Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 380 (2021) 113723
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A hyper-reduction method using adaptivity to cut the assembly costs of reduced order models

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

[a] *Institute of Computational Engineering, University of Luxembourg, Luxembourg*
[b] *School of Engineering, Cardiff University, United Kingdom*
[c] *Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen University, Germany*

Received 20 August 2018; received in revised form 27 January 2021; accepted 3 February 2021
Available online xxxx

Abstract

At every iteration or timestep of the online phase of some reduced-order modelling schemes for non-linear or time-dependent systems, 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 become 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 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; Model order reduction; Non-linear PDEs

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 and parameter-dependent partial differential equation (PDE). This PDE cannot usually be solved using analytical approaches and we must resort to numerical methods. The original non-linear infinite-dimensional problem can be transformed via a suitable discretisation into a sequence of discrete linear algebra problems of finite dimension.

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:

* Corresponding author at: Institute of Computational Engineering, University of Luxembourg, Luxembourg.
E-mail address: stephane.bordas@alum.northwestern.edu (S.P.A. Bordas).

https://doi.org/10.1016/j.cma.2021.113723

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 are 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 the cost of assembly. An obvious way to reduce computational cost is to reduce the size of the linear problems 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 triangulation (mesh) adaptivity, see e.g. [1]. 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 [2], accelerating parametric studies and stochastic simulation [3].

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 computed, the information within them is in some way compressed into a more efficient representation. In the reduced basis (RB) approach, e.g. [4–7], a linear combination of the snapshots selected by a greedy procedure is used as the global basis. In the Galerkin proper orthogonal decomposition (Galerkin-POD) approach e.g. [8–14] 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. [2,15] the solution is expressed in a separated basis 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-dependent 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 are 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 and projection.

A large class of techniques commonly referred to as hyper-reduction techniques have been created to reduce the burden of assembly and projection. 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) [16,17] or discrete empirical interpolation (DEIM) [18,19], can be used to approximate parametric or non-linear functions by separable interpolation functions. Other approaches include missing point estimation [20], Empirical Cubature Method [21], Petrov–Galerkin reduced integration [22], Gappy-POD [23], Gauss–Newton Approximated Tensors (GNAT) [24], Empirical Quadrature (EQ) [25,26] and Solution-based Nonlinear Subspace (SNS) [27]. In recent work, [28,29] applied adaptive coarse mesh reduction to construct hyper-reduced flow models. Simply put, these methods work by identifying a subset of terms in the non-linear equations that are important. In the case of DEIM [30] 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 [31–37] 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 globally supported basis functions from a set of solution snapshots in parameter space. The key idea behind RA is that the POD 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 that 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.

A distinct feature of RA compared with most other hyper-reduction methods is that by using triangulation adaptation, a new finite element model is constructed on the adapted triangulation based on the important features detected in the solution snapshots via the POD procedure. We call this standard finite element model on an adapted triangulation RAFEM and is a useable numerical discretisation itself. In contrast, most hyper-reduction methods work by sampling a subset of the degrees of freedom (e.g. DEIM's magic points [19]) from the original full-order model. Our approach has the advantage that the new full-order model directly inherits the underlying numerical properties (e.g. ellipticity/coercivity, full-rank) of the original full-order model used to generate the snapshot set, as long as the adapted triangulation is of sufficiently good quality (e.g. no heavily distorted cells). The integration point locations and weights come naturally from the definition of the triangulation as in the finite element method. Because of this, the online stage of the RAPOD procedure can use entirely standard finite element assembly routines to construct the full-order model, making it relatively straightforward to implement. In contrast, for example, an efficient implementation of DEIM requires a custom assembler to be written that loop over only the degrees of freedom associated with the magic points. A disadvantage of our approach is that triangulation adaptivity brings its own complications, particularly with respect to domains with complex or concave curved boundaries or POD basis functions with highly anisotropic behaviour. As already mentioned, the adapted mesh must also be of good quality according to the usual metrics, and this is not always possible. These issues are the subject of ongoing research.

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 parameterised non-linear and/or time-dependent partial differential equations. Throughout we use lower case italics e.g. u to refer to continuous functions and spaces, and bold-faced italics e.g. $\mathbf{u}$ to refer to discrete operators like matrices and vectors. Consider a spatial domain $\Omega \subset \mathbb{R}^d$ with $x \in \Omega$. For a fixed parameter $\bar{m}$ in some admissible set M, and for each $t \in (0, T]$ find a $u(x, t) \in V$ such that

$$\forall\, \tilde{u} \in \hat{V}. \quad F(u, t, m = \bar{m}; \tilde{u}) = 0, \tag{1}$$

where $F : V \times (0, T] \times M \times \hat{V} \to \mathbb{R}$ is a semi-linear form, that is, linear in the arguments (here, the test function $\tilde{u}$) following the semicolon. As the parameter m is fixed at $\bar{m}$ here, we drop the explicit dependence on m in what follows.

The proposed RAPOD method is applicable to the following two broad classes of discrete solution methods that can be used to solve problems of the type (1).

2.1. Non-linear stationary problems solved with Newton–Raphson method

In the non-linear stationary case, we can exclude the dependency on time and simply write the problem as: Find $u(x) \in V$ such that

$$\forall\, \tilde{u} \in \hat{V}, \quad F(u; \tilde{u}) = 0, \tag{2}$$

It is well known that problems of this type can be solved with the Newton–Raphson method. We apply the Newton–Raphson method at the continuous level[1] before discretising in space using the finite element method. Given an initial approximation $u^0 \in V$ to (2) we seek an improved approximation $u^{K+1} \in V$ through a sequence of K steps

$$u^{k+1} = u^k + \delta u^k, \quad k = 0, \dots, K. \tag{3}$$

The increments δu^k can be found by solving the following update equation for each k

$$\forall \tilde{u} \in \hat{V} \quad J(u^k; \delta u^k, \tilde{u}) = -F(u^k; \tilde{u}), \tag{4}$$

where $J(u^k; \delta u^k, \tilde{u})$ is the Jacobian, which can be obtained by taking the Gateaux derivative of the residual at a point $u \in V$ in a direction $\delta u \in V$

$$J(u; \delta u, \tilde{u}) := D_u[F(u; \tilde{u})][\delta u]. \tag{5}$$

Typically K is chosen after a certain stopping criterion is reached, e.g. $\|F(u^{K+1}; \tilde{u})\| < \epsilon$.

We then discretise the solution u in space using e.g. a Galerkin H^1-conforming finite element method. We introduce a triangulation[2] $\mathcal{T}_h^0$ of the domain Ω consisting of N_c^0 simplicial cells (e.g. triangles or tetrahedrons) and then build discrete trial and test spaces $V_h \subset V$ and $\hat{V}_h \equiv V_h$ spanned with piecewise linear polynomial finite element basis functions $\{\phi_i\}_{i=1}^N$ with $N = \dim(V_h)$

$$u_h(x) = \sum_{i=1}^{N} \phi_i(x) u_i, \tag{6}$$

$$\boldsymbol{u} = \{u_1, u_2, \dots, u_N\}^T \in \mathbb{R}^N. \tag{7}$$

where the u_i are unknown coefficients that can be found using the standard Galerkin procedure, and $\boldsymbol{u}$ are the coefficients u_i arranged in a vector.

Following standard arguments, by substituting the basis (6) into (3) and (4) we obtain the following fully discrete Newton–Raphson method: Given an initial approximation $\boldsymbol{u}^0 \in \mathbb{R}^N$, find $\boldsymbol{\delta u}^{K+1} \in \mathbb{R}^N$ through a sequence of K steps

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{\delta u}^k, \tag{8a}$$

$$\boldsymbol{J}(\boldsymbol{u}^k)\boldsymbol{\delta u}^k = -\boldsymbol{f}(\boldsymbol{u}^k), \tag{8b}$$

where $\boldsymbol{J}(\boldsymbol{u}^k) \in \mathbb{R}^{N \times N}$ is the matrix arising from the finite element discretisation of the Jacobian $J(u^k; \delta u^k, \tilde{u})$, and $\boldsymbol{f}(\boldsymbol{u}^k) \in \mathbb{R}^N$ is the vector arising from the finite element discretisation of the residual $F(u^k; \tilde{u})$. We explicitly remark on the dependence of $\boldsymbol{J}(u^k)$ and $\boldsymbol{f}(u^k)$ on the current solution $\boldsymbol{u}^k$, i.e. they must typically be re-assembled at every Newton step.

2.2. Non-linear time-dependent problems solved with semi-implicit time integration

In the time-dependent case, after first discretising in time $u^n \approx u(t^n)$ at $L+1$ times $t^0 < t^1 < \cdots < t^L \in [0, T]$ using a semi-implicit time-stepping scheme, and then in space using finite elements $u(t^n) \approx u_h(t^n) \in V_h$, we end up with a sequence of discrete linear problems to solve: for $n = 0, \dots, L-1$ and initial condition $\boldsymbol{u}^0 \in \mathbb{R}^N$, find $\boldsymbol{u}^n \in \mathbb{R}^N$ such that

$$\boldsymbol{K}\boldsymbol{u}^{n+1} = \boldsymbol{b}(\boldsymbol{u}^n), \tag{9}$$

[1] Performing differentiation at the continuous level before discretising is consistent with the approach taken in the FEniCS Project [38], which we use to generate the numerical results in this paper.

[2] It should be possible to apply the proposed method to other discretisations of the domain that support adaptivity, e.g. quadtree/octree meshes [39,40].

where $\boldsymbol{K}$ and $\boldsymbol{b}(\boldsymbol{u}^n)$ are matrices and vectors arising from the finite element discretisation of the semi-discrete solution in time. Again, we explicitly remark on the dependence of $\boldsymbol{b}$ on the current (known) solution $\boldsymbol{u}^n$, i.e. it 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, some of the matrix and/or vector operators in the linear problems are dependent on the current and/or previous solutions and must be assembled at every step of the Newton–Raphson or timestepping algorithm. As discussed in the introduction, it is this assembly cost that becomes dominant in the context of many reduced-order modelling schemes applied to time-dependent or non-linear problems.

3. Proper orthogonal decomposition

Let us briefly recall the main steps to construct a proper orthogonal decomposition (POD) basis via the method of snapshots. More details can be found in [10,12]. 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.

We generate snapshots by repeatedly solving (1) at S discrete points in time and parameter space $s = \left\{(t^1, m^1), (t^2, m^2), \ldots, (t^S, m^S)\right\}$. Each pair of points in s is then associated with a solution snapshot u_i for $i = 1, \ldots, S$ generated by solving (1) with fixed m and t. With the set of coefficients of the discrete solution snapshots $\boldsymbol{U} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_S\} \in \mathbb{R}^{N\times S}$ computed, we seek a O-dimensional subspace $V_R \subset V_h$ that optimally represents the data contained in the snapshots $\boldsymbol{U}$ with the smallest possible O, i.e. we seek a projection operator[3] $\Pi_O : V_h \to V_R$ that minimises the following least-squares distance

$$\sum_{i=1}^{O} \|u_i - \Pi_O u_i\|_{V_h}^2. \tag{10}$$

It is well known that the optimal basis in the above least-squares sense $\boldsymbol{\Phi} = \left\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_O\right\} \in \mathbb{R}^{N\times O}$ spanning V_R can be found by solving the following eigenvalue problem

$$\boldsymbol{C}\boldsymbol{\psi}_i = \lambda_i \boldsymbol{\psi}_i, \quad \boldsymbol{\varphi_i} = \frac{1}{\sqrt{\lambda_i}}\boldsymbol{U}\boldsymbol{\psi}_i, \quad \|\boldsymbol{\psi}_i\|_{V_h} = 1, \quad i = 1, \ldots, S, \tag{11}$$

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

$$u \approx u_R = \sum_{i=1}^{O} \varphi_i(x) u_{Ri}, \tag{12}$$

$$\boldsymbol{u}_R = \{u_{R1}, u_{R2}, \ldots, u_{RO}\} \in \mathbb{R}^O. \tag{13}$$

An additional practicality is that we do not work with the ansatz (12) 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 (8)

$$\widetilde{\boldsymbol{J}} = \boldsymbol{\Phi}^T \boldsymbol{J}\, \boldsymbol{\Phi} \in \mathbb{R}^{O\times O}, \quad \widetilde{\boldsymbol{f}} = \boldsymbol{\Phi}^T \boldsymbol{f} \in \mathbb{R}^O, \tag{14}$$

or in a timestepping scheme (9)

$$\widetilde{\boldsymbol{K}} = \boldsymbol{\Phi}^T \boldsymbol{K}\, \boldsymbol{\Phi} \in \mathbb{R}^{O\times O}, \quad \widetilde{\boldsymbol{b}} = \boldsymbol{\Phi}^T \boldsymbol{b} \in \mathbb{R}^O. \tag{15}$$

[3] Given a set of basis functions $\left\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_O\right\}$ that spans the sub-space V_R, the projection operator Π_O, defined as $(\Pi_O(f), \boldsymbol{\varphi_i})_{V_h} = (f, \boldsymbol{\varphi_i})_{V_h}$, $\forall f \in V_h$, $i \in [1, O]$ is given as $\Pi_O(f) = \sum_{i=1}^{O} (f, \boldsymbol{\varphi}_i)_{V_h} \boldsymbol{\varphi}_i$.

4. The 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 (14) and (15) 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 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 [41], part of the FEniCS Project [38], but most finite element toolboxes could be used with little effort. Linear algebra operations are all performed using PETSc [42] and SLEPc [43].

We start with the original problem triangulation $\mathcal{T}_h^0$ consisting of N_c^0 simplicial quasi-uniform cells $\{T_1^0, \ldots, T_{N_c^0}^0\}$ that form an approximation to the problem domain Ω

$$\Omega \approx \mathcal{T}_h^0 = \cup_{i=1}^{N_c^0} T_i^0. \tag{16}$$

Using this triangulation and the standard methodology outlined in Section 3 we generated the solution snapshots $\boldsymbol{U}$ from which we can calculate O POD basis functions $\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_O\}$. We then create the coarsest possible triangulation $\mathcal{T}_h^1$ with $N_c^1 \ll N_c^0$ simplicial cells $\{T_1^1, \ldots, T_{N_c^1}^1\}$ that also covers the problem domain

$$\Omega \approx \mathcal{T}_h^1 = \cup_{i=1}^{N_c^1} T_i^1. \tag{17}$$

Then, starting with POD basis function φ_1, we begin the adaptation/refinement of triangulation $\mathcal{T}_h^1$. The POD basis function φ_1 has global support and is interpolated on the finite element function space $V_h(\mathcal{T}_h^0)$ that was used to generate the snapshots of the non-linear problem (1). Furthermore, it is the basis function associated with the leading eigenvalue of (11) and has the least oscillatory behaviour.

For every cell T_i^1 of the coarse triangulation $\mathcal{T}_h^1$ we calculate the following cell local error indicator that represents the local error in the integral of $|\varphi_1|$ over the support of T_i^1

$$\eta_i = \int_{T_i^1(Q_n)} |\varphi_1(x)| \, \mathrm{d}x - \int_{T_i^1(Q_{n+1})} |\varphi_1(x)| \, \mathrm{d}x, \tag{18}$$

where $T_i^1(Q_n)$ denotes that the corresponding integral is calculated using a quadrature rule of order n on the simplex T_i^1. The notation $|\cdot|$ means to take the absolute value of the argument. Note that in the numerical integration procedure the value of the function φ_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 $\mathcal{T}_h^0$. From a heuristic perspective, assuming that basis function φ_1 is sufficiently smooth, the cell error indicator η_h^i tells us of whether further refinement of the cell T_i could lead to an improved approximation to the integral of the basis function φ_1. The basis functions are used as indicators of areas of the domain in which important features in the solution are occurring. The use of Gauss quadrature rules of increased order to estimate integration error is well-established, e.g. hierarchical Gauss–Kronrod rules, as discussed in [44]. The use of the absolute function $|\cdot|$ is to ease the situation in which the integral of the oscillatory basis function (across the domain or even across the cell) is close to zero.

We calculate two further global quantities. The first, I_e, is the integral of the absolute value of the basis function φ_1 on the original triangulation $\mathcal{T}_h^0$ using a sufficiently high quadrature rule of order m (with respect to the polynomial order of the basis functions in V_h)

$$I_e = \sum_{i=1}^{N_c^0} \int_{T_i^0(Q_m)} |\varphi_1| \, \mathrm{d}x. \tag{19}$$

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 basis function calculated on the coarse triangulation $\mathcal{T}_h^1$

$$I_a = \sum_{i=1}^{N_c^1} \int_{T_i^1(Q_n)} |\varphi_1| \, \mathrm{d}x. \tag{20}$$

We then calculate the following global relative error

$$E = |I_e - I_a|/I_e. \tag{21}$$

If E is greater than some specified tolerance, e.g. 1%, we use the cell-based error indicators η_h^i to mark some cells in triangulation $\mathcal{T}_h^0$ for refinement following the strategy proposed by Dörfler [45]. 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 basis function φ_1 as many times as necessary to drop the error E below the specified tolerance.

Once E is sufficiently small, we have the specially adapted triangulation $\mathcal{T}_h^1$ for basis function φ_1. We then repeat the above process using the triangulation $\mathcal{T}_h^1$ as input for the adaptive refinement algorithm on basis function φ_2 to find $\mathcal{T}_h^2$, and so on, until we have performed the triangulation adaptation for all O basis functions. The output of the algorithm is a triangulation $\mathcal{T}_h^O$.

A more formal version of the procedure is detailed in Algorithm 1. We would encourage the reader to refer to Fig. 2 for a visual representation of the evolution of the RAPOD algorithm, and Fig. 4 for the final output.

Algorithm 1 Reduced assembly triangulation refinement algorithm.

Original triangulation $\mathcal{T}_h^0$ with N_0 simplicial cells.
Set of global basis functions, $\{\varphi_1, \ldots, \varphi_O\}$ defined on a function space V_h on the original triangulation $\mathcal{T}_h^0$.
Initial coarse triangulation $\mathcal{T}_h^1$ with N_c^1 simplicial cells with $N_c^1 \ll N_c^0$.
A tolerance on the integration error, tol.
for $i = 1 : O$ (loop over POD basis functions) **do**
  Calculate 'exact' integral $I_e = \sum_{j=1}^{N_c^0} \int_{T_j^0(Q_m)} |\varphi_i| \, \mathrm{d}x$.
  Let N_c^i be the number of cells in current triangulation $\mathcal{T}_h^i$.
  Calculate 'approximate' integral $I_a = \sum_{k=1}^{N_c^i} \int_{T_k^i(Q_n)} |\varphi_i| \, \mathrm{d}x$.
  Calculate $E = |I_e - I_a|/I_e$.
  while $E >$ tol **do**
    for $l = 1 : N_c^i$ (loop over cells in triangulation) **do**
      Calculate local error estimator
        $\eta_l = \int_{T_l^i(Q_n)} |\varphi_i(x)| \, \mathrm{d}x - \int_{T_l^i(Q_{n+1})} |\varphi_i(x)| \, \mathrm{d}x$.
    end for
    Based on η_l, refine triangulation $\mathcal{T}_h^i$ with Dörfler algorithm.
    Let N_c^i be the number of cells in the refined triangulation $\mathcal{T}_h^i$.
    Calculate improved 'approximate' integral on refined triangulation $I_a = \sum_{k=1}^{N_c^i} \int_{T_k^i(Q_n)} |\varphi_i| \, \mathrm{d}x$.
    Calculate $E = |I_e - I_a|/I_e$.
  end while
  Let $\mathcal{T}_h^{i+1}$ be $\mathcal{T}_h^i$.
end for
return $\mathcal{T}_h^O$.

We now reconstruct the Galerkin-POD problem on the triangulation $\mathcal{T}_h^O$ associated with the Oth basis function. We construct a finite element function space V_h^R on the triangulation $\mathcal{T}_h^O$ with $\dim(V_h^R) = N_R$. We then generate an interpolation operator $P : V_h \to V_h^R$ and project every POD basis function

$$\varphi_i|_R = P\varphi_i \quad i = 1, \ldots, O\,, \tag{22}$$

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

$$u|_R = \sum_{i=1}^{O} \varphi_i|_R(x) \times u_i|_R, \tag{23}$$

$$\boldsymbol{\Phi}|_R = \left\{\boldsymbol{\varphi}_1|_R, \boldsymbol{\varphi}_2|_R, \ldots, \boldsymbol{\varphi}_O|_R\right\} \in \mathbb{R}^{N_R \times O}. \tag{24}$$

Note that in the above and in what follows $\cdot|_R$ 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 $\boldsymbol{\Phi}|_R$ interpolated in the finite element space V_h^R. Furthermore, the finite element operators e.g. $\boldsymbol{J}|_R$ and $\boldsymbol{f}|_R$ for the Newton–Raphson scheme are now assembled on the finite element space V_h^R with $N_R = \dim(V_h^R) \ll \dim(V_h)$

$$\widehat{\boldsymbol{J}} = \boldsymbol{\Phi}|_R^T \boldsymbol{J}|_R \boldsymbol{\Phi}|_R, \ \widehat{\boldsymbol{f}} = \boldsymbol{\Phi}|_R^T \boldsymbol{f}|_R. \tag{25}$$

We emphasise again that the RAPOD procedure introduces a new full-order model on the finite element space V_h^R constructed according to the information contained in the POD basis functions. This is in contrast to other hyper-reduction approaches, e.g. DEIM, that use a modified version of the same full-order model that was used to generate the snapshots.

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 dimensionless form non-linear Fisher, Kolmogorov, Petrovsky and Piscounov (FKPP) problem on a square domain Ω with homogeneous Dirichlet boundary conditions on $\partial\Omega$. The full specification of the problem in strong form is

$$\begin{aligned} &\text{Find } u(x,t) \in H_0^1(\Omega) \text{ such that} \\ &\qquad \begin{aligned} \partial_t u - \Delta u &= cu(1-u) && \text{in } \Omega = [0,3]^2 \times (0,T], \\ u &= 0 && \text{on } \partial\Omega \times (0,T], \\ u(x,0) &= u_0(x) && \text{in } \Omega, \end{aligned} \end{aligned} \tag{26}$$

where Δ is the Laplace 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

$$u_0(x) = \exp\left(-\frac{(x-x_0)^2}{\sigma^2}\right), \tag{27}$$

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 $t \in (0,T]$ form our parameters, and choose the discrete training set for the snapshot generation process as

$$s = \left\{\tfrac{1}{2}, \tfrac{6}{10}, \ldots, 1\right\}^2 \times \left\{\tfrac{T}{10}, \tfrac{2T}{10}, \ldots, \tfrac{9T}{10}, T\right\}, \quad |s| = 360. \tag{28}$$

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 (26) can be written as

$$\begin{aligned} &\text{Find for each } t \in (0,T]\ u(x,t) \in V \text{ such that} \\ &\forall v \in \hat{V}, \quad F(u,c;v) := (\partial_t u, v) + (\nabla u, \nabla v) - (cu, v) + (cu^2, v) = 0. \end{aligned} \tag{29}$$

5.1.1. Finite difference discretisation in time

We first discretise (29) 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 $L+1$ times $t^0 < t^1 < \cdots < t^L$, and for simplicity we take the timestep $k \in \mathbb{R}$ as a constant giving $t^n = nk$ and final time $T = Lk$. We approximate the time derivative $\partial_t u$ with the following semi-implicit scheme

$$\begin{aligned} (\partial_t u, v) &\approx \left(k^{-1}\left(u^{n+1} - u^n\right), v\right) \\ &= A(u^{n+1}, v) + B(u^n; v) + \tfrac{1}{2}C(u^n, v) + \tfrac{1}{2}C(u^{n+1}, v) \end{aligned} \tag{30}$$

with the following linear and semi-linear forms

$$A(u, v) = -(\nabla u, \nabla v), \tag{31a}$$

$$B(u; v) = -(cu^2, v), \tag{31b}$$

$$C(u, v) = (cu, v). \tag{31c}$$

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

Rearranging (30) and substituting (31) gives the following sequence of problems to solve; for $n = 0, \ldots, L$ and $u^0 = u_0$, find $u^{n+1} \in V$ such that

$$\forall v \in \hat{V} \quad \left(\left\{k^{-1} - \tfrac{c}{2}\right\} u^{n+1}, v\right) + (\nabla u^{n+1}, \nabla v) = \left(\left\{k^{-1} + \tfrac{c}{2} - cu^n\right\} u^n, v\right). \tag{32}$$

Note that the only term that is a function of u^n is on the right-hand side of (32).

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 $V_h \subset V := H_0^1(\Omega)$ and $\hat{V}_h \equiv V_h$ spanned with piecewise linear polynomial finite element basis functions $\left\{\phi_j\right\}_{j=1}^N$ with $N = \dim(V_h)$ giving

$$u_h(x) = \sum_{i=1}^{N} \phi_i(x) u_i. \tag{33}$$

Following standard arguments, the discrete governing equations of problem (26) can then be written using the finite element basis functions as the following sequence of linear systems; for $n = 0, \ldots, L$ find $\boldsymbol{u}^{n+1} \in \mathbb{R}^N$

$$\left(\left\{k^{-1} - \tfrac{c}{2}\right\} \boldsymbol{M} + \boldsymbol{K}\right) \boldsymbol{u}^{n+1} = \left(k^{-1} + \tfrac{c}{2}\right) \boldsymbol{M}\boldsymbol{u}^n - \boldsymbol{b}(\boldsymbol{u}^n), \tag{34}$$

where $\boldsymbol{M}$ is the usual finite element mass matrix, $\boldsymbol{K}$ is the stiffness matrix associated with the Laplace operator Δ and the vector $\boldsymbol{b}(u)$ represents the non-linear term, with entries

$$\forall i, j = \{1, 2, \ldots, N\}, \quad M_{ij} = (\phi_j, \phi_i), \quad K_{ij} = (\nabla\phi_j, \nabla\phi_i), \quad b(u_h)_i = (cu_h^2, \phi_i). \tag{35}$$

5.1.3. Galerkin-POD discretisation

We now turn to the construction of reduced-order model of (26) 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 $\{\boldsymbol{\varphi_i}\}_{i=1}^O$ with $\varphi_i \in V_h$ and $O = \dim(V_R)$

$$u_R(x) = \sum_{i=1}^{O} \varphi_i(x) u_{Ri}, \tag{36}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_O] \in \mathbb{R}^{N \times O}. \tag{37}$$

We can then write a new set of discrete governing equations of (26), but instead of employing the finite element space V_h as in (34), we instead use the POD space V_R; for $n = 0, \ldots, L$ find $\boldsymbol{u}_R^{n+1} \in \mathbb{R}^O$ such that

$$((k^{-1} - \tfrac{c}{2})\widetilde{\boldsymbol{M}} + \widetilde{\boldsymbol{K}})\boldsymbol{u}_R^{n+1} = (k^{-1} + \tfrac{c}{2})\widetilde{\boldsymbol{M}}\boldsymbol{u}_R^n - \widetilde{\boldsymbol{b}}(\boldsymbol{u}^n), \tag{38}$$

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

$$\widetilde{\boldsymbol{M}} = \boldsymbol{\Phi}^T \boldsymbol{M} \boldsymbol{\Phi}, \quad \widetilde{\boldsymbol{K}} = \boldsymbol{\Phi}^T \boldsymbol{K} \boldsymbol{\Phi}, \tag{39}$$

and $\widetilde{\boldsymbol{b}}(u) \in \mathbb{R}^O$ represents the projection of $\boldsymbol{b}(u) \in \mathbb{R}^N$ into the POD space

$$\widetilde{\boldsymbol{b}}(\boldsymbol{u}) = \boldsymbol{\Phi}^T \boldsymbol{b}(\boldsymbol{u}). \tag{40}$$

Note that the terms in (38) involving $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{K}}$ are not a function of the solution u and can therefore be assembled once at the beginning of the online phase (38). The overall assembly cost is amortised across the online phase. In contrast, the non-linear term $\widetilde{\boldsymbol{b}}(\boldsymbol{u}) \in \mathbb{R}^O$ is a function of the current solution $\boldsymbol{u}^n$, and its computation involves the assembly of the term $\boldsymbol{b}(\boldsymbol{u})$ 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 $\widetilde{\boldsymbol{b}}(u)$ 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_h^R: for $n = 0, \dots, L$ find $\boldsymbol{u}^{n+1}|_R \in \mathbb{R}^O$ such that

$$((k^{-1} - \tfrac{c}{2})\widehat{\boldsymbol{M}} + \widehat{\boldsymbol{K}})\boldsymbol{u}^{n+1}|_R = (k^{-1} + \tfrac{c}{2})\widehat{\boldsymbol{M}}\boldsymbol{u}^{n}|_R - \widehat{\boldsymbol{b}}(\boldsymbol{u}^{n}|_R), \tag{41}$$

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

$$\widehat{\boldsymbol{M}} = \boldsymbol{\Phi}^T|_R\, \boldsymbol{M}|_R\, \boldsymbol{\Phi}|_R,\ \widehat{\boldsymbol{K}} = \boldsymbol{\Phi}^T|_R\, \boldsymbol{K}|_R \boldsymbol{\varphi}|_R, \tag{42}$$

and $\widehat{\boldsymbol{b}} \in \mathbb{R}^O$ is the projection of the vector $\boldsymbol{b}|_R \in \mathbb{R}^{N_R}$ assembled on V_h^R and projected onto the reduced assembly POD space

$$\widehat{\boldsymbol{b}} = \boldsymbol{\Phi}^T|_R \boldsymbol{b}|_R. \tag{43}$$

5.1.5. Numerical results

In this section we solve the FKPP problem (26) with the POD (38) and the proposed RAPOD (41) methods and compare the results to the standard FEM.

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

All timing results were generated on a workstation with a 4-core Intel Core i7-6700 CPU with 32 GB 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 [46].

The particulars of the standard FEM solver are given as follows. The 360 snapshots in the set s and the reference solution for x_0 are computed on a uniformly refined triangular cross-pattern triangulation with 256×256 divisions resulting in 262 144 cells. This leads to a finite element space with $\dim(V_h) = 131\,585$. The resulting linear systems at 100 timesteps (34) 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 (34) does not change. The complete set of snapshots takes around 5 min to compute using 4 MPI processes and consumes 3.1 GB space in an HDF5 file. The FEM reference solution is shown at three timesteps in Fig. 5a.

Once the snapshots at s have been computed, the POD eigenvalue problem (11) 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 $\boldsymbol{C}$. We compute this matrix and its eigenvalue decomposition in the l^2 inner-product space before re-orthonormalising the eigenvectors using the classical Gram–Schmidt algorithm in the discrete L^2 inner-product space induced by the finite element basis, i.e. the $\boldsymbol{M}$ inner-product.

The first part of the spectrum of the POD eigenvalue problem (11) is shown in Fig. 1. We retain $O = 17$ basis functions using the 99.9% total energy cutoff criteria. Of course, we do not claim that our choice of snapshots or cutoff criteria are 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 $O = 17$, we can now apply the RAPOD algorithm to construct the reduced triangulation $\mathcal{T}_h^{17}$. Beginning with an initial triangulation $\mathcal{T}_h^0$ covering the original domain Ω with a triangular right-aligned triangulation with 2×2 divisions resulting in 8 cells. As an example, in Fig. 3, for $\text{tol} = 10^{-2}$ we show the first four original unscaled POD basis functions and the corresponding triangulations $\mathcal{T}_h^i$ created by the RAPOD algorithm. Note how the algorithm refines the regions in the domain where the basis functions are most oscillatory, leaving the regions outside of the bottom-left quadrant of the triangulation relatively unrefined. The evolution of the number of cells in triangulations $\left\{\mathcal{T}_h^1, \dots, \mathcal{T}_h^{17}\right\}$ is shown in Fig. 2. The final triangulation used for the RAPOD results $\mathcal{T}_h^{17}$ 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

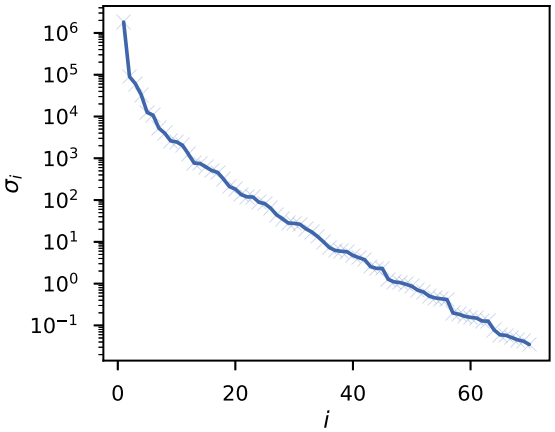


Fig. 1. First part of spectrum of POD eigenvalue problem for the FKPP equation. We use the engineering criteria of retaining the portion of the spectrum which contains approximately 99.9% of the energy of the total spectrum. For the FKPP equation these criteria correspond to retaining 17 basis functions.

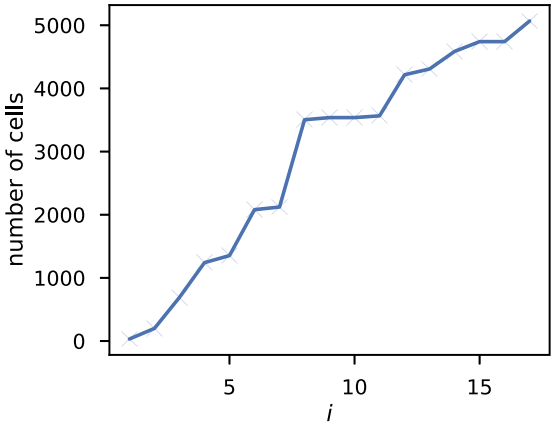


Fig. 2. Evolution of the number of cells in the reduced assembly triangulations for the FKPP problem $\{\mathcal{T}_h^1, \ldots, \mathcal{T}_h^{17}\}$. Note that sometimes no refinement is necessary to achieve the integration tolerance on the subsequent basis function, e.g. $\mathcal{T}_h^{15} = \mathcal{T}_h^{16}$.

converges to that of the standard POD method (tol $= 10^{-5}$ gives $\|e\|_{H^1} = 4.38\%$). Note that with RAPOD we converge to the error committed by the underlying POD model. This implies that we cannot achieve a lower error using RAPOD than with POD alone. This feature of RAPOD is shared with DEIM [14,18]. The blue dashed line shows the wall time for the standard POD simulation (5.32 s). 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 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.

In Fig. 7 we show a comparison in the error committed using different cost function norms L^1 and L^2 to produce the RA triangulations used in the RAPOD solution stage. The L^2 cost indicators and integrals η_i^2, I_a^2 and I_e^2 are the same as those in Eqs. (18)–(20), but replacing $|\varphi_i|$ with φ_i^2. As the RA tolerance is decreased, the error committed by RAPOD with cost function in L^1 norm is close to the one in the L^2 norm. Of course, it is also important to compare the number of cells in the triangulations. In Fig. 8 we compare the number of cells obtained with RAPOD algorithm using the L^1 and L^2 cost functions to produce the RA triangulations. The L^2 norm is slightly more economical than the L^1 norm. However, we note that the error committed in the resulting RAPOD online simulations using meshes generated with the L^1 cost function is consistently lower than the L^2 cost function, but at the expense of more cells (and hence a higher online cost). We use the L^1-norm in the results henceforth in this section.

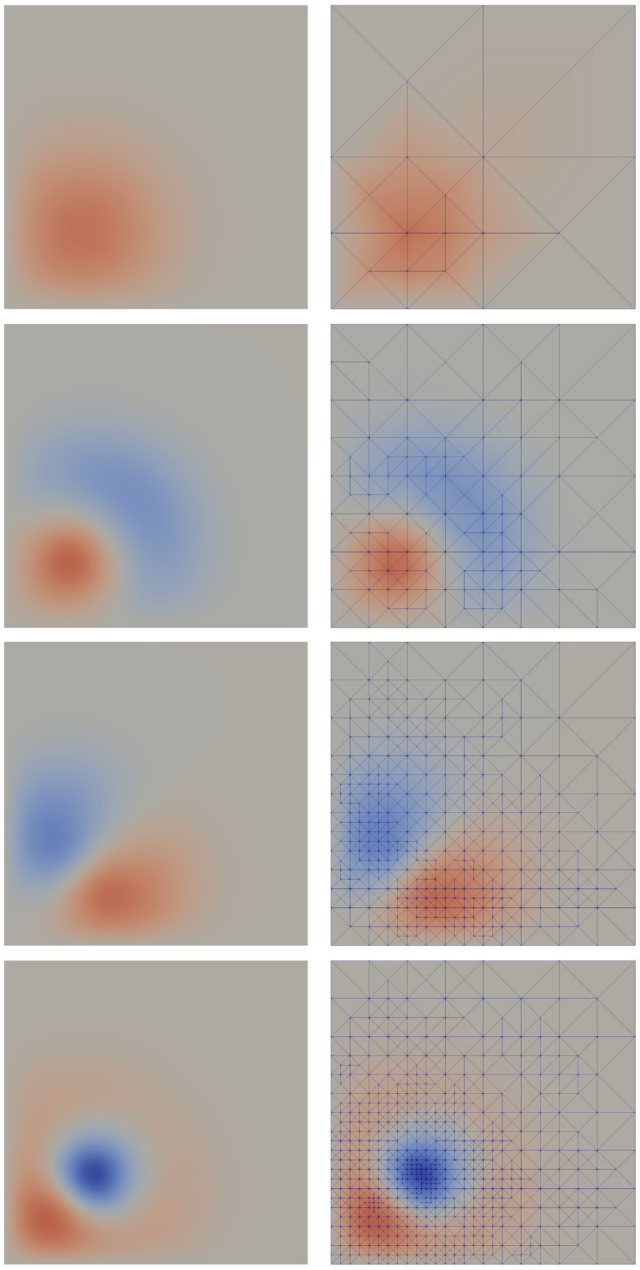


Fig. 3. Left column: original unscaled POD basis function $\sqrt{\sigma_i}\,\varphi_i$ represented on the finite element function space V_h for $i = \{1, 2, 3, 4\}$. Right column: interpolated unscaled POD basis function $\sqrt{\sigma_i}\,\varphi_i|_R$ and corresponding reduced assembly triangulations $\mathcal{T}_h^i$ obtained using the RAPOD algorithm with 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 $\mathcal{T}_h^{17}$ in the online stage for all operations, see Fig. 4.

Figs. 9 (linear scale) and 10 (log scale) show a breakdown of wall time in key stages for each method. *Projections* are the operations taking operators on the finite element spaces to the POD (or RAPOD) space, or from the POD (or RAPOD) space back to the finite element space. *Assemble RHS* are the standard finite element assembly operations on the finite element space, here primarily the right-hand side vector $\boldsymbol{b}$. *Linear algebra* are linear system solves, e.g. Cholesky (POD and RAPOD) or preconditioned conjugate gradient (FEM). *Total* time is the wall (clock) time. The runtime of the FEM is dominated by assembly of the right-hand side operator $\boldsymbol{b}$ and linear algebra solves.

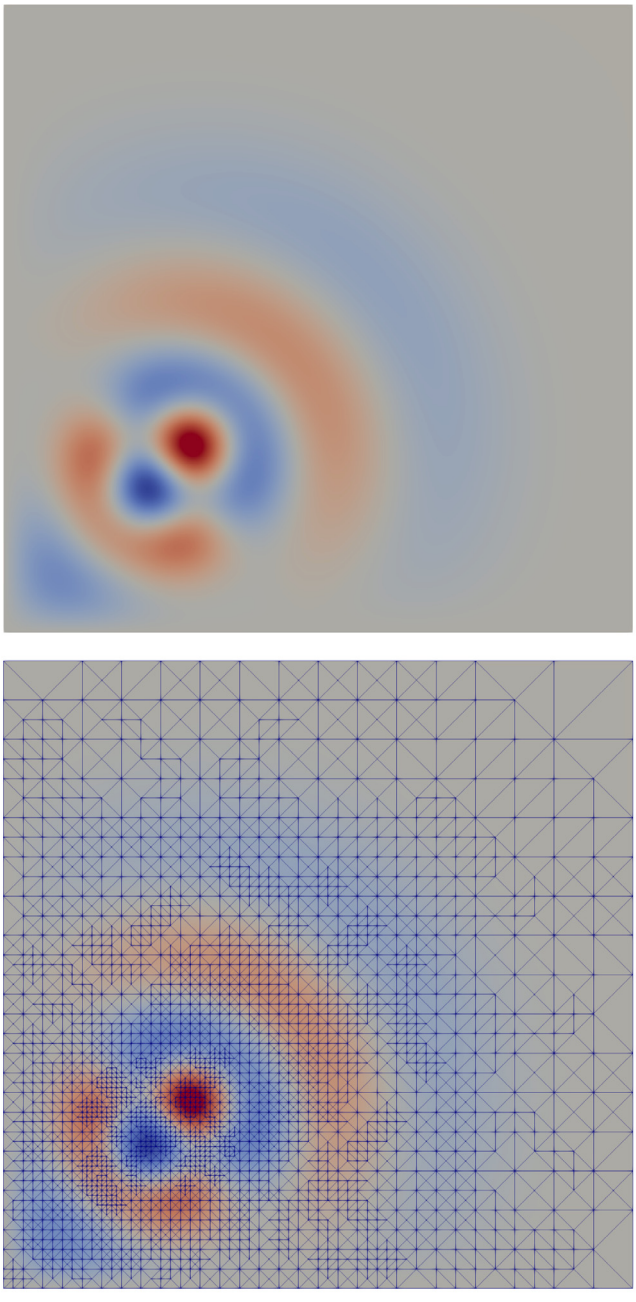
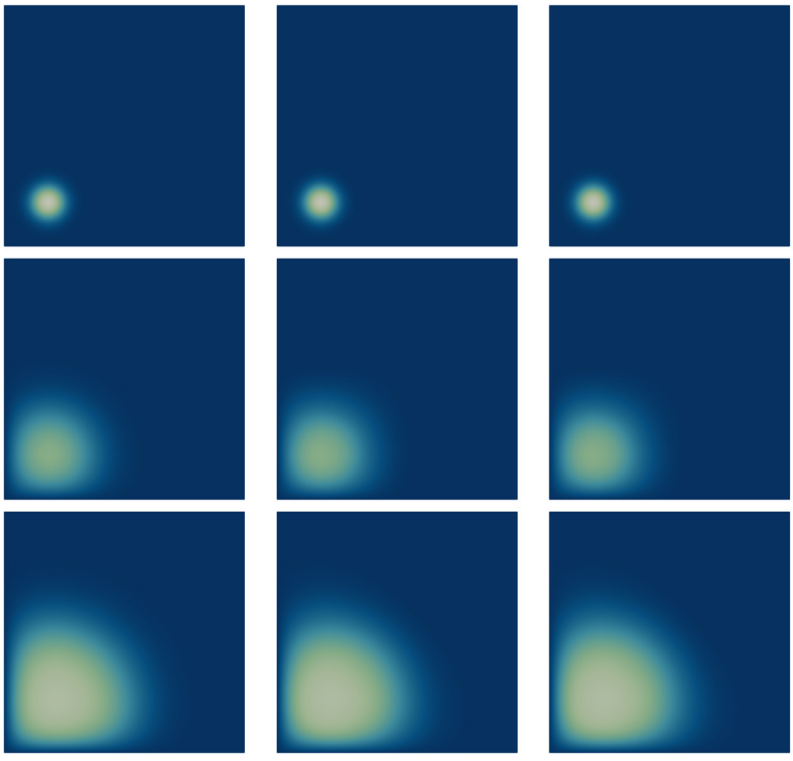


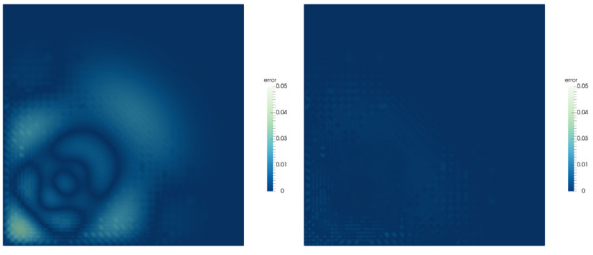
Fig. 4. Top: original unscaled POD basis function $\sqrt{\sigma_{17}}\,\varphi_{17}$ Bottom: interpolated unscaled POD basis function $\sqrt{\sigma}_{17}\,\varphi_{17}|_R$ and corresponding reduced assembly triangulation $\mathcal{T}_h^{17}$ obtained using the RAPOD algorithm with $\text{tol} = 10^{-2}$. The RAPOD procedure interpolates all 17 basis functions 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, 9 and 10.

Moving to POD, linear algebra solves become an almost negligible cost, 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. 11 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



(a) FEM (left column), POD (centre column), RAPOD (right column) solution at $t = \{0.0, 0.05, 0.1\}$ (rows).



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

Fig. 5. Solutions and pointwise l^1 errors of FKPP problem using the three methods.

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. 12. In this context this speed-up could be useful for problems requiring near real-time performance, e.g. online optimal control or interactive simulations. 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–reaction equation. Speed-ups of around 5 times are possible, with only a small increase in error (5%) with respect

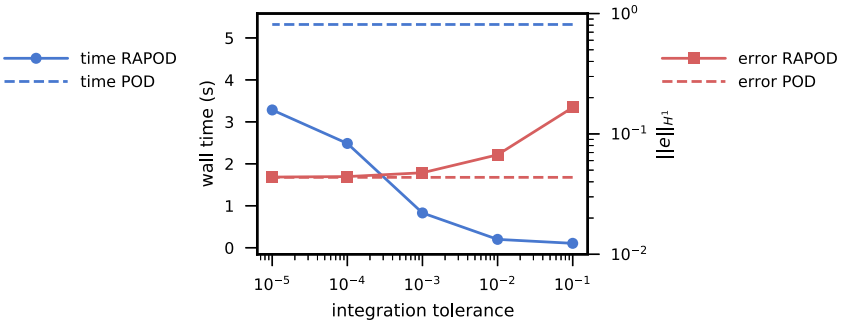


Fig. 6. Wall time and relative error in H^1-norm with respect to FEM simulation against RAPOD integration tolerance tols $= \{10^{-1}, 10^{-2}, \ldots, 10^{-5}\}$. For an integration tolerance of 10^{-3} a speedup of around five times is possible with RAPOD with respect to POD, with only a small increase in error. 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

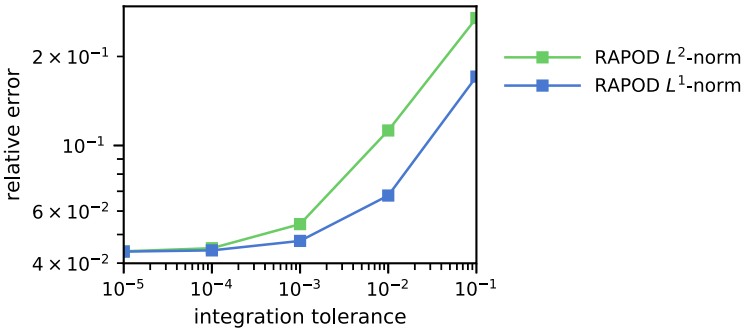


Fig. 7. Relative error of RAPOD simulation in H^1-norm with respect to FEM simulation using L^1 and L^2 norms for RA cost function, integration tolerance tols $= \{10^{-1}, 10^{-2}, \ldots, 10^{-5}\}$. As the integration tolerance of the RAPOD algorithm is decreased (leading to triangulation refinement) we can see that the RAPOD error converges to the same error when using either the L^1 and L^2 norms for the RA cost function.

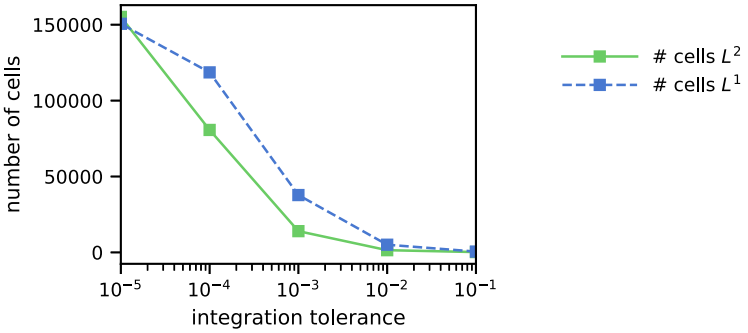


Fig. 8. Evolution in the number of cells in the triangulation for integration tolerances tols $= \{10^{-1}, 10^{-2}, \ldots, 10^{-5}\}$ using the L^1 and L^2 norms for the RA cost function. The L^2 norm is slightly more economical for this problem, but also produces simulations with higher errors (c.f. Fig. 7).

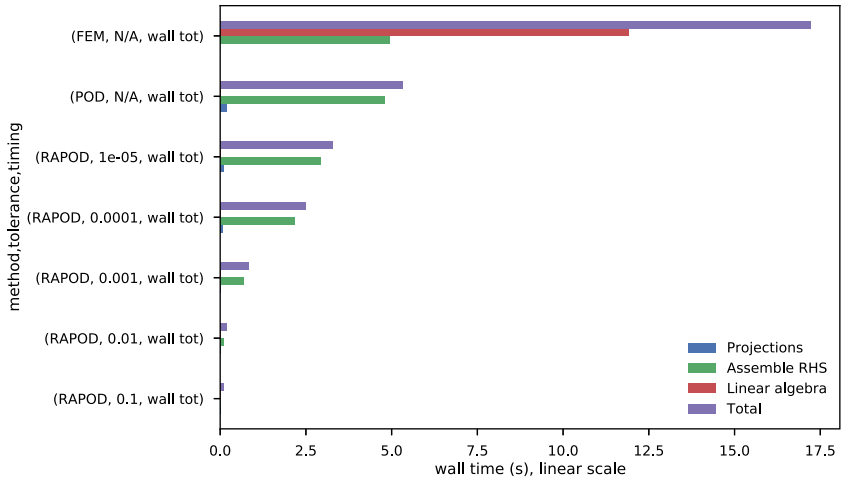


Fig. 9. Breakdown of computational time spent in key stages of the total wall time for the FKPP problem: projections (POD and RAPOD), assembly of right-hand sides, linear algebra solves (preconditioned conjugate-gradient for FEM, dense Cholesky solves for POD/RAPOD). The same data is shown on a log scale in Fig. 10. 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.

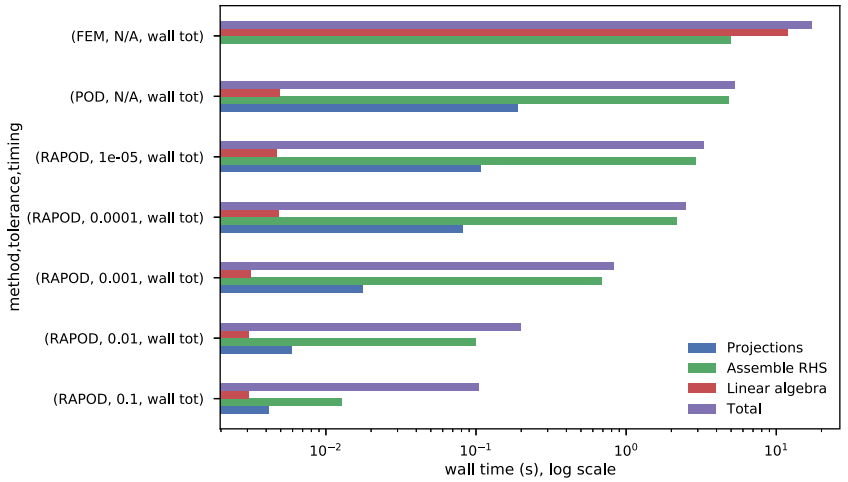


Fig. 10. The same timing data is shown on a linear scale in Fig. 9. With this scaling we can more clearly see the dominance of assembly over solution time for the RAPOD and POD methods.

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

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 [47]. 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 $\Omega_0 \coloneqq \chi_0(\mathcal{B}) = [0, 1]^3$ and in the deformed configuration $\Omega \coloneqq \chi(\mathcal{B})$. Then, for every point in the continuum body $\mathcal{B}$, are related to points $\mathsf{X} \in \Omega_0$ in the underformed configuration

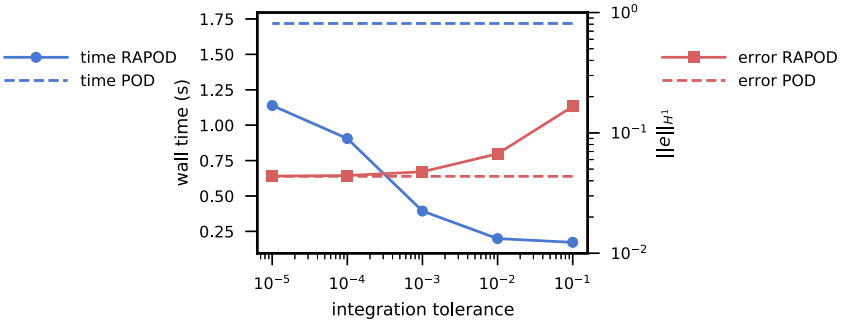


Fig. 11. Initial results from parallel implementation of using MPI running on 4 MPI processes, cf. Fig. 6 for 1 MPI process. Assembly and projections are performed in parallel across all processes while the small dense Cholesky solve is performed serially on every process simultaneously. For the larger sized problems (POD and RAPOD tols $= \{10^{-3}, \ldots, 10^{-5}\}$) we can achieve a speed-up of around 3.1 times.

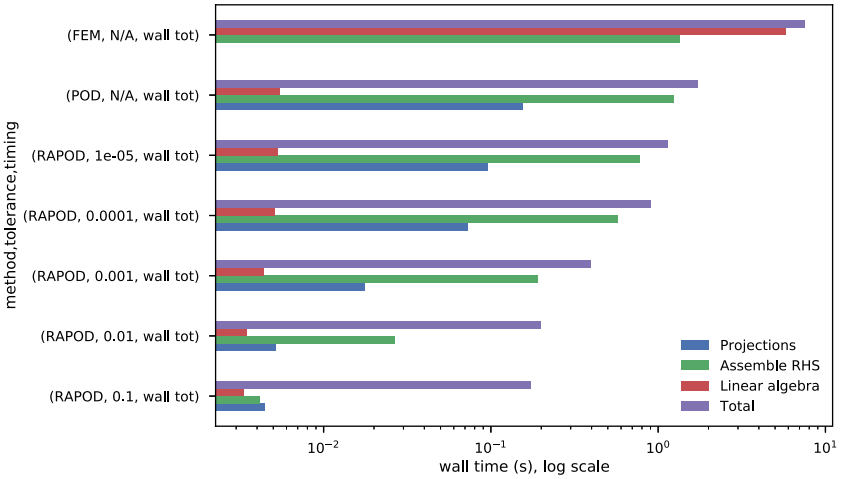


Fig. 12. Breakdown of computational time spent in key stages of the total wall time for the FKPP problem running on 4 MPI processes, cf. Fig. 10. For the loose RAPOD tolerances e.g. 10^{-2}, we have significant overhead unrelated to the main computational operations.

and $\mathsf{x} \in \Omega$ in the deformed configuration through the maps χ_0 and χ respectively, we can construct a sufficiently smooth deformation map $\psi : \Omega_0 \ni \mathsf{X} \mapsto \mathsf{x} \in \Omega$

$$\psi = \chi \circ \chi_0^{-1}, \tag{44}$$

from which we can define the deformation gradient tensor as

$$\mathsf{F}(\mathsf{X}) := \frac{\partial \psi}{\partial \mathsf{X}}, \tag{45}$$

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

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

$$\mathsf{C} := \mathsf{F}^T\mathsf{F}, \quad \mathsf{E} = \frac{1}{2}(\mathsf{C} - \mathsf{I}), \tag{46}$$

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

$$W(\mathsf{F}) := \frac{\mu}{2}(I_{\mathsf{C}} - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2, \tag{47}$$

where $I_{\mathsf{C}} := \mathrm{tr}(\mathsf{C})$ and $III_{\mathsf{C}} = \det(\mathsf{C}) = J^2 = [\det(\mathsf{F})]^2$ 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.

We set the Young's modulus $E = 10$ Pa and Poisson's ratio $\nu = 0.3$ which can be related to μ and λ through

$$\mu = \frac{E}{2(1+\nu)}, \tag{48a}$$

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}. \tag{48b}$$

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

$$\mathsf{u}^* = \arg\min_{\mathsf{u}\in V} \left\{ \int_{\Omega_0} W \mathrm{d}x_0 - \int_{\partial_N \Omega_0} \mathsf{t}\cdot\mathsf{u}\, \mathrm{d}s_0 \right\} \tag{49}$$

$$= \arg\min_{\mathsf{u}\in V} \mathcal{E}(\mathsf{u}), \tag{50}$$

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

$$\partial_D \Omega_0 := \{(x,y,0)\times(x,y,0)\}. \tag{51}$$

$\mathsf{t} \in [L^2(\partial_N\Omega_0)]^3$ are surface tractions (Neumann boundary conditions) applied on the top surface of the cube $\partial_N\Omega_0 := \{(x,y,1)\times(x,y,1)\}$ and $\mathrm{d}x_0$ and $\mathrm{d}s_0$ are measures on the undeformed domain Ω_0 and its boundary $\partial\Omega_0$, respectively. The traction vector t on $\partial_N\Omega_0$ is set to be

$$\mathsf{t}(\mathsf{X}) := \left\{ 0, 0, -6p\exp\left(-\frac{(\mathsf{X}-\mathsf{X}_0)^2}{\sigma^2}\right)\right\}^T, \tag{52}$$

with $\sigma = 0.15$ and $\mathsf{X}_0 \in \partial_N\Omega_0$. We let the load position X_0 and magnitude p Pa form our parameters space M, and choose the discrete training set s for the snapshot generation in the POD process as

$$s_{\mathsf{X}_0} = \{0.3, 0.4, 0.5, 0.6\}^2 \times \{1.0, 1.0, \ldots, 1.0\}, \tag{53a}$$

$$s_p = \{0.1, 0.2, \ldots, 0.9, 1.0\}, \tag{53b}$$

$$s = s_{\mathsf{X}_0} \times s_p, \quad |s| = 160. \tag{53c}$$

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

$$\forall \tilde{\mathsf{u}} \in \hat{V}, \quad D_{\mathsf{u}}[\mathcal{E}(\mathsf{u})][\tilde{\mathsf{u}}]|_{\mathsf{u}=\mathsf{u}^*} = F(\mathsf{u}^*;\tilde{\mathsf{u}}) = 0, \tag{54}$$

where $D_{\mathsf{u}}[\mathcal{E}(\mathsf{u})][\tilde{\mathsf{u}}]|_{\mathsf{u}=\mathsf{u}^*}$ denotes the usual Gateaux derivative of the functional $\mathcal{E}$ in a direction $\tilde{\mathsf{u}}$ evaluated at the minimum u^*. The above equation can be interpreted as the equilibrium equation. We use the notation $\tilde{\mathsf{u}}$ to signify that these are test functions in a Galerkin sense.

The first order optimality condition can be written in full as: Find $\mathsf{u}^* \in V$ such that

$$\forall \tilde{\mathsf{u}} \in \hat{V}, \quad \int_{\Omega_0} \mathsf{S}(\mathsf{u}^*) : D_{\mathsf{u}}[\mathsf{E}(\mathsf{u})][\tilde{\mathsf{u}}]|_{\mathsf{u}=\mathsf{u}^*}\, \mathrm{d}x_0 - \int_{\partial_N\Omega_0} \mathsf{t}\cdot\tilde{\mathsf{u}}\, \mathrm{d}s_0 = 0, \tag{55}$$

$$D_{\mathsf{u}}[\mathsf{E}(\mathsf{u})][\tilde{\mathsf{u}}] := \frac{1}{2}\left[(\nabla\tilde{\mathsf{u}})^T\mathsf{F}(\mathsf{u}) + (\mathsf{F}(\mathsf{u}))^T\nabla\tilde{\mathsf{u}}\right], \tag{56}$$

where the second Piola–Kirchoff stress tensor $\mathsf{S} := 2\frac{\partial W}{\partial \mathsf{C}}$ is work conjugate with the incremental Green strain tensor. Clearly the residual F is non-linear in the displacement unknown u. The standard method for solving this problem is the Newton–Raphson method as described in Section 2. We derive the Jacobian for the Newton–Raphson method symbolically using the automatic differentiation capabilities of the Unified Form Language [48] of the FEniCS Project.

5.2.1. Finite element discretisation in space

We discretise the hyperelasticity problem in space using a Galerkin H^1-conforming finite element method by introducing discrete trial and test spaces $V_h \subset V := [H^1(\Omega)]^3$ and $\hat{V}_h \equiv V_h$ spanned with vector piecewise linear polynomial finite element basis functions $\{\boldsymbol{\phi}_j\}_{j=1}^N$ with $N = \dim(V_h)/3$ giving

$$\mathsf{u}_h(x) = \sum_{i=1}^{N} \boldsymbol{\phi}_i(\boldsymbol{x})\boldsymbol{u}_i, \tag{57}$$

$$\boldsymbol{u} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N\}^T \in \mathbb{R}^{3N\times 1}, \tag{58}$$

resulting in a discrete Newton–Raphson step

$$\boldsymbol{J}(\boldsymbol{u}^k)\boldsymbol{\delta u}^k = -\boldsymbol{f}(\boldsymbol{u}^k), \tag{59}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{\delta u}^k. \tag{60}$$

We use continuation in the loading parameter p within the set s to ensure the convergence of the Newton–Raphson 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 $\boldsymbol{u}^k$ at the previous time step. In contrast, the hyperelastic problem in this section leads to a Newton–Raphson system containing both a matrix operator (the Jacobian) $\boldsymbol{J}$ and vector operator (the residual) $\boldsymbol{f}$ that are dependent on the solution at the previous Newton–Raphson step $\boldsymbol{u}^k$. Therefore, we must apply the RAPOD procedure to both of these terms to ensure good performance in the online stage.

5.2.2. Galerkin-POD discretisation

The Galerkin-POD procedure is performed using the same procedure described in Section 3 resulting in POD basis functions $\boldsymbol{\Phi} \in \mathbb{R}^{3N\times O}$. This results in the following expressions for the finite element Jacobian and residual projected into the POD space

$$\widetilde{\boldsymbol{J}} = \boldsymbol{\Phi}^T \boldsymbol{J}\, \boldsymbol{\Phi}, \quad \widetilde{\boldsymbol{f}} = \boldsymbol{\Phi}^T \boldsymbol{f}. \tag{61}$$

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

5.2.3. Reduced assembly Galerkin-POD discretisation

We take the pointwise magnitude of every POD basis function $\{\boldsymbol{\varphi}_i\}_{i=1}^O$

$$\|\boldsymbol{\varphi}_i(x)\| = (\varphi_i^x(x)^2 + \varphi_i^y(x)^2 + \varphi_i^z(x)^2)^{1/2}, \tag{62}$$

before applying the RAPOD algorithm 1 to $\|\boldsymbol{\varphi}_i(x)\|$ to obtain triangulation $\mathcal{T}_h^O$. Note that in this case the $\|\boldsymbol{\varphi}_i(x)\|$ is positive everywhere and thus the abs-norm in the RAPOD procedure does nothing.

As before, we reconstruct the Galerkin-POD problem on the finite element function space $V_h^R \subset V$, associated with the triangulation $\mathcal{T}_h^O$ derived from the RAPOD algorithm applied to the Oth POD basis function. We construct a discrete interpolation operator $P : V_h \to V_h^R$ and interpolate every POD basis function

$$\boldsymbol{\varphi}_i|_R = P\boldsymbol{\varphi}_i \quad i = 1, \ldots, O, \tag{63}$$

into V_h^R.

With this procedure complete, we simply follow the standard Galerkin-POD as in Section 5.1.3 but using the new POD basis functions $\boldsymbol{\varphi}|_R$ interpolated in the finite element space V_h^R and new Jacobian operators $\boldsymbol{J}|_R$ and $\boldsymbol{f}|_R$ assembled in the finite element space V_h^R

$$\widehat{\boldsymbol{J}} = \boldsymbol{\Phi}^T|_R\, \boldsymbol{J}|_R\, \boldsymbol{\Phi}|_R, \quad \widehat{\boldsymbol{f}} = \boldsymbol{\Phi}^T|_R \boldsymbol{f}|_R. \tag{64}$$

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. In addition, we solve the hyperelasticity problem using the standard FEM method constructed on the meshes created using the RA algorithm. We call this method RAFEM henceforth.

We set the position parameter controlling the centre of the source term as $x_0 = (0.55, 0.55)$. This parameter is not in the set s, so we are testing the predictive capabilities of the POD and RAPOD models.

The particulars of the standard FEM solver are given as follows. The 160 snapshots computed at the points in the set s 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 196 608 cells. This leads to a finite element space $\dim(V_h) = 107\,811$. The resulting linear systems at each Newton iteration (34) are solved using a algebraic multigrid preconditioned GMRES

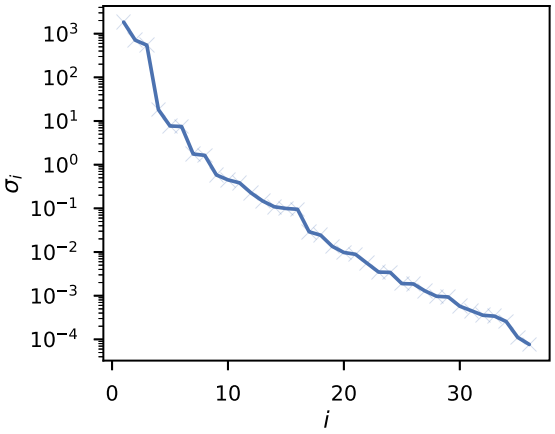


Fig. 13. First part of spectrum of POD eigenvalue problem for the hyperelasticity equation. We use the engineering criteria of retaining the portion of the spectrum which contains approximately 99.99% of the energy of the total spectrum. For the hyperelasticity problem these criteria correspond to retaining 14 basis functions.

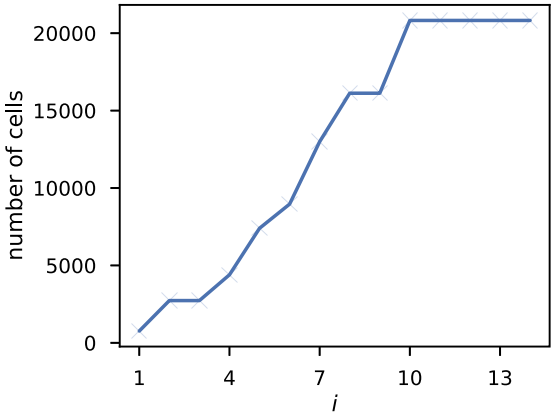


Fig. 14. Evolution of the number of cells in the reduced assembly triangulations for the hyperelasticity problem. $\{\mathcal{T}_h^1, \ldots, \mathcal{T}_h^{14}\}$. Note that sometimes no refinement is necessary to achieve the integration tolerance on the subsequent basis function, e.g. $\mathcal{T}_h^{13} = \mathcal{T}_h^{14}$.

method in PETSc. The same solution strategy is used in the RAFEM method. The complete set of snapshots takes around 3 h to compute using 4 MPI processes and consumes 2.3 GB space in an HDF5 file.

Once the snapshot set has been computed we solve the POD eigenvalue problem as for the FKPP problem Section 5.1.

The first part of the spectrum of the POD eigenvalue problem (11) is shown in Fig. 13. We retain $O = 14$ basis functions using a 99.99% total energy cutoff criteria.

We can now apply the RAPOD algorithm to construct the reduced triangulation $\mathcal{T}_h^{14}$. Beginning with an initial triangulation $\mathcal{T}_h^0$ covering the original domain Ω with a tetrahedral right-aligned triangulation with $2 \times 2 \times 2$ divisions resulting in 48 cells. Fig. 14 shows the evolution of the number of cells in triangulations $\{\mathcal{T}_h^1, \ldots, \mathcal{T}_h^{14}\}$. Fig. 15 shows a cut through of the triangulation $\mathcal{T}_h^{14}$ created with the RAPOD algorithm with a tolerance $\text{tol} = 10^{-2}$.

Figs. 16–18 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.

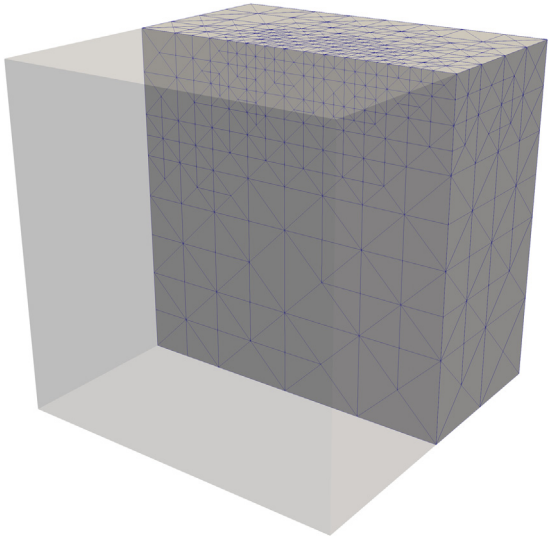


Fig. 15. RAPOD triangulation $\mathcal{T}_h^{14}$ on domain Ω_0 created with tol $= 10^{-2}$. Used in result shown in Fig. 18. Grey transparent box is initial configuration Ω_0.

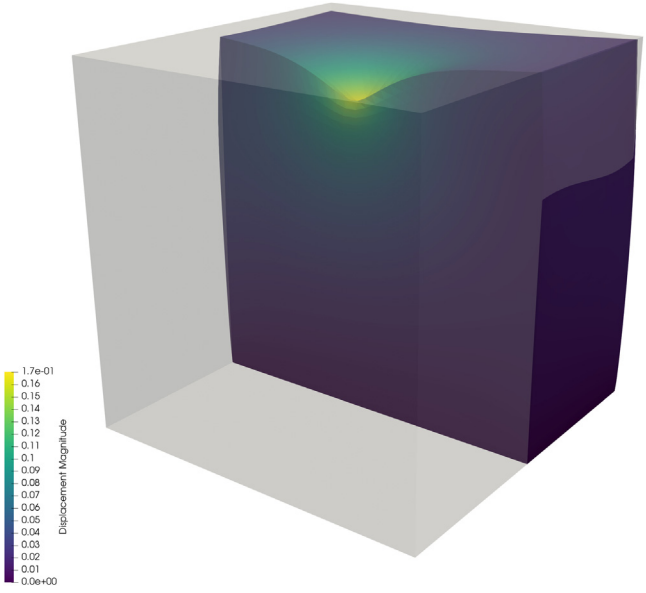


Fig. 16. Results of Hyperelasticity problem with FEM method. Cut through of deformed domain Ω coloured with magnitude of the displacements $\|\mathsf{u}^*\|_2$. Grey transparent box is initial configuration Ω_0.

The RAPOD method can achieve this at significantly reduced computational cost with respect to the standard POD method. In Fig. 19 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 s). 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–Raphson 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

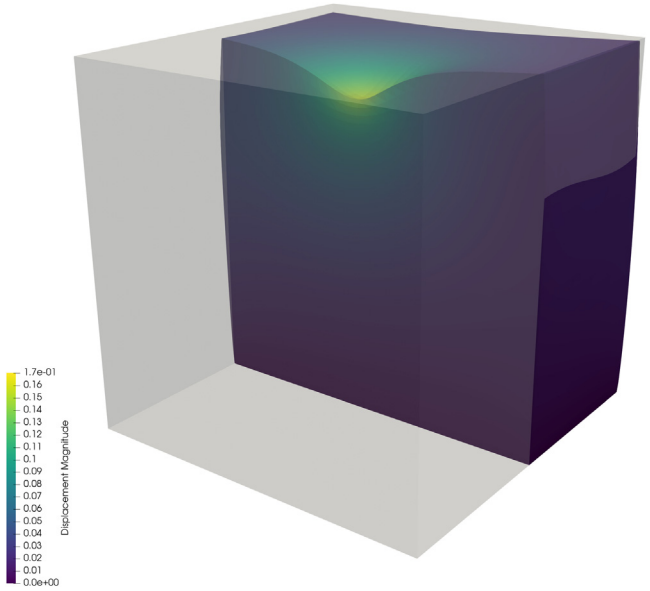


Fig. 17. Results of Hyperelasticity problem with POD method with $O = 14$.

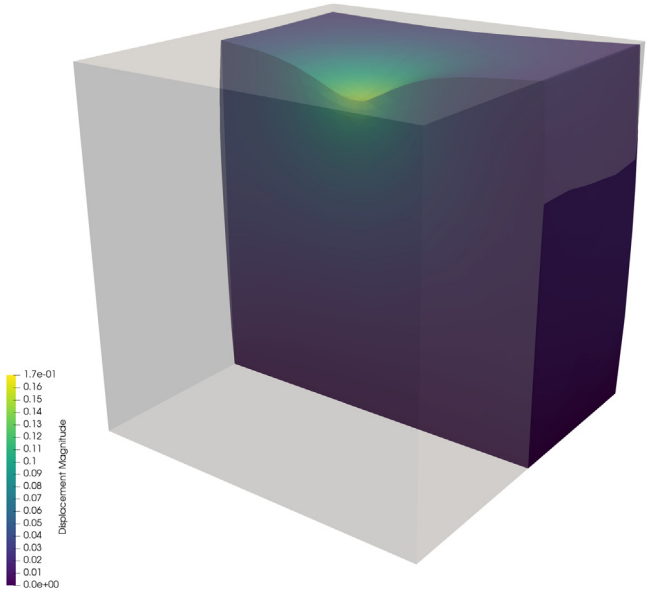


Fig. 18. Results of Hyperelasticity problem with RAPOD method with $O = 14$ and $\text{tol} = 10^{-2}$.

all tolerances. We can see that the RAFEM method provides a slightly lower error than the RAPOD method for the same tolerance, but at the expense of a significantly longer runtime due to the increased cost of linear system solutions on the large sparse Newton–Raphson systems.

A breakdown of the amount of time spent in key computational areas for the FEM, POD, RAFEM and RAPOD methods (for different tolerances) is shown in Figs. 20 (linear scale) and 21 (log scale). The FEM solution time is dominated by mainly linear system solves, with a significant proportion of time also spent assembling Jacobians and residuals. In contrast, for the standard POD method, linear solves become almost free and the total computational time is dominated by assembly on the finite element space. The proposed RAPOD method significantly reduces these assembly costs. The RAFEM method gains all of the advantages of decreased assembly costs from the proposed

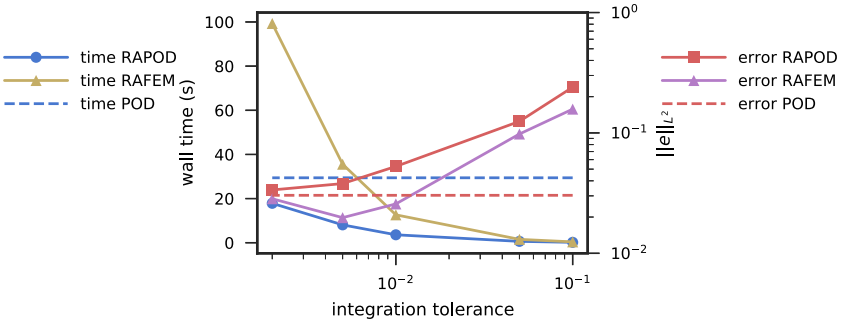


Fig. 19. Wall time and relative error for RAPOD, RAFEM and POD in L^2-norm with respect to FEM simulation against RA integration tolerance for the hyperelasticity problem using a $32 \times 32 \times 32$ mesh. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

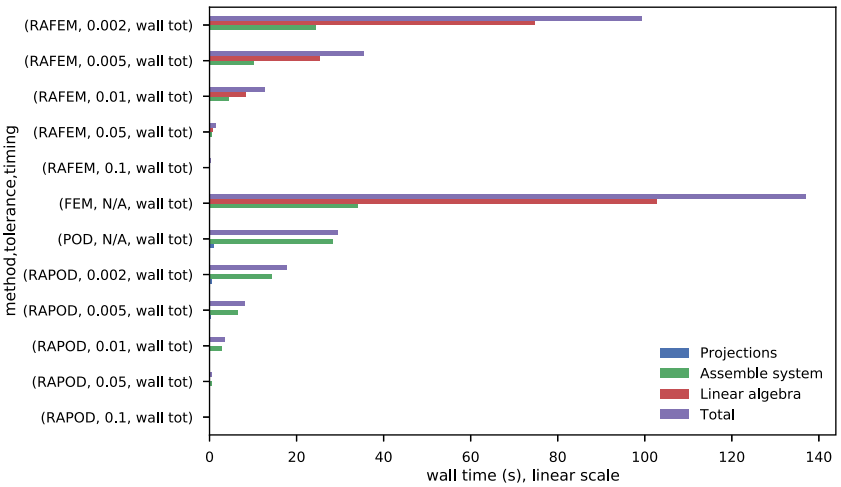


Fig. 20. Breakdown of computational time spent in key stages of the total wall time for the hyperelasticity problem: projections (POD and RAPOD), assembly of Jacobians and residuals, linear algebra solves (preconditioned GMRES for FEM, dense Cholesky solves for POD/RAPOD). The same data is shown on a log scale in Fig. 10. 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.

RA method, but none of the advantages of the reduced solution time in POD or RAPOD. Linear system solves dominate the computational cost of the RAFEM method for RA triangulations with more cells. The lower time for assembly of the POD method vs. FEM is due to the slightly lower number of Newton steps required in the POD method. We are currently unsure as to why the dense Cholesky solves for the RAPOD method require less time at the higher tolerances. In total however, the solve time for POD and RAPOD is negligible compared with the assembly costs. In summary, RAPOD alleviates the assembly bottleneck of POD, creating new reduced-order models with significantly lower runtimes.

In Fig. 22 we show the total number of Newton–Raphson iterations required to solve the hyperelasticity problem using the RAFEM and RAPOD methods for varying RA algorithm tolerances. For all RA tolerances, RAPOD requires the same number of Newton iterations as the standard POD method, indicating that the Jacobian and residual of the Newton–Raphson iterations are sufficiently well approximated.

In Fig. 23, we give an indication of the actual scaling behaviour of our implementation of the RAPOD method. We see a slightly sublinear trend in the total time taken for increasing reduced assembly finite element space size $\dim(\hat{V}_h)$. This is what we would expect given that assembly time dominates the overall cost of the RAPOD method and that assembly time should scale linearly in an optimal FEM code.

Finally, we show the ability of RAPOD to achieve errors lower than 1%. Using exactly the same problem setup as outlined above, except using a finer mesh with $64 \times 64 \times 64$ divisions to generate the snapshots and POD basis.

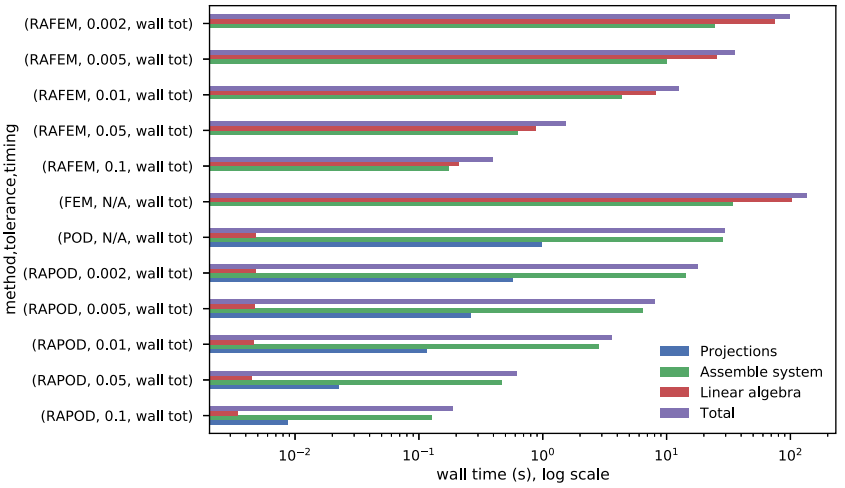


Fig. 21. The same timing data is shown on a linear scale in Fig. 20. With this scaling we can more clearly see the dominance of assembly over solution time for the RAPOD and POD methods.

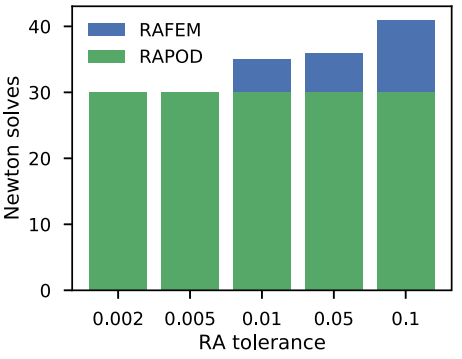


Fig. 22. Total number of Newton–Raphson iterations to solve hyperelasticity problem using RAFEM and RAPOD methods for varying RA tolerances. POD required 30 Newton iterations, and the FEM model required 34 Newton iterations.

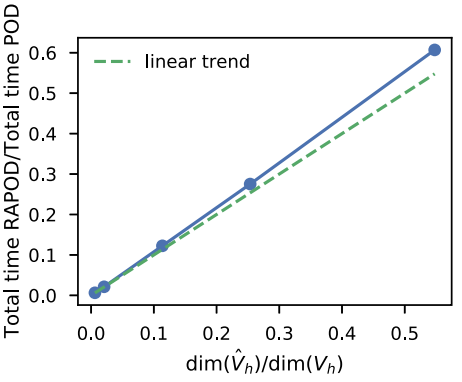


Fig. 23. Scaling plot showing runtime of RAPOD method normalised against the POD runtime vs. the RAPOD finite element space dimension normalised against the POD/FEM function space dimension. A linear trend is shown for reference. We obtain close to linear scaling in the RAPOD problem size.

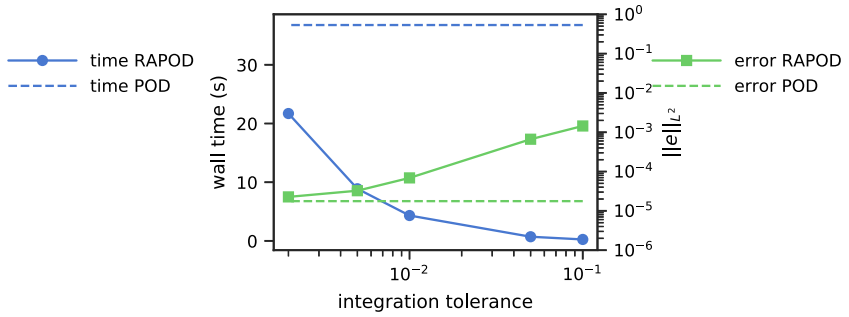


Fig. 24. Wall time and relative error for RAPOD, RAFEM and POD in L^2-norm with respect to FEM simulation against RA integration tolerance for the hyperelasticity problem using a $64 \times 64 \times 64$ mesh.

This results in a POD model with $\|e\|_{L^2}$ of around 0.002%. In Fig. 24 we show the error and wall times of the POD model and RAPOD models constructed with varying tolerances. When using a tight enough tolerance, the RAPOD method can achieve an error on the same order as the POD method. For all tolerances RAPOD method has a lower wall time than the POD method, with a resulting tradeoff in terms of accuracy.

6. Conclusion

In this paper we have presented a new hyper-reduction method called Reduced Assembly (RA) to cut 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, resulting in a method we called 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 make a few closing remarks about the applicability and limitations of the proposed method. If the POD basis functions are highly oscillatory throughout the entire domain, then our RA method will provide no advantages. In this case, the POD basis functions are suggesting that all areas of the domain are of equal importance. This limitation applies uniformly to hyper-reduction methods that rely on choosing a limited subset of degrees of freedom or regions of space for integration. However, if the POD basis functions are highly oscillatory but have local support in the domain, then our RA method is still applicable.

In this contribution we have not tackled the issue of how to compute each snapshot on a different triangulation and then how to compute a single POD basis. This is necessary for an efficient offline stage. The authors of [49] tackle this important problem. The output of that method could be used as input for our RAPOD procedure, to produce a hyper-reduced online model.

We are currently investigating improvements to the cell error indicator performance and anisotropic triangulation refinement strategies to produce triangulations with even fewer cells. Another interesting line of research would be to come up with multiple full-order models using the RAPOD procedure and then pick the best one in the online stage depending on the requested point in parameter space.

Supplementary Material

Full code to produce all of the examples and figures in this paper is available at [50].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank both the anonymous reviewers for their feedback which has improved the quality of the manuscript greatly. We would also like to thank our colleague Prof. Andreas Zilian for proofreading a revision of the manuscript and providing valuable feedback.

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 the Marie Curie Actions of the European Commission (FP7-COFUND) Grant No. 6693582. Davide Baroli is supported by RP MACAFIVE, and funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612. The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [51] and at Cardiff University.

References

[1] P. Morin, R.H. Nochetto, K.G. Siebert, Convergence of adaptive finite element methods, SIAM Rev. 44 (4) (2002) 631–658, http://dx.doi.org/10.1137/S0036144502409093.

[2] S. Niroomandi, D. González, I. Alfaro, F. Bordeu, A. Leygue, E. Cueto, F. Chinesta, Real-time simulation of biological soft tissues: a PGD approach, Int. J. Numer. Methods Biomed. Eng. 29 (5) (2013) 586–600, http://dx.doi.org/10.1002/cnm.2544.

[3] P.G. Constantine, Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies, SIAM, 2015, http://dx.doi.org/10.1137/1.9781611973860.

[4] Y. Maday, E. Rønquist, A reduced-basis element method, J. Sci. Comput. 17 (1) (2002) 447–459, http://dx.doi.org/10.1016/S1631-073X(02)02427-5.

[5] C. Prud'Homme, D. Rovas, L. Veroy, L. Machiels, Y. Maday, A. Patera, G. Turinici, Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods, J. Fluids Eng. 124 (1) (2002) 70–80, http://dx.doi.org/10.1115/1.1448332.

[6] P. Constantine, Q. Wang, Residual minimizing model interpolation for parametrized nonlinear dynamics systems, SIAM J. Sci. Comput. 34 (4) (2012) http://dx.doi.org/10.1137/100816717, A21118–A2144.

[7] E. Schenone, S. Veys, C. Prud'Homme, High performance computing for the reduced basis method. Application to natural convection, ESAIM: Proc. 43 (December) (2013) 255–273, http://dx.doi.org/10.1051/proc/201343016.

[8] K. Pearson, On lines and planes of closest fit to systems of points, Phil. Mag. 2 (6) (1901) 559–572, http://dx.doi.org/10.1080/14786440109462720.

[9] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 23 (1933) 417–441, http://dx.doi.org/10.1037/h0071325.

[10] J.L. Lumley, The structure of inhomogeneous turbulence, Atmos. Turbul. Radio Wave Propag. (1967) 166–178.

[11] L. Sirovich, Turbulence and the dynamics of coherent structures. Part I: Coherent structures, Quart. Appl. Math. 45 (1987) 561–571, URL https://www.jstor.org/stable/i40145482.

[12] L. Sirovich, C.H. Sirovich, Low dimensional description of complicated phenomena, Contemp. Math. 99 (1989) 277–305, http://dx.doi.org/10.1090/conm/099.

[13] M. Boulakia, E. Schenone, J. Gerbeau, Reduced-order modeling for cardiac electrophysiology. Application to parameter identification, Int. J. Numer. Methods Biomed. Eng. 28 (6–7) (2012) 727–744, http://dx.doi.org/10.1002/cnm.2465.

[14] A. Radermacher, S. Reese, POD-based model reduction with empirical interpolation applied to nonlinear elasticity, Internat. J. Numer. Methods Engrg. (2015) http://dx.doi.org/10.1002/nme.5177.

[15] F. Chinesta, A. Ammar, A. Leygue, R. Keunings, An overview of the proper generalized decomposition with applications in computational rheology, J. Non-Newton. Fluid Mech. 166 (11) (2011) 578–592, http://dx.doi.org/10.1016/j.jnnfm.2010.12.012.

[16] M. Barrault, Y. Maday, N. Nguyen, A. Patera, An 'empirical interpolation' method: Application to efficient reduced-basis discretization of partial differential equations, C. R. Math. Acad. Sci. Paris 339 (2004) 667–672, http://dx.doi.org/10.1016/j.crma.2004.08.006.

[17] M. Grepl, Y. Maday, N. Nguyen, A. Patera, Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equation, ESAIM Math. Model. Numer. Anal. 41 (3) (2007) 575–605, http://dx.doi.org/10.1051/m2an:2007031.

[18] S. Chaturantabut, D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764, http://dx.doi.org/10.1137/090766498.

[19] S. Chaturantabut, D. Sorensen, A state space error estimate for POD-DEIM nonlinear model reduction, SIAM J. Numer. Anal. 50 (1) (2012) 46–63, http://dx.doi.org/10.1137/110822724.

[20] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Trans. Automat. Control 53 (10) (2008) 2237–2251, http://dx.doi.org/10.1109/TAC.2008.2006102.

[21] J.A. Hernández, M.A. Caicedo, A. Ferrer, Dimensional hyper-reduction of nonlinear finite element models via empirical cubature, Comput. Methods Appl. Mech. Engrg. 313 (2017) 687–722, http://dx.doi.org/10.1016/j.cma.2016.10.022.

[22] D. Ryckelynck, Hyper-reduction of mechanical models involving internal variables, Internat. J. Numer. Methods Engrg. 77 (1) (2009) 75–89, http://dx.doi.org/10.1002/nme.2406.

[23] R. Everson, L. Sirovich, Karhunen–Loève procedure for gappy data, J. Opt. Soc. Amer. A 12 (8) (1995) 1657, http://dx.doi.org/10.1364/JOSAA.12.001657.

[24] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations, Internat. J. Numer. Methods Engrg. 86 (2) (2011) 155–181, http://dx.doi.org/10.1002/nme.3050.

[25] T. Taddei, An offline/online procedure for dual norm calculations of parameterized functionals: empirical quadrature and empirical test spaces, Adv. Comput. Math. 45 (5) (2019) 2429–2462, http://dx.doi.org/10.1007/s10444-019-09721-w.

[26] M. Yano, A.T. Patera, An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs, Comput. Methods Appl. Mech. Engrg. 344 (2019) 1104–1123, http://dx.doi.org/10.1016/j.cma.2018.02.028.
[27] Y. Choi, D. Coombs, R. Anderson, SNS: A solution-based nonlinear subspace method for time-dependent model order reduction, SIAM J. Sci. Comput. 42 (2) (2020) A1116–A1146, http://dx.doi.org/10.1137/19M1242963.
[28] R. Reyes, R. Codina, Element boundary terms in reduced order models for flow problems: Domain decomposition and adaptive coarse mesh hyper-reduction, Comput. Methods Appl. Mech. Engrg. 368 (2020) 113159, http://dx.doi.org/10.1016/j.cma.2020.113159.
[29] R. Reyes, R. Codina, Projection-based reduced order models for flow problems: A variational multiscale approach, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112844, http://dx.doi.org/10.1016/j.cma.2020.112844.
[30] F. Fritzen, B. Haasdonk, D. Ryckelynck, S. Schöps, An algorithmic comparison of the hyper-reduction and the discrete empirical interpolation method for a nonlinear thermal problem, Math. Comput. Appl. 23 (1) (2018) 8, http://dx.doi.org/10.3390/mca23010008.
[31] E. Tadmor, R. Philips, M. Ortiz, Mixed atomistics and continuum models of deformation in solids, Langmuir 12 (1996) 4529–4534, http://dx.doi.org/10.1021/la9508912.
[32] J. Knap, M. Ortiz, An analsys of the quasicontinuum method, J. Mech. Phys. Solids 49 (2001) 1899–1923, http://dx.doi.org/10.1016/S0022-5096(01)00034-5.
[33] M. Gunzburger, Y. Zhang, A quadrature-type approximation to the quasi-continuum method, Multiscale Model. Simul. 8 (2010) 571–590, http://dx.doi.org/10.1137/080722151.
[34] Q. Yang, E. Biyikli, A. To, Multiresolution molecular mechanics: statics, Comput. Methods Appl. Mech. Engrg. 258 (2013) 26–38, http://dx.doi.org/10.1016/j.cma.2013.01.014.
[35] L. Beex, R. Peerlings, M. Geers, Central summation in the quasicontinuum method, J. Mech. Phys. Solids 70 (2014) 242–261, http://dx.doi.org/10.1016/j.jmps.2014.05.019.
[36] L.A.A. Beex, P. Kerfriden, T. Rabczuk, S.P.A. Bordas, Quasicontinuum-based multiscale approaches for plate-like beam lattices experiencing in-plane and out-of-plane deformation, Comput. Methods Appl. Mech. Engrg. 279 (Supplement C) (2014) 348–378, http://dx.doi.org/10.1016/j.cma.2014.06.018.
[37] J. Amelang, G. Venturini, D. Kochmann, Summation rules for a fully nonlocal energy-based quasicontinuum method, J. Mech. Phys. Solids 82 (2015) 378–413, http://dx.doi.org/10.1016/j.jmps.2015.03.007.
[38] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, G.N. Wells, The fenics project version 1.5, Arch. Numer. Softw. 3 (100) (2015) http://dx.doi.org/10.11588/ans.2015.100.20553.
[39] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The `deal.II` library, version 9.0, J. Numer. Math. 26 (4) (2018) 173–183, http://dx.doi.org/10.1515/jnma-2018-0054.
[40] C. Burstedde, L.C. Wilcox, O. Ghattas, P4est : Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133, http://dx.doi.org/10.1137/100791634.
[41] A. Logg, G.N. Wells, DOLFIN: Automated finite element computing, ACM Trans. Math. Software 37 (2) (2010) 20:1–20:28, http://dx.doi.org/10.1145/1731022.1731030.
[42] S. B., S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016, URL http://www.mcs.anl.gov/petsc.
[43] V. Hernandez, J.E. Roman, V. Vidal, SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems, ACM Trans. Math. Softw. 31 (3) (2005) 351–362, http://dx.doi.org/10.1145/1089014.1089019.
[44] P. Gonnet, A review of error estimation in adaptive quadrature, ACM Comput. Surv. 44 (4) (2012) 22:1–22:36, http://dx.doi.org/10.1145/2333112.2333117.
[45] W. Dörfler, A convergent adaptive algorithm for Poisson's equation, SIAM J. Numer. Anal. 33 (3) (2006) 1106–1124, http://dx.doi.org/10.1137/0733054.
[46] J.S. Hale, L. Li, C.N. Richardson, G.N. Wells, Containers for portable, productive, and performant scientific computing, Comput. Sci. Eng. 19 (6) (2017) 40–50, http://dx.doi.org/10.1109/MCSE.2017.2421459.
[47] P. Wriggers, Nonlinear Finite Element Methods, Springer Science & Business Media, 2008.
[48] M.S. Alnæs, A. Logg, K.B. Ølgaard, M.E. Rognes, G.N. Wells, Unified form language: A domain-specific language for weak formulations of partial differential equations, ACM Trans. Math. Software 40 (2) (2014) 9:1–9:37, http://dx.doi.org/10.1145/2566630.
[49] C. Gräßle, M. Hinze, POD reduced-order modeling for evolution equations utilizing arbitrary finite element discretizations, Adv. Comput. Math. 44 (6) (2018) 1941–1978, http://dx.doi.org/10.1007/s10444-018-9620-x.
[50] J.S. Hale, E. Schenone, D. Baroli, L.A.A. Beex, S.P.A. Bordas, An implementation of the reduced assembly proper orthogonal decomposition method, 2018, http://dx.doi.org/10.6084/m9.figshare.5753292.
[51] S. Varrette, P. Bouvry, H. Cartiaux, F. Georgatos, Management of an academic HPC cluster: The UL experience, in: Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014), IEEE, Bologna, Italy, 2014, pp. 959–967, http://dx.doi.org/10.1109/HPCSim.2014.6903792.