Model order reduction for speeding up computational homogenisation methods of type FE^2

Olivier Goury, Pierre Kerfriden, Stéphane Bordas
Cardiff University

CARDIFF
UNIVERSITY
PRIFYSGOL
CAERDYDD

Outline

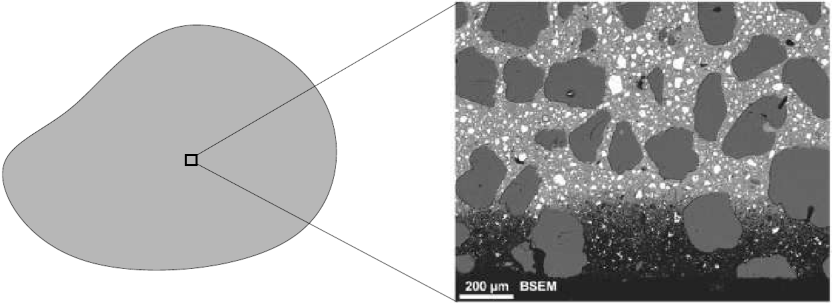
Introduction
- Heterogeneous materials
- Computational Homogenisation

Model order reduction in Computational Homogenisation
- Proper Orthogonal Decomposition (POD)
- System approximation
- Results

Conclusion

Heterogeneous materials

Many natural or engineered materials are **heterogeneous**



- Homogeneous at the macroscopic length scale
- Heterogeneous at the microscopic length scale

Heterogeneous materials

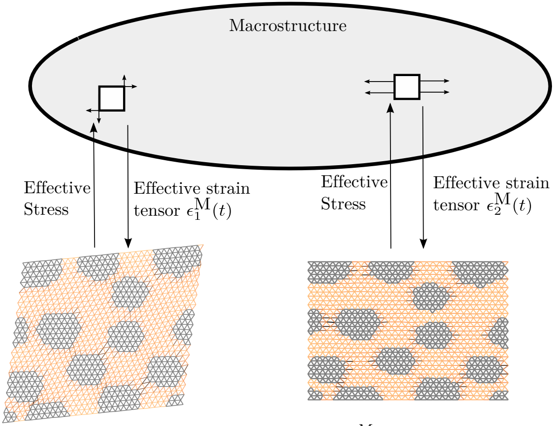
Need to model the macro-structure while taking the micro-structures into account

$\Longrightarrow$ better understanding of material behaviour, design, etc..

Two choices:

- Direct numerical simulation: brute force!
- Multiscale methods: when modelling a non-linear materials
  $\Longrightarrow$ Computational Homogenisation

Semi-concurrent Computational Homogenisation (FE^2, ...)



Macrostructure

Effective Stress

Effective strain tensor $\epsilon_1^{\mathrm{M}}(t)$

Effective Stress

Effective strain tensor $\epsilon_2^{\mathrm{M}}(t)$

$$\mathbf{u_1}(t) = \epsilon_1^M(t)(\mathbf{x} - \bar{\mathbf{x}}) + \tilde{\mathbf{u_1}}(t)$$

$$\mathbf{u_2}(t) = \epsilon_2^M(t)(\mathbf{x} - \bar{\mathbf{x}}) + \tilde{\mathbf{u_2}}(t)$$

Problem

- For non-linear materials: Have to solve a RVE boundary value problem at each point of the macro-mesh where it is needed. Still expensive!
- Need parallel programming

Strategy

- Use model order reduction to make the solving of the RVE boundary value problems computationally achievable
- Linear displacement:

$$\boldsymbol{\epsilon}^{\mathrm{M}}(t) = \begin{pmatrix} \epsilon_{xx}(t) & \epsilon_{xy}(t) \\ \epsilon_{xy}(t) & \epsilon_{yy}(t) \end{pmatrix}$$

$$\mathbf{u}(t) = \boldsymbol{\epsilon}^{\mathrm{M}}(t)(\mathbf{x} - \bar{\mathbf{x}}) + \tilde{\mathbf{u}} \quad \text{with} \quad \tilde{\mathbf{u}}_{|\Gamma} = \mathbf{0}$$

Fluctuation $\tilde{\mathbf{u}}$ approximated by: $\tilde{\mathbf{u}} \approx \sum_i \boldsymbol{\phi_i} \alpha_i$

Projection-based model order reduction

The RVE problem can be written:

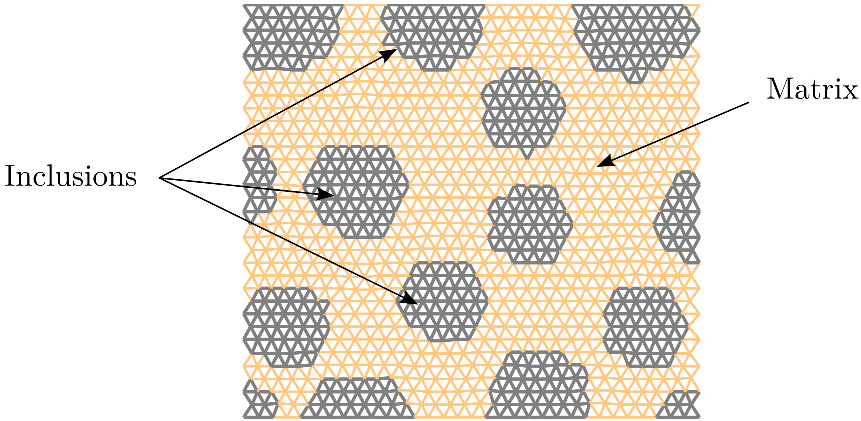
$$\underbrace{\underline{\mathbf{F}}_{\mathrm{int}}(\underline{\tilde{\mathbf{u}}}(\epsilon^{\mathrm{M}}(t)), \epsilon^{\mathrm{M}}(t))}_{\text{Non-linear}} + \underline{\mathbf{F}}_{\mathrm{ext}}(\epsilon^{\mathrm{M}}(t)) = \underline{\mathbf{0}} \qquad (1)$$

We are interested in the solution $\underline{\tilde{\mathbf{u}}}(\epsilon^{\mathrm{M}})$ for many different values of $\epsilon^{\mathrm{M}}(t \in [0, T]) \equiv \epsilon_{xx}, \epsilon_{xy}, \epsilon_{yy}$.

Projection-based model order reduction assumption:

Solutions $\underline{\tilde{\mathbf{u}}}(\epsilon^{\mathrm{M}})$ for different parameters ϵ^{M} are contained in a space of small dimension $span((\phi_{\boldsymbol{i}})_{i \in [\![1,n]\!]})$

RVE boundary value problem



Proper Orthogonal Decomposition (POD)

How to choose the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$?

Proper Orthogonal Decomposition (POD)

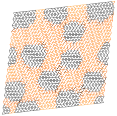
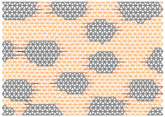
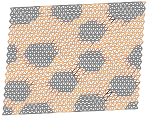
How to choose the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$?

- “Offline“ Stage $\equiv$ Learning stage : Solve the RVE problem for a certain number of chosen values of ϵ^M

Proper Orthogonal Decomposition (POD)

How to choose the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$?

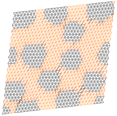
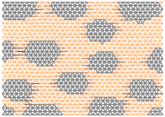
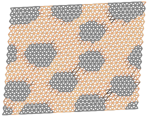
- “Offline“ Stage $\equiv$ Learning stage : Solve the RVE problem for a certain number of chosen values of ϵ^M
- We obtain a base of solutions (the snapshot): $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{n_S}) = \mathbf{S}$

 ,  ,  , ...

Proper Orthogonal Decomposition (POD)

How to choose the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$?

- “Offline“ Stage $\equiv$ Learning stage : Solve the RVE problem for a certain number of chosen values of ϵ^M
- We obtain a base of solutions (the snapshot): $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{n_S}) = \mathbf{S}$

 ,  ,  , ...

- That snapshot may be large and have linearly dependent components $\Rightarrow$ Need to extract the core information from it

- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle . \rangle}(\mathbf{\Phi}) = \sum_{\mu \in \mathcal{P}^{\mathrm{s}}} \left\| \mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}} . \left\langle \phi_{\mathbf{k}}, \mathbf{u}_i \right\rangle \right\|^2 \tag{2}$$

with the constraint $\left\langle \phi_{\mathbf{i}}, \phi_{\mathbf{j}} \right\rangle = \delta_{ij}$

- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle\cdot\rangle}(\mathbf{\Phi}) = \sum_{\mu\in\mathcal{P}^{\mathrm{s}}} \|\mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}}.\left\langle \phi_{\mathbf{k}}, \mathbf{u}_i \right\rangle \|^2 \tag{2}$$

  with the constraint $\left\langle \phi_{\mathbf{i}}, \phi_{\mathbf{j}} \right\rangle = \delta_{ij}$

- What is the quantity of interest here? Which scalar product and norm should we choose?

- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \mathbf{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle . \rangle}(\mathbf{\Phi}) = \sum_{\mu \in \mathcal{P}^{\mathrm{s}}} \|\mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}} . \langle \phi_{\mathbf{k}}, \mathbf{u}_i \rangle \|^2 \qquad (2)$$

with the constraint $\left\langle \phi_{\mathbf{i}}, \phi_{\mathbf{j}} \right\rangle = \delta_{ij}$

- What is the quantity of interest here? Which scalar product and norm should we choose?
- The canonic scalar product? $\Rightarrow \quad \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$

- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \boldsymbol{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle . \rangle}(\boldsymbol{\Phi}) = \sum_{\mu \in \mathcal{P}^{\mathrm{s}}} \| \mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}} . \langle \phi_{\mathbf{k}}, \mathbf{u}_i \rangle \|^2 \tag{2}$$

  with the constraint $\left\langle \phi_{\mathbf{i}}, \phi_{\mathbf{j}} \right\rangle = \delta_{ij}$

- What is the quantity of interest here? Which scalar product and norm should we choose?
- The canonic scalar product? $\Rightarrow \quad \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$
- Some kind of energy scalar product makes more sense: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{K}_{\{\tau | \tau < t\}} \, \mathbf{y}$

- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \boldsymbol{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle\cdot\rangle}(\boldsymbol{\Phi}) = \sum_{\mu \in \mathcal{P}^{\mathrm{s}}} \left\| \mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}} . \left\langle \phi_{\mathbf{k}}, \mathbf{u}_i \right\rangle \right\|^2 \qquad (2)$$

  with the constraint $\left\langle \phi_{\mathbf{i}}, \phi_{\mathbf{j}} \right\rangle = \delta_{ij}$
- What is the quantity of interest here? Which scalar product and norm should we choose?
- The canonic scalar product? $\Rightarrow \quad \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$
- Some kind of energy scalar product makes more sense: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{K}_{\{\tau | \tau < t\}}\, \mathbf{y}$
- Simplify to $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{K}_0\, \mathbf{y}$ since we want a fixed basis with time

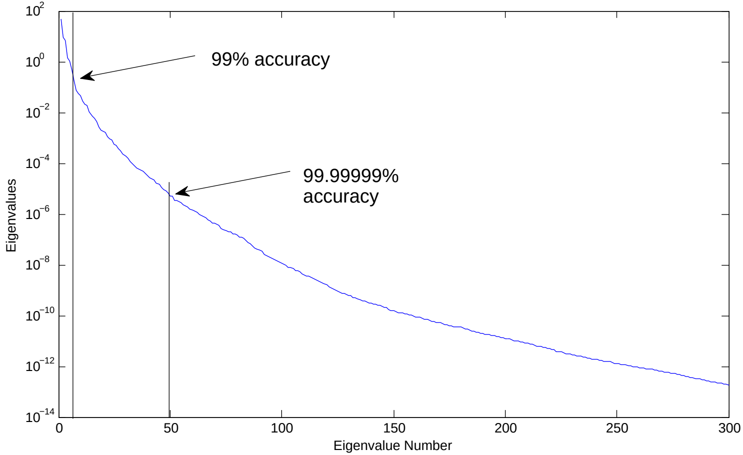
- Find the basis $[\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots] = \boldsymbol{\Phi}$ that minimises the cost function:

$$J^{\mathrm{s}}_{\langle\cdot\rangle}(\boldsymbol{\Phi}) = \sum_{\mu\in\mathcal{P}^{\mathrm{s}}} \|\mathbf{u}_i - \sum_{k}^{n_{\mathrm{POD}}} \phi_{\mathbf{k}}\cdot\langle\phi_{\mathbf{k}}, \mathbf{u}_i\rangle\|^2 \tag{2}$$

  with the constraint $\langle\phi_{\mathbf{i}}, \phi_{\mathbf{j}}\rangle = \delta_{ij}$

- What is the quantity of interest here? Which scalar product and norm should we choose?
- The canonic scalar product? $\Rightarrow \quad \langle\mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^T\mathbf{y}$
- Some kind of energy scalar product makes more sense: $\langle\mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^T\mathbf{K}_{\{\tau|\tau<t\}}\,\mathbf{y}$
- Simplify to $\langle\mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^T\mathbf{K}_0\,\mathbf{y}$ since we want a fixed basis with time
- One can prove analytically that the solution is given by the eigenvectors of $\mathbf{K}_0\,\mathbf{S}\,\mathbf{S}^T\mathbf{K}_0$

Next question: how many vectors should we pick?



Reduced equations

- Reduced system after linearisation: $\min_{\underline{\boldsymbol{\alpha}}} \quad \|\mathbf{K}\boldsymbol{\Phi}\,\alpha + \mathbf{F}_{\mathbf{ext}}\|$

Reduced equations

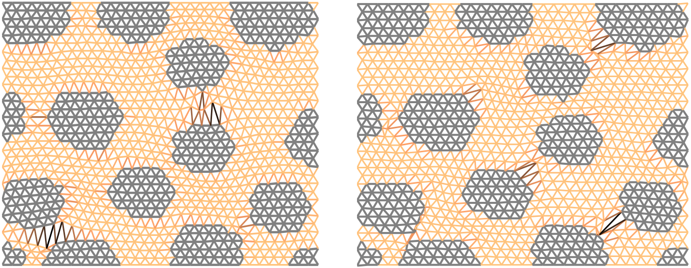
- Reduced system after linearisation: $\min_{\underline{\boldsymbol{\alpha}}} \quad \|\mathbf{K}\boldsymbol{\Phi}\,\alpha + \mathbf{F}_{\mathbf{ext}}\|$
- In the Galerkin framework: $\boldsymbol{\Phi}^T\,\mathbf{K}\boldsymbol{\Phi}\,\alpha + \boldsymbol{\Phi}^T\,\mathbf{F}_{\mathbf{ext}} = 0$

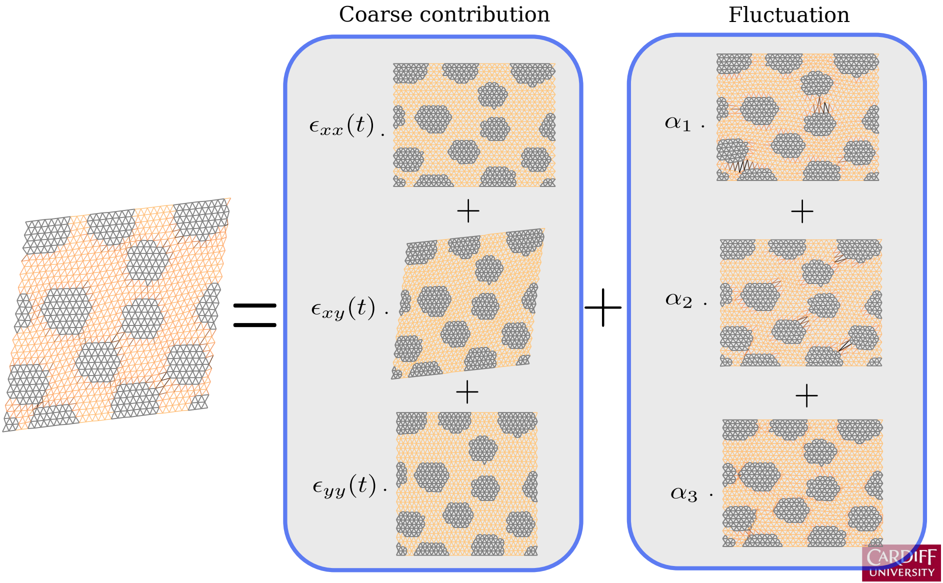
Reduced equations

- Reduced system after linearisation: $\min\limits_{\underline{\boldsymbol{\alpha}}} \quad \|\mathbf{K}\boldsymbol{\Phi}\,\alpha + \mathbf{F}_{\mathbf{ext}}\|$
- In the Galerkin framework: $\boldsymbol{\Phi}^T\,\mathbf{K}\boldsymbol{\Phi}\,\alpha + \boldsymbol{\Phi}^T\,\mathbf{F}_{\mathbf{ext}} = 0$
- That's it! In the online stage, this much smaller system will be solved.

Example

Snapshot selection:
simplify to monotonic loading in $\epsilon_{xx}, \epsilon_{xy}, \epsilon_{yy}$. 100 snapshots .
First 2 modes:



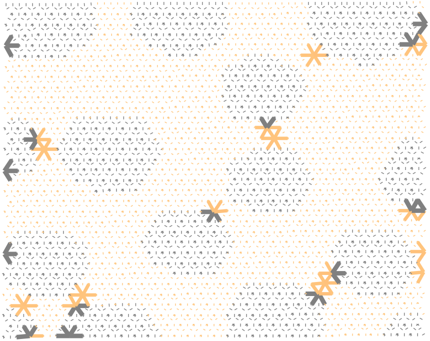

Coarse contribution
Fluctuation
=
$\epsilon_{xx}(t)$.
+
$\epsilon_{xy}(t)$.
+
$\epsilon_{yy}(t)$.
+
α_1 .
+
α_2 .
+
α_3 .

Is that good enough?

- Speed-up actually poor
- Equation “$\mathbf{\Phi}^T \mathbf{K}\mathbf{\Phi}\,\alpha + \mathbf{\Phi}^T \mathbf{F}_{\mathbf{ext}} = 0$“ quicker to solve but $\mathbf{\Phi}^T \mathbf{K}\mathbf{\Phi}$ still expensive to evaluate
- Need to do something more $\Longrightarrow$ system approximation

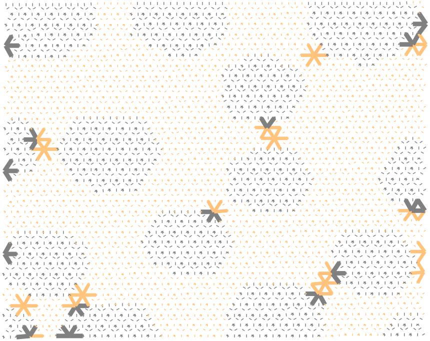
Idea

- Define a surrogate structure that retains only very few elements of the original one



Idea

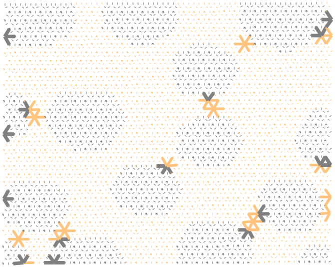
- Define a surrogate structure that retains only very few elements of the original one



- Reconstruct the operators using a second POD basis representing the internal forces

“Gappy“ technique

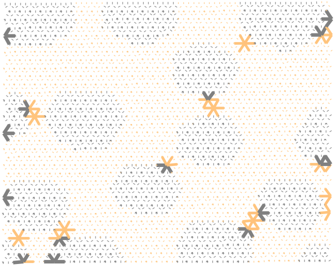
Originally used to reconstruct altered signals



- $\underline{\mathbf{F}}_{\mathbf{int}}\,(\mathbf{\Phi}\,\alpha)$ approximated by $\underline{\mathbf{F}}_{\mathbf{int}}\,(\mathbf{\Phi}\,\alpha) \approx \mathbf{\Psi}\,\beta$

“Gappy“ technique

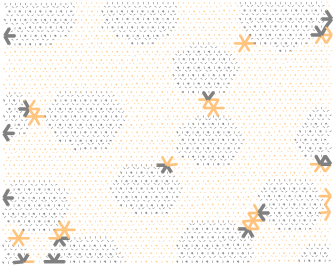
Originally used to reconstruct altered signals



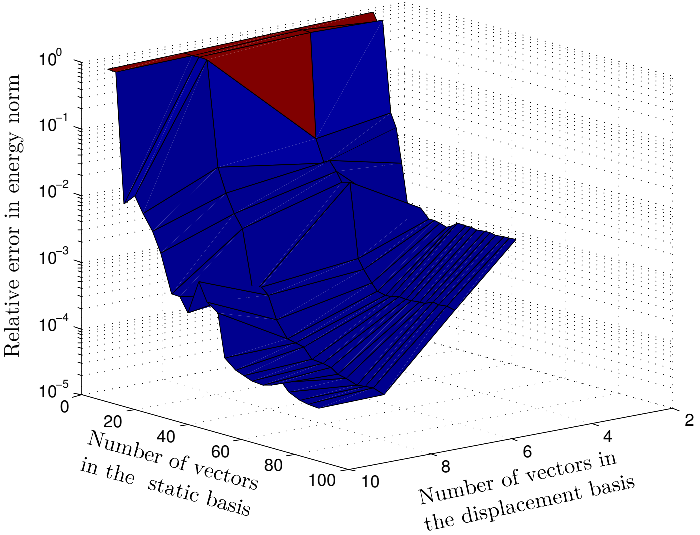
- $\underline{\mathbf{F}}_{\textbf{int}}\,(\boldsymbol{\Phi}\,\alpha)$ approximated by $\underline{\mathbf{F}}_{\textbf{int}}\,(\boldsymbol{\Phi}\,\alpha) \approx \boldsymbol{\Psi}\,\beta$
- $\underline{\mathbf{F}}_{\textbf{int}}\,(\boldsymbol{\Phi}\,\alpha)$ is evaluated exactly only on a few selected nodes: $\underline{\mathbf{F}}_{\textbf{int}}\,\widehat{(\boldsymbol{\Phi}\,\alpha})$

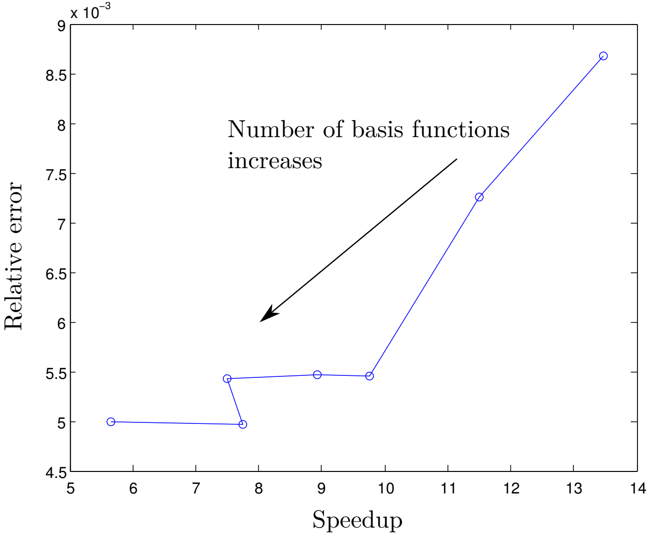
“Gappy“ technique

Originally used to reconstruct altered signals



- $\underline{\mathbf{F}}_{\mathbf{int}}\,(\boldsymbol{\Phi}\,\alpha)$ approximated by $\underline{\mathbf{F}}_{\mathbf{int}}\,(\boldsymbol{\Phi}\,\alpha) \approx \boldsymbol{\Psi}\,\beta$
- $\underline{\mathbf{F}}_{\mathbf{int}}\,(\boldsymbol{\Phi}\,\alpha)$ is evaluated exactly only on a few selected nodes: $\widehat{\underline{\mathbf{F}}_{\mathbf{int}}\,(\boldsymbol{\Phi}\,\alpha)}$
- β found through: $\min_{\boldsymbol{\beta}} \left\| \widehat{\boldsymbol{\Psi}}\,\beta - \widehat{\underline{\mathbf{F}}_{\mathbf{int}}\,(\boldsymbol{\Phi}\,\alpha)} \right\|_2$


Relative error in energy norm
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
20
40
60
80
100
Number of vectors in the static basis
2
4
6
8
10
Number of vectors in the displacement basis


x 10^−3
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
5
6
7
8
9
10
11
12
13
14
Relative error
Speedup
Number of basis functions increases

Conclusion

- Model order reduction can be used to solved the RVE problem faster and with a reasonable accuracy
- Can be thought of as a bridge between analytical and computational homogenisation:
  the reduced bases are pseudo-analytical solutions of the RVE problem that is still computationally solved at very reduced cost

Thank you for your attention!