From medical imaging to numerical simulations

C. Prud’homme

joint work with A. Ancel, V. Chabannes, J. Jomier, M. Szopos

1 University de Strasbourg, Strasbourg, France
2 University Joseph Fourier, Grenoble, France
3 Kitware, Lyon, France

The International Congress of
ADVANCED TECHNOLOGIES & TREATMENTS FOR GLAUCOMA

October 2015, Politecnico di Milano, Milan, Italy.
Vivabrain: A Multi-Disciplinary Project

Physics
- Medical imaging: MRI, MRA

Computer Science
- Image processing
- Model generation

Mathematics
- Numerical analysis
- Uncertainty quantification
- High performance computing

Medicine
- Vascular anatomy
- Haemodynamics
AngioTK Platform

From physical images to virtual images

Simulate angiographic data in MRI based on realistic anatomical and dynamical models obtained from real data

Open Source Software

- ITK (Kitware): Angiographic images analysis
- FEEL++ (Feel++ Consortium): blood flow and rheology simulation
- VMTK / VTK: Mesh generation and processing
- RORPO (ESIEE, URCA): Path-opening
- GMSH: mesh generation and remeshing
- JEMRIS: MRI simulation

http://www.github.com/vivabrain/angiotk
Stores 3D and 3D+t Vascular models and virtual angiographies
http://vivabrain.u-strasbg.fr
Starting from initial acquired data, filter it to emphasize tubular structures
AngioTK Platform: Segmentation/Surface extraction/Mesh processing using VMTK

Extract veins and arteries using level-set segmentation; time ~40s
AngioTK Platform: Centerlines to define tubular structures

Centerline points associated to data (radius inscribed sphere); graphical tool for centerlines: seed points, corrections; time 20 min.
Build an image with a very good quality without artifacts using centerlines and compute signed distance function; controlled accuracy.
Extract the surface corresponding to level 0 of distance function; smooth, remesh if necessary; time ~30s
AngioTK Platform: Building computational mesh using GMSH

- Opening the tubular structure at extremities of centerlines
- Remesh the surface by using centerlines
  - Based on [1]
  - Good quality (frontal algo)
  - Same accuracy between small and large vessels
- Mesh the volume by using centerlines
- Optional: build the arterial wall (boundary layer extrusion with thickness as percent of radius)
- Partition the mesh for numerical simulation
- Current sample time: 10 min

Simulation using some Feel++ toolboxes:

- Fluid and Solid Mechanics and Interaction
- Convection-Diffusion

About the modeling:

- Blood models:
  - Incompressible Navier-Stokes
  - Newtonian and non-Newtonian
  - Inlets BC: Velocity, Pressure
  - Outlets BC: Free, Windkessel

- Arterial wall models
  - Elastic, hyper-elastic
  - Compressible, nearly-incompressible
  - BC: fixed, free, external tissue

Solvers:

- Finite element method
- HPC (e.g. Prace) up to tens of thousands of cores

In-Situ Visualisation:

- Paraview/Catalyst (second per timestep)
Challenge: Computational mesh generation

- Build good computational meshes and allow extension such as building vessel wall, currently we control the mesh quality and accuracy.
Challenge: Fusion of vessels

- Automate the process as much as possible, current difficulties centerlines and fusion of vessels

Initial Data
Challenge: Fusion of vessels

- Automate the process as much as possible, current difficulties centerlines and fusion of vessels

Initial Data

Reconstruction failed
Solution: Automatic Vessel Fusion Handling

- Automate the process as much as possible, current difficulties centerlines and fusion of vessels

Initial Data | Reconstruction failed | Reconstruction succeeded
---|---|---

C. Prud’homme et al. From medical images to numerical simulations
Solution: Automatic Vessel Fusion Handling

- Automate the process as much as possible, current difficulties centerlines and fusion of vessels
What do we do with these realistic geometries and simulations?

Study the influence of modeling choices in the cerebral venous network

Using a deterministic analysis framework, study the influence of
- Inflow boundary conditions
- Outflow boundary conditions
- Blood constitutive law: Newtonian vs non-Newtonian

HPC is a requirement: many simulations and very intensive post-processing computations!

What do we do with these realistic geometries and simulations?

Eye2Brain project
Multiscale modeling of fluid-dynamical and metabolic links between eye and brain: towards ocular biomarkers for neurodegenerative disorders

Joint work with
- IUPUI J. Arciero, L. Carichi, S. Cassani, D. Prada, G. Guidoboni
- Glick Institute A. Harris, B. Siesky
- Politecnico di Milano R. Sacco
- U. Strasbourg M. Szapos C. Prud’homme

- AngioTK is the framework for constructing realistic geometries from medical images
- Feel++ is the underlying computational framework to support detailed views in the eye and in the brain of
  - blood flows
  - oxygen and metabolites transport
What do we do with these realistic geometries and simulations?

Eye2Brain project

A first reconstruction from retina image of some vessels using AngioTK from a 2D image
Thank you for your attention!
Example: MRI data of the vivabrain project

Circle of Willis
Example: IRM data of the vivabrain project

Cerebral venous network
Example: Phantom with FSI simulation

Wave pressure propagation
Services for AngioTK

- Custom development
- Deployment
- Consulting
- Training

Contacts
- C. Prud’homme: christophe.prudhomme@cemosis.fr
- J. Jomier: julien.jomier@kitware.com
Feel++

Finite Element Embedded Library and Language in C++
A Domain Specific Language for PDEs embedded in C++ providing a syntax very close to the mathematical language

Features

▶ Supports generalized arbitrary order Galerkin methods (cG, dG) in 1D, 2D and 3D
▶ Supports simplex, hypercube and high order meshes
▶ Supports seamless parallel computing, scales up to tens of thousands of cores (Tier-0/Prace,Tier-1/Genci)
▶ Supports large scale parallel linear and non-linear solvers (PETSc/SLEPc)

Contact
C. Prud’homme: christophe.prudhomme@cemosis.fr, feelpp.org
Feel++ Sample code: Laplacian

\textbf{Laplacian problem}

\begin{align*}
\text{Find } u \text{ such that:} & \\
-\Delta u &= 1 \text{ in } \Omega \\
u &= 0 \text{ on } \partial \Omega
\end{align*}

\begin{align*}
\text{variational} & \quad \Rightarrow \quad \text{formulation} \\
\int_\Omega \nabla u \cdot \nabla v &= \int_\Omega fv, \quad \forall v \in V_h
\end{align*}

\begin{verbatim}
auto mesh = loadMesh(_mesh=new Mesh<Simplex<2> >); auto Vh = Pch<2>( mesh ); auto u = Vh->element(), v = Vh->element(); auto f = expr( "2*x*y+cos(y):x:y" );
\end{verbatim}

\begin{verbatim}
// a(u,v) = \int_\Omega \nabla u \cdot \nabla v auto a = form2(_trial=Vh,_test=Vh);
a = integrate(_range=elements(mesh),
\_expr=gradt(u)*trans(grad(v)) ) ;
\end{verbatim}

\begin{verbatim}
// l(v) = \int_\Omega fv auto l = form1(_test=Vh);
l = integrate(_range=elements(mesh),
\_expr=f*id(v)) ;
\end{verbatim}

\begin{verbatim}
// u = 0 on \partial \Omega a+=on(_range=boundaryfaces(mesh),
\_rhs=1,\_element=u,
\_expr=cst(0.) ) ;
\end{verbatim}

\begin{verbatim}
// solve algebraic system a.solve(_rhs=1,_solution=u);
\end{verbatim}
Stokes problem

\[
\begin{cases}
\text{Find} \ (u, p) \ \text{such that}: \\
\quad -\Delta u + \nabla p = 0 \ \text{in} \ \Omega \\
\quad \nabla \cdot u = 0 \ \text{in} \ \Omega
\end{cases}
\]

\[
\begin{cases}
\quad \text{Find} \ (u, p) \in V_h \times Q_h \ \text{such that} \\
\quad \int_{\Omega} \nabla u : \nabla v + p \nabla \cdot v = 0, \ \forall v \in V_h \\
\quad \int_{\Omega} q \nabla \cdot u = 0, \ \forall q \in Q_h
\end{cases}
\]

```cpp
auto mesh = loadMesh(_mesh=new Mesh<Simplex<2>));
auto Vh = THch<1>( mesh ); // P2P1
auto U = Vh->element(); V = Vh->element();
auto u = U.element<0>(), v = V.element<0>();
auto p = U.element<1>(), q = V.element<1>();

// \(a(u, p, v, q) = \int_{\Omega} \nabla u : \nabla v - p \ \nabla \cdot v + q \ \nabla \cdot u\)
auto a = form2(_trial=Vh, _test=Vh);
a = integrate(_range=elements( mesh ),
\quad _expr=inner( \text{grad}(u), \text{grad}(v) ) );
a += integrate(_range=elements( mesh ),
\quad _expr=-\text{idt}(p)*\text{div}(v) + \text{id}(q)*\text{divt}(u) );

// \(l(v) = 0\)
auto l = form1(_test=Vh);

// \(u = g\) on wall
a+=on(_range=markedfaces(mesh,"wall"),_rhs=l,
\quad _element=u,_expr=g);

// solve algebraic system
a.solve(_rhs=l,_solution=U);
```
Some Feel++ Features

Simulation using Feel++ tool boxes:
- Fluid Mechanics
- Solid Mechanics
- Fluid-Structure Interaction

About the modelisation:
- Blood models:
  - Incompressible Navier-Stokes
  - Newtonian and non-newtonian
  - Inlets BC: Velocity, Pressure
  - Outlets BC: Free, Windkessel
- Arterial wall models:
  - Elastic, hyper-elastic
  - Compressible, nearly-incompressible
  - BC: fixe, free, external tissue

Solvers:
- Finite element method
- HPC
Services for Feel++

- Custom development
- Deployment
- Consulting
- Training

Contact

C. Prud’hommee: christophe.prudhomme@{cemosis.fr,feelpp.org}