A simple method to cut assembly costs in reduced order models of non-linear PDEs

Jack S. Hale[a], Elisa Schenone[a], Davide Baroli[a], Lars A. A. Beex[a], Stéphane P. A. Bordas[a,*]

[a] *Institute of Computational Engineering, University of Luxembourg, Maison du Nombre, 6 Avenue de la Fonte, Belval, L-4362, Luxembourg.*

Abstract

In several reduced-order-modelling (ROM) approaches, the size of the systems, stemming from partial differential equations (PDEs), is reduced by the interpolation of the original degrees of freedom (DOFs). This reduces the solution time. For systems originating from *non-linear* PDEs however, a significant part of the computational efforts is not only related to the solution of these systems, but also to their construction because the systems have to be constructed from the ground up for each iteration. In those cases, the reduction of the number of DOFs can be combined with a reduction of the number of integration points. This is often referred to as hyperreduction. Relatively few strategies to decrease the number of integration points can be found for global, non-linear interpolation/basis functions - as used in many ROM approaches. In this contribution, a new and simple strategy is proposed to establish reduced integration points. The ansatz of our approach is based on the discretisation of the global, non-linear basis functions into sets of linear functions in subdomains, for which the locations and weights of the integration points can be straightforwardly determined. The discretisation of the basis functions is achieved using existing finite element mesh refinement algorithms and it was therefore relatively straightforward to implement. The discretisation can furthermore be as fine as desired. This entails that the number of reduced integration points is not constrained by the number of basis functions computed. Two-dimensional and three-dimensional numerical examples are presented based on proper-orthogonal-decomposition, amongst others in combination with the indentation of a hyperelastic solid.

Keywords: *hyperreduction, reduced integration, reduced order modelling, model order reduction, non-linear PDEs, hyperelasticity*

*Corresponding author

1. Introduction

The response of many physical systems across the sciences and engineering can be predicted by finding the solution of a non-linear and possibly time-dependant partial differential equation (PDE). In most non-trivial cases this PDE cannot be tackled using analytical methods and we must resort to numerical methods such as the finite element method to find a solution.

We begin with a brief recap of applying the finite element method to the find the solution of a non-linear PDE. The spatial domain of many physically relevant systems is very large and the discretisation must be sufficiently fine to capture the local behaviour of the solution. In a non-linear setting where a Newton algorithm is commonly used, this approach leads to a sequence of very large and sparse linear algebra problems that must be solved.

The construction of this sequence of linear algebra problems is often very computationally demanding. At each Newton iteration we can split the computational cost between two phases

i. The assembly (integration and insertion of element cell tensors into sparse matrices and vectors) of a large and sparse linear system arising from the finite element discretisation.
ii. The solution of that linear system to find the Newton update.

The solution is updated using the Newton update, and the assembly and solution procedure repeats itself until some convergence criteria is reached. We emphasise that both of these steps must be performed at every Newton iteration as the solution evolves through the Newton procedure.

Without going into unnecessary detail, we wish to point out that the algorithmic scaling with problem size of the assembly operation is usually[1] superior to that of the solution of the linear system. Thus for any sufficiently large problem, the cost of the solution of the linear system dominates the assembly. The precise problem size at which the linear system solution dominates assembly of course depends on a whole variety of mathematical and implementation-dependent factors, but this scaling argument is a good rule-of-thumb for many problems.

With that in mind, the obvious way to reduce computational cost is to reduce the size of the linear systems that must be solved. In the context of the finite element method this can be achieved by, for example, using a posteriori error estimation techniques to drive mesh adaptivity, see e.g. []. The adaptive mesh then suits the spatially local behaviour of the problem at hand. The adapted mesh has fewer cells, and as cells give rise to degrees of freedom in the linear system, both the assembly and solution of the linear system will be cheaper. We mention mesh adaptivity not at random from the huge selection of procedures that can be used to reduce the cost of the finite element method, but because it forms a core part of the techniques we develop in this paper.

[1] Optimal multigrid methods, which only exist for some problem types, being the notable exception.

Another approach to reduce computational cost is reduced-order modelling. In many notable applications, e.g. real-time simulation [], accelerating parametric studies and stochastic simulation [], it is necessary to rapidly find a solution to the same PDE at different points in parameter space.

Many, although by no means all, reduced-order modelling approaches consist of an offline and an online phase. A great deal of computational work is performed upfront in the offline phase with a view to producing an online phase where multipe solutions in the parameter space can be calculated very quickly.

Typically this offline phase includes pre-computation of a set of solution snapshots at different points in the parameter space. This pre-computation is performed using a standard numerical method such as the finite element method. Once these snapshots have been have been computed, the information within them is in some way compressed into a more efficient global basis representation. In the reduced basis (RB) approach, e.g. [19, 22, 10, 26], a linear combination of the snapshots themselves are used as the global basis. In the proper orthogonal decomposition (POD) approach e.g. [21, 14, 18, 27, 28, 6, 23] an orthonormal reduced basis is constructed that is optimal in the sense that the sum of the squared errors between the snapshots and their approximation in the POD basis is minimised. In this paper we exclusively work with basis functions computed with the POD approach, but certainly other approaches such as RB could be used.

In contrast with the finite element basis, where complex behaviour of a single problem is captured via simple cell-local polynomial basis functions on a specially adapted mesh, the reduced-order modelling approach optimally represents the complex behaviour contained within the range of snapshots using a specially selected set of global basis functions. Coming back to our brief discussion a posteriori mesh refinement, the output of which is a finite element mesh specially suited to the problem at hand, a reduced-order modelling strategy instead outputs a set of global basis functions specially suited to the snapshots at hand.

We now turn to the online phase. In this phase we would like to quickly compute many thousands of new solutions at points in parameter space that were not previously computed offline in the set of snapshots. We do this by using the POD modes to interpolate the the solution space. Many works in the literature follow this approach, e.g. [24, 17, 25, 9, 20, 11]. In a non-linear setting we still need to use a Newton algorithm, so there is still a sequence of linear algebra problems to solve as in the finite element method. The construction of this sequence of linear algebra problems in the reduced-order modelling setting consists of the following steps:

i. The assembly of a large and sparse linear system arising from the finite element discretisation. This linear system is then projected through the reduced-order basis functions, resulting in a small and dense linear system.
ii. The solution of the new small and dense linear system to find the Newton update.

Solving these small dense linear systems is usually quicker than solving their

large sparse finite element counterparts, although we feel it is important to point out that algorithmic complexity of large sparse linear system solvers is typically better than small dense linear systems solvers. In practice however, the dense linear systems are usually so much smaller (tens to hundreds of degrees of freedom) than the sparse systems (tens of thousands to billions) that they take far less time to solve. The Newton update is performed as in the finite element method, and the procedure repeats until the convergence is reached.

Here is the critical problem with the online phase in this non-linear setting; the solution of the small linear system is now so much cheaper that the assembly step massively dominates the overall cost of the online phase. This wasn't an issue in the finite element method; as already discussed for any sufficiently large problem linear system solution time usually dominates assembly. In short, because of the effectiveness of the reduced-order basis approach, the main computational burden from solution back to assembly.

2. Problem formulation

Our starting point is the following weak residual form of a system of nonlinear and time-dependent partial differential equations:

$$F(u, t, m; \tilde{u}) = 0 \quad \forall\, (\tilde{u},\, t) \in V \times \mathbb{R}^+ \tag{1}$$

where $u \in V$ is the solution in some suitable function space V, $t \in \mathbb{R}^+$ is time, $m \in M$ are some parameters of the PDE and $F : V \times \mathbb{R}^+ \times M \times V \to \mathbb{R}$ is a semi-linear form, that is, linear in the arguments (here, the test function $\tilde{u} \in V$) following the semicolon.

We can then linearise the

2.1. Construction of reduced integration meshes

In this section we present an approach to obtain a mesh that locally approximates all N_m basis functions simultaneously in a linear fashion. The reason is that we can straightforwardly determine the number, locations and weights of the integration points in a linear cell, but not for the oscillating global basis functions. To ensure that a relatively small number of integration points is obtained, the number of cells of the new mesh must be significantly smaller than that of the original mesh ($N_c^{N_m} \ll N_c^0$).

The aim here is to start with a coarse mesh and adaptively refine it such that each POD mode ($\boldsymbol{\psi}_i$) is accurately approximated up to a user-defined tolerance. When this tolerance is sufficiently small, the solution of the reduced integration POD approach should be similar to that of the standard POD approach.

In effect, we drive an offline mesh adaptivity process for the global POD basis functions that results in a mesh specially suited to the reduced order problem. Clearly, this approach resembles the adaptation of a mesh based on the solution, except that we perform the adaptation *a priori* and on the information contained in the POD basis.

Remark 1. *Our algorithm relies on a novel combination of tools readily found in most modern finite element toolboxes, namely; adaptive mesh refinement, projection and interpolation between functions on non-matching meshes, assembly of bilinear and linear forms and a few basic linear algebra operations. We have chosen to implement our method into DOLFIN, part of the FEniCS Project, but any modern finite element toolbox would be sufficient. It was not necessary to perform any work in the core parts of DOLFIN to implement our method.*

Remark 2. *For a given tolerance our algorithm only needs to be applied once in the offline stage. Then, in the online phase the user can select an appropriate number of modes and the associated precomputed mesh is used.*

We start with the original mesh $\mathcal{T}^0$ consisting of N_c^0 simplicial cells $\{T_1^0, \ldots, T_{N_c^0}^0\}$ that form an approximation to our polyconvex problem domain $\cup_{i=1}^{N_c} T_i^0 = \Omega_h$. Using this mesh and the POD-based ROM approach of the previous subsection, the N_m POD basis functions are generated. We then create the coarsest possible mesh $\mathcal{T}^1$ with N_c^1 simplicial cells $\{T_1^1, \ldots, T_{N_c^1}^1\}$ that also covers the approximate polyconvex problem domain, i.e. $\cup_{i=1}^{N_c^1} T_i^1 = \cup_{i=j}^{N_c^0} T_j^0 = \Omega_h$.

Example 1. *In the case of a square domain Ω_h, $\mathcal{T}_h^1$ is a two cell diagonal mesh, in which case the exact domain is covered exactly by the approximate one ($\Omega_h = \Omega$).*

Starting with the first POD basis function $\boldsymbol{\psi}_1$, we begin the adaptation of mesh $\mathcal{T}^1$. Recall that this basis function (like all of them) is interpolated on the original finite element function space $\mathcal{V}_h$ and since it is the mode associated with the leading eigenvalue, it is the least oscillatory function in the entire basis.

For each cell T_i^1 in the coarse mesh $\mathcal{T}^1$ we calculate the following local error estimator:

$$\eta_i = \int_{T_i^1(Q_n)} |\boldsymbol{\psi}_1| \, \mathrm{d}x - \int_{T_i^1(Q_{n+1})} |\boldsymbol{\psi}_1| \, \mathrm{d}x, \tag{2}$$

where the notation $T(Q_n)$ denotes that the corresponding integral is calculated using a quadrature rule on the simplex of order n. Note that in the numerical integration procedure the value of the function $\boldsymbol{\psi}_1$ at a point $x \in T$ is evaluated using the basis functions in the finite element function space $\mathcal{V}_h$ defined on the original mesh $\mathcal{T}^0$.

At least from a heuristic perspective, assuming that the basis function $\boldsymbol{\psi}_1$ is sufficiently smooth, cell error indicator η_i provides an indication of further refinement of cell T_i which would lead to an improved approximation to the integral of the mode.

We also calculate two global quantities. The first, I_e is the integral of the absolute value of basis function $\boldsymbol{\psi}_1$ on the original mesh $\mathcal{T}^0$, using a sufficiently high quadrature rule of order m (with respect to the order of the basis functions in $\mathcal{V}_h$):

$$I_e = \sum_{i=1}^{N_c^0} \int_{T_i^0(Q_m)} |\boldsymbol{\psi}_1| \, \mathrm{d}x. \tag{3}$$

This can be considered the 'exact' integral, at least to the extent that it is the best approximation in the space $\mathcal{V}_h$.

The second quantity is the approximate integral I_a of the absolute value of the mode calculated on the coarse mesh $\mathcal{T}^1$:

$$I_a = \sum_{i=1}^{N_c^1} \int_{T_i^1(Q_n)} |\boldsymbol{\psi}_1| \, \mathrm{d}x. \tag{4}$$

We then calculate the following relative error:

$$E = |I_e - I_a|/I_e \tag{5}$$

If E is greater than some specified tolerance, e.g. 5%, we use the cell-based error indicators η^i to mark some cells in the mesh $\mathcal{T}_h^0$ based on the well-known Dörfler algorithm, see []. The percentage of the cells that is marked for refinement is defined by the user. The refinement of the mesh is performed using the PLAZA algorithm built-in to DOLFIN. The algorithm is repeated for mode $\boldsymbol{\psi}_1$ as many times as necessary to reduce error E below the specified tolerance.

Once the condition on E is met, we have our specially adapted mesh $\mathcal{T}^1$ for mode $\boldsymbol{\psi}_1$. The above process using the mesh $\mathcal{T}^1$ is then used as input for the adaptive refinement algorithm for mesh $\mathcal{T}^2$ on mode $\boldsymbol{\psi}_2$. Mesh $\mathcal{T}^2$ is thus accurate for modes $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ (and similarly, mesh $\mathcal{T}^i$ for modes $\boldsymbol{\psi}_1$ through $\boldsymbol{\psi}_i$). This process is continued until the mesh adaptation is performed for all N_m modes. This results in a collection of N_m meshes $\left\{\mathcal{T}^1, \ldots, \mathcal{T}^{N_\mathrm{m}}\right\}$. We emphasise that not all of the meshes are unique as typically meshes suited to the integration of mode $\boldsymbol{\psi}_i$ are also suitable for a number of subsequent modes.

A more formal version of this algorithm is detailed in Algorithm 1 and is implemented in DOLFIN using only 60 lines of code.

Algorithm 1 Adaptive construction of reduced integration meshes.

Retrieve original mesh $\mathcal{T}^0$ with N_c^0 simplicial cells
Retrieve $\mathbf{\Psi}$ (the N_m POD modes defined on $\mathcal{V}_h$)
Set tolerance on the integration error, tol, and refinement tolerance, ref
Create initial mesh $\mathcal{T}^1$ with N_c^1 simplicial cells
for $i = 1 : N_{\mathrm{m}}$ **do**
  Set N_c^i as the number of cells of current mesh $\mathcal{T}^i$
  Calculate exact integral I_e (Eq. (3))
  Calculate approximate integral I_a (Eq. (4))
  Calculate $E = |I_e - I_a|/I_e$
  while $E > tol$ **do**
    for $l = 1 : N_c^i$ **do**
      Calculate local error estimator η_l (Eq. (2))
    end for
    Based on η_l refine mesh $\mathcal{T}^i$ with Dörfler and PLAZA algorithms
    Calculate approximate integral I_a (Eq. (4))
    Calculate $E = |I_e - I_a|/I_e$
  end while
  Store $\mathcal{T}^i$
  Set $\mathcal{T}^{i+1}$ as $\mathcal{T}^i$
end for
return $\mathcal{T}^1, \mathcal{T}^2, \ldots, \mathcal{T}^{N_{\mathrm{m}}}$.

Remark 3. *We would like to point out that the employed algorithm is certainly not the only way that the reduced integration meshes can be constructed. With more advanced error estimators and anisotropic refinement strategies one could likely reduce the number of cells further. For the considered examples however, the proposed strategy gives a satisfactory balance between accuracy and efficiency.*

2.2. Construction of reduced integration meshes

In this subsection we present an adaptive algorithm to construct the reduced integration meshes necessary to reduce the expense of computing the non-linear terms in the reduced order space for any given problem. that accurately approximate the energy of each mode up to a user-defined tolerance. If we can do this well, the solution with our reduced integration POD approach should not be dissimilar to that obtained using the standard POD approach, but with reduced computational expense.

Summarising the approach outlined below, we adaptively generate a sequence of refined meshes that are specifically tailored to integrate the POD basis functions, which in turn are specifically tailored to the selected snapshots generated in the offline phase. In effect, we drive an offline mesh adaptivity process for the global POD basis functions that results in a mesh specially suited to the reduced order problem. Clearly, this idea closely related to the common idea of driving an adaptation of the mesh based on the solution, except that

in our case we perform all of our adaptation *a priori* based on the information contained in the POD basis.

Remark 4. *Our algorithm relies on a novel combination of tools readily found in most modern finite element toolboxes, namely; adaptive mesh refinement, projection and interpolation between functions on non-matching meshes, assembly of bilinear and linear forms and a few basic linear algebra operations. We have chosen to implement our method into DOLFIN, part of the FEniCS Project, but any modern finite element toolbox would be sufficient. It was not necessary to do* any *work in the core parts of DOLFIN to implement our method.*

Remark 5. *For a given tolerance our algorithm only needs to be applied once in the offline stage. Then, in the online phase the user can select an appropriate number of modes and the associated precomputed mesh is used.*

We begin with our original mesh $\mathcal{T}_h^0$ consisting of N_c^0 simplicial cells $\{T_1^0, \ldots, T_{N_c^0}^1\}$ that form an approximation to our polyconvex problem domain $\cup_{i=1}^{N_c} T_i^0 = \Omega_h$. Using this mesh and the standard methodology outlined in §**??** we generated our N_s solution snapshots $\{u_1, \ldots, u_{N_s}\}$ from which we can calculate our N_m POD basis functions $\{\psi_1, \ldots, \psi_{N_\mathrm{m}}\}$.

We then create the coarsest possible mesh $\mathcal{T}_h^1$ with N_c^1 simplicial cells $\{T_1^1, \ldots, T_{N_c^1}^1\}$ that also covers the approximate polyconvex problem domain, i.e. $\cup_{i=1}^{N_c^1} T_i^1 = \cup_{i=j}^{N_c^0} T_j^0 = \Omega_h$.

Example 2. *In the case of a square domain Ω_h, $\mathcal{T}_h^1$ could be the two cell diagonal mesh shown in* **??** *and in that case the exact problem domain is covered exactly by the approximate one $\Omega_h = \Omega$.*

Then starting with POD basis function ψ_1, we begin the adaptation of mesh $\mathcal{T}_h^1$. Recall that this POD basis function has global support and is interpolated on the finite element function space $\mathcal{V}_h$ that was used to generate the snapshots of the non-linear problem. Furthermore, it is the mode associated with the leading eigenvalue of the POD spectrum and thus it is the least oscillatory function in the entire basis.

For each cell T_i^1 in the coarse mesh $\mathcal{T}_h^1$ we calculate the following simple local error estimator η_h^i:

$$\eta_h^i = \int_{T_i^1(Q_n)} |\psi_1(x)| \, \mathrm{d}x - \int_{T_i^1(Q_{n+1})} |\psi_1(x)| \, \mathrm{d}x, \tag{6}$$

where the notation $T_h(Q_n)$ denotes that the corresponding integral is calculated using a quadrature rule on the simplex of order n. Note that in the numerical integration procedure the value of the function ψ_1 at a point $x \in T$ is evaluated using the basis functions in the finite element function space $\mathcal{V}_h$ defined on the original mesh $\mathcal{T}_h^0$.

At least from a heuristic perspective, assuming that the basis function ψ_1 is sufficiently smooth, then the cell error indicator η_h^i gives us an indication of

whether further refinement of the cell T_i would lead to an improved approximation to the integral of the mode.

We calculate two further global quantities. The first, I_e is the integral of the absolute value of the mode ψ_1 on the original mesh $\mathcal{T}_h^0$, using a sufficiently high quadrature rule of order m (with respect to the order of the basis functions in $\mathcal{V}_h$):

$$I_e = \sum_{i=1}^{N_c^0} \int_{T_i^0(Q_m)} |\psi_1| \, \mathrm{d}x. \tag{7}$$

This can be considered the 'exact' integral, at least to the extend that it is the from the best approximation in the space $\mathcal{V}_h$.

The second quantity is the approximate integral I_a of the absolute value of the mode calculated on the coarse mesh $\mathcal{T}_h^1$:

$$I_a = \sum_{i=1}^{N_c^1} \int_{T_i^1(Q_n)} |\psi_1| \, \mathrm{d}x. \tag{8}$$

We then calculate the following relative error:

$$E = |I_e - I_a|/I_e \tag{9}$$

If E is greater than some specified tolerance, e.g. 5%, we use the cell-based error indicators η_h^i to mark some cells in the mesh $\mathcal{T}_h^0$ based on the well-known Dörfler algorithm, see []. The refinement of the mesh is performed using the PLAZA algorithm built-in to DOLFIN. We then repeat the above algorithm for mode ψ_1 as many times as necessary to drop the error E below the specified tolerance.

Once the condition on E is met, we have our specially adapted mesh $\mathcal{T}_h^1$ for mode ψ_1. We then repeat the above process using the mesh $\mathcal{T}_h^1$ as input for the adaptive refinement algorithm for mesh $\mathcal{T}_h^2$ on mode ψ_2, and so on until we have performed the mesh adaptation for all N_m modes. The result is a collection of N_m meshes $\left\{\mathcal{T}_h^1, \ldots, \mathcal{T}_h^{N_\mathrm{m}}\right\}$. We emphasise that not all of the meshes are unique as typically meshes suited to the integration of mode ψ_i are suitable for a number of subsequent modes as well.

A more formal version of this algorithm is detailed in Algorithm 2 and can be implemented in DOLFIN quite easily with around 60 lines of code.

Algorithm 2 Construct adapted reduced integration meshes.

The original mesh $\mathcal{T}_h^0$ with N_c^0 simplicial cells.
A set of POD modes, $\{\psi_1, \psi_{N_m}\}$ defined on a function space $\mathcal{V}_h$ on the original mesh.
An initial mesh $\mathcal{T}_h^1$ with N_c^1 simplicial cells with $N_c^1 \ll N_c^0$.
A tolerance on the integration error, tol.
Refinement percentage, ref.
for $i = 1 : N_{\mathrm{m}}$ **do**
  Let N_c^i be the number of cells in current mesh $\mathcal{T}_h^i$.
  Calculate exact integral $I_e = \sum_{j=1}^{N_c^0} \int_{T_j^0(Q_m)} |\psi_i| \, \mathrm{d}x$.
  Calculate approximate integral $I_a = \sum_{k=1}^{N_c^i} \int_{T_k^1(Q_n)} |\psi_i| \, \mathrm{d}x$.
  Calculate $error = |I_e - I_a|/I_e$.
  while $error > tol$ **do**
    for $l = 1 : N_c^i$ **do**
      Calculate local error estimator
      $$\eta_l = \int_{T_l^i(Q_n)} |\psi_i(x)| \, \mathrm{d}x - \int_{T_l^i(Q_{n+1})} |\psi_i(x)| \, \mathrm{d}x.$$
    end for
    Based on η_l refine mesh $\mathcal{T}_h^i$ with Dörfler and PLAZA algorithms.
    Calculate approximate integral $I_a = \sum_{k=1}^{N_c^i} \int_{T_k^1(Q_n)} |\psi_i| \, \mathrm{d}x$.
    Calculate $error = |I_e - I_a|/I_e$.
  end while
  Store $\mathcal{T}_h^i$.
  Let $\mathcal{T}_h^{i+1}$ be $\mathcal{T}_h^i$.
end for
return $\mathcal{T}_h^1, \ldots, \mathcal{T}_h^{N_{\mathrm{m}}}$.

Remark 6. *Our algorithm is certainly not the only way that the reduced integration meshes could be constructed. We believe that with more advanced error estimators and anisotropic refinement strategies could reduce the number of cells yet further. However, at least for the examples we have considered, the proposed strategy has been sufficient.*

3. Numerical examples

3.1. Reduction of a two-dimensional non-linear time-dependent problem

In this subsection, using the reduction method described above, we solve the time-dependent non-linear Fisher-KPP problem on a square domain Ω with pure Dirichlet boundary conditions on $\partial\Omega$. The full specification of the problem in strong form is:

$$\begin{aligned} &\text{Find } u(x,t) \text{ such that} \\ &\qquad \begin{aligned} \partial_t u - \Delta u &= cu(1-u) && \text{in } \Omega = [0,3]^2 \times (0,T], \\ u &= 0 && \text{on } \partial\Omega \times (0,T], \\ u(x,0) &= u_0(x) && \forall x \in \Omega, \end{aligned} \end{aligned} \tag{10}$$

where Δ is the Laplacian operator, ∂_t is the partial derivative operator with respect to time t, $c \in \mathbb{R}$ is a constant scalar parameter in space and time and $u_0(x)$ is a sufficiently smooth initial condition given by:

$$u_0(x) = \exp\left(-\frac{(x-x_0)^2}{\sigma^2}\right) \tag{11}$$

with $\sigma = 0.2$, $c = 50$, and $x_0 \in \mathbb{R}^2$. We let the position $x_0 \in \mathbb{R}^2$ and time $t \in \mathbb{R}$ form our parameters, and choose the discrete training set for the snapshot generation process as:

$$M_N = \left\{\tfrac{1}{2}, \tfrac{6}{10}, \ldots, 1\right\}^2 \times \left\{0, \tfrac{T}{10}, \tfrac{2T}{10}, \ldots, \tfrac{9T}{10}, T\right\}, \quad |M_N| = 396. \tag{12}$$

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 (10) can be written as:

$$\begin{aligned} &\text{Find } u(x,t) \in V \text{ such that:} \\ &F(u,c;v) := (\partial_t u; v) + (\nabla u; \nabla v) - (cu; v) + (cu^2; v) = 0 \quad \forall v \in \hat{V}. \end{aligned} \tag{13}$$

3.1.1. Finite difference discretisation in time

We first discretise (13) in time using a semi-implicit first-order finite difference scheme. We replace the solution $u(x,t)$ with an approximation $u^n \approx u(t^n)$ at $M+1$ times $t^0 < t^1 < \cdots < t^M$, and for simplicity we take the timestep $k \in \mathbb{R}$ as a constant giving $t^n = nk$ and final time $T = Mk$. We approximate the time derivative $\partial_t u$ with the following semi-implicit scheme:

$$\begin{aligned} (\partial_t u; v) &\approx \left(k^{-1}\left(u^{n+1} - u^n\right); v\right) \\ &= A(u^{n+1}; v) + B(u^n; v) + \tfrac{1}{2}C(u^n; v) + \tfrac{1}{2}C(u^{n+1}; v) \end{aligned} \tag{14a}$$

with the (semi-)linear forms:

$$A(u; v) = -(\nabla u; \nabla v), \tag{14b}$$

$$B(u; v) = -(c(u)^2; v), \tag{14c}$$

$$C(u; v) = (cu; v). \tag{14d}$$

with a slight abuse of notation on the exponent in (14c). In the numerical results presented in we take the final time $T = 0.1$ and $k^{-1} = 1000$. The snapshots in time in (12) are then collected from the discrete

Rearranging (14) gives the following sequence of problems to solve; for $n = 0, \ldots, M$ and $u^0 = u_0$, find $u^n \in V$ such that:

$$\left(\left\{k^{-1} - \tfrac{c}{2}\right\} u^{n+1}; v\right) + (\nabla u^{n+1}; \nabla v) = \left(\left\{k^{-1} + \tfrac{c}{2} - cu^n\right\} u^n; v\right) \ \forall v \in \hat{V} \tag{15}$$

Note that via this choice of timestepping scheme the only non-linear term is treated explicitly on the right-hand side of (15).

3.1.2. Finite element discretisation in space

We then discretise the problem in space using a Galerkin H^1-conforming finite element method by introducing discrete trial and test spaces $V_h \subset V := H_0^1(\Omega)$ and $\hat{V} \equiv V$ spanned with piecewise linear polynomial finite element basis basis functions $\{\phi_j\}_{j=1}^N$ with $N = \dim(V_h)$ giving:

$$u_h(x) = \sum_{i=1}^{N} \phi_i u_i. \tag{16}$$

Following standard arguments, e.g. [], the discrete governing equations of problem (10) can then be written using the finite element basis functions as the following sequence of linear systems; for $n = 0, \ldots, M$ find $\vec{u}^{\,n+1} \in \mathbb{R}^N$:

$$\left(\left\{k^{-1} - \tfrac{c}{2}\right\} M + K\right) \vec{u}^{\,n+1} = \left(k^{-1} + \tfrac{c}{2}\right) M \vec{u}^{\,n} - L(\vec{u}^{\,n}) \tag{17}$$

where M is the usual finite element mass matrix, K is the stiffness matrix associated with the Laplacian operator Δ and the vector $L(u)$ represents the non-linear term, with entries:

$$M_{ij} = (\phi_j; \phi_i), \quad K_{ij} = (\nabla\phi_j; \nabla\phi_i), \quad L(u_h)_i = (c(u_h)^2; \phi_i) \quad i, j = 1, 2, \ldots N. \tag{18}$$

We implement the above space-time discretisation using the DOLFIN [] problem solving environment of the FEniCS Project [].

3.1.3. Galerkin-POD discretisation

We now turn to the construction of reduced-order model of (10) using the Galerkin-POD approach. In the same manner we can write the reduced-order approximation of the solution $u_R \in V_R \subset V$ spanned using a POD basis $\{\vec{\varphi}_i\}_{i=1}^R$ with $\vec{\varphi}_i \in V_h$ and $R = \dim(V_R)$:

$$u_R(x) = \sum_{i=1}^{R} \vec{\varphi}_i u_{Ri}. \tag{19}$$

or in vector notation $u_R = \varphi\, \vec{u}_R$, where $\varphi = [\vec{\varphi}_1, \ldots, \vec{\varphi}_R] \in \mathbb{R}^{N \times R}$ is the matrix of the POD basis functions discretized on the finite element space V_h with $\dim(V_h) = N$.

We can then write a new set of discrete governing equations of (10) but instead of employing the finite element space V_h as in (17) we instead use the POD space V_R; for $n = 0, \ldots M$ find $\vec{u}^{\,n+1} \in \mathbb{R}^R$ such that:

$$((k^{-1} - \tfrac{c}{2})\widetilde{M} + \widetilde{K})\vec{u}_R^{\,n+1} = (k^{-1} + \tfrac{c}{2})\widetilde{M}\vec{u}_R^{\,n} - \widetilde{L}(\vec{u}_h^{\,\prime,n}) \tag{20}$$

where $\widetilde{M}, \widetilde{K} \in \mathbb{R}^{R \times R}$ are respectively the finite element mass and the stiffness matrices projected into the POD space:

$$\widetilde{M} = \varphi^T M \varphi, \quad \widetilde{K} = \varphi^T K \varphi, \tag{21}$$

and $\widetilde{L}(u) \in \mathbb{R}^R$ represents the projection of $L(u)$ into the POD space:

$$\widetilde{L}(u) = \varphi^T L(u).$$

Note that the terms in (20) involving $\widetilde{M}$ and the $\widetilde{K}$ are not a function of the solution u and can therefore be assembled once at the beginning of the online phase for subsequent re-use throughout the sequence of problems in (20). The the overall assembly cost is amortised across the online phase. In contrast, the non-linear term $\widetilde{L}(u) \in \mathbb{R}^R$ is a function of the current solution $\vec{u}_n^n$, and its computation involves the assembly of the term $L(u)$ on the finite element space and its projection into the POD space. As we do not know *a priori* how the the solution will evolve, clearly we must perform this operation *at every timestep* of the semi-implicit scheme.

3.1.4. Reduced assembly Galerkin-POD discretisation

It is the non-linear term $\widetilde{L}(u)$ to which it is *most* important to apply the proposed reduced assembly method. For the POD modes $\{\varphi_i\}_{i=1}^R$ represented in the original finite element space V_h, for a given tolerance tol, Algorithm 2 is applied and we obtain a sequence of meshes, each mesh associated with the i-th mode $\left\{\mathcal{T}_h^i\right\}_{i=1}^R$. These meshes are consequently associated with the finite element spaces $\left\{V_h^i\right\}_{i=1}^R$ with $\dim(V_h^i) = S_i$. As already discussed, for sufficiently localised POD basis functions $\{\varphi_i\}_{i=1}^R$ we expect that $S_i \ll N \; \forall i = 1, \ldots, R$.

We now reconstruct the Galerkin-POD problem on the mesh associated with the R-th mode $\mathcal{T}_h^R$. We construct a discrete interpolation operator $P : V_h \to V_h^R$ and interpolate every POD mode:

$$\vec{\varphi}_i|_R = P\vec{\varphi}_i \quad i = 1, \ldots, R \tag{22}$$

into V_h^R, resulting in a new approximation of the solution $u_R \in V_R|_R \subset V$ spanned by the interpolated POD basis $\{\vec{\varphi}_i|_R\}_{i=1}^R$ with $\vec{\varphi}_i \in V_h^R$ and $R = \dim(V_R) = \dim(V_R|_R)$ given by:

$$u|_R = \sum_{i=1}^{R} \vec{\varphi}_i|_R \times u_i|_R \tag{23}$$

or in vector notation $u|_R = \varphi|_R \, \vec{u}\,|_R$ with $\varphi|_R = [\vec{\varphi}_1|_R, \ldots, \vec{\varphi}_R|_R] \in \mathbb{R}^{S_R \times R}$.

With this procedure complete will simply follow the standard Galerkin-POD as in §3.1.3 but using the new POD basis functions $\varphi|_R$ interpolated in the finite element space V_h^R: for $n = 0, \ldots M$ find $\vec{u}^{\,n+1}|_R \in \mathbb{R}^R$ such that:

$$((k^{-1} - \tfrac{c}{2})\widehat{M} + \widehat{K})\vec{u}^{\,n+1}|_R = (k^{-1} + \tfrac{c}{2})\widehat{M}\vec{u}^{\,n}|_R - \widehat{L}(\vec{u}^{\,n}|_R) \tag{24}$$

where $\widehat{M}, \widehat{K} \in \mathbb{R}^{S_R \times S_R}$ are respectively the finite element mass and the stiffness matrices assembled on V_h^R and projected into the reduced assembly POD space:

$$\widehat{M} = \varphi^T|_R \, M\varphi|_R, \quad \widehat{K} = \varphi^T|_R \, K\varphi|_R, \tag{25}$$

and $\widehat{L}(u) \in \mathbb{R}^{S_R}$ represents the projection of $L(u)$ into the reduced assembly POD space:

$$\widetilde{L}(u) = \varphi^T|_R L(u). \tag{26}$$

3.1.5. Numerical results

In this section we solve the FKPP problem (10) with the POD (20) and the proposed RAPOD (24) methods and compare the results to the standard FEM method (**??**).

We set the position parameter controlling the centre of the source term as $x_0 = (0.55, 0.55)$. We note that this parameter is not in the set M_N.

All timing results were generated on a workstation with a 4-core Intel Core i7-6700 CPU with 32GB RAM running Ubuntu Linux 16.04.2 LTS. FEniCS is run inside a Docker container running the `quay.io/fenicsproject/stable:2017.1.0.r1` image. Running FEniCS inside a container leads to negligible computational overhead compared with running directly on the host system [].

The particulars of the standard FEM solver are given as follows. The 396 snapshots in the set M_N and the reference solution for x_0 are computed on a uniformly refined triangular cross-pattern mesh with 256×256 divisions resulting in 262144 cells. This leads to a finite element space $\dim(V_h) = 131585$. The resulting linear systems at one hundred timesteps (17) are solved using a algebraic multigrid (HYPRE BoomerAMG []) preconditioned conjugate gradient method in PETSc []. The preconditioner is re-used between timesteps as the matrix operator on the left-hand side of (17) does not change. The complete set of snapshots takes around 5 minutes to compute using 4 MPI processes and consumes 3.1GB space in an HDF5 [] file. The FEM reference solution is shown at three timesteps in fig. 5a.

Once the snapshot set M_N has been computed, the POD eigenvalue problem (**??**) is solved using the iterative Krylov Schur eigenvalue decomposition algorithm in SLEPc []. The computational time of the eigenvalue decomposition is negligible compared with reading in the snapshots from the HDF5 file and constructing the Gramian matrix (**??**). We compute the Gramian matrix and its eigenvalue decomposition in the l^2 inner-product space before re-orthonormalizing the eigenvectors using the classical Gram-Schmidt algorithm [] in the discrete L^2 inner-product space induced by the finite element basis, i.e. $(\psi_i, \psi_j)_{L^2(\Omega)} = \psi_i M \psi_j = 0 \; \forall i \neq j$ and $(\psi_i, \psi_i)_{L^2(\Omega)} = 1 \; \forall i$.

The first part of the spectrum of the POD eigenvalue problem (**??**) is shown in fig. 1. We retain $R = 17$ modes using the a 99.9% total energy cutoff criteria. Of course, we in no way claim that our choice of snapshots M_N or cutoff criteria is optimal. However, by fixing the POD problem (and the resulting error with respect to the standard FEM simulation) we set a baseline for comparison to which the proposed RAPOD technique can be compared.

With the POD spectrum length fixed at $R = 17$, we can now apply the RAPOD algorithm to construct the reduced mesh $\mathcal{T}^{17}$. Beginning with an initial mesh $\mathcal{T}^0$ covering the original domain Ω with a triangular right-aligned mesh with 2×2 divisions resulting in 8 cells. As an example, in fig. 4, for $\text{tol} = 10^{-2}$ we show the first four original unscaled POD modes and the corresponding meshes

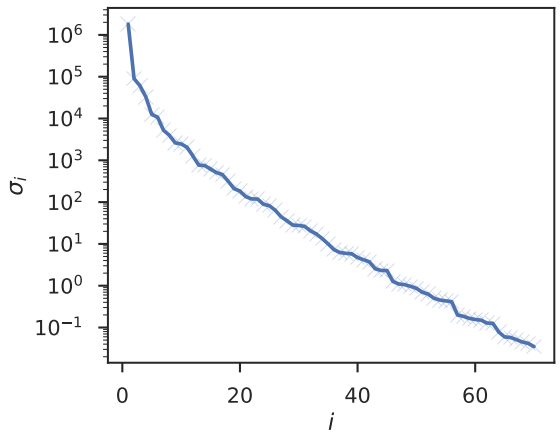


Figure 1: First part of spectrum of POD eigenvalue problem for the FKPP equation. We use the engineering criteria of retaining the portion of the spectrum which contains approximately 99.9% of the energy of the total spectrum. For the FKPP equation this criteria corresponds to retaining 17 modes.

$\mathcal{T}^i$ created by the RAPOD algorithm. Note how the algorithm refines the regions in the domain where the modes are most oscillatory, leaving the regions outside of the bottom quarter of the mesh relatively unrefined. The evolution of the number of cells in the meshes $\left\{\mathcal{T}^1, \ldots, \mathcal{T}^{17}\right\}$ is shown in fig. **??**. The final mesh has The final mesh $\mathcal{T}^{17}$ is shown in fig. 4.

We remark that our is also able to compute snapshots, perform eigenvalue decompositions and run the online phase in parallel using the MPI capabilities built-in to DOLFIN, PETSc and SLEPc. This parallelisation is transparent to the user and requires no modification to their code.

3.2. Reduction of a three-dimensional nonlinear quasi-static problem

In this subsection we use the reduction method described above to reduce the solution of a PDE describing a geometrically non-linear hyperelastic material on a three-dimensional domain.

The following total Lagrangian formulation of a Neo-Hookean hyperelastic material is standard, and a full description can be found in []. We repeat the essential details here. Consider a three-dimensional body $\mathcal{B}$ that can be modelled as a continuum. Let $\chi_0 : \mathcal{B} \to \mathbb{R}^3$ be the known reference configuration and $\chi : \mathcal{B} \to \mathbb{R}^3$ be the unknown deformed configuration after some external loads have been applied. The domain occupied by the undeformed configuration is then denoted $\Omega_0 := \chi_0(\mathcal{B})$ and in the deformed configuration $\Omega := \chi(\mathcal{B})$. Then, for every point p in the continuum body $\mathcal{B}$, related to points $X \in \Omega_0$ in the underformed configuration and $x \in \Omega$ in the deformed configuration through the maps χ_0 and χ respectively, we can construct a sufficiently smooth deformation

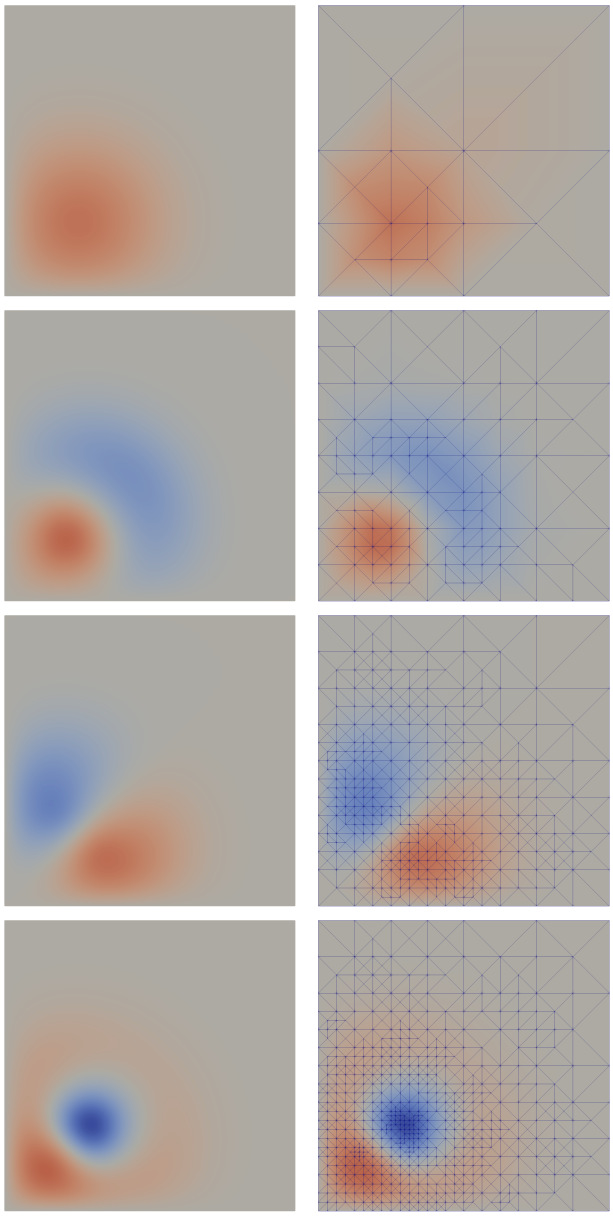


Figure 2: Left column: original unscaled POD modes $\sqrt{\sigma_i}\,\vec{\varphi}_i$ represented on the finite element function space V_h for $i = \{1, 2, 3, 4\}$. Right column: interpolated unscaled POD modes $\sqrt{\sigma_i}\,\vec{\varphi}_i|_R$ and corresponding reduced assembly meshes $\mathcal{T}^i$ obtained using the RAPOD algorithm with tol $= 10^{-2}$ for $i = \{1, 2, 3, 4\}$. Note that while these intermediate meshes form stages of the mesh refinement algorithm, we only use the mesh $\mathcal{T}^{17}$ in the online stage for all operations, see fig. 4.

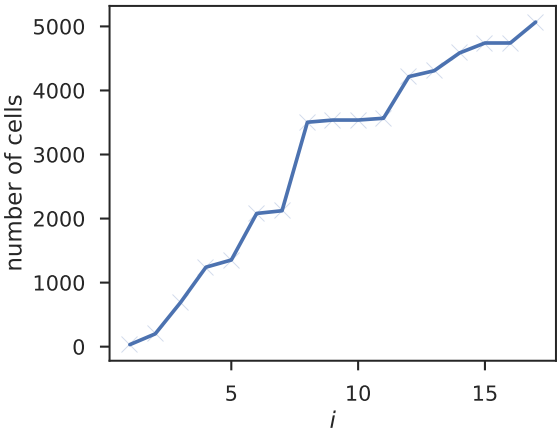


Figure 3: Evolution of the number of cells in the reduced assembly meshes $\{\mathcal{T}^1, \ldots, \mathcal{T}^{17}\}$. Note that it is sometimes the case that no refinement is necessary to acheive the integration tolerance on the subsequent mode, e.g. $\mathcal{T}^{15} = \mathcal{T}^{16}$.
??

map $\psi : \Omega_0 \ni X \to x \in \Omega$:

$$\psi = \chi \circ \chi_0^{-1} \tag{27}$$

from which we can define the deformation gradient tensor as:

$$F(X) := \frac{\partial \psi}{\partial X}, \tag{28}$$

with strictly positive determinant $\det(F) > 0$ everywhere in $\mathbb{R}^n$.

Following standard arguments, we define the right Cauchy-Green strain tensor and Green strain tensor as:

$$C := F^T F, \quad E = \frac{1}{2}(C - I) \tag{29}$$

from which we can define the following standard Neo-Hookean type strain energy density function:

$$W(X, I_C, III_C) := \frac{\mu}{2}(I_c - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2 \tag{30}$$

where $I_C := \mathrm{tr}(C)$ and $III_C = \det(C) = J^2 = [\det(F)]^2$ are the first and third invariants of the right Cauchy-Green strain tensor C and μ and λ are material constants related to the shearing and volumetric behaviour of the material.

The displacement field $u^* = \psi - X \in \mathcal{V}$ can be found as the solution to the

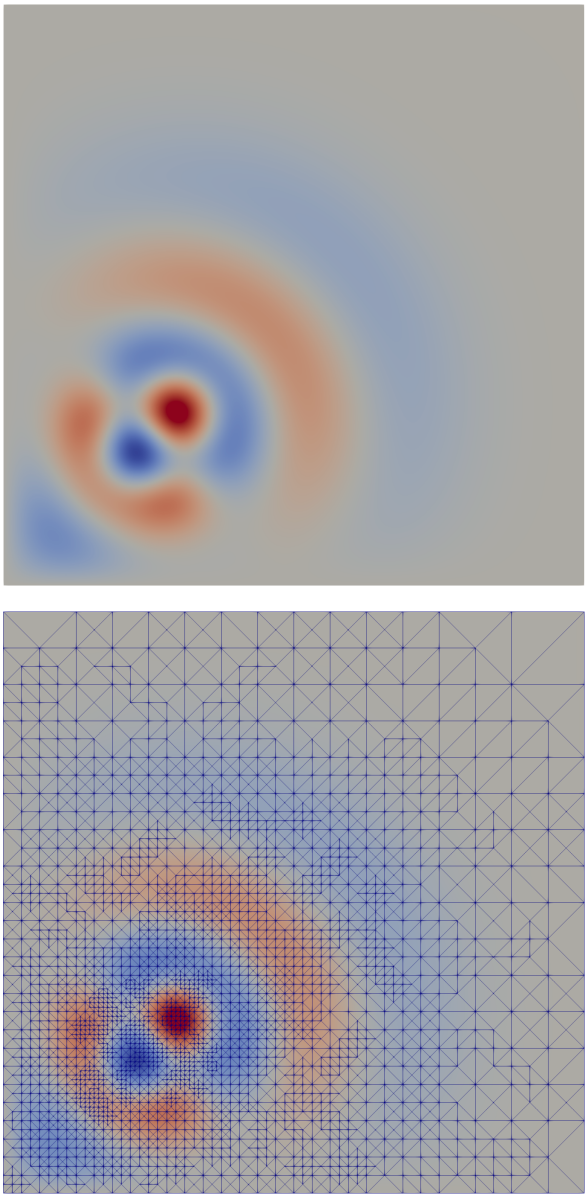
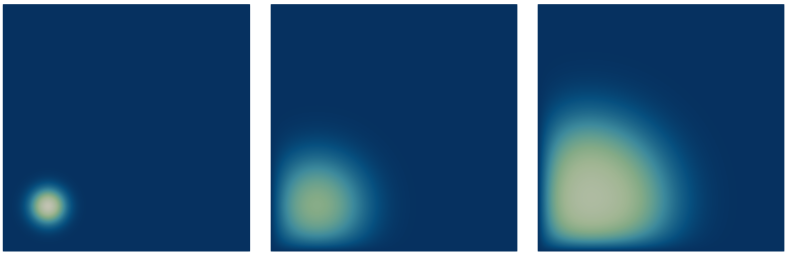
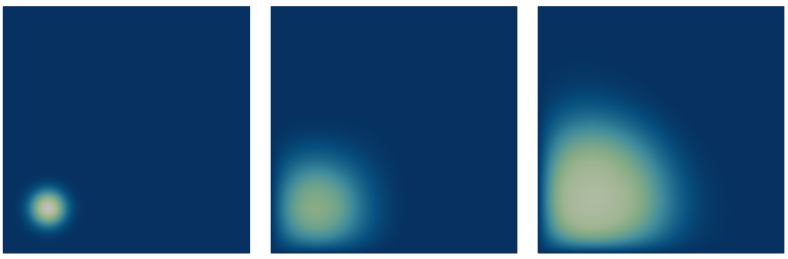


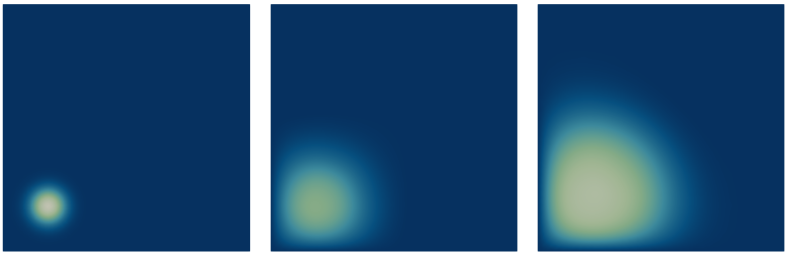
Figure 4: Top: original unscaled POD mode $\sqrt{\sigma_{17}}\,\vec{\varphi}_{17}$ Bottom: interpolated unscaled POD mode $\sqrt{\sigma}_{17}\,\vec{\varphi}_{17}|_R$ and corresponding reduced assembly mesh $\mathcal{T}^{17}$ obtained using the RAPOD algorithm with tol $= 10^{-2}$. The RAPOD procedure interpolates all 17 modes onto the associated function space V_h^{17} and uses it for all further assembly and projection operations. It is this mesh that is used to generate the RAPOD results with tol $= 10^{-2}$ in figs. 6, 7, 5, 8.



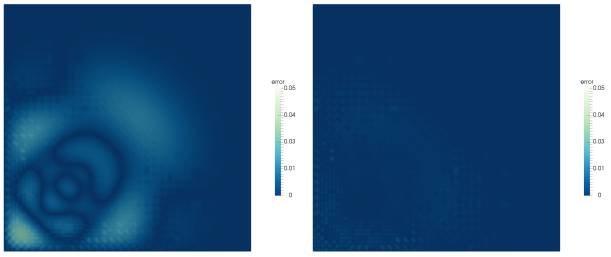
(a) FEM solution at $t = \{0.0, 0.05, 0.1\}$.



(b) POD solution at $t = \{0.0, 0.05, 0.1\}$.



(c) RAPOD solution with $\text{tol} = 10^{-2}$ at $t = \{0.0, 0.05, 0.1\}$.



(d) Pointwise error RAPOD-FEM (left) and RAPOD-POD (right) at final time $t = 0.1$.

Figure 5: Solutions and pointwise errors of FKPP problem using the three methods.

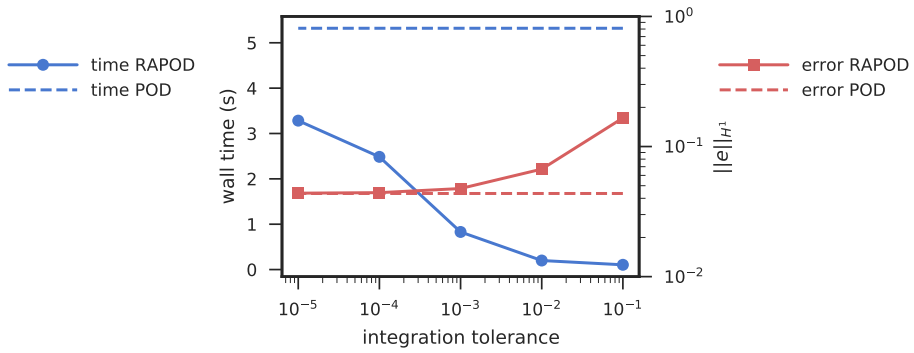


Figure 6: Wall time and relative error in H^1-norm with respect to FEM simulation against RAPOD integration tolerance tols $= \{10^{-1}, 10^{-2}, \ldots, 10^{-5}\}$. The red dashed line shows the error in standard POD simulation with respect to the FEM simulation (4.36%). As the RAPOD tolerance is decreased, the error committed by the RAPOD method converges to that of the standard POD method (tol $= 10^{-5}$ gives $||e||_{H^1} = 4.38\%$). The blue dashed line shows the wall time for the standard POD simulation (5.32 seconds). As the RAPOD tolerance is decreased, the wall time increases due to the greater work associated with assembling the larger finite element right-hand side at each timestep. With RAPOD tol $= 10^{-3}$ we get around a factor of 5 speed-up versus the standard POD method, with only a small increase in error $||e||_{H^1} = 4.76\%$. The RAPOD method gives a speed-up over the standard POD method for all tolerances. For comparison (not shown on graph) the full FEM solution for this problem takes 17.2 seconds including 11.9 seconds of linear solves and 5.1 seconds of assembly.

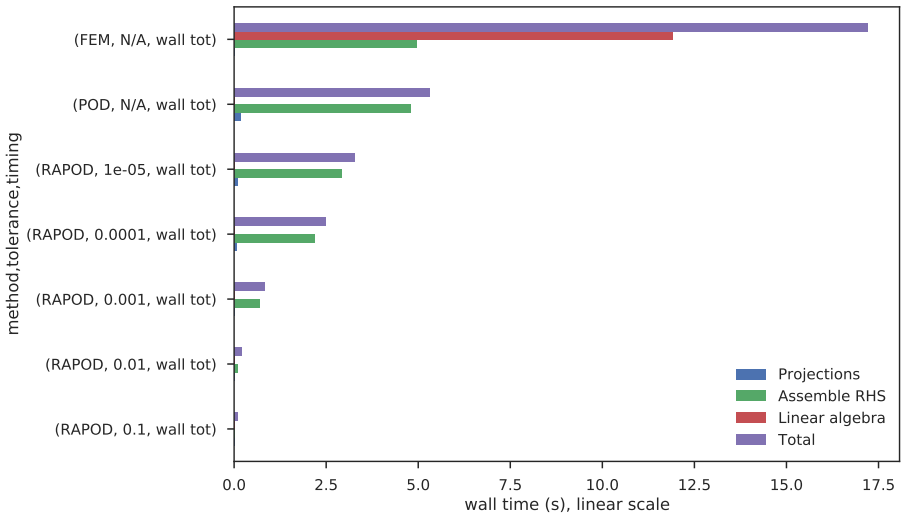


Figure 7: Breakdown of computational time spent in key stages of the total wall time for the FKPP problem: projections (POD and RAPOD), assembly of right-hand sides, linear algebra solves (preconditioned conjugate-gradient for FEM, dense Cholesky solves for POD/RAPOD). The same data is shown on a log scale in fig. 8. Note that on this linear scale the time taken by the linear algebra solves for the RAPOD and POD methods are not visible as they are dominated by assembly time.

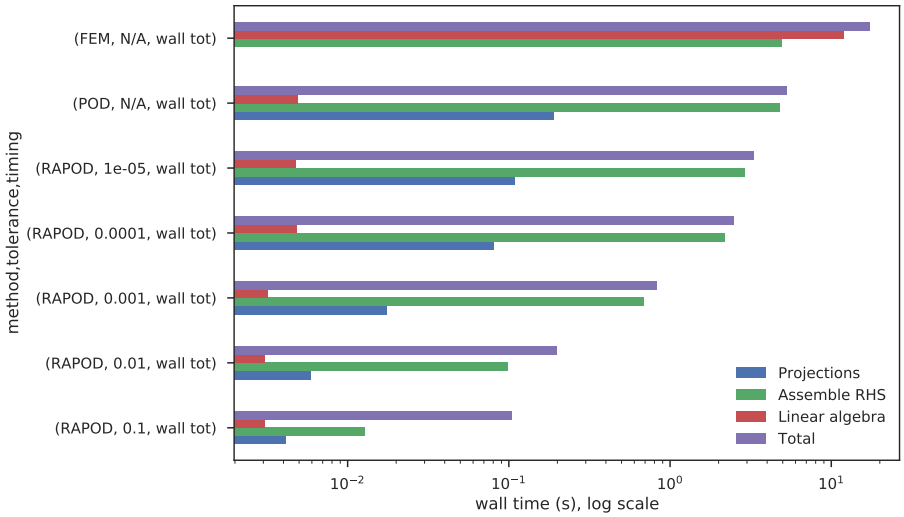


Figure 8: The same timing data is shown on a linear scale in fig. 7. With this scaling we can more clearly see the dominance of assembly over solution time for the RAPOD and POD methods.

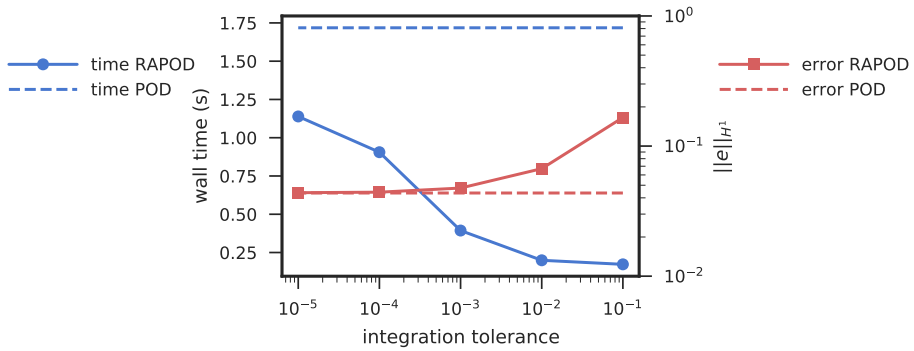


Figure 9: Initial results from parallel implementation of using MPI running on 4 MPI processes, cf. fig 6 for 1 MPI process. Assembly and projections are performed in parallel across all processes while the small dense Cholesky solve is performed serially on every process simultaneously. For the larger sized problems (POD and RAPOD tols $= \{10^{-3}, \ldots, 10^{-5}\}$) we can acheive a speed-up of around 3.1 times.

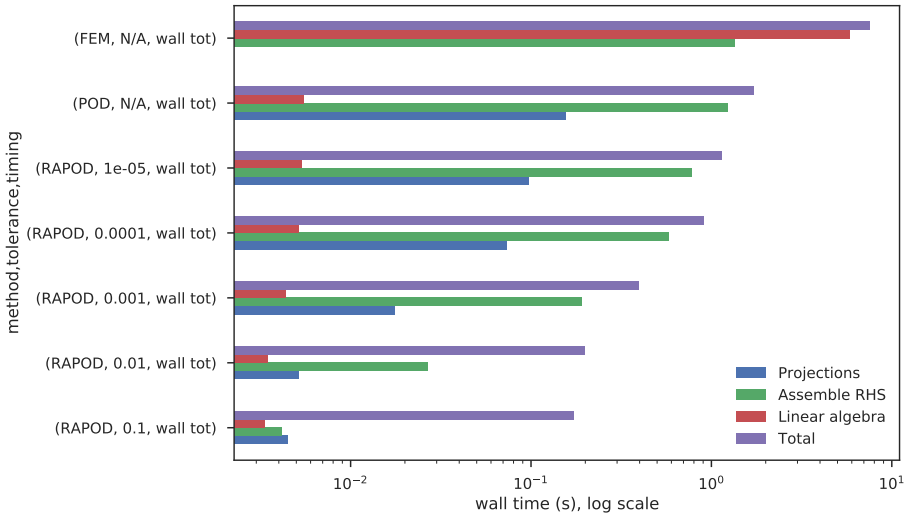


Figure 10: Breakdown of computational time spent in key stages of the total wall time for the FKPP problem running on 4 MPI processes, cf. fig. 8. For the loose RAPOD tolerances e.g. 10^{-2}, we have significant overhead unrelated to the main computational operations.

following minimisation problem:

$$u^* = \arg\min_{u \in \mathcal{V}} \left\{ \int_{\Omega_0} W dx_0 - \int_{\partial_N \Omega_0} t \cdot u \, ds_0 \right\} \tag{31}$$

$$= \arg\min_{u \in \mathcal{V}} \mathcal{E}(u) \tag{32}$$

where $\mathcal{V}$ is a sufficiently regular Hilbert space that satisfies the Dirichlet boundary condition $u = u_0$ on a portion of the boundary $\partial_D \Omega_0$, t are surface tractions (Neumann boundary conditions) applied on a portion of the boundary $\partial_N \Omega_0$ and dx_0 and ds_0 are measures on the undeformed domain and its boundary, respectively.

Assuming that a unique minimum exists, the first optimality condition can be written:

$$D_{\tilde{u}} \mathcal{E}(u^*) = F(u^*; \tilde{u}) = 0 \quad \forall \tilde{u} \in \mathcal{V} \tag{33}$$

where $D_{\tilde{u}} \mathcal{E}(u^*)$ denotes the usual Fréchet derivative of the functional $\mathcal{E}$ in a direction $\tilde{u}$ evaluated at the minimum u^*. The above equation can be intepreted as the equilibrium or residual equation. We use the notation $\tilde{u}$ to signify that these are test functions in a Galerkin sense.

The first order optimality condition can be written in full as:

$$\int_{\Omega_0} S(u^*) : D_{\tilde{u}} E(u^*) \, \mathrm{d}x - \int_{\partial_N \Omega_0} t \cdot \tilde{u} \, \mathrm{d}s = 0 \quad \forall \tilde{u} \in \mathcal{V}. \tag{34}$$

where the second Piola-Kirchoff stress tensor $S := 2\frac{\partial W}{\partial C}$, which is work conjugate with the incremental Green strain tensor with I the usual identity tensor.

$$D_{\tilde{u}} E(u^*) := \frac{1}{2} \left[\nabla(\tilde{u})^T F(u^*) + (F(u^*))^T \nabla \tilde{u} \right] \tag{35}$$

Clearly the residual is non-linear in the displacement unknown u. The standard method for solving this problem is Newton's method which requires the Jacobian, formally the further Fréchet derivative of the residual equation which we now about an arbitrary point in solution space u_-:

$$J(u_-; \bar{u}; \tilde{u}) := D_{\bar{u}} F(u_-; \tilde{u}) = \int_{\Omega_0} 2 \frac{\partial S}{\partial C} : D_{\tilde{u}} E(u_-) \, dx_0 \tag{36}$$

$$+ \int_{\Omega_0} S : D_{\bar{u}} [D_{\tilde{u}} E(u_-)] \, \mathrm{d}x_0 \tag{37}$$

with:

$$D_{\tilde{u}} E(u_-) = \frac{1}{2} \left[\nabla(\bar{u})^T \nabla \tilde{u} + (\nabla \tilde{u})^T \nabla \bar{u} \right] \tag{38}$$

and the notation $\bar{u}$ used to signify that these are trial functions in a Galerkin sense.

This results in the following system of equations to be solved at each step of Newton's method:

$$J(u^k; \delta \bar{u}; \tilde{u}) = -F(u^k; \tilde{u}) \tag{39}$$

$$u^{k+1} = u^k + \delta \bar{u} \tag{40}$$

The Fisher-KPP system we reduced in the previous section led to a discrete time dependent system where the linear form on the right hand side was dependent on the solution u^k at the previous time step. In contrast, the hyperelastic problem we consider in this section leads to a Newton system containing both a bilinear form (Jacobian) J and linear form (residual) F that are dependent on the solution at the previous Newton step u^k. Despite these differences, we will see that the the techniques used to construct the reduced integration meshes and the reduced bilinear and linear operators are almost the same.

3.2.1. Problem discretization

We discretise the full unreduced problem at each Newton iteration using a standard Galerkin H^1-conforming finite element space $\mathcal{V}_h \subset \mathcal{V}$:

$$u(x) \simeq \sum_{i=1}^{\dim(\mathcal{V}_h)} u_i v_i(x),\ u_i = \langle u, v_i \rangle, \forall i = 1, \dots, \dim(V_h) \tag{41}$$

We are in particular interested in solving equation (**??**) on a 3d cube shaped mesh where a traction force is applied on a small part of the boundary, see Figure **??**. In order to solve this problem we apply a Newton-Raphson algorithm with a continuation method for the traction force. Let us consider a FE discretization of size $\mathcal{N}$ of the problem. At each sub-iteration of the Newton method the problem reads

$$\mathbf{J}(u)\mathbf{du} = \mathbf{R}(u) \tag{42}$$

where $\mathbf{J}(u) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is the Jacobian matrix and $\mathbf{R}(u) \in \mathbb{R}^{\mathcal{N}}$ is the residual vector that depend on the displacement at the previous iteration, and $\mathbf{du} \in \mathbb{R}^{\mathcal{N}}$ is the actual increment of displacement. It is clear that if the size of the discrete system $\mathcal{N}$ is large, the resolution of the problem is very time consuming since the Jacobian and the residual need to be updated at each sub-iteration of the algorithm. The application of a RO method can indeed reduce this computational costs.

HERE WE SHOULD PROBABlY INCLUDE SOME COMMENTS ABOUT PGD ... "LIGHT" COMMENTS!

In our approach, we do not apply any linearization of the problem but solve it with a Newton-Raphson method on the RO space. We apply a method similar to the one proposed in [26]. In [26] the heat convection equations are solved with a Newton algorithm where the Jacobian matrix is computed and projected once for all in the *off-line* phase thanks to some tensor properties and to the fact that the variational problem can be written with some tri-linear forms. In our case, this is not possible and the Jacobian and the residual need to be updated at every iteration due to their strong non-linearities.

3.2.2. Numerical results

We apply a reduced integration POD method as proposed above. Let us consider a cubic mesh of size $6 \times 6 \times 6$. We build our POD basis using the snapshots of several simulation run with a FE method on a regular mesh with

226981 nodes and 1296000 tetrahedra. Each simulation corresponds to a traction force applied in one of the grid points defined in Figure **??** (left), and fixed parameters $E = 10.0$, $\nu = 0.3$. We apply a force with increment of 0.1, up to 6.0. The spectrum of the POD basis is plotted in Figure **??**. Then, we apply Algorithm 2 with fixed tolerance of 10% to the POD basis and get the meshes for the reduced integration. In Figure **??** we plot the number of nodes of the meshes obtained with different number of modes, we observe that the size of these meshes stabilize to 17.5K for more than 40 modes. This mesh is shown is Figure **??**(b).

In Figure **??**(b) we can see the displacement obtained with our reduced integration POD method compare to the reference solution (FEM, Figure **??**(a)). We can see that our method give good results with only 30 modes and about 6K points. Also, the (absolute) error is shown in Figure **??**(c) and we see that is less than 0.1 (on a domain of size $6 \times 6 \times 6$.

Also, we analyze the relative error compared to the number of modes both with standard POD and reduced integration POD, see Figure **??**. Analogously to the mesh size, the relative error with reduced integration POD method stabilizes with more than 40 modes and is less than 10%. In Figure **??** we compare the maximum displacement (x axes) versus the applied traction force (y axes) with different sizes of the RO method. We observe that the curve converges to the reference one (FE, blue line) and with only 40 modes we can reproduce the characteristic non-linear behavior.

Finally, let us compare the computational time with standard and reduced integration POD methods with the FEM. In Figure **??** we see that both standard and reduced integration POD methods are about 10^5 times faster than the FEM during the solving of the linear system. Since the assembling of the Jacobian and the residual is very time consuming with the standard POD method, the total computational time of this method is comparable with the FEM. On the opposite, the reduced integration POD method is about 25 times faster than the FEM (with 40 to 100 modes and 6K points) since the assembling of the reduced system is faster.

[1] J.S. Amelang, G.N. Venturini, and D.M. Kochmann. Summation rules for a fully nonlocal energy-based quasicontinuum method. *Journal of the Mechanics and Physics of Solids*, 82:378–413, 2015.

[2] B. Baroli. *Multiscale domain decomposition methods for high heterogeneous Darcy flows*. PhD thesis, Department of Mathematics, Politecnic University of Milan, 2016.

[3] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. Paris*, 339:667–672, 2004.

[4] L.A.A. Beex, R.H.J. Peerlings, and M.G.D Geers. Central summation in the quasicontinuum method. *Journal of the Mechanics and Physics of Solids*, 70:242–261, 2014.

[5] L.A.A. Beex, R.H.J. Peerlings, and M.G.D Geers. Quasicontinuum-based multiscale approaches for plate-like beam lattices experiencing in-plane and out-of-plane deformation. *Computer Methods in Applied Mechanics and Engineering*, 279:348–378, 2014.

[6] M. Boulakia, E. Schenone, and J.F. Gerbeau. Reduced-order modeling for cardiac electrophysiology. application to parameter identification. *International Journal for Numerical Methods in Biomedical Engineering*, 28(6-7):727–744, 2012.

[7] S. Chaturantabut and D. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal of Scientific Computing*, 32(5):2737–2764, 2010.

[8] S. Chaturantabut and D. Sorensen. A state space error estimate for pod-deim nonlinear model reduction. *SIAM Journal of Numerical Analysis*, 50(1):46–63, 2012.

[9] F. Chinesta, A. Ammar, and E. Cueto. Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. *Archives of Computational Methods in Engineering*, 17(4):1287–1296, 2010.

[10] P.G. Constantine and Q. Wang. Residual minimizing model interpolation for parametrized nonlinear dynamics systems. *SIAM Journal of Scientific Computing*, 34(4):A21118–A2144, 2012.

[11] J.F. Gerbeau, D. Lombardi, and E. Schenone. Reduced order model in cardiac electrophysiology with approximated lax pairs. *Advances in Computational Mathematics*, 41(5):1103–1130, 2015.

[12] M.A. Grepl, Y. Maday, N.C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(3):575–605, 2007.

[13] M. Gunzburger and Y. Zhang. A quadrature-type approximation to the quasi-continuum method. *Multiscale Modeling and Simulation*, 8:571–590, 2010.

[14] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 23:417–441, 1933.

[15] P. Kerfriden, O. Goury, T. Rabczuk, and S.P.A. Bordas. A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 256:169–188, 2013.

[16] J. Knap and M. Ortiz. An analsys of the quasicontinuum method. *Journal of the Mechanics and Physics of Solids*, 49:1899–1923, 2001.

[17] P. Ladevèze, J.C. Passieux, and D. Néron. The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21):1287–1296, 2009.

[18] J.L. Lumley. The structure of inhomogeneous turbulence. *Atmospheric turbulence and radio wave propagation*, pages 166–178, 1967.

[19] Y. Maday and E.M. Rønquist. A reduced-basis element method. *Journal of scientific computing*, 17(1):447–459, 2002.

[20] A. Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1603–1626, 2010.

[21] K. Pearson. On lines and planes of closest fit to systems of points. *Philosophical Magazine*, 2(6):559–572, 1901.

[22] C. Prud'Homme, D.V. Rovas, L. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, 2002.

[23] A. Radermacher and S. Reese. Pod-based model reduction with empirical interpolation applied to nonlinear elasticity. *International Journal for Numerical Methods in Engineering*, 2015.

[24] D. Ryckelynck. Hyper-reduction of mechanical models involving internal variables. *International Journal for Numerical Methods in Engineering*, 77(1):75–89, 2008.

[25] D. Ryckelynck and D.M Benziane. Multi-level a priori hyper-reduction of mechanical models involving internal variables. *Computer Methods in Applied Mechanics and Engineering*, 199(17-20):1134–1142, 2010.

[26] E. Schenone, S. Veys, and C. Prud'Homme. High performance computing for the reduced basis method. application to natural convection. *ESAIM: Proceedings*, 43(December):255–273, 2013.

[27] L. Sirovich. Turbulence and the dynamics of coherent structures. part i: Coherent structures. *Quarterly of Applied Mathematics*, 45:561–571, 1987.

[28] L. Sirovich. Low dimensional description of complicated phenomena. *Contemporary Mathematics*, 99:277–305, 1989.

[29] E.B. Tadmor, R. Philips, and M. Ortiz. Mixed atomistics and continuum models of deformation in solids. *Langmuir*, 12:4529–4534, 1996.

[30] Q. Yang, E. Biyikli, and A. To. Multiresolution molecular mechanics: statics. *Computer Methods in Applied Mechanics and Engineering*, 258:26–38, 2013.