Industrial Training: Final Report

Saskia Loosveldt Tomàs

Fall 2017

1. Internship agreement

This internship was carried out during the summer of 2017 at LaCàN Multi-scale and multi-physics modelling in mechanobiology (4MB) research group. The decision to choose this research group was motivated by the possibility of extending the work carried out during the internship into the master's thesis.

Even though the main motivation of the activities developed during this internship is focused into an academic application, since the goal of this course is to apply the knowledge and skills learned into an industrial context, the final project consisted in developing a set of subroutines written in Fortran so that an external user can compile and run a program that solves different sets of examples and problems in Computational Mechanics.

2. Multi-scale and multi-physics modelling in mechanobiology (4MB) research group

The group's current efforts focus on the description of the smaller scales of biological tissues. They focus on understanding and defining the motility of single cell as well as interactions of cell aggregates by means of mathematical tools. The main interests focus in the interaction between mechanical clues and the intra-cellular chemical balance, and in bridging the gap between the organ level, e.g. the brain, or organoids and the subcellular level, e.g. adhesion molecules and actin dynamics. The modelling approaches cover multidisciplinary aspects of biology, mathematical models and state-of-the-art numerical methods.

3. Work performed

The internship started with the study of a set of Matlab codes previously written by other members of LaCàN. These codes were then adapted and re-written in Fortran and organized so that when the Makefile is executed, the main program lets a user choose which problem to solve and then calls the corresponding subroutines needed in order to discretise and solve the problem. The main advantage of re-writting these codes in Fortran is the reduction of the computation time. The subroutines are organized in modules in order to be able to include several of them in the same *use* statement at the main program.

Each problem requires an input file containing physical parameters, boundary conditions, as well as the data regarding the meshing of the domain. Unlike some of the original Matlab programs, the mesh is not created by the program and needs to be loaded from external files containing the node coordinates and the connectivity matrix. The nodes where boundary and initial conditions are applied also need to be loaded from an external file.

Once the mesh is loaded and the corresponding numerical quadrature and shape functions are called, the program then proceeds computing the matrices arising from the discretization of the problem for each element.

In order to solve the assembled system, the program calls the UMFPACK solver from SuiteSparse, which solves sparse systems using multifrontal LU factorization. The assembled matrix needs to be stored in a Compressed Sparse Row (CSR) format before entering the solver. Furthermore, since UMFPACK is written in C, the matrix needs to be stored following a 0-based index format and the program makes use of a wrapper to call the needed routines from C to Fortran. Both the subroutine for the conversion of the matrix to a 0-based CSR format and the wrapper were obtained from `https://github.com/timleslie/sparse`.

Finally, to represent the results, the program calls subroutines to generate vtk files for the physical paramaters of interest to be opened with Paraview.

Even though the program solves a specific set of examples, the way in which most of the subroutines are written allows the user to easily re-use them in the case of wanting to solve different examples and will probably be used for the master thesis.

The program developed currently allows to solve the following set of problems:

- Transient convection-diffusion

This example solves a transient convection-diffusion equation with a source term consisting of a rotating pulse:

$$u_t + \vec{a} \cdot \nabla u - \nabla \cdot (\nu \nabla u) = s \quad in\ \Omega = [0,1] \times [0,1] \tag{1}$$

$$\vec{a} = (-y - 0.5, x - 0.5) \tag{2}$$

$$s = e^{-t^{10}} cos(\frac{\pi}{2} \|\hat{\mathbf{x}}\|)\ if\ \hat{\mathbf{x}} = \frac{\mathbf{x} - (0.2, 0.2)}{0.2} \leq 1 \tag{3}$$

with the following initial and boundary conditions:

$$u(\vec{x}, 0) = 0 \quad in\ \Omega \tag{4}$$

$$u(0, y, t) = u(1, y, t) = 0, \quad 0 \leq y \leq 1, t \geq 0 \tag{5}$$

$$u(x, 0, t) = u(x, 1, t) = 0, \quad 0 \leq x \leq 1, t \geq 0 \tag{6}$$

The domain is discretized using Galerkin with linear quadrilatera and a four-point Gauss quadrature. As stated previously, the node coordinates and connectivities need to be loaded in external files. The time is discretized following a θ-scheme, where the value of θ is introduced in the input file. The final time and time step have to be defined as an input as well.

At each time step, the program writes the computed values of u in the corresponding vtk file so that the evolution of the rotating pulse can be plotted over time.

(a) Timestep 1

(b) Timestep 11

Figure 1: Solution of the transient convection-diffusion problem using linear elements

- Hyperelastic problem

This example solves the deformation of an hyperlastic 3D solid. The material is assumed to be Neo-hookian, in which the first Piola-Kirchhoff stress tensor and the tangent operator are defined as:

$$P_{ij} = \mu_0 F_{ij} + [\lambda_0 ln(det(F_{ij})) - \mu_0] F^{-1}_{ji} \tag{7}$$

$$A_{ijkl} = \lambda_0 F^{-1}_{ji} F^{-1}_{lk} + \mu_0 I^{-1}_{ik} I^{-1}_{jl} + [\mu_0 - \lambda_0 ln(det(F_{ij}))] F^{-1}_{li} F^{-1}_{jk} \tag{8}$$

Considering the strong residual form of the balance of the linear momentum:

$$\mathbf{R}^{\phi} = -\nabla \cdot \mathbf{P} = 0 \; in \; \Omega \tag{9}$$

And considering the fact that the material is hyperelastic, the residual needs to be linearized, in this case, using Newton-Raphsons method:

$$R_I^{\phi(k+1)} = R_I^{\phi(k)} + dR_I^{\phi} = 0 \tag{10}$$

Where the last term depends on the tangent operator defined in (8).

The domain is discretized using Galerkin with tetrahedra elements and an eight-point Gauss quadrature. The number of steps in which the material gets loaded is stated in the input file, along with the Lamé parameters and the tolerance of the Newton-Raphson iteration. Once convergence has been reached, the deformed solid is plotted in Paraview.

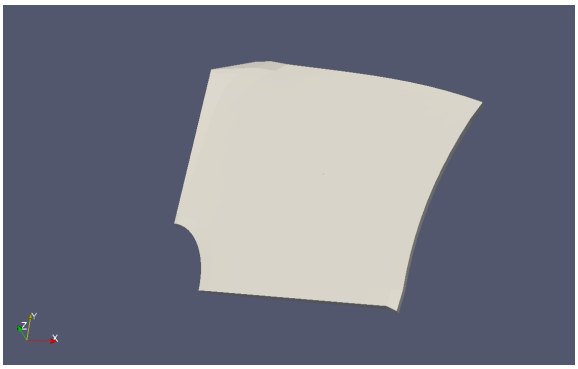


Figure 2: Solution for the hyperelastic problem

- Stokes problem

This example solves the Stokes leaky cavity problem. The upper wall of the domain moves with a velocity $\vec{v}$=(1,0) and the flow is confined setting the value of the pressure p=0 at the point (0,0)

$$-\nu\nabla^2\vec{v} + \nabla p = \vec{0} \quad in\ \Omega = [0,1] \times [0,1] \tag{11}$$

$$\nabla \cdot \vec{v} = 0\ in\ \Omega \tag{12}$$

$$v(0,y) = v(1,y) = \vec{0}\ for \quad 0 \leq y \leq 1 \tag{13}$$

$$v(x,0) = \vec{0}\ for \quad 0 \leq x \leq 1 \tag{14}$$

$$v(x,1) = (1,0)\ for \quad 0 \leq x \leq 1 \tag{15}$$

$$p(0,0) = 0 \tag{16}$$

The problem is discretized using quadratic quadrilatera for the velocity, linear quadrilatera for the pressure, a pair that is compliant with the inf-sup condition, and a nine-point Gauss quadrature.

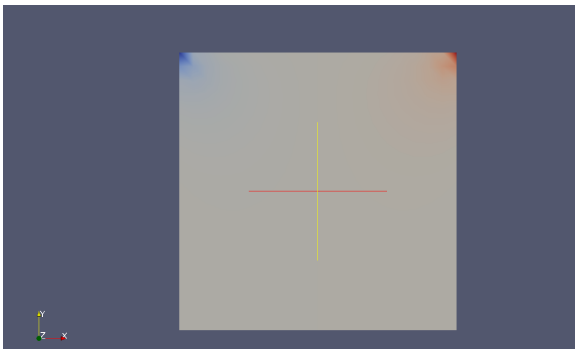


Figure 3: Solution for the pressure field in the Stokes problem

- Stokes problem discretized using an HDG formulation

In this examples, a Stokes flow problem is solved using the Hybridizable Discontinous Galerkin method, which has the following advantatges with respect to continous and discontinous Galerkin methods:

-The computational efficiency is similar, but can be better, than continous methods and outperforms discontinous methods.

-The methods yields to a superconvergent solution.

-For incompressible flows, u-p couples that do not satisfy the LBB condition yield to stable solutions.

This formulation solves an element-by-element local problem and then solves a global problem that depends on the trace of the velocity in the elements face and the average pressure in the element.

- Coupled mechanic-transport problem

This example implements a monolithic incremental iterative Newton-Raphson method in order to linearize the system of equations that describe the first stage of a wound healing model. In this stage, an epithelial tissue modelled as a 2D hyperelastic solid in an equilibrium state gets deformed as a result of a wound infliction. The mechanical deformation caused by the wound triggers a burst of calcium, which behaves as a transient diffusive flow with a source term that couples the calcium transport to the mechanical deformation. This particular model neglects body forces for the mechanical part and considers an isotropic diffusion coefficient for the transport part.

Considering the strong residual form of the mechanical and transport equations:

$$\mathbf{R}^{\phi} = -\nabla \cdot \mathbf{P} = 0 \; in \; \Omega \tag{17}$$

$$R^{u} = u_t - \nu \nabla^2 u - F^{u} = 0 \; in \; \Omega \tag{18}$$

$$\boldsymbol{\phi}(\vec{x}, t) = 0 \; on \; \Gamma_D \tag{19}$$

$$u(x, t) = u_o \; at \; t = 0 \tag{20}$$

where the calcium source term F^u is a function of the first strain invariant I_1, u and parameters related to maximum and minimum calcium creation rates, tissue sensibility and the value of the strain that triggers the calcium generation :

$$F^{u} = F(I_1, u) = (F_{min} + \frac{\alpha}{1 + e^{-\beta(I_1 - \gamma)}})(1 - (\frac{u}{u_{lim}})^2) \tag{21}$$

The domain gets discretized using linear quadrilatera and a Gauss quadrature of four points, and the temporal discretization is done following an implicit backward Euler scheme. Then the residual gets linearized at each time step n following a Newton-Raphson iteration k, where the tolerance needs to be stated as an input:

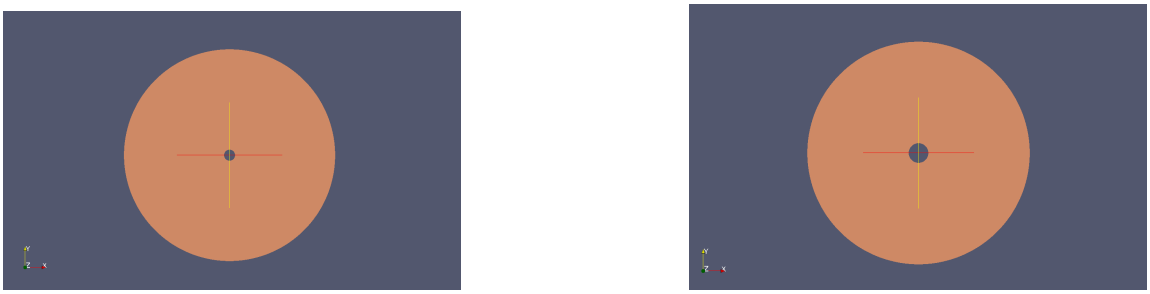
$$R^{\phi(k+1)}_{I(n+1)} = R^{\phi(k)}_{I(n+1)} + dR^{\phi}_{I} = 0 \tag{22}$$

$$R^{u(k+1)}_{J(n+1)} = R^{u(k)}_{J(n+1)} + dR^{u}_{J} = 0 \tag{23}$$

and where the variation of the residuals couples both equations in the following way:

$$dR_I^{\phi} = \sum_K \frac{dR_I^{\phi}}{d\phi_K} du_K + \sum_L \frac{dR_I^{\phi}}{d\phi_L} d\phi_L \tag{24}$$

$$dR_J^{u} = \sum_K \frac{dR_J^{u}}{du_K} du_K + \sum_L \frac{dR_J^{u}}{d\phi_L} d\phi_L \tag{25}$$



(a) Timestep 1

(b) Timestep 30

Figure 4: Evolution of the wound with respect to time

4. Personal Evaluation

On one hand, this internship has helped me review concepts and problems already learned in other master courses such as solving transient convection-diffusion problems, Stokes problems or using Newton-Raphson to linearize equations. On the other hand, with this internship I have learned new concepts regarding the theory and full implementation of the finite element method for solving hyperelastic problems, the discretization and implementation using HDG and the implementation of the finite element method for solving a coupled problem.

From a coding point of view, even though I had some previous experience writing code in Fortran 77, I have learned how to use and take advantage of the updated version of the language. Furthermore, I have learned how to call external libraries needed in order to solve systems of equations, as well as properly linking them to the Makefile.

Overall, with this internship, I have learned a lot of new concepts and skills that will certainly be useful and will help me in the completion of this Masters degree.