POD-based reduction methods, the Quasi-continuum method and their resemblance

Lars Beex, Stéphane Bordas, Elisa Schenone, Jack S. Hale, Ron Peerlings, Marc Geers, Pierre Kerfriden
Large nonlinear models are inefficient due to:

1. Many DOFs

2. Many integration points
Large nonlinear models are inefficient due to:

1. Many DOFs

2. Many integration points
1. Applications: discrete materials & discrete models
1. Applications: discrete materials & discrete models

2. The Quasicontinuum method:
   interpolation & integration
Outline

1. Applications: discrete materials & discrete models

2. The Quasicontinuum method:
   interpolation & integration

3. POD-based reduction methods:
   interpolation & integration
Outline

1. Applications: discrete materials & discrete models

2. The Quasicontinuum method:
   interpolation & integration

3. POD-based reduction methods:
   interpolation & integration

4. Concluding remarks
Outline

1. Applications: discrete materials & discrete models

2. The Quasicontinuum method:
   interpolation & integration

3. POD-based reduction methods:
   interpolation & integration

4. Concluding remarks
Some discrete materials

- Foams
- Additive manufacturing
- Collagen
- Paper/cardboard
- Textiles
Electronic textile

R. Peerlings, M. Geers

2.5 mm
Elastoplastic spring lattice
Outline

1. Applications: discrete materials & discrete models

2. The Quasicontinuum method:
   interpolation & integration

3. POD-based reduction methods:
   interpolation & integration

4. Concluding remarks
Quasicontinuum method (Tadmor et al., 1996)

Direct simulation of elastic lattice

\[ E_{\text{tot}}(u) = E_{\text{int}}(u) - f_{\text{ex}}^T u \]
Quasicontinuum method (Tadmor et al., 1996)

Direct simulation of elastic lattice

\[ E_{tot}(u) = E_{int}(u) - f_{ex}^T u \]

\[ E_{tot}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]
**Quasicontinuum method (Tadmor et al., 1996)**

**Interpolation**

\[
E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^Tu
\]

\[
E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t
\]

\[u = Nu_t\]
Quasicontinuum method (Tadmor et al., 1996)

Linear interpolation

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ u = Nu_t \]
Quasicontinuum method (Tadmor et al., 1996)

Linear interpolation

\[ u = Nu_t \]
Quasicontinuum method (Tadmor et al., 1996)

Integration for linear triangles

\[ E_{tot}(u) = \sum_{i=1}^{n} E_i(u) - f_{ext}^T u \]

\[ E_{tot}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ext}^T Nu_t \]

\[ u = Nu_t \]
Quasicontinuum method (Tadmor et al., 1996)

Integration for linear triangles

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ u = Nu_t \]

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ \sum_{i=1}^{n} E_i(Nu_t) \approx \sum_{i=1}^{s} w_i E_i(Nu_t) \]
Virtual-power-based QC method

Integration for linear triangles
Virtual-power-based QC method

Integration 1 for linear triangles
Virtual-power-based QC method

Integration 2 for linear triangles
Virtual-power-based QC method

Accuracy and efficiency

Plastic strain at 10% horizontal stretch
Euler Bernoulli beams:  
- Hermite interpolation in each beam  
- nodal displacements  
- nodal rotations
QC method for planar beam lattice

Interpolation

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ u = Nu_t \]
QC method for planar beam lattice

Nodal displacements: Linear
Nodal rotations: Linear
Conforming triangulations
QC method for planar beam lattice

Test cases
QC method for planar beam lattice

Nodal displacements: Cubic
Nodal rotations: Quadratic
Conforming triangulations
QC method for planar beam lattice

Nodal displacements: Cubic
Nodal rotations: Quadratic
Non-conforming triangulations
QC method for planar beam lattice

Interpolation

\[
E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u
\]

\[
E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t
\]

\[u = Nu_t\]
QC method for planar beam lattice

Integration

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ u = N u_t \]

\[ E_{\text{tot}}(N u_t) = \sum_{i=1}^{n} E_i(N u_t) - f_{ex}^T N u_t \]

\[ \sum_{i=1}^{n} E_i(N u_t) \approx \sum_{i=1}^{s} w_i E_i(N u_t) \]
QC method for planar beam lattice

Sampling beams near Gauss points
1. Applications: discrete materials & discrete models

2. The Quasicontinuum method: interpolation & integration

3. POD-based reduction methods: interpolation & integration

4. Concluding remarks
POD-based ROM (Ladevèze, Farhat, Chinesta,...)

E. Schenone, J.S. Hale, S. Bordas

Largescale model

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ u = Nu_t \]

E(interpolation)

E(summation)

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ \sum_{i=1}^{n} E_i(Nu_t) \approx \sum_{i=1}^{s} w_i E_i(Nu_t) \]
POD-based ROM

Nonlinear reaction diffusion

\[ K \Delta u = f(u) \]
POD-based ROM (Ladevèze, Farhat, Chinesta, …)

Largescale model

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ u = Nu_t \]
POD-based ROM

How to find N: *Offline snapshot/training generation*

\[ K \Delta u = f(u) \]

many results: \( u_1, u_2, \ldots, u_{150} \)

\[ E_{\text{tot}}(N_{\text{ut}}) = E_i(N_{\text{ut}}) \approx E_i(N_{\text{ut}}) \]

How to find N:
*Offline snapshot/training generation*

\( \sum_{i=1}^{n} \approx \sum_{i=1}^{n} \)

\( u_1, u_2, \ldots, u_{150} \)
POD-based ROM

How to find $N$: Offline snapshot/training generation

$$K \Delta u = f(u) \quad \text{many results: } u_1, u_2, \ldots, u_{150} \quad N = [u_1, u_2, \ldots, u_{150}]$$

reduced basis

- \( K \Delta u = f(u) \)
- \( N = [u_1, u_2, \ldots, u_{150}] \)
- \( \sum \)
POD-based ROM

Or apply first singular value decomposition to

\[ N = [u_1, u_2, \ldots, u_{150}] \]

and use the modes with the largest eigenvalues

\[ N = [\varphi_1, \varphi_2, \ldots, \varphi_{30}] \]

**Note:** \( N \) is full
Largescale model → Offline training to find modes → E(summation)

\[ E_{\text{tot}}(u) = \sum_{i=1}^{n} E_i(u) - f_{ex}^T u \]

\[ E_{\text{tot}}(Nu_t) = \sum_{i=1}^{n} E_i(Nu_t) - f_{ex}^T Nu_t \]

\[ \sum_{i=1}^{n} E_i(Nu_t) = \sum_{i=1}^{s} w_i E_i(Nu_t) \]
POD-based ROM

\[\varphi_1\]

\[\varphi_2\]

\[\varphi_3\]

\[\varphi_4\]

\[\varphi_5\]

\[\varphi_6\]
How to select reduced integration points for oscillatory basis functions?
How to select reduced integration points for oscillatory basis functions?

Locally approximate the modes linearly!

\[ \varphi_i(N_{\text{ut}}) = \sum_{j=1}^{n} \varphi_j(N_{\text{ut}}) \approx \varphi_i(N_{\text{ut}}) \]
POD-based ROM

Let’s look at $\varphi_1$
POD-based ROM

Let's look at $\varphi_1$

Start with coarse mesh
Let's look at $\varphi_1$
Let’s look at $\varphi_1$
Let's look at $\varphi_1$
POD-based ROM

Let’s look at $\varphi_1$
Final mesh of $\varphi_1$
POD-based ROM

Final mesh of $\varphi_1$ is the first mesh for $\varphi_2$
POD-based ROM

$\varphi_1$  $\varphi_2$  $\varphi_3$

$\varphi_4$  $\varphi_5$  $\varphi_6$
POD-based ROM

![Graph showing the speed-up of POD-based ROM for different tolerances (tol = 10%, tol = 5%, tol = 1%) and comparing it to the standard POD (Std POD). The x-axis represents the number of modes, and the y-axis represents the speed-up. The graph illustrates the trade-off between the number of modes and the speed-up for different tolerances.](image)
POD-based ROM

The graphs show the speed-up and relative error for different tolerances (tol = 10%, tol = 5%, tol = 1%) and the standard POD method. The speed-up increases as the number of modes decreases, indicating faster computation. The relative error decreases rapidly at the beginning, then plateaus as the number of modes increases. The blue line represents tol = 10%, the red line tol = 5%, and the green line tol = 1%, with the black triangles representing the standard POD method.
POD-based ROM

Indentation of a hyperelastic cube

30x faster than standard POD

FEM

Reduced integrated POD
## Concluding remarks

<table>
<thead>
<tr>
<th>Offline training to find $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>A LOT, so only useful for optimisation</td>
</tr>
</tbody>
</table>
Concluding remarks

<table>
<thead>
<tr>
<th>Offline training to find $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline training to find $N$</td>
<td>None</td>
<td>A LOT, so only useful for optimisation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finding integration points</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding integration points</td>
<td>Every time the same</td>
<td>Changes every time</td>
</tr>
</tbody>
</table>
### Concluding remarks

<table>
<thead>
<tr>
<th>Offline training to find $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>A LOT, so only useful for optimisation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finding integration points</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every time the same</td>
<td>Changes every time</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localized behavior</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully resolved regions</td>
<td>Limited... not easy</td>
<td></td>
</tr>
</tbody>
</table>
## Concluding remarks

<table>
<thead>
<tr>
<th>Offline training to find $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>A LOT, so only useful for optimisation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finding integration points</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every time the same</td>
<td>Changes every time</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localized behavior</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully resolved regions</td>
<td>Limited... not easy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptivity of $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple, borrow from FE</td>
<td>Difficult I guess..</td>
<td></td>
</tr>
</tbody>
</table>
Concluding remarks

<table>
<thead>
<tr>
<th>Offline training to find $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline training to find $N$</td>
<td>None</td>
<td>A LOT, so only useful for optimisation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finding integration points</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding integration points</td>
<td>Every time the same</td>
<td>Changes every time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localized behavior</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localized behavior</td>
<td>Fully resolved regions</td>
<td>Limited… not easy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptivity of $N$</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptivity of $N$</td>
<td>Simple, borrow from FE</td>
<td>Difficult I guess..</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of discretisation</th>
<th>QC method</th>
<th>POD-based ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of discretisation</td>
<td>ONLY REGULAR</td>
<td>ANY</td>
</tr>
</tbody>
</table>
Ongoing & future work

1. QC for irregular networks
2. (goal-oriented) Adaptivity for QC
3. Apply QC to true materials, no academic examples
4. Geometrical scaling of POD-modes
5. Coupling of POD domains