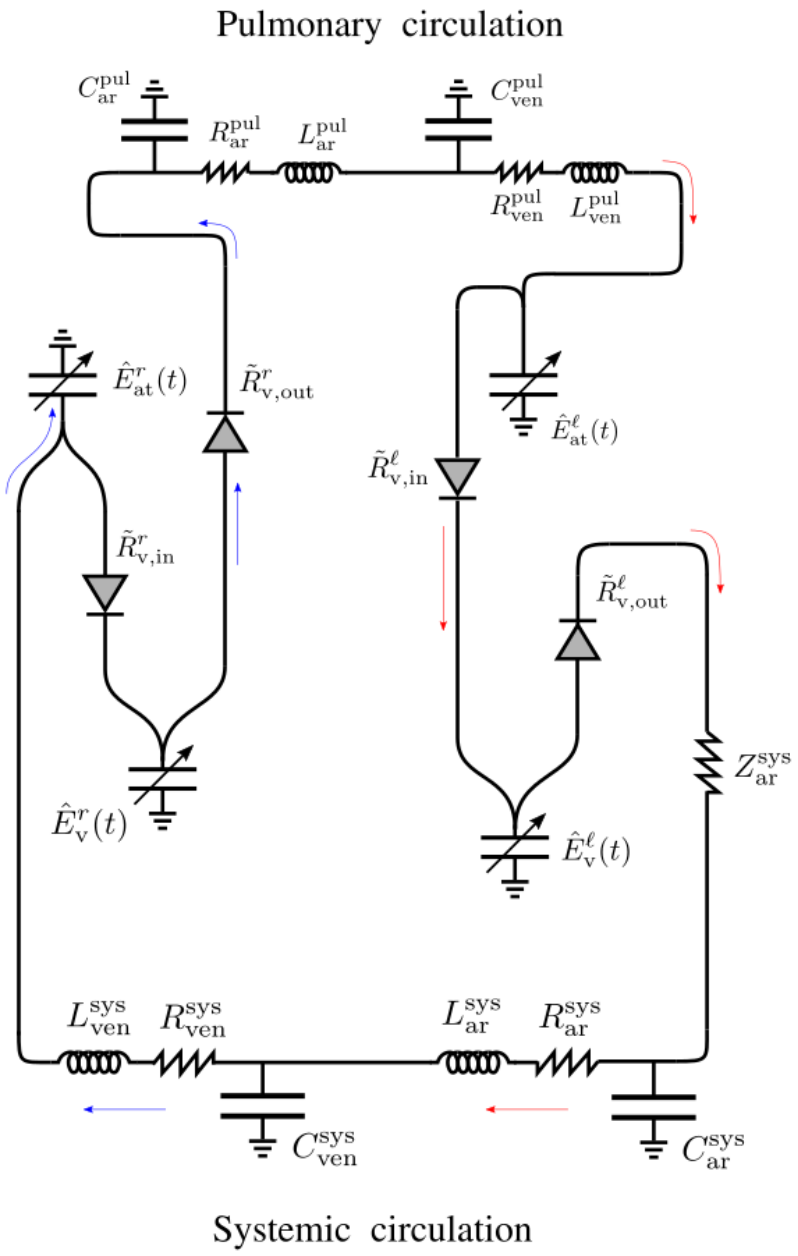

Figure 5: 0D heart, systemic and pulmonary circulation, problem setup.

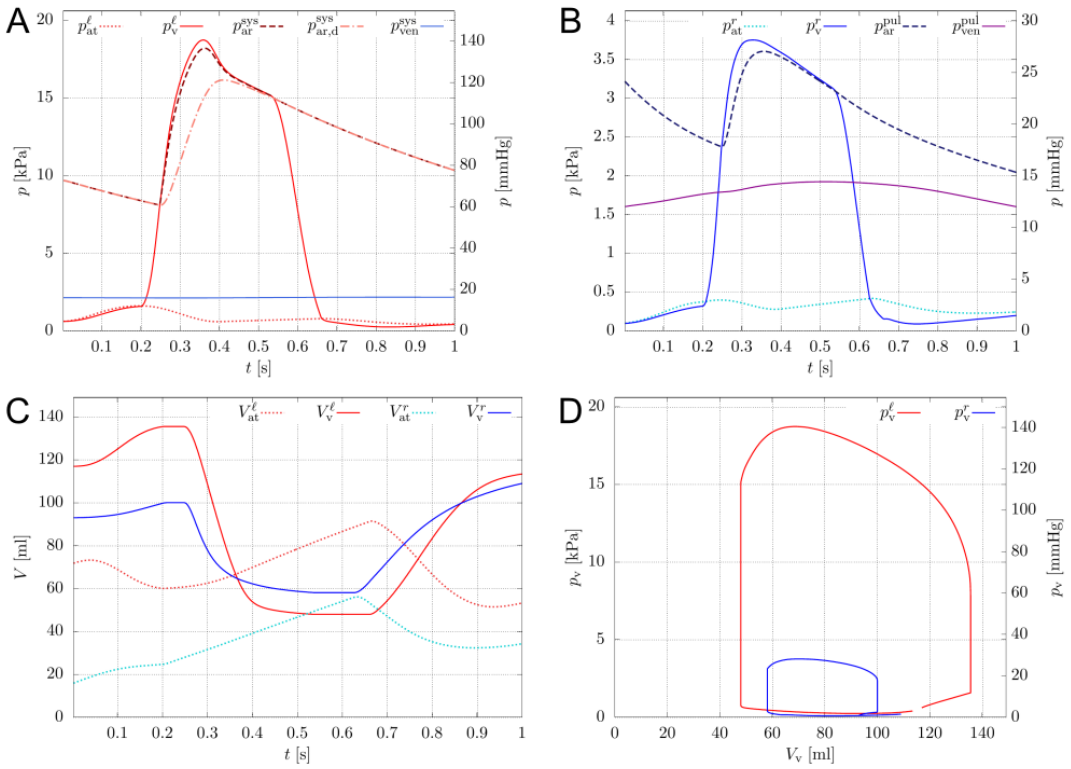

Figure 6: A. Left heart and systemic pressures over time. B. Right heart and pulmonary pressures over time. C. Left and right ventricular and atrial volumes over time. D. Left and right ventricular pressure-volume relationships of periodic (5th) cycle.

7.4 Solid + 0D flow

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 [13] and a strain rate-dependent viscous model [3]. We start the simulation with "prestressing" using the MULF method [5, 15], 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
```

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

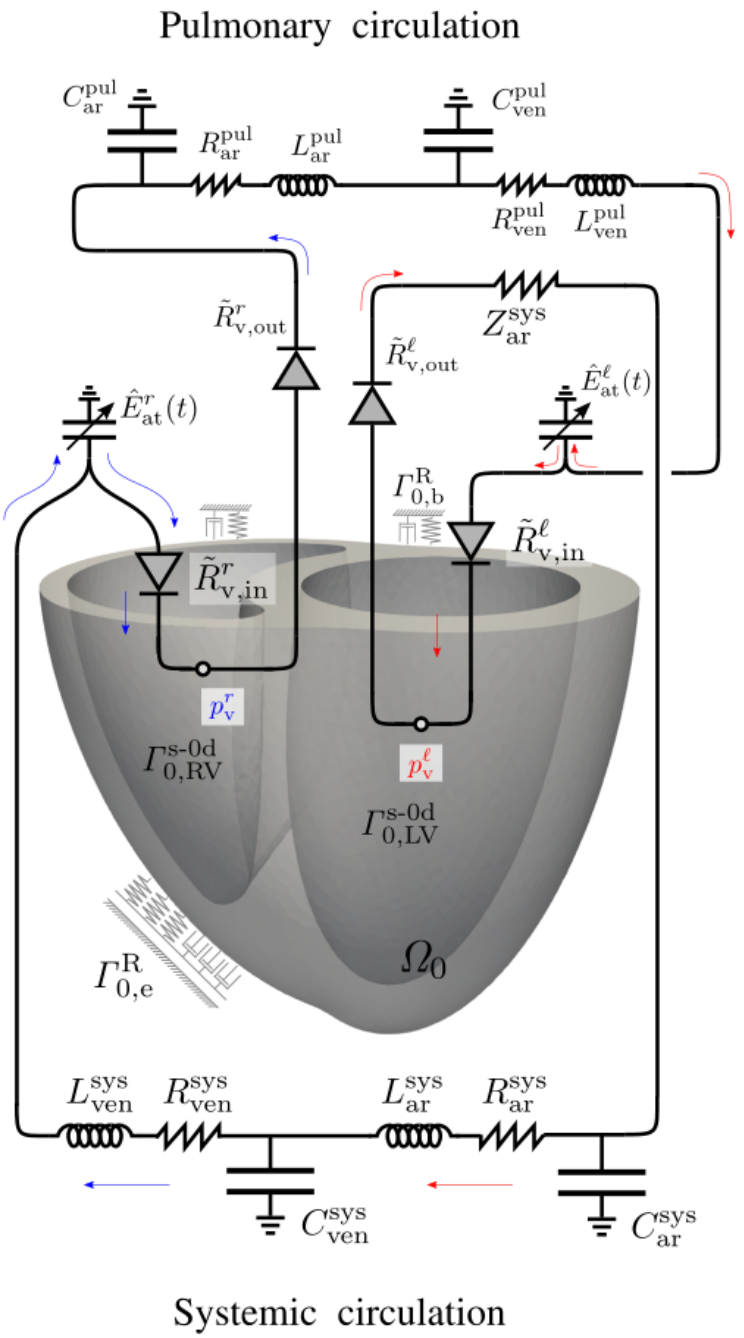

Figure 7: Generic 3D ventricular heart model coupled to a closed-loop systemic and pulmonary circulation model.

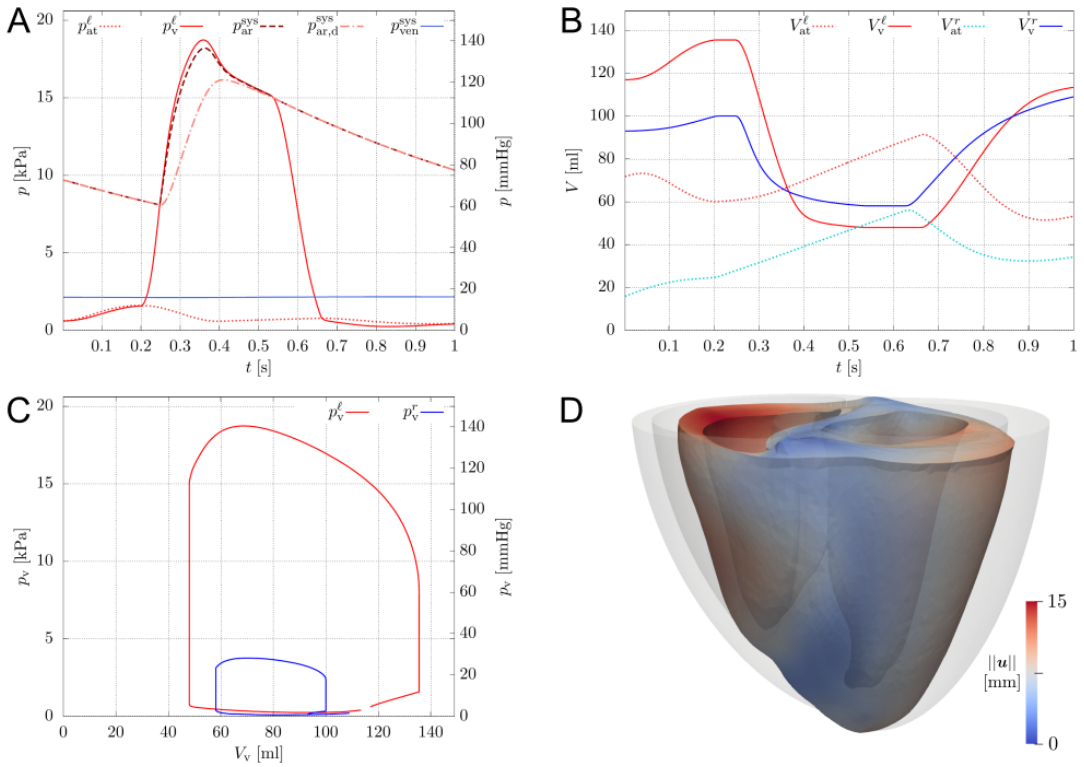

Figure 8: A. Left heart and systemic pressures over time. B. Left and right ventricular and atrial volumes over time. C. Left and right ventricular pressure-volume relationships. D. Snapshot of heart deformation at end-systole, color indicates displacement magnitude.

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?

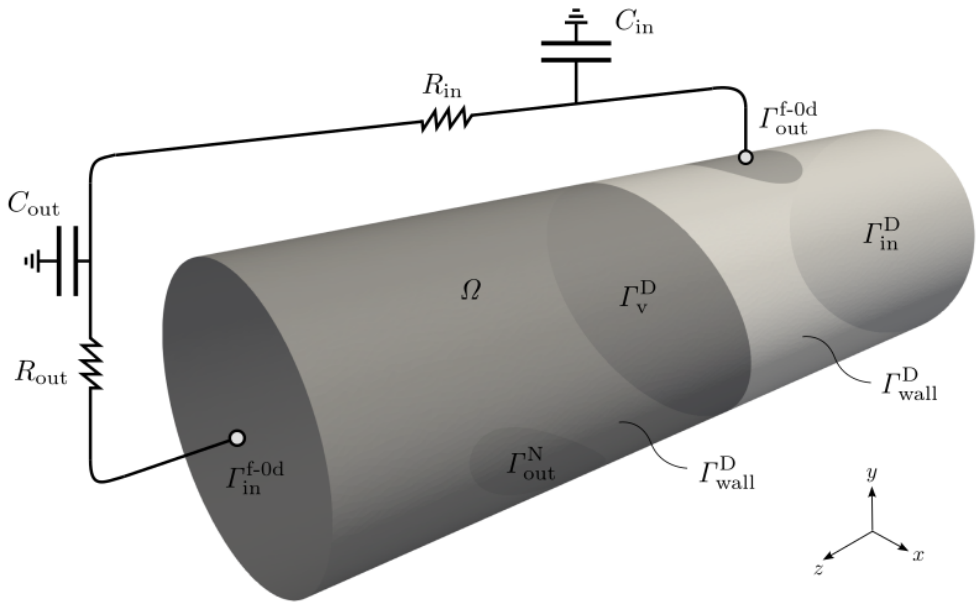

Figure 9: Blocked pipe with 0D model bypass, simulation setup.

7.5 Fluid + 0D flow

Blocked pipe flow with 0D model bypass

This example demonstrates how to couple 3D fluid flow to a 0D lumped-parameter model. Incompressible transient Navier-Stokes flow in a pipe with prescibed 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 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
```

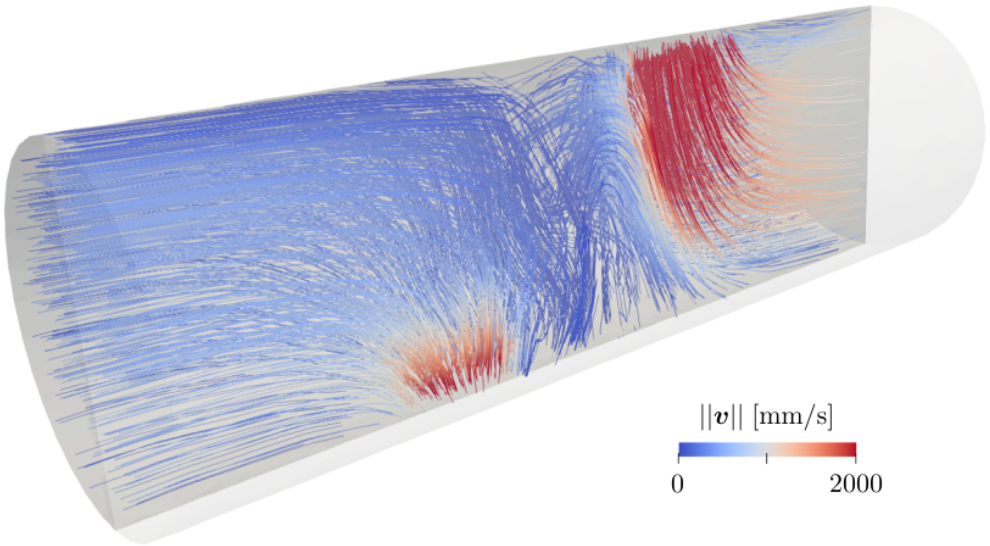

Figure 10: Streamlines of velocity at end of simulation, color indicates velcity magnitude.

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 $\Gamma_{\text{in}}^{\text{f-0d}}$), b) almost the same flow across $\Gamma_{\text{out}}^{\text{f-0d}}$ and $\Gamma_{\text{in}}^{\text{f-0d}}$. 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.

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, 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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