

Reduced basis methods and high performance computing.

Applications to non-linear multi-physics problems

Christophe Prud'homme
prudhomme@unistra.fr

Cemosis - <http://www.cemosis.fr>
IRMA
Université de Strasbourg

23 octobre 2014

Collaborators on the framework

- Cecile Daversin (IRMA - UdS, Strasbourg)
- Christophe Prud'homme (IRMA - UdS, Strasbourg)
- Alexandre Ancel (IRMA - UdS, Strasbourg)
- Christophe Trophime (CNRS/LNCMI, Grenoble)
- Stephane Veys (LJK - UJF, Grenoble)
- Jean-Baptiste Wahl (IRMA, Strasbourg)

and former collaborators : S. Vallaghe, E. Schenone.

Non-intrusive RB

- Rachida Chakir (LJLL - UPMC, Paris)
- Yvon Maday (LJLL - UPMC, Paris)

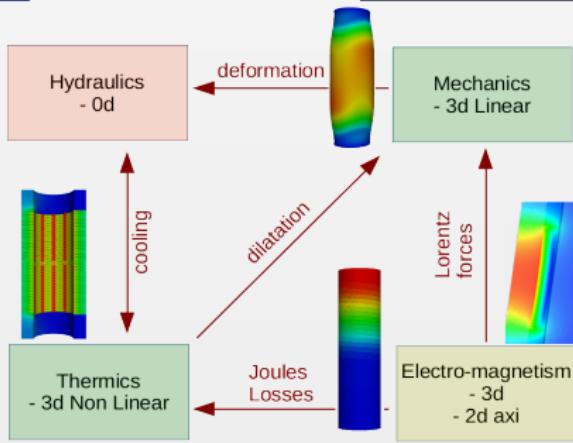
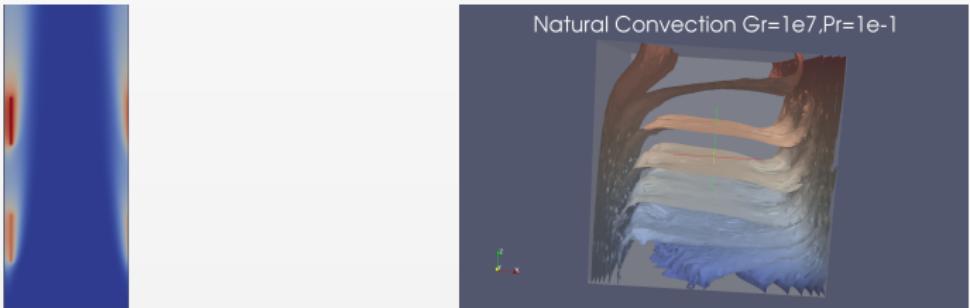
Current Funding

ANR/CHORUS,
ANR/HAMM,
FRAE/RB4FASTSIM,
ANR/Vivabrain, Labex
IRMIA, IDEX

- ① Motivations and Framework
- ② Reduced Basis for Non-Linear Problems and Extension to Multiphysics
- ③ Applications
- ④ Conclusions

Motivations and Framework

Motivations and Context



Objectives

- Provide an open framework for certified reduced basis methods and low rank methods
- Provide a rapid prototyping framework using the Feel++ language for the standard finite/spectral element methods
- Provide interfaces to various open "mathematical" programming environments such as Python/OpenTURNS(UQ) or Octave

Where to get it ?

- sources are available at <http://www.feelpp.org>
- available as Debian/Ubuntu packages

Features

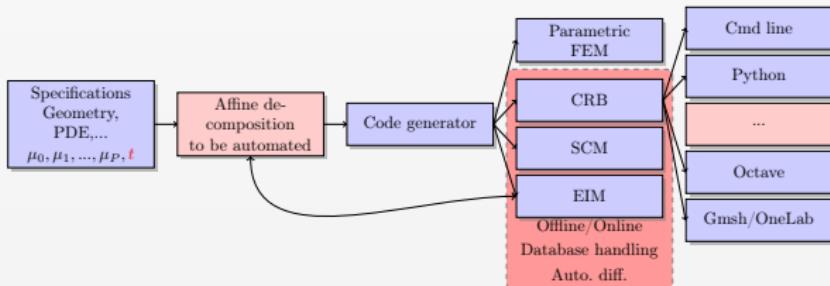
- Galerkin methods (fem,sem, cG, dG) in 1D, 2D and 3D on simplices and hypercubes
- Interfaces to PETSc/SLEPc
- Language embedded in C++ close to variational formulation language that shortens tremendously the “time to results”

```
//  $\mathcal{T}_h = \{\text{elements}\}$ 
auto mesh = loadMesh( new Mesh<Simplex<3> > );
//  $X_h = \{v \in C^0(\Omega) | v_K \in \mathbb{P}_2(K), \forall K \in \mathcal{T}_h\}$ 
auto Xh = Pch<2>(mesh);
auto u = Xh->element(), v = Xh->element();
auto a = form2( _test=Xh, _trial=Xh, );
//  $a(u,v) = \int_{\Omega} \nabla u \cdot \nabla v$ 
a = integrate(elements(mesh), gradt(u)*trans(grad(v)));
auto b = form2( _test=Xh, _trial=Xh );
//  $b(u,v) = \int_{\Omega} u \cdot v$ 
b = integrate( elements(mesh), idt(u)*id(v) );
```

Feel++ Reduced Basis/Low Rank Methods Features

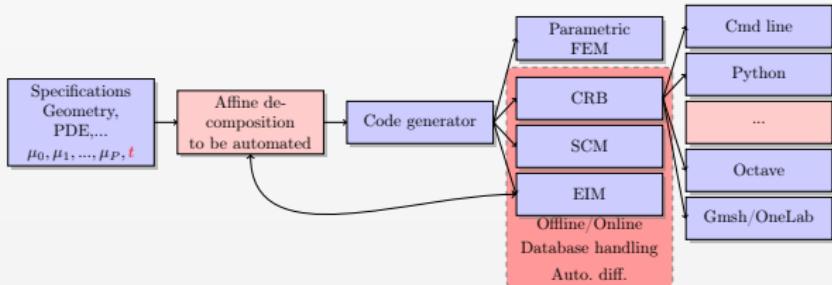
Lower bound for coercivity/inf-sup	OK	90%
CRB Linear Elliptic case	OK	100% (coercive) 90% (non-coercive)
CRB Linear Parabolic case	OK	100%
Automatic differentiation	OK	90%
EIM	OK	100%
CRB Nonlinear case	Ok	80%
CRB automated affine decomposition	Not OK	50%
CRB for multiphysics	OK	80%
Low Rank Tensor	OK	Demonstrator

Feel++ Framework : the User point of view



```
class MyModel :  
public ModelBase<ParameterSpace<4>,  
decltype(Pch<5>(Mesh<Simplex<3>::New()))>{  
...  
init(){  
    auto mesh=loadMesh(_mesh=new Mesh<Simplex<3>);  
    this->setFunctionSpaces( Pch<5>( mesh ) );  
  
    // define bounds for $D^{\mu}$  
    ...  
    // affine decomposition here  
    ...  
}
```

Wrappers : Python/OpenTURNS

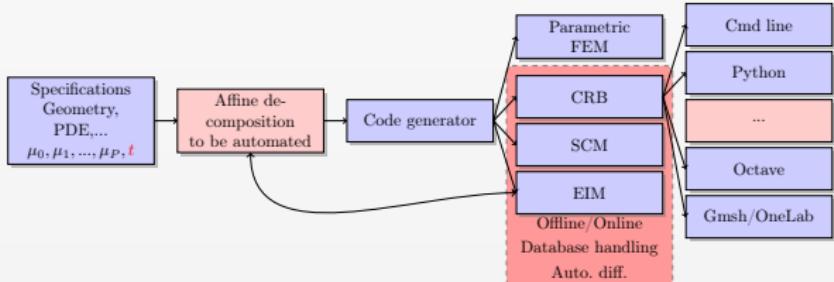


Wrappers are automatically generated by the framework

Python Code (OpenTURNS wrapper for UQ)

```
# fem code
modelfem = NumericalMathFunction("modelfem")
# crb code
modelrb = NumericalMathFunction("modelrb")
mu[0] = 10
mu[1] = 7e-3
outputfem = modelfem(mu)
outputrb = modelrb(mu)
```

Wrapper : Octave



Wrappers are automatically generated by the framework

Octave code

```
# setup parameters
# kIC : thermal conductivity (default: 2)
inP(1) = 1.0e+1;
# D : fluid flow rate (default: 5e-3)
inP(2) = 7.0e-3;
...
for i=1:N
    inP(1)= 0.2+(i-1)*(150-0.2)/N;    D=[D inP(1)];
    s= [s opuseadscrb( inP )];
end
```

High Performance Computing

HPC Strategies

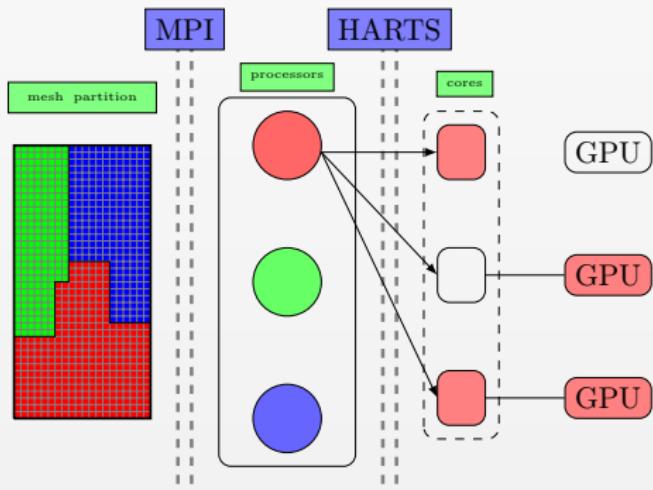
Computing ressources

Objectives :

- Realistic geometries
- Accurate computing

Consequences :

- Memory
- Performance



Technologies

- MPI : Data partitioning
- MT : Global assembly
- GPU : Local assembly

- Training parameter sampling
 - Offline : identical on all cores
 - Online : distributed (greedy)
- Parallel in spatial domain
 - RB construction
 - Residual parameter independent terms
 - SCM
- Collective communications
 - Scalar products
 - Outputs
- Opportunities : Robust preconditioners reuse
- Memory hungry RB : matrix free operators

Reconstruction of reduced basis fields : for visualisation or usage in other context might be tricky due to spatial partitioning.

HPC Strategies

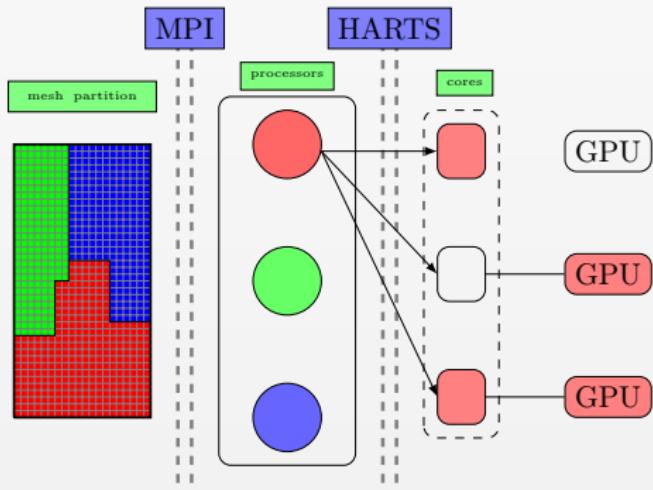
Computing ressources

Objectives :

- Realistic geometries
- Accurate computing

Consequences :

- Memory
- Performance



Technologies

- MPI : (Spatial,Parameter)Data partitioning
- (MT,)GPU : **Online computations**

“Truth” FEM Approximation

Let $I = (0, T_f)$, with T_f is the final time. Let $\mu \in \mathcal{D}^\mu$ and $t \in I$, evaluate

$$s^{\mathcal{N}}(t; \mu) = \ell(t; u^{\mathcal{N}}(\mu)) ,$$

where $u^{\mathcal{N}}(t; \mu) \in X^{\mathcal{N}} \subset X$ satisfies

$$m \left(\frac{\partial u^{\mathcal{N}}(\mu)}{\partial t}, v; \mu \right) + a(u^{\mathcal{N}}(\mu), v; \mu) = f(t; v), \quad \forall v \in X^{\mathcal{N}} \quad \forall t \in I .$$

- $a(\cdot, \cdot; \mu)$ and $m(\cdot, \cdot; \mu)$ are bilinear, $f(\cdot; \mu)$ and $\ell(\cdot; \mu)$ are linear
- f and ℓ are bounded

“Truth” FEM Approximation : Inner products and Norms

Let Δt the time step defined by $\Delta t = \frac{T_f}{K}$.

$\forall w, v \in X$ and $1 \leq k \leq K$ we next define the

- energy inner product and associated norm (parameter dependent)

$$\left(\left(\left(w(t^k), v(t^k) \right) \right) \right)_\mu = m(w(t^k), v(t^k); \mu) + \sum_{k'=1}^k a_S(w(t^{k'}), v(t^{k'}); \mu) \Delta t$$

$$\left| \left| \left| v(t^k) \right| \right|_\mu = \sqrt{m(v(t^k), v(t^k); \mu) + a_S(v(t^{k'}), v(t^{k'}); \mu) \Delta t}$$

- X -inner product and associated norm (parameter independent)

$$\left(w(t^k), v(t^k) \right)_X = \left(\left(\left(w(t^k), v(t^k) \right) \right) \right)_{\bar{\mu}} \quad \forall w, v \in X, 1 \leq k \leq K$$

$$\left| \left| \left| v(t^k) \right| \right|_X = \left| \left| \left| v(t^k) \right| \right|_{\bar{\mu}} \quad \forall v \in X, 1 \leq k \leq K$$

where a_s denotes the symmetric part of a .

Spaces

Let \bar{N} a positive number such that $1 \leq \bar{N} \leq N_{\max}$.

Parameter Samples :

Sample : $S_{\bar{N}} = \{\mu_1 \in \mathcal{D}^\mu, \dots, \mu_{\bar{N}} \in \mathcal{D}^\mu\} \quad 1 \leq N \leq \bar{N}$,

with

$$S_1 \subset S_2 \dots S_{\bar{N}} \subset \mathcal{D}^\mu$$

Let $\text{Snap}(\mu) = \left\{ u^{\mathcal{N}}(t^k, \mu), 1 \leq k \leq K \right\}$ a snapshot set with $\mu \in S_{\bar{N}}$.

Let N_m such that $1 \leq N_m \leq K$, So we have $\bar{N} = \frac{N_{\max}}{N_m}$.

Let $\rho_i(\mu) \in X^{\mathcal{N}}, 1 \leq i \leq N_m$, eigen modes (associated to μ) selected a Proper Orthogonal Decomposition algorithm

$$\left\{ \rho_i(\mu) \in X^{\mathcal{N}}, 1 \leq i \leq N_m \right\} = \text{POD}\left(\text{Snap}(\mu), N_m\right)$$

$$W_N = \text{span} \left\{ \zeta_n \equiv \rho_i(\mu) \text{ such that } \lfloor n/N_m \rfloor N_m + i \leq n, \mu \in S_{\bar{N}}, n = 1, \dots, N \right\}$$

with

$$W_1 \subset W_2 \dots W_{N_{\max}} \subset X^{\mathcal{N}} \subset X$$

Algorithm

POD algorithm :

Algorithm 1 $\{\rho_i(\mu) \in X^{\mathcal{N}}, 1 \leq i \leq N_m\} = POD(Snap(\mu), N_m)$

Build matrix \mathcal{M} such that $(\mathcal{M})_{ij} = (u^{\mathcal{N}}(t^i, \mu), u^{\mathcal{N}}(t^j, \mu))_X \quad 1 \leq i, j \leq K$

Solve $\mathcal{M} \varphi_i = \lambda_i \varphi_i$ for $(\varphi_i \in \mathbb{R}^K, \lambda_i \in \mathbb{R})_{1 \leq i \leq N_m}$ associated to N_m larger eigen values of \mathcal{M}

Build $\rho_i(\mu) = \sum_{k=1}^K \varphi_k u^{\mathcal{N}}(t^k, \mu)$ with $1 \leq i \leq N_m$.

A posteriori error estimation

Given $\mu \in \mathcal{D}^\mu$ we define

$$\varepsilon_N(t^k; \mu) = \|r(u_N(t^k; \mu), v; \mu)\|_{X'}, \quad 1 \leq k \leq K.$$

Let $e(t^k, \mu) = \|u^N(t^k, \mu) - u_N(t^k, \mu)\|_X$, we can write

$$e(t^k, \mu) \leq \Delta_N(t^k, \mu) \equiv \sqrt{\frac{\Delta t}{\alpha_{LB}(\mu)} \sum_{k'=1}^k \varepsilon_N(t^{k'}; \mu)^2}, \quad 1 \leq k \leq K.$$

$\forall \mu \in \mathcal{D}^\mu$ and $1 \leq k \leq K$ the output error bound is given by

$$|s^N(t^k, \mu) - s_N(t^k, \mu)| \leq \Delta_N^s(t^k, \mu) \equiv \Delta_N(t^k, \mu) \Delta_N^{du}(t^k, \mu).$$

Example Heat Shield : Problem statement

$$\begin{cases} -\Delta u + \frac{\partial u}{\partial t} = 0 & \text{in } \Omega, \\ \text{boundaries conditions} & \text{on } \partial\Omega. \end{cases}$$

Boundaries conditions

$\partial\Omega_{ext}$: heat transfert with $T_{air} = 1$

- $-\nabla u \cdot \mathbf{n} = Biot_{ext}(u - T_{air});$

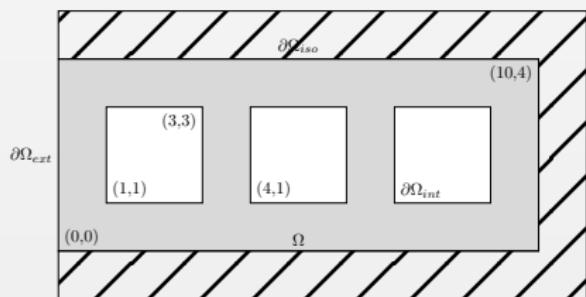
$\partial\Omega_{int}$: heat transfert, $T_{air} = 0$

- $-\nabla u \cdot \mathbf{n} = Biot_{int}(u - T_{air});$

$\partial\Omega_{iso}$: Insulated.

2 parameters

- $Biot_{ext}$ and $Biot_{int}.$



Example Heat Shield

- final time : 20 s ;
- time step : 0.2 s ;
- Minimum parameters values.

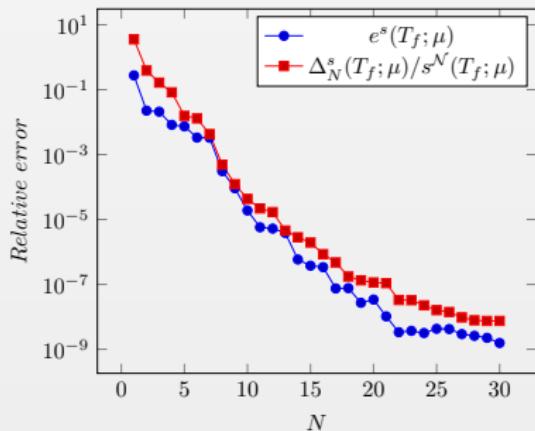


- Maximum parameters values.



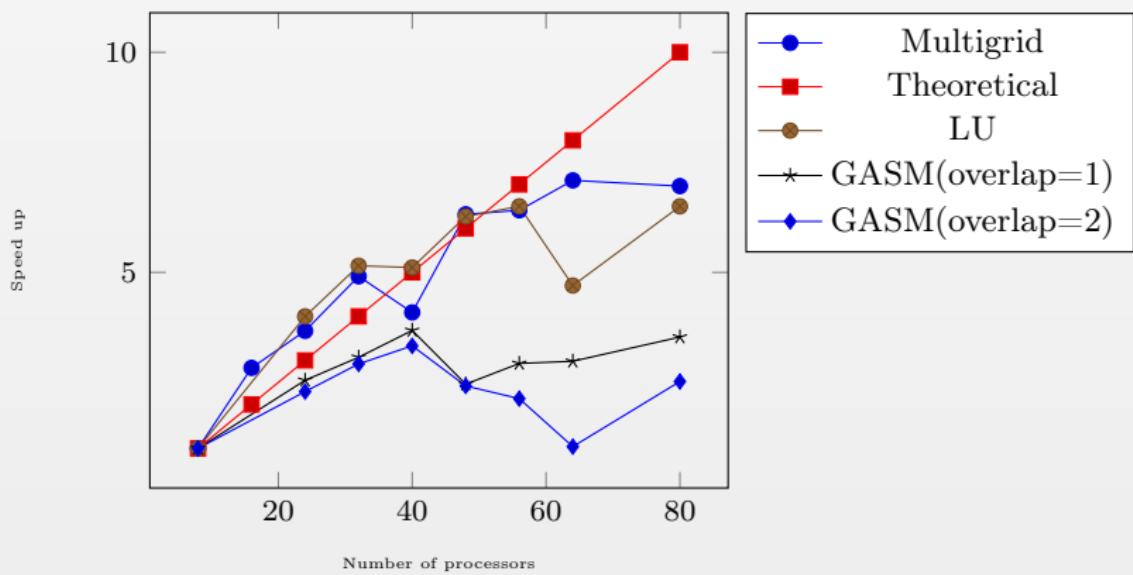
Example Heat Shield

- Configuration :
 - 33 000 dofs ;
 - Preconditioner : LU(MUMPS) – Solver : KSP
 - Ξ : parameter sampling of dimension 430.
- Let $e^s(T_f; \mu) = \frac{|s^N(T_f; \mu) - s_N(T_f; \mu)|}{s^N(T_f; \mu)}$, plot $\max_{\mu \in \Xi} e^s(T_f; \mu)$



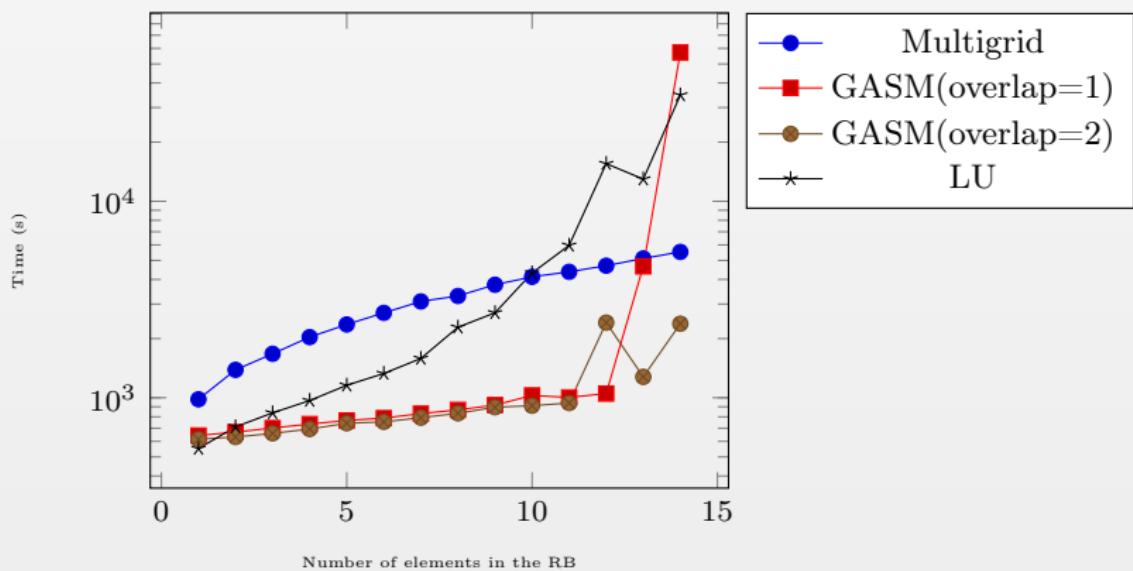
Example Heat Shield : Scalability

- Time to build the first reduced basis
- Configuration : 292 000 dofs for FEM approximation.
- Solver : KSP. Preconditioner : LU(MUMPS), GASM(1 or 2), Multigrid



Example Heat Shield : Basis construction

- Time to build the full reduced basis on 32 procs
- Configuration : 292 000 dofs for FEM approximation.
- Solver : KSP Preconditioner : LU(MUMPS, GASM(1 or 2), Multigrid



Reduced Basis for Non-Linear Problems and Extension to Multiphysics

Notations

NonLinear μ -parametrized PDE

- Set of parameters : $\mu \in \mathcal{D}^\mu \subset \mathbb{R}^p$,
- Solution of the nonlinear μ -PDE : $u(\mu) \in X \equiv H^1(\Omega \subset \mathbb{R}^d)$,
- PDE weak formulation : We look for $u(\mu) \in X$ such that

$$g(v; \mu, u(\mu)) = 0, \quad \forall v \in X .$$

Bootstrap Truth approximation

- $X^{\mathcal{N}} \subset X$: finite element approximation of dimension $\mathcal{N} \gg 1$.
- $u^{\mathcal{N}}(\mu) \in X^{\mathcal{N}}$ is solution of $g(v; \mu, u^{\mathcal{N}}(\mu)) = 0, \forall v \in X^{\mathcal{N}}$.
- Solution strategies such as Newton or Picard iterations, e.g. given ${}^0u^{\mathcal{N}}$, build the nonlinear iterates ${}^1u^{\mathcal{N}}, \dots, {}^k u^{\mathcal{N}}, \dots$

$$j(\delta^k u^{\mathcal{N}}(\mu), v; {}^k u^{\mathcal{N}}(\mu), \mu) = -g(v; \mu, {}^k u^{\mathcal{N}}(\mu))$$

where

$$\delta^k u^{\mathcal{N}}(\mu) = {}^{k+1} u^{\mathcal{N}}(\mu) - {}^k u^{\mathcal{N}}(\mu).$$

We recover the linear case.

- Equate $u(\mu)$ and $u^{\mathcal{N}}(\mu)$, i.e. $\|u(\mu) - u^{\mathcal{N}}(\mu)\|_X \leq \text{tol}, \forall \mu \in \mathcal{D}^\mu$.
- **Bootstrapping the reduced basis method**

Decomposition of the jacobian and residual

Non affine decomposition of g and j

We suppose that we have $\forall v \in X$ and $\forall \mu \in \mathcal{D}^\mu$, j reads

$$j(u(\mu), v; \mu, \mathbf{w}(\mu)) = \sum_{q=1}^{Q_j} \int_{\omega_q} \sigma_q^j(x, \mu; \mathbf{w}(\mu)) \mathbf{j}_q(u(\mu), v)$$

and g reads

$$g(v; \mu, \mathbf{w}(\mu)) = \sum_{q=1}^{Q_g} \int_{\omega_q} \sigma_q^g(x, \mu; \mathbf{w}(\mu)) \mathbf{g}_q(v).$$

where \mathbf{j}_q and \mathbf{g}_q no longer depend on μ , however σ_q^j and σ_q^g depend on x, μ and u . ω_q denotes a part of the computational domain.

Recovering affine decomposition (Nonlinear-case)

Apply EIM on all σ_q^g and build $g_{AD}(u(\mu), v; \mu)$ such that

$$g(v; \mu, \mathbf{w}(\mu)) \approx g_{AD}(v; \mu, \mathbf{w}(\mu)) = \sum_{q=1}^{Q_g} \sum_{m=1}^{M_q^g} \beta_g^{qm}(\mu; \mathbf{w}(\mu)) \underbrace{\int_{\omega_q} q_m(x) \mathbf{g}^q(v)}_{g^{qm}(v)},$$

and similarly build EIM expansion for σ_q^j and build $j_{AD}(u(\mu), v; \mu, \mathbf{w}(\mu))$ such that

$$\begin{aligned} j(u(\mu), v; \mu; \mathbf{w}(\mu)) &\approx j_{AD}(u(\mu), v; \mu; \mathbf{w}(\mu)) = \\ &\sum_{q=1}^{Q_j} \sum_{m=1}^{M_q^j} \beta_j^{qm}(\mu; \mathbf{w}(\mu)) \underbrace{\int_{\omega_q} q_{qm}^j(x) \mathbf{j}^q(u(\mu), v)}_{j^{qm}(u(\mu), v)}, \end{aligned} \quad (1)$$

Truth Finite element approximations

- $X^{\mathcal{N}} \subset X$: finite element approximation of dimension $\mathcal{N} \gg 1$.
- $u_{AD}^{\mathcal{N}}(\mu) \in X^{\mathcal{N}}$ is solution of $g_{AD}(u_{AD}^{\mathcal{N}}(\mu), v; \mu) = 0, \forall v \in X^{\mathcal{N}}$.
- Solution strategies such as Newton or Picard iterations, e.g. given ${}^0 u_{AD}^{\mathcal{N}}$, build the nonlinear iterates ${}^1 u_{AD}^{\mathcal{N}}, \dots, {}^k u_{AD}^{\mathcal{N}}, \dots$

$$j_{AD} (\delta^k u_{AD}^{\mathcal{N}}(\mu), v; \mu; {}^k u_{AD}^{\mathcal{N}}(\mu)) = -g_{AD} (v; \mu, {}^k u_{AD}^{\mathcal{N}}(\mu)) ,$$

for the increment $\delta^k u^{\mathcal{N}}(\mu)$ defined by

$$\delta^k u^{\mathcal{N}}(\mu) = ({}^{k+1} u^{\mathcal{N}}(\mu) - {}^k u^{\mathcal{N}}(\mu)) . \quad (2)$$

Ingredients

- Training set $\Xi_{train}^\mu \subset \mathcal{D}^\mu$
- Offline step
 - Sample $S_M = \{\mu_1 \in \Xi_{train}^\mu, \dots, \mu_M \in \Xi_{train}^\mu\}$, Interpolation points $t_1, \dots, t_M \in \Omega$
 - Approximation space $W_M = \text{span}\{q_1(x), \dots, q_M(x)\}$
 - Residual $r_m(x) = \sigma(x, \mu_m; \mathbf{u}^N(\mu_m)) - \sigma_m(x, \mu_m; \mathbf{u}^N(\mu_m))$
 - $q_{m+1}(x) = \frac{r_m(x)}{r_m(t_m)}$ (matrix $(B_{i,j}) = q_j(t_i)$ lower triangular)
- Online step : Compute approximation coefficients $\beta_m(\mu; \mathbf{u}_{AD}^N(\mu))$

$$\sigma_M(t_i; \mu; \mathbf{u}_{AD}^N(t_i, \mu)) = \sum_{m=1}^M \beta_m(\mu; \mathbf{u}_{AD}^N(t_i, \mu)) q_m(t_i) = \sigma(t_i; \mu; \mathbf{u}_{AD}^N(t_i, \mu))$$

$$\forall i = 1, \dots, M$$

A language embedded in C++ : EIM expansion

Let u be the solution of $g(u, v; \mu) = 0 \forall v \in X^{\mathbb{N}}$ and $\sigma(u)$ the non linear expression involving a field.

```
parameterspace_ptrtype D; parameter_type mu; // $\mu \in \mathcal{D}^\mu$ 
auto TrainSet = Dmu->sampling();
int eim_sampling_size = 1000;
TrainSet->randomize(eim_sampling_size);

//expression we want EIM expansion
auto sigma = ref(mu(0))/(1+ref(mu(2))*(idv(u)-u0));
//call Feel++ function eim
auto eim_sigma = eim(_model=solve( g(u, v; mu; x) = 0 ),
                     _element=u,           //  $u\mathcal{N}(\mu)$ 
                     _parameter=mu,        //  $\mu$ 
                     _expr=sigma,          //  $\sigma(u)$ 
                     _space=X_N,
                     _name="eim_sigma",
                     _sampling=TrainSet );

//then we can have access to  $\beta$  coefficients
// of EIM expansion  $\sum_{m=1}^M \beta(\mu, u\mathcal{N}(\mu)) q_m(x)$ 
std::vector<double> beta_sigma = eim_sigma->beta( mu );
```

Reduced Basis Approximation

- Build $S_N = \{\mu_i, i = 1, \dots, N\}$: a parameter sampling
- Build $X_N = \{u_{AD}^N(\mu_i), i = 1, \dots, N\}$: reduced basis approximation space of dimension $N << \mathcal{N}$.
- $u_{AD}^N(\mu) \in X_N$ is solution of $g_{AD}(u_{AD}^N(\mu), v; \mu) = 0, \forall v \in X_N$.
- Solution strategies such as Newton or Picard iterations, e.g. given ${}^0 u_{AD}^N$, build the nonlinear iterates ${}^1 u_{AD}^N, \dots, {}^k u_{AD}^N, \dots$

$$j_{AD} (\delta^k u_{AD}^N(\mu), v; \mu; {}^k u_{AD}^N(\mu)) = -g_{AD} (v; \mu, {}^k u_{AD}^N(\mu)) ,$$

with the increment $\delta^k u^N(\mu) = ({}^{k+1} u_{AD}^N(\mu) - {}^k u_{AD}^N(\mu))$

- EIM Online step : Compute $\beta_m(\mu; u_{AD}^N(t_i, \mu)), i = 1, \dots, M$

$$\sigma_M(t_i; \mu; u_{AD}^N(t_i, \mu)) = \sum_{m=1}^M \beta_m(\mu; u_{AD}^N(t_i, \mu)) q_m(t_i) = \sigma(t_i; \mu; u_{AD}^N(t_i, \mu))$$

where $u_{AD}^N(t_i; \mu) = \sum_{n=1}^N u_{AD,n}^N(\mu) u_{AD}^N(t_i; \mu_n)$

Non-affine Non-Linear decomposition : Wrap-up

- Build EIM approximations of non-linear terms using the initial finite element approximation
 - Generate databases of \mathcal{N} independent terms and \mathcal{N} dependent terms
 - Optimisation opportunities in EIM right hand side online step evaluation

$$\sigma(t_i; \mu; u^{\mathcal{N}|N}(\mu);)$$

by storing the (FEM or RB) basis functions associated to $u^{\mathcal{N}|N}(\mu)$ at the t_i .

- Build the generalized affine decomposition of the non-linear problem (Newton or Picard iterations)
- Compute the RB approximations (and associated reduced quantities) using the FEM approximation of the generalized affine problem
- Store all the \mathcal{N} -independent terms in database (we get rid of the finite element space dimension) and depend solely N , Q and the complexity of the generalized affine expansion.

Affine decomposition in Feel++

given Q_g , Q_j and Q_ℓ the EIM determines $(M_q^g)_{q=1,\dots,Q_g}$,
 $(M_q^j)_{q=1,\dots,Q_j}$ and $(M_q^\ell)_{q=1,\dots,Q_\ell}$ such that we can write :

$$g_{AD} \left({}^k u(\mu), v; \mu \right) = \sum_{q=1}^{Q_g} \sum_{m=1}^{M_q^g} \beta_g^{qm}(\mu; {}^k u(\mu)) g^{qm}(v) ,$$

$$j_{AD} \left(u(\mu), v; \mu; {}^k u(\mu) \right) = \sum_{q=1}^{Q_j} \sum_{m=1}^{M_q^j} \beta_j^{qm}(\mu; {}^k u(\mu)) j^{qm}(u(\mu), v) ,$$

and

$$\ell_{AD}(v; \mu) = \sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_q^\ell} \beta_\ell^{qm}(\mu) \ell^{qm}(v) .$$

The standard affine decomposition is in fact a special case of the generalized one.

We are able to extend the framework from a single non-linear equation to a system of non-linear equations.

- Monolithic view

- Support product of function spaces which might be different (scalar vs vectorial, approximation properties,...) or not

$$X = \prod_{i=1}^{N_{\text{spaces}}} X_i, \quad u = (u_1, \dots, u_{N_{\text{spaces}}}) \in X, \quad u_i \in X_i$$

- Support bilinear and linear forms on product of function spaces

$$a : X \times X \rightarrow \mathbb{R},$$

$$(u, v) \rightarrow a((u_1, \dots, u_{N_{\text{spaces}}}),(v_1, \dots, v_{N_{\text{spaces}}}))$$

- Requires advanced strategies for solvers/preconditioners

Applications

- V : electrical potential

$$\begin{cases} -\nabla \cdot (\sigma(T) \nabla V) = 0 \text{ on } \Omega \\ -\nabla \cdot (k(T) \nabla T) = \sigma(T) \nabla V \cdot \nabla V \text{ on } \Omega \end{cases}$$

Boundary conditions

- Applied potential
 - $V = 0$ on *Bottom*
 - $V = V_{Top}$ on *Top*
- Water / Glue electrically isolant
 - $-\sigma(T) \nabla V \cdot \mathbf{n} = 0$
- No thermic exchange with air
 - $-k(T) \nabla T \cdot \mathbf{n} = 0$
- Thermic exchange (h) with cooling water (T_w)
 - $-k(T) \nabla T \cdot \mathbf{n} = h(T - T_w)$

- T : temperature

Non-linearity

Electrical conductivity

$$\sigma(T) = \frac{\sigma_0}{1 + \alpha(T - T_0)}$$

- $\sigma_0 = \sigma(\text{ref} = 20^\circ\text{C})$
- $T_0 = 20^\circ\text{C}$
- $\alpha = \text{temperature coeff.}$

Thermal conductivity

$$k(T) = LT\sigma(T)$$

- $L = \text{Lorentz number}$

Electro-thermal model : Inputs/Outputs

Parameters

Material properties

- Electrical conductivity (σ_0)
- Temperature coeff (α)
- Lorentz number (L)

$$\mu = (\sigma_0, \alpha, L, V_{Top}, h, T_w)$$

Operating conditions

- Applied potential (V_D)
- Heat transfert coeff. (h)
- Water temperature (T_w)

Outputs

$$s(\mu) = \ell(V(\mu), T(\mu))$$

Possibilities for ℓ :

- Mean temperature in the domain
- Magnetic field on specific point (Biot & Savart's law)
- Power of the magnet

Variational formulation

$$\begin{cases} \nabla \cdot (\sigma(T) \nabla V) = 0 \text{ on } \Omega_V \\ \nabla \cdot (k(T) \nabla T) = \sigma(T) \nabla V \cdot \nabla V \text{ on } \Omega_T \end{cases}$$
$$\begin{cases} V = V_D \text{ on } D_V \\ -\sigma(T) \nabla V \cdot \mathbf{n} = V_N \text{ on } N_V \end{cases} \quad \begin{cases} -\sigma(T) \nabla T \cdot \mathbf{n} = 0 \text{ on } N_T \\ -\sigma(T) \nabla T \cdot \mathbf{n} = T_{R1}T + T_{R2} \text{ on } R_T \end{cases}$$

Electrical Potential

Find $V \in X \subset H_1(\Omega)$ such that $\forall \phi_V \in X$:

$$\int_{\Omega} \sigma(T) \nabla V \cdot \nabla \phi_V - \int_{D_V} \sigma(T) (\nabla V \cdot \mathbf{n}) \phi_V + \int_{D_V} \sigma(T) \left(\frac{\gamma}{h_s} V \phi_V - (\nabla \phi_V \cdot \mathbf{n}) V \right)$$
$$- \int_{D_V} \sigma(T) \left(\frac{\gamma}{h_s} V_D \phi_V - (\nabla \phi_V \cdot \mathbf{n}) V_D \right) = 0$$

Temperature

Find $T \in X \subset H_1(\Omega)$ such that $\forall \phi_T \in X$:

$$\int_{\Omega} k(T) \nabla T \cdot \nabla \phi_T + \int_{R_T} T_{R1} T \phi_T = \int_{\Omega} \sigma(T) \nabla V \cdot \nabla V \phi_T - \int_{R_T} T_{R2} \phi_T$$

Non-affine parameter dependance

Considering first term of electrical potential formulation :

$$a_v = \int_{\Omega} \sigma(T) \nabla V \cdot \nabla \phi_V = \int_{\Omega} \frac{\sigma_0}{1 + \alpha(T - T_0)} \nabla V \cdot \nabla \phi_V$$

As σ_0 and α are input parameters :

$$\nexists a_q^v, \theta_q^v \mid a_v(V, T; \mu) = \sum_q \Theta_q^v(\mu) a_q^v(V, T, \phi_V, \phi_T)$$

EIM : Empirical Interpolation Method

Build an affine approximation a_v^{aff} of a_v such that :

$$a_v^{aff} = \sum_q \Theta_q^v(\mu) a_q^v(V, T, \phi_V, \phi_T)$$

exact on a set of interpolation points $\{t_i\}$: $a_v^{aff}(t_i) = a_v(t_i) \forall i.$

Electrical potential

$$\int_{\Omega} \sigma(T) \nabla V \cdot \nabla \phi_V - \int_{D_V} \sigma(T) \left((\nabla V \cdot \mathbf{n}) \phi_V + \frac{\gamma}{h_s} V \phi_V - (\nabla \phi_V \cdot \mathbf{n}) V \right)$$
$$- \int_{D_V} \sigma(T) V_D \left(\frac{\gamma}{h_s} \phi_V - (\nabla \phi_V \cdot \mathbf{n}) \right) = 0$$

$$\sigma(T) = \frac{\sigma_0}{1 + \alpha(T - T_0)} \longrightarrow \sigma^{aff}(T) = \sum_{m_\sigma=0}^{M_\sigma} \beta_{m_\sigma}(\mu) q_{m_\sigma}(T)$$

Temperature

$$\int_{\Omega} k(T) \nabla T \cdot \nabla \phi_T + \int_{R_T} T_{R1} T \phi_T = \int_{\Omega} \sigma(T) \nabla V \cdot \nabla V \phi_T - \int_{R_T} T_{R2} \phi_T$$

$$k(T) = \sigma(T) L T \longrightarrow k^{aff}(T) = \sum_{m_k=0}^{M_k} \beta_{m_k}(\mu) q_{m_k}(T)$$

$$J(V, T) = \sigma(T) \nabla V \cdot \nabla V \longrightarrow J^{aff}(V, T) = \sum_{m_J=0}^{M_J} \beta_{m_J}(\mu) q_{m_J}(V, T)$$

Coupled formulation

Find $(V, T) \in X \times X \subset [H_1(\Omega)]^2$ such that $\forall (\phi_V, \phi_T) \in X \times X :$

$$a((V, T), (\phi_V, \phi_T); \mu) = f((\phi_V, \phi_T); \mu) \quad \forall (\phi_V, \phi_T) \in X \times X$$

Affine decomposition

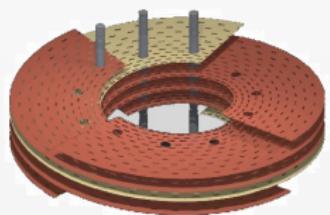
$$\boxed{\sum_{q_a=1}^{Q_a} \Theta_a^q(\mu) a^q((V, T), (\phi_V, \phi_T)) = \sum_{q_f=1}^{Q_f} \Theta_f^q(\mu) f^q((\phi_V, \phi_T))}$$

$$\begin{aligned} & \sum_{m_\sigma=1}^{M_\sigma} \beta_{m_\sigma}(\mu) \left[\int_{\Omega} q_{m_\sigma}(T) \nabla V \cdot \nabla \phi_V - \int_{D_V} q_{m_\sigma}(T) \left((\nabla V \cdot \mathbf{n}) \phi_V + \frac{\gamma}{h_s} V \phi_V - (\nabla \phi_V \cdot \mathbf{n}) V \right) \right] \\ & - \sum_{m_\sigma=1}^{M_\sigma} \beta_{m_\sigma}(\mu) V_D \int_{D_V} q_{m_\sigma}(T) \left(\frac{\gamma}{h_s} \phi_V - (\nabla \phi_V \cdot \mathbf{n}) \right) + \sum_{m_k=1}^{M_k} \beta_{m_k}(\mu) \int_{\Omega} q_{m_k}(T) \nabla T \cdot \nabla \phi_T \\ & + T_{R_1} \int_{R_T} T \phi_T = \sum_{m_J=1}^{M_J} \beta_{m_J}(\mu) \int_{\Omega} q_{m_J}(V, T) \phi_T - T_{R_2} \int_{R_T} \phi_T \end{aligned}$$

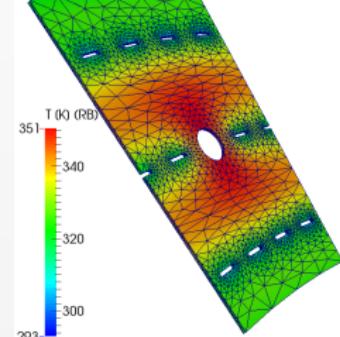
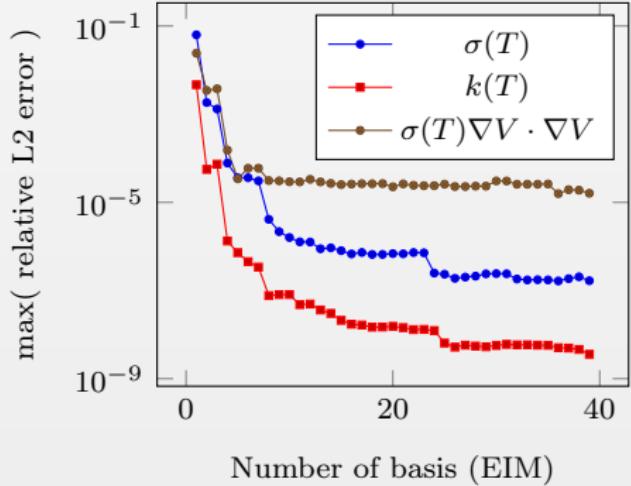
Reduced Electro-thermal model

From small towards large simulations

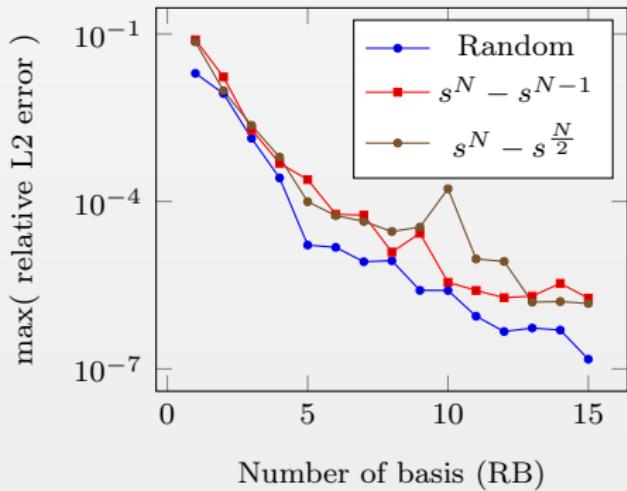
Bitter Magnet



EIM - Convergence study



CRB - Convergence study



Bitter Magnet - Parametric study

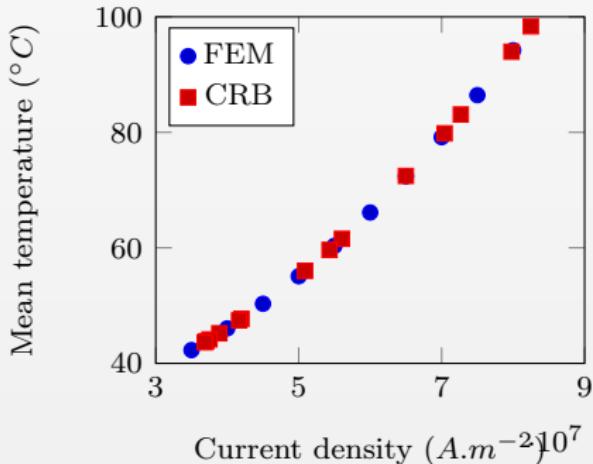
Objective

- Generate higher magnetic field
 - Increase current density
 $\mathbf{j} = \sigma \nabla V$
 - \Rightarrow Temperature increase !

Question

$\text{Max}(\mathbf{j})$ without thermal damages ?

- $\sigma_0 = 58 \times 10^6 S$
- $\alpha = 3.5 \times 10^{-3} K^{-1}$
- $L = 2.5^{-8}$
- $\mathbf{j} \in [30; 90].10^6 A.m^{-2}$
- $h = 80000 W.m^{-2}.K^{-1}$
- $T_w = 293 K$



40 °C to 60 °C : + 1 Tesla

Bitter Magnet - Sensitivity analysis

Inputs ranges (Uniform distribution)

- $\sigma_0 \in [5.5 \times 10^4, 6 \times 10^4] S$
- $\alpha \in [3.3^{-3}, 3.5 \times 10^{-3}] K^{-1}$
- $L \in [2.5^{-8}, 2.9 \times 10^{-8}]$
- $\mathbf{j} \in [60e+6, 70e+6] A.m^{-2}$
- $h \in [70000, 90000] W.m^{-2}.K^{-1}$
- $T_w \in [293, 313] K$

Temperature Range

Mean of outputs :

$$T = 332.25 K \approx 59.25^\circ C$$

Standard deviation : 6.03

⇒ Range for T :

$$[326.22; 338.28] K = [53.22; 65.28]^\circ C$$

Sobol indices

$$S_i = \frac{V(E[Y | X_i])}{V(Y)}$$

σ_0	:	0.022
α	:	3.6×10^{-5}
L	:	0.0042
j	:	0.16
h	:	0.044
T_w	:	0.77

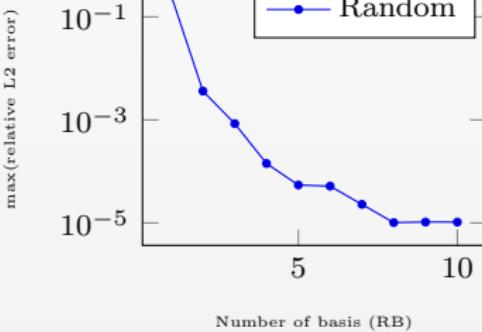
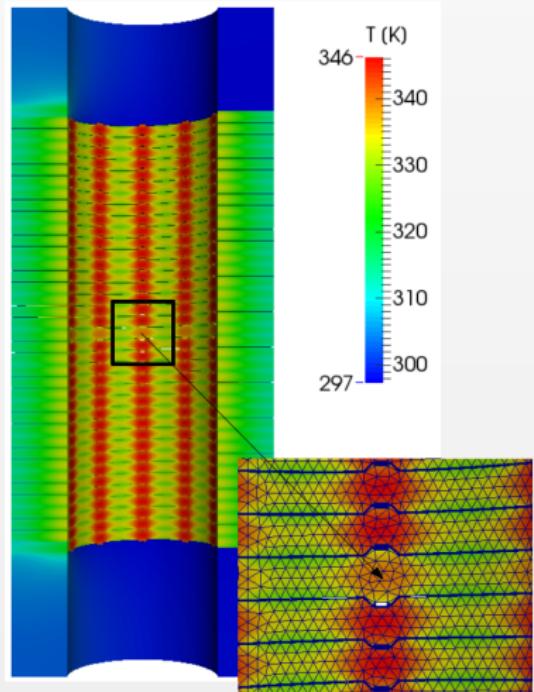
Quantiles

Determine $q(\gamma)$ such that
 $P(Y < q(\gamma)) > \gamma$

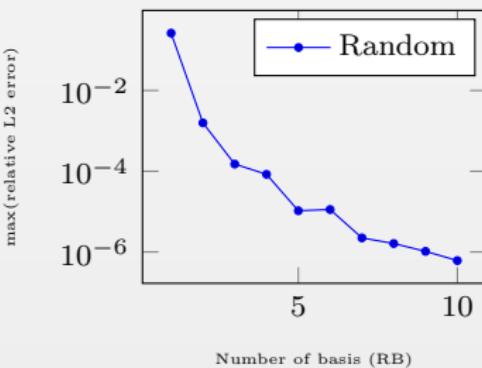
$$99.0\% : 343 K = 70^\circ C$$

$$80.0\% : 336.5 K = 63.5^\circ C$$

Helix magnet - Convergence study



Output error



Helix magnet - Performances

Simulation characteristics

- Number of dofs $\approx 2.4 \times 10^6$
 - 16 processors
 - Dofs / proc ≈ 155000
- Number of inputs : 6
 - $\sigma_0, \alpha, L, U, h, T_w$
- Reduced basis approximation :
 - EIM basis space : 10 basis
(sampling size = 100)
 - RB space : 25 basis
(sampling size = 1000)

PFEM time

- FEM approx. using affine decomposition
- 16 processors

Mean time $\approx 35min$

RB time

- RB approximation
- For any number of procs

Mean time $\approx 2min$

Gain factor ≈ 17

Ongoing work on electro-thermal model

- Continue investigations with large simulations
 - Increase number of basis
 - Analyse convergence (EIM, RB)
 - ...
- Work on error estimation for such a model
 - Error estimation for EIM approximation
 - Dealing with non-linearity

Towards full reduced model

- Add Linear Elasticity model
- Add Magnetostatic model

Conclusions

- Framework in place, need to add rigor in the nonlinear case if possible
- HPC is definitely required for (non-linear multiphysics) RB but it should be as seamless as possible
- Still some ingredients are missing
 - hp framework for eim and rb ;
 - Lego simulation (with domain decomposition) ;
- Another application : Aerothermal problems
(Navier-Stokes+Heat Transfer) ;
- Embed advanced analysis tools (automatic differentiation, polynomial chaos, ...)

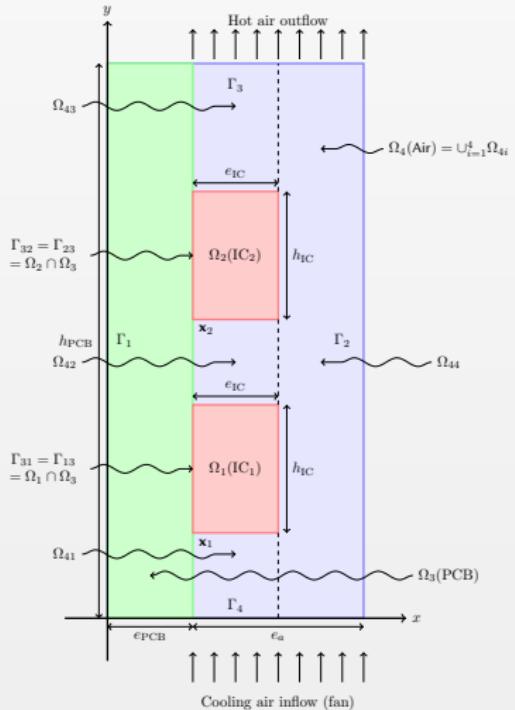
References

-  Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. (2004).
 An empirical interpolation method : application to efficient reduced-basis discretization of partial differential equations.
 Comptes Rendus Mathematique, 339(9) :667–672.

OPUS Heat Transfer Benchmark

▶ Back ◀

Thermal Testcase Description

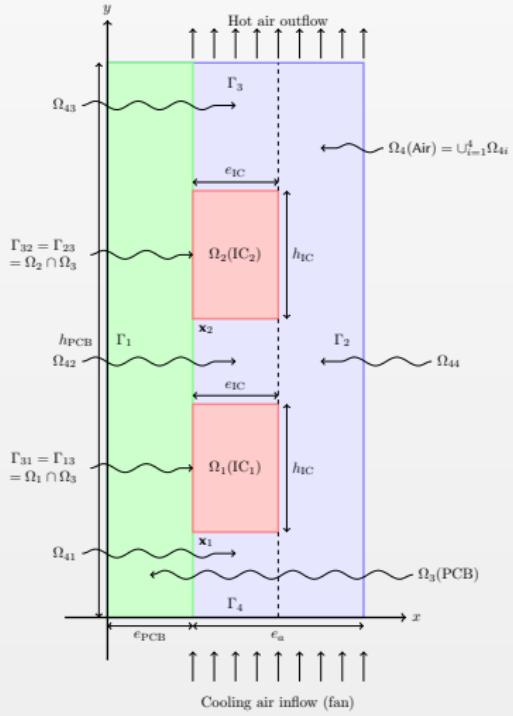


Overview

- Heat-Transfer with conduction and convection possibly coupled with Navier-Stokes
- Simple but complex enough to contain all difficulties to test the certified reduced basis
 - non symmetric, non compliant
 - steady/unsteady
 - physical and geometrical parameters
 - coupled models
- Testcase can be easily complexified

▶ Back ◀

Thermal Testcase Description



Heat transfer equation

$$\rho C_i \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \nabla \cdot (k_i \nabla T) = Q_i, \quad i = 1, 2, 3, 4$$

Inputs

$$\mu = \{e_a; k_{\text{IC}}; D; Q; r\}.$$

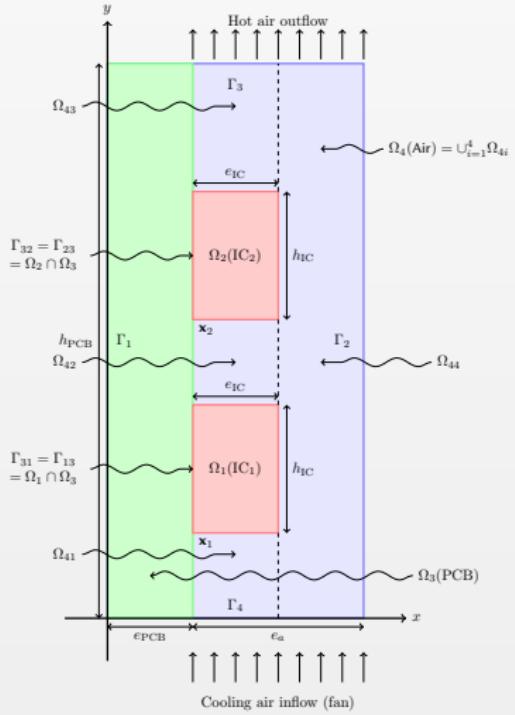
Outputs

$$s_1(\mu) = \frac{1}{e_{\text{IC}} h_{\text{IC}}} \int_{\Omega_2} T$$

$$s_2(\mu) = \frac{1}{e_a} \int_{\Omega_4 \cap \Gamma_3} T$$

Back ▲

Thermal Testcase Description



Fluid model

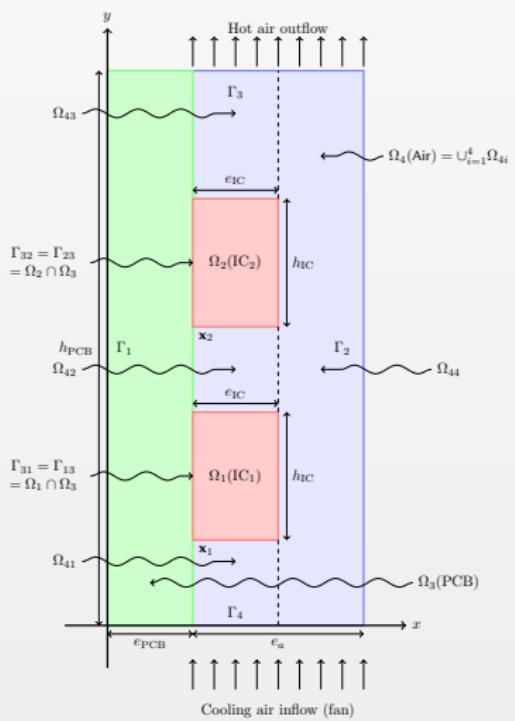
Poiseuille flow or Navier-Stokes flow

Boundary conditions

- on $\Gamma_3 \cap \Omega_3$, a zero flux
- on $\Gamma_3 \cap \Omega_4$, outflow
- on Γ_4 , $(0 \leq x \leq e_{\text{PCB}} + e_a, y = 0)$ temperature is set
- Γ_1 and Γ_2 periodic
- at interfaces between the ICs and PCB, thermal discontinuity (conductance)
- on other internal boundaries, the continuity of the heat flux and temperature

▶ Back ◀

Thermal Testcase Description



Finite element method

- $\mathbb{P}_k, k = 1, \dots, 4$ Lagrange elements
- Weak treatment of Dirichlet conditions
- CIP Stabilisation
- Locally Discontinuous FEM functions

Validation

- Comparison between Comsol(EADS) and Feel++
- Extensive testing and comparisons
- Implementation validated, ref. config. max rel error < 1%
- Diff. : mesh, stabilisation, Dirichlet

▶ Back ◀

Thermal Testcase Description

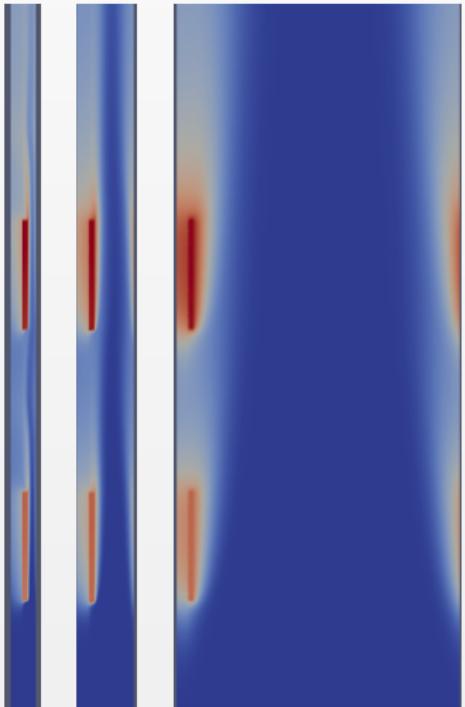
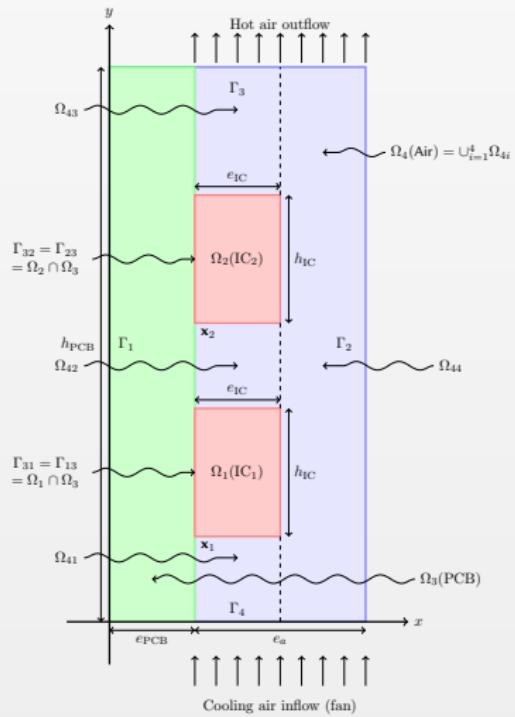


Figure : Temperature plot for $e_a \in \{2.5e-3, 8e-3, 5e-2\}$

Back ▶

Aerothermic : air conditioning/environment control systems

- Steady Navier-Stokes/Heat transfer problem
- 2 parameters : Grashof and Prandtl number
- Study the average temperature on heated surface with respect to parameters
- Application to aero thermal studies (buildings, cars, airplane cabins,...)

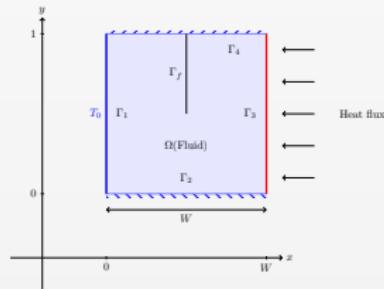
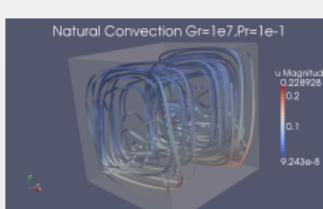


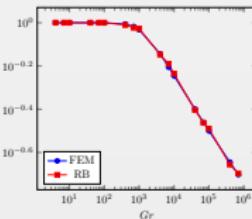
Figure : Geometry of the 2D model. Consider an extrusion of this geometry in 3D case is the extrusion in z axis of length 1.



(a) temperature isosurfaces
C. Prud'homme



(b) Stream lines
Aristote ROC & ROM



(c) T_{av} versus Gr

Aerothermic : air conditioning/environment control systems

- Steady Navier-Stokes/Heat transfer problem
- Coupling 2D/3D mesh based aerothermal model with ECS modelica model
- ECS and A/C design parameters to be optimized

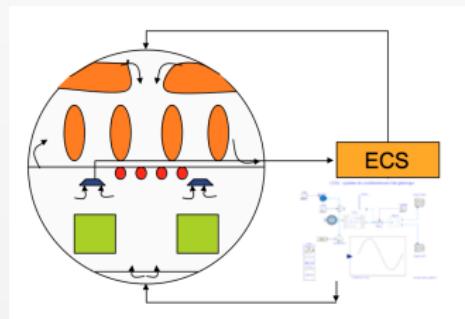
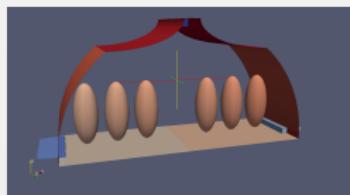
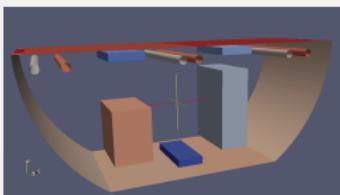


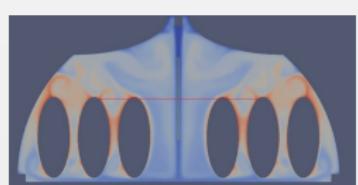
Figure : Design and sizing of thermal control of avionic bay and cabin



(a) Airplane Cabin



(b) Airplane Bay



(c) Temperature Field

High performance computing is required !

HiFiMagnet project

High Field Magnet Modeling

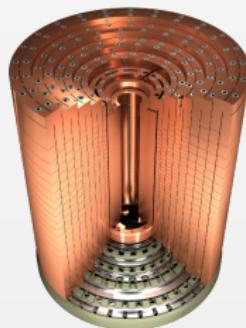
▶ Back ◀

Large scale user facility in France

- High magnetic field : from 24 T
- Grenoble : continuous magnetic field (36 T)
- Toulouse : pulsed magnetic field (90 T)

Application domains

- Magnetoscience
- Solide state physic
- Chemistry
- Biochemistry



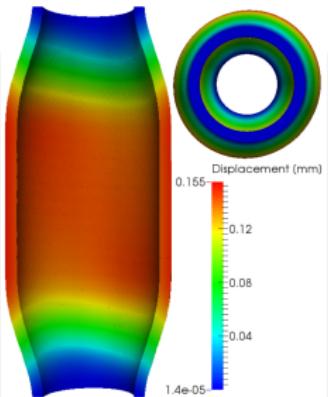
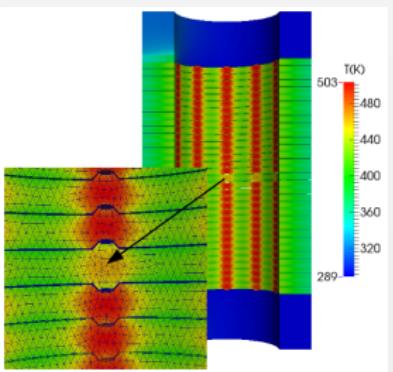
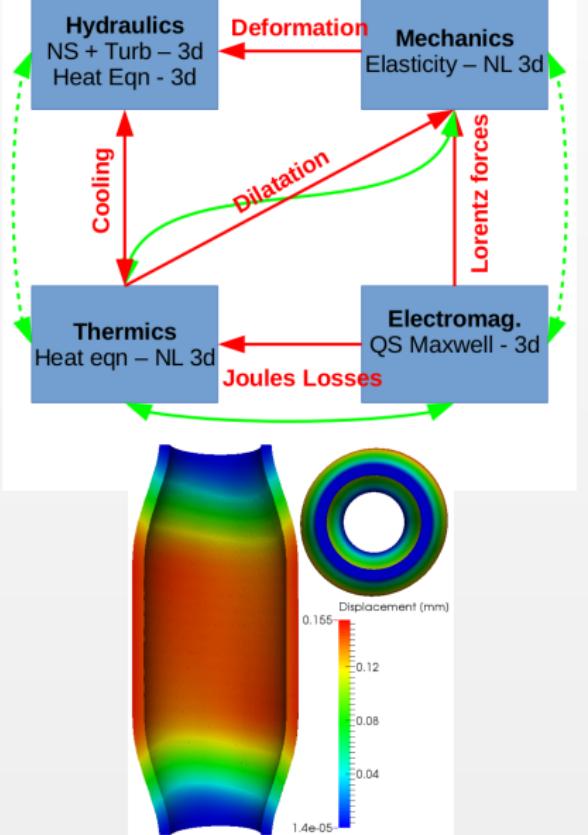
Magnetic Field

- Earth : $5.8 \cdot 10^{-4} T$
- Supraconductors : 24T
- Continuous field : 36T
- Pulsed field : 90T

Access

- Call for Magnet Time : 2 × per year
- ≈ 140 projects per year

High Field Magnet Modeling



▶ Back ◀

Why use Reduced Basis Methods ?

Challenges

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

Objective 1 : Fast

- Complex geometries
 - Large number of dofs
- Uncertainty quantification
 - Large number of runs

Objective 2 : Reliable

- Field quality
- Design optimization
 - Certified bounds
 - Reach material limits

▶ Back ◀

Context and Notations

- $(\mathcal{D}, \mathcal{B}, \mathbb{P})$ a probability space,
- $\mathcal{D} = \mathcal{D}^1 \times \dots \times \mathcal{D}^d \subset \mathbb{R}^d$,
- \mathcal{B} a σ -algebra on \mathcal{D} ,
- $\mathbb{P} = \bigotimes_{k=1}^d \mathbb{P}^k$ a probability measure,
- Parametric Linear Problem : find $u(\mu) \in \mathbb{R}^N$ such that

$$\hat{A}(\mu)u(\mu) = \hat{b}(\mu), \quad \forall \mu \in \mathcal{D}, \quad (3)$$

Non-Intrusive Low Rank Approximation

Principle

$$L_{\mathbb{P}}^2(\mathcal{D}; \mathbb{R}^N) \simeq \mathbb{R}^N \otimes L_{\mathbb{P}}^2(\mathcal{D}; \mathbb{R}) \simeq \mathbb{R}^N \otimes \left(\bigotimes_{k=1}^d L_{\mathbb{P}_k}^2(\mathcal{D}^k; \mathbb{R}) \right),$$

Goal : compute a low-rank approximation of the solution u

$$u_r \in \mathcal{W} = \mathbb{R}^N \otimes \mathcal{S}^1 \otimes \dots \otimes \mathcal{S}^d$$

With $\mathcal{S}^k = \text{span}\{\varphi_i^k\}_{i=1}^{n_k} \subset L_{\mathbb{P}_k}^2(\mathcal{D}^k; \mathbb{R})$ defined on the orthonormal family $\{\varphi_i^k\}_{i=1}^{n_k}$

Optimization Framework

Looking for an approximation of u by solving

$$\min_{v \in \mathcal{W}} J_{\mathcal{D}}(v) = \int_{\mathcal{D}} J(\mu; v(\mu)) \mathbb{P}(\mathrm{d}\mu).$$

where the quadratic functional $J : \mathcal{D} \times \mathbb{R}^N \rightarrow \mathbb{R}$ defined by

$$J(\mu; v) = \frac{1}{2} v^T A(\mu) v - v^T b.$$

Discrete formulation

$$J_{\mathcal{D}}(v) \approx J_{\mathcal{Z}}(v) = \sum_{z \in \mathcal{Z}} \omega_z J(\mu_z; v(\mu_z))$$

Alternating Minimization Algorithm

Resolution alternatively until convergence :

$$\min_{q \in \mathbb{R}^N} J_{\mathcal{Z}}(q \otimes \beta^1 \otimes \dots \otimes \beta^d), \quad \min_{\beta^1 \in \mathcal{S}^1} J_{\mathcal{Z}}(q \otimes \beta^1 \otimes \dots \otimes \beta^d), \dots, \quad \min_{\beta^d \in \mathcal{S}^d} J_{\mathcal{Z}}(q \otimes \beta^1 \otimes \dots \otimes$$

Algorithms

Algorithm 2 Alternating minimization algorithm.

Initialize β^k , $k = 1, \dots, d$. $\ell \leftarrow 0$. ($\ell \leq \ell_{\max}$) and ($q \otimes \beta^1 \otimes \dots \otimes \beta^d$ has converged) Compute q by solving Equation (28) $k = 1, \dots, d$ Compute β^k by solving Equation (??) $q \leftarrow \|\beta^k\|_k q$ $\beta^k \leftarrow \beta^k / \|\beta^k\|_k$ $\ell \leftarrow \ell + 1$

Algorithm 3 Greedy rank-one algorithm.

$u_0 \leftarrow 0$ $r \leftarrow 1$. ($r < r_{\max}$) and (u_r has converged) Compute v_{r+1} with previous Algorithm with $b = b_r$ $u_{r+1} \leftarrow u_r + v_{r+1}$ $r \leftarrow r + 1$

First Implementation In Feel++

Application to a 3-Parameters Heat Transfert Problem

Collaboration

- A. Nouy (ECN)
- L. Giraldi (ECN)
- C. Prud'homme (UDS/IRMA)
- JB Wahl (UDS/IRMA)

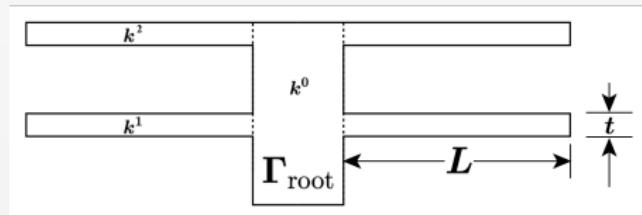


Figure : Geometry of the thermal fin

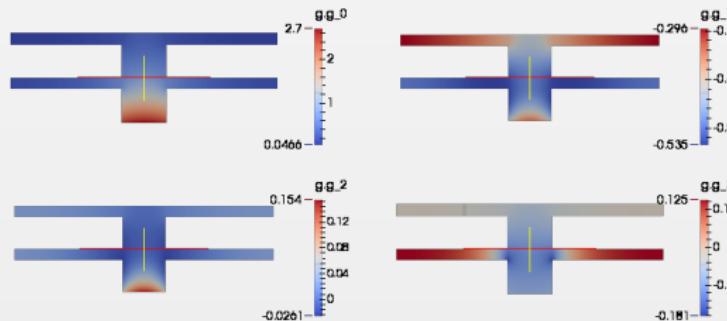


Figure : First modes obtained with rank-one greedy algorithm

Problem Setting

- Steady Heat Transfert
- 3 Parameters (heat transfert coefficients)
- Study of Average Temperature on Γ_{Root}