PACS Project

Coupled problems on one-dimensional networks with application to microcirculation

Advisor:
Prof. Paolo ZUNINO

Stefano BRAMBILLA
Matr. 853558

Academic Year 2017-2018
Abstract

The aim of this project is to develop a finite element solver for diffusion-advection-reaction models for microcirculation.
We employ dimensional model reduction techniques in order to describe the vessel network as a one-dimensional manifold immersed in a three-dimensional interstitial volume. These techniques allow us to consider the microcirculation network as a concentrated source in the 3D domain.
The outline of this paper follows: Section 1 describes the general mathematical framework of the coupled multiscale problem. Section 2 contains some numerical results from different benchmarks, in order to validate our solver. Finally, Section 3 presents the implementation of the C++ code and provides a detailed description of the use of the Finite Element library GetFEM++. 
Contents

1 Mathematical formulation .................................................. 2
   1.1 Dimension analysis ..................................................... 4
   1.2 Boundary and initial conditions ..................................... 5
   1.3 Weak formulation ..................................................... 6
      1.3.1 Weak formulation for the tissue problem ...................... 6
      1.3.2 Weak formulation for the vessel problem ...................... 7
   1.4 Numerical Approximation ............................................. 9
   1.5 Algebraic formulation ............................................... 11
      1.5.1 Coupling terms ................................................... 12

2 Numerical Results ............................................................ 14
   2.1 Uncoupled 1D and 3D test ............................................ 14
      2.1.1 Stand-alone vessel network problem ......................... 14
      2.1.2 Stand-alone tissue problem .................................... 16
   2.2 Coupled 3D-1D problem on a single branch ....................... 18
   2.3 Test on a single bifurcation ........................................ 20
      2.3.1 Uncoupled problem on the single bifurcation network ....... 21
      2.3.2 Coupled 3D-1D problem with a single bifurcation .......... 23
   2.4 Coupled 3D-1D problem on a complex network .................... 25

3 C++ Code ............................................................................ 28
   3.1 Design of the code ..................................................... 28
   3.2 Assembling routines .................................................... 32
   3.3 The library transport3d1d ............................................ 40
   3.4 Doxygen documentation .............................................. 41

Bibliography ......................................................................... 43
Chapter 1

Mathematical formulation

We present a mathematical model for mass transport for a capillary network immersed in a permeable biological tissue. The domain of our model is a subspace of $\mathbb{R}^3$ composed by two parts, namely $\Omega_t$ and $\Omega_v$, the interstitial tissue and the vessel bed respectively. We assume that the capillaries can be described as cylindrical vessels, with $\Gamma$ defined as the outer surface of $\Omega_v$ and $\Lambda$ as the one-dimensional manifold representing the centerline of the capillary network. The vessel radius $R$ may change in the network.

The physical quantity of interest is the concentration of transported solutes $c(x,t)$. In particular, we denote $c_v(x,t)$ the concentration in the domain $\Lambda$, and $c_t(x,t)$ the concentration in the domain $\Omega_t$.

To model drug transport we assume that the molecules are advected by the fluid and diffuse in all the domain. Furthermore, chemical species can be metabolized by the cells in the interstitial tissue. We assume that the capillary walls behave as semipermeable membranes: there is both a leakage of fluid and a selective filtration of molecules in the vessel walls. We also consider the lymphatic system, which absorbs the fluid in excess due to capillary leakage; we assume the volumetric flow rate to be proportional to the pressure difference between the interstitium and the lymphatic system.

Therefore, we model transport by means of advection-diffusion equations in the net-

Figure 1.1: On the left the permeable tissue perfused by a single capillary; on the right the dimensional reduction from 3D to 1D description of the vessel.
work, and by means of advection-diffusion-reaction equations in the volume.

\[
\frac{\partial c_t}{\partial t} + \nabla \cdot (c_t \mathbf{u}_t - D_t \nabla c_t) + m c_t + L_P^L F S (p_t - p_L) \frac{\partial c_t}{\partial t} = f_c \delta \Lambda \quad \text{on } \Omega \times (0, T),
\]

\[
\frac{\partial c_v}{\partial t} + \frac{\partial}{\partial s} \left( (\mathbf{u}_v \cdot \mathbf{\lambda}) c_v - D_v \frac{\partial c_v}{\partial s} \right) = -\frac{1}{\pi R^2} f_c \quad \text{on } \Lambda \times (0, T),
\]

where \(D_t\) and \(D_v\) are the molecular diffusivities, in the tissue and in the vessels, respectively, assumed to be constant in each region. The metabolism rate \(m\) can be modeled as a function of concentration values, but we consider it constant at this stage.

In the lymphatic drainage term, \(L_{LP}^F\) is the idraulic permeability of the lymphatic wall, the ratio \(S\) is the surface area of lymphatic vessels per unit volume of tissue and \(p_L\) is the hydrostatic pressure within the lymphatic channels.

The function \(f_c = f_c(\bar{p}_t, p_v, \bar{c}_t, c_v)\) represents the mass flux per unit length of capillary vessels, between the capillary bed and the interstitial tissue; this function is written according to the Kedem-Katchalsky equations, which describes the behaviour of a selective semipermeable membrane. This reads as:

\[
f_c(\bar{p}_t, p_v, \bar{c}_t, c_v) = 2\pi R \left[ L_p (1 - \sigma) [(p_v - \bar{p}_t) - \sigma (\pi_v - \pi_t)] [w \bar{c}_t + (1 - w)c_v] + P(c_v - \bar{c}_t) \right]
\]

(1.2)

on \(\Lambda \times (0, T)\), where \(L_p\) is the hydraulic permeability of the vessel wall, \(\sigma\) is the osmotic reflection coefficient, \(\sigma\) is the sieving coefficient, \(p_v - \bar{p}_t\) is the pressure drop between vessels and tissue, \(\pi_v - \pi_t\) is the oncoytic pressure jump; finally, \(0 < w < 1\) is a weight depending on the Péclet number of the solute transport through the wall, usually set \(w = \frac{1}{2}\), and \(P\) is the permeability of the vessel wall with respect to solutes. We recall that we defined \(\Lambda\) as the one-dimensional manifold representing the centerline of the capillary network, as in Figure 1.1, we would then be inappropriate to use the pointwise \(c_t\) on \(\Lambda\) in order to compute the coupling term. In fact, we employ the average value of concentration over the cylindrical surface of the real domain of the capillary vessel, \(\Gamma\). We introduce the average value of a field \(g\):

\[
\bar{g}(s) := \frac{1}{2\pi R} \int_0^{2\pi} g(s, \theta) R d\theta.
\]

(1.3)

We apply the average operator to every quantity of interest in the mass flux \(f_c(\bar{p}_t, p_v, \bar{c}_t, c_v)\) which is defined on the volume \(\Omega_t\), namely \(c_t\), \(p_t\) and \(\pi_t\).

We finally model the advection and pressure fields to be the solutions of the equations for the fluid dynamics of the blood. We consider the tissue an isotropic porous medium and the blood flow in the vessels to be steady and incompressible: then, the advection field is described by Darcy’s law in the interstitial volume and by Poiseuille’s flow in the microcirculation. Then, the velocity fields \(u_t\) and \(u_v\), and the hydrostatic pressures \(p_t\) and \(p_v\) satisfy the following equations:
\[-\nabla \cdot \left( \frac{k}{\mu} \nabla p_t \right) + L_P^F \frac{S}{V} (p_t - p_L) - f_b (p_L, p_v) \delta_\Lambda = 0 \quad \text{in } \Omega \times (0, T), \]

\[u_t = - \frac{k}{\mu} \nabla p_t \quad \text{in } \Omega \times (0, T), \]

\[-\frac{\pi R^4}{8 \mu} \partial^2 p_v \partial s^2 + f_b (p_L, p_v) = 0 \quad \text{on } \Lambda \times (0, T), \]

\[u_v = - \frac{R^2}{8 \mu} \partial p_v \partial s \quad \text{on } \Lambda \times (0, T), \]

where \(k\) is the hydraulic permeability of the interstitial volume, \(\mu\) is the blood viscosity and the function \(f_b (p_L, p_v) \delta_\Lambda\) denotes the fluid flux leaking from the capillaries to the interstitial volume, defined as:

\[f_b (p_L, p_v) : = 2\pi RL_p ((p_v - p_L) - \sigma (\pi_v - \pi_l)) \quad \Lambda \times (0, T). \]  

**1.1 Dimension analysis**

In this section we want to perform a dimension analysis in order to study the relative impact of the different mechanics, namely the molecular diffusion, advection, consumption and leakage. The physical variables on which our analysis will depend are length, velocity, pressure and concentration; the characteristic length is the average spacing between capillary vessels \(d\), the characteristic velocity is the average velocity in the capillary bed \(U\), the characteristic pressure is the average pressure in the interstitial space \(\Delta P\), and the characteristic concentration is the maximum value \(C_{\text{max}}\) of concentration of that chemical species we can find in an healthy organism. The adimensional parameters follow:

\[R' = \frac{R}{d}, \quad \text{non-dimensional radius}, \]

\[A_t = \frac{D_t}{dU}, \quad \text{inverse of Péclet number in the interstitium}, \]

\[A_v = \frac{D_v}{dU}, \quad \text{inverse of Péclet number in the blood stream}, \]

\[D_\alpha = \frac{m}{dU}, \quad \text{Damkohler number}, \]

\[Q = \frac{L_P \Delta P}{U}, \quad \text{hydraulic conductivity of the capillary walls}, \]

\[Q_{PL} = \frac{L_P^F S d \Delta P}{V U}, \quad \text{hydraulic conductivity of the lymphatic walls}, \]

\[\Upsilon = \frac{P}{U}, \quad \text{magnitude of leakage from the capillary bed}, \]

\[\kappa_t = \frac{k \Delta P}{\mu Ud}, \quad \text{hydraulic conductivity of the tissue}, \]

\[\kappa_v = \frac{\pi R^4 d \Delta P}{8 \mu U}, \quad \text{hydraulic conductivity of the capillary bed}. \]
Using these parameters, the equations \([1.1]\), \([1.2]\) and \([1.4]\) read as follows:

\[
\frac{\partial c_t}{\partial t} + \nabla \cdot (c_t u_t - A_t \nabla c_t) + D \alpha c_t + Q_{PL} (p_t - p_L) c_t = f_c \delta \Lambda \quad \text{in } \Omega \times (0, T),
\]

\[
\frac{\partial c_v}{\partial t} + \frac{\partial}{\partial s} \left( u_v c_t - A_v \frac{\partial c_v}{\partial s} \right) = -\frac{1}{\pi R^2} f_c \quad \text{on } \Lambda \times (0, T),
\]

\[
f_c (p_t, p_v, c_t, c_v) = 2\pi R' \left[ Q (1 - \sigma) [(p_v - \bar{p}_t) - \sigma (\pi_v - \pi_t)] [w \bar{c}_t + (1 - w)c_v] + \Upsilon (c_v - \bar{c}_v) \right]
\]

\[
\nabla \cdot u_t + Q_{PL} (p_t - p_L) - Q (p_v - \bar{p}_t) \delta \Lambda = 0 \quad \text{in } \Omega \times (0, T),
\]

\[
\frac{1}{\kappa_t} u_t + \nabla p_t = 0 \quad \text{in } \Omega \times (0, T),
\]

\[
\frac{\partial u_v}{\partial s} + \frac{1}{\pi R^2} Q (p_t - \bar{p}_t) = 0 \quad \text{on } \Lambda \times (0, T),
\]

\[
\frac{\pi R^2}{\kappa_v} u_v + \frac{\partial p_v}{\partial s} = 0 \quad \text{on } \Lambda \times (0, T).
\]

For simplicity, the variables for concentration, velocity and pressure have maintained the same symbol after the re-scaling for \(C_{max}, U\) and \(\Delta P\) respectively. However, note that the transport equations are linear in the variables \(c_t\) and \(c_v\), so the adimensionalization of this quantities does not affect the equations.

Besides, we used the notation \(u_v = (u_v \cdot \lambda)\) for the velocity in the one-dimensional vessels.

### 1.2 Boundary and initial conditions

In order to guarantee the uniqueness of the solution of the dimensionless problem, we need to specify some boundary conditions on \(\partial \Omega\) and \(\partial \Lambda\), and an initial condition for \(t = 0\). The choice of the following boundary conditions depends on both the variational formulation and the available data. In fact, we must impose essential and natural conditions over the boundary integrals derived from integrations by parts.

The boundary of the vessel network is the set of the capillary extrema; we distinguish these points between inflow and outflow extrema, respectively the points with velocity inward-pointing and outward-pointing; we define these sets \(\partial \Lambda \equiv \Lambda^{IN} \cup \Lambda^{OUT}\). We claim to be a constant concentration of the chemical species on the inflow boundary; this condition is linked to a constant injection of the solute in the network. On the outflow boundary, the particles are free to leave the system: a homogeneous Neumann boundary condition correctly models this phenomenon. The conditions are:

\[
c_v = c_{in} \quad \text{on } \partial \Lambda^{IN} \times (0, T),
\]

\[
\frac{\partial c_v}{\partial s} = 0 \quad \text{on } \partial \Lambda^{OUT} \times (0, T).
\]
At the boundary of the volume $\Omega$, we have flow of fluid with particles; the quantity of solution exchanged with the exterior depends on the concentration of the solution itself. We model this condition with a Robin condition:

$$-A_t \nabla c_t \cdot n = \beta_t (c_t - c_{out}) \quad \text{on } \partial \Omega \times (0, T).$$

**Remark 1.** We described the physical boundary condition that we need to close the problem; nevertheless, we developed a more general framework for boundary conditions in the code, allowing the user to test different cases. Therefore, from now on we will use this boundary conditions:

\begin{align*}
  c_v &= c_{in} \quad \text{on } \partial \Lambda^{IN} \times (0, T), \\
  -A_v \frac{\partial c_v}{\partial s} &= \beta_v (c_v - c_{0,v}) \quad \text{on } \partial \Lambda^{OUT} \times (0, T), \\
  c_t &= c_{ext} \quad \text{on } \partial \Omega^{DIR} \times (0, T), \\
  -A_t \nabla c_t \cdot n &= \beta_t (c_t - c_{0,t}) \quad \text{on } \partial \Omega^{MIX} \times (0, T).
\end{align*}

The choice of the parameters defines all the standard boundary conditions, namely Dirichlet, Neumann and Robin conditions.

Finally, we recall that our problem is time-dependent, so we must impose an initial condition over all domain:

\begin{align*}
  c_t (t = 0) &= c_{in,t} \quad \text{on } \Omega, \\
  c_v (t = 0) &= c_{in,v} \quad \text{on } \Lambda.
\end{align*}

### 1.3 Weak formulation

The coupled problem we want to solve has no explicit analytical solution, so we must solve it numerically, through finite element methods; therefore we must write the variational form. In particular, in this section we will propose a dual mixed weak formulation of both the tissue and vessel problems.

#### 1.3.1 Weak formulation for the tissue problem

To obtain a variational formulation of the particle transport problem in the interstitial tissue, the test space for the concentration is

$$Q_t := H^1_{0, \partial \Omega^{DIR}}(\Omega).$$

Let us proceed multiplying (1.6)(a) with a sufficiently smooth function $q_t$ and inte-
grating over $\Omega$:

$$
\int_{\Omega} \frac{\partial c_t}{\partial t} q_t \, d\Omega + \int_{\Omega} \nabla \cdot (c_t u_t - A_t \nabla c_t) q_t \, d\Omega + \int_{\Omega} D_\alpha c_t q_t \, d\Omega + \int_{\Omega} Q^{PL} (p_t - p_L) c_t q_t \, d\Omega

- \int_{\Omega} 2\pi R' \{ (1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] w - \Upsilon \} \delta_\Lambda c_t q_t \, d\Omega

= + \int_{\Omega} 2\pi R' \{ (1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] (1 - w) + \Upsilon \} \delta_\Lambda c_v q_t \, d\Omega.
$$

We apply the Green’s theorem and the boundary condition to the diffusion term:

$$
\int_{\Omega} \nabla \cdot (-A_t \nabla c_t) q_t \, d\Omega = \int_{\Omega} A_t \nabla c_t \cdot \nabla q_t \, d\Omega - \int_{\partial\Omega} A_t q_t \nabla c_t \cdot n \, d\sigma

= \int_{\Omega} A_t \nabla c_t \cdot \nabla q_t \, d\Omega - \int_{\partial\Omega^{DIR}} A_t q_t \nabla c_t \cdot n \, d\sigma + \int_{\partial\Omega^{MIX}} \beta_t c_t q_t \, d\sigma - \int_{\partial\Omega^{MIX}} \beta_t c_0 q_t \, d\sigma
$$

We re-write the advection term in non-conservative form:

$$
\int_{\Omega} \nabla (u_t c_t) q_t \, d\Omega = \int_{\Omega} u_t \cdot \nabla c_t q_t \, d\Omega + \int_{\Omega} \nabla \cdot u_t c_t q_t \, d\Omega
$$

Finally, we substitute these two results into this equation:

$$
\int_{\Omega} \frac{\partial c_t}{\partial t} q_t \, d\Omega + \int_{\Omega} A_t \nabla c_t \cdot \nabla q_t \, d\Omega + \int_{\partial\Omega^{MIX}} \beta_t c_t q_t \, d\sigma - \int_{\partial\Omega^{MIX}} \beta_t c_0 q_t \, d\sigma

+ \int_{\Omega} u_t \cdot \nabla c_t q_t \, d\Omega + \int_{\Omega} \nabla \cdot u_t c_t q_t \, d\Omega

- \int_{\Omega} 2\pi R' \{ (1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] w - \Upsilon \} \delta_\Lambda c_t q_t \, d\Omega

= + \int_{\Omega} 2\pi R' \{ (1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] (1 - w) + \Upsilon \} \delta_\Lambda c_v q_t \, d\Omega.
$$

$\forall q_t \in H^1_{\Omega, \partial \Omega^{DIR}}(\Omega)$.

Notice that the boundary term on the Dirichlet boundary vanishes, thanks to the choice of the space of the test functions: the Dirichlet boundary condition will be enforced in an essential way, directly on the matrix.

### 1.3.2 Weak formulation for the vessel problem

Accounting the vessel problem, the test space for the concentration is:

$$
Q_v := H^1_{0, \partial \Omega^{IN}}(\Lambda)
$$
We approach the weak formulation in a standard way, by means of multiplying equation (1.6) (b) by a test function $q_v$ and integrating over $\Lambda$:

$$
\int_{\Lambda} \frac{\partial c_v}{\partial t} q_v \, ds + \int_{\Lambda} \frac{\partial}{\partial s} \left( u_v c_v - A_v \frac{\partial c_v}{\partial s} \right) q_v \, ds
$$

$$
+ \int_{\Lambda} \frac{2}{\pi R} \left\{ (1 - \sigma) (p_v - \bar{p}_v) - \sigma (\pi_v - \bar{\pi}_v) \right\} c_v q_v \, ds =
$$

$$
= - \int_{\Lambda} \frac{2}{\pi R} \left\{ (1 - \sigma) (p_v - \bar{p}_v) - \sigma (\pi_v - \bar{\pi}_v) \right\} w - \Upsilon \right\} c_v q_v \, ds.
$$

At this point, it is not possible to simply integrate by parts, because of the presence of multiple junctions. To tackle this issue, must divide the integral over the whole network between each branch; in order to handle the terms at the junctions, we write the mass balance at each junction, that is:

$$
\sum_{i \in P_{j}^{out}} \pi R_i^2 \left( -A_v \frac{\partial c_v}{\partial s} + u_v c_v \right) \bigg|_{\Lambda_j^+} = \sum_{i \in P_{j}^{in}} \pi R_i^2 \left( -A_v \frac{\partial c_v}{\partial s} + u_v c_v \right) \bigg|_{\Lambda_j^-}, \quad \forall j \in J,
$$

(1.11)

where $J$ is the set of all the junctions, $P_{j}^{out}$ and $P_{j}^{in}$ is the set of the indexes of the branches exiting and entering the junction $j$, $\Lambda_j^+$ and $\Lambda_j^-$ denote the inflow and outflow extrema of branch $\Lambda_j$. Notice that, for the choice of the functional space, $c_v$ must be continuous in every point; in addition, the conservation of local flow rate in every vessel junctions must hold. For this reasons, the mass balance of the advective fluxes holds separately from the mass balance of the diffusive fluxes:

$$
\sum_{i \in P_{j}^{out}} \pi R_i^2 u_v c_v \bigg|_{\Lambda_j^+} = \sum_{i \in P_{j}^{in}} \pi R_i^2 u_v c_v \bigg|_{\Lambda_j^-}, \quad \forall j \in J,
$$

(1.12)

$$
\sum_{i \in P_{j}^{out}} \pi R_i^2 A_v \frac{\partial c_v}{\partial s} \bigg|_{\Lambda_j^+} = \sum_{i \in P_{j}^{in}} \pi R_i^2 A_v \frac{\partial c_v}{\partial s} \bigg|_{\Lambda_j^-}, \quad \forall j \in J.
$$

(1.13)

For this reason, before we integrate by parts, we multiply the equation by a factor $\pi R^2$; in this way, the diffusive and advective term are correctly scaled by the cross section area.

We can now apply the Green’s theorem to the diffusive term:

$$
\int_{\Lambda} \frac{\partial}{\partial s} \left( -\pi R^2 A_v \frac{\partial c_v}{\partial s} \right) q_v \, d\Lambda = \sum_{i} \int_{\Lambda_i} \frac{\partial}{\partial s} \left( -\pi R_i^2 A_v \frac{\partial c_v}{\partial s} \right) q_v \, d\Lambda
$$

$$
= \sum_{i} \left\{ \int_{\Lambda_i} \pi R_i^2 A_v \frac{\partial c_v}{\partial s} \frac{\partial q_v}{\partial s} \, d\Lambda + \left[ -\pi R_i^2 A_v \frac{\partial c_v}{\partial s} q_v \right]_{\Lambda_i^+}^{\Lambda_i^-} \right\}
$$

$$
= \int_{\Lambda} \pi R^2 A_v \frac{\partial c_v}{\partial s} \frac{\partial q_v}{\partial s} \, d\Lambda + \sum_{i} \left[ -\pi R_i^2 A_v \frac{\partial c_v}{\partial s} q_v \right]_{\Lambda_i^+}^{\Lambda_i^-}.
$$

The extrema terms can be distinguished in boundary terms and junction terms; thanks to the compatibility condition of the diffusive mass balance, the junction terms vanish. Finally the diffusive term reads as:
\[
\int_{A} \frac{\partial}{\partial s} \left( -\pi R^2 A_v \frac{\partial c_v}{\partial s} \right) q_v d\Lambda = \int_{A} \pi R^2 A_v \frac{\partial}{\partial s} c_v q_v d\Lambda - [\pi R^2 A_v \frac{\partial c_v}{\partial s}]_{\partial \Lambda^{IN}} + [\pi R^2 \beta_v c_v q_v]_{\partial \Lambda^{OUT}} - [\pi R^2 \beta_v c_{0,v} q_v]_{\partial \Lambda^{OUT}}.
\]

As done in the tissue terms, we write in a non conservative form the advection term:

\[
\int_{A} \pi R^2 \frac{\partial}{\partial t} (u_v c_v) q_v ds = \int_{A} \pi R^2 u_v \frac{\partial c_v}{\partial s} q_v ds + \int_{A} \pi R^2 \frac{\partial u_v}{\partial s} c_v q_v ds
\]

(1.14)

Globally, combining the previous results, the weak formulation for the vessel problem is

\[
\int_{A} \pi R^2 \frac{\partial c_v}{\partial t} q_v ds + \int_{A} \pi R^2 A_v \frac{\partial}{\partial s} c_v q_v d\Lambda + [\pi R^2 \beta_v (c_v - c_{0,v}) q_v]_{\partial \Lambda^{OUT}} + \int_{A} 2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] (1 - w) + \Upsilon \} c_v q_v ds
\]

(1.15)

\[= - \int_{A} 2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] w - \Upsilon \} \bar{q}_t q_v ds.\]

Therefore, the whole problem is: find \(c_t \in Q_t \times (0,T)\) and \(c_v \in Q_v \times (0,T)\) such that

\[
\begin{align*}
\left\{ \begin{array}{l}
(\frac{\partial c_t}{\partial t}, q_t)_{\Omega} + (A_v \nabla c_t, \nabla q_t)_{\Omega} + (\mathbf{u}_t \cdot \nabla c_t, q_t)_{\Omega} + (\nabla \cdot \mathbf{u}_t c_t, q_t)_{\Omega} \\
+ (Q^{LF} (p_t - p_L) c_t, b_t)_{\Omega} + (\beta c_t, q_t)_{\partial \Omega} + \\
- (2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] w - \Upsilon \} \bar{c}_t, q_t)_{\Lambda} + \\
- (2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] (1 - w) + \Upsilon \} c_v, q_t)_{\Lambda} \\
= - (\beta c_{0,v} q_t)_{\partial \Lambda^{IN}} \forall q_t \in Q_t, (1.16) \\
\end{array} \right. \\
+ \left( \pi R^2 \frac{\partial c_v}{\partial t}, q_v \right)_{\Lambda} + (\pi R^2 u_v \frac{\partial c_v}{\partial s}, q_v)_{\Lambda} + (\pi R^2 \frac{\partial u_v}{\partial s} c_v, q_v)_{\Lambda} + \\
+ \left( \pi R^2 A_v \frac{\partial c_v}{\partial s}, \frac{\partial c_v}{\partial s} \right)_{\Lambda} + [\pi R^2 \beta_v c_v q_v]_{\partial \Lambda^{OUT}} \\
+ (2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] (1 - w) + \Upsilon \} c_v, q_v)_{\Lambda} + \\
+ (2\pi R' \{(1 - \sigma) Q [(p_v - \bar{p}_t) - \sigma (\pi_v - \bar{\pi}_t)] w - \Upsilon \} \bar{c}_v, q_v)_{\Lambda} \\
= - [\pi R^2 \beta_v c_{0,v} q_v]_{\partial \Lambda^{OUT}} \forall q_v \in Q_v,
\end{align*}
\]

with \(c_t(t = 0) = 0\) and \(c_v(t = 0) = 0\).

### 1.4 Numerical Approximation

We proceed with the description of finite element implementation of our problem. First of all, we will discretize the domains \(\Omega\) and \(\Lambda\); next, we will define the discrete functional spaces for our solutions; finally, we will derive the Galerkin formulation for problem (1.16).

Our formulation let us discretize independently the partitions \(\Omega\) and \(\Lambda\).
In order to discretize the domain for the tissue interstitium problem, we introduce an admissible triangulation $\mathcal{T}^h_t$ of $\bar{\Omega}$, i.e.

$$\bar{\Omega} = \bigcup_{K \in \mathcal{T}^h_t} K,$$

which satisfies the usual conditions of a conforming triangulation of $\Omega$, while we are implicitly assuming that $\Omega$ is a polygonal domain. With a standard notation, $h = \max_{K \in \mathcal{T}^h_t} h_K$, where $h_K$ is the diameter of the element $K$. The solutions of (1.16)(a) are approximated using continuous piecewise-polynomial finite elements for the concentration. More precisely, we have

$$X^h_k := \{ v_h \in C^0(\bar{\Omega}) \text{ s.t. } v_h|_K \in P_k(K) \quad \forall K \in \mathcal{T}^h_t \}$$

for every integer $k \geq 0$, where $P_k$ indicates the standard space of polynomials of degree $\leq k$ in the variables $x = (x_1, \ldots, x_d)$.

Concerning the capillary network problem, we adopt the same splitting of the domain described at the continuous level, denoted by

$$\Lambda_h = \bigcup_{i=1}^N \Lambda^h_i,$$

where $\Lambda^h_i$ is a partition of the one-dimensional manifold $\Lambda_i$ made by segments $S$. For the concentration, we define the finite element space over the whole network $\Lambda_h$ as

$$Y^h_k(\Lambda) := \{ w_h \in C^0(\bar{\Lambda}) \text{ s.t. } w_h|_S \in P_k(S) \quad \forall S \in \Lambda_h \} ,$$

for every integer $k \geq 0$.

The discrete formulation arising from (1.16) is easily obtained by projecting the equations on the discrete spaces

$$Q^h_t = X^h_k(\Omega) \quad \text{and} \quad Q^h_v = Y^h_k(\Lambda)$$

for $k \geq 0$ and adding the subscript $h$ to each variable ($c^h_t$ and $c^h_v$).

The space discretization must be complemented with the time advancing scheme. Let us subdivide the time interval $[0, T]$ in $N$ time steps of size $\Delta t > 0$, so that $t^n = n\Delta t$, with $n = 0, \ldots, N - 1$. The equations have been solved with the backward Euler finite difference scheme:

$$\frac{\partial y}{\partial t} = f(y) \quad \Rightarrow \quad \frac{y^{n+1} - y^n}{\Delta t} = f(y^{n+1}).$$

Let us denote with $c^{h,n}_t$ and $c^{h,n}_v$, the numerical approximation of $c^h_t(t^n)$ and $c^h_v(t^n)$, respectively, therefore the fully discrete formulation of problem (1.16) reads as follows:
∀ \, n = 0, \ldots, N - 1, to find \, c^{h,n+1}_{t} \in B^{h}_{t} \text{ and } c^{h,n+1}_{v} \in B^{h}_{v} \text{ such that}

\[\begin{align*}
\frac{1}{\Delta t} \left( c^{h,n+1}_{t}, q^{h}_{t} \right)_{\Omega} + \left( A_{t} \nabla c^{h,n+1}_{t}, \nabla q^{h}_{t} \right)_{\Omega} + \left( D_{t} c^{h,n+1}_{t}, q^{h}_{t} \right)_{\Omega} \\
+ \left( u^{h}_{t}, \nabla c^{h,n+1}_{t}, q^{h}_{t} \right)_{\Omega} + \left( \nabla \cdot u^{h}_{t}, c^{h,n+1}_{t}, q^{h}_{t} \right)_{\Omega} + \left( \beta c^{h,n+1}_{t}, q^{h}_{t} \right)_{\partial \Omega^{MIX}} - \left( 2 \pi R \{ (1 - \sigma) Q \left[ \left( p^{h}_{t} - \bar{p}^{h}_{t} \right) - \sigma \left( \pi^{h}_{v} - \bar{\pi}^{h}_{t} \right) \right] w - \Upsilon \right) c^{h,n+1}_{t}, q^{h}_{t} \right)_{\Lambda} \\
- \left( 2 \pi R \{ (1 - \sigma) Q \left[ \left( p^{h}_{t} - \bar{p}^{h}_{t} \right) - \sigma \left( \pi^{h}_{v} - \bar{\pi}^{h}_{t} \right) \right] (1 - w) + \Upsilon \right) c^{h,n+1}_{v}, q^{h}_{t} \right)_{\Lambda} \\
= \frac{1}{\Delta t} \left( c^{h,n}_{t}, q^{h}_{t} \right)_{\Omega} - \left( \beta c^{0,0}_{t}, q^{h}_{t} \right)_{\partial \Omega^{MIX}} \\
+ \left( 2 \pi R \{ \left( 1 - \sigma \right) Q \left[ \left( p^{h}_{t} - \bar{p}^{h}_{t} \right) - \sigma \left( \pi^{h}_{v} - \bar{\pi}^{h}_{t} \right) \right] \right) \left( 1 - w \right) + \Upsilon \right) c^{h,n+1}_{v}, q^{h}_{t} \right)_{\Lambda} \\
+ \left( 2 \pi R \{ \left( 1 - \sigma \right) Q \left[ \left( p^{h}_{t} - \bar{p}^{h}_{t} \right) - \sigma \left( \pi^{h}_{v} - \bar{\pi}^{h}_{t} \right) \right] \right) (-w) - \Upsilon \right) c^{h,n+1}_{v}, q^{h}_{t} \right)_{\Lambda} \\
= \frac{1}{\Delta t} \left( \pi R^{2} \bar{c}^{h,n}_{t}, q^{h}_{t} \right)_{\Lambda} - \left[ \pi R^{2} \beta \varepsilon c^{0,0}_{t}, q^{h}_{t} \right]_{\partial \Omega^{MIX}} \\
\forall q^{h}_{t} \in Q^{h}_{v}, \quad (1.17)
\end{align*}\]

with \( c^{h,0}_{t} = 0 \) and \( c^{h,0}_{v} = 0 \).

### 1.5 Algebraic formulation

We aim at studying the algebraic counterpart of the discrete problem \[ (1.17) \]. The number of degrees of freedom of the discrete spaces are defined as

\[ N^{h}_{t} := dim \left( Q^{h}_{t} \right) \quad \text{and} \quad N^{h}_{v} := dim \left( Q^{h}_{v} \right), \]

Let us introduce the finite element basis for \( Q^{h}_{t} \) and \( Q^{h}_{v} \): \( \{ \varphi_{i}^{t} \}_{i=1}^{N^{h}_{t}} \) and \( \{ \varphi_{j}^{v} \}_{j=1}^{N^{h}_{v}} \), respectively. These two sets are completely independent, since the 3D and 1D meshes do not conform.

Let \( C^{n}_{t} = \{ C^{j,n}_{t} \}_{j=1}^{N^{h}_{t}} \) and \( C^{n}_{v} = \{ C^{j,n}_{v} \}_{j=1}^{N^{h}_{v}} \) be the degrees of freedom of the finite element approximation, using the finite element basis it is possible to set:

\[ c^{h,n}_{t} \left( \mathbf{x} \right) = \sum_{j=1}^{N^{h}_{t}} C^{j,n}_{t} \varphi_{j}^{t} \left( \mathbf{x} \right), \quad \forall \mathbf{x} \in \Omega \quad \text{and} \quad c^{h,n}_{v} \left( s \right) = \sum_{j=1}^{N^{h}_{v}} C^{j,n}_{v} \varphi_{j}^{v} \left( s \right), \quad \forall s \in \Lambda. \]

Exploiting the linear combinations in the discrete weak form and the linearity of the inner products, the fully discrete form \[ (1.17) \] of the model leads to the linear system

\[\begin{bmatrix}
\frac{1}{\Delta t} M_{t} + A_{t} + B_{tt} \\
B_{vt}
\end{bmatrix}
\begin{bmatrix}
C^{n+1}_{t} \\
C^{n+1}_{v}
\end{bmatrix}
= \frac{1}{\Delta t} M_{t} C^{n}_{t} + F_{t} + \frac{1}{\Delta t} M_{v} C^{n}_{v} + F_{v}. \quad (1.18)\]
Submatrices and subvectors in (1.18) are defined as follows:

\[
[M_t]_{i,j} := (\varphi_i^t, \varphi_j^t)_{\Omega},
[A_t]_{i,j} := (A_t \nabla \varphi_i^t, \nabla \varphi_j^t)_{\Omega} + (u_i^h \cdot \nabla \varphi_i^t, \varphi_j^t)_{\Omega} + (\nabla \cdot u_i^h \varphi_i^t, \varphi_j^t)_{\Omega} + (D_o \varphi_i^t, \varphi_j^t)_{\Omega} + (\beta \varphi_i^t, \varphi_j^t)_{\partial \Omega_{mix}},
\]

\[
[M_v]_{i,j} := (\pi R^2 \varphi_i^v, \varphi_j^v)_{\Lambda},
[A_v]_{i,j} := 
\left(\pi R^2 \frac{\partial \varphi_i^v}{\partial s} \frac{\partial \varphi_j^v}{\partial s}\right)_{\Lambda} + 
\left(\pi R^2 u_i^h \frac{\partial \varphi_j^v}{\partial s}, \varphi_j^v\right)_{\Lambda} + 
\left(\pi R^2 \frac{\partial u_i^h}{\partial s} \varphi_j^v, \varphi_j^v\right)_{\Lambda}
\]

\[
[B_t]_{i,j} := -2\pi R' \left\{(1 - \sigma) Q \left[\left(p_i^h - \bar{p}_i^h\right) - \sigma (\bar{\pi}_v^h - \bar{\pi}_v^h)\right] w - \gamma\right\} \varphi_i^t, \varphi_j^t\right\}_{\Lambda},
\]

\[
[B_v]_{i,j} := 
\left(2\pi R' \left\{(1 - \sigma) Q \left[\left(p_i^h - \bar{p}_i^h\right) - \sigma (\bar{\pi}_v^h - \bar{\pi}_v^h)\right] w - \gamma\right\} \varphi_i^v, \varphi_j^v\right\}_{\Lambda},
\]

\[
[F_t]_{i,k} := -\left(\beta c_0 \varphi_i^t\right)_{\partial \Omega_{out}},
[F_v]_{i,k} := -\left(\pi R^2 \beta c_0 v \varphi_i^v\right)_{\partial \Omega_{out}},
\]

where the bar operator corresponds to the average operator as in (1.3). In particular, it holds

\[
M_t \in \mathbb{R}^{N_t^h \times N_t^h}, \quad A_t \in \mathbb{R}^{N_t^h \times N_t^h}, \quad B_t \in \mathbb{R}^{N_t^h \times N_t^h}, \quad B_{tv} \in \mathbb{R}^{N_t^h \times N_v^h}, \quad F_t \in \mathbb{R}^{N_t^h},
M_v \in \mathbb{R}^{N_v^h \times N_v^h}, \quad A_v \in \mathbb{R}^{N_v^h \times N_v^h}, \quad B_{tv} \in \mathbb{R}^{N_v^h \times N_v^h}, \quad B_{vt} \in \mathbb{R}^{N_v^h \times N_t^h}, \quad F_v \in \mathbb{R}^{N_v^h}
\]

### 1.5.1 Coupling terms

Concerning the implementation of the exchange matrices $B_{tv}, B_{vt}$ and $F_{vt}$, it is necessary to introduce a discrete average operator $\bar{\pi}_v : Q_t^h \to Q_t^h$ that extracts the mean value of a generic basis function of $Q_t^h$ and a discrete interpolation operator $\pi_{tv} : Q_t^h \to Q_v^h$ that returns the value of a basis function of $Q_t^h$ in correspondence of nodes of $Q_v^h$.

For every node $s_k \in \Lambda_h$, we let $T_\gamma (s_k)$ be the discretization of the perimeter of the vessel $\gamma (s_k)$, assuming that $\gamma (s_k)$ is a circle of radius $R$ defined on the orthogonal plane to $\Lambda_h$ at point $s_k$ (see Figure 1.2). The set of points of $T_\gamma (s_k)$ is used to interpolate the basis function $\varphi_i^t$. The average operator $\bar{\pi}_v$ is defined in such a way that $\bar{\pi}_v = \pi_v q_t$ and each row of the corresponding matrix $\Pi_{vt} \in \mathbb{R}^{N_v^h \times N_t^h}$ is defined as

\[
\bar{\pi}_v|_k = w^T (s_k) \Pi_\gamma (s_k) \quad k = 1, \ldots, N_v^h,
\]

where $w$ is the vector of weights of the quadrature formula for the approximation of $\bar{\pi}_v|_k = \frac{1}{2\pi R} \int_0^{2\pi} q_t (s, \theta) R d\theta$ in the nodes belonging to $T_\gamma (s_k)$ and $\Pi_\gamma (s_k)$ is the local interpolation matrix that returns the values of each test function $\varphi_i^t$ on the set of points belonging to $T_\gamma (s_k)$. Omitting the parameters, it is possible to analyze the structure of the exchange matrices. Therefore, thanks to these operators, the exchange matrices are
Figure 1.2: Illustration of the vessel with its centerline $\Lambda_h$, a cross section, its perimeter $\gamma(s_k)$ and its discretization $T_\gamma(s_k)$ used for the definition of the operators $\bar{\pi}_{vt}$ and $\pi_{tv}$.

implemented as

\[
\begin{align*}
\mathcal{B}_{uv} & \propto \mathcal{M}_{uv} \\
\mathcal{B}_{ut} & \propto \bar{\Pi}_{vt}^T \mathcal{M}_{uv} \bar{\Pi}_{vt} \\
\mathcal{B}_{tv} & \propto \bar{\Pi}_{vt}^T \mathcal{M}_{uv} \\
\mathcal{B}_{vt} & \propto \mathcal{M}_{uv} \bar{\Pi}_{vt}.
\end{align*}
\]
Chapter 2

Numerical Results

We develop a C++ code which provides a solution for the model described in Chapter 1. The code will be discussed in Chapter 3; in this section we show the numerical results which validate the code.

Firstly we will present benchmarks on very simple networks, where we have the exact solution or, at least, we can predict the trend of the solution; in addiction, we will not use physiological parameters.

Subsequently, we will test a more complex network with physiological parameters.

2.1 Uncoupled 1D and 3D test

Let us first consider an uncoupled problem, to validate separately the equations in the tissue and in the network. To achieve this framework, we set \( Q = 0 \) and \( \Upsilon = 0 \), that means the capillary walls are impermeable to both fluid and particles.

2.1.1 Stand-alone vessel network problem

Given the uncoupled framework, in a network made of a single unitary branch, \( \Lambda = [0, 1] \) the equation on vessels become a standard one-dimensional advection-diffusion problem. Assigning Dirichlet conditions on the boundary, we have:

\[
\begin{align*}
\frac{\partial}{\partial t} c_v(s, t) - A_v \frac{\partial^2}{\partial s^2} c_v(s, t) + u_v \frac{\partial}{\partial s} c_v(s, t) &= 0 \quad s \in (0, 1), t > 0, \\
c_v(0, t) &= 1, \quad c_v(1, t) = 0, \quad c_v(s, 0) = 0, \quad s \in (0, 1), t > 0,
\end{align*}
\] (2.1)

Under the hypothesis of a constant advection field, the stationary problem has an easy exact solution, that is:

\[
c_v^{ex}(s) = \frac{\exp\left(\frac{u_v A_v}{s}\right) - \exp\left(\frac{u_v A_v}{s}\right)}{\exp\left(\frac{u_v A_v}{s}\right) - 1}.
\] (2.2)

We then proceed to test some numerical simulation on the exact solution: as shown in Figure [2.1] the results fit the theoretical expectations.
Figure 2.1: Uncoupled problem 2.1 on a single branch network. In all the simulations $A_t = 1$, $R' = 1$. In panels A and B, $\Lambda$ is discretized with 21 equally distributed nodes. Panel A shows concentration profiles on the branch for different velocities. Panel B shows concentration profiles on the branch for different velocities, compared to the exact solution. In this simulations Péclet number ranges between 0 and 0.8. Panel C shows how the mesh size affect the simulation in case of large Péclet. We compare the numerical approximations with $u_v = 0$ and $u_v = 16$ for different discretizations; notice that in the second case, with $h_v \in \{0.2, 0.1, 0.05, 0.025\}$, Péclet take values $Pe_v = \{3.2, 1.6, 0.8, 0.4\}$. For Péclet larger than 1, artificial diffusivity is added to stabilize the solution, therefore the numerical data fit the exact solution only for small enough $h_v$. Panel D shows a time-dependent solution converging to the stationary solution after a few time-steps. In this simulation: $u_v = 16$ and $h_v = 0.025$. 
Remark 2. The advection field is computed by imposing Dirichlet conditions on the pressure at the extrema of the vessel. In the uncoupled case, the pressure is linear and the velocity, proportional to the first derivative of pressure, is constant. Setting \( p_{in} \) and \( p_{out} \) the Dirichlet conditions of pressure of inlet and outlet, exact pressure and velocity is:

\[
\begin{align*}
\hat{p}_v(s) &= p_{in} + (p_{out} - p_{in}) s & s \in (0, 1), \\
\hat{u}_v(s) &= -\frac{\kappa_v}{\pi R'^2} \frac{dp_v}{ds} = \frac{\kappa_v}{\pi R'^2} (p_{in} - p_{out}) & s \in (0, 1).
\end{align*}
\]

The numerical tests in this section are made with unitary parameters, \( \kappa_v = 1 \) and \( R' = 1 \); therefore the velocity in the branch is easily computed as \( \hat{u}_v(s) = \frac{\pi - 1}{\pi} (p_{in} - p_{out}) \).

Remark 3. The well-known theory of advection-diffusion warns us of the risks of an advection dominated setting. For this reason, the code computes Péclet number in both tissue and vessels before solve the linear system; in case of Péclet number greater than 1, in order to avoid instability, we add artificial diffusion, namely

\[
A^* v = (1 + Pe_v) A v
\]

for vessels and

\[
A^* t = (1 + Pe_t) A t,
\]

where the Péclet numbers are defined as

\[
Pe_v = \frac{A - 1}{v h} v_{max}(u_v)
\]

and

\[
Pe_t = \frac{A - 1}{t h} t_{max}(|u_t|).
\]

This stabilization method has many drawbacks, such as an excessive diffusion in the solution, but, on the other hand, it has a very simple implementation. In the future, one could try to implement a stronger stabilization method, like streamline diffusion or SUPG. Nevertheless, this seems not to be a priority, since in physiological simulations (see Section 2.6) the diffusion is usually dominating advection.

2.1.2 Stand-alone tissue problem

We study the uncoupled tissue problem on \( \Omega = (0, 1)^3 \), which becomes an advection-diffusion-reaction problem:

\[
\frac{\partial}{\partial t} c_t + \nabla \cdot (c_t u_t - A_t \nabla c_t) + D_\alpha c_t = 0 \quad \text{on } \Omega, \ t > 0.
\]

In this case, we just want to confront qualitatively some numerical simulations with different advection fields. Therefore, we test a tissue slab with fixed concentration on two parallel faces, in order to see how particles diffuse in the domain; the particles can filtrate through the other four faces. We give appropriate boundary conditions, namely Dirichlet and Robin conditions, and an initial condition:

\[
\begin{align*}
&c_t(x, y, z, t) = 1 & \text{on } x = 0, \ y \in [0, 1], \ z \in [0, 1], \ t > 0, \\
&c_t(x, y, z, t) = 0 & \text{on } x = 1, \ y \in [0, 1], \ z \in [0, 1], \ t > 0, \\
&-A_t \nabla c_t(x, y, z, t) \cdot n = \beta_c c_t(x, y, z, t) & \text{on } x \in [0, 1], \ y \in [0, 1], \ z \in [0, 1] \ t > 0, \\
&c_t(x, y, z, t) = 0 & \text{on } x \in [0, 1], \ y \in [0, 1], \ z \in [0, 1], \ t = 0.
\end{align*}
\]

In this test, we want to focus on the effect of the advection with respect to the diffusion; therefore, we set \( D_\alpha = 0 \) and look at stationary solutions with \( Pe < 1 \). We performed four tests: \( u_t = 0, \ u_t = 8i, \ u_t = -8j, \) and \( u_t = -8k \). We see that in the first
Figure 2.2: Uncoupled problem 2.5 on $\Omega = (0,1)^3$, discretized with 21 points on each direction. In every panel, the figure on the left shows the concentration in all the domain; the figure on the right shows the concentration on the y-z plane $x = 0.5$. Panel A has $u = 0$ (only diffusion). Panel B has $u = 8i$. Panel C has $u = -8j$. Panel D has $u = -8k$. 
case, which has no advection, the concentration decrease linearly on the x axis, between the two Dirichlet conditions $c_t = 1$ and $c_t = 0$; besides, we can see that in an arbitrary y-z plane, more particles are in the center of the domain, as they exit at the borders because of the Robin conditions. On the other hand, when there is a velocity field directed along the x axis, particles diffuse faster in this direction, and the Dirichlet condition $c_t = 1$ spread in a larger part of the domain. Finally, the advection fields directed along the y and z axis, give a concentration which diffuse linearly in the x-axis, but in the y-z slices particles are no longer in the center, but move according the advection field.

**Remark 4.** The advection field is computed numerically using opportune conditions. In fact, in this case the fluid equations are:

\[
\begin{align*}
\frac{1}{\kappa_t} u_t + \nabla p_t &= 0 \quad \text{in } \Omega, \\
\nabla \cdot u_t &= 0 \quad \text{in } \Omega.
\end{align*}
\]

(2.7)

If we give Dirichlet conditions on pressure on two parallel faces, and homogeneous Neumann conditions on velocity on the remaining four faces of the domain, the fluid behaves as in a pipe: We find a linear pressure profile and constant velocity (depending on the gradient of pressure). For example, let us suppose we give Dirichlet conditions on the faces with normal directed as the x-axis, exact pressure and velocity are:

\[
\begin{align*}
p_{\text{ex}}^x(x, y, z) &= p_{\text{in}} + (p_{\text{out}} - p_{\text{in}}) x \quad \text{in } \Omega, \\
u_{\text{ex}}^x(x, y, z) &= -\kappa_t \nabla p_t = \kappa_t (p_{\text{in}} - p_{\text{out}}) i \quad \text{in } \Omega.
\end{align*}
\]

(2.8) (2.9)

In our tests, $\kappa_t = 1$, therefore the module of the advection field is equal to the pressure jump.

### 2.2 Coupled 3D-1D problem on a single branch

We want to test the coupling terms between one-dimensional network and the three-dimensional tissue. We consider the case of the single unitary vessel immersed in a unitary cube of interstitial tissue. Unfortunately, we don’t have any analytical solution for this case, so we simplify the model in order to see the effect of the exchange of mass: we set $A_t = 1$, $A_v = 1$, $u_t = 0$, $u_v = 0$ and $D_\alpha = 0$. As described in 1.7, the exchange term is made of two parts: one depending on oncotic pressure and the other caused by the vessel permeability to the particles. For this test case, we decided to model an equilibrium case, where pressure jump of blood between vessel and tissue is equal to the oncotic pressure jump; this is achieved by setting $Q = 0$, $\Upsilon_\alpha = 4$, and therefore we only show the effects of the permeability.

We used a discretization of 21 points for the single branch $\Lambda$ and a regular mesh with $h = 0.05$ (21 points for each side of the cube $\Omega$), as shown in Fig. 2.1. The boundary conditions are Dirichlet in the vessel ($c_v = 1$ in inlet and $c_v = 0$ in outlet), while there are Neumann conditions in all the tissue faces, that is: $-\nabla c_t \cdot n = \beta_t c_t$ with $\beta_t = 0.1$. A stationary solution is computed. The figure shows how the concentration diffuses from the vessel to the surrounding tissue. As told before, we don’t have any exact solution for this problem, but the plot represents the expected solution.
Figure 2.3: Coupled problem [1.6] on single unitary branch Λ discretized with 41 points; tissue domain Ω = (0, 1)^3 has 21 points in each directions. The radius of the vessel is \( R' = 0.1 \). Panel A shows the mesh we used for the simulations. Besides the 3D and the 1D mesh, the real vessel is plotted in red. Panel B shows the concentration \( c_t \) in the domain. Panel C shows concentration \( c_t \) on the plane \( z = 0.5 \), \( c_v \) in the vessel, and the real vessel. Panel D plots the concentration \( c_v \) along the branch, for different value of \( \Upsilon \), ranging from \( \Upsilon = 0 \) to \( \Upsilon = 32 \).
To investigate further the effect of permeability we show the plot of $c_v$ in the single branch for different values of permeability, namely $\Upsilon = \{0, 1, 2, 4, 8, 16, 32\}$. We see that the Dirichlet condition still holds for all the case. For $\Upsilon = 0$ we find the linear solution of the uncoupled benchmark; increasing the leakage of particles, the mass in the center of the vessel decreases. It is interesting to note the for high values of permeability, there is a point in which the exchange of mass is balanced by the diffusivity in the tissue: since the leakage of particles depends on the concentration jump on the vessel wall, if permeability and diffusivity are sufficiently high, concentration in tissue surrounding the outlet exchanges back particles in the vessel; this phenomenon results in the inflection point and in the super-linear plot for $\Upsilon \geq 16$.

### 2.3 Test on a single bifurcation

We now verify the code with a slightly more complex one-dimensional domain, that is a single bifurcation. This network is made by the junction of three capillaries with same length; we will call the bifurcation $\Lambda$ and the branches $\Lambda_0$, $\Lambda_1$ and $\Lambda_2$. We set the first branch, $\Lambda_0$, from $x_A = (0, 0.5, 0.5)$ to $x_M = (0.58, 0.5, 0.5)$, to be an inlet capillary; $\Lambda_1$ and $\Lambda_2$, from $x_M$ to $x_B = (1, 0.9, 0.5)$ and $x_C = (1, 0.1, 0.5)$ respectively, will be the outlet vessels. We discretize each branch with 21 equidistant points; the interstitial tissue $\Omega$ is modeled again as a unitary cube, discretized in a uniform mesh of tetrahedra with 15 points on each side.

For this geometrical setting, neither the uncoupled nor the coupled problem have an exact solution. We provide some tests for this two cases in order to see if the code behaves according to our predictions. Precisely, we will pay attention to the junction point $x_M$ and to the behaviour in asymmetrical conditions.

The junction point is crucial in order to see if the mass balance is respected: for the sake of clarity, we write again the equation of the mass fluxes (1.11):

$$
\sum_{i \in P_{out}} \pi R_i^2 \left( -A_v \frac{\partial c_v}{\partial s} + u_v c_v \right)_{\Lambda_i^+} = \sum_{i \in P_{in}} \pi R_i^2 \left( -A_v \frac{\partial c_v}{\partial s} + u_v c_v \right)_{\Lambda_i^-} \forall j \in J.
$$

(2.10)

In our simple case, the sum of the fluxes in branches $\Lambda_1$ and $\Lambda_2$ at the junction point must be equal to the flux in branch $\Lambda_0$ in $x_M$. Notice that we also proved, with (1.12) and (1.13), that the diffusive and advective fluxes must hold separately.

**Remark 5.** The following simulations will have radius constant in each branch but variable on different branches. This is physiologically correct: it is unlikely that the two daughter vessels and the parent vessel have the same radius. The radii of the daughter branches are calculated on the basis of the Murray’s Law, that is $R_3^2 = R_1^3 + R_2^3$. This relation derives from the minimization of the cost of the transport, that is the sum of the power required for transport itself and the power required to maintain the transport medium. This law is observed in respiratory and vascular systems of animals, and has many applications in bio-engineering. This model is not sensitive to the angle of the daughter branches.

In the following sections, we will study the uncoupled case and successively the coupled case. We will initially study symmetric problems, with $R'_1 = R'_2 = R'_0 \sqrt[3]{1/2}$ and the same
Table 2.1: Boundary conditions of the test cases on the uncoupled bifurcation, and mass balance at the junction. In each table, we have: the list of the branches (first column), the value of the Dirichlet conditions on the pressure (second column), the value of the Dirichlet conditions on the concentration (third column), the radius in the branch (fourth column), the diffusive flux $\pi R_i^2 A_v \frac{\partial c_v}{\partial s}$ at the junction (fifth column) and the advective flux $\pi R_i^2 u c_v$ at the junction (sixth column). The last row of each table give the mass balance in the junction due to diffusive and advective fluxes, respectively. Table A refers to the simulation with the same Dirichlet condition on pressure on the two outlets, and same radius in the outlet branches. Table B refers to the simulation where the Dirichlet condition on pressure is different on the two outlets. Table C refers to the simulation where the radii in the outlet branches are different.

boundary conditions on both outlets. Next, we will see the behaviour of the simulation with $R_1' \neq R_2'$ and with different boundary conditions in the outlets.

### 2.3.1 Uncoupled problem on the single bifurcation network

Using the geometrical setting above described, we study the transport of particles in the bifurcation when there is no exchange of mass through the wall. Therefore, we set $Q = 0$, $\Upsilon = 0$ and $A_v = 1$, and solve only the equation in the network. We set Dirichlet conditions on the network: $c_v = 1$ at the inlet and $c_v = 0$ at the outlet. We then run three simulation with different advection fields and different value for the radii, as shown in Fig. (2.4) and in Tab. (2.1). The following test will be diffusion-dominated, so there will no need to stabilize; the Péclet number in the network, under the advection field of the first test, is $Pe = 0.3331$.

In the panel A of Fig. (2.4) we see the discrete solution of a symmetric bifurcation. The advection field is computed by imposing the Dirichlet conditions $p_v = 0.1\pi$ on the
Figure 2.4: Uncoupled($Q = 0, \Upsilon = 0$) problem on a simple bifurcation. For every panel, we show the concentration in the bifurcation $\Lambda$ (left), the plot of $c_v$ for every branch (center) and the plot of $u_v$ for every branch (right). The boundary condition on the transport problem is $c_v = 1$ in the inlet and $c_v = 0$ in both outlets, in each panel. In panel A, the radii in the branches are $R_0' = 0.1$, $R_1' = R_2' = 0.793$; the boundary condition on the pressure-velocity problem are Dirichlet condition $p_v = 0.1\pi$ in the inlet and $p_v = 0$ in both outlets. In panel B, the radii in the branches are $R_0' = 0.1$, $R_1' = R_2' = 0.793$; the boundary condition on the pressure-velocity problem are Dirichlet condition $p_v = 0.1\pi$ in the inlet, $p_v = 0.2$ in the upper branch and $p_v = 0$ in the lower branch. In panel C, the radii in the branches are $R_0' = 0.1$, $R_1' = 0.872$, $R_2' = 0.713$; the boundary condition on the pressure-velocity problem are Dirichlet condition $p_v = 0.1\pi$ in the inlet and $p_v = 0$ in both outlets.
inlet and \( p_v = 0 \) on the two outlets; the radii of the outlet branches are the same and follow the Murray’s Law: \( R'_0 = 0.1 \) and \( R'_1 = R'_2 = 0.793 \). Therefore the velocities in branches \( \Lambda_1 \) and \( \Lambda_2 \) are equal. The plots show that effectively the particles are equally distributed in the two outlet branches.

In panel B, we change the boundary conditions on pressure, in order to change the advection field: \( p_v|_{x_A} = 0.1 \pi \), \( p_v|_{x_B} = 0.2 \) and \( p_v|_{x_C} = 0 \). Under these conditions, the velocity in the upper branch \( \Lambda_1 \) is negative, that means that the fluid is actually entering from this branch and not exiting; we see that this affects the concentration of particles, that is greater in the lower branch \( \Lambda_2 \).

In panel C, we change the value of the radii: we increase \( R'_1 \) by 10% and reduce \( R'_2 \) according the Murray’s Law: \( R'_0 = 0.1 \), \( R'_1 = 0.872 \) and \( R'_2 = 0.713 \). The velocity \( u_v \) is therefore slightly greater in the smaller branches, and this advection field brings slightly more mass in \( \Lambda_2 \).

We now check the mass balance at the junction: the values are reported in Tab. (2.1). We see that in all the cases the advective flux is correctly balanced with a high precision; this means that the conservation in the local flow rate is well imposed, and that the continuity of \( c_v \) is respected at the junction, as we could see in the previous plots. On the other hand, the diffusive flux is correctly balanced but with small precision: in the tests we presented, and in all the other simulations we computed, the sum of the fluxes is usually a order \( 10^{-1} \) smaller than the higher of the diffusive fluxes, instead of being close to zero. This small precision may be due to different factors: this mass balance is weakly enforced on the junction points, from the integration by parts, in (1.15); maybe changing the formulation more accurate results can be achieved. Besides, the diffusive flux depend on the first derivative of \( c_v \): we recall that we use piecewise linear finite element, and therefore the first derivative of \( c_v \) is piecewise constant. Higher order of finite elements may result in higher accuracy in mass balance.

### 2.3.2 Coupled 3D-1D problem with a single bifurcation

We now test the bifurcation with the coupled model, and see the effect of permeability with multiple branches. We use the same geometrical framework described above, but we now start using the physiological parameters from Tab. (2.2), taken from [2, 4]; under these settings, we still have a diffusion-dominated problem, as the Pécelt numbers are \( P_e = 8.586 e^{-02} \) and \( P_{e_t} = 4.417 e^{-05} \). We set Dirichlet conditions on the network: \( c_v = 1 \) at the inlet and \( c_v = 0 \) at the outlet; we set mixed conditions on the faces of the 3D domain, \( -A_t \nabla c_t = 0.1 c_t \). As done before, we test a symmetric and an asymmetric framework, as shown in Fig. (2.5).

In the panel A of Fig. (2.5) we see the discrete solution of a symmetric bifurcation. The advection field is computed by imposing the Dirichlet conditions \( p_v = 32 \) on the inlet and \( p_v = 28.5 \) on the two outlets; the radii of the outlet branches are the same and follow the Murray’s Law: \( R'_0 = 10^{-5} \) and \( R'_1 = R'_2 = 7.93 \cdot 10^{-6} \). Therefore the velocities in branches \( \Lambda_1 \) and \( \Lambda_2 \) are equal. The plots show that effectively the particles are equally distributed in the two outlet branches. The particles in the tissue are distributed along the whole bifurcation; we notice that the highest concentration in tissue is around the junction. This can be explained by the fact that only source of \( c_t \) is the permeability of the network, which is determined by the quantity of surface of the vessel which can
Table 2.2: Physiological parameters from [1, 4]. First table contains parameters for the equations and of the dimension analysis. Second table contains parameters for boundary conditions.
Figure 2.5: Coupled problem on a bifurcation Λ with physiological parameters from table 2.2. The plot shows the bifurcation Λ and the tissue Ω cut in half by a x-y plane. On the left, a symmetric junction; the radii are \( R_0' = 0.1, R_1' = R_2' = 0.793 \); the Dirichlet conditions on pressure are \( p_v = 32 \) in inlet and \( p_v = 28.5 \) in both outlets. On the right, an asymmetric junction; the radii are \( R_0' = 0.1, R_1' = R_2' = 0.793 \); the Dirichlet conditions on pressure are \( p_v = 32 \) in inlet, \( p_v = 28.5 \) in the upper outlet and \( p_v = 23 \) in the lower outlet.

exchange mass; around the junction we have three surfaces that can exchange mass, and therefore higher values of \( c_t \) are found here.

In panel B, we change the boundary conditions on pressure, in order to change the advection field: \( p_v|_{x_A} = 32 \), \( p_v|_{x_B} = 28.5 \) and \( p_v|_{x_C} = 23 \). Under these conditions, the velocity field advects higher values of \( c_v \) in the lower branch; therefore we notice that higher values of \( c_t \) are found near Λ.

In Tab. (2.3) we report the diffusive and advective fluxes in the junction. The same considerations of the uncoupled case can be done.

### 2.4 Coupled 3D-1D problem on a complex network

As a final step, we test the code on a complex network. We consider the coupled domain \((Ω, Λ)\) consisting of a non trivial one-dimensional manifold immersed in a three-dimensional unitary cube. The discretization of \( Ω \) is a regular mesh of tetrahedra with size \( h = 0.1 \) (11 points for each side of the cube \( Ω \)). The network is made of 47 branches of variable length; the total length of the network is 15.0. Each branch is discretized with 31 equidistant points; the radius is constant in the network, \( R = 4 \cdot 10^{-6} \).

The boundary conditions on concentration are:

\[
\begin{align*}
  c_v & = 1 & \text{on } & \partial Λ^{IN} \times (0, T), \\
  -A_v \frac{∂c_v}{∂s} & = β_v c_v & \text{on } & \partial Λ^{OUT} \times (0, T), \\
  -A_v \nabla c_v \cdot n & = β_t c_t & \text{on } & \partial Ω \times (0, T).
\end{align*}
\]

We choose the parameters from Tab. (2.2); under these assumption, we have a diffusion-dominated problem, where \( Pe_v = 2.728ε - 01 \) and \( Pe_t = 3.848ε - 05 \).

Since the validation on the simple geometries, and the absence of evident anomalies in Fig. (2.6), we can deduce the correctness of the solution. We can observe, as in the
Figure 2.6: Complex network with physiological parameters from Table 2.2. The figure shows different points of view of the same simulation on a network of 47 branches, each discretized with 31 points; each side of \( \Omega \) is subdivided with 11 points. The boundary conditions on the inlet points are \( p_v = 32 \) and \( c_v = 1 \); the boundary condition on the outlet points are \( p_v = 28.5 \) and \( -A_v \frac{\partial c_v}{\partial s} = 0.1c_v \). The boundary conditions on the faces of \( \Omega \) are Robin-like for both pressure and concentration, namely \( -\kappa_t \nabla p_t \cdot n = 0.1p_t \) and \( -A_t \nabla c_t \cdot n = 0.1c_t \). The radius is constant in the network, and it is used the physiological value \( R = 4 \times 10^{-6} \). On the left we have the concentrations \( c_t \) and \( c_v \), while on the right we have the advection field \( u_t \) and \( u_v \), described by both the color plot and the vectors. Panel A shows only \( c_v \) and \( u_v \) in the whole network. Panel B shows also tissue values \( c_t \) and \( u_t \), with the domain \( \Omega \) cut in half by an x-y plane. Panel C shows all the variables with the domain \( \Omega \) cut in half by an y-z plane.
Table 2.3: Boundary conditions of the test cases on the coupled bifurcation, and mass balance at the junction. In each table, we have: the list of the branches (first column), the value of the Dirichlet conditions on the pressure (second column), the value of the Dirichlet conditions on the concentration (third column), the radius in the branch (fourth column), the diffusive flux $\pi R^2 \Delta_v \frac{\partial c_v}{\partial s}$ at the junction (fifth column) and the advective flux $\pi R^2 u_v c_v$ at the junction (sixth column). The last row of each table give the mass balance in the junction due to diffusive and advective fluxes, respectively. Table A refers to the simulation with the same Dirichlet condition on pressure on the two outlets, and same radius in the outlet branches. Table B refers to the simulation where the Dirichlet condition on pressure is different on the two outlets.

bifurcation, that the higher values of $c_t$ are near groups of many branches. Nevertheless, one can observe also that the regions where the branches are more tangled contains less particles; in fact, the mass is advected more efficiently in sequence of vessels with few bifurcations. Although these complex regions with many bifurcation have large surface for exchanging mass, they have so low values of $c_v$ that cannot relevantly affect the values of $c_t$. This is an important result, since it reproduces an experimental phenomenon of one of the many applications of this model: the cardiovascular system around tumoral cells is more tangled with respect to an healthy one; for this reason, these cells receive less oxygen but, on the other hand, it is harder to deliver them drugs such as chemotherapy \[2, 7\].
Chapter 3

C++ Code

The latest stable release of the code is available at the following link: [https://github.com/stefano-brambilla-853558/MANworks](https://github.com/stefano-brambilla-853558/MANworks).

Let us briefly review the main achievements. The code has been developed in the context of the MANworks project; the purpose is to develop a C++ library based on Getfem++ [6] for the solution of PDEs on networks coupled with the surrounding environment. The starting point of this project was the code by Domenico Notaro [3], that solve the equations of fluid dynamics [1,4]; our work was to implement a new solver for the transport equations [1,1] that could read the advection and pressure fields from the previous solver.

3.1 Design of the code

Before starting to write the new solver, we decided to follow three main rules: separation, consistency and synthesis.

First, the solver for fluid problem is continuously being improved, in the whole context of the MANwork project. Therefore, we needed something that could use the different features of the first solver, but that was able to easily integrate the new versions of the code: we decided to keep the two solver as separated as possible.

Second, in order to make the code more readable, we decided to unify the terminology between the two solvers. Also, the whole structure of the library has been maintained as far as possible.

Third, we recognized that, apart from the structure of the monolithic matrix, the structure of the code was quite similar; we tried to not duplicate parts of code or objects used in both the solvers.

Since the fluid solver was written following the two main design principles of OOP, namely encapsulation and information hiding, it has been easy to achieve our goals.

The fluid equations solver contains a main class, problem3d1d, containing all the attributes and methods which are needed to solve the fluid problems. We built a new class, transport3d1d, that inherits from the class problem3d1d. As a good programming practise, the declaration of the class transport3d1d was exported to an appropriate header file (transport3d1d.hpp), while its definition was moved to a source file (transport3d1d.cpp).
In the class `transport3d1d` we added all the new attributes for the transport problem, only when needed: for example, the header files `descr3d1d.hpp`, `dof3d1d.hpp` and `param3d1d.hpp` contained the definitions of the descriptors of the algorithm, the dimensions of the problem and the physical parameters, respectively, used in the fluid problem; we added the header files `descr3d1d_transp.hpp`, `dof3d1d_transp.hpp` and `param3d1d_transp.hpp` that contained the definitions of the descriptors of the algorithm, the dimensions of the problem and the physical parameters, respectively, that are used only in the transport problem. Furthermore, other attributes that are shared between the two problems, like the meshes, are not declared again.

```cpp
//! Main class defining the coupled 3D/1D transport problem.
class transport3d1d: public problem3d1d {

public:
transport3d1d(void) :
mf_Ct(mesht), mf_Cv(meshv){}

//! Initialize the transport problem
void init_transp (int argc, char *argv[]);

//! Assemble the transport problem
void assembly_transp (void);

//! Solve the transport problem
bool solve_transp (void);

//! Export the transport solution
void export_vtk_transp (const string & time_suff = "", const string & suff = "");

//! Compute residuals for mass balance at each junction
void mass_balance (void);

// Aux methods for interface with problem3d1d class

//! Initialize the fluid problem
void init_fluid (int argc, char *argv[]);

//! Assemble the fluid problem
void assembly_fluid (void);

//! Solve the fluid problem
bool solve_fluid (void);

//! Export the fluid solution
void export_vtk_fluid (const string & suff = "");

protected:

//! Finite Element Method for the tissue concentration @f$c_t@f$
mesh_fem mf_Ct;

//! Finite Element Method for the vessel concentration @f$c_v@f$
mesh_fem mf_Cv;

//! Algorithm description strings (mesh files, FEM types, solver info, ...)
descr3d1d_transp descr_transp;
```
```cpp
// Physical parameters
param3d1d_transp param_transp;

// Number of degrees of freedom
dof3d1d_transp dof_transp;

// List of BC nodes of the network
vector< node > BCv_transp;

// List of BC nodes of the tissue
vector< node > BCT_transp;

// List of junction nodes of the network
vector< node_transp > Jv_transp;

// Monolithic matrix for the coupled problem
sparse_matrix_type AM_transp;

// Monolithic array of unknowns for the coupled problem
vector_type UM_transp;

// Monolithic right hand side for the coupled problem
vector_type FM_transp;

// Monolithic temporary matrix for update
sparse_matrix_type AM_temp;

// Monolithic temporary right hand side for update
vector_type FM_temp;

// Aux methods for init

// Import algorithm specifications
void import_data_transp(void);

// Import mesh for tissue (3D) and vessel (1D)
void build_mesh_transp(void);

// Set finite elements methods and integration methods
void set_im_and_fem_transp(void);

// Build problem parameters
void build_param_transp(void);

// Build the list of tissue boundary data

/*! Face numbering:
0 : \{x = 0\} "back"
1 : \{x = Lx\} "front"
2 : \{y = 0\} "left"
3 : \{y = Ly\} "right"
4 : \{z = 0\} "bottom"
5 : \{z = Lz\} "top"
*/
void build_tissue_boundary_transp(void);

// Build the list of vessel boundary (and junctions) data
void build_vessel_boundary_transp(void);

// Aux method for assembly

// Build the monolithic matrix AM_transp by blocks
void assembly_mat_transp(void);

// Build the monolithic rhs FM_transp by blocks
void assembly_rhs_transp(void);

// Aux method for solve
```
// Aux function for update of rhs at each time step
void update_transp(void);

}; // end of class trasport3did

The structure of the user-interface is preserved: the code is divided in four main steps: (i) initializing of the problem, (ii) assembling of the linear system, (iii) solving it and (iv) saving the solution for post-processing. These phases are the only methods that remain public.

Remark 6. In order to have only this methods in the user-interface, we choosed to run the time loop inside the function solve_transp(); therefore, in order to save the solution at each time step, we decided to insert the method export_vtk_transp directly in the solver. Anyway, one can call the export method at the end of the main.cpp.

```cpp
// Declare a new problem
getfem::transport3d1d p;

//////// fluid problem: velocity field and pressure

// Initialize the problem
p.init_fluid(argc, argv);
// Build the monolithic system
p.assembly_fluid();
// Solve the problem
if (!p.solve_fluid()) GMM_ASSERT1(false, "solve procedure has failed");
// Save results in .vtk format
p.export_vtk_fluid();

//////// transport problem: concentration

// initialize
p.init_transp(argc, argv);
// assemble
p.assembly_transp();
// solve
if (!p.solve_transp()) GMM_ASSERT1(false, "solve procedure has failed")
   ; // the export is in the solve at each time step
```

The package MANworks contains:

- fluid/: A stable version of the code for fluid problem
- transport/: The code for transport problem
- Makefile: Instruction to install the whole project

The folder fluid/ can be replaced with any new release of the code. The folder transport/ has the following structure:

- doc/: Code documentation (to be generated).
3.2 Assembling routines

After the design of the main class, we assemble the different terms arising from \( L.18 \).
The build-in functions of GetFem++, and the tool `generic_assembly` for non-standard terms, are the basis for the assembling phase.

In `assembling1d_transp.hpp` we build the matrices \( M_v \) and \( A_v \). The function `asm_network_transp` builds in particular the mass matrix for time derivative and the stiffness matrix for the diffusion term; the function `asm_advection_network` builds the two advection terms \( L.14 \).
template<typename MAT, typename VEC, typename VEC2>
void
asm_network_transp
(MAT & M, MAT & D,
 const mesh_im & mim,
 const mesh_fem & mf_c,
 const mesh_fem & mf_data,
 const VEC & diff,
 const VEC2 & R,
 const mesh_region & rg = mesh_region::all_convexes())
{
GMM_ASSERT1(mf_c.get_qdim() == 1, 
    "invalid data mesh fem (Qdim=1 required)");
// build mass matrix Mv for time derivative
VEC param(mf_data.nb_dof()); gmm::clear(param);
gmm::add(R, param);
gmm::vscale(R, param);
gmm::scale(param, pi); // param = pi*R^2
getfem::asm_mass_matrix_param(M, mim, mf_c, mf_data, param, rg);
// Build the diffusion matrix Dv
    gmm::vscale(diff, param); // param = 2pi*R^2*Av
getfem::asm_stiffness_matrix_for_laplacian(D, mim, mf_c, mf_data, param, rg);}
} // end of asm_network_transp

template<typename MAT, typename VEC>
void
asm_advection_network
(MAT & B,
 const mesh_im & mim,
 const mesh_fem & mf_c,
 const mesh_fem & mf_data,
 const mesh_fem & mf_u,
 const mesh_fem & mf_R,
 const VEC & U,
 const VEC & lambdax, const VEC & lambday, const VEC & lambdaz,
 const VEC & R,
 const mesh_region & rg = mesh_region::all_convexes())
{
generic_assembly
assem1("l1=data$1(#2); l2=data$2(#2); l3=data$3(#2); u=data$4(#3); R=
data$5(#4);";
"t=comp(Base(#1).Grad(#1).Base(#2).Base(#3).Base(#4).Base(#4));"
"M$1(#1,#1)+=t(:,:,1,i,p,m,n).l1(i).u(p).R(m).R(n)+t(:,:,2,i,p,m,n).l2(
    i).u(p).R(m).R(n)+t(:,:,3,i,p,m,n).l3(i).u(p).R(m).R(n);";
assem1.push_mi(mim);
assem1.push_mf(mf_c);
assem1.push_mf(mf_data);
assem1.push_mf(mf_u);
assem1.push_mf(mf_R);
assem1.push_data(lambdax);
assem1.push_data(lambday);
In assembling 3d_transp.hpp we build the matrices $M_t$ and $A_t$. The function `asm_tissue_transp` builds in particular the mass matrix for time derivative, the stiffness matrix for the diffusion term and the reaction terms; the function `asm_advection_tissue` builds the two advection terms\[1.9\].

```cpp
template<typename MAT, typename VEC>
void
asm_tissue_transp
(MAT & M, MAT & D, MAT & R,
 const mesh_im & mim,
 const mesh_fem & mf_c,
 const mesh_fem & mf_coef,
 const VEC & diff_data,
 const VEC & reac_data,
 const mesh_region & rg = mesh_region::all_convexes())
{ }
```
The coupling terms $B_{tt}$, $B_{tv}$, $B_{vt}$ and $B_{vv}$ are easily assembled by the routines in assembling3d1d.hpp, that essentially exploits the corresponding functions in assembling3d1d.hpp paying attention at the parameter for permeability.

The boundary conditions are assembled by the functions asm_network_bc_transp, asm_tissue_bc_transp and asm_coupled_bc_transp. As for the fluid equations, the user can choose between Dirichlet and Robin conditions for every face of the tissue $\Omega$ and every inlet or outlet point of the network $\Lambda$. 

```cpp
template<typename MAT, typename VEC>
void asm_network_bc_transp (VEC & F, MAT & M,
const mesh_im & mim,
const mesh_fem & mf_c,
const mesh_fem & mf_data,
const std::vector<getfem::node> & BC,
const scalar_type beta,
const VEC & R)
{
GMM_ASSERT1(mf_c.get_qdim()==1, "invalid data mesh fem (Qdim=1 required)");
GMM_ASSERT1(mf_data.get_qdim()==1, "invalid data mesh fem (Qdim=1
```
for (size_type bc=0; bc < BC.size(); bc++) {
    GMM_ASSERT1(mf_c.linked_mesh().has_region(bc), "missed mesh region" << bc);
    if (BC[bc].label=="DIR") { // Dirichlet BC
        VEC BC_temp(mf_c.nb_dof(), BC[bc].value);
        getfem::assembling_Dirichlet_condition(M, F, mf_c, BC[bc].rg, BC_temp);
        gmm::clear(BC_temp);
    }
    else if (BC[bc].label=="MIX") { // Robin BC
        VEC BETA(mf_data.nb_dof(), beta*pi);
        gmm::vscale(R, BETA); gmm::vscale(R, BETA);
        getfem::asm_mass_matrix_param(M, mim, mf_c, mf_data, BETA,mf_c. linked_mesh().region(BC[bc].rg) ); //int(beta*cv*bv)
        VEC BETA_C0(mf_data.nb_dof(), pi*beta*BC[bc].value);
        gmm::vscale(R, BETA_C0); gmm::vscale(R, BETA_C0);
        asm_source_term(F,mim, mf_c, mf_data,BETA_C0); //int(beta*c0*bv)
    }
    else if (BC[bc].label=="INT") { // Internal Node
        GMM_WARNING1("internal node passed as boundary.");
    }
    else if (BC[bc].label=="JUN") { // Junction Node
        GMM_WARNING1("junction node passed as boundary.");
    }
    else {
        GMM_ASSERT1(0, "Unknown Boundary Condition"<< BC[bc].label << endl);
    }
}

template<typename MAT, typename VEC>
void
asm_tissue_bc_transp
(VEC & F,
 MAT & M,
 const mesh_im & mim,
 const mesh_fem & mf_c,
 const mesh_fem & mf_data,
 const std::vector<getfem::node> & BC,
 const scalar_type beta
 ) {

    GMM_ASSERT1(mf_c.get_qdim()==1, "invalid data mesh fem (Qdim=1 required)");
    GMM_ASSERT1(mf_data.get_qdim()==1, "invalid data mesh fem (Qdim=1 required)");
}
for (size_type bc = 0; bc < BC.size(); ++bc) {
  GMM_ASSERT1(mf_c.linked_mesh().has_region(bc), "missed mesh region" << bc);
  if (BC[bc].label == "DIR") { // Dirichlet BC
    VEC BC_temp(mf_c.nb_dof(), BC[bc].value);
    getfem::assembling_Dirichlet_condition(M, F, mf_c, BC[bc].rg, BC_temp);
    gmm::clear(BC_temp);
  } else if (BC[bc].label == "MIX") { // Robin BC
    VEC BETA(mf_data.nb_dof(), beta);
    getfem::asm_mass_matrix_param(M, mim, mf_c, mf_data, BETA, mf_c.
      linked_mesh().region(BC[bc].rg));
    VEC BETA_C0(mf_data.nb_dof(), beta*BC[bc].value);
    asm_source_term(F, mim, mf_c, mf_data, BETA_C0);
  } else if (BC[bc].label == "INT") { // Internal Node
    GMM_WARNING1("internal node passed as boundary.");
  } else if (BC[bc].label == "JUN") { // Junction Node
    GMM_WARNING1("junction node passed as boundary.");
  } else {
    GMM_ASSERT1(0, "Unknown Boundary Condition " << BC[bc].label << endl);
  }
}
} /* end of asm_tissue_bc */

template<typename MATRM, typename VECT1, typename VECT2>
void assembling_Dirichlet_condition_coupled_tissue
  (MATRM &B, VECT1 &F, const mesh_fem &mf1, const mesh_fem &mf2,
   size_type boundary, const VECT2 &DIR) {
  size_type Q1 = mf1.get_qdim();
  size_type Q2 = mf2.get_qdim();
  size_type nb_dof1 = mf1.nb_dof();
  size_type nb_dof2 = mf2.nb_dof();
  GMM_ASSERT1(!(mf1.is_reduced()), "This function is not adapted to "
    "reduced finite element methods");
  GMM_ASSERT1(!(mf2.is_reduced()), "This function is not adapted to "
    "reduced finite element methods");
  dal::bit_vector nndof = mf1.basic_dof_on_region(boundary);
  pfem pf1;
  for (dal::bv_visitor cv(mf1.convex_index()); !cv.finished(); ++cv) { // per tutti i convessi cv della mesh 1
    pf1 = mf1.fem_of_element(cv);
    pdof_description ldof = lagrange_dof(pf1->dim());
size_type nbd = pf1->nb_dof(cv);
for (size_type i = 0; i < nbd; i++) {  // per tutti i dof i del convesso cv
size_type dof1 = mf1.ind_basic_dof_of_element(cv)[i*Q1];  // trova l'indice delle colonne riferite all
if (nndof.is_in(dof1) && pf1->dof_types()[i] == ldof) {  // se il dof i del convesso cv è in "boundary"
    for (size_type j = nb_dof1; j < nb_dof1+ nb_dof2; j++) {  // allora per tutti i dof j della mesh 2
        for (size_type l = 0; l < Q1; ++l) {
            F[j] -= B(j, dof1+l) * DIR[dof1+l];
            B(j, dof1+l) = 0;
            B(dof1+l, j) = 0;
        }
    }
}
}
}  /* end of assembling_Dirichlet_condition_coupled_tissue*/

template<typename MATRM, typename VECT1, typename VECT2>
void assembling_Dirichlet_condition_coupled_vessel
(MATRM &B, VECT1 &F, const mesh_fem &mf1, const mesh_fem &mf2,
 size_type boundary,
 const VECT2 &DIR) {
size_type Q1=mf1.get_qdim();
size_type Q2=mf2.get_qdim();

size_type nb_dof1=mf1.nb_dof();
size_type nb_dof2=mf2.nb_dof();

GMM_ASSERT1(!(mf1.is_reduced()), "This function is not adapted to "
"reduced finite element methods");
GMM_ASSERT1(!(mf2.is_reduced()), "This function is not adapted to "
"reduced finite element methods");
dal::bit_vector nndof = mf2.basic_dof_on_region(boundary);
pfem pf2;
for (dal::bv_visitor cv(mf2.convex_index()); !cv.finished(); ++cv) {  // per tutti i convessi cv della mesh 1
    pf2 = mf2.fem_of_element(cv);
pdof_description ldof = lagrange_dof(pf2->dim());
    size_type nbd = pf2->nb_dof(cv);
    for (size_type i = 0; i < nbd; i++) {  // per tutti i dof i del convesso cv
        size_type dof2 = mf2.ind_basic_dof_of_element(cv)[i*Q2];  // trova l'indice delle colonne riferite all
        if (nndof.is_in(dof2) && pf2->dof_types()[i] == ldof) {  // se il dof i del convesso cv è in "boundary"
            for (size_type j = nb_dof1; j < nb_dof1+ nb_dof2; j++) {  // allora per tutti i dof j della mesh 2
                for (size_type l = 0; l < Q1; ++l) {
                    F[j] -= B(j, dof2+l) * DIR[dof2+l];
                    B(j, dof2+l) = 0;
                    B(dof2+l, j) = 0;
                }
            }
        }
    }
}  /* end of assembling_Dirichlet_condition_coupled_vessel*/


for (size_type j = 0; j < nb_dof1; j++) { // allora per tutti i
dof j della mesh 2
for (size_type l = 0; l < Q2; ++l) {
F[j] -= B(j, nb_dof1 + dof2+l) * DIR[nb_dof1 + dof2+l];
B(j, nb_dof1 + dof2+l) = 0;
B(nb_dof1 + dof2+l, j) = 0;
}
}
} /* end of assembling_Dirichlet_condition_coupled_vessel*/

template<typename MAT, typename VEC>
void
asm_coupled_bc_transp
(MAT & M,
VEC & F,
const mesh_fem & mf_ct,
const mesh_fem & mf_cv,
const std::vector<getfem::node> & BC_tissue,
const std::vector<getfem::node> & BC_vessel )
{
GMM_ASSERT1(mf_ct.get_qdim()==1, "invalid data mesh fem (Qdim=1
required)");
GMM_ASSERT1(mf_cv.get_qdim()==1, "invalid data mesh fem (Qdim=1
required)");

// cycle over the tissue boundary nodes
for (size_type bc=0; bc < BC_tissue.size(); ++bc) {
GMM_ASSERT1(mf_ct.linked_mesh().has_region(bc), "missed mesh region" <<
bc);
if (BC_tissue[bc].label=="DIR") { // Dirichlet BC
VEC BC_temp(mf_ct.nb_dof(), BC_tissue[bc].value);
getfem::assembling_Dirichlet_condition_coupled_tissue(M, F, mf_ct,
mf_cv, BC_tissue[bc].rg, BC_temp);
gmm::clear(BC_temp);
}
}

// cycle over the vessels boundary nodes
for (size_type bc=0; bc < BC_vessel.size(); ++bc) {
GMM_ASSERT1(mf_cv.linked_mesh().has_region(bc), "missed mesh region" <<
bc);
if (BC_vessel[bc].label=="DIR") { // Dirichlet BC
VEC BC_temp(mf_cv.nb_dof(), BC_vessel[bc].value);
getfem::assembling_Dirichlet_condition_coupled_vessel(M, F, mf_ct,
mf_cv, BC_vessel[bc].rg, BC_temp);
gmm::clear(BC_temp);
}
3.3 The library transport3d1d

With the code above described, we build a dynamic library collecting all the routines, external functions and variables defining our coupled 3D/1D transport problem. There are many pros and cons to take in considerations when choosing between a static and a dynamic library. For example, the library of the fluid problem, libproblem3d1d.a, was buildt statically: the size of the executable with the static linking is comparable to the total size of the executable with the dynamic linking and the files of the library; therefore, there were no need to build the library dynamically, preferring a stand-alone executable. On the other hand, for the transport problem we opted for dynamic linking: the MANwork project is fastly getting wider, with different versions of the code. In this context, we prefer to have more flexible executable, that can quickly load different libraries when necessary, without the need of recompilation. Anyway, we finally decided to let the user choose between static and dynamic linking: the Makefile in MANworks/transport/include contains both the instructions. The default make will install the library libtransport3d1d.so dynamically, while a make static will install the library libtransport3d1d.a statically. The library for transport problem is builtt from the following headers:

`assembling1d_transp.hpp`: Miscellaneous assembly routines for the 1D network problem

`assembling3d_transp.hpp`: Miscellaneous assembly routines for the 3D tissue problem

`assembling3d1d_transp.hpp`: Miscellaneous assembly routines for the 3D/1D coupling

`descr3d1d_transp.hpp`: Definition of the aux class for algorithm description strings

`dof3d1d_transp.hpp`: Definition of the aux class for the number of degrees of freedom

`node_transp.hpp`: Definition of the class `node_transp`

`param3d1d_transp.hpp`: Definition of the aux class for physical parameters

`transport3d1d.cpp`: Definition of the main class for the 3D/1D coupled transport problem

`transport3d1d.hpp`: Declaration of the main class for the 3D/1D coupled transport problem

`utilities_transp.hpp`: Miscellaneous aux functions for the 3D/1D coupled transport problem

See the README file for installation instructions.
3.4 Doxygen documentation

The whole code has been extensively commented and documented. The last release of the code provides the possibility to automatically generate a detailed code documentation using Doxygen. See the README file for installation instructions.
Conclusion and future perspective

We developed a solver for 3D/1D coupled transport solver. This is a considerable step forward in the MANworks project, but it’s surely not the final point; while developing this code, many progresses were made.

G. Raimondi extended the problem to curve network [5]; L. Possenti et al. added the transport of hematocrit in the fluid equations, in order to have a more precise model [4]. We are currently testing how the transport of particles is affected by these improvements.

A. Tiozzo used our code in order to test the diffusion of nanoparticles with adhesion [8], with good results.

This code has currently being developed in the context of a Master Thesis: the aim is to use the library to compute the convergence and the model errors in these 3D/1D coupled problems. This developments are in the same repository of the validated code, in the branch transport: https://github.com/stefano-brambilla-853558/MANworks/tree/transport.
Bibliography

[1] L. Cattaneo. Fem for pdes with unfitted interfaces: Application to flow through heterogeneous media and microcirculation. 2.3.2 2.2


[4] di Gregorio S. Gerosa F.M. Raimondi G. Casagrande G. Costantino M.C. Possenti, L. and P. Zunino. A computational model for microcirculation including fahraeus-lindqvist effect, plasma skimming and fluid exchange with the tissue interstitium. https://github.com/lpossenti/MANworks_ht_curvature 2.3.2 2.2 3.4


