PI: **Alfio Quarteroni**

co-PI: **Simone Deparis**

EPFL - Lausanne

CMCS

EPFL

HPC for Cardiovascular System Simulations

Kick-off meeting

HP2C High Performance and High Productivity Computing

USI - Lugano, March 16th-17th 2010

Team

PI: Alfio Quarteroni
coPI: Simone Deparis, PhD
Senior: Gilles Fourestey, PhD

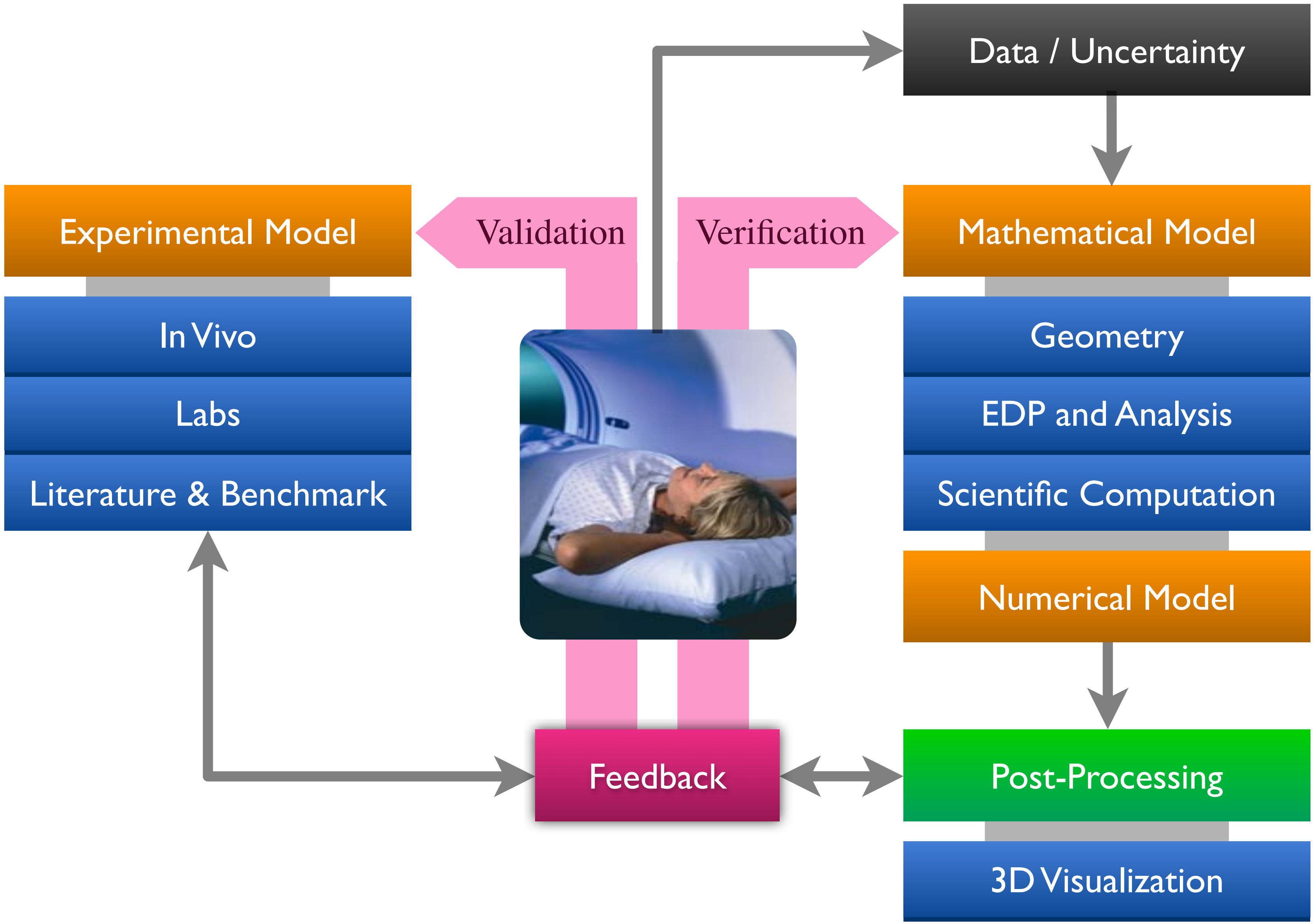
Developers involved in HP2C:
G. Grandperrin, PhD student (HP2C)
XXX, PhD student (HP2C)
YYY, post-doc (HP2C)
P. Crosetto, PhD student (FSI), post-doc from Jan 2011 (HP2C)

Other LifeV developers @CMCS (not all on Blood Flow Modeling):
2 post-docs, 4 PhD students

LifeV: a C++ finite element research library, about 135000 lines
http://www.lifev.org/

Other related projects:
MATHCARD (ERC Advanced-Grant), VPH2 (EU-FP7), 2 FNS projects.

Mathematical Modeling: a Mathematician's Perspective



Blood-flow Modeling: Main Motivations

- *Medical research.* A number of vascular diseases are linked to local haemodynamics. For instance atherosclerosis or cerebral aneurysms

- *Design of prostheses.* The design of endoprosthesis or other devices (such as stents) may be aided by the knowledge of haemodynamics

- *Surgical planning and optimization.* Alterated flow conditions following a surgical procedure (i.e. a bypass) may have negative effects and cause post-surgical failure

- *Training surgeons and anesthesiologists.* Simulator of the cardiovascular system may be used for training purposes

THE MATHEMATICAL MODEL

The coupled fluid-structure problem

Equations for the geometry:

$$\widehat{\eta}_f = \mathrm{Ext}(\widehat{\eta}_{s|\widehat{\Gamma}}),\ \widehat{\mathbf{w}} = \frac{\partial \widehat{\eta}_f}{\partial t}, \Omega_f(t) = (I + \widehat{\eta}_f)(\widehat{\Omega}_f)$$

Equations for the fluid:

$$\frac{\rho_f}{J_{\widehat{\mathcal{A}}}} \frac{\partial J_{\widehat{\mathcal{A}}} \mathbf{u}_f}{\partial t}_{|\widehat{\mathbf{x}}} + \mathrm{div}(\rho_f \mathbf{u}_f \otimes (\mathbf{u}_f - \mathbf{w}) - \sigma_f(\mathbf{u}_f, P)) = 0, \quad \text{in} \quad \Omega_f(t)$$

$$\mathrm{div}\mathbf{u}_f = 0, \quad \text{in} \quad \Omega_f(t)$$

$$\mathbf{u}_f = \mathbf{u}_D, \quad \text{on} \quad \Gamma_{f,D}$$

$$\sigma_f(\mathbf{u}_f, P)\mathbf{n}_f = \mathbf{g}_{f,N}, \quad \text{on} \quad \Gamma_{f,N}$$

$$\mathbf{u}_f = \mathbf{w}, \quad \text{on} \quad \Gamma(t)$$

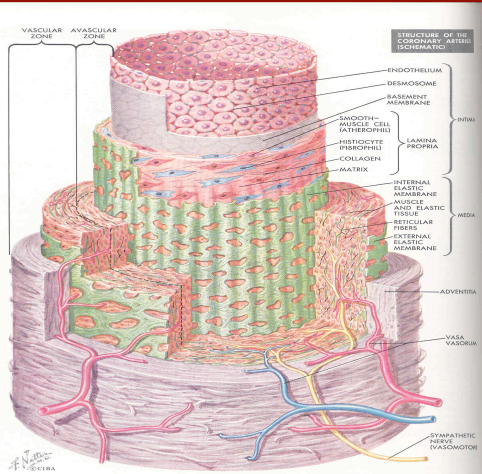
Equations for the structure:

$$\widehat{\rho}_{s,0} \frac{\partial^2 \widehat{\eta}_s}{\partial t^2} - \mathrm{div}_{\widehat{\mathbf{x}}}(\widehat{\mathbf{F}}_s \widehat{\Sigma}) = 0, \quad \text{in} \quad \widehat{\Omega}_s$$

$$\widehat{\eta}_s = 0 \quad \text{on} \quad \widehat{\Gamma}_{s,D}$$

$$\widehat{\mathbf{F}}_s \widehat{\Sigma} \widehat{\mathbf{n}}_s = \widehat{J}_s |\widehat{\mathbf{F}}_s^{-T} \widehat{\mathbf{n}}_s| \widehat{\mathbf{g}}_{s,N}, \quad \text{on} \quad \widehat{\Gamma}_{s,N}$$

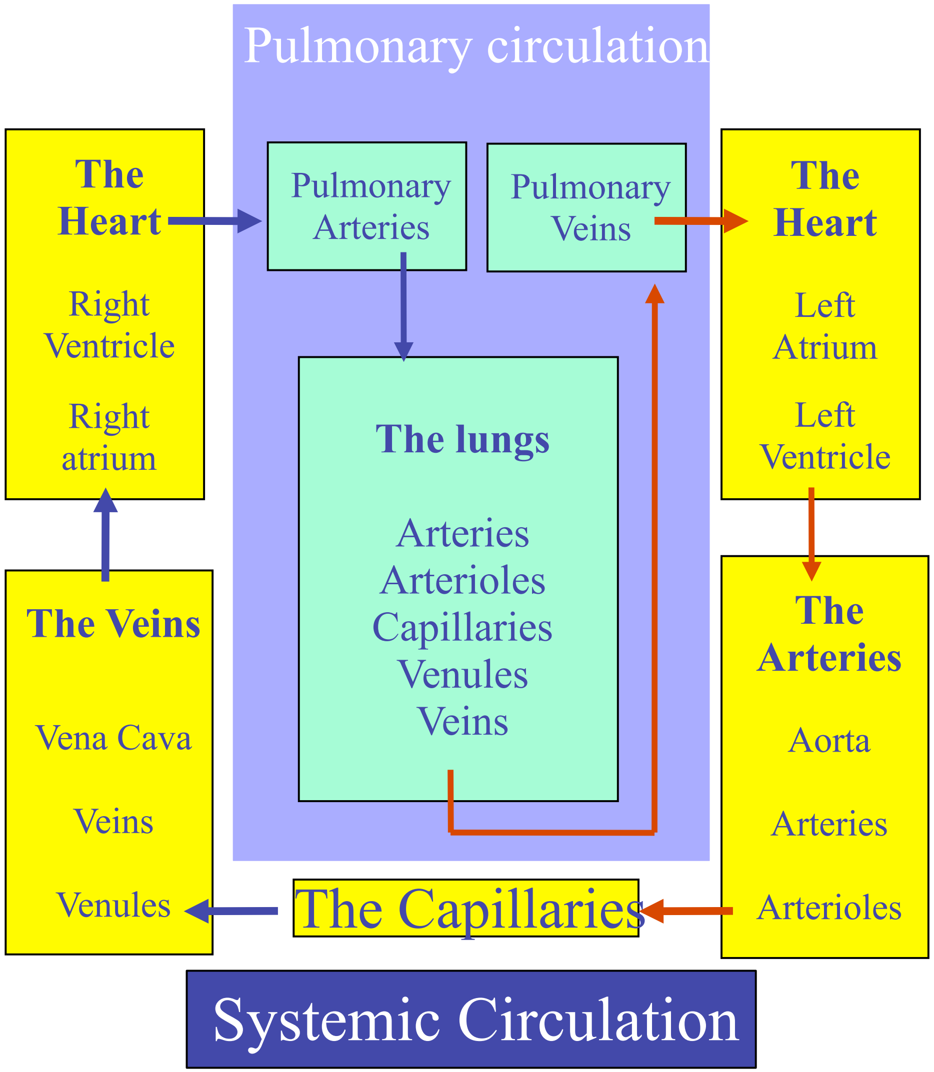
$$\widehat{\mathbf{F}}_s \widehat{\Sigma} \widehat{\mathbf{n}}_s = \widehat{J}_s \widehat{\sigma}_f(\mathbf{u}_f, P) \widehat{\mathbf{F}}_s^{-T} \widehat{\mathbf{n}}_s, \quad \text{on} \quad \widehat{\Gamma}$$

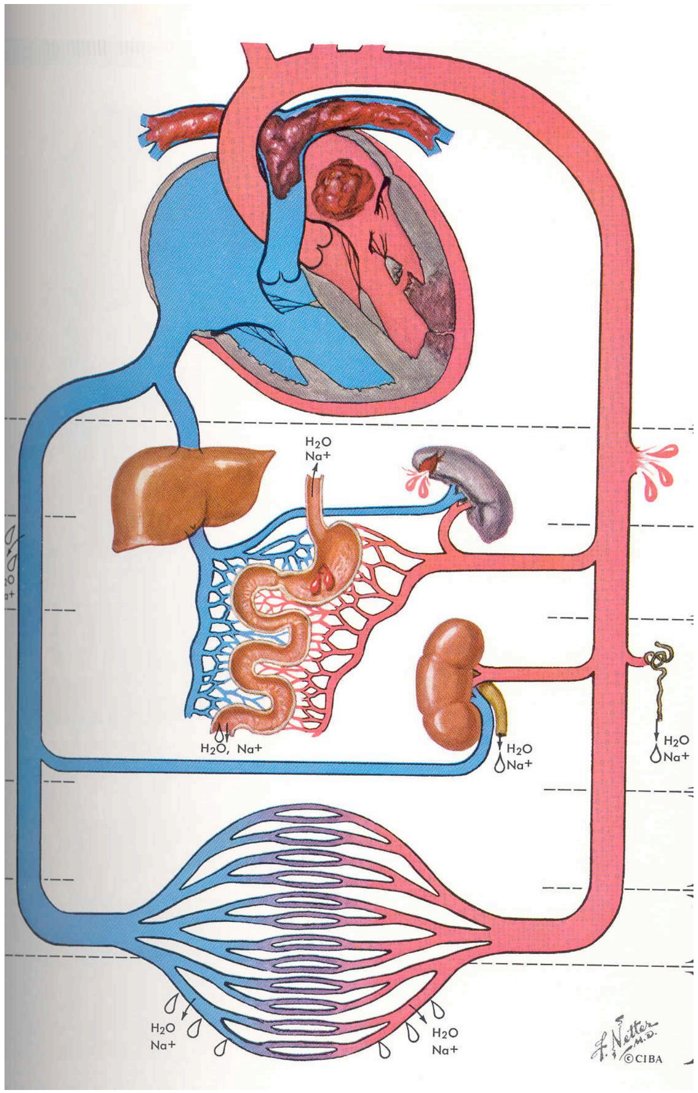


Mathematical Issues for Blood-flow Modelling

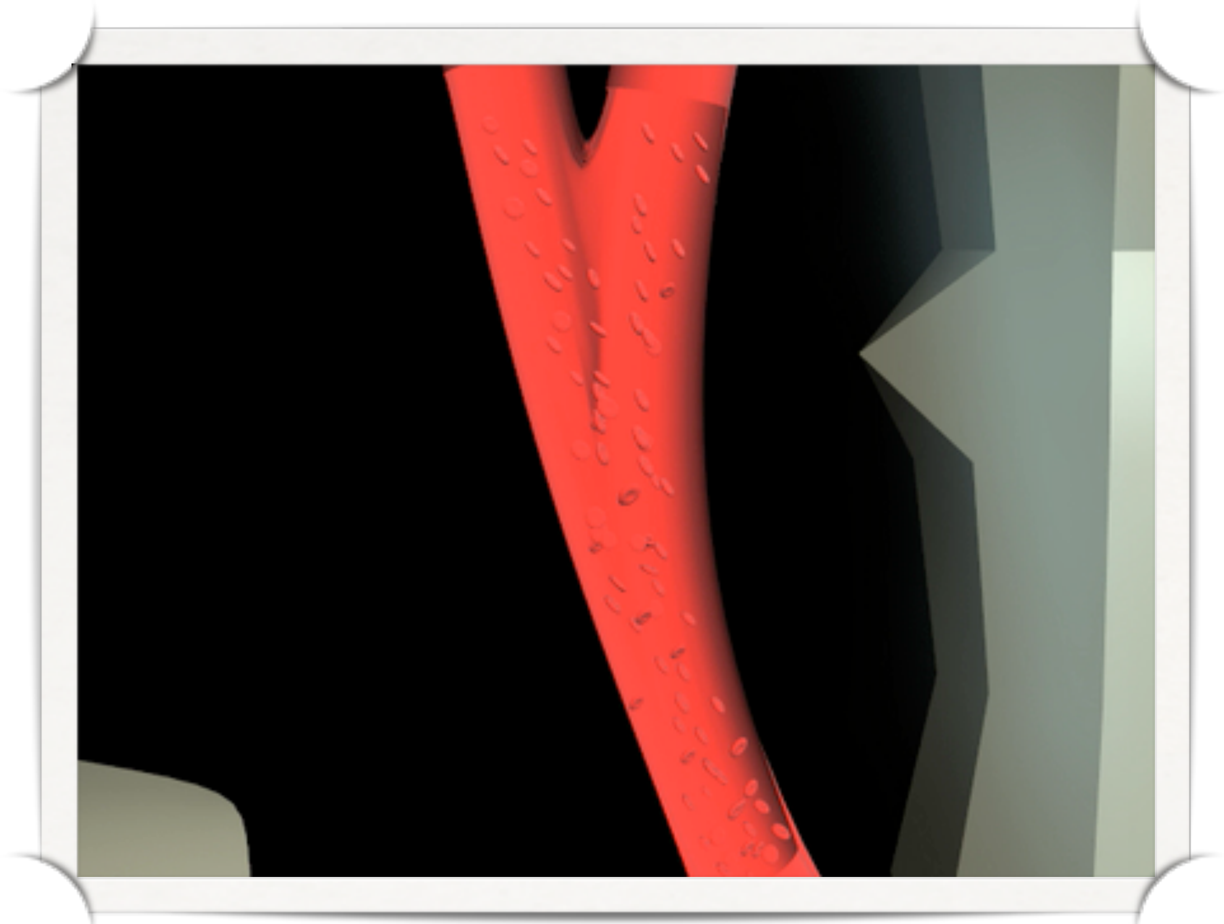
- *Defective Boundary Conditions for Local Analysis.* Inflow-Outflow treatment of Navier-Stokes Equations
- *Shape Optimization of Prosthetic Devices.* Arterial or coronary by-pass grafts
- *Geometric Multiscale Models for Global Analysis.* Mathematical interplay between 3D, 1D and 0D models
- *Development of Fluid Structure Algorithms.* Parallel preconditioners for Fluid-Solid coupling
- *Development of* ad hoc *Preconditioners.* Parallel preconditioners for Fluid or Solid solvers
- *Modelling of the Biochemical-Mechanical-Flow Interaction.* Drug delivery stents

The Whole Picture





Geometric multiscaling in the circulatory system



Local (level1): 3D flow model (FSI between NS equations and elastodynamic eqs)

Global (level 2): 1D network of major arteries and veins (Euler hyperbolic system)

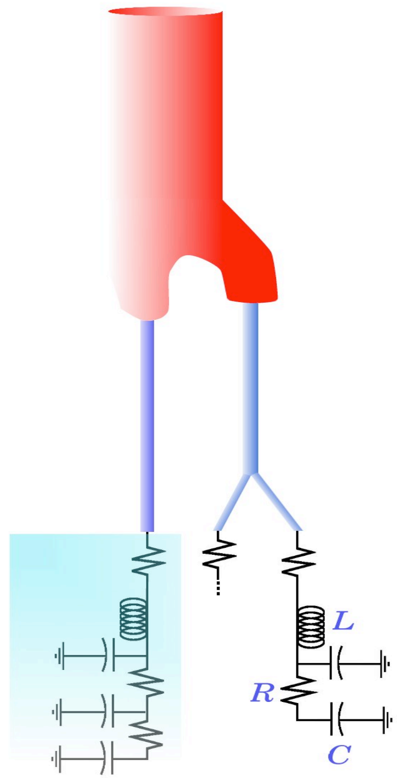
Global (level 3): 0D capillary network (DAE system)

Geometric multiscaling in the circulatory system

3*D*

1*D*

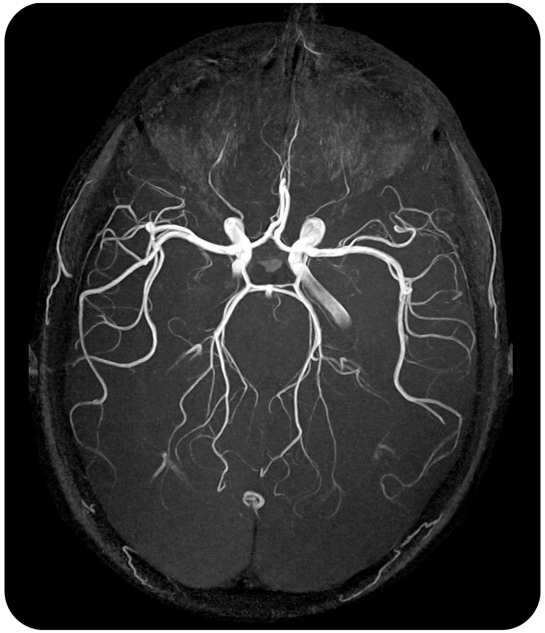
0*D*

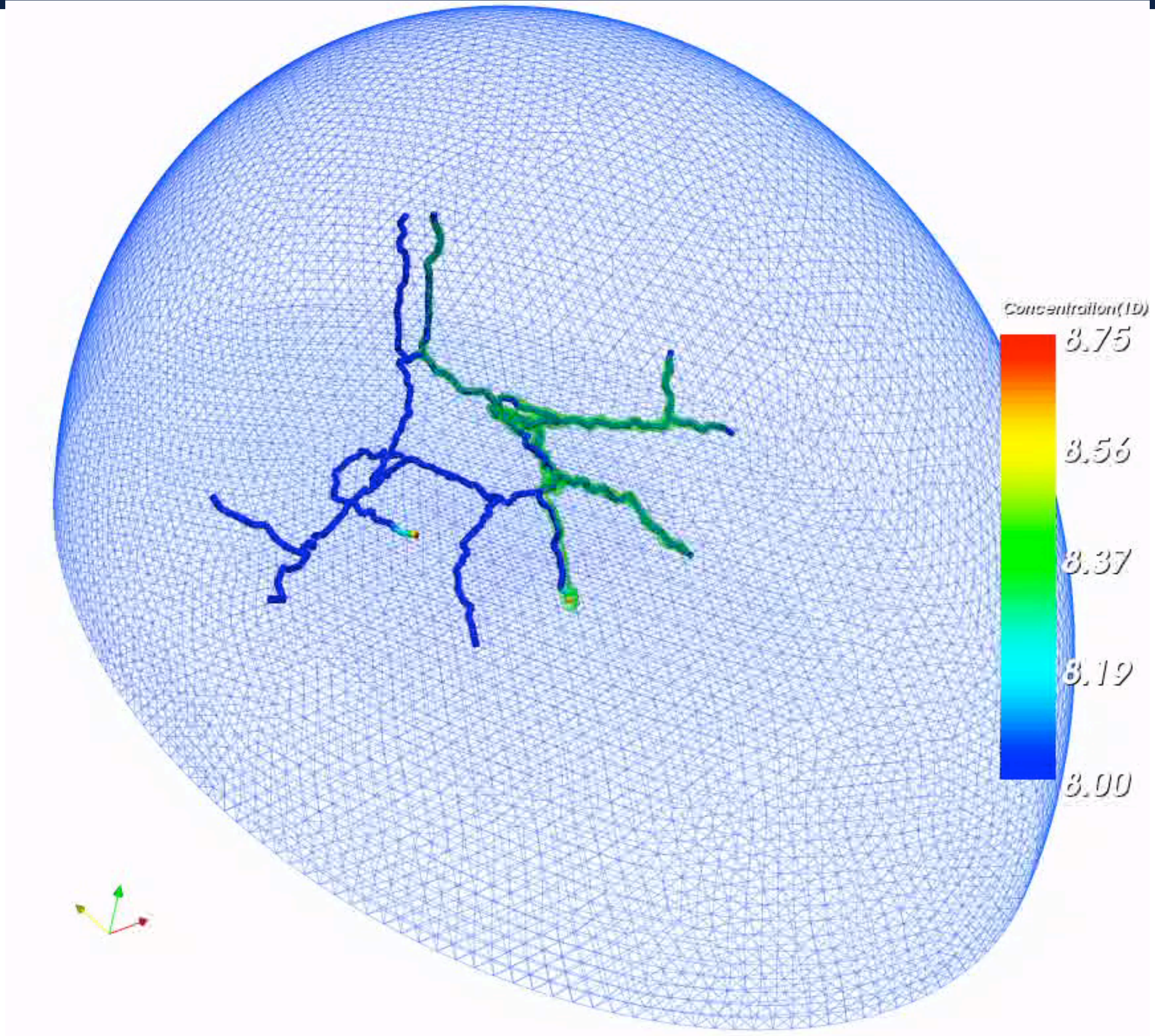


3D Navier-Stokes (F)
+
3D ElastoDynamics (V-W)

1D Euler (F)
+
Algebraic pressure law

0D lumped parameters
(system of linear ODEs)





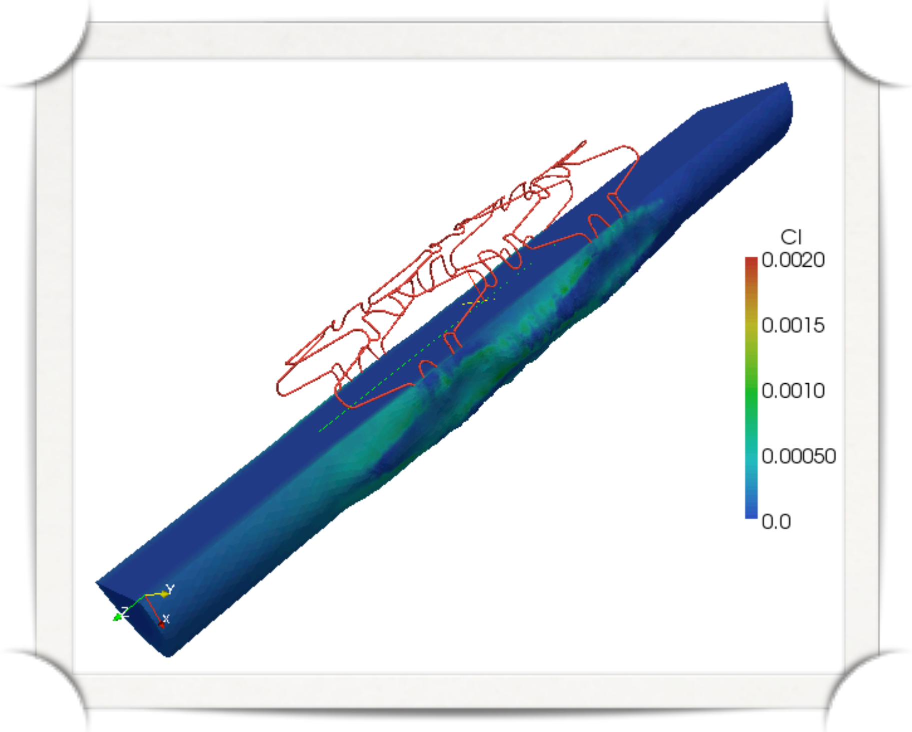
Application to oxygen transport in the brain:

isosurface of oxygen concentration.

Impairment due to left carotid artery occlusion.

(Simulations by C.D'Angelo)

Reduced Models for Drug Delivery



Stent struts modeled as 1D objects

Real geometries are easily handled since refining the meshes near the struts is no more required

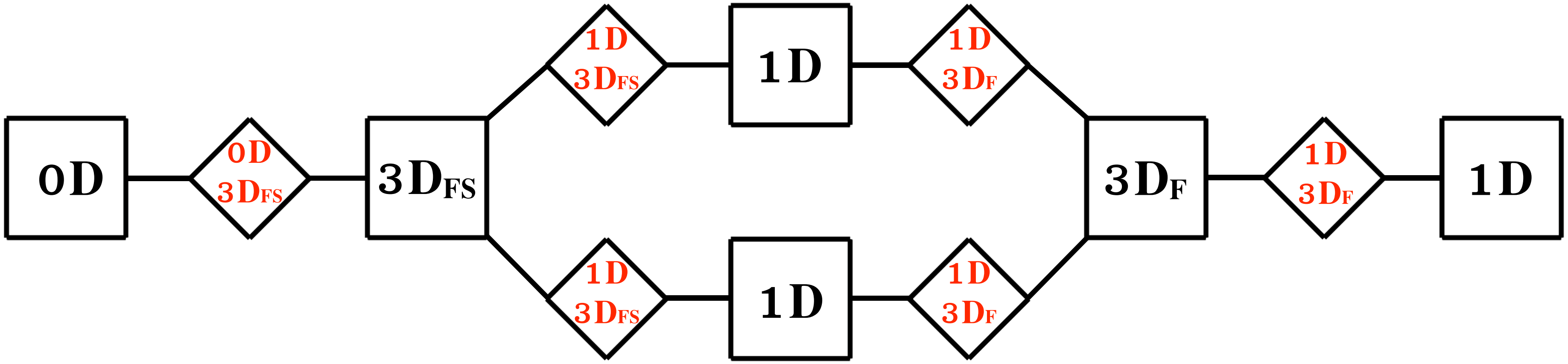
Mass fluxes are regarded as sources for the 3D problems

Isosurfaces of drug concentration (lumen)

The Mathematical Model

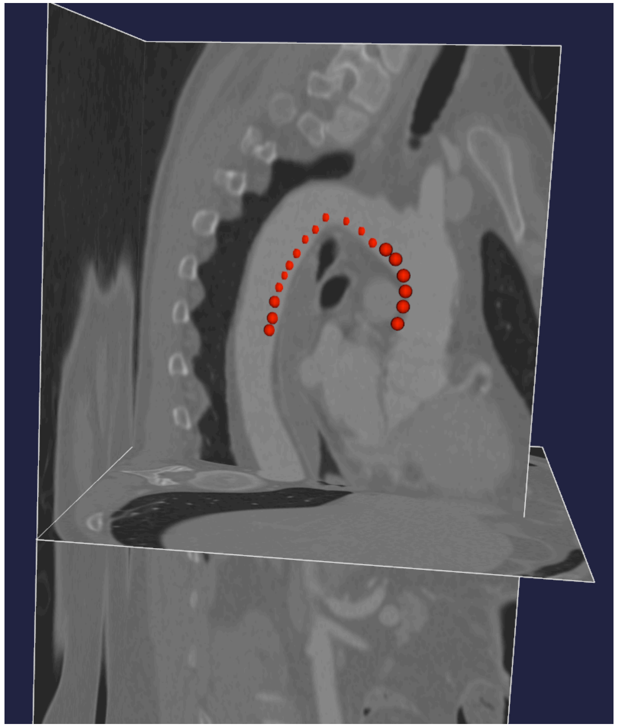
A suitable way to deal effectively with such an abstract problem is to keep the same level of abstraction in the code.

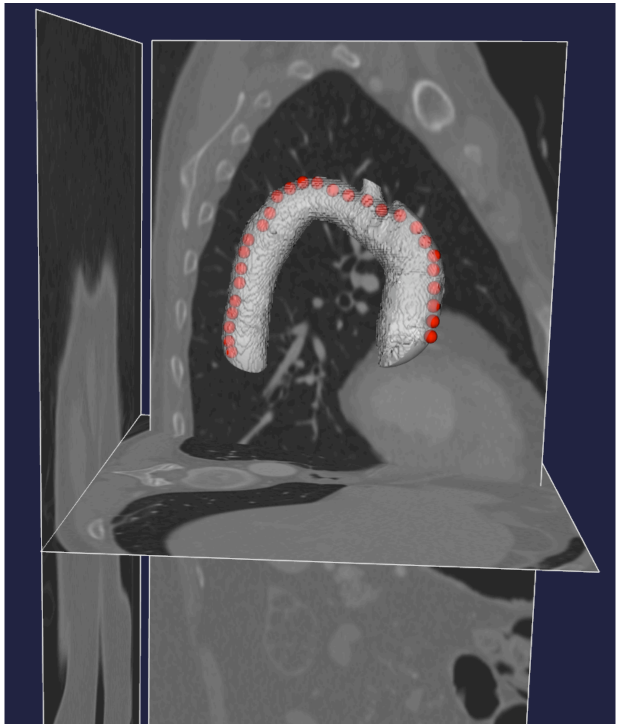
The idea is to build a Multiscale Multiblock Solver which makes use of common interfaces to operate on any specific model, coupling conditions, and algorithms.

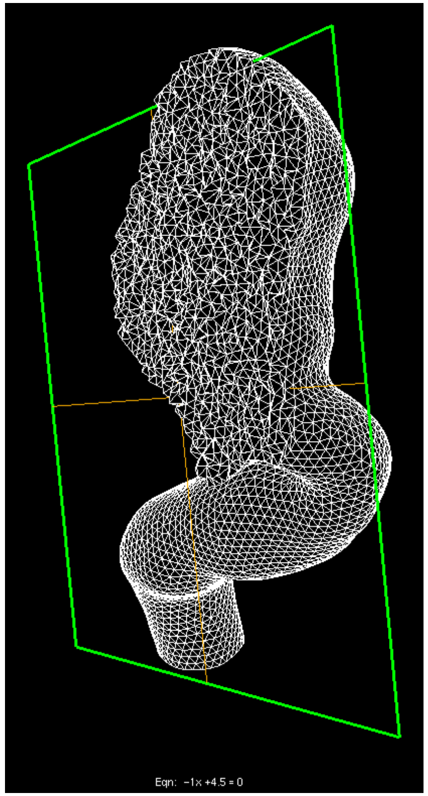


From Medical Images to Grid Generation

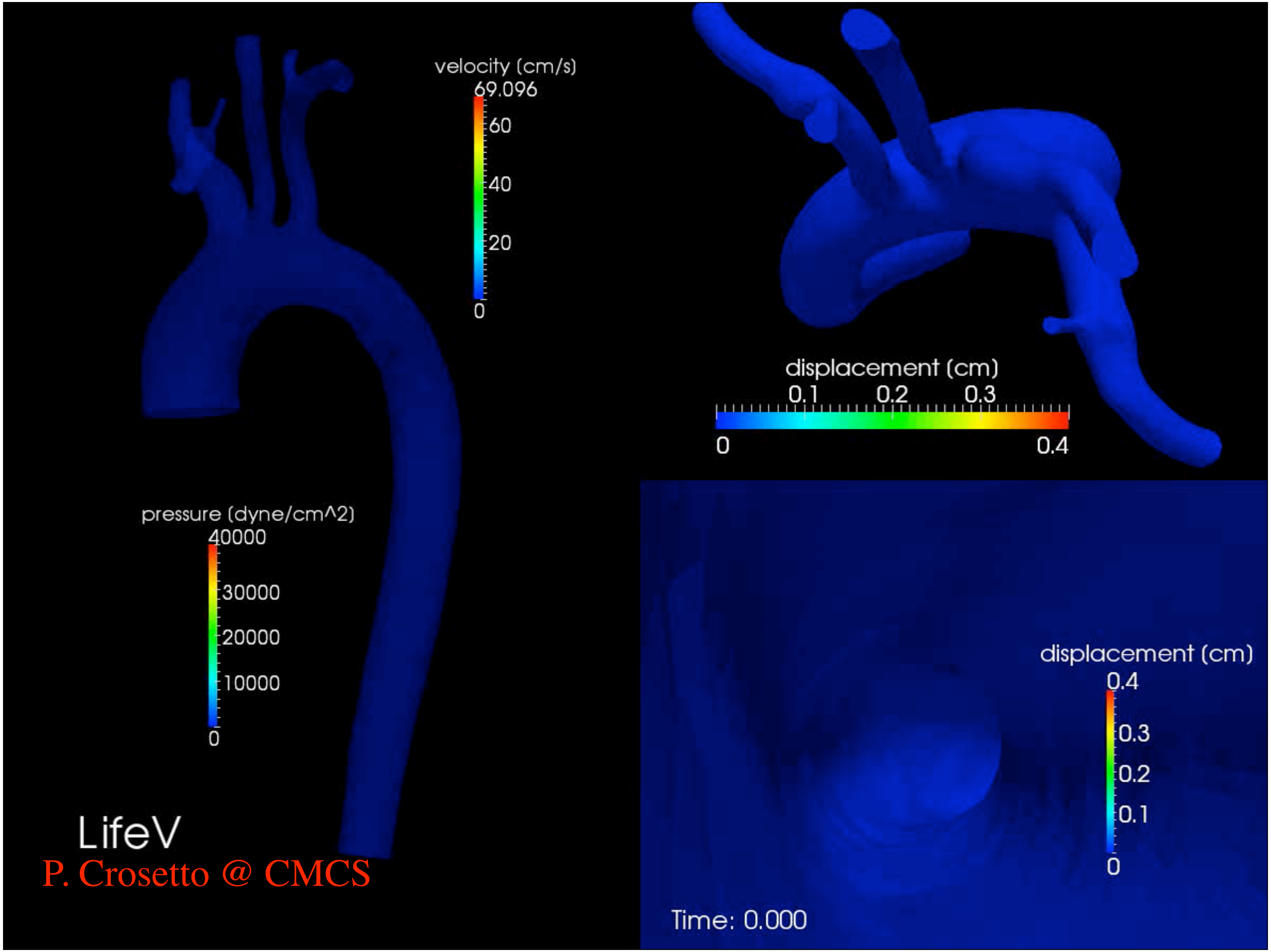
DICOM image → segmentation → unstructured mesh
(e.g. CT scan) (e.g. vmtk, 3DSlicer) (e.g. gmsh, netgen)



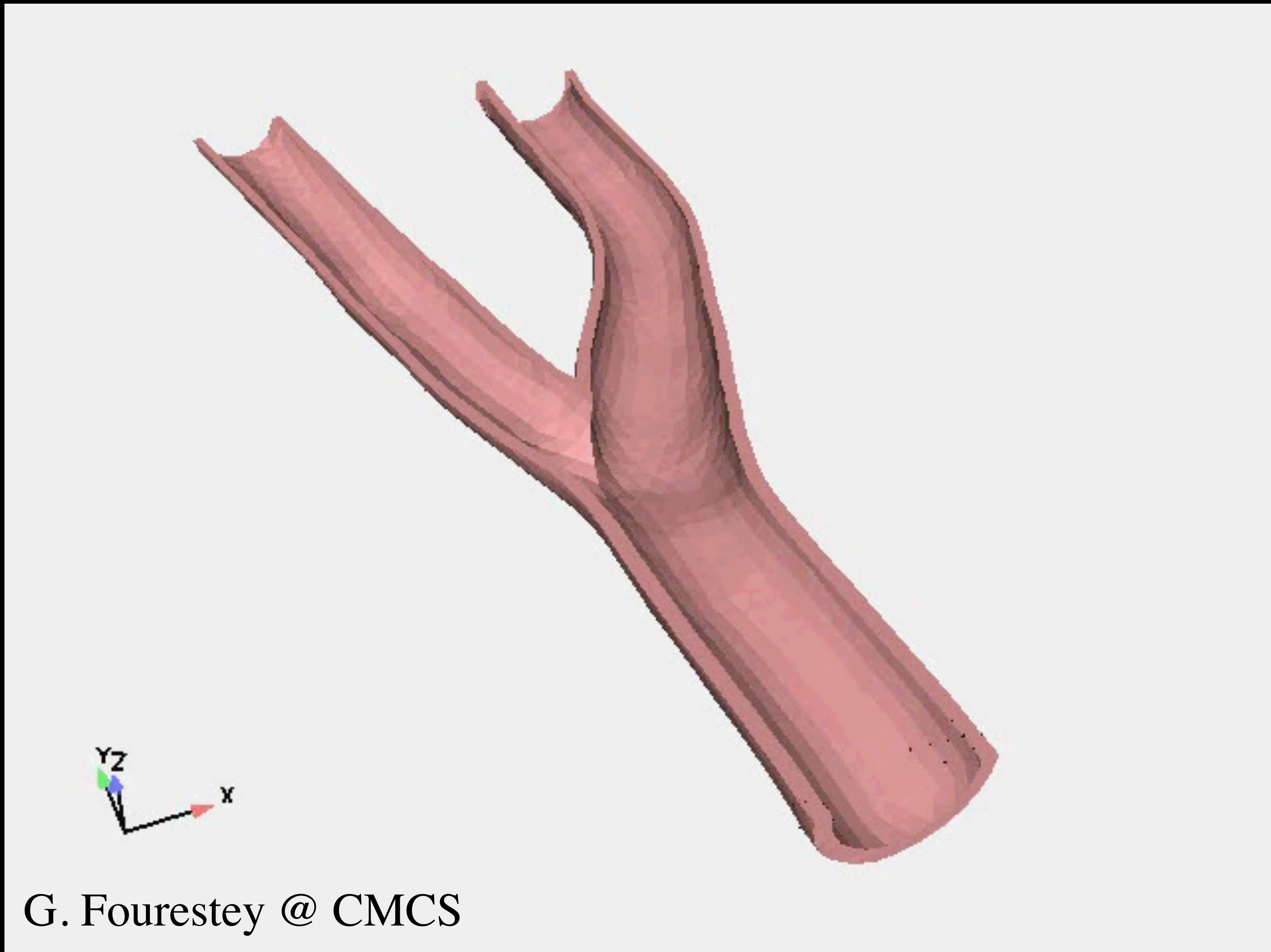




FSI, Navier-Stokes → Finite Elements → Solvers
(e.g. P2,P1,...) (e.g. LifeV, Trilinos)


velocity (cm/s)
69.096
60
40
20
0
displacement (cm)
0.1
0.2
0.3
0
0.4
pressure (dyne/cm^2)
40000
30000
20000
10000
0
displacement (cm)
0.4
0.3
0.2
0.1
0
LifeV
P. Crosetto @ CMCS
Time: 0.000

FSI for carotid bifurcation : wall deformation



FSI: Choice of the Preconditioner

Block Gauss-Seidel (Dirichlet-Neumann) preconditioner for the coupled system:

$$\underbrace{\left(\begin{array}{cc|cc|c} C_{ff} & C_{f\Gamma} & 0 & 0 & 0 \\ C_{\Gamma f} & C_{\Gamma\Gamma} & 0 & 0 & I \\ \hline 0 & 0 & N_s & N_{s\Gamma} & 0 \\ 0 & 0 & N_{\Gamma s} & N_{\Gamma\Gamma} & -I \\ \hline 0 & I & 0 & -\Delta_t & 0 \end{array}\right)}_{A} \rightarrow \underbrace{\left(\begin{array}{cc|cc|c} C_{ff} & C_{f\Gamma} & 0 & 0 & 0 \\ C_{\Gamma f} & C_{\Gamma\Gamma} & 0 & 0 & I \\ \hline 0 & 0 & N_s & N_{s\Gamma} & 0 \\ 0 & 0 & N_{\Gamma s} & N_{\Gamma\Gamma} & 0 \\ \hline 0 & I & 0 & -\Delta_t & 0 \end{array}\right)}_{P_{DN}}$$

Remarks on P_DN :

— coupling partly neglected;

\+ memory usage;

\+ modularity

- can be split in a fluid and a solid parts
- specialized preconditioners for fluid and structure

An Inexact Factorized D-N Preconditioner

Note that $P_{DN} =$

$$\left(\begin{array}{cc|cc|c} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & N_s & N_{s\Gamma} & 0 \\ 0 & 0 & N_{\Gamma s} & N_{\Gamma\Gamma} & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right) \cdot \left(\begin{array}{cc|cc|c} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & \Delta_t & I \end{array}\right)^{-1} \cdot \left(\begin{array}{cc|cc|c} C_{ff} & C_{f\Gamma} & 0 & 0 & 0 \\ C_{\Gamma f} & C_{\Gamma\Gamma} & 0 & 0 & I \\ \hline 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ \hline 0 & I & 0 & 0 & 0 \end{array}\right)$$

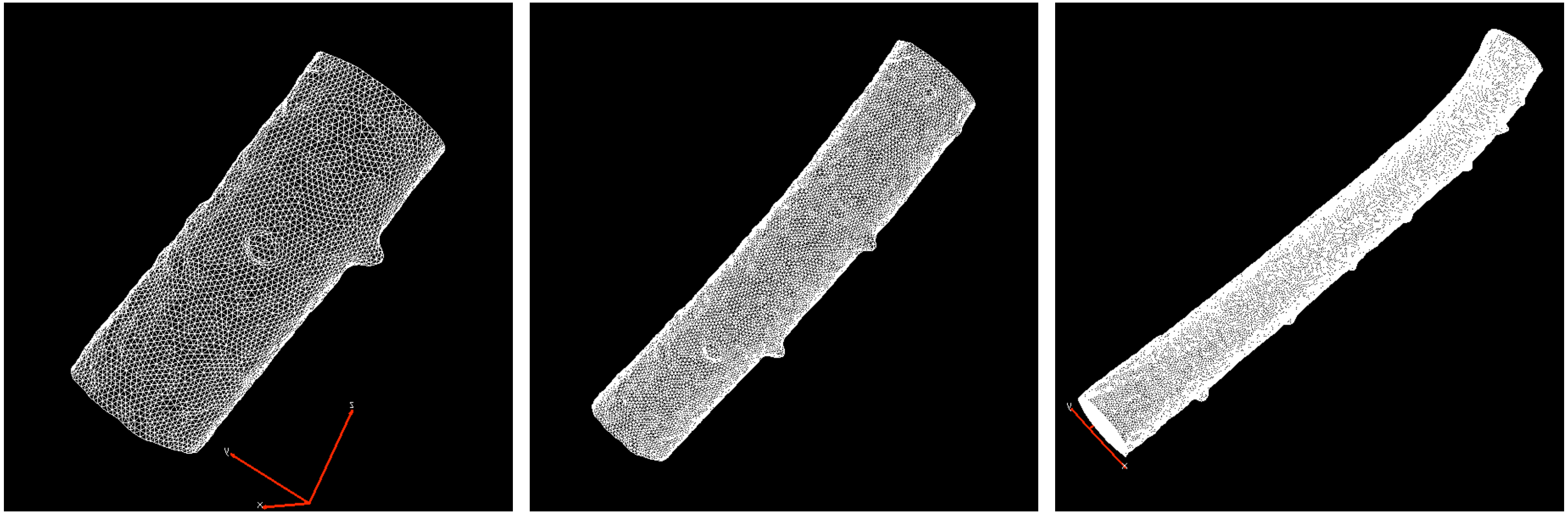
$$\begin{array}{ccc} P_1 & Q^{-1} & P_2 \\ \Downarrow & & \Downarrow \\ O_{P_1} & \longleftarrow \text{ overlap. Schwarz prec. } \longrightarrow & O_{P_2} \end{array}$$

Then approximate P_{DN} by

$$P_{DN}^{-1} \approx O_{P_2}^{-1} \cdot Q \cdot O_{P_1}^{-1} = \widehat{P}_{DN}{}^{-1}$$

O_{P_1} and O_{P_1} are overlapping Schwarz preconditioners of P_1 and P_2.
The condition number of the preconditioned matrix is governed by those of the local preconditioned problems [Crosetto, SD, Fourestey and Quarteroni 2010].

Navier-Stokes Solver, Weak Scalability (on BG/P)



	128 CPUs	256 CPUs	512 CPUs	W. Scalability	W. Scalability
Total dofs.	1'811'485	3'708'983	8'033'459	256/128	512/256
Tot.dofs/Num.procs	14'152	14'488	15'690	1.02	1.08
Matrix assembly	5.12	5.19	5.74	1.01	1.10
Stab. terms comp.	8.98	9.23	9.95	1.03	1.08
Prec	12.74	15.07	14.71	1.18	0.97
Sol time p.t.s.	5.96	6.27	6.13	1.05	0.97
Total time	31.32	33.64	34.11	1.07	1.02

TABLE: Wall-time (in sec) versus number of CPUs. 15'000 Dofs per CPU

FSI, Strong Scalability on XT4 and BG/P

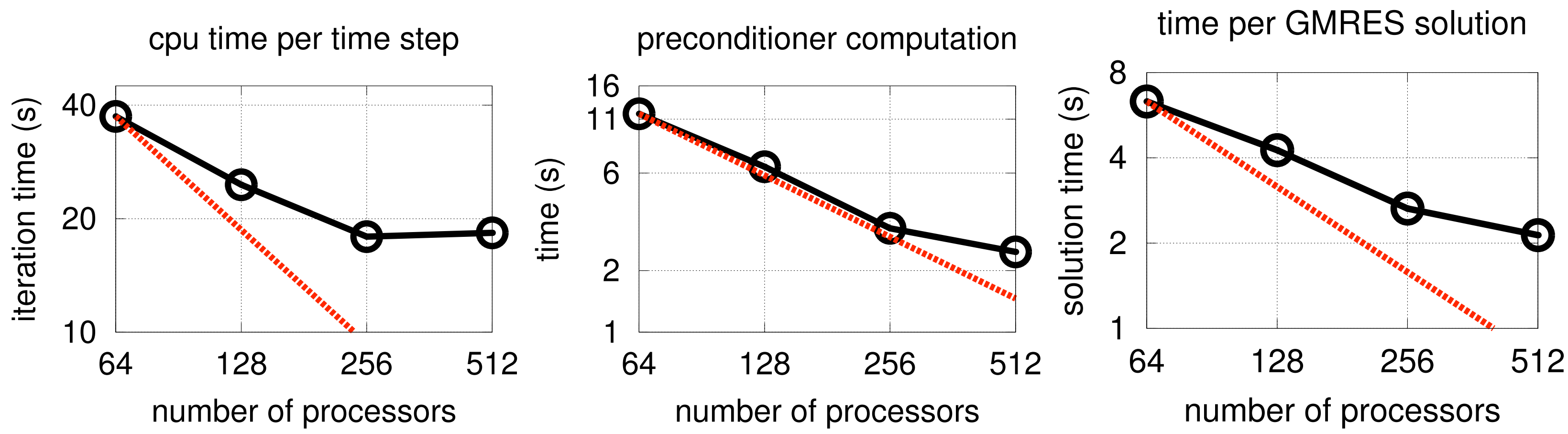
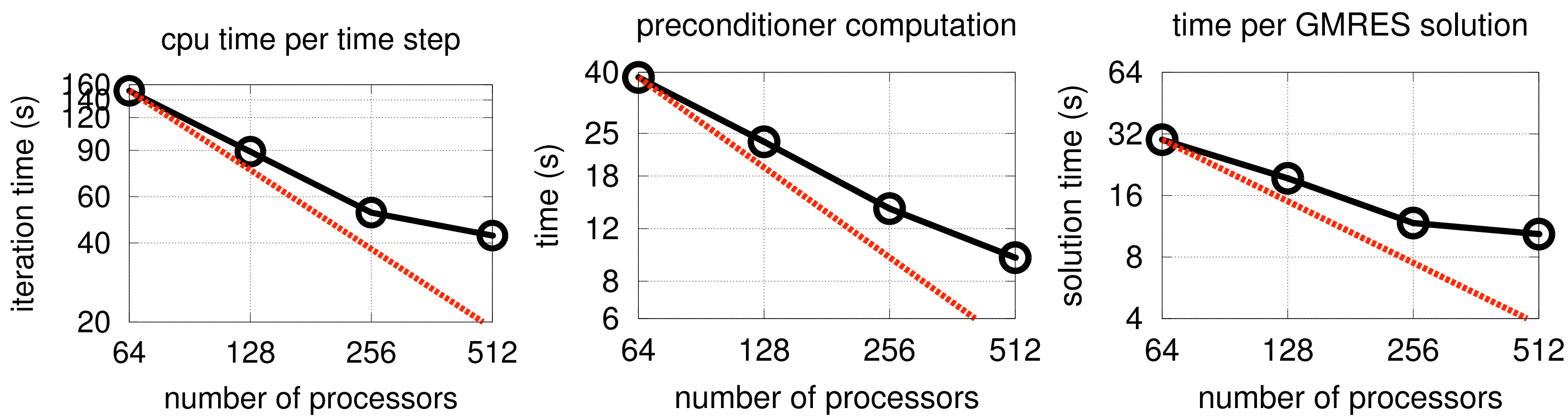


FIGURE: GCE time discretization, stabilized P1-P1 FE for fluid sub-problem, results on Cray XT4 (above) and IBM BlueGene/P (below).



578594 tetrahedra and 630468 dofs. $P_{GS-AS} = \widehat{P}_{DN}$

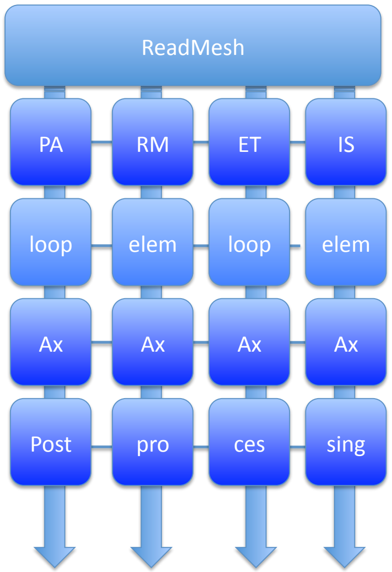
LifeV Project

LifeV Serial:

- C++ FE library,
- CMCS/MOX/REO,
- Aztec,
- efficient math. core,
- Poor overall design.

LifeV Parallel:

- CMCS@EPFL,
- BOOST, MPI, ParMetis, Trilinos,
- same math. core,
- enchanced design (HPC).



- All processors read the same mesh,
- ParMetis partitions the mesh,
- FE physical solver: loop on local mesh,
- Linear solver in parallel,
- post-processing on hdf5 or separate ensight files.

Challenges in HPC

Huge computational power is needed:

- complex and large geometries (e.g full aorta + iliac),
- several heartbeats to ensure the elimination of the initial transient.

Full body description (Mathcard): 3D/1D/0D network

- geometrical descriptions of solvers and connections,
- abstract classes for physical solvers with common interface.

HPC Multiphysics and Multiscale coupling (HP2C):

- very different order of magnitude for the dofs (10^2 for 1D models, > 10^6 for FSI),
- monolithic vs segregated treatment (FSI + 1D + ...),
- Implicit coupling (Newton).

Load balancing: coupling with physical phenomena that have different time and length scales:

- multiphysics models for more realistic problems resolution,
- multiscale models for realistic boundary conditions and general arterial tree description.

Programming: our wish list

- Parallel scalable preconditioners, in particular for saddle-points problems (Navier-Stokes) and finite elements,

- Customization of the numerical solvers/preconditioners in LifeV and Trilinos (preconditioning, matrix operations, ...) with architecture specific optimization to reach maximum performance:
    - chosing compilers and compiling options, architecture specific librairies (such as Trilinos on Cray, ESSL and MASS on BlueGene, ...),
    - CUDA/openCL, pthread, openMP, architecture specific MPI implementations,

- Class abstraction to use specific hardware architecture, where needed at runtime.