Politecnico di Milano

School of Industrial and Information Engineering
M.Sc. of Mathematical Engineering
Computational Science and Engineering

Mixed Finite Element Methods for Coupled 3D/1D Fluid Curved Problem

Project of Advance Programming for Scientific Computing

Giorgio Raimondi
Matr. 853037

• May 2017 •
Accademic Year 2016/2017

Contents

Introduction **2**

1 Coupled 3D/1D Problem **2**
1.1 Model set up . . . 2
1.2 Coupling microcirculation with interstitial flow . . . 3
1.3 Derivation of the 1D approximation in case of rectilinear axis . . . 3
1.3.1 Model Assumptions . . . 3
1.3.2 Model Derivation . . . 5
1.3.3 Full linear coupled equations . . . 6
1.3.4 Boundary conditions . . . 6
1.4 Derivation of the 1D approximation for curved segments . . . 7
1.4.1 Preliminary Model Assumptions . . . 7
1.4.2 Derivation of the average velocity profile . . . 7
1.4.3 Model derivation for curved segments . . . 10
1.4.4 Full curved coupled equations . . . 11
1.5 Dimensional Analysis . . . 11

2 Mixed Formulation for velocity and pressure **13**
2.1 Weak Formulation for tissue interstitium . . . 13
2.2 Weak formulation for the vessel . . . 14
2.3 Coupled weak formulation . . . 16

3 Discrete Model **17**
3.1 Finite elements set up . . . 17
3.2 Discrete Model . . . 17
3.2.1 Fixed Point Method discretization . . . 18
3.2.2 Newton Method discretization . . . 20

4 Model Validation **21**
4.1 Equations comparison . . . 21
4.2 Numerical Comparison . . . 21
4.2.1 Advective effect . . . 22
4.2.2 SingleBranch . . . 22
4.2.3 Bifurcartion . . . 24
4.2.4 Rhombus . . . 26
4.2.5 Conclusion . . . 26

5 C++ code **27**
5.1 CurvedFormulation 27
5.1.1 c_problem3d1d 28
5.1.2 Mesh and Parameters 31
5.1.3 Assembling 35
5.1.4 The library c_problem3d1d 36
5.2 GraphGenerator 36
5.2.1 Implementation 36
5.2.2 The library graphgenerator 37

Introduction

The main goal of this project is to develop a general-purpose finite element solver for large scale simulation of microcirculation for a generic network geometry.In particular we want to extend the previous models, which are able to solve the problem only for rectilinear networks, to one able to take in account the presence of curved branches.
Thanks to dimensional model reduction techniques, the vessel tree is described as a one-dimensional (1D) manifold immersed in a three-dimesional (3D) interstitial volume. Vessels can be seen as concentrated sources lead to reduce the computational cost of simulations. However, concentrated sources lead to singular solutions that still require computationally expensive meshes to guarantee accurate approximation. The main computational barrier consists in the ill-posedness of restriction operators applied on manifolds with co-dimension larger than one. We overcome the computational challenges of approximating PDEs on manifolds with high dimensionality gap by means of nonlocal restriction operators that combine standard trace with mean values of the solution on low dimensional manifolds.

Chapter 1

Coupled 3D/1D Problem

1.1 Model set up

The domain in $\mathbb{R}^3$ where the model is defined is composed by two parts, Ω_t and Ω_v, denoting the interstitial volume and the vessel bed respectively. Assuming thet the vessels can be described as cylindrical tubes, we denote with Γ the outer surface of Ω_v while Λ indicates the 1D manifold representing the vessel centerline. The vessel radius R is generally subject to change in the network.
We consider the interstitial volume Ω_t as an isotropic porous medium, descibed by the Darcy's law. Ignoring inertial and body forces, it reads:

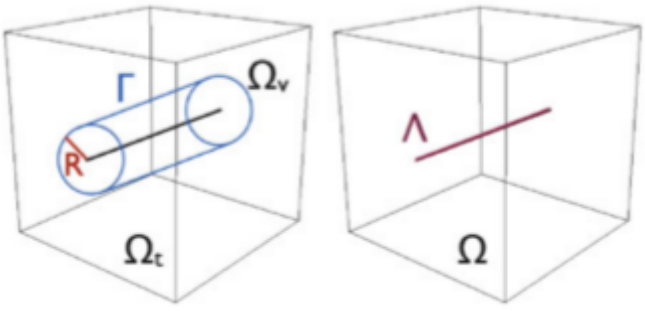
$$\mathbf{u}_t = -\frac{1}{\mu}\mathbb{K}\nabla p_t \tag{1.1}$$

where $\mathbf{u}_t$ is average filtration velocity vector in the tissue, $\mathbb{K}$ is the permeability tensor, μ is the viscosity of the fluid and p_t is the fluid pressure. Recall that in the simple isotropic case the permeability tensor is given by $\mathbb{K} = k\mathbb{I}$, being k the scalar permeability and $\mathbb{I}$ the identity tensor.

Concerning the vessels, we start assuming a steady incompressible Navier-Stokes model for blood flow, namely:

$$\rho(\mathbf{u}_v \cdot \bigtriangledown)\mathbf{u}_v - \mu \bigtriangleup \mathbf{u}_v + \bigtriangledown p_v = 0 \tag{1.2}$$

where $\mathbf{u}_v$ is the flow velocity, p_v is the pressure and ρ the density of the fluid.



1.2 Coupling microcirculation with interstitial flow

At this stage the two problems are completely uncorrelated. Indeed to close the problem we need to impose the continuity of the flow at interface $\Gamma = \partial\Omega_v \cap \partial\Omega_t$, namely:

$$\mathbf{u}_v \cdot \mathbf{n} = \mathbf{u}_t \cdot \mathbf{n} = L_p(p_v - p_t), \mathbf{u}_v \cdot \boldsymbol{\tau} = 0, \tag{1.3}$$

where $\mathbf{n}$ and $\boldsymbol{\tau}$ are the outward unit normal vector and the unit tangent vector on surface Γ, respectively, while L_p is the hydraulic conductivity of the vessel wall. This equation together to the previous ones identify a fully three-dimensional model able to capture approximation of the phenomena we are interested in. However, many technical difficulties arise in the numerical approximation of the coupling between a complex network with surrounding volume. To this purpose, we adopt the multiscale approach to avoid the complex 3D geometry of the network. We exploit the *Immersed Boundary Method (IBM)* combined with the assumption of large aspect ratio between vessel radius and capillary axial length. More precisely, we apply a suitable rescaling of the equation and let the capillary radius go to zero ($R \to 0$).
As a consequence, the three-dimensional description of the vessels is reduced to a simplified one-dimensional representation by replacing the immersed interface and the related interface conditions with an equivalent mass source, namely:

$$\mathbf{u}_t \cdot \mathbf{n} = f(p_v, p_t) \quad on \quad \Gamma, \tag{1.4}$$

being f the flux per unit area released through surface Γ: it is a point-wise constitutive law for the capillary leakage in term of the fluid pressure.

1.3 Derivation of the 1D approximation in case of rectilinear axis

Here we introduce the simplest non-linear 1D flow in compliant vessels.
The basic equations are derived for a track of vessel free of bifurcations, which is idealised as a cylindrical compliant tube.
We will mainly use the Cartesian coordinates, but when dealing with cylindrical geometries it is handy to introduce a cylindrical coordinate system. Therefore, in the following we indicate with $\boldsymbol{\epsilon}_r$, $\boldsymbol{\epsilon}_\theta$ and $\boldsymbol{\epsilon}_z$ the radial circumferential and axial unit vectors, respectively, (r, θ, z) being the corresponding coordinate.

1.3.1 Model Assumptions

The basic model is deduced by making the following assumptions:

1. **Axial symmetry.** All quantities are independent from the angular coordinate θ. As a consequence, every axial section $z = const$ is assumed circular on all the vessel. The tube radius R is a function of z.

2. **Fixed cylinder axis.** This simply means that the axis is assumed as rectilinear for the whole vessel. This hypothesis is indeed consistent with that of axial symmetry.

3. **Constant pressure on each section.** We assume that the pressure P is constant on each section. This comes as result of the following count:
   using the axial symmetry we have $\mathbf{u}_v(r, \theta, z) = u_z(r, \theta, z)\boldsymbol{\epsilon}_z$ and splitting the equation [1.2] by components we obtain

$$\begin{cases} (\mathbf{u} \cdot \nabla)u_r - \nu\triangle u_r + \partial_r p/\rho = 0 \\ (\mathbf{u} \cdot \nabla)u_\theta - \nu\triangle u_\theta + \partial_\theta p/\rho = 0 \\ (\mathbf{u} \cdot \nabla)u_z - \nu\triangle u_z + \partial_z p/\rho = 0 \end{cases} \quad \Rightarrow \quad \begin{cases} \partial_r p = 0 \\ \partial_\theta p = 0 \\ \rho(\mathbf{u} \cdot \nabla)u_z - \mu\triangle u_z + \partial_z p = 0 \end{cases} \tag{1.5}$$

So we see that p depends only on z.

4. **No body forces.** We neglect body forces.

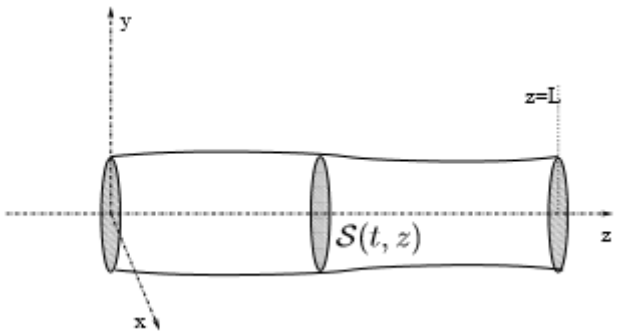
5. **Dominance of axial velocity.** The velocity components orthogonal to the z axis are negligible compared to the component along z. The latter is indicate by u_z and its expression in cylindrical coordinates is supposed to be of the form

$$u_z(r,z) = \overline{u}(z)s(rR^{-1}(z)) \tag{1.6}$$

where $\overline{u}$ is the mean velocity on each section, $s : \mathbb{R} \to \mathbb{R}$ is the velocity profile and reads as

$$s(y) = \gamma^{-1}(\gamma+2)(1+y^{\gamma}). \tag{1.7}$$

The fact that the velocity profile does not vary in space is in contrast with experimental observations and numerical results carried out with full-scale models. However, it is a necessary condition for the derivation of the reduced model. One may think to s as being a profile representative of an average flow configuration.



A generic axial section will be indicated by $S = S(z)$. Its measure A is given by

$$A(z) = \int_{S(z)} d\sigma = \pi R^2(z). \tag{1.8}$$

The mean velocity $\overline{u}$ is given by

$$\overline{u} = A^{-1} \int_S u_z d\sigma. \tag{1.9}$$

We will indicate with α the momentum-flux correction coefficient, (sometimes also called Coriolis coefficent) defined as

$$\alpha = \frac{\int_S u_z^2 d\sigma}{A\overline{u}^2} = \frac{\int_S s^2 d\sigma}{A} \tag{1.10}$$

where the dependence of various quantities on the spatial coordinates is easy to understand. In general this coefficient will vary in time and space, yet in our model it is taken constant as a consequence of [1.6].

1.3.2 Model Derivation

To derive our model, we use an approach consisting of integrating the Navier-Stokes equations on a generic portion P of the vessel. Moreover we can provide the vessel wall in parametric form:

$$\Gamma = \{(r, \theta, z) : r = R(z), \theta \in [0, 2\pi), z \in (0, L)\} \tag{1.11}$$

where L is the length of the vessel in the axial direction and $\mathbf{n}$ is the out oriented normal to $\partial\Omega_v$. Under the previous assumption, the momentum along z and continuity equations, in the hypothesis of constant viscosity, are

$$\begin{cases} \rho(\mathbf{u} \cdot \nabla)u_z - \mu\triangle u_z + \partial_z p = 0 \\ div\mathbf{u} = 0 \end{cases} \tag{1.12}$$

The convective term in the momentum equation has been taken in divergence form because it simplifies the further derivation. Now we are ready to derive our reduced model and we start first from the continuity equation

$$\begin{aligned} 0 = \int_P [div\mathbf{u}]d\Omega \quad = \int_{\partial P} [\mathbf{u} \cdot \mathbf{n}]d\sigma \quad &= \int_{S(z_1)} [\mathbf{u} \cdot \mathbf{n}]d\sigma \quad + \int_{S(z_2)} [\mathbf{u} \cdot \mathbf{n}]d\sigma \quad + \int_\Gamma [\mathbf{u} \cdot \mathbf{n}]d\sigma \quad = \\ &= -\int_{S(z_1)} [u_z]d\sigma \quad + \int_{S(z_2)} [u_z]d\sigma \quad + \int_\Gamma [f(\overline{p}_t, p_v)]d\sigma \end{aligned} \tag{1.13}$$

using the fact that $\mathbf{n} = \boldsymbol{\epsilon_z}$ on $S(z_1)$ and $S(z_2)$ and the equation [1.4]. Now using equation [1.9] for the first two integrals and a change of variable on the third one we obtain:

$$\begin{aligned} [1.13] = -\overline{u}(z_1)A(z_1) + \overline{u}(z_2)A(z_2) + \int_{z_1}^{z_2} [f(\overline{p_t}, p_v)]dz = \\ = \int_{z_1}^{z_2} [f(\overline{p_t}, p_v) + \partial_z(A\overline{u})]dz \end{aligned} \tag{1.14}$$

since $\overline{p_t} = \frac{1}{2\pi R}\int_{\partial S} p_t ds$ is the mean interstitial pressure on the boundary of a section S and the fundamental theorem of calculus hold. At the end we use the arbitrary of z_1 and z_2 to obtain

$$\partial_z(A\overline{u}) + f(\overline{p_t}, p_v) = 0 \tag{1.15}$$

Now it's the turn of the momentum equation. Lets start with the Stokes term:

$$\int_P [\partial_z p]d\Omega = \int_{z_1}^{z_2} dz \int_{S(z)} [\partial_z p]d\sigma = \int_{z_1}^{z_2} [A(z)\partial_z p]dz \tag{1.16}$$

Now we manipulate the advective term in divergence form:

$$\int_P [div(u_z\mathbf{u})]d\Omega = \int_{\partial P} [u_z\mathbf{u} \cdot \mathbf{n}]d\sigma = -\int_{S(z_1)} [u_z^2]d\sigma + \int_{S(z_2)} [u_z^2]d\sigma + \int_\Gamma [u_z\mathbf{u} \cdot \mathbf{n}]d\sigma \tag{1.17}$$

In order to eliminate the boundary integral we exploit the fact that $u_z = 0$ on Γ. Subsequently using equation [1.10] and the fundamental theorem of calculus we obtain:

$$-\int_{S(z_1)} [u_z^2]d\sigma + \int_{S(z_2)} [u_z^2]d\sigma = \alpha[-\overline{u}(z_1)^2 A(z_1) + \overline{u}(z_1)^2 A(z_1)] = \int_{z_1}^{z_2} \alpha\partial_z[A(z)\overline{u}(z)^2]dz \tag{1.18}$$

We finally consider the viscous term:

$$\int_P [\triangle u_z]d\Omega = \int_{\partial P} [\nabla u_z \cdot \mathbf{n}]d\sigma = -\int_{S(z_1)} [\partial_z u_z]d\sigma + \int_{S(z_2)} [\partial_z u_z]d\sigma + \int_\Gamma [\nabla u_z \cdot \mathbf{n}]d\sigma. \tag{1.19}$$

Here, we neglect the $\partial_z u_z$ term by the fact that $0 = div(\mathbf{u}) = \partial_z u_z$ almost everywhere in Ω_v. Moreover we split the $\mathbf{n}$ vector into two components, the radial component $\mathbf{n}_r = (\mathbf{n} \cdot \boldsymbol{\epsilon}_r)\boldsymbol{\epsilon}_r = n_r \boldsymbol{\epsilon}_r$ and axial component $\mathbf{n}_z = \mathbf{n} - \mathbf{n}_r$. Remember that owing to the cylindrical geometry, $\mathbf{n}$ has no component along the circumferential coordinate and, consequently, $\mathbf{n}_z$ is indeed oriented along z. Thus, we can write:

$$\int_P [\triangle u_z] d\Omega = \int_\Gamma [\nabla u_z \cdot \mathbf{n}_r + \nabla u_z \cdot \mathbf{n}_z] d\sigma. \tag{1.20}$$

Again, we neglect the $\nabla u_z \cdot \mathbf{n}_z$, which is proportional to $\partial_z u_z$. We recall now the relation [1.6] to write:

$$\int_\Gamma [\nabla u_z \cdot \mathbf{n}_r] d\sigma = \int_\Gamma [n_r \nabla u_z \cdot \boldsymbol{\epsilon}_r] d\sigma = \int_\Gamma [\overline{u} R^{-1} s'(1) n_r] d\sigma = \int_{z_1}^{z_2} [2\pi s'(1) \overline{u}] dz \tag{1.21}$$

where we used $n_r d\sigma = 2\pi R dz$ and indicating s' as the first derivative of s. Then using the arbitrary of the extreme on [1.16], [1.18], [1.21] and calling $K_r = 2\pi\nu s'(1)$ we obtain

$$K_r \overline{u} + \frac{A}{\rho} \partial_z p + \alpha \partial_z (A \overline{u}^2) = 0 \tag{1.22}$$

1.3.3 Full linear coupled equations

Now that we have derived the 1D model equatiosn we need to generalize them to a more complex topology. To this purpose, we decompose the network in Λ_i branches, $i = 1, ..., N$. The branches are parametrized by the arc length s_i; a tangent unit vector $\boldsymbol{\lambda}_i$ is also defined over each branch, accounting for an arbitrary branch orientation. Differentiation over the branches is defined using the tangent unit vector, namely $\boldsymbol{\partial}_{s_i} := \nabla \cdot \boldsymbol{\lambda}_i$ on Λ_i, i.e. $\boldsymbol{\partial}_{s_i}$ represents the projection of ∇ along $\boldsymbol{\lambda}_i$.
So the coupled 3D-1D problem we have to solve, in case of vessel with rectilinear axis, is:

$$\begin{cases} \mathbf{u}_t + \frac{1}{\mu} \mathbb{K} \nabla p_t = 0 & on \quad \Omega_t \\ div(\mathbf{u}_t) - 2\pi R L_p (p_v - \overline{p_t}) \delta_\Lambda = 0 & on \quad \Omega_t \\ K_r \overline{u} + \frac{A}{\rho} \partial_z p + \alpha \partial_z (A \overline{u}^2) = 0 & on \quad \Omega_v \\ \partial_z (A \overline{u}) + 2\pi R L_p (p_v - \overline{p_t}) = 0 & on \quad \Omega_v \end{cases} \tag{1.23}$$

1.3.4 Boundary conditions

For the well-poseness of problem [1.23], we have to specify suitable boundary conditions ($\mathbf{BC}_s$) on both the tissue and vessel boundary, i.e. $\partial\Omega$ and $\partial\Lambda$ respectively.
Since we aim to present the most generic setting, we assume the tissue interstitium boundary to be partitioned as follows:

$$\partial\Omega = \Gamma_p \cup \Gamma_u, \quad \mathring{\Gamma}_p \cap \mathring{\Gamma}_u = \emptyset. \tag{1.24}$$

As suggested by the apices p and u, we enforce a given pressure distribution on Γ_p and/or a fixed value for the normal flux over Γ_u, namely:

$$p_t = g_t \qquad on \quad \Gamma_p, \tag{1.25}$$

$$\mathbf{u}_t \cdot \mathbf{n} = \beta(p_t - p_0) \qquad on \quad \Gamma_u, \tag{1.26}$$

Here p_0 represent far field pressure value, while β can be interpreted as an effective conductivity accounting for layers of tissue surrounding the considered sample. Assuming that the interstitial pressure decay from p_t to p_0 over a distance comparable to the sample characteristic size, D, dimensional analysis shows that a rough estimate of conductivity is $\beta = k_t/D$. For the pressure datum we require $g_t \in L^2(\Gamma_p)$.
Concerning the network, we split the collection of extrema into two subset: on the *boundary extrema*,

$\boldsymbol{\varepsilon}_p$, we enforce a pressure distribution, on *immersed extrema* , $\boldsymbol{\varepsilon}_u$, we enforce the flux (hence the vessels velocity); namely:

$$p_v = g_v \qquad on \quad \boldsymbol{\varepsilon}_p \tag{1.27}$$

$$\pi R'^2 u_v = \beta(p_v - p_0) \qquad on \quad \boldsymbol{\varepsilon}_u \tag{1.28}$$

where g_v is a boundary datum for which is required measurability and square-summability, namely $g_v \in L^2(\boldsymbol{\varepsilon}_p)$, while p_0 and β are as above. In particular, in future applications we will always enforce constant pressure drop $P_v^{out} - P_v^{in}$, that means we will adopt piecewise-constant boundary data.

1.4 Derivation of the 1D approximation for curved segments

1.4.1 Preliminary Model Assumptions

Now we want to derive similar results in the case of curvilinear axis.
To do this we define the parametric arc length $\boldsymbol{\Psi} : \mathbb{R} \to \mathbb{R}^3$ as the axis trajectory, taken such that

$$\begin{aligned} \boldsymbol{\Psi} &\in C^3(\mathbb{R}) \\ |\boldsymbol{\Psi}'(z)| = 1 \quad &\forall z \in [0, L], \end{aligned} \tag{1.29}$$

where L is the length of the arc and the norm is compute as the Euclidean norm. In the following we will use parametric cylindrical coordinate, and we indicate $\boldsymbol{\epsilon}_{r(z)}$, $\boldsymbol{\epsilon}_{\theta(z)}$ and $\boldsymbol{\epsilon}_{\psi(z)}$ respectively, the radial, circumferential and axial unit vectors, $(r, \theta, \psi)(z)$ being the corresponding coordinates at each position z and, due to the choose of $\boldsymbol{\Psi}$, we obtain that $\boldsymbol{\psi(z)} = \boldsymbol{\Psi}'(z)$. From now on we will omit the parametric dependence. To derive the model we need these preliminary assumptions:

1. **Circular Section.** For each value of the parameter z the intersection between the orthogonal plane to $\boldsymbol{\psi}$ and the vessel wall is assumed as circular.

2. **Constant pressure on each section.** The pressure is assumed constant on each section due to the same reason of the rectilinear case.

3. **No body forces.** We neglect all forces effect.

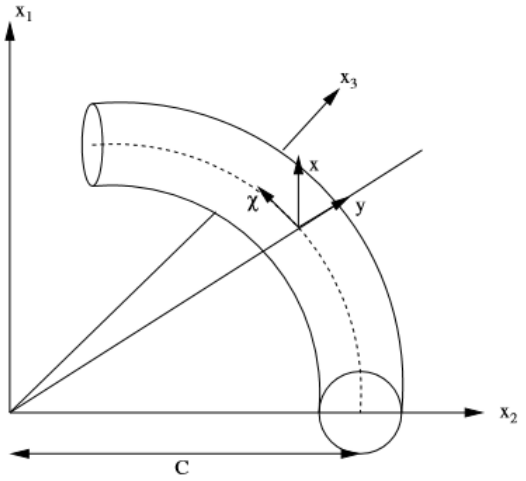
4. **Dominance of axial velocity.** The velocity components orthogonal to the ψ axis are negligible compared to the component along ψ. The latter is indicate by u_ψ and its expression is supposed to be of the form:

   $$u_\psi(r, \theta, \psi) = \overline{u}(\psi)\Phi(r, \theta, \psi), \tag{1.30}$$

   where $\overline{u}$ is the mean velocity on each axial section and $\Phi : \mathbb{R}^3 \to \mathbb{R}$ is the velocity profile on the curvilinear vessel. As for the linear case, this profile can also be seen as the average flow configuration.

1.4.2 Derivation of the average velocity profile

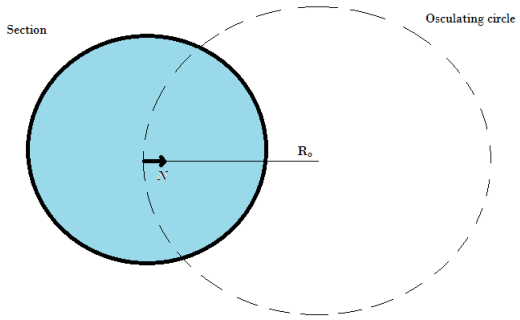
To derive the velocity profile we are going to study it in the simpler case of a curve vessel with fixed curvature and we will generalize it in a more complete problem adding more hypothesis.



Surely there are many possible ways to choose the configuration of Φ, but we are going to use the following second order approximation:

$$\Phi(r,\theta,\psi) = s(rR^{-1})(1 + ar\cos\theta + br\sin\theta + cr^2\cos\theta\sin\theta + dr^2\cos^2\theta + er^2\sin^2\theta), \tag{1.31}$$

where a,b,c,d and e are constants depending on the curvature and s is the velocity profile of the linear case. To improve the significance of this configuration is possible to add higher order term and compute the respectively coefficients.
Now to better understand the procedure, let's take for example a uniform curvilinear motion. On it we have that the tangential velocity is proportional to the distance from the center, so in out problem we are going to expect similar result.
Given κ, the curvature of the arc, defined as $\kappa = |\mathbf{\Psi}''|$, and the centripetal unitary direction $\mathbf{N} = \mathbf{\Psi}''/\kappa$, we have that the center C_0 of the *osculating circle* is the point in direction $\mathbf{N}(z)$ whose distance from $\mathbf{\Psi}(z)$ is R_0, where R_0 is the curvature radius and it's equal to $1/\kappa$.



Now we have to impose suitable condition to our velocity profile in order to compute the value of all the constants:

1. **Choice of θ.** To simplify our calculation we assume that on each section all the vectors with $\theta = 0$ are in $\mathbf{N}$ direction.

2. **Symmetry of the profile.** In each section the profile must be symmetric with respect to the axes connecting C_0 and $\mathbf{\Psi}(z)$. So:

   $$\Phi(r,\theta,\psi) = \Phi(r,-\theta,\psi) \quad \forall r,\theta,\psi$$

   and we obtain that $b = c = 0$.

3. **Zero velocity in the Center.** For the linear dependence of the velocity with the distance from the center of the osculating circle, our profile must be zero in $C_0 = (r = 1/\kappa, \theta = 0, \psi)$.

$$\Phi(C_0) = 0 \rightarrow (1 + a/\kappa + d/\kappa^2) = 0 \rightarrow d = -a\kappa - \kappa^2$$

4. **Linear dependence.** Due to the assumptions that the velocity profile is linear dependent to the distance from the center of the osculating circle we have that all the points with distance $1/\kappa$ from it must have the same dependence.
All these are the points having the following property:

$$\phi = \{(r, \theta) : \quad r = \frac{2\cos\theta}{\kappa}, \quad \theta \in [-\frac{\pi}{2}; +\frac{\pi}{2}]\}.$$

Moreover we have that

$$\Phi(r = 0, \theta, \psi) = s(0)$$

so $\forall (r, \theta) \in \phi$ then $\Phi(r, \theta, \psi) = s(r/R)$. It follows that $\forall (r, \theta) \in \phi$:

$$0 = ar\cos\theta + dr^2\cos^2\theta + er^2\sin^2\theta = 2\frac{a}{\kappa}\cos^2\theta + 4\frac{d}{\kappa^2}\cos^4\theta + 4\frac{e}{\kappa^2}\cos^2\theta\sin^2\theta.$$

Now for $\theta = \pm\frac{\pi}{2}$ the equation is verified. In the other cases we can divide all by $2\cos^2\theta/\kappa^2$, to obtain:

$$0 = a\kappa + 2d\cos^2\theta + 2e\sin^2\theta \qquad \forall\theta \in (-\frac{\pi}{2}; +\frac{\pi}{2}).$$

To find the value of the constant, we test it on two particular cases: $\theta = \pi/4, \theta = \pi/3$.
For $\theta = \pi/4$, using assumption 3:

$$0 = a\kappa + 2d(\frac{1}{2}) + 2e(\frac{1}{2}) = a\kappa + d + e = a\kappa - \kappa^2 - a\kappa + e = e - \kappa^2.$$

For that $e = \kappa^2$. Instead for $\theta = \pi/3$, using assumption 3 and the previous result:

$$0 = a\kappa + 2d(\frac{1}{4}) + 2e(\frac{3}{4}) = a\kappa + \frac{d}{2} + \frac{3e}{2} = a\kappa - \frac{\kappa^2}{2} - \frac{a\kappa}{2} + \frac{3\kappa^2}{2} = \frac{a\kappa}{2} + \kappa^2$$

So $a = -2\kappa$ and $d = \kappa^2$.

Due to this assumptions, our velocity profile is of the form:

$$\Phi(r, \theta, \psi) = s(rR^{-1})(1 + r^2\kappa^2 - 2\kappa r\cos\theta). \tag{1.32}$$

In a general geometry we have that the curvature is dependent on the trajectory position: $\kappa = \kappa(\psi)$. But thanks to the assumption of regularity on the trajectory, $\mathbf{\Psi} \in C^3(\mathbb{R})$, then $\kappa(\psi) \in C^1(\mathbb{R})$ and our velocity profile still holds, by continuity of κ:

$$\Phi(r, \theta, \psi) = s(rR^{-1})(1 + r^2\kappa^2(\psi) - 2\kappa(\psi)r\cos\theta). \tag{1.33}$$

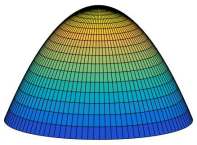


Figure 1.1: $\kappa R = 0$

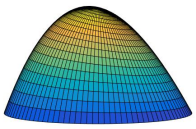


Figure 1.2: $\kappa R = 0.1$

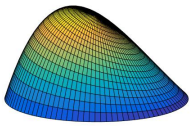


Figure 1.3: $\kappa R = 0.3$

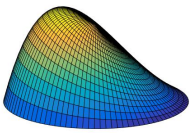


Figure 1.4: $\kappa R = 0.5$

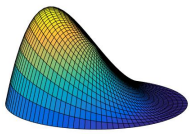


Figure 1.5: $\kappa R = 1.0$

Figure 1.6: Velocity profile with imposition of Poiseuille flow ($\gamma = 2$) for different curvature value. As we can see from these plots, we have that for increasing curvature value, the velocity profile on the section starts to concentrate farther from the center of the osculating circle.

1.4.3 Model derivation for curved segments

Now that we have the velocity profile, the derivation of the equation is the same of the rectilinear case, the only difference is for the coefficients which depend on the profile.
In fact the integral of an integrable function $f : \mathbb{R}^3 \to \mathbb{R}$ on trajectory $\mathbf{\Psi}$ is

$$\int_{\mathbf{\Psi}} [f(\psi)] d\psi = \int_0^L [f(\mathbf{\Psi}(z))|\mathbf{\Psi}'|] dz = \int_0^L [f(\mathbf{\Psi}(z))] dz. \tag{1.34}$$

To denote our portion P we can use two arbitrary position z_1 and $z_2 \in [0, L]$, $z_1 < z_2$ and we obtain the same results.
The first difference is on equation [1.10]:

$$\alpha^{**} = \frac{\int_S [u_\psi^2] d\sigma}{A\overline{u}^2} = \frac{\int_S [\Phi^2] d\sigma}{A} = \frac{\int_S [s^2(1 - 4\kappa r \cos\theta + 4\kappa^2 r^2 \cos^2\theta + 2\kappa^2 r^2 + \kappa^4 r^4)] d\sigma}{A}.$$

Now for the periodicity of $\cos\theta$:

$$\alpha^{**} = \frac{\int_S [s^2(1 + 4\kappa^2 r^2 \cos^2\theta + 2\kappa^2 r^2 + \kappa^4 r^4)] d\sigma}{A} = \alpha + \kappa^2\beta + \kappa^4\gamma, \tag{1.35}$$

where α is the momentum-flux correction coefficient for the rectilinear case,

$$\begin{aligned}\beta =& \frac{\int_S [2s^2r^2(1+\cos^2\theta)]d\sigma}{A} \\ \gamma =& \frac{\int_S [s^2r^4]d\sigma}{A}\end{aligned} \tag{1.36}$$

On the other hand, for the permeability term we have:

$$\begin{aligned}\int_P [\triangle u_\psi]d\Omega &= \int_\Gamma [\overline{u}\partial_r p n_r]d\sigma = \int_\Gamma \{\overline{u}\partial_r[s(rR^{-1})(1-2\kappa r\cos\theta+\kappa^2r^2]\}n_r d\sigma = \\ &= \int_\Gamma \{\overline{u}[R^{-1}s'(rR^{-1})(1-2\kappa r\cos\theta+\kappa^2r^2)+(\kappa^2 r-2\kappa\cos\theta)s(rR^{-1})]\} = \\ &= \int_{\mathbf{\Psi}}\int_0^{2\pi} \{\overline{u}R[R^{-1}s'(1)(1-\kappa\cos\theta+\kappa^2R^2)+(\kappa^2R-2\kappa\cos\theta)s(1)]\}d\theta d\psi.\end{aligned}$$

Now using the fact that $s(1)=0$, the periodicity of $\cos\theta$ and doing a change of variable we obtain

$$\int_P [\triangle u_\psi]d\Omega = \int_{z_1}^{z_2} [2\pi s'(1)(1+\kappa^2R^2)\overline{u}(t)]dt, \tag{1.37}$$

so the permeability term becomes $K_r^{**} = K_r(1+\kappa^2R^2)$, where K_r is the permeability coefficient of the rectilinear case.
Assembling all these terms we obtain:

$$K_r^{**}\overline{u} + \frac{A}{\rho}\partial_z p_v + \partial_z(\alpha^{**}A\overline{u}^2) = 0. \tag{1.38}$$

1.4.4 Full curved coupled equations

Now, as for the linear coupled model, we need to generalize the equations to a more complex topology. As before we decompose the network in Λ_i branches, $i=1,...,N$. The branch is parametrized by the arc length $\mathbf{\Psi}_i$; and the tangent unit vector defined as $\boldsymbol{\lambda}_i := \mathbf{\Psi}_i'$ over each branch, accounting for an arbitrary branch orientation. Differentiation over the branch is defined using the tangent unit vector, namely $\boldsymbol{\partial}_{s_i} := \nabla\cdot\boldsymbol{\lambda}_i$ on Λ_i, i.e. $\boldsymbol{\partial}_{s_i}$ is the projection of ∇ along $\boldsymbol{\lambda}_i$.
So the coupled 3D-1D problem's equations read as:

$$\begin{cases} \mathbf{u}_t + \frac{1}{\mu}\mathbb{K}\nabla p_t = 0 & on \quad \Omega_t \\ div(\mathbf{u}_t) - 2\pi R L_p(p_v - \overline{p_t}) = 0 & on \quad \Omega_t \\ K_r^{**}\overline{u}_v + \frac{A}{\rho}\partial_{s_i}p_v + \alpha^{**}\partial_{s_i}(A\overline{u}_v^2) = 0 & on \quad \Lambda_i \quad \forall \quad i=1,...,N \\ \partial_{s_i}(A\overline{u}_v) + 2\pi R L_p(p_v - \overline{p_t}) = 0 & on \quad \Lambda_i \quad \forall \quad i=1,...,N \end{cases} \tag{1.39}$$

1.5 Dimensional Analysis

Now, let's rewrite now the problem in dimensionless form in order to highlight the most significant mechanisms governing the flow between microcirculation and biological tissue, under the assumption of constant radius over each branch of the vessel. First, we indentify the characteristic dimension of our problem: length, velocity and pressure are chosen as primary variables for the analysis. The corresponding characteristic values are: (I) the average spacing between capillary vessel d, (II) the

average velocity in the capillary bed U and (III) the average pressure in the interstitial space P. Correspondingly, the dimensionless groups affecting our equations are:

$$
\begin{aligned}
R' &= \frac{R}{d} && \textit{dimensionless radius} \\
\kappa' &= \frac{\kappa}{d} && \textit{dimensionless curvature} \\
k_t &= \frac{kP}{\mu U d} && \textit{dimensionless interstitial permeability} \\
Q &= 2\pi R' L_p \frac{P}{U} && \textit{dimensionless wall permeability} \\
k_v &= \frac{\pi R'^4}{K_r}\frac{Pd}{U} = \frac{\pi R'^4}{2\mu(\gamma+2)}\frac{Pd}{U} && \textit{dimensionless vessel permeability} \\
G &= \frac{\rho U^2}{P} && \textit{dynamic pressure gain}
\end{aligned}
$$

Therefore, the coupled dimensionless problem of microcirculation and tissue interstitium reads as follows:

$$
\left\{
\begin{array}{lllllll}
a) & \frac{1}{k_t}\mathbf{u}_t + \nabla p_t = 0 & on & \Omega_t & & & \\
b) & div(\mathbf{u}_t) - Q(p_v - \overline{p_t}) = 0 & on & \Omega_t & & & \\
c) & \frac{\pi R'^2}{k_v}(1+\kappa'^2 R'^2)\overline{u}_v + \partial_{s_i} p_v + G\partial_{s_i}(\alpha^{**}\overline{u}_v^2) = 0 & on & \Lambda_i & \forall & i = 1, ..., N \\
d) & \partial_{s_i}(\overline{u}_v) + \frac{Q}{2\pi R'^2}(p_v - \overline{p_t}) = 0 & on & \Lambda_i & \forall & i = 1, ..., N
\end{array}
\right. \tag{1.40}
$$

Chapter 2

Mixed Formulation for velocity and pressure

2.1 Weak Formulation for tissue interstitium

First of all we need to take suitable spaces for the velocity and the pressure in the tissue.

$$\mathbf{V_t} := H^{div}_{\alpha,\beta}(\Omega_t) \quad and \quad Q_t = L^2_\alpha(\Omega_t) \tag{2.1}$$

with $\alpha, \beta \in (-1, 1)$.
After that we are going to multiply the equations [1.35 a] and [1.35 b] with sufficiently smooth functions and integrate over the volume Ω_t, namely:

$$\int_{\Omega_t} [\frac{1}{k_t}\mathbf{u}_t \cdot \mathbf{v}_t] d\Omega + \int_{\Omega_t} [\nabla p_t \cdot \mathbf{v}_t] d\Omega = 0 \tag{2.2}$$

$$\int_{\Omega_t} [div(\mathbf{u}_t) q_t] d\Omega - Q \int_{\Omega_t} [(p_v - \overline{p_t})\delta_\Lambda q_t] d\Omega = 0 \tag{2.3}$$

Now applying the Green's theorem to [2.2] we obtain an anti-symmetric formulation of Darcy's problem in the tissue:

$$\int_{\Omega_t} [\frac{1}{k_t}\mathbf{u}_t \cdot \mathbf{v}_t] d\Omega - \int_{\Omega_t} [div(\mathbf{v}_t) p_t] d\Omega + \int_{\partial\Omega_t} [pt\mathbf{v}_t \cdot \mathbf{n}] d\sigma = 0 \tag{2.4}$$

Finally, let's discuss the treatment of the boundary term. Therefore, thanks to the linearity of the integral we can rewrite the integral on Γ_p and Γ_u, which are the boundary where are imposed respectively the pressure and velocity boundary conditions.

$$\int_{\partial\Omega_t} [pt\mathbf{v}_t \cdot \mathbf{n}] d\sigma = \int_{\Gamma_p} [g_t \mathbf{v}_t \cdot \mathbf{n}] d\sigma + \int_{\Gamma_u} [p_0 \mathbf{v}_t \cdot \mathbf{n}] d\sigma + \frac{1}{\beta} \int_{\Gamma_u} [(\mathbf{u}_t \cdot \mathbf{n})(\mathbf{v}_t \cdot \mathbf{n})] d\sigma \tag{2.5}$$

So the weak formulation for the tissue interstitium reads as:

$$\begin{aligned} \int_{\Omega_t} [\frac{1}{k_t}\mathbf{u}_t \cdot \mathbf{v}_t] d\Omega + \frac{1}{\beta} \int_{\Gamma_u} [(\mathbf{u}_t \cdot \mathbf{n})(\mathbf{v}_t \cdot \mathbf{n})] d\sigma - \int_{\Omega_t} [div(\mathbf{v}_t) p_t] d\Omega = \\ = - \int_{\Gamma_p} [g_t \mathbf{v}_t \cdot \mathbf{n}] d\sigma - \int_{\Gamma_u} [p_0 \mathbf{v}_t \cdot \mathbf{n}] d\sigma \end{aligned} \tag{2.6}$$

$$\int_{\Omega_t} [div(\mathbf{u}_t) q_t] d\Omega - Q \int_{\Omega_t} [(p_v - \overline{p_t})\delta_\Lambda q_t] d\Omega = 0 \tag{2.7}$$

2.2 Weak formulation for the vessel

As for the vessel problem we start giving a general functional framework. At this point we only require regularity for the vessel velocity and pressure over each branch in a separate way.

$$V_v = \bigcup_{i=1}^{N} H^1(\Lambda_i) \quad and \quad Q_v = \bigcup_{i=1}^{N} L^2(\Lambda_i). \tag{2.8}$$

Now at first thing we rescale equation [1.35 c] and [1.35 d] by the function $\pi R'^2(z)$, therefore we multiply the resulting equations by sufficiently smooth function, v_v and q_v, and integrate over the vessel domain Λ:

$$\int_\Lambda [\frac{\pi^2 R'^4}{k_v}(1+\kappa'^2 R'^2)\overline{u}_v v_v]ds + \int_\Lambda [\pi R'^2 \partial_s p_v v_v]ds + \int_\Lambda [\pi R'^2 G \partial_s(\alpha^{**}\overline{u}_v^2)v_v]ds = 0 \tag{2.9}$$

$$\int_\Lambda [\partial_s(\overline{u}_v)q_v]ds + \int_\Lambda [\frac{Q}{2\pi R'^2}(p_v - \overline{p_t})q_v]ds = 0 \tag{2.10}$$

The integration by parts is not trivial in such a case because the vessel variables u_v and p_v may be discontinuous at multiple junctions. Specifically, we assume that vessels pressure is continuous while the difficulty associate with the vessel velocity remains. Let us consider the second integral of [2.9]. We will derive a proper Green's formula for the network problem. First, we rewrite the integral over the whole network as a summation of the integrals over single branches, namely:

$$\int_\Lambda [\pi R'^2 \partial_s p_v v_v]ds = \sum_{i=1}^{N} \int_{\Lambda_i} [\pi R'^2 \partial_s p_v v_v]ds.$$

Let's assume the vessel radius to be a step function of the arc length $\mathbf{\Psi}$ with constant values over the one-dimensional varieties of Λ_i:

$$R'(s) = \sum_{i=1}^{N} R'_i \delta_{\Lambda_i},$$

being δ_{Λ_i} the Dirac delta function on the $i-th$ branch. Thus the integral becomes

$$\int_\Lambda [\pi R'^2 \partial_s p_v v_v]ds = \sum_{i=1}^{N} \pi R'^2_i \int_{\Lambda_i} [\partial_s p_v v_v]ds$$

and we can finally apply the standard Green's formula over the branch Λ_i:

$$\begin{aligned} \int_\Lambda [\pi R'^2 \partial_s p_v v_v]ds &= \sum_{i=1}^{N} \pi R'^2_i \{ - \int_{\Lambda_i} [p_v \partial_s v_v]ds + [v_v p_v]_{\Lambda_i^-}^{\Lambda_i^+} \} = \\ &= - \int_\Lambda [\pi R'^2_i \partial_s p_v]ds + \sum_{i=1}^{N} [v_v p_v]_{\Lambda_i^-}^{\Lambda_i^+}, \end{aligned} \tag{2.11}$$

were Λ_i^+ and Λ_i^- represent the inflow and the outflow extrema of Λ_i according to the orientation of $\boldsymbol{\lambda}_i$. At this point, we re-organize the local boundary term in order to collect contributions of different branches affecting the same junction point. Let's define the set of indexes of junction points, namely:

$$J := \{ j \in N : \quad s_j \in \Lambda, \quad \#(P_{s_j}) \geq 2 \}$$

where P_{s_j} is the patch of the $j-th$ junction node, i.e. the collection of all branches joining at the node, and $\#$ indicates the counting measure. We also need the following disjoint partition of the indexes in

P_{s_j}. According to the orientation of $\boldsymbol{\lambda}_i$, for any branching points s_j we distinguish branches that are entering the node, whose contribution to mass conservation is positive, from branches leaving the node, whose contribution is negative. The former are branches whose outflow region coincides with the point s_j, while for the latter it is the inflow region:

$$P_j^{out} := \{i \in \{1, ..., N\} : \quad \Lambda_i^+ \equiv \{s_j\}\}, \tag{2.12}$$

$$P_j^{in} := \{i \in \{1, ..., N\} : \quad \Lambda_i^- \equiv \{s_j\}\}, \tag{2.13}$$

for all $j \in J$. At this point we can rewrite the boundary term by isolating the terms relative to inner junction nodes to those relative to outer inflow and outflow nodes, namely:

$$\begin{aligned}\sum_{i=1}^{N} \pi R_i'^2 [v_v p_v]_{\Lambda_i^-}^{\Lambda_i^+} &= [\pi R_i'^2 v_v p_v]_{\Lambda^{in}}^{\Lambda^{out}} + \sum_{j\in J}[\sum_{i\in P_j^{out}} \pi R_i'^2 (p_v v_v)|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 (p_v v_v)|_{\Lambda_i^-}] = \\ &= [\pi R_i'^2 v_v p_v]_{\Lambda^{in}}^{\Lambda^{out}} + \sum_{j\in J} p_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}]\end{aligned} \tag{2.14}$$

where Λ^{out} and Λ^{in} identify the collection of the inflow and outflow tips of the vessel network, i.e. non junction points where the tangent unit vector is inward-pointing and outward-pointing, respectively. This collection contains the boundary points, i.e. extrema belonging to $\partial\Omega$, but the inclusion may be strict. Indeed we have to address the issue of immersed tips, i.e. of network extrema belonging to $\dot{\Omega}$. As for the tissue boundary, we used the apex p and u to identify the subset on which we will enforce pressure and velocity boundary conditions, respectively. For the sake of simplicity, we also implicitly assume that boundary points and bifurcation or branching points can not coincide. Obviously this is not a strong limitation, it only allows us to easily write the equations by avoiding further indexes.
Finally we have the following integration formula:

$$\begin{aligned}\int_\Lambda [\pi R'^2 \partial_s p_v v_v] ds = &- \int_\Lambda [\pi R_i'^2 p_v \partial_s v_v] ds + [\pi R_i'^2 v_v p_v]_{\Lambda^{in}}^{\Lambda^{out}} + \\ &+ \sum_{j\in J} p_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}]\end{aligned} \tag{2.15}$$

After this manipulation, we gain a term that looks like the weak counterpart of the mass conservation constrain (written for a generic test function v_v). It is now explained why we pre-multiplied the vessels problem with the function $\pi R'^2$. In fact, thanks to that trick, the desired constraint come to light in a natural way. The conservation of the local flow reate at vessel junctions can be indeed expressed in term of the above notation as follows:

$$\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} = \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-} \quad \forall j \in J \tag{2.16}$$

Therefore, it is reasonable to weakly enforce it in the variational formulation by multiplying it with the pressure test functions q_v, which act as a Lagrangian multiplier for this constraint, namely:

$$\sum_{j\in J} q_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}] = 0 \tag{2.17}$$

Now we have resolved the problem for the Stokes term, we have to use a similar way to manage the non linear term due to the fact that the curvature in the α^{**} is not regular enough. Again using the

piecewise definition of $R'(s)$, we can split the integral and use the standard Green's formula, obtaining:

$$\int_\Gamma [\pi R'^2 G \partial_s(\alpha^{**} u_v^2) v_v] ds = -\int_\Gamma [\pi R'^2 G(\alpha^{**} u_v^2 \partial_s v_v] ds + \\ + \sum_{i=1}^{N} \pi R_i'^2 G[\alpha^{**} u_v^2 v_v]_{\Lambda_i^-}^{\Lambda_i^+}. \tag{2.18}$$

So the full weak formulation of the vessel problem reads as:

$$\int_\Lambda [\frac{\pi^2 R'^4}{k_v}(1+\kappa'^2 R'^2)\overline{u}_v v_v] ds + \frac{1}{\beta}[\pi^2 R'^4 u_v v_v]_{\varepsilon_u} - \int_\Lambda [\pi R_i'^2 p_v \partial_s v_v] ds + \\ - \int_\Gamma [\pi R'^2 G \alpha^{**} u_v^2 \partial_s v_v] ds + \sum_{i=1}^{N} \pi R_i'^2 G[\alpha^{**} u_v^2 v_v]_{\Lambda_i^-}^{\Lambda_i^+} + \\ + \sum_{j\in J} p_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}] = \\ = -[\pi R'^2 p_0 v_v]_{\varepsilon_u} - [\pi R'^2 g_v v_v]_{\varepsilon_p}, \tag{2.19}$$

$$\int_\Lambda [\partial_s(\overline{u}_v) q_v] ds + \int_\Lambda [\frac{Q}{2\pi R'^2}(p_v - \overline{p_t}) q_v] ds + \\ + \sum_{j\in J} q_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}] = 0 \tag{2.20}$$

2.3 Coupled weak formulation

At this point combining all the equation we obtain the whole weak formulation of our 3D/1D coupled model of fluid exchange between microcirculation and tissue intestitium. The variational formulation of the problem read as:
Find $\mathbf{u}_t \in \mathbf{V}_t$, $p_t \in Q_t$, $u_v \in V_v$, $p_v \in Q_v$ s.t.:

$$\begin{cases} a) & (\frac{1}{k_t}\mathbf{u}_t, \mathbf{v}_t)_{\Omega_t} + \frac{1}{\beta}(\mathbf{u}_t \cdot \mathbf{n}, \mathbf{v}_t \cdot \mathbf{n})_{\Gamma_u} - (p_t, div(\mathbf{v}_t))_{\Omega_t} = -(g_t, \mathbf{v}_t \cdot \mathbf{n}) - (p_0, \mathbf{v}_t \cdot \mathbf{n})_{\Gamma_u} & \forall \mathbf{v}_t \in \mathbf{V}_t \\ b) & (div(\mathbf{u}_t), q_t)_{\Omega_t} - (Q(p_v - \overline{p_t}), q_t)_\Lambda = 0 & \forall q_t \in Q_t \\ c) & (\frac{\pi^2 R'^4}{k_v}(1+\kappa'^2 R'^2) u_v, v_v)_\Lambda + \frac{1}{\beta}[\pi^2 R'^4 u_v v_v]_{\varepsilon_u} - (\pi R'^2 p_v, \partial_s v_v)_\Lambda + \langle v_v, p_v \rangle_J \\ & \qquad -(\pi R'^2 G \alpha^{**} u_v^2, \partial_s v_v)_\Lambda + \sum_{i=1}^{N} \pi R_i'^2 G[\alpha^{**} u_v^2 v_v]_{\Lambda_i^-}^{\Lambda_i^+} = & \forall v_v \in V_v \\ & \qquad -[\pi R'^2 p_0 v_v]_{\varepsilon_u} - [\pi R'^2 g_v v_v]_{\varepsilon_p} & \\ d) & (\pi R'^2 \partial_s u_v, q_v)_\Lambda + (Q(p_v - \overline{p_t}, q_v)_\Lambda - \langle u_v, q_v \rangle_J = 0 & \forall q_v \in Q_v \end{cases} \tag{2.21}$$

Where

$$\langle v_v, p_v \rangle_J := \sum_{j\in J} p_v(s_j)[\sum_{i\in P_j^{out}} \pi R_i'^2 v_v|_{\Lambda_i^+} - \sum_{i\in P_j^{in}} \pi R_i'^2 v_v|_{\Lambda_i^-}]$$

Chapter 3

Discrete Model

3.1 Finite elements set up

The Finite Element Method is adopted to approximately solve [2.21]. We denote by T_t^h an admissible family of partitions of $\overline{\Omega}$ into tetrahedrons K that satisfies the usual conditions of a conforming triangulation of Ω. We use discontinuous piecewise-polynomial finite elements for pressure $\mathbf{H}^{div}$-conforming *Raviart-Thomas* for velocity, namely

$$\mathbb{Y}_h^k := \{w_h \in L^2(\Omega) : \quad w_h|_K \in P_{k-1} \quad \forall K \in T_t^h\}, \tag{3.1}$$

$$\mathbb{RT}_h^k := \{\mathbf{w}_h \in \mathbf{H}(div, \Omega) : \quad \mathbf{w}_h|_K \in P_{k-1}(K, \mathbb{R}^d) \oplus P_{k-1}(K) \quad \forall K \in T^h, t\}, \tag{3.2}$$

for every integer $k \geq 0$, where P_k identifies the standard space of polynomials of degree *leqk* in the variables $\mathbf{x} = (x_1, ..., x_d)$. For the simulations presented later on, the lowest order *Raviart-Thomas* approximation has been adopted, corresponding to $k = 0$ above.
Concerning the capillary network, we split the discrete domain into the segments, $\Lambda_h = \bigcup_{i=1}^N \Lambda_i^h$, where Λ_i^h is a finite element mesh on the one-dimensional manifold Λ_i. The solution of [2.21c] and [2.21d] over the branch Λ_i is approximated using continuous piecewise-polynomial finite element spaces for both pressure and velocity. Since we want velocity tp be discontinuous at multiple junctions, we define the related finite element space over the whole network as the collection of the local spaces of the single branches. We have the following trial spaces for vessel pressure and velocity, respectively:

$$\mathbb{X}_h^{k+1}(\Lambda) := \{w_h \in C^0(\overline{\Lambda}) : \quad w_h|_S \in P_{k+1}(S) \forall S \in \Lambda^h\}, \tag{3.3}$$

$$\mathbb{W}_h^{k+2}(\Lambda) := \bigcup_{i=1}^N \mathbb{X}_h^{k+2}(\Lambda_i), \tag{3.4}$$

for every integer$k \geq 0$. As a result, we use generalized *Taylor-Hood* elements on each network branch, satisfying in this way the local stability of the mixed finite elements pair for the network Λ.

3.2 Discrete Model

Before the derivation of the discrete model we have to take into account the non linear term of the vessel momentum equation. To manage it we are going to use two different iterative methods: a simple Fixed Point Method and a Newton Method. In both cases we need to manage the non linear term of

the equation using a solution computed in a previous step. The convergence of the solution is reached when a certain convergence condition is satisfied. For the implementation it is used an incremental error condition:

$$\frac{||u_v^{n+1} - u_v^n||_{H^1(\Lambda)}}{||u_v^{n+1}||_{H^1(\Lambda)}} + \frac{||p_v^{n+1} - p_v^n||_{L^2(\Lambda)}}{||p_v^{n+1}||_{L^2(\Lambda)}} \leq MaxError \tag{3.5}$$

3.2.1 Fixed Point Method discretization

We can now derive a discrete formulation of our problem by projecting the continuous infinite-dimensional PDE's over the discrete spaces defined above. Doing that we need to take into account the presence of the non linear term in the vessel momentum equation.
Here, for the resolution we are going to use a fixed point iterative method where the non linearity is linearized as follows:

$$G\partial_S(\alpha^{**}{u_v^{n+1}}^2) \sim G\partial_S(\alpha^{**}u_v^{n+1}u_v^n) \tag{3.6}$$

where u_v^{n+1} is the flow in the vessel to compute and u_v^n is the solution computed in the previous step.
Then, by writing the discrete unknowns as linear combinations of the finite element base functions, we can deduce the following linear system:

$$\begin{bmatrix} \mathbb{M}_{tt} & -\mathbb{D}_{tt}^T & \mathbb{O} & \mathbb{O} \\ \mathbb{D}_{tt} & \mathbb{B}_{tt} & \mathbb{O} & -\mathbb{B}_{tv} \\ \mathbb{O} & \mathbb{O} & \mathbb{M}_{vv} + \mathbb{NL}_{vv}(\mathbf{U}_v^{old}) & -\mathbb{D}_{vv}^T - \mathbb{J}_{vv}^T \\ \mathbb{O} & -\mathbb{B}_{vt} & \mathbb{D}_{vv} + \mathbb{J}_{vv} & \mathbb{B}_{vv} \end{bmatrix} \begin{bmatrix} \mathbf{U}_t \\ \mathbf{P}_t \\ \mathbf{U}_v \\ \mathbf{P}_v \end{bmatrix} = \begin{bmatrix} \mathbf{F}_t \\ \mathbf{0} \\ \mathbf{F}_v \\ \mathbf{0} \end{bmatrix} \tag{3.7}$$

Submatrices and subvectors are defined as follows:

$$\begin{aligned}
[\mathbb{M}_{tt}]_{ij} &:= (\frac{1}{k_t}\boldsymbol{\varphi}_t^j, \boldsymbol{\varphi}_t^i)_\Omega + \frac{1}{\beta}(\boldsymbol{\varphi}_t^j \cdot \mathbf{n}, \boldsymbol{\varphi}_t^i \cdot \mathbf{n})_{\Gamma_u} && \mathbb{M}_{tt} \in \mathbb{R}^{N_t^h \times N_t^h}, \\
[\mathbb{D}_{tt}]_{ij} &:= (\nabla \cdot \boldsymbol{\varphi}_t^j, \psi_t^i)_\Omega && \mathbb{D}_{tt} \in \mathbb{R}^{N_t^h \times M_t^h}, \\
[\mathbb{B}_{tt}]_{ij} &:= (Q\overline{\psi}_t^j \delta_{\Lambda_h}, \psi_t^i)_\Omega && \mathbb{B}_{tt} \in \mathbb{R}^{M_t^h \times M_t^h}, \\
[\mathbb{B}_{tv}]_{ij} &:= (Q\psi_v^j \delta_{\Lambda_h}, \psi_t^i)_\Omega && \mathbb{B}_{tv} \in \mathbb{R}^{M_t^h \times M_v^h}, \\
[\mathbb{B}_{vt}]_{ij} &:= (Q\overline{\psi}_t^j, \psi_v^i)_\Lambda && \mathbb{B}_{vt} \in \mathbb{R}^{M_v^h \times M_t^h}, \\
[\mathbb{B}_{vv}]_{ij} &:= (Q\psi_v^j, \psi_v^i)_\Lambda && \mathbb{B}_{vv} \in \mathbb{R}^{M_v^h \times M_v^h}, \\
[\mathbb{M}_{vv}]_{ij} &:= (\frac{\pi^2 R'^4}{k_v}(1 + \kappa'^2 R'^2)\varphi_v^j, \varphi_v^i) + \frac{1}{\beta}[\pi^2 R'^4 \varphi_v^j \varphi_v^i]_{\varepsilon_u} && \mathbb{M}_{vv} \in \mathbb{R}^{N_v^h \times N_v^h}, \\
[\mathbb{NL}_{vv}(\mathbf{U}_v^{old})]_{ij} &:= \sum_{k=1}^{N_v^h} [\mathbf{U}_v^{old}]^k \{-(\pi R'^2 G\alpha^{**}\varphi_v^j\varphi_v^k, \partial_s\varphi_v^i)_\Lambda + \sum_{i=1}^{N} \pi R_i'^2 G[\alpha^{**}\varphi_v^k\varphi_v^j\varphi_v^i]_{\Lambda_i^-}^{\Lambda_i^+}\} && \mathbb{NL}_{vv} \in \mathbb{R}^{N_v^h \times N_v^h}, \\
[\mathbb{D}_{vv}]_{ij} &:= (\pi R'^2 \partial\varphi_v^j, \psi_v^i)_\Lambda && \mathbb{D}_{vv} \in \mathbb{R}^{N_v^h \times M_v^h}, \\
[\mathbb{J}_{vv}]_{ij} &:= \langle [\varphi_v^j], \psi_v^i \rangle_J && \mathbb{J}_{vv} \in \mathbb{R}^{N_v^h \times M_v^h}, \\
[\mathbf{F}_t]_i &:= -(g_t^h, \boldsymbol{\varphi}_t^i \cdot \mathbf{n})_{\Gamma_p} - (p_0^h, \boldsymbol{\varphi}_t^i \cdot \mathbf{n})_{\Gamma_u} && \mathbf{F}_t \in \mathbb{R}^{N_t^h}, \\
[\mathbf{F}_v]_i &:= -[\pi R'^2 g_v \varphi_v^i]_{\varepsilon_p} - [\pi R'^2 p_0 \varphi_v^i]_{\varepsilon_u} && \mathbf{F}_v \in \mathbb{R}^{N_v^h},
\end{aligned}$$

where $\overline{\psi}_t^j$ is the average of ψ_t^j and $\mathbf{U}_v^{old}$ is the discrete vector of the velocity computed in the previous iteration.
Now for the implementation of exchange matrices, namely $\mathbb{B}_{tt}$,$\mathbb{B}_{vt}$,$\mathbb{B}_{tv}$ and $\mathbb{B}_{vv}$, we define two discrete

operators: the first one is the mean value of a generic basis of Q_t^h, while the second is an interpolation between Q_v^h and Q_t^h. For each node $s_k \in \Lambda^h$ we define $T_\gamma(s_k)$ as the discretization of the perimeter of the vessel $\gamma(s_k)$. For simplicity, we assume that $\gamma(s_k)$ is a circle of radius R defined on the orthogonal plane to Λ^h at this point s_k. The set of points of $T_\gamma(s_k)$ is used to interpolate the basis functions ψ_t^i. Let us introduce the interpolation matrix $\prod_\gamma(s_k)$ which returns the values of each test functions ψ_t^i on the set of points belonging to $T_\gamma(s_k)$. Then, we consider the average operator $\overline{\pi}_v t : Q_t^h \to Q_v^h$ such that each row is defined as,

$$\overline{\prod}_{vt}|_k = \mathbf{w}^T(s_k) \prod_\gamma(s_k) \quad k = 1, ..., M_v^h, \tag{3.8}$$

where $\mathbf{w}$ are the weights of the quadrature formula used to approximate the integral

$$\overline{q}_t(s) = \frac{1}{2\pi R} \int_0^{2\pi} q_t(s, \theta) R d\theta,$$

on the nodes belonging to $T_\gamma(s_k)$. The discrete interpolation operator$\pi_{tv} : Q_v^h \to Q_t^h$ returns the value of each basis function belonging to Q_t^h in the corresponding nodes of Q_v^h. In algebraic form it is expressed as an interpolation matrix $\prod_{tv} \in \mathbb{R}^{M_v^h \times M_t^h}$. Using these tools we obtain:

$$\mathbb{B}_{tt} = \prod_{vt}^{T} \mathbb{M}_{vv}^P \overline{\prod}_{vt},$$

$$\mathbb{B}_{tv} = \prod_{vt}^{T} \mathbb{M}_{vv}^P,$$

$$\mathbb{B}_{vt} = \mathbb{M}_{vv}^P \overline{\prod}_{vt},$$

$$\mathbb{B}_{tt} = \mathbb{M}_{vv}^P$$

being $\mathbb{M}_{vv}^P$ the pressure mass matrix for the vessel problem defined by

$$[\mathbb{M}_{vv}^P]_{ij} := (Q\psi_v^j, \psi_v^i)_\Lambda.$$

Concerning the implementation of junction compatibility conditions, we introduce a linear operator giving restriction with sign of a basis function of V_v^h over a given junction node. For a given $k \in J$, we define $R_k : V_v^h \to \mathbb{R}$ such that

$$R_k(\varphi_v^j) := \begin{cases} +\pi R_l'^2 \varphi_v^j(s_k) & j \in \Lambda_l^h \wedge l \in P_k^{out} \\ -\pi R_l'^2 \varphi_v^j(s_k) & j \in \Lambda_l^h \wedge l \in P_k^{out} \end{cases}, \tag{3.9}$$

for all $j = 1, ...N_v^h$, where the expression "$j \in \Lambda_l^h$" means the the $j-th$ dof is linked to some vertices of the $l-th$ branch. Note that we are implicitly using the usual property of Lagrangian finite element basis functions, i.e. that they vanish on all nodes except the related one. As a consequence, our definition is consistent for all junction vertices. Indeed, R_k may only assumes values $\{-\pi R_l'^2, 0, +\pi R_l'^2\}$ for some l and in particular $R_k(\varphi_v^j) = 0$ for all couples of indexes (k, j) that are uncorrelated. Furthermore, the definition of R_k can be trivially extended to all network vertices. Using this operator, the generic (i, j) element of $\mathbb{J}_{vv}$ may be computed as follows:

$$[\mathbb{J}_{vv}]_{ij} = -\sum_{k \in J} R_k(\varphi_v^j) \psi_v^i(s_k).$$

3.2.2 Newton Method discretization

Now we want to do the same thing using the Newton method for the linearization of the advective term of the momentum equation in the vessel. As for the Newton linearization of the General Navier-Stokes equation, our one reads as:

$$G\partial_s(\alpha^{**}u_v^{n+1^2}) \sim 2G\partial_s(\alpha^{**}u_v^{n+1}u_v^n) - G\partial_s(\alpha^{**}u_v^n u_v^n), \tag{3.10}$$

thus the full discrete system is written as:

$$\begin{bmatrix} \mathbb{M}_{tt} & -\mathbb{D}_{tt}^T & \mathbb{O} & \mathbb{O} \\ \mathbb{D}_{tt} & \mathbb{B}_{tt} & \mathbb{O} & -\mathbb{B}_{tv} \\ \mathbb{O} & \mathbb{O} & \mathbb{M}_{vv} + 2\mathbb{NL}_{vv}(\mathbf{U}_v^{old}) & -\mathbb{D}_{vv}^T - \mathbb{J}_{vv}^T \\ \mathbb{O} & -\mathbb{B}_{vt} & \mathbb{D}_{vv} + \mathbb{J}_{vv} & \mathbb{B}_{vv} \end{bmatrix} \begin{bmatrix} \mathbf{U}_t \\ \mathbf{P}_t \\ \mathbf{U}_v \\ \mathbf{P}_v \end{bmatrix} = \begin{bmatrix} \mathbf{F}_t \\ \mathbf{0} \\ \mathbf{F}_v + \mathbf{NLF}_v(\mathbf{U}_v^{old}) \\ \mathbf{0} \end{bmatrix} \tag{3.11}$$

where all sub-matrices are the same of the Fixed Point Method one, and

$$[\mathbf{NLF}_v(\mathbf{U}_v^{old})]_i := \sum_{j=1}^{N_v^h}\sum_{k=1}^{N_v^h} \mathbf{U}_v^{old^j}\mathbf{U}_v^{old^k}\{-(\pi R'^2 G\alpha^{**}\varphi_v^j\varphi_v^k, \partial_s\varphi_v^i)_\Lambda + \sum_{i=1}^{N}\pi R_i'^2 G[\alpha^{**}\varphi_v^k\varphi_v^j\varphi_v^i]_{\Lambda_i^-}^{\Lambda_i^+}\} \quad \mathbf{NLF}_v \in \mathbb{R}^{N_v^h}$$

Chapter 4

Model Validation

As for the validation of the model, we are going to compare it with the *"Coupled 3D/1D Problem"* with imposition of Pouiseuille flow implemented by Domenico Notaro.

4.1 Equations comparison

First of all we are going to compare the equations of both the models, which will look quiet similar

$$\begin{cases} POISEUILLE\,FLOW & & \\ \frac{1}{k_t}\mathbf{u}_t + \nabla p_t = 0 & on & \Omega_t \\ div(\mathbf{u}_t) - Q(p_v - \overline{p_t})\delta_\Gamma = 0 & on & \Omega_t \\ \frac{\pi R'^2}{k_v} u_v + \partial_s p_v = 0 & on & \Lambda \\ \partial_s u_v + \frac{Q}{\pi R'^2}(p_v - \overline{p_t}) = 0 & on & \Lambda \end{cases} \qquad \begin{cases} NAVIER - STOKES\,FLOW & & \\ \frac{1}{k_t}\mathbf{u}_t + \nabla p_t = 0 & on & \Omega_t \\ div(\mathbf{u}_t) - Q(p_v - \overline{p_t})\delta_\Gamma = 0 & on & \Omega_t \\ \frac{\pi R'^2}{k_v} u_v(1 + k'^2 R'^2) + \partial_s p_v + G\partial_s(\alpha^{**} u_z^2) = 0 & on & \Lambda \\ \partial_s u_v + \frac{Q}{\pi R'^2}(p_v - \overline{p_t}) = 0 & on & \Lambda \end{cases}$$

As we can see the differences of the two models are in the momentum equation in the vessel: the presence of a corrective coefficient in the viscous term, which take in account the effect of the curved geometry, and the non linear advective term, typical of Navier-Stokes Problem. Moreover if we have to solve the flow problem in simple case with rectilinear vessel and with null pressure gain ($G = 0$), the equations are right the same.
Moreover, if we look to the corrective coefficient of the viscous term, taken together with the conductivity coefficient,

$$k_{eff} = \frac{k_v}{1 + \kappa'^2 R'^2} \leq k_v \tag{4.1}$$

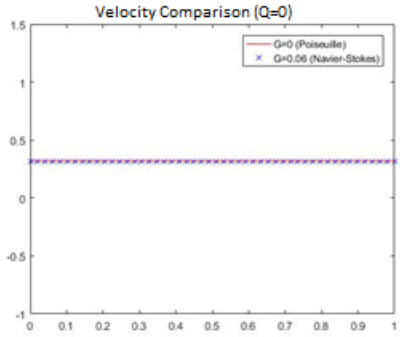
it has the physical meaning of an effective permeability of the vessel, which diminish in presence of high curvature effect ($\kappa' R' \sim 1$).

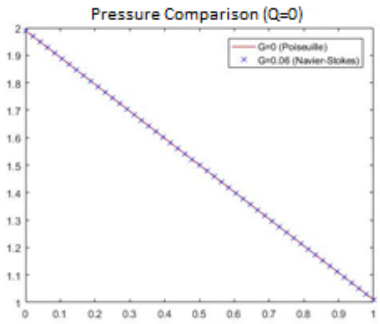
4.2 Numerical Comparison

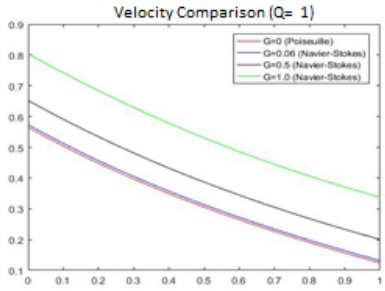
In order to do a numerical comparison we are going to compare how the solutions of the two solvers are different for network with different properties. First of all, we are going to show how the non-linearity of our model will affect the result, and after that we will show the effect of curved geometry through three different test case: Singlebranch, Bifurcation, Rhombus..

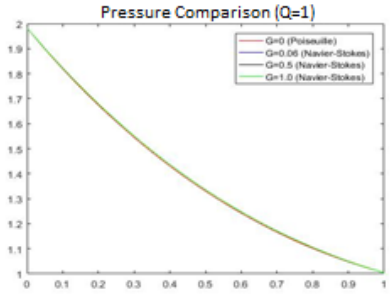
4.2.1 Advective effect

To highlight the effect of advective term, we are going to test the solvers over a rectilinear SingleBranch in order to avoid any geometric effect. For the resolution of the non-linear problem, in the code are implemented both the iterative methods we talk before, the fixed point and the newton method. Both this methods are always able to reach the same solution and more over the Newton Method is able to solve the problem in far less iteration, so for our simulations we are going to use only this one. As







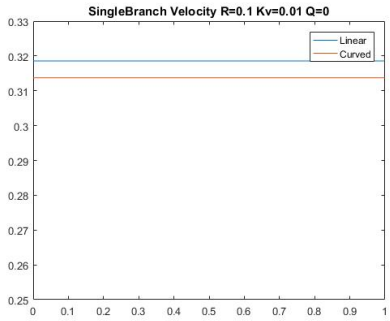


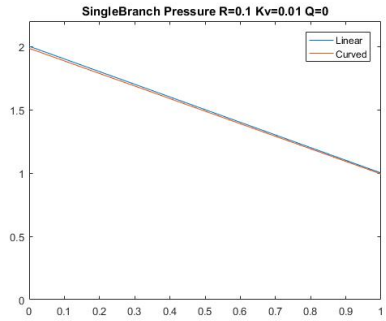
we can see, if we solve both problems with dirichlet boundary conditions, with no exchange between the vessel and the porous medium ($Q = 0$), there is no difference between the two solutions. Instead, if we solve it with non zero wall conductivity ($Q = 1$), we have an increasing difference of the velocity solution as we increase the Dynamic Pressure Gain ($G > 0$). In fact for a fixed pressure difference (due to the dirichlet boundary conditions) the transport term make the increase the velocity in channel.

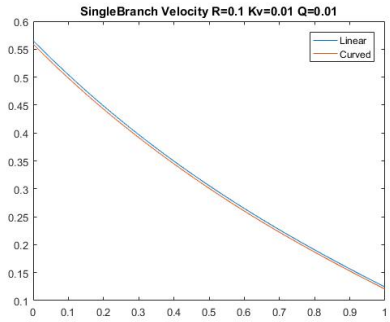
4.2.2 SingleBranch

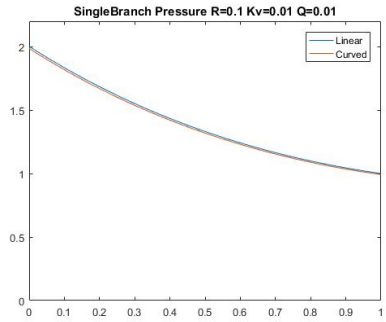
n the second test case, we wanted to emphasize the how the geometry of the problem can affect the solution. So, we decided to test for the Pouiseuille code on a rectilinear vessel, and the Navier-Stokes on a sinusoidal one. Both the geometries are tested with the same parameters and boundary conditions and on the sinusoidal branch we imposed zero Dynamic Pressure Gain, to avoid transport effects.

In the this case we have a diminishing of velocity for the sinusoidal geometry for all choose of wall conductivity ($Q = 0$ or $Q = 0.01$). This effect is due to the fact that the mean effective conductivity for

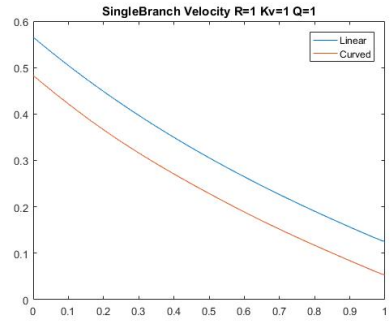


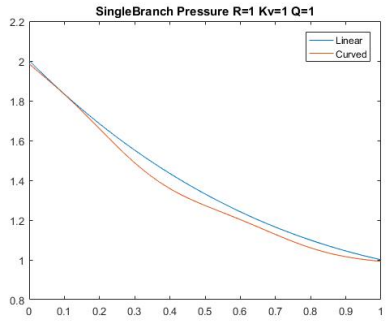


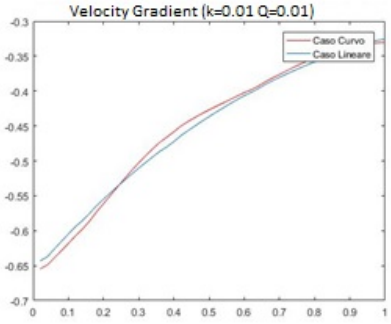


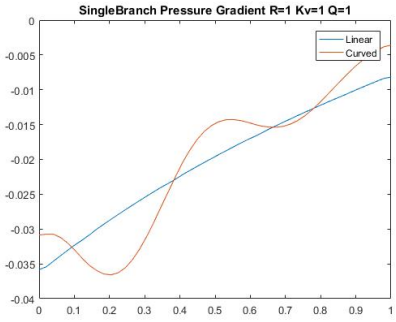


the rectilinear branch is higher than the curved one, and so for fixed pressure gradient (as we can see it's the same for both test) it allows an higher velocity. But for our simulation we choose the problem with a small effect due to the curvature ($\kappa R \ll 1$) , so in the next example we tried to point it out ($\kappa R \sim 1$).



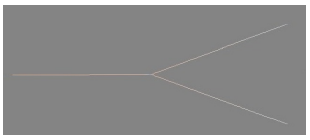


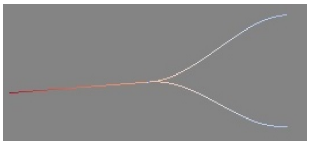


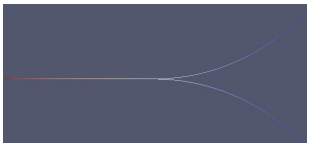


From this plot we have an overview on how the curvature corrective coefficient really affect the solution. Here not only the velocity, but also the pressure has been influenced by the geometry. So in this simulation, we highlighted that the flow velocity decrease due to curvature effect and the pressure gradient follow the curvature trend.

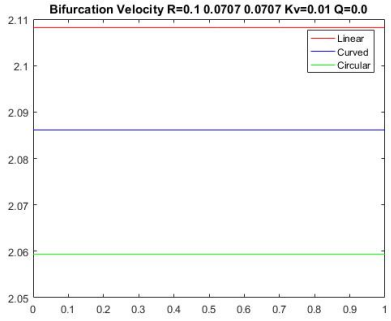
4.2.3 Bifurcartion

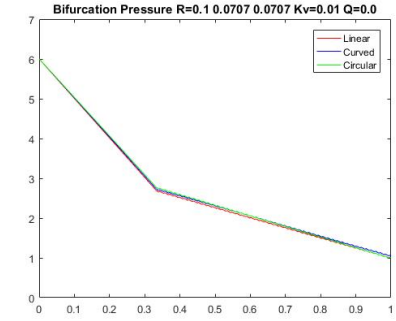


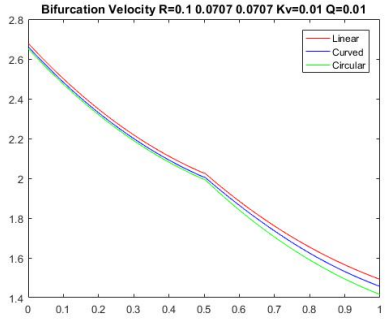


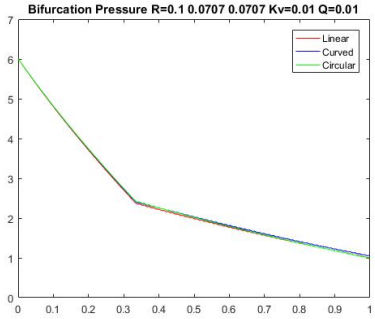


The third test case is done on curved bifurcation with variable radius ($R = \{0.1, 0.075, 0.075\}$) for the Navier-Stokes model, and on a rectilinear bifurcation for the Pouiseuille one. In both simulations, we imposed for every branch the same conditions and the same parameters and for the Navier-Stokes case we impose a small effect of curvature ($\kappa R \ll 1$). For curved test case were chosen two different profile, one sinusoidal and one circular.









Through these plots we have strong disagreement between the solutions also for a low curvature effect. First of all, from the case with null wall permeability ($Q = 0$),we have that the velocity decreases in the network with the increasing of the curvature effect. That is due, as for the SingleBranch, to geometric effect on the conductivity of the channel, which decrease in presence of curvature. To highlight this point we compare the sinusoidal and the circular test and we have lower velocity in the second one, because it has a constant and higher curvature, with respect to the first one, on the branches after the bifurcation. For the non zero wall permeability cases, we have almost the same result but, for presence of the curved geometry, the velocity decrease faster when we have a higher geometric effect.
For this kind of tests case, it is possible to find different velocity profiles before and after the junction, so the solution could have the presence of jump of velocity.
Let's derive how much this change of profile could affect our solutions. Due to the continuity condition in junctions, we have:

$$0 = \sum_{i \in J_{in}} \int_{A_i} u_z^i dA - \sum_{j \in J_{out}} \int_{A_j} u_z^j dA = \sum_{i \in J_{in}} \overline{u}_z^i \int_{A_i} p_i(r,\theta) dA - \sum_{j \in J_{out}} \overline{u}_z^j \int_{A_j} p_j(r,\theta) dA \quad (4.2)$$

where p_i is the velocity profile over the branch i. Now using the full velocity profile formula we have:

$$\int_{A_i} [p_i(r,\theta)] dA = \int_{A_i} [s(r)(1 + \kappa_i^2 r^2 - 2\kappa_i r \cos\theta)] dA = \pi R_i^2 + \pi \kappa_i R_i^4 \sigma \quad (4.3)$$

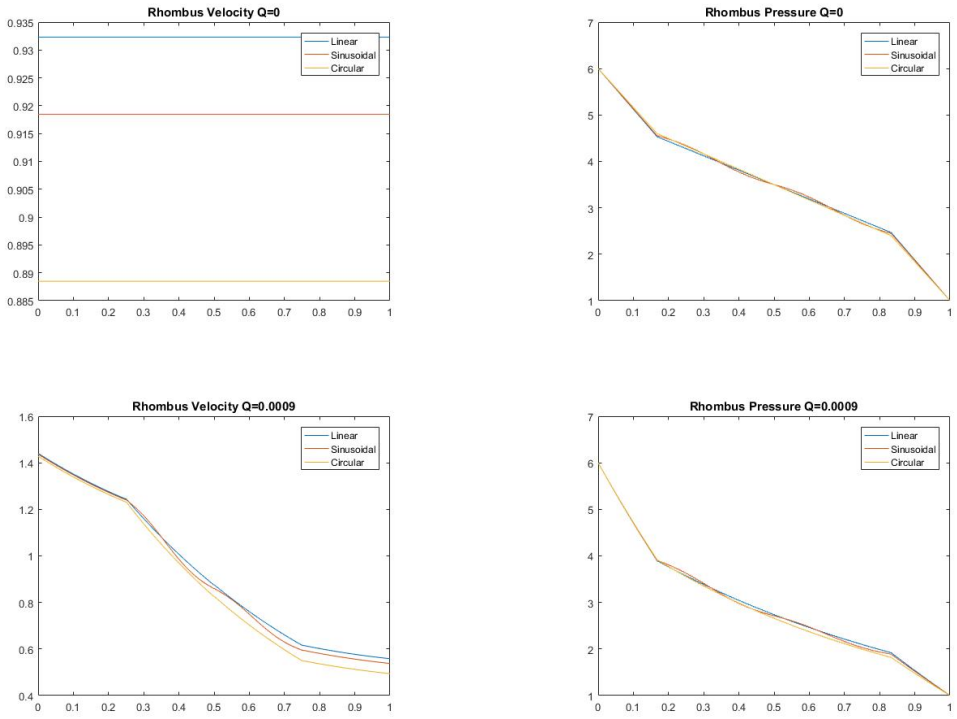
with $\sigma = \frac{\gamma+2}{4(\gamma+2)}$ and γ is the order of velocity profile.
Due to this we have:

$$0 = \sum_{i \in J_{in}} \int_{A_i} u_z^i dA - \sum_{j \in J_{out}} \int_{A_j} u_z^j dA = (\sum_{i \in J_{in}} \overline{u}_z^i \pi R_i^2 - \sum_{j \in J_{out}} \overline{u}_z^j \pi R_j^2) + \sigma(\sum_{i \in J_{in}} \overline{u}_z^i \pi R_i^4 \kappa_i^2 - \sum_{j \in J_{out}} \overline{u}_z^j \pi R_j^4 \kappa_j^2) \quad (4.4)$$

As we can see from this formula, with respect to the linear case we have an additive term which take care of this change of velocity profile and that could create jumps of velocity. But for low curvature effect, this term looks negligible with respect the linear one, as we can see from the circular case.

4.2.4 Rhombus

r the last test case we decided to test the two solvers for a rhomboidal geometry, with a variable radius ($R = \{0.05, 0.035, 0.035, 0.035, 0.05\}$), zero Dynamic Pressure Gain ($G = 0$) and low curvature effect ($\kappa R \ll 1$).



As for the previous test, we have that higher mean curvature implies less conductivity of channel. So for the curved networks, we have lower velocity, in the cases with zero wall permeability, and faster velocity decreasing in the branch with higher curvature, for the case with non zero wall permeability.

4.2.5 Conclusion

Thanks to our tests, we can conclude that for a rectilinear problem with very low Dynamic Pressure Gain, both the solvers reach almost the same solution. Higher Dynamic Pressure Gain has the effect to increase the velocity in the vessel for a constant pressure gradient. In the end, in presence of curved geometries we have a general reduction of the conductivity of the vessel, which can affect all the network in different ways. In the Siglebranch, there is a reduction of the flow velocity due to the lower conductivity. For a more complex geometry, we have a general reduction of the flow velocity in the network, due to the reduction of the mean conductivity, and for diffusing vessel, we have a faster decreasing of velocity in the curved branches with respect to the rectilinear ones.

Chapter 5

C++ code

The latest stable release of the code is available at the following link:

$$https://github.com/lformaggia/model3D1D$$

Let us briefly review the main achievements. The code is divided into three main section: MixedFormulation, CurvedFormulation and GraphGenerator. The first one is the code developed by Domenico Notaro, which contain the solver for simple linear geometry based on Pouiseuille Flow assumption, using getFem++ library. The second one is code for the resolution of microcirculation problems for a more generic network geometry, that we modeled in the previous sections. The last one is code which use the library BGLgeom of M. Tantardini and I. Speranza in order to produce a simple tool for generate a the network.

5.1 CurvedFormulation

For the microcirculation part, we decide to use the same structure implemented by D. Notaro in his code and to do real modification only on those structures which doesn't adapt to the generic geometry, and for that we make great use of the library *libproblem3d1d.a*.
The first real modification is on the declaration of the main structure representing the 3D/1D problem, in an appropriate header(*c_problem3d1d.hpp*), and his definition in a source file (*c_problem3d1d.cpp*). Thanks to the similarities with the Poiseuille model, for this structure is used a class derived from the class *problem3d1d* implemented in the MixedFormulation.
As for the MixedFormulation we keep inside the main class all the attributes who identify an instance of the problem, such as the meshes, finite elements methods, boundary condition and so on. On the other hand, external attributes such as the user-defined descriptors of the algorithm (*c_desc3d1d.hpp*) , the physical parameters (*c_param3d1d.hpp*) have been moved to proper header files. Furthermore, following the two main design principle of OOP, namely *encapsulation* and *information hiding*, we define a simple interface between the computational model and the user: the whole internal structure has **private** scope while public accessors are eventually provided to inspect the attributes.

5.1.1 c_problem3d1d

```
//!  Main class defining the coupled 3D/1D fluid curved problem

  class c_problem3d1d:  public problem3d1d {
//!  Initialize the problem
/*!
     1.Read the .param filename from standard input
     2.Import problem descriptors (file path, GetFEM types,...)
     3.Import mesh for tissue (3D) and vessel network (1D)
     4.Set finite elements and integration methods
     5.Build problem parameters
     6.Build the list of tissue boundary data
     7.Build the list of vessel boundary (and junction) data
*/
init(int argc, char *argv[]);
//!  Assemble the problem
/*!
     1.  Initialize problem matrices and vectors
     2.  Build the monolithic matrix AM
     3.  Build the monolithic rhs FM
*/
void assembly(void);
//!  Solve the problem
/*!
     Solve iteratively the monolithic system at each iteration until stop con-
ditions are reached
*/
bool solve(void);
/*!
     Merge arterial and venous networks.
     Solve iteratively the monolithic system
*/
friend bool merge_and_solve(c_problem3d1d & Pa, c_problem3d1d & Pv);
//!  Export results into vtk files
/*!
     Export solutions U_t,P_t,U_v,P_v from the monolithic array UM
*/
void export_vtk(const string & suff= "");
//!  Compute mean tissue pressure
inline scalar_type mean_pt(void) {
     return asm_mean(mf_pt, mimt,
          gmm::sub_vector(UM,gmm::sub_interval(dof.Ut(),dof.Pt())));
}
//!  Compute mean vessel pressure
inline scalar_type mean_pv(void){
     return asm_mean(mf_pv, mimv,
          gmm::sub_vector(UM,gmm::sub_interval(dof.Ut()+dof.Pt()+dof.Uv(),dof.Pt())));
}
//!  Compute mean vessel velocity
```

```
inline scalar_type mean_uv(void){
     size_type dofi=0;
     size_type shift=dof.Ut()+dof.Pt();
     scalar_type mean=0.0;
     for(size_type branch=0; branch<mf_Uvi.size(); branch++){
          shift+=dofi;
          dofi=mf_Uvi[branch].nb_dof();
          mean+=asm_mean(mf_Uvi[branch],mimv,gmm::sub_vector(UM,
               gmm::sub_interval(shift,dofi)));
     }
     return mean/mf_Uvi.size();
}
//!Compute the dynamic pressure gain
inline scalar_type G(void){
     return c_param.G(0);
}
//!  Compute total flow rate (network to tissue) - pressure
inline scalar_type flow_rate(void){ return TFR; }

protected:

//!  Algorithm description strings of curved model
c_desc3d1d c_descr;
//!  Physical parameters (dimensionless) for curved model
c_param3d1d c_param;
//!  Velocity in the vessel at a previous iteration
vector_type Uv_old;
//!  Non linear matrix for transport term
sparse_matrix_type NLMvv;
//!  Non linear matrix for Junction transport effect
sparse_matrix_type NLJvv;
//!  Full non linear matrix sparse_matrix_type NL;
//!  Non linear RHS effect (used only in Newton Method)
vector_type NLF;
//!  Monolithic matrix with both linear and non linear term
sparse_matrix CM;
//!  Monolithic RHS with both linear and non linear term
vector_type CFM;

//!  Import algorithm specifications
void import_data(void);
//!  Import mesh for tissue (3D) and vessel (1D)
void build_mesh(void);
//!  Set finite elements methods and integration methods
void set_im_and_fem(void);
//!  Build problem parameters
void build_param(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(void);
//!Build the list of vessel boundary (and junctions) data
void build_vessel_boundary(void);
//!  Build the monolithic matrix AM by blocks
void assembly_mat(void);
//!  Build the monolithic rhs FM by blocks
void assembly_rhs(void);
//!  Assemble RHS source term for stand-alone tissue problem
void assembly_tissue_test_rhs(void);
//!  Build non linear matrix NL
assembly_nonlineat_mat(void);
//!  Solve the monolithic system for a given iteration
//!  of the fixed point method
bool solve_pass(size_pass iter=0);
}; /*end of class c_problem3d1d
```

Moreover many other elements (matrix and parameters) are directly imported by the "*problem*3*d*1*d*" class.
This approach allows to build the most generic tool that is at the same time the easiest possible for incoming "non developing" users and, at the same time, to safeguard the developer from the risks of altering the implementation. In the end, the only methods that remain **public** define exactly what one needs to do in a main file, that is to (I) declare a new instance of the problem, (II) set problem's parameters and initialize the problem, (III) assemble the linear system, (IV) solve it and finally (V) save results for post-processing.

```
   // Declare a new problem
getfem::c_problem3d1d p;
// Initialize the problem
p.init(argc,argv);
// Build the monolithic system
p.assembly();
// Solve the problem
if(!p.solve()) GMM_ASSERT1(false,"solve procedure has failed");
// Save results in .vtk format
p.export_vtk();
// Display some global results:  mean pressures, total flow rate
std::cout<<" Pt average = " << p.mean_pt()<< std::endl;
std::cout<<" Pv average =" <<p.mean_pv()<<std::endl;
std::cout<<" Uv average =" <<p.mean_uv()<<std::endl;
std::cout<<" Network-to-Tissue TFR =" <<p-flow_rate()<<std::endl;
...
```

After these manipulations the package has the following hierarchical structure:

`doc/`: Code documentation (to be generate)
`include/`: General header files
`lib/`: Main library (to be generate)
`src/`: Example sources
- `1_LinearSingleBranch/` Solve the linear SingleBranch
- `2_CurvedSingleBranch/` Solve the curved SingleBranch
- `3_LinearBifurcation/` Solve the linear Bifurcation
- `4_CurvedBifurcation/` Solve the curved Bifurcation
- `5_LinearRombus/` Solve the linear Rhombus
- `6_CurvedRombus/` Solve the curved Rhombus

`Doxyfile/`: Instruction to build the code documentation
`Makefile/`: Instruction to install the whole project

It can be seen that the provided examples follows exactly the benchmarks provided in Chapter 4. The identical structure has been preserved in almost all subdirectories, namely:

`vtk/`: Output in vtk format
`input.param/`: List of user-defined parameters
`main.cpp/`: Main program
`Makefile/`: Instruction to install the example
`network.pts/`: File of points of the vessel network
`network_geometry.pts/`: File of geometric parameters

5.1.2 Mesh and Parameters

Particular attentions is necessary when we read the input file for the mesh1d and the parameters. In fact the difference from the linear problem (MixedFormulation) we need to recive as input not only the point of mesh but also the value of the curved parameters in that, and subsequently we need to interpolate that parameter value on the proper mesh_fem. The curvature value is given in each node of the mesh, so for it we used $P(1,1)$ (polynomial interpolation 1d functions of degree 1) finite elements. Instead the tangent versor is computed as constant over each element of the mesh, so for it we used $P(0,1)$ finite elements. For that in the *c_mesh1d.hpp* header are implemented the function able to manage on it.

```
/*!  Create the edge sequence and build the related 1D mesh.
The input stream ist is used to read a file with the following format (gambit-
like):

     BEGIN_LIST
     BEGIN_ARC
     BC KEYWA [VALUES]
     BC KEYWB [VALUES]
     106 0.4421 -1.6311 2.5101 start
     107 0.4421 -1.6311 7.5101 end
     108 0.3546 -1.6524 2.5539 point
     109 0.2621 -1.6695 2.5880 point
     ...
     END_ARC
     ...
     BEGIN_ARC
     ...
```

```
    END_ARC
    ...
    END_LIST
```

1. The list of points of the IS ordered as follows:
   - first we have the coordinates of TWO ENDS (A,B) (i.e. A=start and B=end)
   - then we have all the remaining nodes of the arc, from A to B

2. If a node is shared by more arcs, the arcs will be CONNECTED in the resulting 1D mesh.

3. BC KEYWA [VALUES] and BC KEYWB [VALUES] are keywords/values related to boundary conditions
to be imposed at nodes A, B. Each KEYW [VALUES] entry can be one of the following:
   - BC DIR 1.1
   - BC MIX
   - BC INT

Correspondingly, each node will be marked with the associated boundary condition, that are:
   - Dirichlet node (pressure = 1.1)
   - Mixed node (flux = coef*(pressure - p0))
   - Internal node

At this stage, this is only meant to assign such BC labels to each node.
If one end is INTERNAL, the corresponding BC will be ignored
(for clarity, please write the INT keyword).

The input stream istc is used to read a file with the following format (gambit-like):

```
    BEGIN_LIST
    BEGIN_ARC
    BC KEYWA [VALUES]
    BC KEYWB [VALUES]
    106 0.4421 -1.6311 2.5101 0.1221 0.3454 0.2943 0.7561 start
    107 0.4421 -1.6311 7.5101 0.1221 0.3454 0.2943 0.7561 end
    108 0.3546 -1.6524 2.5539 0.2235 0.2541 0.3012 0.7214 point
    109 0.2621 -1.6695 2.5880 0.2173 0.2497 0.2886 0.7493 point
    ...
    END_ARC
    ...
    BEGIN_ARC
    ...
    END_ARC
    ...
    END_LIST
```

1. The list of points of the IS ordered as follows:
   - first we have the parameters for the TWO ENDS coordinate (A,B) (i.e. A=start and B=end)
   - then we have all the remaining parameters for the nodes of the arc, from

```
A to B
2.  If a node is shared by more arcs, the arcs will be CONNECTED in the re-
sulting 1D mesh.
3.  BC KEYWA [VALUES] and BC KEYWB [VALUES] are keywords/values related to bound-
ary conditions
to be imposed at nodes A, B. For this kind of problem is not required any bound-
ary condition
fot the curved parameters, because all the BC are imposed with the mesh file
input.  If the
reader would find any BC, it will simply avoid to read it.
*/

/*!
     Import the network points and the curved paremeters from the file pts and
build the mesh.
*/
//!  It also build vessel mesh regions (#=0 for branch 0, #=1 for branch 1,
...).
template<typename VEC_SIZE, typename PARAM>
void import_pts_file(
     std::istream & ist,
     std::istream & istc,
     getfem::mesh & mh1D,
     std::vector<getfem::node> & BCList,
     VEC_SIZE & Nn,
     const std::string & MESH_TYPE,
     PARAM & param
     )

/*!
     Reassemble the curved parameters on the data finite elements.
     The parameters are given in the coordinates of the mesh, so it is
     assumed that the parameters are imported using polynomial of degree 1
     for curvature and polynomial of degree 0 for tangent versor.
*/
template<typename VEC, typename VEC_FEM> void rasm_curve_parameter(
     VEC_FEM & mf_Coefi,
     VEC & Curv,
     VEC & lx,
     VEC & ly,
     VEC & lz
     )

   {
     VEC Curv_temp=Curv;
     VEC lx_temp=lx;
     VEC ly_temp=ly;
     VEC lz_temp=lz;
     size_type Nb=Curv_temp.size(); //Number of branch
     size_type Ni=0; //Number of dof of the coefficient mesh at branch i
```

```
//Fem descriptor for curvature
pfem fd_curv = fem_descriptor("FEM_PK(1,1)");
//Fem descriptor for tangent versor
pfem fd_tgv = fem_descriptor("FEM_PK(1,0)");
scalar_type ct;
for(size_type b=0; b<Nb;++b){
     Ni=Curv_temp[b].size();

     //Reodering the value of the parameters
     ct=Curv[b][1];
     //Shifting all elements
     for(size_type i=2; i<Curv_temp[b].size(); i++){
          Curv_temp[b][i-1]=Curv[b][i];
     }
     Curv_temp[b].back()=ct;

     //Adapting parameters to tbe finite element interpolation
     Ni=mf_Coefi[b].nb_dof();
     Curv[b].resize(Ni); gmm::clear(Curv[b]);
     lx[b].resize(Ni); gmm::clear(lx[b]);
     ly[b].resize(Ni); gmm::clear(lx[b]);
     lz[b].resize(Ni); gmm::clear(lx[b]);

     //Generating the P1 finite element for curvature
     getfem::mesh_fem mf_curv(mf_Coefi[b].linked_mesh());
     mf_curv.set_finite_element(mf_Coefi[b].linked_mesh().region(b).index(),
                                     fd_curv);

     //Generating the P0 finite element for tangent versor
     getfem::mesh_fem mf_tgv(mf_Coefi[b].linked_mesh());
     mf_tgv.set_finite_element(mf_Coefi[b].linked_mesh().region(b).index(),
                                     fd_tgv);

     //Interpolating v curvature on a different mesh fem
     gmm::row_matrix<vector_type> M_curv(mf_Coefi[b].nb_dof(),
                                     mf_curv.nb_dof());
     getfem::interpolation(mf_curv,mf_Coefi[b],M_curv);
     gmm::mult(M_curv,Curv_temp[b],Curv[b]);


       //Interpolating v curvature on a different mesh fem
     gmm::row_matrix<vector_type> M_tgv(mf_Coefi[b].nb_dof(),
                                     mf_tgv.nb_dof());
     getfem::interpolation(mf_tgv,mf_Coefi[b],M_tgv);
     gmm::mult(M_tgv,lx_temp[b],lx[b]);
     gmm::mult(M_tgv,ly_temp[b],ly[b]);
     gmm::mult(M_tgv,lz_temp[b],lz[b]);

     gmm::clear(M_tgv);     gmm::clear(M_curv);
} }
```

5.1.3 Assembling

For the system derivation we used some auxiliary functions, defined in the proper header (*assembling1d3d.hpp*, *assembling1d.hpp*, *assembling3d.hpp*, *c_assembling1d.hpp*). Many functions for the matrices implementation are already done in the a MixedFormulation, with some change of parameters, but the nonlinear part still needed to be implemented.

```
   //!  Build the advective non linear matrix for the 1D Navier-Stokes curved
          problem,
//!   NLM = ∫_Λ c  u  uold  ∇v·λ  ds
/*!
     NLM        Computed non linear matrix term
     mim        The integration method to be used
     mf_u       The finite element method for the velocity u
     mf_data    The finite element method for the tangent versor on Λ
     coef       The coefficient for NLM
     lambdax    First cartesian component of the tangent versor λ
     lambday    Second cartesian component of the tangent versor λ
     lambdaz    Third cartesian component of the tangent versor λ
     u_old     The velocity at the previous iteration
     rg         The region where to integrate
*/
template<typename MAT, typename VEC>
void asm_network_nonlinear(
     MAT & NLM,
     const mesh_im & mim,
     const mesh_fem & mf_u,
     const mesh_fem & mf_data,
     const VEC & coef,
     const VEC & lambdax,
     const VEC & lambday,
     const VEC & lambdaz,
     const VEC & u_old,
     const mesh_region & rg = mesh_region::all_convexes()
     )
{
     GMM_ASSERT1(mf_u.get_qdim() == 1,"invalid data mesh fem (Qdim=1 required)");
     // Build the local matrix for the non linear term NLMvvi
     generic_assembly
          assem("l1=data$1(#2); l2=data$2(#2); l3=data$3(#2);
                    u=data$4(#1); c=data$5(#2);"
               "t=comp(Base(#1).Base(#1).Grad(#1).Base(#2).Base(#2));"
               "M$1(#1,#1)+=t(j,:,:,1,m,k).l1(m).u(j).c(k)+
                    t(j,:,:,2,m,k).l2(m).u(j).c(k)+
                    t(j,:,:,3,m,k).l3(m).u(j).c(k);");
     assem.push_mi(mim);
     assem.push_mf(mf_u);
     assem.push_mf(mf_data);
     assem.push_data(lambdax);
     assem.push_data(lambday);
```

```
    assem.push_data(lambdaz);
    assem.push_data(u_old);
    assem.push_data(coef);
    assem.push_mat(NLM);
    assem.assembly(rg);
}
```

5.1.4 The library c_problem3d1d

In light of the final release of the code, we collect all routines external functions and variables defining out coupled 3D/1D curved problem,to build a *static* library. This is hence resolved in a caller at compile-time and copied into a target application by a compiler, linker, or binder, producing an object file and a stand-alone executable.
The library `libc_problem3d1d.a` can be automatically built using the provided Makefile and it is indeed adopted for all provided examples. It is built from following headers and sources:

`c_assembling1d.hpp`: Miscellanous non linear assembly routine for 1D network problem
`c_descr3d1d.hpp`Definition f the aux class for algorithm curved description strings
`c_mesh1d.hpp`: Miscellanous handlers for 1D curved network mesh
`c_param3d1d.hpp` Definition of the aux class for physical parameters
`c_problem3d1d.cpp` Declaration of the main class for 3D/1D coupled curved problem
`c_problem3d1d.hpp` Definition of the main class for 3D/1D coupled curved problem

Remember that this library must be used with the library *libproblem3d1d.a*
See the README file for installation instructions.

5.2 GraphGenerator

The GraphGenerator gives as simple tool in order to easily generate complex networks with curved geometry. For the implementation is used the *BGLgeom* library of M. Tantardini and I. Speranza, which is able manage and create complex network geometry.
To easily manage this problem we split it in three part: read the graph control points, generate the graph, generate two pts files containing the mesh and curved parameters on the points of the mesh.

5.2.1 Implementation

The first part is done through the class `B_reader_netdiff` (in the header *spline*$_r$*eader.hpp*) derived from the parent class `BGLgeom::ASCII_reader`. This class is able to read any ASCII file and to return the parameters needed to create the curved geometry.
The ASCII Input file has to be of the following format:

Each branch is written in the same format and consecutively to the previous one.
They need:

1) Boundaries conditions for the source and the target point.
2) The number of point insert for the branch.
3) The size of the mesh we want in the output for that branch.
4) The source of the branch.
5) The target of the branch.
6) All the other nodes in the correct order.

```
Example:

DIR 1.0 INT 0.0     (first branch)
3 100
0.0 0.0 0.0     (source node)
1.0 1.0 1.0     (target node)
0.5 0.5 0.5     (internal node)
INT 0.0 DIR 2.0     (second branch)
5 50
1.0 1.0 1.0
3.0 3.0 3.0
1.5 1.6 1.5
2.0 2.0 2.0
2.4 2.3 2.4
```

The last two parts are implemented in the function `curved_network_reader` (whose definition is in the header *curved_network_reader.hpp*). This function require as input the name and location of the `input.param` file, in which are specified the descriptors strings needed for the generation of the network.
In this function, each branch is read one for time and after that, through the class `BGLgeom::bspline_geometry`, it is constructed the branch using the *b_spline interpolation method* or *b_spline approximation method* and is generate a uniform mesh on it, whose size is given in the input file.
The output part is already implemented in the `BGLgeom` library and only require the name of the output pts file.
This is done in this form in order to give a simple tool to the user.

```
int main(int argc, char *argv[]){
     //Reading, constructing and exporting the network
     curved_network_reader(argc, argv);
     return 0;
}
```

And in the `input.param` file is only needed to be specify three parameters: `INPUT_ASCII="./ascii_file"` for the input ASCII file name and location, `INTERP=1` in order to use *b_spline interpolation method* for the branch construction, and `OUTPUT_PTS="./network.pts"` for the output pts file name and location.

5.2.2 The library graphgenerator

As for the previous part, all routines are collected to build a *static* library `libgraphgenerator.a`. That is automatically done by the Makefile and it is indeed adopted for the provided example. It is built from the following headers and source:

`spline_reader.hpp`: Definition of reader class for ASCII files
`curved_network_reader.hpp`: Definition of main function for the generation of the network
`curved_network_reader.cpp`: Declaration of main function for the generation of the network

Remember that this library must be used with the library *libBGLgeom.a*.
See the README file for installation instructions.

Bibliography

[1] D. Notaro, *Mixed Finite Element Methods for Coupled 3D/1D Fluid Problem*, Politecnico di Milano, Italy, 2015 .

[2] L. Formaggia and A. Veneziani, *Reduced and multiscale models for the human cardiovascular system*, MOX, Mathematics Department "F. Brioschi", Politecnico di Milano, Italy, 2003 .

[3] D. Lamponi, *One dimensional and multiscale models for blood flow circulation*, Ecole Polytechnique Federale de Lausanne, 2004 .

[4] T. Shifrin, *Differential Geometry: A First Course in Curves and Surfaces*, University of Georgia, 2016.

[5] L. Bergamaschi, S. Mantica and G. Manzini, *A mixed finite element-finite volume formulation of the black-oil model*, SIAM Journal on Scientific Computing, 20(3):970-977,1998.

[6] J. Bradts, S. Korotov and M. Krizek, *Simplicial finite elements in higher dimensionsl*, Appl. Math, 52(3):251-265,2007.