Model reduction

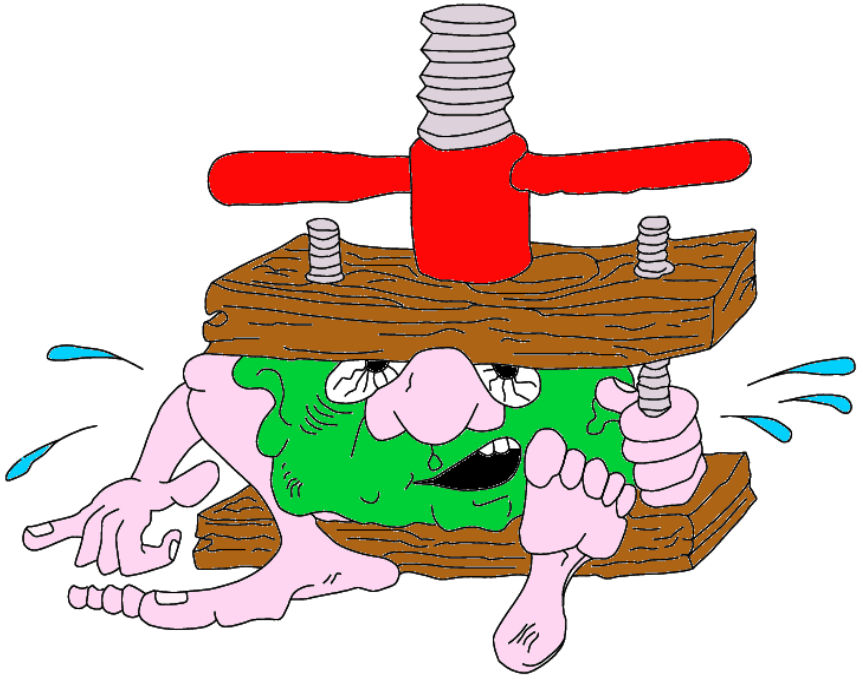
code_aster, Salome-Meca course material

EDF

GENERAL CONCEPTS

General concepts

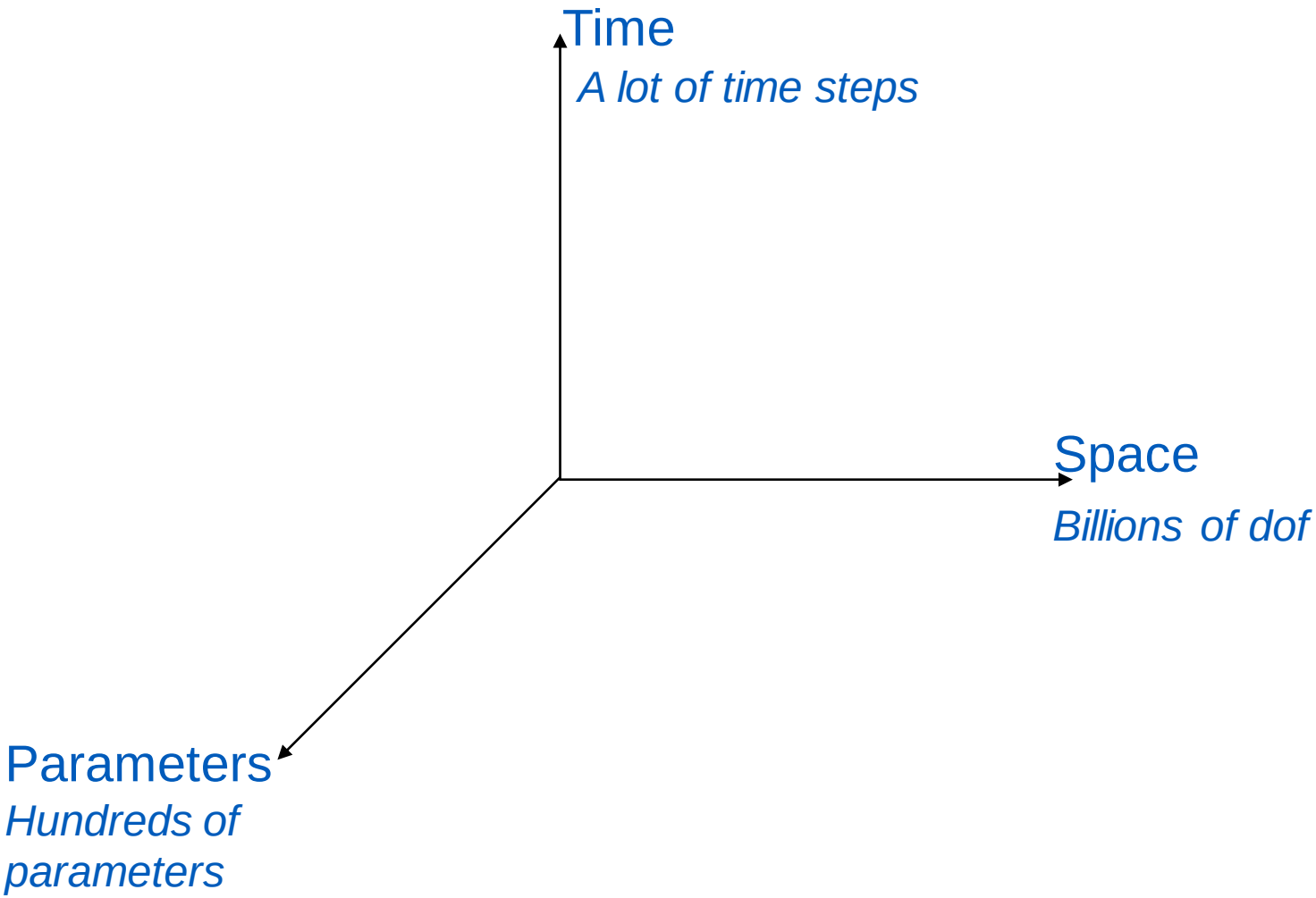
- Why reducing numerical models ?



- … because it’s too fat !

General concepts

- Definition of a « large » model



General concepts

- Lot of « large » problems:
  - Large ***space*** problems: large structures (nuclear plants, dams, …) and/or very fine meshes
  - Large ***time*** problems: long « physic » time (100,000 years) and/or non-linear problems (incremental methods)
  - Large ***parametric*** problems: physical and/or numerical parameters
- Very often the three problems are simultaneous

General concepts

- Reduced representation methods:
  - Reduced Order Methods (ROM) => **this document**
  - Dimension reduction methods (wavelets, ACP, ICA, etc.)
  - Classic response surfaces (linear, polynomial)
  - Metamodels (kriging, chaos polynomials, neural networks, splines...)
- Classification of these methods by type of integration of knowledge of the physical model:
  - Total integration (PDE): ROM, wavelets
  - Partial integration 1 (linearized PDE/PDE) : data assimilation
  - Partial integration 2 (Model hypothesis): kriging
  - None (no physical model hypothesis) : response surfaces

General concepts

- Classification of these methods by error control in relation to physics and hyp. model if … ***perfect*** model is assumed:
  - ROM, wavelets => error control "PDE(ROM) vs PDE" and " PDE(ROM) vs measures "
  - Data assimilation => error control " PDE (more or less linearized) vs measurements "
  - Kriging, splines... => error control " interpolation hypothesis / measurements "
- … ***imperfect*** model is assumed :
  - Bayesian methods with model error => complex error handling
- … ***no hypothesis*** is made about the model (we do not know it)
  - Response surface => the error cannot be controlled
  - Neural networks => the error cannot be controlled

SOME THEORETICAL ELEMENTS

General concepts

- Reducing a model:
  - High-fidelity model (DOM):

$$U = \sum_{k=0}^{\mathcal{N}} \boldsymbol{u}^k a^k$$

Sum of polynomial functions:
- Base functions: poor
- $\mathcal{N}$ is large

  - Reduced model (ROM):

$$U = \sum_{k=0}^{N} \boldsymbol{\psi}^k b^k$$

Sum of empiric functions:
- Base functions: rich
- N is small

Specialization

$N << \mathcal{N}$

Model reduction methods: algorithms to find "optimal" empirical modes + error control

General concepts

What can we reduce?

- A well posed problem (in Hadamard's sense):
  - There is only one solution to the problem
  - The solution continuously depends on the parameters

Coercive problems: elasticity, thermal, not too non-linear mechanic,…

- An easily configurable problem:
  - Material parameters
  - Loads
  - …

Contact, cracks: difficult issues !

Some theoretical elements

- Given a parameter space $\mathcal{P}$ with parameters $\boldsymbol{\mu} \in \mathcal{P}$
- Given a high-fidelity problem in algebraic form:

DOM problem: $\mathbb{A}_{\mathcal{N}}(\boldsymbol{\mu})\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) = \boldsymbol{f}_{\mathcal{N}}(\boldsymbol{\mu})$

with $\mathbb{A}_{\mathcal{N}}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$

- Construct a reduced model in algebraic form:

RB problem: $\mathbb{A}_{N}(\boldsymbol{\mu})\boldsymbol{u}_{N}(\boldsymbol{\mu}) = \boldsymbol{f}_{N}(\boldsymbol{\mu})$ $\qquad N \ll \mathcal{N}$

with $\mathbb{A}_{N}(\boldsymbol{\mu}) \in \mathbb{R}^{\boldsymbol{N}\times\boldsymbol{N}}$

$\boldsymbol{u}_{N}(\boldsymbol{\mu})$ is reduced basis solution

Some theoretical elements

- How to obtain the reduced based solution ?

$$\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \approx \boldsymbol{v}_{\mathcal{N}}(\boldsymbol{\mu}) = \mathbb{V}\boldsymbol{u}_{N}(\boldsymbol{\mu})$$

with $\mathbb{V} \in \mathbb{R}^{\boldsymbol{\mathcal{N}} \times \boldsymbol{N}}$

$\mathbb{V}$ is independent from $\boldsymbol{\mu}$

- Error (residual):

$$\boldsymbol{r}_{\mathcal{N}}(\boldsymbol{v}_{\mathcal{N}}; \boldsymbol{\mu}) = \boldsymbol{f}_{\mathcal{N}}(\boldsymbol{\mu}) - \mathbb{A}_{\mathcal{N}}(\boldsymbol{\mu})\boldsymbol{v}_{\mathcal{N}}(\boldsymbol{\mu})$$

We want $\boldsymbol{r}_{\mathcal{N}}(\boldsymbol{v}_{\mathcal{N}}; \boldsymbol{\mu})$ small for $\boldsymbol{\mu} \in \mathcal{P}$

$\|\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) - \mathbb{V}\boldsymbol{u}_{N}(\boldsymbol{\mu})\|$ is small

Some theoretical elements

- Convert DOM problem to RB problem:

Orthogonal projection: $${}^t\mathbb{V}\left(\boldsymbol{r}_{\mathcal{N}}(\boldsymbol{v}_{\mathcal{N}};\boldsymbol{\mu})\right) = 0$$

$$\mathbb{A}_N(\boldsymbol{\mu})\boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu})$$

$$\mathbb{A}_N(\boldsymbol{\mu}) = {}^t\mathbb{V}\mathbb{A}_{\mathcal{N}}(\boldsymbol{\mu})\,\mathbb{V}$$

$$\boldsymbol{f}_N(\boldsymbol{\mu}) = {}^t\mathbb{V}\boldsymbol{f}_{\mathcal{N}}(\boldsymbol{\mu})$$

Remark: we can use other projections (least squares or Petrov-Galerkin for instance)

Some theoretical elements

- Elements to construct:
  - How to construct $\mathbb{V}$
  - How to evaluate error $\|\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) - \mathbb{V}\boldsymbol{u}_N(\boldsymbol{\mu})\|$
- Methods to construct RB:
  - POD: Proper Orthogonal Decomposition
  - Greedy algorithm

A posteriori method: we need informations to construct RB

- Start from high-fidelity solutions
- Generate functions (RB)
- Compute unknown coefficients

Some theoretical elements

- General algorithm to construct RB:
  - A set of N selected parameters: $\boldsymbol{\mu_N} \in \mathcal{P}_N \subset \mathcal{P}$
  - Compute DOM solutions: $\{\boldsymbol{u}_{\mathcal{N}}(\mu_i), \ldots, \boldsymbol{u}_{\mathcal{N}}(\mu_N)\}$
  - Generate a set of N functions: $\{\boldsymbol{\Psi}_i, \ldots, \boldsymbol{\Psi}_N\}$ (this is RB !)
  - Orthonormalization (with standard scalar product)

The Reduced Space is generated by RB: $V_N = span\{\boldsymbol{\Psi}_i, \ldots, \boldsymbol{\Psi}_N\}$
With high fidelity space: $\mathrm{V} \supseteq V_N$

$$\boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{k=0}^{\boldsymbol{N}} \boldsymbol{\Psi}_k(.) \times a_k(\boldsymbol{\mu})$$

  - Compute unknown coefficients $a_k(\boldsymbol{\mu})$ by Galerkine projection

Some theoretical elements

- Main stages of POD/SVD decomposition
  - Construct snapshot matrix
  - SVD decomposition of snapshots matrix
  - Select most important vectors

Can be very expensive
=> Incremental versions

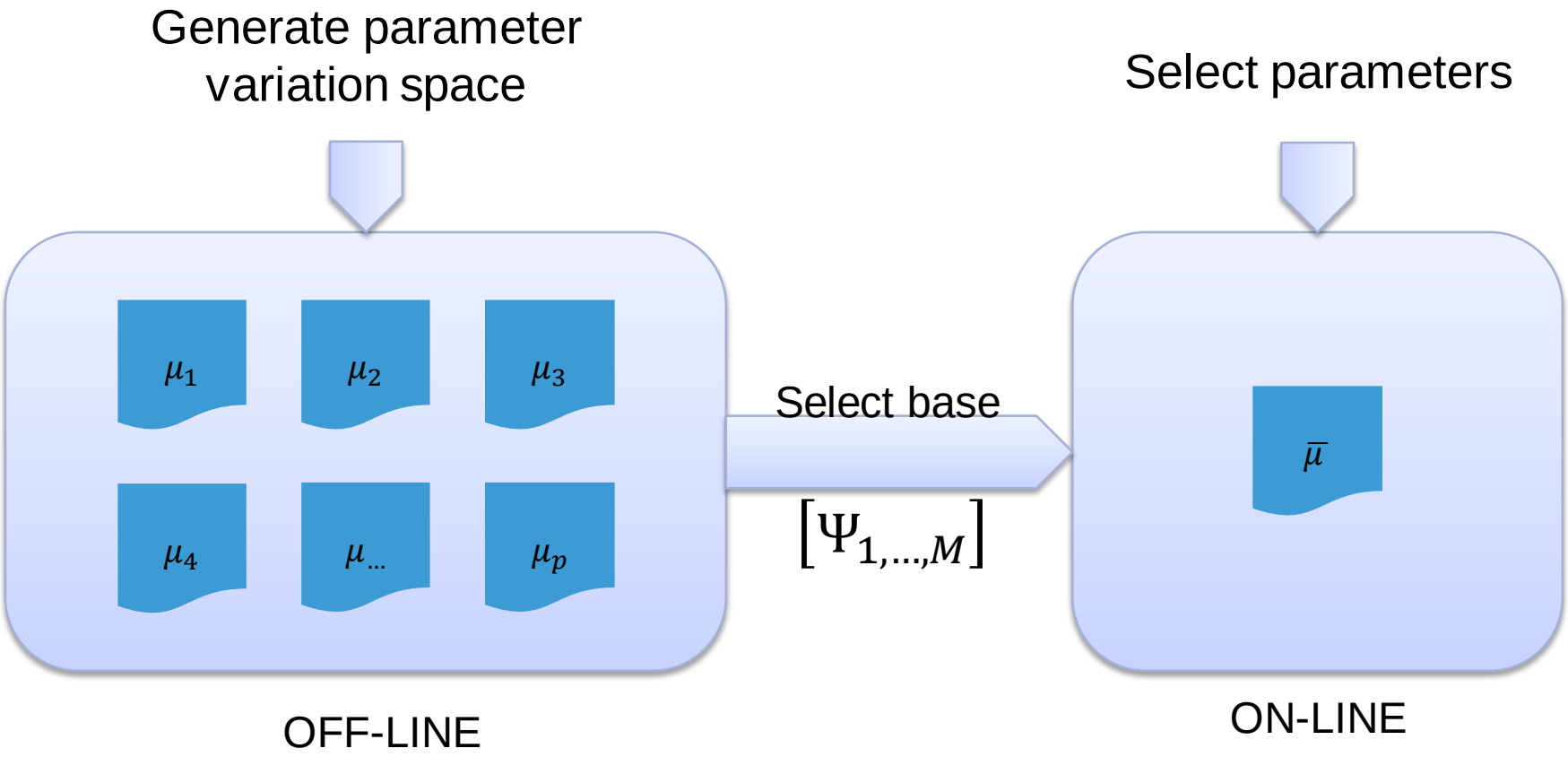
- Main stages of greedy algorithm
  - From initial set of parameters (training set)
  - Compute complete DOM problem with current parameter
  - Generate RB (with Gram-Schmidt algorithm for instance)
  - Compute ROM problem with new RB
  - Compute error between ROM and DOM problems
  - Add worst estimation to reduced basis

Need an error estimate
=> Using residual error

Some theoretical elements

- The off-line / on-line strategy
  - Create an empriric base (off-line):
    - Generate parameter variation space
    - Generate the base by an adapted algorithm
    - Prepare error estimation
  - Using empriric base (on-line):
    - Select right empiric base
    - Compute solution
- The off-line phase can be very expensive and does not exempt you from having access to the HPC features of code_aster !

Some theoretical elements



Some theoretical elements

- Previously, the problem was parametric but linear …
- … or affine:

$$\mathbb{A}_{\mathcal{N}}(\boldsymbol{\mu}) = \sum_{k=1}^{L} \mathbb{A}_{\mathcal{N}}^{k} \times \rho^{k}(\boldsymbol{\mu})$$

Easy to find "by hand" decomposition
Or using automatic method (as EIM)

POD DECOMPOSITION

POD decomposition

- Find a reduced model of non-linear problem:

$$\boldsymbol{X}_{DOM}(\boldsymbol{x},t) = \sum_{i=0}^{\mathcal{N}} \varphi_i(\boldsymbol{x}) \times q_i(t)$$

Unknown **X** by standard FE approach (DOM)

$$\boldsymbol{X}_{ROM}(\boldsymbol{x},t) = \sum_{i=0}^{N} \psi_i(\boldsymbol{x}) \times \gamma_i(t)$$

Unknown **X** by reduced approach (ROM)

$\boldsymbol{N << \mathcal{N}}$

$$\psi_i(\boldsymbol{x}) = \sum_{i=0}^{\mathcal{N}} \varphi_i(\boldsymbol{x}) \times V_{ik}$$

How to find V_{ik} and N ?

POD decomposition

- How to compute empiric base : basis that minimizes error η

$$\eta = \frac{\|\boldsymbol{X}_{ROM}(\boldsymbol{x},t) - \boldsymbol{X}_{DOM}(\boldsymbol{x},t)\|}{\|\boldsymbol{X}_{DOM}(\boldsymbol{x},t)\|}$$

Error with well-chosen norm

$$\|\boldsymbol{X}_{DOM}(\boldsymbol{x},t)\| = \int_0^{t_F} \int_\Omega \boldsymbol{X}_{DOM}\, \boldsymbol{X}_{DOM}\, d\Omega\, dt$$

POD decomposition

- The Proper Orthogonal Decomposition (POD):

$$\boldsymbol{X}_{ROM}(\boldsymbol{x},t) = \sum_{i=0}^{N} \psi_i(\boldsymbol{x}).\gamma_i(t) \qquad \text{with } \eta = \frac{\|\boldsymbol{X}_{ROM}(\boldsymbol{x},t) - \boldsymbol{X}_{DOM}(\boldsymbol{x},t)\|}{\|\boldsymbol{X}_{DOM}(\boldsymbol{x},t)\|} < \varepsilon$$

The functions ψ_i form an **orthonormal basis** for the canonical scalar product on $L_2(\Omega)$:

$$\langle \psi_i, \psi_j \rangle = \int_{\Omega} \psi_i.\psi_j.d\Omega = \delta_{ij}$$

We have:

$$\gamma_i(t) = \int_{\Omega} \boldsymbol{X}_{DOM}(\boldsymbol{x},t).\psi_i.d\Omega = \langle \boldsymbol{X}_{DOM}, \psi_i \rangle$$

POD decomposition

- Find $\psi_{i=1,N}$ solution of:

$$\min \sum_{k=1}^{n} \left\| \boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k) - \sum_{i=0}^{N} \langle \boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k), \psi_i \rangle . \psi_i \right\|_{L_2}^2$$

The $\boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k)$ are the ***snapshots*** of exact solution for time t_k

$\psi_{i=1,N}$ constitutes the POD basis of rank $N \leq n$

POD decomposition

- $\boldsymbol{Q} = [\boldsymbol{q}_i(t_1), \ldots, \boldsymbol{q}_i(t_n)]$ is the ***snapshot*** matrix for DOM problem
- The minimum problem from POD:

$$\min \sum_{k=1}^{n} \left\| \boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k) - \sum_{i=0}^{N} \langle \boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k), \psi_i \rangle . \psi_i \right\|_{L_2}^2$$

- Transforms in:

$\min \sum_{k=1}^{n} \left\| {}^{t}\widetilde{\boldsymbol{q}}(t_k) - \widetilde{\boldsymbol{V}}\left({}^{t}\widetilde{\boldsymbol{V}}\widetilde{\boldsymbol{q}}(t_k) \right) \right\|_F^2$ with $\|.\|_F$ the Frobenius norm

- Solution by Singular Value Decomposition (SVD) of $\boldsymbol{Q}$:

$\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{\Sigma}\ {}^{t}\boldsymbol{W}$ $\boldsymbol{\Sigma}$ is diagonal and contains singular values

POD decomposition

- Solution by Singular Value Decomposition (SVD) of $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{\Sigma}\ {}^{t}\boldsymbol{W}$$

$\boldsymbol{V}$ is a reduced basis for minimum problem

- Selection of « most » important singular values: from singular values $\sigma_{1,\dots,n}$ select $|\sigma_{i<n}| \geq \varepsilon_{SVD} \times max|\sigma_{1,\dots,n}|$

$$\boldsymbol{X}_{DOM}(\boldsymbol{x}, t_k) \approx \sum_{i=1}^{\boldsymbol{d}} \boldsymbol{V}[:, i]\Gamma_{ik}$$

With $\boldsymbol{\Gamma} = \left({}^{t}\boldsymbol{V}\boldsymbol{V} \right)^{-1}\ {}^{t}\boldsymbol{V}\boldsymbol{Q}$ the reduced coordinates

INCREMENTAL POD

Incremental POD

- POD by SVD should been very expensive
  - CPU time
  - Memory
- We can develop an ***incremental*** version of POD (based on Gram-Schmidt orthogonalization)
- From other POD basis or from nothing
- With well chosen parameters INCR-POD is equivalent to POD

Incremental POD

- $\boldsymbol{Q} = [\boldsymbol{q}_i(t_1), \ldots, \boldsymbol{q}_i(t_n)]$ is the ***snapshot*** matrix for DOM problem
- Initial basis $\boldsymbol{V}$ and reduced coordinates matrix $\boldsymbol{\Gamma}$
- Initial if $\boldsymbol{V} = \emptyset$
  - First mode: $\boldsymbol{V}^{(1)} = \frac{\boldsymbol{q}_1}{\|\boldsymbol{q}_1\|}$ and $\boldsymbol{\Gamma}^{(1)} = \|\boldsymbol{q}_1\|$

Incremental POD

- For $i = 2, n$
  - $\boldsymbol{\gamma}_i = \left({}^t\boldsymbol{V}\boldsymbol{V} \right)^{-1} {}^t\boldsymbol{V}\boldsymbol{q}_i$
  - $\delta\boldsymbol{q}_i = \boldsymbol{q}_i - \boldsymbol{V}\boldsymbol{\gamma}_i$
  - If $\|\delta\boldsymbol{q}_i\| \geq \varepsilon_{IPOD}\|\boldsymbol{q}_i\|$ then
    - $\boldsymbol{V} \leftarrow \left[\boldsymbol{V}, \frac{\delta\boldsymbol{q}_i}{\|\delta\boldsymbol{q}_i\|}\right]$ and $\boldsymbol{\Gamma} \leftarrow \left[\begin{pmatrix}\boldsymbol{\Gamma}\\ \boldsymbol{0}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\gamma}_i\\ \|\delta\boldsymbol{q}_i\|\end{pmatrix}\right]$
  - Else
    - $\boldsymbol{\Gamma} \leftarrow [\boldsymbol{\Gamma}, \boldsymbol{\gamma}_i]$

Incremental POD

- Final SVD
  - SVD on $\mathbf{\Gamma} \rightarrow \mathbf{\Gamma} = \mathbf{B}\boldsymbol{\Lambda}\ {}^{t}\boldsymbol{W}$
  - Select d first singular values, we have $\leftarrow \mathbf{B}[:, 1:d]$
  - New basis $\boldsymbol{V} \leftarrow \boldsymbol{V}\widetilde{\mathbf{B}}$ and $\mathbf{\Gamma} \leftarrow\ {}^{t}\widetilde{\mathbf{B}}\mathbf{\Gamma}$
- Two parameters:
  - ε_{IPOD}
  - d or ε_{SVD}

DISCRETE EMPIRIC INTERPOLATION METHOD

DEIM

- Considering non-linear parametric problem

$$\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})\boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu})$$

- Computing $\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})$

$$\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu}) = {}^t\mathbb{V}\mathbb{A}_{\mathcal{N}}(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})\,\mathbb{V}$$

  - Need to have an update of $\mathbb{A}_{\mathcal{N}}$ operator
  - Need to compute matrix-vector products « at the fly »

- We need to decrease size of $\mathbb{A}_{\mathcal{N}}(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})$! => DEIM

DEIM

- Considering non-linear parametric problem

$$\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})\boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu})$$

- Computing $\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})$

$$\mathbb{A}_N(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu}) = {}^t\mathbb{V}\mathbb{A}_{\mathcal{N}}(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})\,\mathbb{V}$$

  - Need to have an update of $\mathbb{A}_{\mathcal{N}}$ operator
  - Need to compute matrix-vector products « at the fly »

- We need to decrease size of $\mathbb{A}_{\mathcal{N}}(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{\mu})$! => DEIM

DEIM

- DEIM comes from Empirical Interpolation Method, approximation of non-linear function:

$$f(\boldsymbol{x};\boldsymbol{\mu}) \approx f_M(\boldsymbol{x};\boldsymbol{\mu}) = \sum_{m=1}^{M} \rho_m(\boldsymbol{x}).\gamma_m(\boldsymbol{\mu})$$

  - With $\boldsymbol{\mu}^m$ are the collocation points or « magic points »
  - Interpolation: $f(\boldsymbol{t}^i;\boldsymbol{\mu}) = f_M(\boldsymbol{t}^i;\boldsymbol{\mu})$ for all $i = 1,..,M$

- Construction with a greedy algorithm to find:
  - Magic points
  - Base functions $\rho_m(\boldsymbol{x})$

DEIM

- The DEIM works on ***discretized*** solution
  - Magic points are ***nodes*** of mesh
  - Base functions are from reduced base(s)
  - Classic greedy algorithm

MODEL REDUCTION IN CODE_ASTER

Model reduction in code_aster

- Three operators:
  - Construct reduced bases : `DEFI_BASE_REDUITE`
  - Construct Reduced Integration Domain (RID) by DEIM: `DEFI_DOMAINE_REDUIT`
  - Construct complete solution: `REST_REDUIT_COMPLET`

Model reduction in code_aster

- Construct a reduced base when you can construct linear system to reduce:
  - Build matrixes and second member with coefficients depending on parameters

```
coef = FORMULE(NOM_PARA=('INST','NEUT1'),VALE_C='''INST*3+NEUT1''')

RESH=DEFI_BASE_REDUITE(OPERATION='GLOUTON',
                       MATR_ASSE=(_F(MATRICE   = RIGI_1,
                                     COEF_R    = 1.0,),
                                  _F(MATRICE   = RIGI_2,
                                     FONC_R    = coef,),)
                       VECTEUR  = Force,
                       COEF_R   = 1.0,)
```

  - Set variation of parameters and use greedy mode by `OPERATION='GLOUTON'`

Model reduction in code_aster

- Construct a reduced base when you have complete results:
  - Compute linear or non-linear problem with THER_LINEAIRE, THER_NON_LINE or STAT_NON_LINE
  - Get results (EVOL_NOLI or EVOL_THER)
  - Post-treatment to have dual results (thermal fluxes FLUX_NOEU or stresses SIGM_NOEU)
- You have to manage domain of variation for parameters when compute results: material properties, loads
- Restrictions:
  - No AFFE_CHAR_THER or AFFE_CHAR_MECA => only AFFE_CHAR_CINE
  - Only 3D, with only displacement or temperature (no mixed models)
  - No dynamic, no contact

Model reduction in code_aster

- Using DEFI_BASE REDUITE:
  - Create a reduced base on primal unknown (temperature), by POD method with $\varepsilon_{POD} = 1.E-3$

```
thnl=THER_NON_LINE(…)
thnl=CALC_CHAMP(reuse=thnl,
                RESULTAT=thnl,
                THERMIQUE='FLUX_NOEU',);
base_p=DEFI_BASE_REDUITE(OPERATION= 'POD',
                         RESULTAT = thnl,
                         NOM_CHAM = 'TEMP',
                         TOLE_SVD = 1.E-3,);
```

  - Create a reduced base on primal unknown (temperature), by POD method with $d = 2$ (number of modes)

```
base_p=DEFI_BASE_REDUITE(OPERATION= 'POD',
                         RESULTAT = thnl,
                         NOM_CHAM = 'TEMP',
                         NB_MODE  = 2,);
```

Model reduction in code_aster

- Using `DEFI_BASE REDUITE`:
  - This operator create the new datastructure `mode_empi` which can been printed (in MED file for instance)
  - The reduced coordinates Γ are saved in `TABLE` attached to `mode_empi`

```
coorredd=RECU_TABLE(CO=base_p,
                    NOM_TABLE='COOR_REDUIT',);

IMPR_RESU(FORMAT='MED',UNITE=80,RESU=_F(RESULTAT=base_p))

IMPR_TABLE(TABLE=coorredp)
```

Model reduction in code_aster

- Using DEFI_BASE_REDUITE:
  - Create a reduced base on primal unknown (displacements), by incremental POD method with $\varepsilon_{SVD} = 1.E-3$ and $\varepsilon_{IPOD} = 1.E-4$

```
base_p=DEFI_BASE_REDUITE(OPERATION = 'POD_INCR',
                         TOLE      = 1.E-4,
                         RESULTAT  = stnl, NOM_CHAM  = 'DEPL',
                         TOLE_SVD  = 1.E-3,);
```

  - Enrich reduced base with new results (from another parameter set for instance):

```
base_p1=DEFI_BASE_REDUITE(OPERATION = 'POD',
                          RESULTAT  = thnl1, NOM_CHAM  = 'TEMP',
                          TOLE_SVD  = 1.E-3,);
base_p1=DEFI_BASE_REDUITE(OPERATION = 'POD_INCR',
                          reuse     = base_p1,
                          BASE      = base_p1,
                          TOLE      = 1.E-10,
                          RESULTAT  = thnl2, NOM_CHAM  = 'TEMP',
                          TOLE_SVD  = 1.E-3,);
```

Model reduction in code_aster

- Using reduced base in computation:
  - Get empiric base from file:

```
base_p=LIRE_RESU(TYPE_RESU  = 'MODE_EMPI',
                 FORMAT     = 'MED',
                 MODELE     = model,
                 UNITE      = 70,
                 FORMAT_MED =_F(NOM_CHAM_MED = 'base_p__TEMP',
                                NOM_CHAM     = 'TEMP',),)
```

  - Use it:

```
redu=THER_NON_LINE(…
                   METHODE='MODELE_REDUIT',
                   MODELE_REDUIT=_F(BASE_PRIMAL = base_p, ));
```

  - Same mesh, same model but not necessary same loads nor same material !
  - Produce a standard results datastructure (`evol_ther` or `evol_noli`)

Model reduction in code_aster

- « hyper » reduction, create the RID by DEIM method:
  - Construct empiric base on ***dual*** variables (thermal fluxes or stresses)

```
base_p=LIRE_RESU(FORMAT_MED =_F(NOM_CHAM_MED = 'base_p__TEMP',
                                NOM_CHAM     = 'TEMP',),);

base_d=LIRE_RESU(FORMAT_MED =_F(NOM_CHAM_MED = 'base_d__FLUX_NOEU',
                                NOM_CHAM     = 'FLUX_NOEU',),);

mesh=DEFI_DOMAINE_REDUIT(reuse=mesh,
                         MAILLAGE=mesh,
                         BASE_PRIMAL=base_p,
                         BASE_DUAL=base_d,
                         NOM_DOMAINE='RID',
                         GROUP_NO_INTERF='INF',);
```

  - Using only with «reuse mode => create new `GROUP_MA` and `GROUP_NO` in mesh

Model reduction in code_aster

- « hyper » reduction, create the RID by DEIM method:
  - Option: **minimum** domain (to require having specific elements in RID)

```
mesh=DEFI_DOMAINE_REDUIT(DOMAINE_MINI=_F());
```

  - Option: maximum domain

```
mesh=DEFI_DOMAINE_REDUIT(DOMAINE_MAXI=_F());
```

  - Option: extend domain with layers (default: 0)

```
mesh=DEFI_DOMAINE_REDUIT(NB_COUCHE_SUPPL = 1);
```

Model reduction in code_aster

- Using reduced base and RID in computation:
  - Get base:

```
base_p=LIRE_RESU(FORMAT_MED =_F(NOM_CHAM_MED = 'base_p__TEMP',
                                NOM_CHAM     = 'TEMP',),);
```

  - Create new model on RID

```
model_r=AFFE_MODELE(AFFE=_F(GROUP_MA='RID',),);
```

  - Restriction of base on RID:

```
base_t=DEFI_BASE_REDUITE(OPERATION='TRONCATURE',
                         MODELE_REDUIT = model_r,
                         BASE = base_p,
```

  - Use it:

```
redu=THER_NON_LINE(…
                   METHODE='MODELE_REDUIT',
                   MODELE_REDUIT=_F(BASE_PRIMAL = base_t,
                                    DOMAINE_REDUIT  = 'OUI',
                                    GROUP_NO_INTERF = 'INF',));
```

Model reduction in code_aster

- Using reduced base and RID in computation:
  - Careful ! Results are known only inside RID. To extend them, use Gappy-POD by `REST_REDUIT_COMPLET` operator:

```
reduR = REST_REDUIT_COMPLET(
    MODELE           = model,
    RESULTAT_REDUIT  = redu,
    BASE_PRIMAL      = base_p,
    REST_DUAL        = 'NON')
```

  - You can reconstruct ***dual*** fields

```
reduR = REST_REDUIT_COMPLET(
    MODELE           = model,
    RESULTAT_REDUIT  = redu,
    BASE_PRIMAL      = base_p,
    REST_DUAL        = 'OUI',
    BASE_DUAL        = base_d,
    GROUP_NO_INTERF  = 'INF',)
```

End of presentation

Is something missing or unclear in this document?

Or feeling happy to have read such a clear tutorial?

Please, we welcome any feedbacks about *code_aster* training materials.

Do not hesitate to share with us your comments on the *code_aster* forum dedicated thread.

POD decomposition

- How to solve this minimum ?

$$\boldsymbol{X}_{DOM}(\boldsymbol{x},t) = \sum_{i=0}^{\mathcal{N}} \varphi_i(\boldsymbol{x}).q_i(t) \qquad \boldsymbol{X}_{ROM}(\boldsymbol{x},t) = \sum_{i=0}^{N}\sum_{i=0}^{\mathcal{N}} \varphi_i(\boldsymbol{x}).V_{ik}.\gamma_i(t)$$

- The scalar product if « mass » matrix:

$$M_{ij} = \int_{\Omega} \varphi_i(\boldsymbol{x}).\varphi_j(\boldsymbol{x}).d\Omega \quad \xrightarrow{\text{Decomposition}} \quad \boldsymbol{M} = \boldsymbol{L}\ {}^{t}\boldsymbol{L}$$

- One can write:

$$\langle \psi_{k_1}, \psi_{k_2} \rangle = {}^{t}\boldsymbol{V}[:,k_1]\boldsymbol{M}\boldsymbol{V}[:,k_2] = {}^{t}\widetilde{\boldsymbol{V}}[:,k_1]\widetilde{\boldsymbol{V}}[:,k_2] \quad \text{with} \quad \widetilde{\boldsymbol{V}} = {}^{t}\boldsymbol{L}\boldsymbol{V}$$

$$\langle \boldsymbol{X}_{DOM}(\boldsymbol{x},t_k), \psi_i \rangle = {}^{t}\boldsymbol{q}\boldsymbol{M}\boldsymbol{V}[:,i] = {}^{t}\widetilde{\boldsymbol{q}}\widetilde{\boldsymbol{V}}[:,i]$$

Non-linear cases

- From decomposition of ROM and DOM solutions:

$$\boldsymbol{X}_{DOM}(\boldsymbol{x},t)=\sum_{i=0}^{\mathcal{N}}\varphi_i(\boldsymbol{x}).q_i(t) \qquad \boldsymbol{X}_{ROM}(\boldsymbol{x},t)=\sum_{i=0}^{N}\sum_{i=0}^{\mathcal{N}}\varphi_i(\boldsymbol{x}).V_{ik}.\gamma_i(t)$$

- The scalar product is « mass » matrix:

$$M_{ij}=\int_{\Omega}\varphi_i(\boldsymbol{x}).\varphi_j(\boldsymbol{x}).d\Omega \quad \xrightarrow{\text{Decomposition}} \quad \boldsymbol{M}=\boldsymbol{L}\,{}^{t}\boldsymbol{L}$$

- One can write:

$$\langle\psi_{k_1},\psi_{k_2}\rangle = {}^{t}\boldsymbol{V}[:,k_1]\boldsymbol{M}\boldsymbol{V}[:,k_2] = {}^{t}\widetilde{\boldsymbol{V}}[:,k_1]\widetilde{\boldsymbol{V}}[:,k_2] \quad \text{with} \quad \widetilde{\boldsymbol{V}} = {}^{t}\boldsymbol{L}\boldsymbol{V}$$

$$\langle\boldsymbol{X}_{DOM}(\boldsymbol{x},t_k),\psi_i\rangle = {}^{t}\boldsymbol{q}\boldsymbol{M}\boldsymbol{V}[:,i] = {}^{t}\widetilde{\boldsymbol{q}}\widetilde{\boldsymbol{V}}[:,i] \quad \text{with} \quad \widetilde{\boldsymbol{q}} = {}^{t}\boldsymbol{L}\boldsymbol{q}$$