Advanced Reduced Order Methods: From POD to coupled Nitsche

Davide Baroli,
Stéphane Bordas,
Lars Beex
Jack Hale

Outline

Reliable and fast method: model reduction method

Given a solution set (a discrete manifold) $\mathcal{M}_h = \{u_h(\mu) \in V_h : \mu \in \mathcal{P}\}$ of high-fidelity problem.

$$a(u_h(\mu), v_h; \mu) = f(v_h; \mu) \forall v in V_h \tag{1}$$

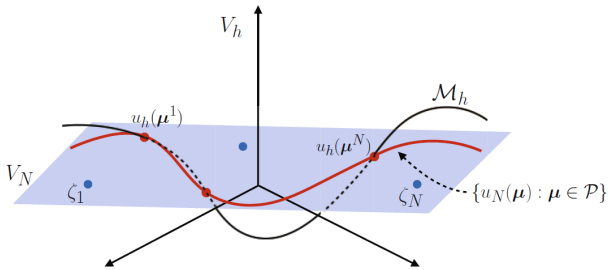


Figure: From "Reduced Basis Method for PDEs"(A.Quarteroni et al.)

The goal of reduced order methods is to find a few functions $V_N = \{\xi_i\}_{i=1}^N \subset V_h$ with $N << dim(V_h)$ such that for any parameter μ the solution of parametrized problem is approximated by:

$$u_h(\mu) = \sum_{i=1}^{N} u_N(\mu)^i \xi_i$$

where the coefficients $u_N(\mu)$ could be obtained via Galerkin reduced basis approximation:

$$[\xi_1|\dots|\xi_N]^T \mathbb{A}_h(\mu)[\xi_1|\dots|\xi_N] u_N = [\xi_1|\dots|\xi_N]^T \mathbf{f}_h(\mu)$$

with $\mathbb{A}_h$ and $\mathbf{f}_h$ are the discrete counterpart of the differential operator.

Accuracy fo reduced order method

Ansantz:

$$\|u(\mu) - u_{ROM}(\mu)\| \leq \|u(\mu) - u_h(\mu)\| + \|u_h(\mu) - u_{ROM}(\mu)\|$$

How efficient is the reduced order method representation? Let

$$E(M_h, V_N) = \sup_{u_h \in h} \inf_{u_{ROM} \in V_N} \|u_h - u_{ROM}\|$$

The Kolmogorov N-width of M_h in V_N is defined as:

$$d_N(M_h) = \inf_{V_{ROM} \subset V_h} \sup_{u_h \in M_h} \inf_{u_{ROM} \in V_N} \|u_h - u_{ROM}\|$$

If N-width decays as N increases then the discrete manifold is well approximated by a V_{ROM} with a small dimension.

Work-flow to put.:Offline-online.

Proper Orthogonal Decomposition

The goal of POD is to investigate the sampling of M μ in the parameter space ($\mathbb{P}$) and compress the discrete manifold of high-fidelity solution corresponding to sampled parameter, retaining the most energetic information.
Regarding the accurancy, the V_N generated by POD minimize the following quantity

$$\sqrt{\frac{1}{M}\sum_{\mu\in\mathbb{P}}\inf_{u_{ROM}\in V_N}\|u_h(\mu)-u_{ROM}\|_V^2}$$

Proper orthogonal decomposition algorithm

Given $\mu_1, \ldots, \mu_M \in \mathbb{P}$ and the high-fidelity approximation $u(\mu_1), \ldots, u(\mu_M) \in \mathcal{M}_h$.

- Construct the symmetric correlation matrix $C \in \mathbb{R}^{M \times M}$ defined by

$$C_{i,j} = \langle u(\mu_i), u(\mu_j) \rangle_V \, 1 \leq i, j \leq M$$

- Compute the eigenvalue-eigenvector pairs (λ_n, ζ_n) of the matric C such that $\|\zeta_n\|_{\ell^2(\mathbb{R}^M)} = 1$ and

$$C\zeta_n = \lambda_n \zeta_n$$

  with descending eigenvalue order.

- Reconstruct the POD basis

$$\xi_n = \frac{1}{\sqrt{\lambda_n}} [u(\mu_1) | \ldots | u(\mu_M)] \zeta_n$$

Application: Non-linear parametrized fkpp equation

Find $u(\mathbf{x}, t) \in H^1(\Omega, (0, T])$ such that

$$\begin{aligned} \partial_t u - \Delta u &= cu(1-u) && \text{in } \Omega = [0,3]^2 \times (0, T], \\ u &= 0 && \text{on } \partial\Omega \times (0, T], \\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}) && \forall \mathbf{x} \in \Omega, \end{aligned} \tag{2}$$

Following the standard Galerkin procedure of forming the L^2 inner product with test functions $v \in \hat{V}$ and performing integration by parts, the semi-linear weak residual formulation of (2) can be written as:

Find $u(x,t) \in V$ such that:

$$F(u,c;v) := (\partial_t u; v) + (\nabla u; \nabla v) - (cu; v) + (cu^2; v) = 0 \quad \forall v \in \hat{V}. \tag{3}$$

Picture here !! POD eigenvalue and some snapshot.

Which is one of the most computational effort in the "on-fly" model reduction?

- Online Galerkin projection when the parametrized problem require to "re-assemble" the matrix or rhs for online parameter (non-linear problem / non affine dependency).

Sequential reduced assembly

Given a subspace generated by a set of ortho-normal basis $V_{RB} = \{\xi_1, \ldots, \xi_N\}$, quadrature rule order m and a coarse mesh $\Omega_H := \mathcal{T}_H(\Omega)$:
Let start with the basis ξ_k $(k = 0)$ and a given tolerance tol.

1. Compute the error :

$$\eta(\xi_k) = \frac{|\int_{\Omega_H} \xi_k - \int_{\Omega_h} \xi_i|}{\int_{\Omega_h} \xi_k}$$

Sequential reduced assembly

Given a subspace generated by a set of ortho-normal basis $V_{RB} = \{\xi_1, \ldots, \xi_N\}$, quadrature rule order m and a coarse mesh $\Omega_H := \mathcal{T}_H(\Omega)$:
Let start with the basis ξ_k $(k = 0)$ and a given tolerance tol.

2. If $err > tol$ then
   Evaluate cell-wise the estimator:

$$\eta_{rel}(\xi_i, \Omega_H) = \int_K |I_m(\xi_k)| - \int_K |I_{m+1}(\xi_k)| \quad \forall\, K \in \Omega_H \tag{4}$$

Sequential reduced assembly

Given a subspace generated by a set of ortho-normal basis $V_{RB} = \{\xi_1, \ldots, \xi_N\}$, quadrature rule order m and a coarse mesh $\Omega_H := \mathcal{T}_H(\Omega)$:
Let start with the basis ξ_k $(k = 0)$ and a given tolerance tol.

3. Set $\alpha := \sum_{K \in \Omega_H} \eta_K(\xi_i, \Omega_H)$.
   If $\alpha > tol$ then
   For $i = 1, \ldots, N_{cells}$
   if $\sum_{j=1}^{i} \eta_{K_j}(\xi_i, \Omega_H) > diam(K_i) \cdot tol$:
   then refine(K_i) untill the $\alpha < tol$

Sequential reduced assembly

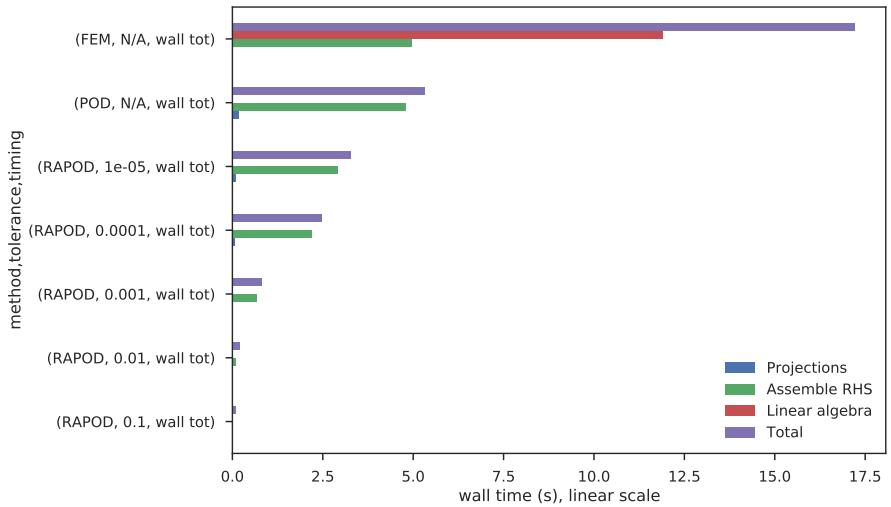
Given a subspace generated by a set of ortho-normal basis $V_{RB} = \{\xi_1, \ldots, \xi_N\}$, quadrature rule order m and a coarse mesh $\Omega_H := \mathcal{T}_H(\Omega)$:
Let start with the basis ξ_k $(k = 0)$ and a given tolerance tol.

4. Go to 1 and $k := k + 1$ and use as coarser mesh the last obtained.

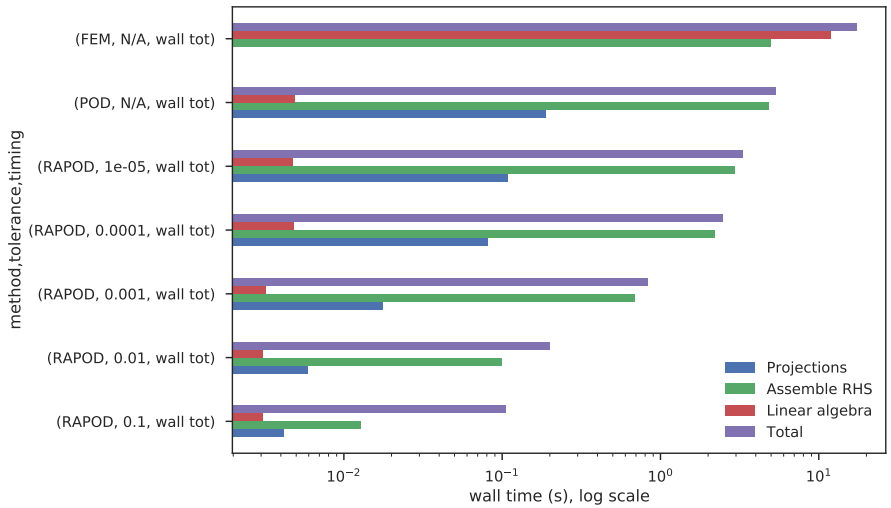
Application of sequential reduced assembly to POD: rapod

here picture.

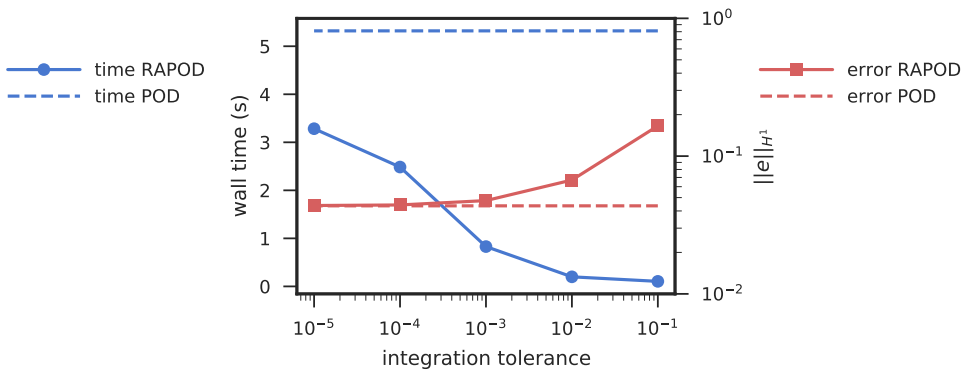
Computational demanding



Wall time


(FEM, N/A, wall tot)
(POD, N/A, wall tot)
(RAPOD, 1e-05, wall tot)
(RAPOD, 0.0001, wall tot)
(RAPOD, 0.001, wall tot)
(RAPOD, 0.01, wall tot)
(RAPOD, 0.1, wall tot)
method,tolerance,timing
Projections
Assemble RHS
Linear algebra
Total
10^{-2}
10^{-1}
10^{0}
10^{1}
wall time (s), log scale

Reliability of POD vs RAPOD



Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.
Initialize the Reduced Basis space:

```
reducedbasis = SLEPc.BV().create(Comm)
```

Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.

1. Compute high-fidelity solution $u_h(\mu_1)$ for μ_1

Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.

1. Compute high-fidelity solution $u_h(\mu_1)$ for μ_1
2. Set $\mathbf{V}_{rb} = span\{u_h(\mu_1)\}$.
   `reducedbasis.insertVec(index, function)`

Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.

1. Compute high-fidelity solution $u_h(\mu_1)$ for μ_1
2. Set $\mathbf{V}_{rb} = span\{u_h(\mu_1)\}$.
3. For each $\mu \in \mathcal{P}$
   Compute $u_{RB} = \sum_{i=1}^{N} u_N^i \xi_i(\mu)$ where ξ_i are the basis of $\mathbf{V}_{rb}$.
   Evaluate the error estimator $\eta(\mu) = \frac{(res_{RB}, M^{-1} res_{RB})}{\alpha_{LB}}$

Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.

1. Compute high-fidelity solution $u_h(\mu_1)$ for μ_1
2. Set $\mathbf{V}_{rb} = span\{u_h(\mu_1)\}$.
3. For each $\mu \in \mathcal{P}$
   Compute $u_{RB} = \sum_{i=1}^{N} u_N^i \xi_i(\mu)$ where ξ_i are the basis of $\mathbf{V}_{rb}$.
   Evaluate the error estimator $\eta(\mu) = \frac{(res_{RB}, M^{-1} res_{RB})}{\alpha_{LB}}$
4. Choose $\tilde{\mu} = arg\ max_{\mu \in \mathcal{P}} \eta(\mu)$

Reduced Basis technique: Greedy algorithm using SLEPc

Given tol and set of parameters $\{\mu_1, \ldots, \mu_n\}$.

1. Compute high-fidelity solution $u_h(\mu_1)$ for μ_1
2. Set $\mathbf{V}_{rb} = span\{u_h(\mu_1)\}$.
3. For each $\mu \in \mathcal{P}$
   Compute $u_{RB} = \sum_{i=1}^{N} u_N^i \xi_i(\mu)$ where ξ_i are the basis of $\mathbf{V}_{rb}$.
   Evaluate the error estimator $\eta(\mu) = \frac{(res_{RB}, M^{-1} res_{RB})}{\alpha_{LB}}$
4. Choose $\tilde{\mu} = arg\ max_{\mu \in \mathcal{P}} \eta(\mu)$
5. If $\eta(\tilde{\mu}) > tol$ and $V_{RB} = V_{RB} \cup \{u(\tilde{\mu})\}$
   `reducedbasis.orthogonalizeVec(function)`
   `reducedbasis.scaleColumn(norm)`

Online Galerkin Project with SLEPc

$$[\xi_1|\dots|\xi_N]^T \mathbb{A}_h(\mu)[\xi_1|\dots|\xi_N]u_N = [\xi_1|\dots|\xi_N]^T \mathbf{f}_h(\mu)$$

with $\mathbb{A}_h$ and $\mathbf{f}_h$ are the discrete counterpart of the differential operator.

```
reducedbasis.matProject(A, reducedbasis)
reducedbasis.dotVec(f)
```

Parametrized compressible linear elasticity

The linear compressible elasticity problem reads: find the displacement u such that

$$\begin{aligned} -div(\sigma(\mathbf{u}, \mu, \lambda)) &= f(\beta, \rho, c_i) \quad \text{in } \Omega \\ \mathbf{u} &= 0 \quad \text{on } \partial\Gamma_D \\ \sigma(\mathbf{u}, \mu, \lambda) \cdot \mathbf{n} &= \mathbf{g} \quad \text{on } \partial\Gamma_N \end{aligned}$$

Here the stress tensor σ is related to the displacement by Hooke's law:

$$\sigma(\mathbf{u}, \mu, \lambda) = 2\mu\epsilon(\mathbf{u}) + \lambda tr(\epsilon(\mathbf{u}))\mathbf{I} \quad \text{in } \Omega$$

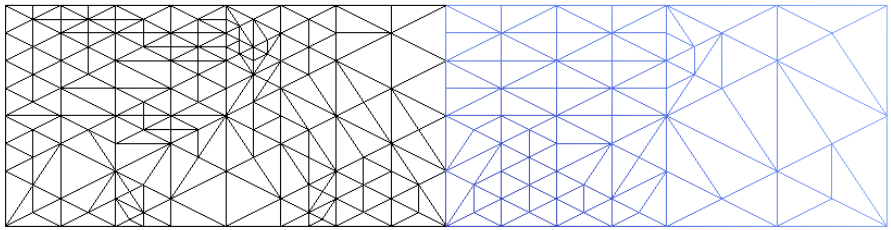
And the source is defined as : $f(\beta, \rho, c_i) = (0.0,\ -\rho \cdot 9.8 \cdot e^{-\sum_{i=1}^{3} c_i \cdot (x_i - \beta_i)^2})$

Locally reduced assembly

Due to local behavior of modes/basis which is sensitive to the zone of application of the external force/moments, we combine the domain-decomposition paradigm "divide-et-impera" with reduced assembly technique.
The algorithm consist in

- Decompose the physical domain in part (regular - ad hoc)
- Restrict each reduced basis on each subdomain, constructing the space $V_{RB,i}$ for $i = 1, \ldots, \mathrm{NbSubDomains}$
- Perform in each subdomain the reduced-integration
- Re-orthonormalization of each local space.

Non-conforming impera algorithm



Some domain decomposition technique: FETI-DP, BDDC, Mortar, Discontinuous Galerkin, InterNODES, Nitsche, ...
ROM + domain decomposition: Reduced Basis Element [RBE], DG-RBE, static condensation reduced basis element, Latin Multiscale

Why we choose Nitsche technique?

Pros.

- do not require any additional unknown
- has the same accurancy property of the underling discretization
- avoid of mesh burden near interface
- could treat also overlapping subdomains

Cons.
The stability of Nitsche method depends on the choose of a penalty scalar value. However, it is possible to estimate this constant using trace inequality (S. Claus et al.).

The Nitsche-type weak formulation associated to this problem is: Find a $u_h \in V_h^k$ such that

$$a_h(u_h, v_h) - b_h(u_h, v_h) - b_h(v_h, u_h) + j_h(u_h, v_h) = L_h(v_h) \quad \forall\, v_h \in V_h^k$$

The bilinear form a_h and b_h are defined as

$$a(u_h, v_h) = \sum_{i=1}^{2} \int_{\Omega_i} \sigma(u_h^i, mu_i, \lambda_i) : \epsilon(v_h^i) dX$$

$$b(u_h, v_h) = \int_{\Gamma} \{(\sigma(u_h, mu, \lambda))\}_{\kappa}, [\![(v \cdot n)]\!] d\Gamma$$

and the penalty-ghost form is

$$j(u_h, v_h) = \int_{\Gamma} \frac{\alpha}{\{h\}} [\![u^h]\!][\![v^h]\!] d\Gamma$$

Create decomposed mesh:

```
multimesh=dolfin.MultiMesh()
multimesh.add(mesh)
multimesh.build()
```

The MultiMeshForm for the first subdomain:

```
a = inner(sigma(u,mu0,lmbda0), grad(v))*dX
b_u= - inner(w_avg(sigma(u,mu0,lmbda0),kappa0,kappa1),
          tensor_jump(v,n))*dI
b_v=- inner(w_avg(sigma(v,mu0,lmbda0),kappa0,kappa1),
          tensor_jump(u,n))*dI
j= alpha/avg(h)*inner(jump(u), jump(v))*dI
```

The bilinear form a_h and b_h are defined as

$$a(u_h, v_h) = \sum_{i=1}^{2} \int_{\Omega_i} \sigma(u_h^i, mu_i, \lambda_i) : \epsilon(v_h^i) dX$$

$$b(u_h, v_h) = \int_{\Gamma} \{(\sigma(u_h, mu, \lambda))\}_{\kappa}, [\![(v \cdot n)]\!] d\Gamma$$

and the penalty-ghost form is

$$j(u_h, v_h) = \int_{\Gamma} \frac{\alpha}{\{h\}} [\![u^h]\!][\![v^h]\!] d\Gamma$$

where the penalty parameter is

$$\alpha = \frac{C_I(d, p) \cdot (E_1 \cdot E_2)}{E_1 + E_2}$$

with d is physical dimension and p the order of the polynomial Lagrange discretization space (Tomar remark).

The bilinear form a_h and b_h are defined as

$$a(u_h, v_h) = \sum_{i=1}^{2} \int_{\Omega_i} \sigma(u_h^i, mu_i, \lambda_i) : \epsilon(v_h^i) dX$$

$$b(u_h, v_h) = \int_{\Gamma} \{(\sigma(u_h, mu, \lambda))\}_{\kappa}, [\![(v \cdot n)]\!] d\Gamma$$

and the penalty-ghost form is

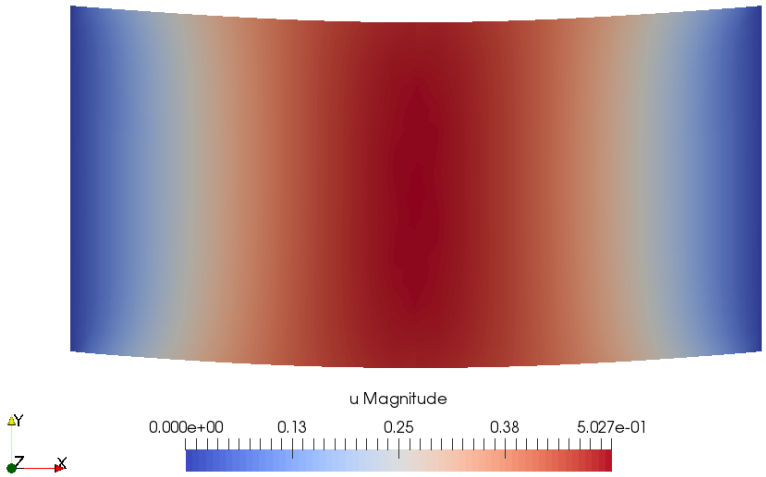
$$j(u_h, v_h) = \int_{\Gamma} \frac{\alpha}{\{h\}} [\![u^h]\!][\![v^h]\!] d\Gamma$$

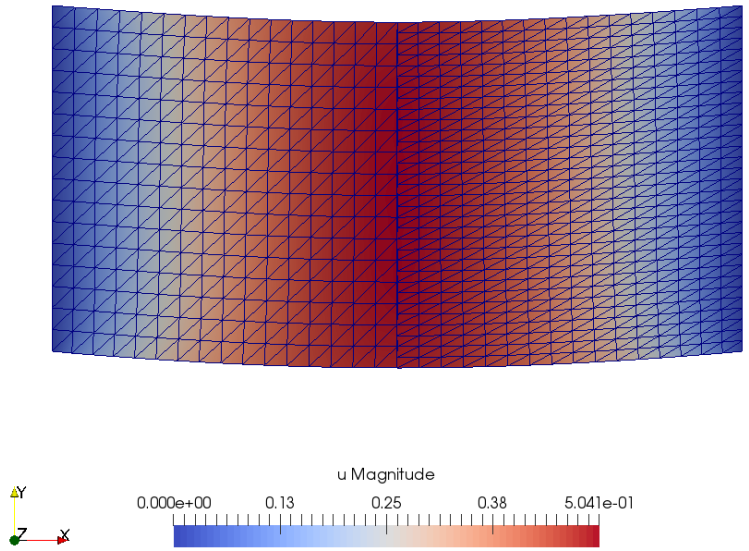
$$\{\mu u\}_{\kappa} = (\kappa_1 \mu_1 u^1 + \kappa_2 \mu_2 u^2)_{|\Gamma}$$

where the weights are

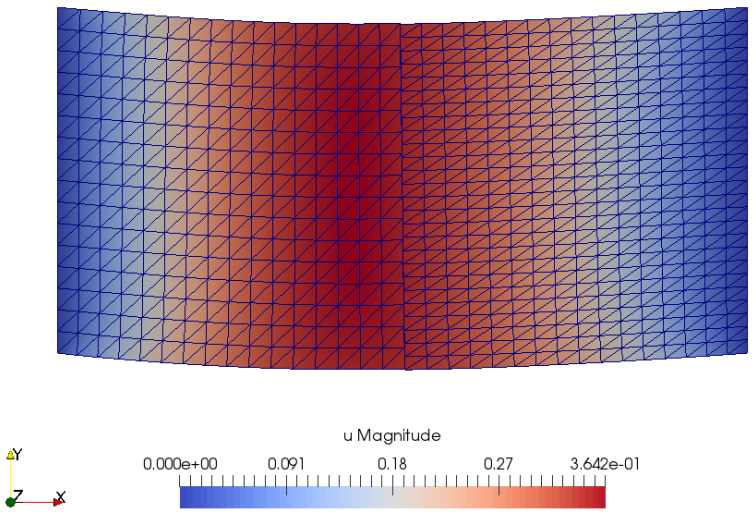
$$\kappa_1 = \frac{E_1}{E_1 + E_2}, \quad \kappa_2 = \frac{E_2}{E_1 + E_2}$$

Numerical test with single material

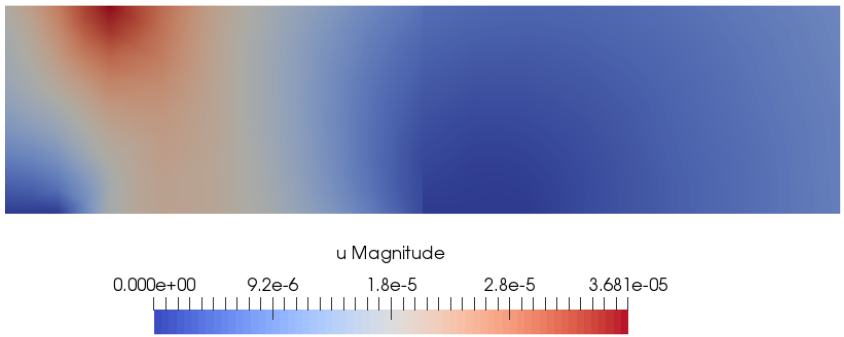


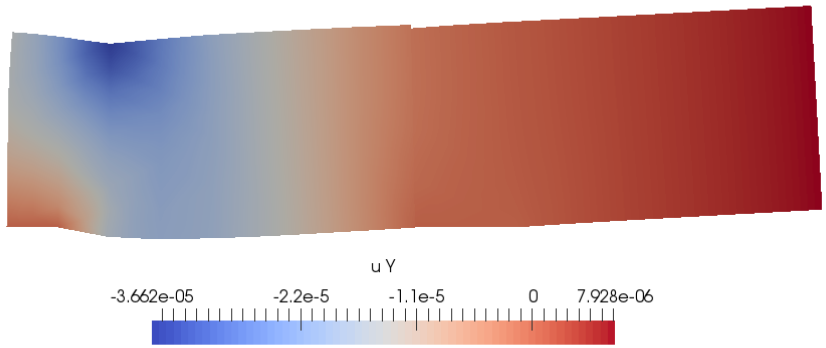


Numerical test with two materials



Numerical Validation Test:2D



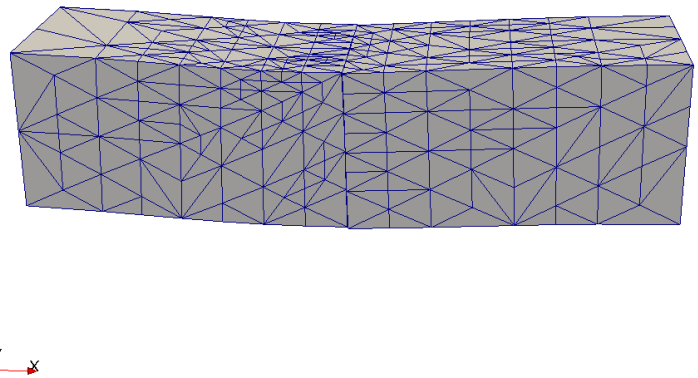


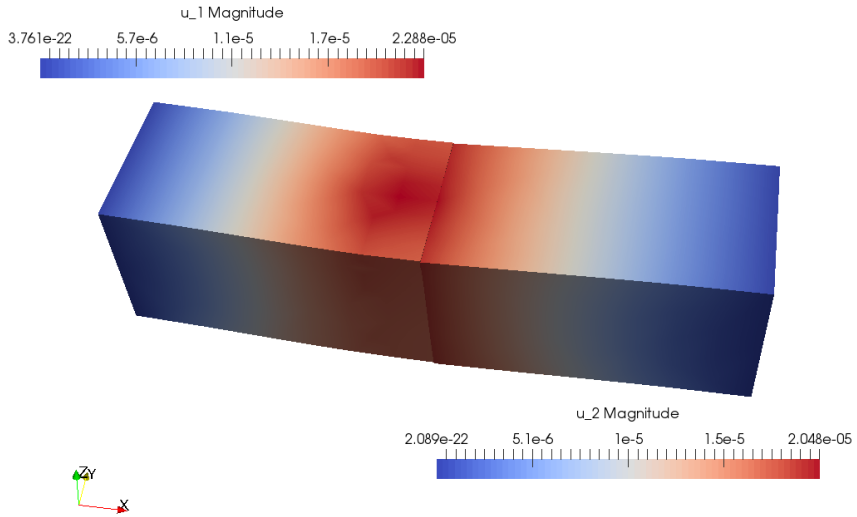
Number of dof of high-fidelity discretization:7442
Number of dof of reduced assembly discretization: 2796
Speed-up: x40 time with iRB+Nitsche
Speed-up: x10 time with reduced assembly.
Error: L_2-norm:=3.92e-07, H_1-norm=2.03e-06 (1st SubDomain)
L_2-norm:=2.23e-08, H_1-norm=5.47e-08 (2nd SubDomain)

RUES | RESEARCH UNIT IN ENGINEERING SCIENCES

UNIVERSITÉ DU LUXEMBOURG

Numerical Validation Test:3D





Number of dof of high-fidelity discretization:9600
Number of dof of reduced assembly discretization: 6953

RUES | RESEARCH UNIT IN ENGINEERING SCIENCES

UNIVERSITÉ DU LUXEMBOURG