LifeV: applications, design, and parallel framework

Simone Deparis
CMCS–MATHICSE–SB–EPFL
EPFL Lausanne, Politecnico di Milano, INRIA, Emory University

Trilinos User Group meeting, SANDIA, November 1st 2011

Acknowledgements

LifeV community http://www.lifev.org
Trilinos community http://www.trilinos.org
EPFL, Politecnico di Milano, INRIA, Emory University
FNS, FP6 (Haemodel), FP7 - ERC-AdvG (Mathcard), HP2C,
Aneurisk, ...

LifeV project

Finite Element Library for the solution of PDEs

- C++, object oriented
- parallel and serial versions
- distributed under LGPL
- about 30 active developers

- CMCS – EPFL
- E(CM)^2 – Emory
- MOX – Polimi
- REO, ESTIME– INRIA

Research code oriented to the development and test of new numerical methods and algorithms
Aim: effective tool for solving complex "real-life" engineering problems

History

- **<2001** LifeI, LifeII, LifeIII: Fortran 2D finite element codes (CRS4, Politecnico di Milano, CMCS)
- **2001** early development of LifeV at Politecnico di Milano, INRIA, and EPFL (leaders A. Quarteroni, L. Formaggia, A. Veneziani, J.F. Gerbau)
- **2006** development of the parallelism in LifeV. Inclusion of Trilinos (G. Fourestey, S. Deparis).
- **2008** $E(CM)^2$ – Emory University joins the project

Developers and target users

Research code oriented to the development and test of new numerical methods and algorithms

Developers are:

- Researchers
- Post docs
- PhD students

Target users are the developers themselves, master students, and other researchers.

Aim: effective tool for solving complex "real-life" engineering problems

Medium term target is to make LifeV more accessible to external users.

Development tools

- Autotools, Automake — transition to CMake
- nightly builds in opt and debug modes
- collaborative tool: Git
- Gforge system `cmcsforge.epfl.ch`,
- Google groups for mailing lists
- Google app educational for `lifev.org`

Why Trilinos (extract form Trilinos survey)

Were there alternatives to Trilinos that you investigated? If so, what made you decide to use Trilinos?

Yes, the most competitive alternative was petsc. However we found Trilinos more mature and with a larger set of tools (actually packages). Trilinos allowed us a smooth switch to MPI and offers interfaces to this API. It was also important that the source is written in c++ and that the following tools are provided:

- distributed sparse matrix data-structures
- linear system solvers and preconditioners
- parallel IO
- generalized eigenvalue solvers.

Trilinos' most used features

Tools	Epetra stack	Tpetra stack
Parallel linear algebra	Epetra	Tpetra/Kokkos
Matrix tools and extensions	EpetraExt	Teuchos
Graph operations	Zoltan & Isorropia	Zoltan2
Iterative linear solver	AztecOO	Belos
Direct solvers	Amesos	Amesos2
DD preconditioners	Ifpack	Ifpack2
Multi-level preconditioners	ML	Muelu

An eventual migration of LifeV to the templated package stack of **Trilinos** is desired.

Overview of sofware design

Overview of sofware design

Boundary Conditions

LifeV provides classes and methods to prescribe the most common boundary conditions *Dirichlet, Neumann, Robin, Defective* (flow rates).

Normal and tangential boundary conditions are also allowed for vectorial (3D) fields. In addition, a few kinds of boundary conditions specific for flow problems are available (absorbing, resistance or lumped parameter models).

Boundary conditions are prescribed by modifying the system matrix and the right-hand side after the assembling phase.
Parameters: functions of the space and time or finite element fields.

Time discretization I

LifeV provides the class `TimeAdvance` to integrate in time a generic non-linear PDE with derivates in time of order $m = 1, 2$. This class is virtual and defines the main features of a generic time advancing scheme:

- stores the unknown and its first (and second) derivatives;
- provides methods for the extrapolation (in time) of the unknown (and its first derivative);
- it computes the right-hand side associated to the discretization of the first (and second) derivative;
- it provides methods to update the states of the stored unknown.

The needed coefficients, variables and methods are specified in derived classes.

Time discretization II

LifeV provides two different implementations, namely:

- *Backward Differentiation Formulae (BDF)* schemes (`TimeAdvanceBDF.hpp`) of order $p \leq 5$ for $m = 1, 2$;
- the family of methods obtained from the *Newmark* schemes for second order problems ($m = 2$), and the *theta-methods* for the first order problems ($m = 1$) (`TimeAdvanceNewmark.hpp`).

Space discretization

LifeV provides classes to quickly and easily perform the assembly of the Finite Element system matrix.

Assemblers are building blocks that can be combined at will. In addition, the user has access to lower level structures and instructions in order to provide the maximal flexibility.

“Classical” FE implemented: continuous P1, P2, P1 bubble Raviart-Thomas

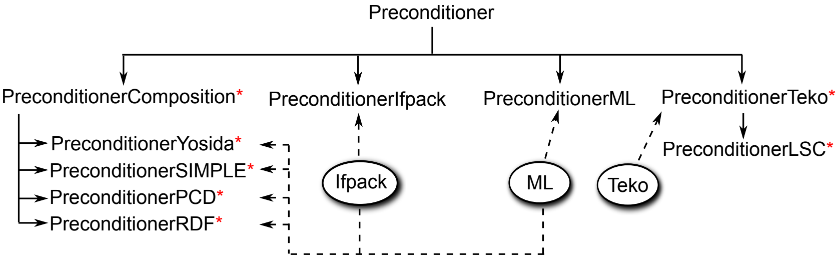
Tools

- **Importers and Exporters:** Mesh format: GMSH, Inria, Netgen, mesh++.
  LifeV is designed with large parallel simulations in mind, which are usually performed on HPC hardware.
  LifeV allows to perform the mesh partitioning phase online, in parallel, or offline, on a workstation

  For postprocessing, class `Exporter.hpp`:
  - Ensight (`ExporterEnsight.hpp`);
  - HDF5 (`ExporterHDF5.hpp`);
  - VTK (`ExporterVTK.hpp`).

  We rely on ParaView for creating high quality visualizations.

Preconditioners in LifeV
with G. Grandperrin



Overview of the preconditioners in LifeV

E.g., Navier-Stokes equations

The Navier-Stokes equations for an incompressible viscous flow reads:

$$\begin{array}{rcll} \frac{\partial}{\partial t}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p & = & \mathbf{f} & \text{in } \Omega \\ \nabla\cdot\mathbf{u} & = & 0 & \text{in } \Omega \\ \mathbf{u} & = & \varphi & \text{on } \Gamma_D \\ \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\mathbf{n} & = & 0 & \text{on } \Gamma_N \end{array}$$

where Γ_D and Γ_N are the Dirichlet and Neumann parts of the boundary respectively, $\mathbf{u}$ is the fluid velocity, p the pressure, ν the kinematic viscosity of the fluid, and $\mathbf{f}$ the external forces.

Discretization

E.g. with semi-implicit Euler scheme:

$$
\begin{aligned}
\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1} - \nu \Delta \mathbf{u}^{n+1} + \nabla p^{n+1} &= \mathbf{f} && \text{in } \Omega \\
\nabla \cdot \mathbf{u}^{n+1} &= 0 && \text{in } \Omega \\
\mathbf{u}^{n+1} &= \varphi && \text{on } \Gamma_D \\
\nu \frac{\partial \mathbf{u}^{n+1}}{\partial \mathbf{n}} - p^{n+1}\mathbf{n} &= 0 && \text{on } \Gamma_N
\end{aligned}
$$

FE discretization using $\mathbb{P}_2 - \mathbb{P}_1$ finite elements:

$$
\begin{pmatrix} F(\mathbf{U}^n) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{U}^{n+1} \\ \mathbf{P}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^n(\mathbf{U}^n) \\ \mathbf{0} \end{pmatrix}
$$

Preconditioning strategy

The `PreconditionerComposition` class exploits the block structure of the FE matrix A to create preconditioners.

$$A = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}$$

The class is able to

- manage composition of operators obtained by factorizing the matrix A.
- replace the inverses of operators by embedded preconditioners (e.g. ML, AAS).

Preconditioning strategy

Let use consider the example of the Pressure Convection-Diffusion (PCD) preconditioner:

$$P = \begin{pmatrix} F & B^T \\ 0 & -\hat{S} \end{pmatrix}$$

where

$$\hat{S} \cong A_p F_p^{-1} M_p$$

is an approximation of the Schur complement $S = BF^{-1}B^T$.

Silvester, Elman, Kay, Wathen. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comput. Appl. Math., 2001.*

Kay, Loghin, Wathen. A preconditioner for the steady-state Navier-Stokes equations. SIAM J. Sci. Comput., 2002.

Preconditioning strategy

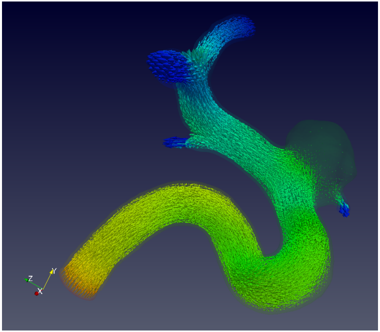
The PCD is implemented in LifeV as a child class of `PreconditionerComposition` class.

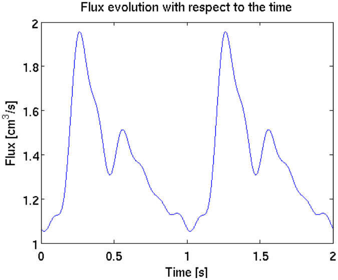
The PCD is factorized into five operators:

$$P^{-1} = \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -M_p^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & F_p \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A_p^{-1} \end{pmatrix}.$$

The inverses F^{-1}, M_p^{-1} and A_p^{-1} are replaced by embedded preconditioners.

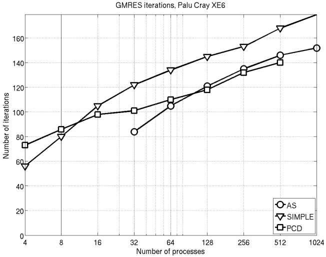
Blood-flow in rigid geometry



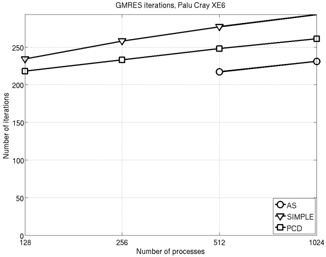


Baek, Jayaraman, Richardson, Karniadakis. Flow instability and wall shear stress variation in intracranial aneurysms.*J R Soc Interface*, 2010.

GMRES iterations



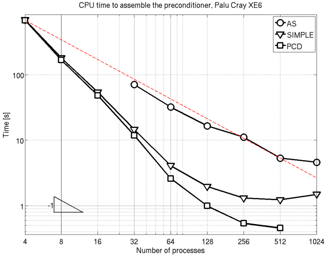
(a) Coarse mesh



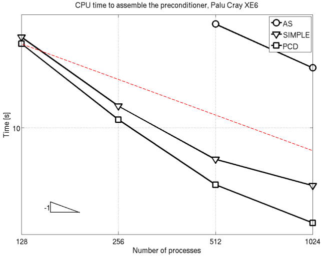
(b) Fine mesh

GMRES iterations

Computing the preconditioner



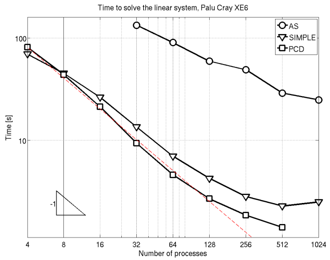
(c) Coarse mesh



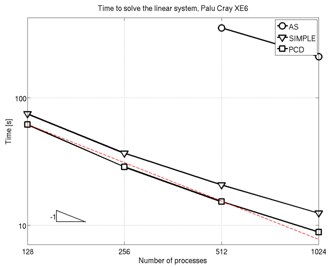
(d) Fine mesh

Strong scalability to assemble the preconditioners.
Palu@CSCS, Cray XE6.

Time to solve the linear system



(e) Coarse mesh



(f) Fine mesh

Strong scalability of the preconditioned iterations.
Palu@CSCS, Cray XE6.

Collaboration with Trilinos Developers

- The efficiency of the preconditioners inheriting the class `PreconditionerComposition` relies on the embedded preconditioner;
- Typical embedded preconditioners make intensive use of the ML and Ifpack packages in Trilinos;
- Collaboration with E. Cyr (Teko), J. Gaidamour, J. Hu, and C. Siefert (ML, mueLU): Experimentation of techniques for the preconditioners to achieve fast solve on parallel machines.

Parallel computing architectures
with R. Popescu

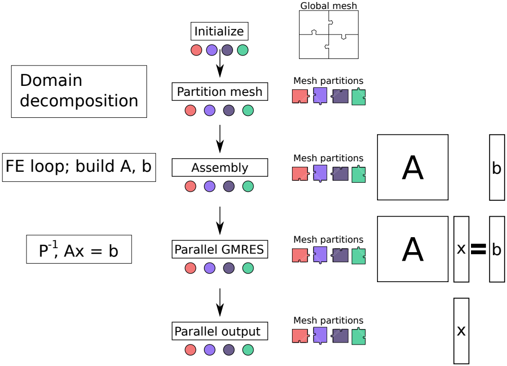
Hardware

- modern supercomputers or clusters are not *flat*
- a mix of distributed and shared memory parallel hardware
- non-uniform memory access (NUMA) node architecture is most common
- memory and interconnect bandwidth are at a premium

LifeV

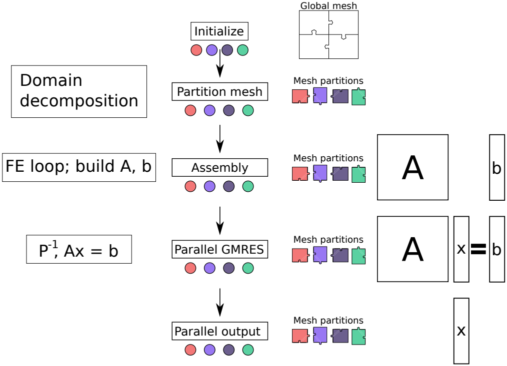
- parallelism through domain-decomposition (DD)
- still an MPI only implementation
- parallel operation is more or less hardware agnostic
- better use of hardware is possible with architecture aware algorithms

Parallel finite element loop



- Each stage could take advantage of hybrid parallel hardware:
  - parallel preprocessing on subdomains
  - hybrid parallel solvers and preconditioners
  - parallel and/or asynchronous I/O
- All stages of the simulation pipeline have to be upgraded, to maintain efficiency!

Parallel finite element loop



- Each stage could take advantage of hybrid parallel hardware:
  - parallel preprocessing on subdomains
  - hybrid parallel solvers and preconditioners
  - parallel and/or asynchronous I/O
- All stages of the simulation pipeline have to be upgraded, to maintain efficiency!

Parallel assembly of the linear system (Ongoing work)

- multi-threaded assembly is useful with a multi-threaded or hybrid parallel solver
- graph coloring on the mesh assures thread-safety (**Zoltan** package from **Trilinos**)
- this is a scalable solution inside a uniform memory region
- it's not scalable to an entire NUMA node

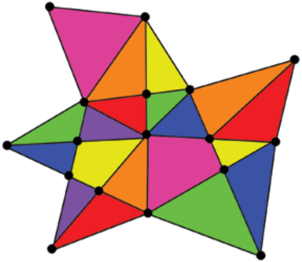


Figure: Example of coloring a 2D triangular mesh. Courtesy of Linnea Duvall

Parallel assembly performance

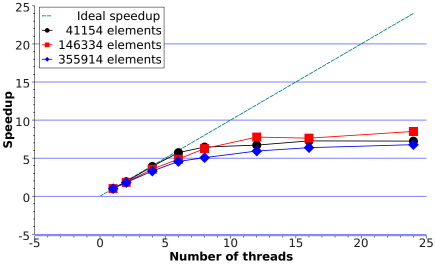


Figure: Speedup Cray XE6

NUMA 4 x 6 cores per node

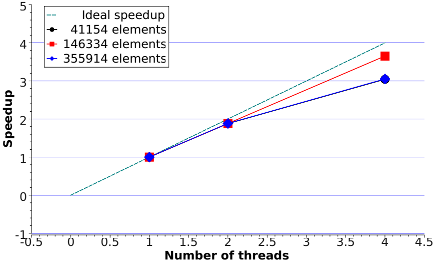


Figure: Speedup BG/P

UMA 4 cores per node

The Algebraic Additive Schwarz (AAS) preconditioner

$$P_{AS}^{-1} = \sum_{i=1}^{N} P_i A_i^{-1} R_i$$

R_i is the restriction operator, P_i is the prolongation operator, A_i^{-1} is an exact or inexact local solve on subdomain

- Number of GMRES iterations increases with the number of subdomains
- Using the implementation from **Ifpack**
- No parallel way to solve the local problems
- 1:1 relationship between number of subdomains and number of MPI ranks
  - Could get better scalability by unlocking these two numbers

Hybrid parallel AAS

ShyLU deveopers: S Rajamanickam, E. Boman, M. Heroux

- Flexible MPI subcommunicator functionality for **Ifpack** is under development
- It will be possible to use multiple MPI ranks inside a subdomain or span multiple compute nodes with a single subdomain
- It is designed to be used in conjuction with a parallel subdomain solver

ShyLU (**S**calable **hy**brid **LU**) package:

- developed at Sandia National Labs by Erik Boman and Siva Rajamanickam
- a hybrid subdomain solver:
  - hybrid parallel: uses MPI and multiple threads
  - hybrid algorithm: direct and iterative techniques
- **ShyLU** is still not ready for public use

Solvers and Applications

Physical solvers and applications

Darcy solver

The Darcy equations descibe a fluid flow through a porous medium. The implementation of the solver (`DarcySolver.hpp`) uses the dual-mixed-hybrid formulation, entailing good approximation of the velocity field as well as of the pressure field.
The global system in saddle point form is recast to an equivalent positive definite system, using the hybridization and static condensation procedure.

FE spaces: P1 for the pressure and low order Raviart-Thomas for the velocity.

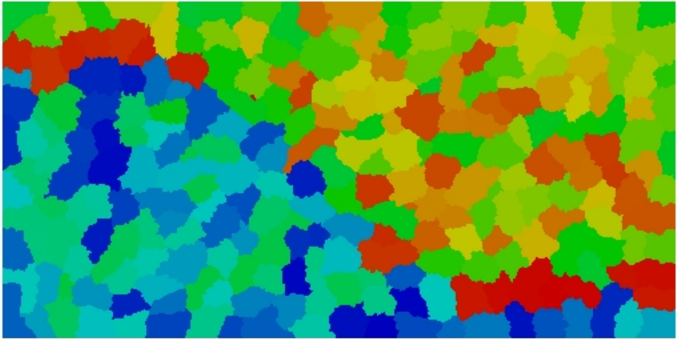
Darcy flow
A. Fumagalli and M. Kern

2 phase flow computation in a medium with a fault (i.e. highly heterogeneous). Water is injected at the top left, and displaces oil.

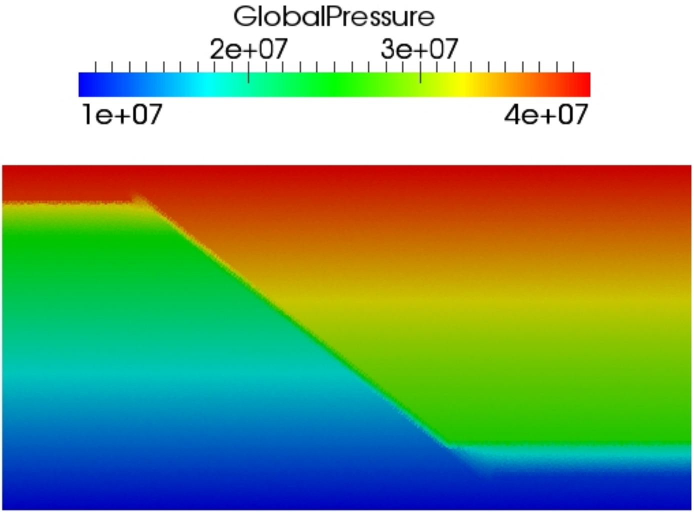
- fractional flow / global pressure formulation
- several 2D slices
- Darcy (parabolic): mixed finite elements
- Buckley Leverett (hyperbolic): Godunov solver

1.4 million tetra / 2.8 million dofs

64 cores, 25 time steps, 10 hours
256 cores, 195 time steps, 24 hours   SGI Altix, Genci, France.



Mesh partitioning

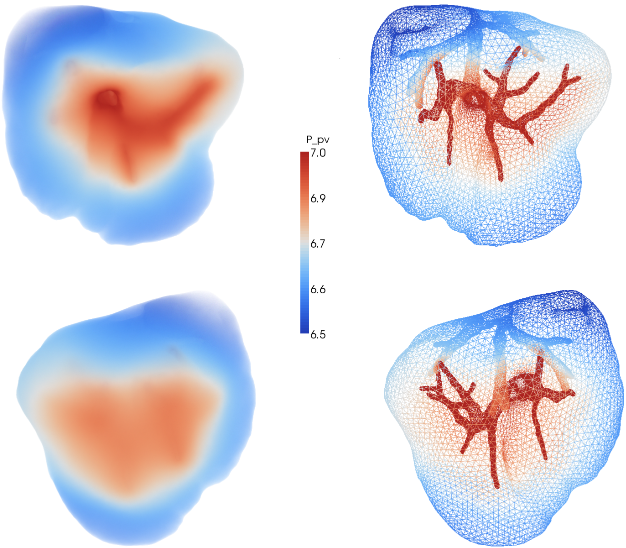


Pressure distribution

MOVIE

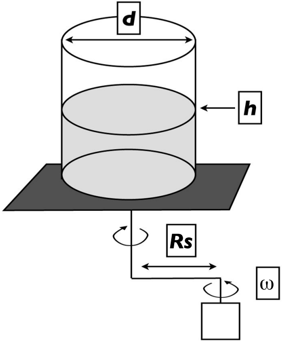
Three phase flow in the liver

M. Perego, M. Piccinelli e A. Veneziani



Cell culture in Orbitally Shaken Reactor

S. Quinodoz



Despite their *simple configuration*, OSRs represent *high efficiency* solution for cell culture. However, the mechanisms acting in OSRs are *poorly understood.*

LifeV is used to

- better understanding the hydrodynamics in OSRs
- study gas transfers (mainly of oxygen and of carbon dioxide)
- simulate the cell growth and eventual sedimentation

Free surface flow with level set
Breaking wave

MOVIE

StructuralSolver module

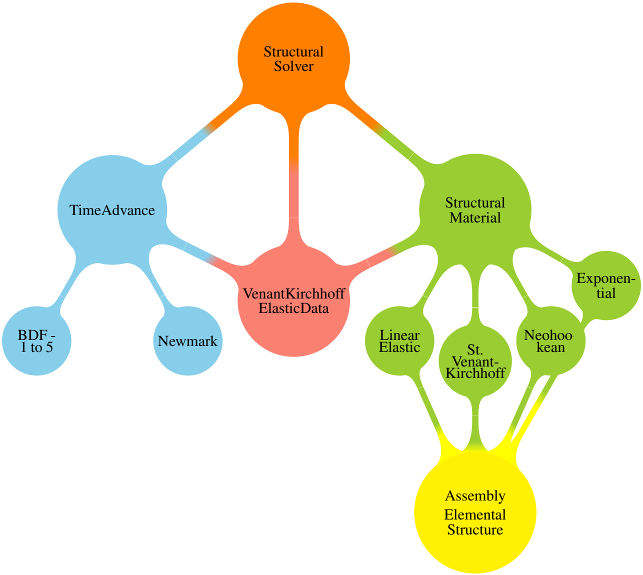
P. Tricerri, G. Mengaldo, P. Crosetto

Balance of forces acting on a body $\mathcal{B}_0$:

$$\rho_s \frac{\partial^2 \widehat{\boldsymbol{\eta}}}{\partial t^2} = \rho_s \mathbf{b} + \mathrm{Div}(\mathbf{P}) \qquad \forall \mathbf{X} \in \mathcal{B}_0 \times [0, T]. \tag{1}$$

- ρ_s is the solid density;
- $\widehat{\boldsymbol{\eta}}$ is the displacement field;
- $\mathbf{P}$ define the type of material is being modelled;
- $\mathbf{b}$ external loads;

The StructuralSolver module



Main parts of the module:

- *StructuralSolver*, for the Newton method to Pb. (1);
- *StructuralMaterial & AssemblyElementalStructure*, to define the constitutive relation in Pb. (1);
- *VenantKirchhoffElasticData*: to insert the material parameters;
- *TimeAdvance*: to carry out the time discretization of Pb. (1).

[*PhD thesis M-R. De Luca, MOX 2009*]

Electrical Activity in the Heart

Physiology The electrical signal dictates the contraction, the strain of the soft tissue activates ionic currents (electromechanical feedback).

Diseases Altered patterns of electrical signals may originate cardiac arhythmias, yielding an ineffective mechanical contraction and poor fluid ejection volume.

Unsatisfactory Diagnostics The mere analysis of an electrocardiogram pattern may not reveal pathologies of the inner dynamics of the voltage signal in the heart.

Heart electrophysiological solver

Both bidomain (`HeartBidomainSolver.hpp`) and monodomain (`HeartMonodomainSolver.hpp`) models are available as models for the electrophysiology behavior of cardiac tissue.
These models consist on anisotropic reaction-diffusion equations governing the propagation of electrical potentials, coupled with a system of ODEs describing the physics of the cellular membrane and time evolution of ionic quantities.

Several variants for membrane models are already present in the library: Luo-Rudy phase I, Rogers-McCulloch and Mitchell-Schaeffer models.

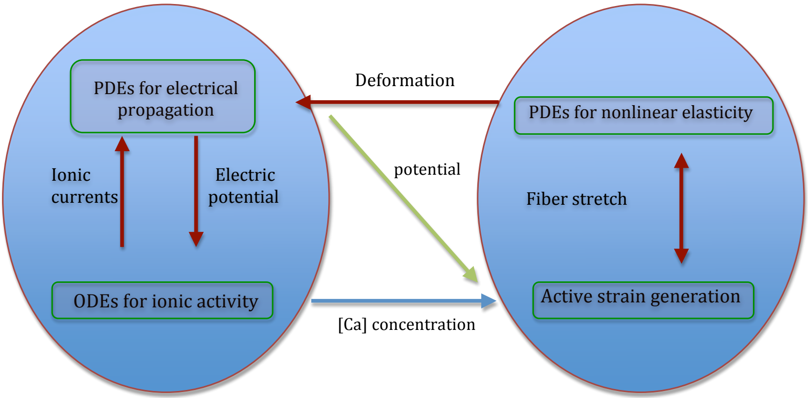
Currently with P1 Finite Elements.

Miocardium with scar.
L. Mirabella

MOVIE

Electromechanical Activity in the Heart

R. Ruiz



MOVIE

Fluid-Structure Interaction (FSI)

Coupled Problem

The ALE frame of reference

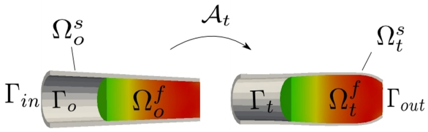


Figure: ALE mapping

ALE map $\mathcal{A}_t : \Omega_o^{\text{f}} \longrightarrow \Omega_t^{\text{f}}$

Property of the ALE derivative:

$$\partial_t \mathbf{u}_{\mathbf{f}}|_{\mathbf{x}_o}(\mathbf{x}, t) = \partial_t \mathbf{u}_{\mathbf{f}}(\mathbf{x}, t) + (\mathbf{w}(\mathbf{x}, t) \cdot \nabla)\, \mathbf{u}_{\mathbf{f}}(\mathbf{x}, t)$$

being $\mathbf{w}(\mathbf{x}) = \frac{d\mathcal{A}_t(\mathbf{x}_o)}{dt}$, $\mathbf{x} = \mathcal{A}_t(\mathbf{x}_o)$, the fluid domain velocity.

MOVIE

MOVIE

The fluid model: ALE formulation

How to obtain **w** from a given vessel wall displacement $\mathbf{d}_s$:

The Harmonic Extension equation for the fluid domain:

$$
\begin{aligned}
-\Delta \mathbf{d}_f &= 0 && \text{in } \Omega_o^{\mathrm{f}} \\
\mathbf{d}_f &= \mathbf{d}_s && \text{on } \Gamma_o \\
\mathcal{A}_t(\mathbf{x}_o) &= \mathbf{x}_o + \mathbf{d}_f(\mathbf{x}_o, t) && \forall \mathbf{x}_o \in \Omega_o^{\mathrm{f}}
\end{aligned}
$$

Alternatives: replace Laplacian by:

- a linear elasticity problem;
- Stokes problem (**w** divergence free)

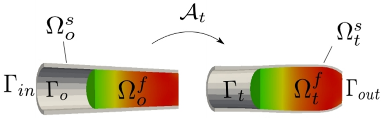
The Navier–Stokes equations in ALE form:

$$
\begin{aligned}
\rho_f \partial_t \mathbf{u_f}|_{x_o} + \rho_f(\mathbf{u_f} - \mathbf{w}) \cdot \nabla \mathbf{u_f} - \nabla \cdot \boldsymbol{\sigma}_f &= \mathbf{f}_f && \text{in } \Omega_t^{\mathrm{f}} \\
\nabla \cdot \mathbf{u_f} &= 0 && \text{in } \Omega_t^{\mathrm{f}}
\end{aligned}
$$

Boundary conditions

- Γ_{in}, Γ_{out} and external wall: problem dependent
- on Γ_t: defined by the FSI coupling (transmission conditions)

FSI problem: coupling conditions



Continuity of stresses

$$\boldsymbol{\sigma}_{os} \cdot \mathbf{n_o} = J\boldsymbol{\sigma}_f \mathbf{F}^{-T} \circ \mathcal{A}_t \cdot \mathbf{n_o} = \text{ on } \Gamma_o$$

Continuity of velocities

$$\mathbf{u}_f \circ \mathcal{A}_t = \frac{d\mathbf{d}_s}{dt} \text{ on } \Gamma_o$$

Geometry adherence

$$\mathcal{A}_t(\mathbf{x}_o) = \mathbf{x}_o + \mathbf{d}_s(\mathbf{x}_o) \text{ on } \Gamma_o, \quad \text{or}$$

$$\mathbf{d}_f = \mathbf{d}_s \quad \text{on } \Gamma_o$$

Nonlinearities and discretizations

Nonlinearity due to

- convective term in Navier–Stokes equations;
- nonlinear equation for the structure;
- moving fluid integration domain.

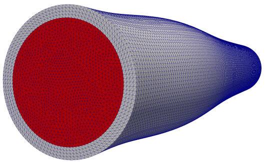
Time and space discretizations

- fully implicit (FI)
- convective explicit (CE)
- geometry-convective explicit (GCE) ($\rightarrow$ linear problem)

Galerkin Finite Element Method

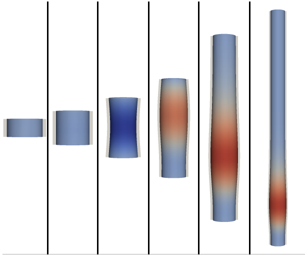
Fluid	Structure	ALE	(Harmonic Extension)
stabilized P1-P1	P1	P1	
P1 bubble-P1	P1	P1	
P2-P1	P2	P1	(straight triangles!)

Weak scalability, FSI
Meshes



Length	# Elements	# Vertices
0.5 cm	75'480	14'487
1 cm	149'520	27'405
2 cm	301'920	53'997
4 cm	603'840	106'677
8 cm	1'207'680	212'037
16 cm	2'415'360	422'757

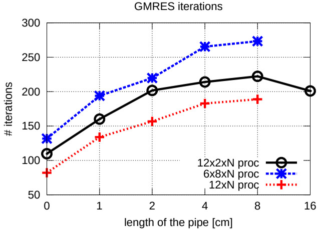
Fluid mesh

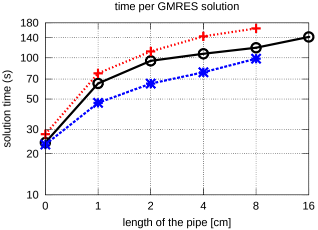


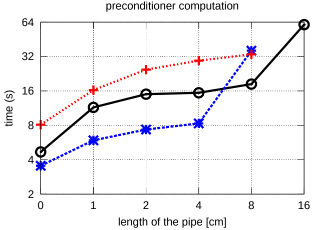
Length	# Elements	# Vertices
0.5 cm	27'840	6'380
1 cm	55'680	12'180
2 cm	111'360	23'780
4 cm	222'720	46'980
8 cm	445'440	93'380
16 cm	890'880	186'180

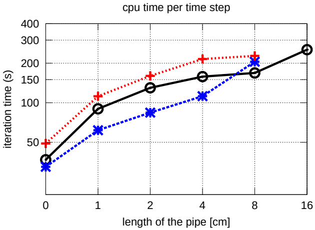
Solid mesh

Weak scalability, FSI, GCE, on Cray XT5 Rosa

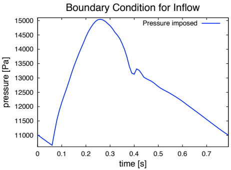




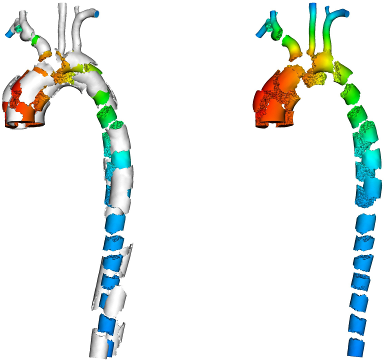




Aortic Flow
P. Crosetto



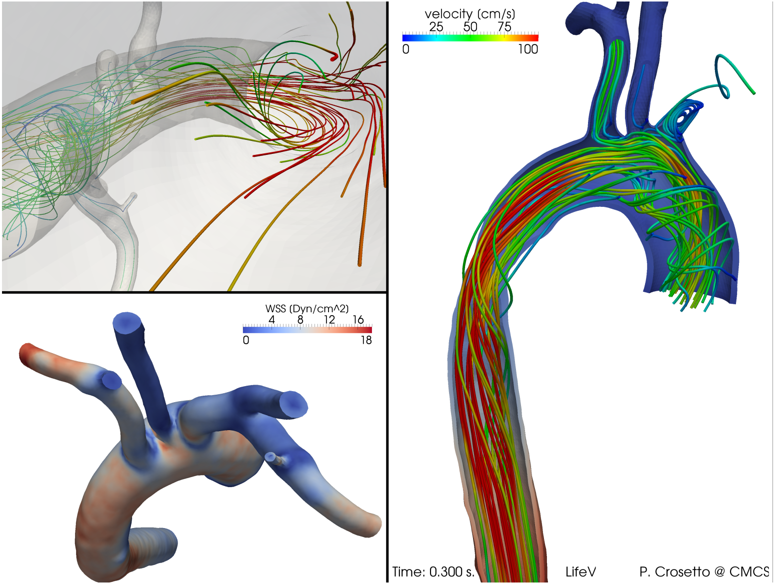
Mean pressure at aortic valve



The fluid and the solid meshes are partitioned in 2x32 subdomains. 380′690 tetrahedra and 486′749 dofs

[Crosetto, Reymond, SD, Kontaxakis, Stergiopoulos, Quarteroni(2010 and 2011)]

Aorta

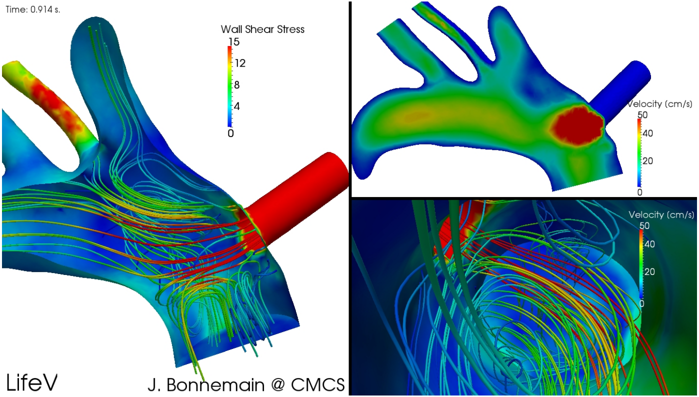


WSS (Dyn/cm^2)
4 8 12 16
0 18
velocity (cm/s)
25 50 75
0 100
Time: 0.300 s.
LifeV
P. Crosetto @ CMCS

VAD connection to an Aorta

MD J. Bonnemain

WSS and streamlines (steady state simulation)
Recirculation and secondary flows, velocity magnitude



VAD

MOVIE

1-D model

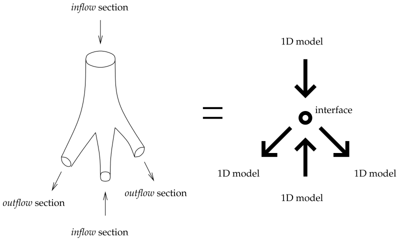
The equations of the 1-D model are (derivation in A. Quarteroni and L. Formaggia, *Mathematical modelling and numerical simulation of the cardiovascular system*, 2004):

$$\begin{cases} \dfrac{\partial A}{\partial t} + \dfrac{\partial Q}{\partial z} = 0 \\ \dfrac{\partial Q}{\partial t} + \dfrac{\partial}{\partial z}\left(\alpha \dfrac{Q^2}{A}\right) + \dfrac{A}{\rho}\dfrac{\partial P}{\partial z} + K_r\left(\dfrac{Q}{A}\right) = 0 \\ P - P_{\mathrm{ext}} = \psi(A, A^0, \beta, \gamma) = \beta\left(\sqrt{\dfrac{A}{A^0}} - 1\right) + \gamma \dfrac{1}{A\sqrt{A}}\dfrac{\partial A}{\partial t} \end{cases}$$

- $\alpha = \dfrac{1}{A}\displaystyle\int_{\mathcal{S}} s^2 d\sigma$ is the Coriolis coefficient;
- $K_r = -2\nu s'(r)$ is the friction coefficient;
- $s = \theta^{-1}(\theta + 2)(1 - r^{\theta})$ is the “assumed” velocity profile;
- β is the elastic coefficient of the wall;
- γ is the viscoelastic coefficient of the wall.

1D network, oriented graph

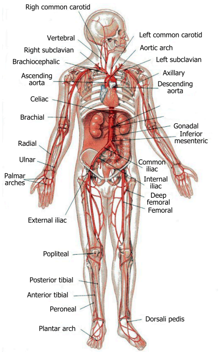
T. Passerini, J. Alastruey, L. Formaggia, J. Peiró

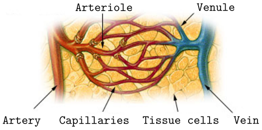


1D network can be seen as **oriented** *graphs*

edges stand for the 1D models
vertices represent the interfaces between the models.
vascular districts networks: *inflow* and *outflow* sections, ⇒ blood flow *path* ⇒ **orientation**
Data structures via `boost::graph` library (**BGL**), providing a generic interface for traversing graphs.

Geometrical multiscale modeling





The geometrical multiscale approach is a strategy for modeling the cardiovascular system, including the reciprocal interactions between local and systemic hemodynamics, i.e.:

- the effect of global circulation on specific components;
- the effect of local pathologies and diseases on the global system.

[*Formaggia et al. Comput. Visual. Sci. 1999, Vignon-Clementel et al. CMAME 2006, Blanco et al. CMAME 2007*]

Geometric multiscale I

Models and applications

- 3-D Fluid-Structure Interaction (FSI) models for studying complex pathologies (e.g., aneurysms, stenoses, . . .), or specific components which require a detailed geometrical description;
- non-linear 1-D models of hyperbolic equations for modeling the global network of arteries, in particular waves propagation;
- 0-D models for modeling valves, pulmonary circulation, capillaries and other peripheral terminations, which can be described by few parameters.

The objective is to design a flexible framework where different ingredients can be mixed together.

Geometric multiscale II

Ingredients

- Models: 0-D, 1-D, 3-D rigid fluid, 3-D FSI, . . . ;
- Couplings: pointwise, integrated/averaged, . . . ;
- Algorithms: semi-explicit; implicit with relaxed fix-point, Newton, Broyden. . . .

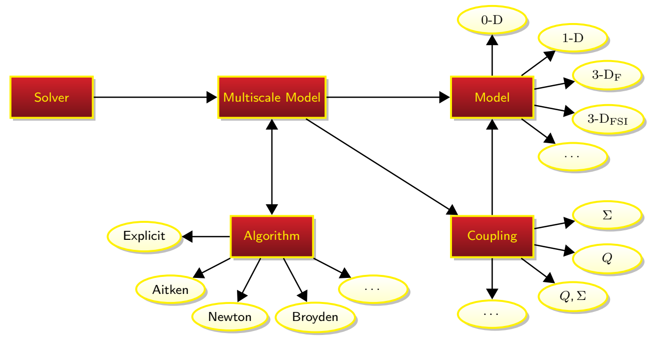
T. Passerini, M. Piccinelli, U. Villa, A. Veneziani, L. Formaggia. A. Quarteroni (oriented graph)

C. Malossi, P. Blanco, SD, A. Quarteroni (implicit coupling)

Geometrical multiscale framework
with C. Malossi, P. Blanco

Framework C++ implementation

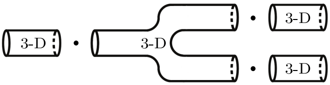
Abstract interfaces between models, coupling conditions, and algorithms.



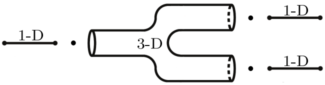
Modeling of the couplings

From the geometrical viewpoint, we have two different scenarios for the couplings:

1. the models are defined in the same geometrical space



2. the models are defined in different geometrical spaces



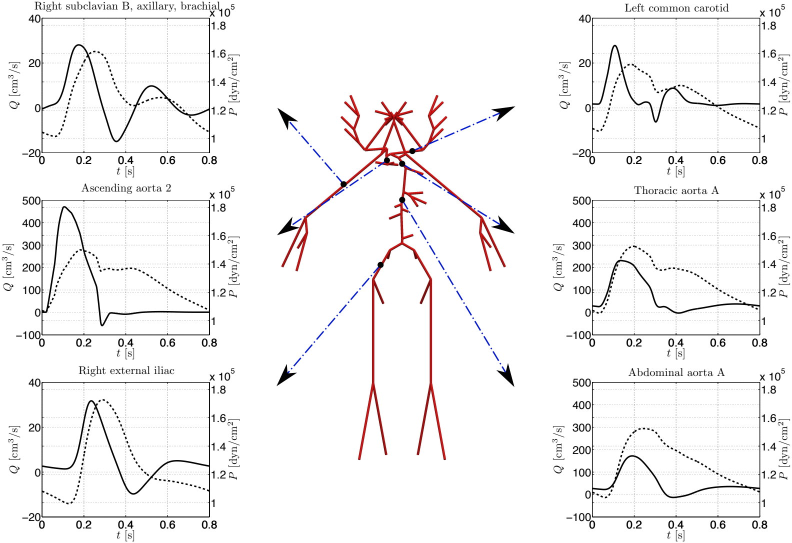
Fluid averaged/integrated coupling quantities

Given a flat coupling interface Γ equipped with the outgoing normal $\mathbf{n}$ we have:

- volumetric flow rate: $Q = \displaystyle\int_\Gamma \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}\Gamma$
- normal stress: $\Sigma = \dfrac{1}{|\Gamma|} \displaystyle\int_\Gamma (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \mathbf{n} \, \mathrm{d}\Gamma$

MOVIE

1-D network results



1-D arterial network + 3-D FSI aorta

Problem numbers

Models:

- 1 3-D FSI aorta;
- 95 1-D segments;
- 46 0-D windkessel RCR terminals.

Couplings:

- 98 coupling nodes;
- 243 coupling variables (unknowns).

Algorithms:

- 3-5 Broyden iters/time step;
- 30 CPU hours per heart beat (on 4 nodes = 32 cores).

MOVIE

MOVIE

LifeV developers

Alessandro Melani, MOX; Alessandro Veneziani, E(CM)^2; Alessio Fumagalli, MOX; Alexis Aposporidis, E(CM)^2; Antonio Cervone, MOX; Claudia Colciago, CMCS; Christian Vergara, MOX; Cristiano Malossi, CMCS; Gianmarco Mengaldo, MOX; Guido Iori, MOX; Gwenol Grandperrin, CMCS; Jean Bonnemain, CMCS; Laura Cattaneo, MOX; Luca Bertagna, MOX; Luca Formaggia, MOX; Lucia Mirabella, CFM Lab Marta D'Elia, E(CM)^2; Matteo Pozzoli, MOX; Mauro Perego, CS - Florida State Univ. Michel Kern, ESTIME - INRIA Nur Fadel, MOX; Paolo Crosetto, CMCS; Radu Popescu, CMCS; Ricardo Ruiz Baier, CMCS; Samuel Quinodoz, CMCS; Simone Deparis, CMCS; Simone Pezzuto, MOX; Simone Rossi, CMCS; Tiziano Passerini, E(CM)^2; Toni Lassila, CMCS; Tricerri Paolo, CMCS; Umberto Emanuele Villa, E(CM)^2; ...