Abstract

At every iteration or timestep of the online phase of some reduced-order modelling schemes, large linear systems must be assembled and then projected onto a reduced order basis of small dimension. The projected small linear systems are cheap to solve, but assembly and projection are now the dominant computational cost. In this paper we introduce a new hyper-reduction strategy called reduced assembly (RA) that drastically cuts these costs. RA consists of a triangulation adaptation algorithm that uses a local error indicator to construct a reduced assembly triangulation specially suited to the reduced order basis. Crucially, this reduced assembly triangulation has fewer cells than the original one, resulting in lower assembly and projection costs. We demonstrate the efficacy of RA on a Galerkin-POD type reduced order model (RAPOD). We show performance increases of up to five times over the baseline Galerkin-POD method on a non-linear reaction-diffusion problem solved with a semi-implicit time-stepping scheme and up to seven times for a 3D hyperelasticity problem solved with a continuation Newton-Raphson algorithm. The examples are implemented in the DOLFIN finite element solver using PETSc and SLEPc for linear algebra. Full code and data files to produce the results in this paper are provided as supplementary material.

Keywords: hyper-reduction, reduced order modelling, model order reduction, non-linear PDEs, hyperelasticity, FKPP.

1. Introduction

The response of many physical systems across the sciences and engineering can be predicted by finding the solution of non-linear and possibly time and parameter-dependent coupled partial differential equation (PDE). This PDE cannot usually be solved using analytical approaches and we must resort to
numerical methods. In the non-linear case, the original infinite-dimensional problem can be transformed into a sequence of discrete linear algebra problems of finite dimension which can in turn be solved using Newton-Raphson’s method. In the time-dependent case, we have a sequence of problems discretised in both space and time, e.g. with finite differences.

The construction and solution of this sequence of linear algebra problems is often computationally demanding. In each step we can split the computational cost between two phases:

i. The assembly of a large and sparse linear system;

ii. The solution of that linear system.

The assembly and solution procedure repeats itself until some convergence criteria is met, or the final time is reached. It is generally true that for any sufficiently large problem, the cost of the solution of the linear system dominates assembly. An obvious way to reduce computational cost is to reduce the size of the linear systems. In the context of the finite element method this can be achieved by, for example, using a posteriori error estimation techniques to drive triangulation adaptivity, see e.g. [28]. The triangulation adapts locally to the solution. The adapted triangulation has fewer cells leading to reduced computational cost. Another approach to reduce computational cost is reduced-order modelling (ROM). ROM is used in many notable applications, e.g. real-time simulation [29], accelerating parametric studies and stochastic simulation [13].

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 in the offline phase to produce an online phase where we can compute thousands of new solutions at points in parameter space that were not previously computed offline. 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 been computed, the information within them is in some way compressed into a more efficient representation. In the reduced basis (RB) approach, e.g. [27, 31, 14, 33], a linear combination of the snapshots selected by a greedy procedure are used as the global basis. In the Galerkin proper orthogonal decomposition (Galerkin-POD) approach e.g. [30, 23, 26, 35, 34, 8, 32] 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 the proper generalised decomposition approach (PGD) e.g. [12, 29] the solution is expressed in a separated representation which can be solved independently. In this paper we exclusively work with basis functions computed with the POD approach, but other approaches for constructing reduced order models such as RB could be considered.

We now turn to the online phase. In this phase we would like to quickly compute many thousands of new solutions at points in the parameter space that were not previously computed offline in the set of snapshots. We do this by using the POD modes to span or interpolate the solution space. Note that
we are still dealing with a non-linear and/or time-dependant problem, 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 Galerkin-POD setting consists of the following steps:

i. The assembly of a large and sparse linear system arising from the finite element discretisation.
ii. That linear system is projected through the POD basis functions, resulting in a small and dense linear system.
iii. The solution of the small and dense linear system to find the Newton-Raphson update or the next solution in time.
iv. Projection of the reduced solution back to the finite element space.

The assembly, projection, solution and reconstruction procedure repeats itself until some convergence criteria is met, or the final time is reached. The issue is that the solution of the small linear system is now so much cheaper that the assembly step dominates the overall cost of the online phase. In short, because of the effectiveness of the POD approach, the main computational burden shifts from solution back to assembly.

A large class of techniques commonly referred to as hyper-reduction techniques have been created to reduce the burden of assembly. Hyper-reduction methods usually work by locating a set of points and weights in the domain which are important for capturing non-linearity. Empirical interpolation (EIM) or discrete empirical interpolation (DEIM), can be used to approximate parametric or non-linear functions by separable interpolation functions. Other approaches include missing point estimation, Gappy-POD or Gauss-Newton Approximated Tensors (GNAT). Simply put, these methods work by identifying a subset of terms in the non-linear equations that are important. In the case of DEIM, this subset consists of a set of so-called magic points in space. Implicit here is the idea that the key non-linearities are somehow locally supported in space. The quasicontinuum (QC) method uses this idea to select representative points in space to create continuum representations of underlying discrete models.

The method proposed in this paper is a novel hyper-reduction method. We call this method the Reduced Assembly (RA) method. RA is conceptually simple, can be implemented in many off-the-shelf finite element codes with little or no modification, and provides significant computational gain over standard reduced-order approaches in the case of non-linear or time-dependent problems. We apply the reduced assembly approach to a standard Galerkin-POD formulation, resulting in what we call the Reduced Assembly Proper Orthogonal Decomposition (RAPOD) method.

A high-level overview of RAPOD is as follows; we use a standard Galerkin-POD procedure to generate a set of basis functions from a set of solution snapshots in parameter space. The basis functions are global and oscillatory. The key idea behind RA is that the POD or RB functions act as indicators of the region of the domain that are important or not. The RA method uses a simple error indicator and standard local piecewise basis functions to drive the adaptation.
of a new problem triangulation which is specially tailored to the reduced-order basis functions. The location and weights of the integration points in the triangulation comes naturally and automatically as in the finite element method. This adapted triangulation typically has far fewer cells than the original triangulation. The POD basis functions are then interpolated to this new adapted triangulation. The online phase is exactly the same as the standard POD procedure, except that the discrete finite element operators are assembled on this adapted triangulation and projected through the interpolated POD basis functions. Note that the RA approach is general and could be adapted to other cases where numerical integration and assembly is particularly costly, e.g. high-order non-polynomial approximations such as meshfree methods or isogeometric analysis.

An outline of this paper is as follows; in section 2 we give an overview of the generic problem formulation, before giving a reminder of the standard POD procedure in section 3. A detailed description of the new RAPOD method is given in section 4. Finally, illustrative numerical results are shown in section 5, before concluding in section 6.

2. Problem formulation

Our starting point is the following weak residual form of a system of nonlinear and/or time-dependent partial differential equations. Find \( u \in V(\Omega) \times \mathbb{R} \) such that:

\[
\forall (\hat{u}, t) \in V \times \mathbb{R}^+, \quad F(u, t, m; \hat{u}) = 0,
\]

where \( t \in \mathbb{R}^+ \) 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 \( \hat{u} \in V \)) following the semicolon. Unless otherwise stated in what follows, inner products \( (\cdot, \cdot) \) are assumed to be the usual \( L^2(\Omega) \) inner-product.

The RAPOD method is applicable to the following two broad classes of discrete solution methods that can be applied to (1):

In the non-linear stationary case, we first apply the Newton-Raphson method and then discretise in space \( u \approx u_h \in V_h \) where \( V_h \) is some finite element function space of dimension \( N \). This results in a sequence of discrete linear problems (commonly called Newton steps): for \( k = 0, \ldots, M \) and an initial guess \( u^0 \in \mathbb{R}^N \), find \( \delta u^k \in \mathbb{R}^N \) such that:

\[
J(u^k)\delta u^k = -f(u^k),
\]

\[
u^{k+1} = u^k + \delta u^k,
\]

where \( J(u^k) \in \mathbb{R}^{N \times N} \) is the matrix arising from the finite element discretisation of the Jacobian:

\[
J(u^k; \hat{u}; \bar{u}) := D_u F(u^k; \bar{u}),
\]

where \( D_u \) denotes the usual Fréchet derivative, and \( f(u^k) \in \mathbb{R}^N \) is the vector arising from the finite element discretisation of the residual \( F \). The number of
iterations \( M \) is controlled by a stopping criteria e.g. \( \| f \|_2 < \epsilon \). We remark on the dependence of \( J \) and \( f \) on the current solution \( u_k \); i.e. they must typically be re-assembled at every Newton step.

In the time-dependent case, after first discretising in time \( u^n \approx u(t^n) \) at \( M + 1 \) times \( t^0 < t^1 < \cdots < t^M \in \mathbb{R}^+ \) and then in space \( u(t^n) \approx u_h(t^n) \in V_h \), we end up with a sequence of discrete linear problems to solve: for \( n = 0, \ldots, M \) and initial condition \( u^0 \in \mathbb{R}^N \), find \( u^n \in \mathbb{R}^N \) such that:

\[
K(u^n)u^{n+1} = b(u^n),
\]

where \( K(u^n) \) and \( b(u^n) \) are matrices and vectors arising from the finite element discretisation of the semi-discrete solution in time. Again, we explicitly remark on the possible dependence of \( K \) and \( b \) on the current solution \( u^n \), i.e. they must typically be re-assembled at every timestep.

In both cases, we end up with a sequence of linear problems that must be solved. Critically, the matrix and vector operators in the linear problems are dependent on the current and/or previous solutions and must be assembled at every step.

3. Proper Orthogonal Decomposition

Let us briefly recall the main steps to construct a proper orthogonal decomposition basis. More details can be found for instance in \([26, 34]\). We pay no special attention to the quality of the POD procedure in this paper; instead the totally standard POD procedure outlined below forms a benchmark by which the proposed RAPOD procedure can be assessed. Further improvements to the POD procedure (e.g. snapshot selection) would result in improvements to the results of the RAPOD method.

For simplicity we work in a finite-dimensional setting \( V_h \equiv \mathbb{R}^N \) with a set of solution snapshots \( U = \{ u_1, u_2, \ldots, u_S \} \in \mathbb{R}^{N \times S} \) generated at discrete points \( m^k \in \mathbb{R}^+ \times M \) in time and the parameter space by repeatedly solving (1). These points are specified in a discrete set \( M_N = \{ m_1, \ldots, m_S \} \) of size \( S \). In POD we seek a \( R \)-dimensional subspace \( V_R \subset V_h \) which optimally represents the data contained in the snapshots \( U \) with the smallest possible \( R \), i.e. we seek a projection operator \( \Pi : V_h \to V_R \) that minimises the following least-squares distance:

\[
\sum_{i=1}^{S} ||u_i - \Pi u_i||_{V_h}^2.
\]

It is well known that the optimal basis in the above least-squares sense \( \Phi = \{ \varphi_1, \ldots, \varphi_R \} \in \mathbb{R}^{N \times S} \) spanning \( V_R \) can be found by solving the following eigenvalue problem:

\[
C \psi_i = \lambda_i \psi_i, \quad \varphi_i = \frac{1}{\sqrt{\lambda_i}} U \psi_i, \quad i = 1, \ldots, N,
\]

where \( C = U^T U \in \mathbb{R}^{S \times S} \) is the empirical correlation matrix of the snapshots \( U \). In practice, only the leading \( R \ll S \) eigenvalues and eigenvectors containing
the dominant energy of the spectrum are retained. We can then write the following ansatz for the solution in the space $V_R$:

$$u \approx u_R = \sum_{i=1}^{R} \varphi_i(x) u_{Ri} = \Phi u_R$$

(7)

An additional practicality is that we do not work with the ansatz (7) directly. Instead, we use the POD-Galerkin procedure, directly projecting the finite element linear algebra objects assembled on the full space $V_h$ onto $V_R$ in the online phase, e.g. for the Newton scheme (2):

$$\tilde{J} = \Phi^T J \Phi \in \mathbb{R}^{R \times R}, \quad \tilde{f} = \Phi^T f \in \mathbb{R}^R,$$

(8)

or in a timestepping scheme (4):

$$\tilde{K} = \Phi^T K \Phi \in \mathbb{R}^{R \times R}, \quad \tilde{b} = \Phi^T b \in \mathbb{R}^R.$$

(9)

4. Reduced assembly POD method

In this subsection we present the algorithm to adaptively construct the reduced assembly triangulation necessary to reduce the expense of computing the non-linear terms (3, 3) in the reduced order space.

Summarising the approach detailed below, we adaptively generate a sequence of refined triangulations 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 triangulation adaptivity process for the global POD basis functions that results in a triangulation specially suited to the reduced order problem. Clearly, this idea is related to the common idea of driving an adaptation of the triangulation based on the solution $u$, except that in our case we perform all of our adaptation offline based on the information contained in the POD basis. For a given tolerance our algorithm only needs to be applied once in the offline stage, as long as the snapshot space and POD basis remains the unchanged.

The algorithm relies on a combination of tools readily found in most finite element toolboxes, namely; adaptive refinement, projection and interpolation between functions on non-matching triangulations, assembly of bilinear and linear forms and a few basic linear algebra operations. We have chosen to implement our method using DOLFIN [25], part of the FEniCS Project [1], but most finite element toolboxes could be used with little effort. Linear algebra operations are all performed using PETSc [4] and SLEPc [22].

We start with the original problem triangulation $T_h^0$ consisting of $N_0$ simplicial quasi-uniform cells $\{T_1^0, \ldots, T_N^0\}$ that form an approximation $\Omega_h$ to the problem domain $\Omega$:

$$\Omega \approx \Omega_h = \bigcup_{i=1}^{N_0} T_i^0.$$  

(10)
Using this triangulation and the standard methodology outlined in §3 we generated the solution snapshots \( U \) from which we can calculate \( R \) POD basis functions \( \{ \varphi_1, \ldots, \varphi_R \} \).

We then create the coarsest possible triangulation \( T^1_h \) with \( N_1 \) simplicial cells \( \{ T^1_1, \ldots, T^1_{N_1} \} \) that also covers the problem domain:

\[
\Omega_h = \cup_{i=1}^{N_0} T^1_i
\]  

(11)

Note that \( N_1 \ll N_0 \).

Then, starting with POD basis function \( \varphi_1 \), we begin the adaptation/refinement of triangulation \( T^1_h \). The POD basis function \( \varphi_1 \) has global support and is interpolated on the finite element function space \( V_h(T^0_h) \) that was used to generate the snapshots of the non-linear problem \( \Omega \). Furthermore, it is the mode associated with the leading eigenvalue of (6) and contains the fewest oscillations.

For every cell \( T^1_i \) of the coarse triangulation \( T^1_h \) we calculate the following cell error indicator that represents the local error in the integral of \( |\varphi_1| \):

\[
\eta^i_h = \int_{T^1_i(Q_n)} |\varphi_1(x)| \, dx - \int_{T^1_i(Q_{n+1})} |\varphi_1(x)| \, dx,
\]  

(12)

where \( T^1_i(Q_n) \) denotes that the corresponding integral is calculated using a quadrature rule of order \( n \) on the simplex \( T^1_i \). The notation \( |\cdot| \) means to take the absolute value of the argument. Note that in the numerical integration procedure the value of the function \( \varphi_1 \) at any point \( x \in \Omega \) is evaluated using the basis functions in the finite element function space \( V_h \) defined on the original triangulation \( T^0_h \). From a heuristic perspective, assuming that basis function \( \varphi_1 \) is sufficiently smooth, the cell error indicator \( \eta^i_h \) tells us of whether further refinement of the cell \( T^1_i \) could lead to an improved approximation to the integral of the basis function \( \psi_1 \).

We calculate two further global quantities. The first, \( I_e \), is the integral of the absolute value of the basis function \( \varphi_1 \) on the original triangulation \( T^0_h \) using a sufficiently high quadrature rule of order \( m \) (with respect to the polynomial order of the basis functions in \( V_h \)):

\[
I_e = \sum_{i=1}^{N^0} \int_{T^0_i(Q_m)} |\varphi_1| \, dx.
\]  

(13)

This can be considered the ‘exact’ integral of \( |\varphi_1| \). The second quantity is the approximate integral \( I_a \) of the absolute value of the mode calculated on the coarse triangulation \( T^1_h \):

\[
I_a = \sum_{i=1}^{N^1} \int_{T^1_i(Q_n)} |\varphi_1| \, dx.
\]  

(14)

We then calculate the following global relative error:

\[
E = |I_e - I_a| / I_e
\]  

(15)
If $E$ is greater than some specified tolerance, e.g. 5%, we use the cell-based error indicators $\eta_i$ to mark some cells in triangulation $T_h^0$ for refinement following the strategy proposed by Dörfler [15]. In short, the cells are ordered in reverse numerical order of their local error and then a fixed proportion is refined. We then repeat the above algorithm for mode $\varphi_1$ as many times as necessary to drop the error $E$ below the specified tolerance.

Once $E$ is sufficiently small, we have our specially adapted triangulation $T_h^1$ for basis function $\varphi_1$. We then repeat the above process using the triangulation $T_h^1$ as input for the adaptive refinement algorithm on basis function $\varphi_2$ to find $T_h^2$, and so on until we have performed the triangulation adaptation for all $R$ basis functions. The result is a triangulation $T_h^R$.

A more formal version of the procedure is detailed in Algorithm 1. We would encourage the reader to refer to fig. 4 for a visual representation of the output of the RAPOD algorithm.

**Algorithm 1 Reduced assembly triangulation refinement algorithm.**

Original triangulation $T_h^0$ with $N_0$ simplicial cells.
Set of global basis functions, $\{\varphi_1, \ldots, \varphi_R\}$ defined on a function space $V_h$ on the original triangulation $T_h^0$.
Initial triangulation $T_h^1$ with $N_1$ simplicial cells with $N_1 \ll N_0$.
A tolerance on the integration error, tol.
Refinement percentage, ref.
Calculate ‘exact’ integral $I_e = \sum_{j=1}^{N_0} \int_{T_h^0(j)} |\varphi_i(x)| \, dx$.
Calculate ‘approximate’ integral $I_a = \sum_{k=1}^{N_1} \int_{T_h^1(k)} |\varphi_i(x)| \, dx$.
Calculate $E = |I_e - I_a|/I_e$.
for $i = 1 : R$ (loop over modes) do
  Let $N_i$ be the number of cells in current triangulation $T_h^i$.
  while $E > \text{tol}$ do
    for $l = 1 : N_i$ (loop over basis functions) do
      Calculate local error estimator
      $\eta_l = \int_{T_h^i(l)} |\varphi_l(x)| \, dx - \int_{T_h^i(l)} |\varphi_{l+1}(x)| \, dx$.
    end for
    Based on $\eta_l$, refine triangulation $T_h^i$ with Dörfler algorithm.
    Calculate ‘approximate’ integral $I_a = \sum_{k=1}^{N_i} \int_{T_h^i(k)} |\varphi_l(x)| \, dx$.
    Calculate $E = |I_e - I_a|/I_e$.
  end while
  Let $T_h^{i+1}$ be $T_h^i$.
end for
return $T_h^R$.

We now reconstruct the Galerkin-POD problem on the triangulation associated with the $R$-th mode $T_h^R$. We construct a finite element function space $V_h^R$ on the triangulation $T_h^R$ with $\dim(V_h^R) = N_R$. We then generate an interpola-
tion operator $P : V_h \to V_h^R$ and interpolate every POD mode:

$$\varphi_i \big|_R = P \varphi_i \quad i = 1, \ldots, R$$

into $V_h^R$, resulting in a new RAPOD approximation of the solution $u \big|_R \in V_R \big|_R \subset V_h$ spanned by the RAPOD basis $\Phi_R = \{ \varphi_i \big|_R \}_{i=1}^R$ with $\varphi_i \big|_R \in V_h^R$ given by:

$$u \big|_R = \sum_{i=1}^R \varphi_i \big|_R(x) \times u_i \big|_R = \Phi_R \mathbf{u} \big|_R \quad (17)$$

Note that in the above and in what follows we use notation $\cdot |_R$ is used to denote an interpolation, rather than a restriction operation as is more common.

With this procedure complete, the online phase follows the standard Galerkin-POD procedure as in §5.1.3 but using the new POD basis functions $\Phi_i |_R$ interpolated in the finite element space $V_h^R$. Furthermore, the finite element operators e.g. $\hat{J}$ and $\hat{f}$ for the Newton-Raphson scheme are now assembled on the finite element space $V_h^R$ with $\dim(V_h^R) \ll \dim(V_h)$:

$$\hat{J} = \Phi_R^T J \Phi_R, \quad \hat{f} = \Phi_R^T f \quad (18)$$

5. Numerical examples

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

In this subsection, using the RAPOD method described in section 4, we solve the time-dependent non-linear Fisher, Kolmogorov, Petrovsky and Piscounov (FKPP) problem on a square domain $\Omega$ with homogeneous Dirichlet boundary conditions on $\partial \Omega$. The full specification of the problem in strong form is:

Find $u(x,t) \in H_0^1(\Omega)$ such that

$$\frac{\partial}{\partial t} u - \Delta u = c \cdot u \cdot (1 - u) \quad \text{in } \Omega = [0,3]^2 \times (0,T],$$

$$u = 0 \quad \text{on } \partial \Omega \times (0,T],$$

$$u(x,0) = u_0(x) \quad \text{in } \Omega,$$ 

where $\Delta$ is the Laplacian operator, $\frac{\partial}{\partial t}$ 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:

$$u_0(x) = \exp \left( -\frac{(x-x_0)^2}{\sigma^2} \right) \quad (20)$$

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:

$$M_N = \left\{ \frac{1}{2}, \frac{9}{10}, \ldots, 1 \right\}^2 \times \left\{ 0, \frac{T}{10}, \frac{2T}{10}, \ldots, \frac{9T}{10}, T \right\}, \quad |M_N| = 396. \quad (21)$$
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 (19) can be written as:

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

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

(22)

5.1.1. Finite difference discretisation in time

We first discretise (22) 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:

$$ (\partial_t u,v) \approx (k^{-1}(u^{n+1} - u^n),v) = A(u^{n+1},v) + B(u^n,v) + \frac{1}{2}C(u^n,v) + \frac{1}{2}C(u^{n+1},v) $$  

(23a)

with the following linear and semi-linear forms:

$$ A(u,v) = -(\nabla u, \nabla v), $$  

(23b)

$$ B(u,v) = -(cu^2,v), $$  

(23c)

$$ C(u,v) = (cu,v). $$  

(23d)

In the numerical results presented in we take the final time $T = 0.1$ and $k^{-1} = 1000$.

Rearranging (23) 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:

$$ \forall v \in \hat{V}, \quad \left( \left\{ k^{-1} - \frac{c}{2} \right\} u^{n+1},v \right) + (\nabla u^{n+1}, \nabla v) = \left( \left\{ k^{-1} + \frac{c}{2} - cu^n \right\} u^n,v \right) . $$  

(24)

Note that the only term that is a function of $u^n$ is on the right-hand side of (24).

5.1.2. 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 $V_h \subset V := H^1_0(\Omega)$ 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:

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

(25)

Following standard arguments, the discrete governing equations of problem (19) can then be written using the finite element basis functions as the following sequence of linear systems; for $n = 0,\ldots,M$ find $u^{n+1} \in \mathbb{R}^N$:

$$ \left( \left\{ k^{-1} - \frac{c}{2} \right\} M + K \right) u^{n+1} = \left( k^{-1} + \frac{c}{2} \right) Mu^n - b(u^n) $$  

(26)
where $\mathbf{M}$ is the usual finite element mass matrix, $\mathbf{K}$ is the stiffness matrix associated with the Laplacian operator $\Delta$ and the vector $\mathbf{b}(u)$ represents the non-linear term, with entries:

$$\forall i,j = \{1,2,\ldots,N\}, \quad M_{ij} = (\phi_j, \phi_i), \quad K_{ij} = (\nabla \phi_j, \nabla \phi_i), \quad b_{ij} = (cu_i^2, \phi_i).$$

(27)

5.1.3. Galerkin-POD discretisation

We now turn to the construction of reduced-order model of (19) 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 $\{\varphi_i\}_{i=1}^R$ with $\varphi_i \in V_h$ and $R = \text{dim}(V_R)$:

$$u_R(x) = \sum_{i=1}^R \varphi_i(x) u_{Ri} = \Phi u_R.$$  

(28)

where $\Phi = [\varphi_1, \ldots, \varphi_R] \in \mathbb{R}^{N \times R}$ is the matrix of the POD basis functions discretized on the finite element space $V_h$ with $\text{dim}(V_h) = N$.

We can then write a new set of discrete governing equations of (19), but instead of employing the finite element space $V_h$ as in (26), we instead use the POD space $V_R$; for $n = 0, \ldots, M$ find $u^{n+1} \in \mathbb{R}^R$ such that:

$$((k^{-1} - \frac{\xi}{2})\tilde{M} + \tilde{K})u_R^{n+1} = (k^{-1} + \frac{\xi}{2})\tilde{M}u_R^n - \tilde{b}(u^n)$$

(29)

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

$$\tilde{M} = \Phi^T M \Phi, \quad \tilde{K} = \Phi^T K \Phi,$$

(30)

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

$$\tilde{b}(u) = \Phi^T b(u).$$

Note that the terms in (29) involving $\tilde{M}$ and $\tilde{K}$ are not a function of the solution $u$ and can therefore be assembled once at the beginning of the online phase (29). The the overall assembly cost is amortised across the online phase. In contrast, the non-linear term $\tilde{b}(u) \in \mathbb{R}^R$ is a function of the current solution $u^n$, and its computation involves the assembly of the term $b(u)$ on the finite element space and its projection into the POD space. As we do not know a priori how the solution will evolve, clearly we must perform this operation at every timestep of the semi-implicit scheme.

5.1.4. Reduced assembly Galerkin-POD discretisation

It is the non-linear term $\tilde{b}(u)$ to which it is most important to apply the proposed reduced assembly method. With this procedure complete, we will simply follow the standard Galerkin-POD as in section 5.1.3 but using the new
POD basis functions $\varphi_i|_R$ interpolated in the finite element space $V^R_h$: for $n = 0, \ldots, M$ find $u^{n+1}|_R \in \mathbb{R}^R$ such that:

$$((k^{-1} - \frac{\varepsilon}{2})\hat{M} + \hat{K})u^{n+1}|_R = ((k^{-1} + \frac{\varepsilon}{2})\hat{M}u^n|_R - \hat{b}(u^n|_R))$$

(31)

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

$$\hat{M} = \Phi^T|_R M\Phi|_R, \quad \hat{K} = \Phi^T|_R K\Phi|_R,$$

(32)

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

$$\tilde{b}(u) = \Phi^T|_R b(u).$$

(33)

5.1.5. Numerical results

In this section we solve the Fisher, Kolmogorov, Petrovsky and Piscounov (FKPP) problem (19) with the POD (29) and the proposed RAPOD (31) methods and compare the results to the standard FEM.

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 a slightly customised version of 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 [21].

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 triangulation with $256 \times 256$ divisions resulting in 262144 cells. This leads to a finite element space $\dim(V_h) = 131585$. The resulting linear systems at one hundred timesteps (26) are solved using an algebraic multigrid preconditioned conjugate gradient method in PETSc. The preconditioner is re-used between timesteps, as the matrix operator on the left-hand side of (26) 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 (6) 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 empirical covariance matrix $C$. We compute this matrix and its eigenvalue decomposition in the $l^2$ inner-product space before re-orthonormalizing the eigenvectors using the classical Gram-Schmidt algorithm.
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.

in the discrete \( L^2 \) inner-product space induced by the finite element basis, i.e. the \( M \) inner-product.

The first part of the spectrum of the POD eigenvalue problem (6) is shown in fig. 1. We retain \( R = 17 \) modes using the a 99.9\% total energy cutoff criteria. Of course, we do not 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 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 triangulation \( \mathcal{T}^{17}_h \). Beginning with an initial triangulation \( \mathcal{T}^0_h \) covering the original domain \( \Omega \) with a triangular right-aligned triangulation with \( 2 \times 2 \) 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 triangulations \( \mathcal{T}^i_h \) 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 triangulation relatively unrefined.

The evolution of the number of cells in triangulations \( \{ \mathcal{T}^1, \ldots, \mathcal{T}^{17} \} \) is shown in fig. 3. The final triangulation \( \mathcal{T}^{17} \) is shown in fig. 4.

In fig. 6 we show the tradeoff between error and wall time for the RAPOD method for different tolerances with respect to the POD solution. The red dashed line shows the error in the 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 (\( \text{tol} = 10^{-5} \) gives \( ||e||_{H^1} = 4.38\% \)). The blue dashed line shows the wall
Figure 2: Left column: original unscaled POD modes $\sqrt{\sigma_i} \varphi_i$ represented on the finite element function space $V_h$ for $i = \{1, 2, 3, 4\}$. Right column: interpolated unscaled POD modes $\sqrt{\sigma_i} \varphi_i|_R$ and corresponding reduced assembly triangulations $T^i$ obtained using the RAPOD algorithm with $\text{tol} = 10^{-2}$ for $i = \{1, 2, 3, 4\}$. Note that while these intermediate triangulations form stages of the triangulation refinement algorithm, we only use triangulation $T^{17}$ in the online stage for all operations, see fig. [3]
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 $\text{tol} = 10^{-3}$ we get around a factor of five 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. The trade-off in terms of error is small.

Figs. 7 (linear scale) and 8 (log scale) show a breakdown of wall time in key stages for each method. The runtime of the FEM is dominated by assembly and linear algebra solves. Moving to POD, linear algebra solves become almost negligible, and consequently assembly becomes the dominant cost. RAPOD dramatically cuts the cost of assembly due to the reduced number of cells in the triangulation.

In fig. 9 we show the potential of parallelising the assembly operations in accelerating the RAPOD (or POD) method even further. We use 4 MPI processes and partition the triangulation equally between each process. We perform finite element assembly and the Galerkin projection onto the POD basis in parallel. The small dense Cholesky solve is performed on every MPI process simultaneously. We are currently achieving a parallel speed up of around 3.1 times over running with one MPI process. Detailed timings are shown in fig. 10. Producing a truly scalable reduced-order solver that can run on high-performance computers is a topic of current work.

In summary, in this section we have shown that the proposed RAPOD approach can greatly reduce the runtime of the online phase of a Galerkin-POD type reduced order model applied to a non-linear time-dependent diffusion-
Figure 4: Top: original unscaled POD mode $\sqrt{\sigma_{17}} \varphi_{17}$ Bottom: interpolated unscaled POD mode $\sqrt{\sigma_{17}} \varphi_{17}/R$ and corresponding reduced assembly triangulation $T^{17}$ obtained using the RAPOD algorithm with $\text{tol} = 10^{-2}$. 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 triangulation that is used to generate the RAPOD results with $\text{tol} = 10^{-2}$ in figs. 5, 6, 7 and 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 $l^1$ errors of FKPP problem using the three methods.
As the integration tolerance of the RAPOD algorithm is decreased (leading to triangulation refinement) we can see that the RAPOD error converges to the POD error.

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.
reaction equation. Speed-ups of around 5 times are possible, with only a small increase in error (5%) with respect to the standard Galerkin-POD method. Furthermore we have shown numerically that the RAPOD method recovers the standard Galerkin-POD method in the limit of triangulation refinement.

5.2. 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 [38]. We repeat the essential details here. Consider a three-dimensional body $B$ that can be modelled as a continuum. Let $\chi_0 : B \to \mathbb{R}^3$ be the known reference configuration and $\chi : B \to \mathbb{R}^3$ be the unknown deformed configuration after some external loads have been applied. The domain occupied by the undeformed configuration is then denoted $\Omega_0 := \chi_0(B) = [0, 1]^3$ and in the deformed configuration $\Omega := \chi(B)$. Then, for every point $p$ in the continuum body $B$, are related to points $X \in \Omega_0$ in the underformed configuration and $x \in \Omega$ in the deformed configuration through the maps $\chi_0$ and $\chi$ respectively, we can construct a sufficiently smooth deformation map $\psi : \Omega_0 \ni X \mapsto x \in \Omega$:

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

(34)
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 achieve 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.
from which we can define the deformation gradient tensor as:

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

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:

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

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

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

where $I_C := \text{tr}(C)$ 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 $\mu$ and $\lambda$ are material constants related to the shearing and volumetric behaviour of the material. We set the Young’s modulus $E = 10$ Pa and Poisson’s ratio $\nu = 0.3$ which can be related to $\mu$ and $\lambda$ through:

$$ \mu = \frac{E}{2(1 + \nu)} \quad (38a) $$

$$ \lambda = \frac{E \nu}{(1 + \nu)(1 - 2\nu)} \quad (38b) $$

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

$$ u^* = \text{arg min}_{u \in \mathcal{V}} \left\{ \int_{\Omega_0} W dx_0 - \int_{\partial_N \Omega_0} t \cdot u \ ds_0 \right\} $$

$$ = \text{arg min}_{u \in \mathcal{V}} \mathcal{E}(u) $$

where $\mathcal{V}$ is a sufficiently regular Hilbert space that satisfies the Dirichlet boundary condition $u = \{0,0,0\}^T$ on the bottom surface of the cube:

$$ \partial_D \Omega_0 := \{(x,y,0) \times (x,y,0)\} \subset \partial_N \Omega_0, $$

$t \in [L^2(\partial_N \Omega_0)]^3$ are surface tractions (Neumann boundary conditions) applied on the top surface of the cube $\partial_N \Omega_0 := \{(x,y,1) \times (x,y,1) \subset \partial_N \Omega_0\}$ and $dx_0$ and $ds_0$ are measures on the undeformed domain $\Omega_0$ and its boundary $\partial \Omega_0$, respectively. The traction vector $t$ on $\partial_N \Omega_0$ is set to be:

$$ t(X) := \left\{ \begin{array}{c} 0, 0, -6s \exp \left( -\frac{(X - X_0)^2}{\sigma^2} \right) \end{array} \right\}^T $$

$$ \sigma > 0, X_0 \in \mathbb{R}^n $$
with \( \sigma = 0.15 \) and \( x_0 \in \partial_N \Omega_0 \). We let the load position \( X_0 \) and load magnitude \( s \) form our parameters space \( M \), and choose the discrete training set \( M_N \) for the snapshot generation in the POD process as:

\[
M_{X_0} = \{0.3, 0.4, 0.5, 0.6\}^2 \in \partial_N \Omega_0
\]

\[
M_s = \{0, 0.1, 0.2, \ldots, 0.9, 1.0\}
\]

\[
M_N = M_{x_0} \times M_s, \quad S = |M_N| = 176
\]

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

\[
\forall \tilde{u} \in \mathcal{V}, \quad D_{\tilde{u}} \mathcal{E}(u^*) = F(u^*; \tilde{u}) = 0,
\]

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 interpreted as the equilibrium 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: Find \( u^* \in \mathcal{V} \) such that

\[
\forall \tilde{u} \in \mathcal{V}, \quad \int_{\Omega_0} S(u^*) : D_{\tilde{u}} E(u^*) \, dx_0 - \int_{\partial_N \Omega_0} t \cdot \tilde{u} \, ds = 0,
\]

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

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

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 take about an arbitrary point in solution space \( u_- \):

\[
J(u_-; \tilde{u}; \hat{u}) := D_{\tilde{u}} F(u_-; \hat{u}) = \int_{\Omega_0} \frac{\partial S}{\partial C} : D_{\tilde{u}} E(u_-) \, dx_0 \]

\[
+ \int_{\Omega_0} S : D_{\tilde{u}} [D_{\tilde{u}} E(u_-)] \, dx_0
\]

with:

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

and the notation \( \hat{u} \) is again 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:

\[
J(u^k; \delta \tilde{u}; \hat{u}) = -F(u^k; \tilde{u})
\]

\[
u^{k+1} = u^k + \delta \tilde{u}
\]
5.2.1. 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 $V_h \subset \tilde{V} = V$ spanned with vector piecewise linear polynomial finite element basis functions $\{\phi_j\}_{j=1}^N$ with $N = \dim(V_h)/3$ giving:

$$ u_h(x) = \sum_{i=1}^N \phi_i(x) u_i = \phi u, \quad (52) $$

resulting in a discrete Newton step:

$$ J(u_k) \delta u = -f(u^k), \quad (53) $$

$$ u^{k+1} = u^k + \delta u. \quad (54) $$

We use continuation in the loading parameter $s$ using the set $M_s$ to ensure the convergence of the Newton algorithm. The FKPP system we reduced in section 5.1 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 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 $u^k$.

5.2.2. Galerkin-POD discretisation

The Galerkin-POD procedure is performed identically to the procedure described in section 3. This results in the following expressions for the finite element Jacobian and residual projected into the POD space:

$$ \tilde{J} = \Phi^T J \Phi, \quad \tilde{f} = \Phi^T f \quad (55) $$

Again, we must perform the assembly of the matrix $J$ and vector $f$ in the finite element space $V_h$ at each Newton step.

5.2.3. Reduced assembly Galerkin-POD discretisation

We take the pointwise magnitude of the POD modes $\{\varphi_i\}_{i=1}^R$:

$$ ||\varphi_i(x)|| = (\varphi_i^x(x)^2 + \varphi_i^y(x)^2 + \varphi_i^z(x)^2)^{1/2} \quad (56) $$

before applying the RAPOD algorithm to $||\varphi_i(x)||$ and obtaining triangulation $T_h^R$.

As before, we then reconstruct the Galerkin-POD problem on the triangulation associated with the $R$-th mode $T_h^R$. We construct a discrete interpolation operator $P : V_h \rightarrow V_h^R$ and interpolate every POD mode:

$$ \varphi_i|_R = P \varphi_i \quad i = 1, \ldots, R \quad (57) $$

into $V_h^R$.

With this procedure complete, we will simply follow the standard Galerkin-POD as in section 5.1.3 but using the new POD basis functions $\varphi|_R$ interpolated in the finite element space $V_h^R$:

$$ \tilde{J} = \Phi_{|R}^T J \Phi_{|R}, \quad \tilde{f} = \Phi_{|R}^T f \quad (58) $$
5.2.4. Numerical results

In this section we solve the hyperelasticity problem with the POD and the proposed RAPOD method 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 $M_N$.

The particulars of the standard FEM solver are given as follows. The 176 snapshots in the set $M_N$ and the reference solution for $x_0$ are computed on a uniformly refined tetrahedral right-aligned triangulation with $32 \times 32 \times 32$ divisions resulting in 196608 cells. This leads to a finite element space $\dim(V_h) = 107811$. The resulting linear systems at each Newton iteration are solved using an algebraic multigrid preconditioned GMRES method in PETSc. The complete set of snapshots takes around 3 hours to compute using 4 MPI processes and consumes 2.3GB space in an HDF5 file.

Once the snapshot set $M_N$ has been computed we solve the POD eigenvalue problem as for the FKPP problem sec. 5.1. We retain $R = 14$ modes using a 99.99% total energy cutoff criteria.

We can now apply the RAPOD algorithm to construct the reduced triangulation $T^{14}$. Beginning with an initial triangulation $T^0$ covering the original domain $\Omega$ with a tetrahedral right-aligned triangulation with $2 \times 2 \times 2$ divisions resulting in 48 cells. Fig. 12 shows a cut through of the triangulation $T^{14}$ created with the RAPOD algorithm with a tolerance $tol = 10^{-2}$.

Figs. 13, 14 and 15 show the solution of the hyperelasticity problem using...
FEM, POD and RAPOD methods, respectively. The solution using POD retains the key features of the full FEM solution, including the global deformation of the block and the shape of the indentation caused by the external traction. The RAPOD method, in turn, keeps all the main features of the solution obtained with the POD method.

The RAPOD method can achieve this at significantly reduced computational cost with respect to the standard POD method. In fig. 16 the red dashed line shows the error of the 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 = 0.002 gives $\|e\|_{L^2} = 3.0\%$). The blue dashed line shows the wall time for the standard POD simulation (26 seconds). As the RAPOD tolerance is decreased, the wall time increases due to the greater work associated with assembling the larger finite element matrices at each Newton iteration. With RAPOD tol = $10^{-2}$ we obtain a speed-up of approximately 7 versus the standard POD method, with only a small increase in error $\|e\|_{L^2} = 5\%$. The RAPOD method gives a speed-up over the standard POD method for all tolerances.

A breakdown of the amount of time spent in key computational areas for the FEM, POD and RAPOD methods (for different tolerances) is shown in figs. 17 (linear scale) and 18 (log scale). The FEM solution time is dominated by linear algebra operations, mainly linear solves, with a significant proportion of time also spent assembling Jacobians and residuals. In contrast, for the standard
Figure 13: Results of Hyperelasticity problem with FEM method. Cut through of deformed domain $\Omega$ coloured with magnitude of displacements $u^*$. Grey transparent box is initial configuration $\Omega_0$. 
Figure 14: Results of Hyperelasticity problem with POD method with snapshot set $M_n$ and $R = 14$.

Figure 15: Results of Hyperelasticity problem with RAPOD method with snapshot set $M_n$, $R = 14$ and $\text{tol} = 10^{-2}$.
POD method, linear solves become almost free and the total computational time is dominated by assembly. RAPOD alleviates this bottleneck creating new reduced-order models with significantly lower runtimes.

6. Conclusion

In this paper we have presented a new hyper-reduction method called Reduced Assembly (RA) to cuts assembly costs in the online phase of non-linear reduced order models. We have applied RA to basis functions created using the Galerkin-POD procedure (RAPOD). We have demonstrated that the RAPOD method can provide speed-ups of up to 5 times over standard POD, with minimal error committed over the baseline POD method. We are currently investigating improvements to the cell error indicator performance and anisotropic triangulation refinement strategies to produce triangulations with even fewer cells.

Supplementary material

Full code to produce all of the examples and figures in this paper is available at [20].

Acknowledgements

We thank the financial support of the European Research Council Starting Independent Research Grant (ERC Stg grant agreement No. 279578) entitled ‘Towards real time multiscale simulation of cutting in non-linear materials with applications to surgical simulation and computer guided surgery.’ Jack S. Hale is supported by the National Research Fund, Luxembourg, and cofunded under
Figure 17: Breakdown of computational time spent in key stages of the total wall time for the hyperelasticity problem: projections (POD and RAPOD), assembly of Jacobians and residuals, linear algebra solves (preconditioned GMRES 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 18: The same timing data is shown on a linear scale in fig. 17. With this scaling we can more clearly see the dominance of assembly over solution time for the RAPOD and POD methods.
the Marie Curie Actions of the European Commission (FP7-COFUND) Grant No. 6693582. The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [37] and at Cardiff University.

Bibliography

References


