From medical imaging to numerical simulations

C. Prud'homme [1]

joint work with A. Ancel [1] V. Chabannes [2] J. Jomier [3] M. Szopos[1]

[1]University de Strasbourg, Strasbourg, France

[2]University Joseph Fourier, Grenoble, France

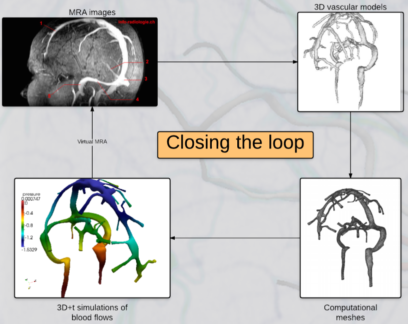
[3]Kitware, Lyon, France

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

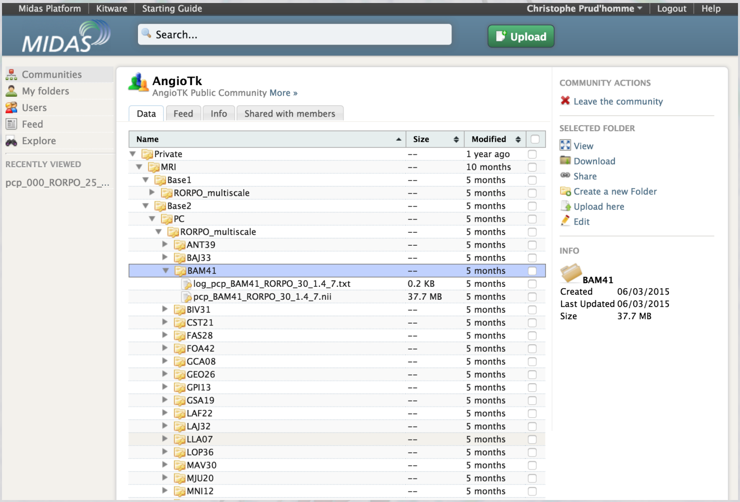
Simulate angiographic data in MRI based on realistic anatomical and dynamical models obtained from real data

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

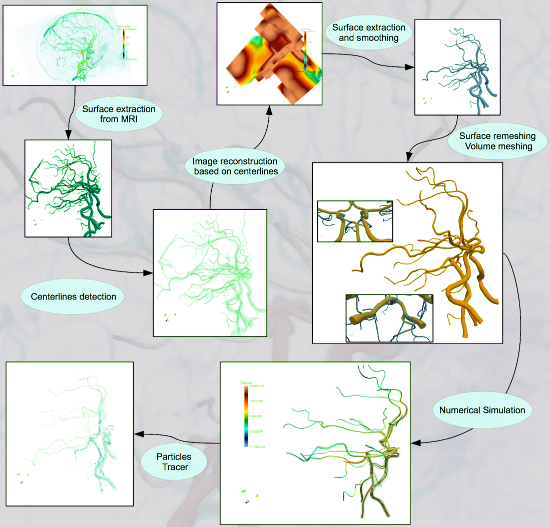
http://www.github.com/vivabrain/angiotk

AngioTK Platform: Open Data Server using MIDAS

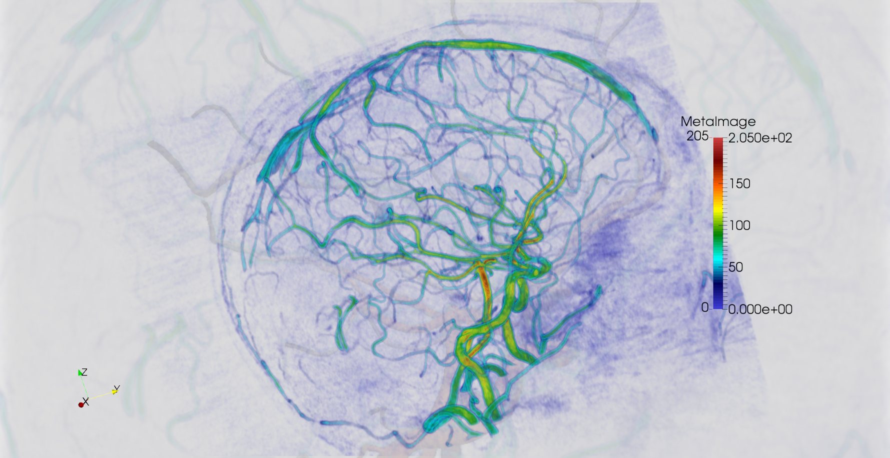


Stores 3D and 3D+t Vascular models and virtual angiographies
http://vivabrain.u-strasbg.fr

AngioTK Platform: A Software Pipeline

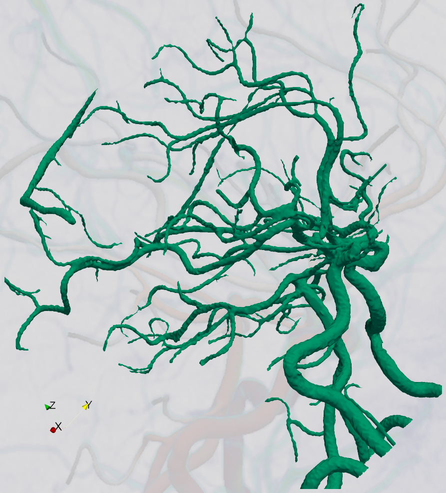


AngioTK Platform: Filtering using RORPO



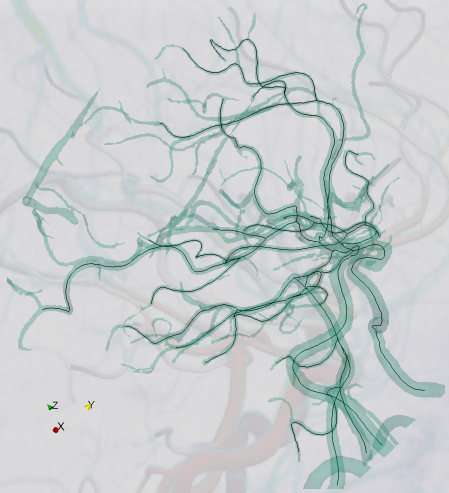
Starting from initial acquired data, filter it to emphasize tubular structures

AngioTK Platform: Segmentation/Surface extraction/Mesh processing using VMTK



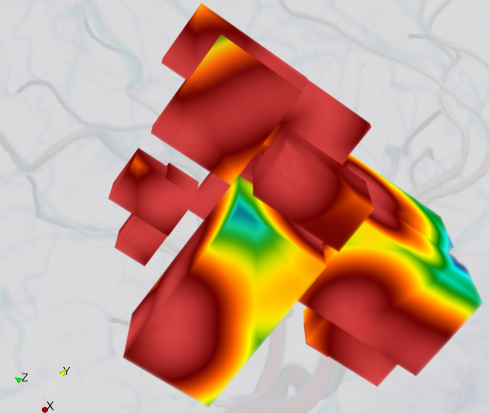
Extract veins and arteries using level-set segmentation; time ~40s

AngioTK Platform: Centerlines to define tubular structures



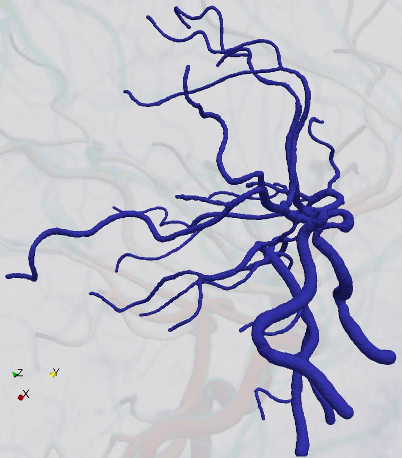
Centerline points associated to data (radius inscribed sphere); graphical tool for centerlines: seed points, corrections; time 20 min.

AngioTK Platform: Image reconstruction



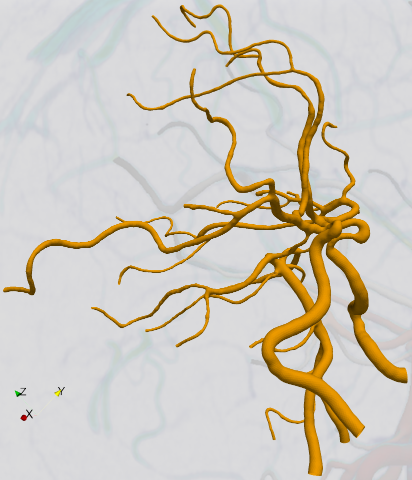
Build an image with a very good quality without artifacts using centerlines and compute signed distance function; controlled accuracy.

AngioTK Platform: Surface extraction



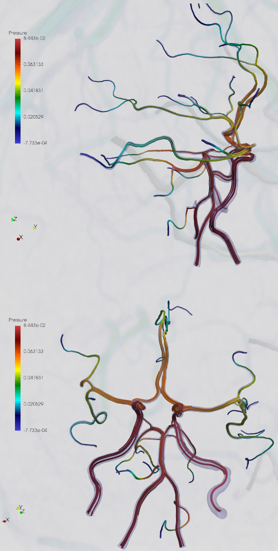
Extract the surface corresponding to level 0 of distance function; smooth, remesh if necessary; time ~30s

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

Marchandise, E and Geuzaine, C and Remacle, JF Cardiovascular and lung mesh generation based on centerlines. *International journal for numerical methods in biomedical engineering.*, Vol. **29** 665–682, 2013.



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 :

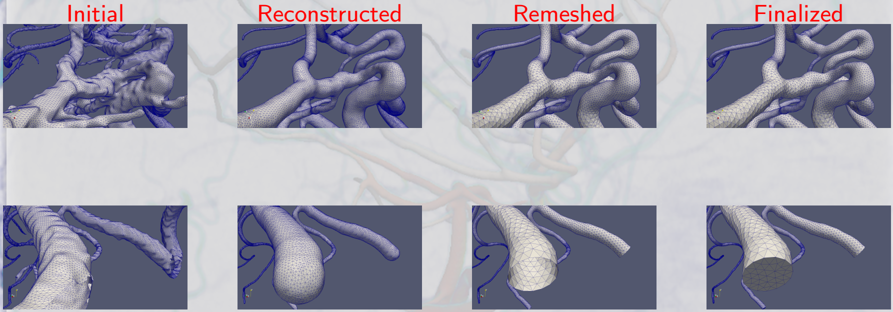
- Finite element method
- HPC (e.g. Prace) up to tens of thousands of cores

In-Situ Visualisation :

- Paraview/Catalyst (second per timestep)

Challenge: Computational mesh generation

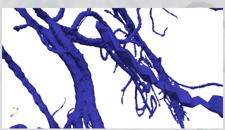
- Build good computational meshes and allow extension such as building vessel wall, currently we control the mesh quality and accuracy

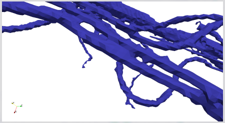


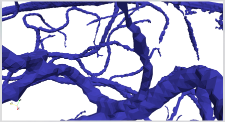
Challenge: Fusion of vessels

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

Initial Data



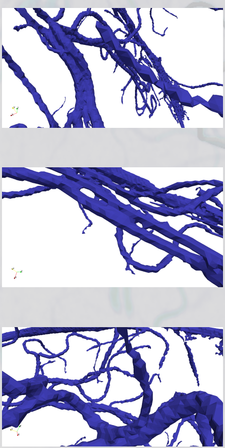




Challenge: Fusion of vessels

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

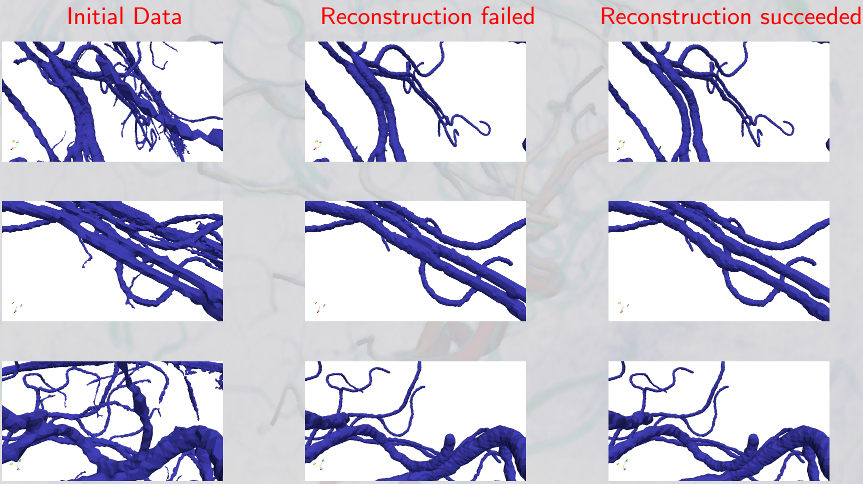
Initial Data



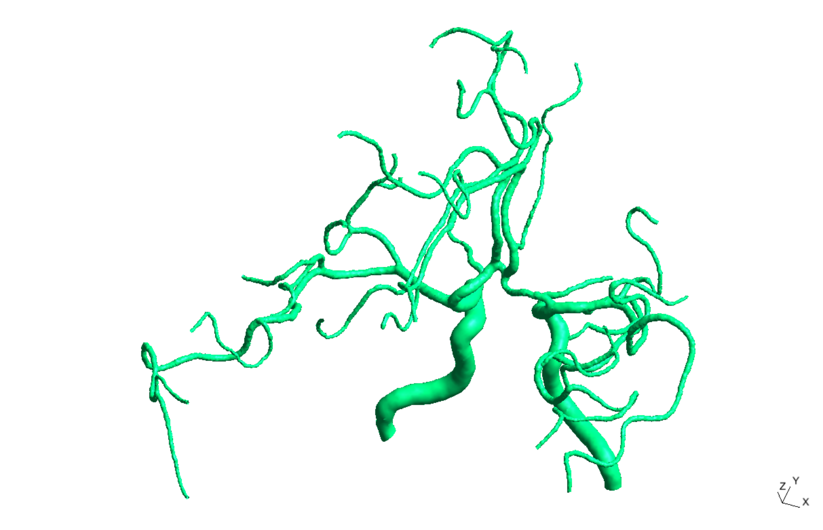
Reconstruction failed

Solution: Automatic Vessel Fusion Handling

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

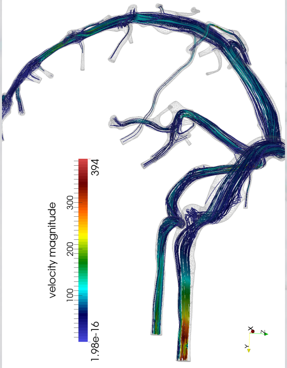


- Automate the process as much as possible, current difficulties centerlines and fusion of vessels



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!

V. Chabannes, M. Ismail, C. Prud'Homme, M. Szopos. Hemodynamic simulations in the cerebral venous network: A study on the influence of different modeling assumptions. Journal of Coupled Systems and Multiscale Dynamics, 2015, 3 (1), pp.23-37. <10.1166/jcsmd.2015.1062>

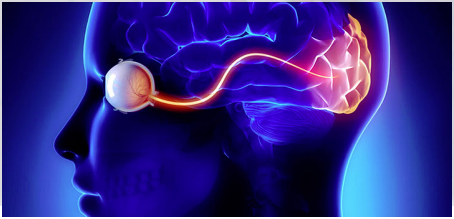
Eye2Brain project

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

Joint work with

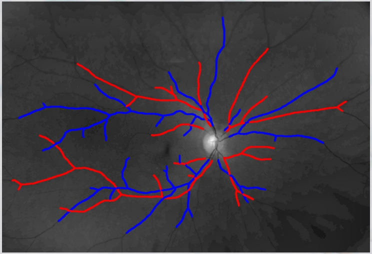
- IUPUI **J. Arciero, L. Carichi, S. Cassani, D. Prada, G. Guidoboni**
- Glick Institute **A. Harris, B. Siesky**
- Politechnico di Milano **R. Sacco**
- U. Strasbourg **M. Szopos C. Prud'homme**

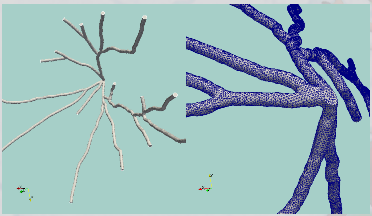
- AngioTK is the framework for constructing realistic geometries from medical images
- Feel++ is the underlying computational framework to support detailed views in the eye and in the brain of
  - blood flows
  - oxygen and metabolites transport



What do we do with these realistic geometries and simulations?

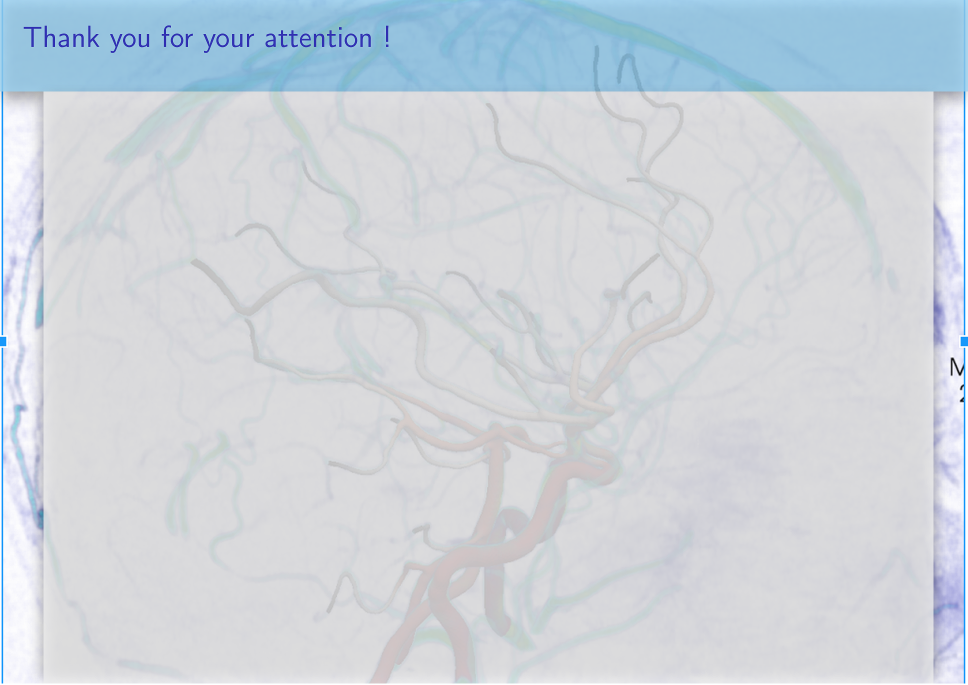
Eye2Brain project

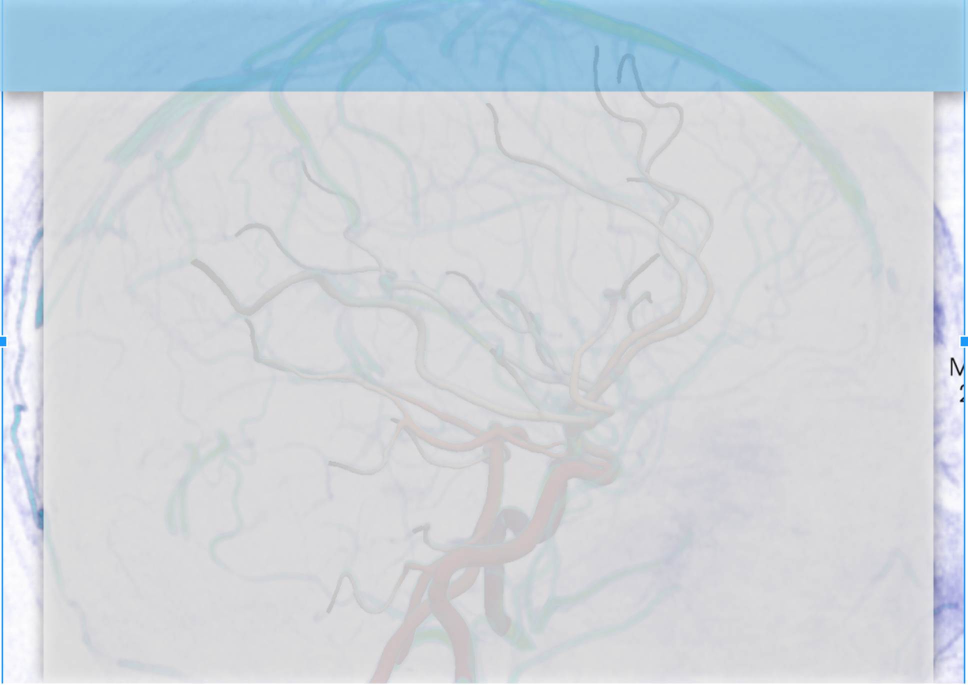




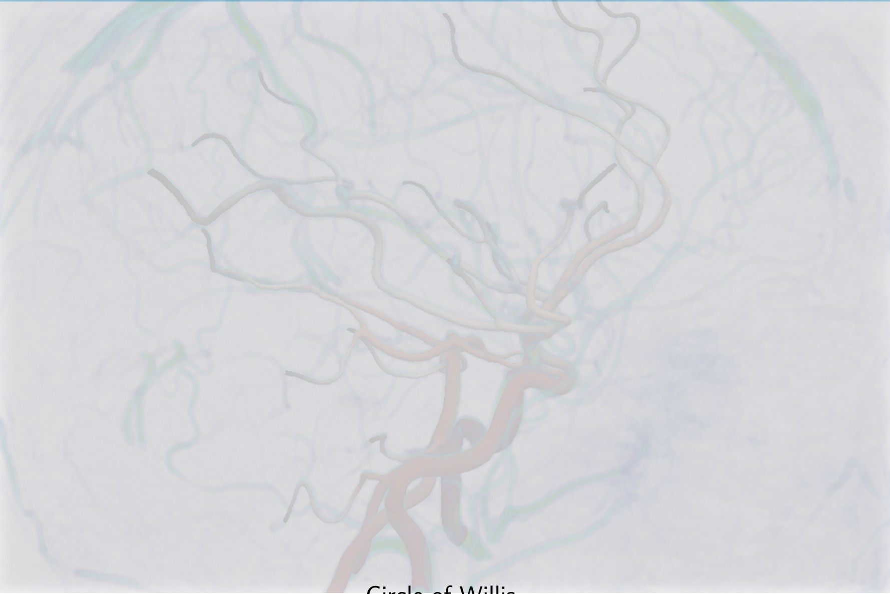
A first reconstruction from retina image of some vessels using AngioTK from a 2D image

Thank you for your attention !



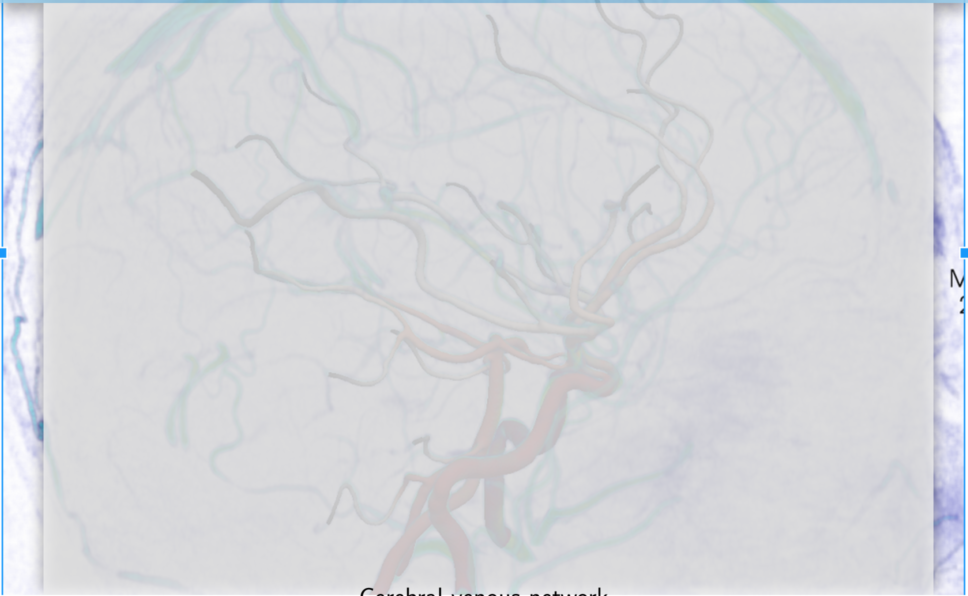


Example: MRI data of the vivabrain project



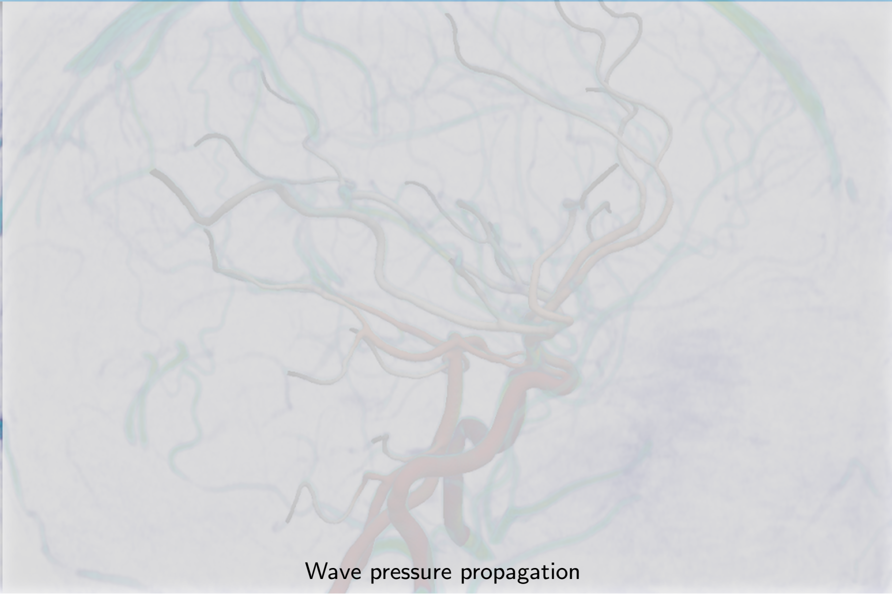
Circle of Willis

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

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}

Laplacian problem

$$\begin{cases} \text{Find } u \text{ such that :} \\ -\Delta u = 1 \quad \text{in } \Omega \\ \quad u = 0 \quad \text{on } \partial\Omega \end{cases} \boxed{\overset{\text{variational}}{\Longrightarrow} \text{formulation}} \begin{cases} \text{Find } u \in V_h \text{ such that} \\ \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv, \ \forall v \in V_h \end{cases}$$

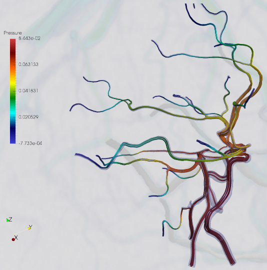
```
auto mesh = loadMesh(_mesh=new Mesh<Simplex<2ls);
auto Vh = Pch<2>( mesh );
auto u = Vh->element(), v = Vh->element();
auto f = expr( "2*x*y+cos(y):x:y" );
// a(u,v) = ∫_Ω ∇u · ∇v
auto a = form2(_trial=Vh,_test=Vh);
a = integrate(_range=elements(mesh),
              _expr=gradt(u)*trans(grad(v)) );
// l(v) = ∫_Ω fv
auto l = form1(_test=Vh);
l = integrate(_range=elements(mesh),
              _expr=f*id(v));
// u = 0 on ∂Ω
a+=on(_range=boundaryfaces(mesh),
      _rhs=l,_element=u,
      _expr=cst(0.) );
// solve algebraic system
a.solve(_rhs=l,_solution=u);
```

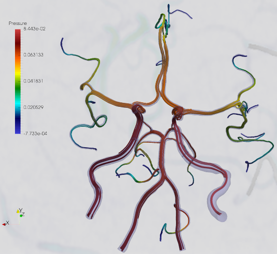
Stokes problem

$$\begin{cases} \text{Find } (u,p) \text{ such that :} \\ -\Delta u + \nabla p = 0 \quad \text{in } \Omega \\ \nabla \cdot u = 0 \quad \text{in } \Omega \end{cases} \quad \boxed{\overset{\text{variational}}{\Longrightarrow} \atop \text{formulation}} \quad \begin{cases} \text{Find } (u,p) \in V_h \times Q_h \text{ such that} \\ \int_\Omega \nabla u : \nabla v + p \nabla \cdot v = 0, \forall v \in V_h \\ \int_\Omega q \nabla \cdot u = 0, \ \forall q \in Q_h \end{cases}$$

```
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) = ∫_Ω ∇u : ∇v − p ∇·v + q ∇·u
auto a = form2(_trial=Vh, _test=Vh);
a = integrate(_range=elements( mesh ),
              _expr=inner( gradt(u),grad(v) ) );
a +=integrate(_range=elements( mesh ),
              _expr=-idt(p)*div(v) + id(q)*divt(u) );
// l(v) = 0
auto l = form1(_test=Vh );
// u = g on wall
a+=on(_range=markedfaces(mesh,"wall"),_rhs=l,
      _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'homme: christophe.prudhomme@{cemosis.fr,feelpp.org}