Feel++ : A Versatile High Performance Finite Element Embedded Library into C++

Vincent Chabannes, Christophe Prud'homme

Some features

A Domain Specific Language for PDEs embedded in C++ providing a syntax very close to the mathematical language in order to describe the variational formulation :

- Supports generalized arbitrary order Galerkin methods (cG, dG) in 1D, 2D, 3D
- Supports simplex, hypercube and high order meshes
- Supports finite elements : Lagrange, Hermite, Nedelec, Raviart-Thomas
- Supports seamless parallel computing
- Supports seamless interpolation between grids/function spaces
- Supports symbolic computation thanks to GiNaC
- Supports large scale parallel linear and non-linear solvers (PETSc/SLEPc)
- Supports hybrid computing : MPI, multi-thread, GPU (HARTS)
- Supports in-situ visualization with ParaView

Laplacian problem :

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

```
auto mesh = loadMesh(_mesh=new Mesh<Simplex<2>>);
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);
```

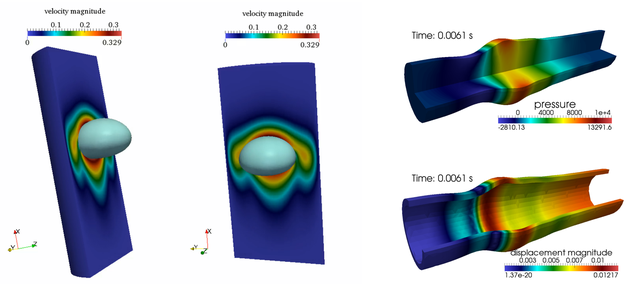
Stokes problem :

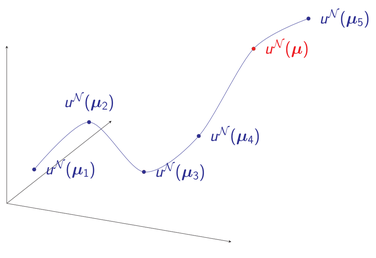
$$\begin{cases} \text{Find } (u,p) \text{ such that :} \\ -\Delta u + \nabla p = 0 \ \text{ in } \Omega \\ \nabla \cdot u = 0 \ \text{ in } \Omega \end{cases} \xrightarrow[\text{formulation}]{\text{variational}} \begin{cases} \text{Find } (u,p) \in V_h \times Q_h \text{ such that} \\ \int_\Omega \nabla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

```
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 advanced methods

- Fluid-Structure Interaction
  - Levelset method
  - High order ALE
- Non-conforming methods
  - *h*-*p* mortar methods
  - Fictitious domain method (FBM)
- Certified Reduced Basis methods
  - Fast evaluation of functional outputs
  - Certifiable accuracy of functional outputs
  - Applicable to non-linear multi-physics problems





$u^{\mathcal{N}}(\boldsymbol{\mu})$: finite element solution , u^N reduced basis solution

$$\underbrace{W^N = span\{u^{\mathcal{N}}(\boldsymbol{\mu}_1), \ldots, u^{\mathcal{N}}(\boldsymbol{\mu}_N)\}}_{\text{Reduced basis space}} \rightarrow u^N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_i^N(\boldsymbol{\mu}) \underline{u}^{\mathcal{N}}(\boldsymbol{\mu}_i)$$

$$A^N(\boldsymbol{\mu}) u^N(\boldsymbol{\mu}) = F^N(\boldsymbol{\mu}) \longrightarrow \text{system } N \times N \text{ with } N \ll \mathcal{N}$$

Preconditioners and HPC

Objectives : Scalable solvers from a few processors to thousands of processors and billions of unknowns

- PETSc preconditioners
  - Block-Jacobi, GASM, Multigrid (GAMG, ML)
  - Multi-Physics with fieldsplit (Gauss-Seidel, Schur)
  - Mixing and tuning in Feel++ configuration file
- In-House Substructuring preconditioners
  - Mortar (Elliptic)
  - BDD-GenEO

$$\mathbf{P}_{MORTAR} = \begin{pmatrix} \mathbf{P}_V^{DG} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_E \end{pmatrix}$$

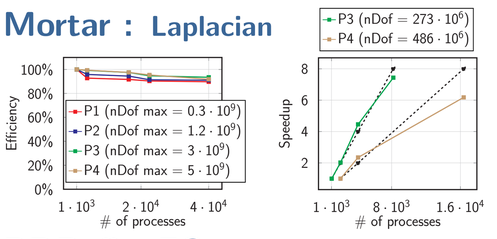
$$\mathbf{P}_{BDD}^{-1} = P^T \left(\sum_{i=1}^{N} B^{(i)} D_i S_i^{\dagger} D_i B^{(i)^T} \right) P$$

- Navier-Stokes preconditioners
  - SIMPLE
  - LSC
  - PMM
  - PCD

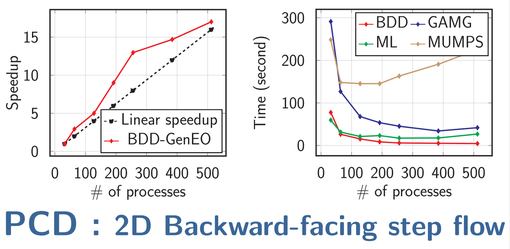
$$\mathbf{P}_{PCD} = \begin{pmatrix} \mathbf{F} & \mathbf{B^T} \\ \mathbf{0} & -\hat{\mathbf{S}} \end{pmatrix} \text{ with } \hat{\mathbf{S}} = \mathbf{Q}_p \mathbf{F_p}^{-1} \mathbf{A}_p$$

$\mathbf{A_p} = \mathbf{B}\mathbf{Q}_u^{-1}\mathbf{B}^T$,
$\mathbf{Q}_u$ (resp $\mathbf{Q}_p$) velocity (resp pressure) mass matrix,
$\mathbf{F_p}$ convection-diffusion operator + BC

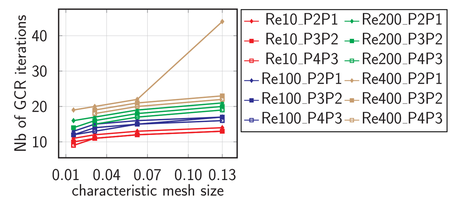
Mortar : Laplacian



BDD-GenEO : Linear Elasticity



PCD : 2D Backward-facing step flow



Hemodynamic simulations in the cerebral venous network

Question: What are the effects of different models and flow conditions on the quantification of hemodynamic variables ?

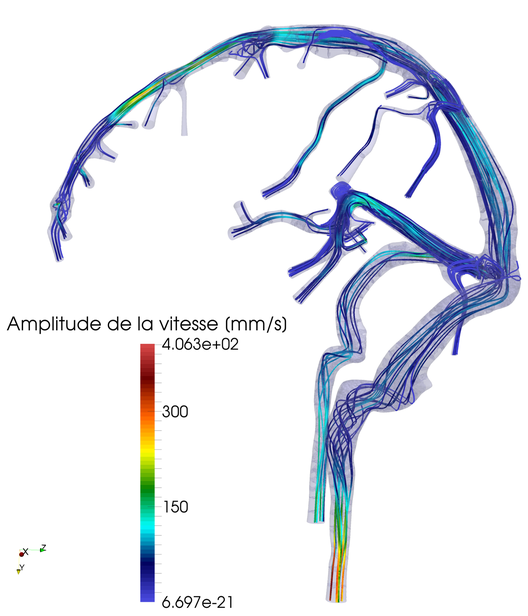
- Influence of the inflow boundary condition (order of magnitude).
- Influence of the outflow boundary conditions:
  - Free outlet condition vs. 3D-0D coupling;
  - Choice of parameters in the 0D model.
- Influence of the non-Newtonian characteristics of blood flow.

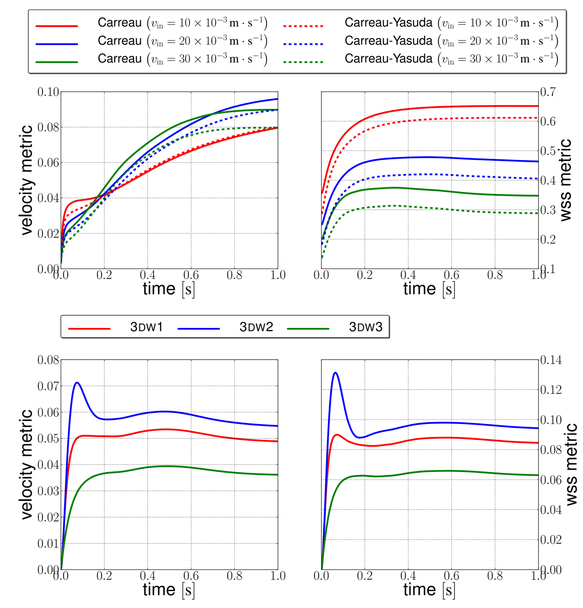
Strategy: computational modeling + sensitivity analysis approach $\Longrightarrow$ require HPC and scalable preconditioner

Preconditioner used:

$$\begin{pmatrix} A & B^T & D \\ B & 0 & 0 \\ C & 0 & E \end{pmatrix} \begin{pmatrix} U \\ P \\ \Lambda \end{pmatrix} = \begin{pmatrix} F \\ 0 \\ G \end{pmatrix}$$

- block Gauss-Seidel preconditioner is first applied to the 3D-0D blocks (velocity and pressure are aggregated in the 3D block)
- SIMPLE (+GASM) preconditioner on the velocity/pressure saddle-point problem





Design of high field magnet

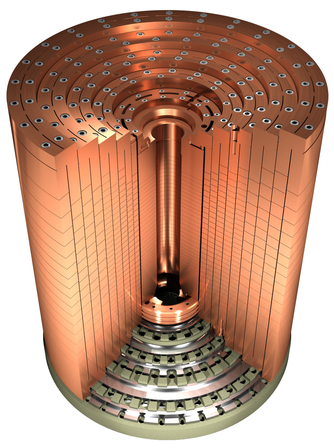
LNCMI : Laboratory of high magnetic fields

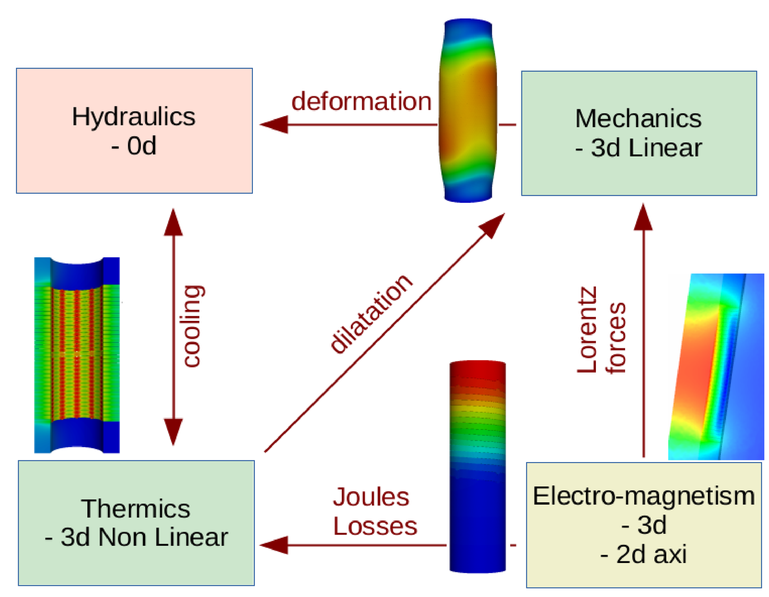
- Application domains : chemistry, magnetoscience, superconductors
- Continuous magnetic field (36 Tesla)

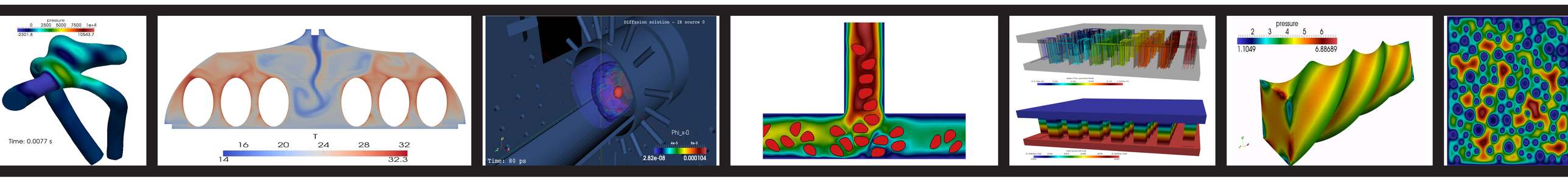
Challenges:

- Modeling : multi-physics non-linear models, complex geometries, genericity
- Account for uncertainties : uncertainty quantification, sensitivity analysis
- Optimization : shape of magnets, robustness of design

Strategy: Reduced basis method







WWW: http://www.feelpp.org Contacts: {christophe.prudhomme,vincent.chabannes}@feelpp.org,contact@cemosis.fr