Local Reduced Order Basis Interpolation Applied to the Parabolic Diffusion Equation with a Random Coefficient Field

Sachin Natesh

October 23, 2019

Abstract

In this work, a local reduced order basis (RB) interpolation method is described and explored numerically on a parameterized parabolic diffusion equation with a log-normal coefficient field. RBs and their corresponding reduced order models (ROMs) are often not robust with respect to changes in the parameter $\lambda \in \mathbb{R}^p$ to which the RB, generated by proper orthogonal decomposition (POD) of transient solution approximations, is associated. As such, interpolation of a set of RBs corresponding to a collection of parameters $\Lambda = \{\lambda_i\}_{i=1}^{N_R-1}$ may be a sensible approach to approximating the RB for parameter $\lambda_{N_R} \notin \Lambda$. However, straightforward interpolation of RBs does not guarantee that the interpolated vectors satisfy the properties of a RB, namely linear independence and orthogonality. The interpolation method considered here, and first introduced by [1], represents a RB generated by POD and associated to a parameter, or "operating point", as a point on the Grassmanian manifold. Representing RBs in this way offers a more amenable starting point to their interpolation, and results from differential geometry actually expose an algorithm to do so. In the remainder of this paper, the considered model problem is presented, its discretization reviewed, and the local RB interpolation algorithm is explicated and applied on the model problem. Lastly, an alternate formulation that overcomes the computational bottleneck inherent to the local RB interpolation method for problems with non-affine parameter dependence is described.

1 The Model Problem

The strong form of the parameterized parabolic, log-normal coefficient diffusion problem with stochastic parameter $\lambda \in \mathbb{R}$ is given by

$$u_t = \nabla \cdot (k(\mathbf{x};\lambda)\nabla u(\mathbf{x},t;\lambda)), \quad \mathbf{x} \in \Omega = (0,1)\times(0,1), \tag{1.1}$$

$$u(\mathbf{x},\mathbf{t};\lambda) = \mathbf{0}, \quad \mathbf{x} \in \Gamma_{\text{top}} = (0,1)\times 1, \tag{1.2}$$

$$\nabla u(\mathbf{x},t;\lambda)\cdot\vec{n} = 0, \quad \mathbf{x} \in \Gamma_{\text{side}} = \{0,1\}\times(0,1), \tag{1.3}$$

$$\nabla u(\mathbf{x},t;\lambda)\cdot\vec{n} = 1, \quad \mathbf{x} \in \Gamma_{\text{base}} = (0,1)\times\{0\}, \tag{1.4}$$

where ∇ is the spatial gradient and $t \in [0,T]$.

The requirements of the strong solution, namely existence in $C^2(\Omega)$, are relaxed by considering a weak formulation. Let $\mathbb{V} = \left\{v \in H^1(\Omega) \quad | \quad v|_{\Gamma_{\text{top}}=0}\right\}$, where $H^1(\Omega)$ is the Sobolev space of square-integrable functions with likewise defined first derivatives. Suppose $v \in \mathbb{V}$ and consider the

$L^2(\Omega)$ inner product of v with the terms of (1.1). Let $\Gamma = \Gamma_{\text{top}} \cup \Gamma_{\text{side}} \cup \Gamma_{\text{base}}$. Using a sort of "reverse" version of the integration by parts formula for $u, v \in C^1(\bar{\Omega})$ yields

$$\begin{aligned}\int_\Omega v\nabla(k\nabla u)d\mathbf{x} &= \int_\Gamma v(k\nabla u)\cdot\vec{n}d\mathbf{x} - \int_\Omega k\nabla u\cdot\nabla v d\mathbf{x}\\ &= \int_{\Gamma_{\text{base}}} kv d\mathbf{x} - \int_\Omega k\nabla u\cdot\nabla v d\mathbf{x} = \int_\Omega v u_t d\mathbf{x},\end{aligned}$$

where the other boundary terms are eliminated by applying the essential Dirichlet BCs on v and the natural Neumann BCs (1.3) on u. Thus, the weak form of (1.1) given the BCs is

$$\begin{gathered}\text{For some } \lambda, \text{ find } u(\mathbf{x},t;\lambda)\in\mathbb{V} \text{ s.t.}\\ \big(u_t(\mathbf{x},t;\lambda),v\big)_{L^2(\Omega)} + a\big(u(\mathbf{x},t;\lambda),v;\lambda\big) = f(v),\end{gathered} \tag{1.5}$$

subject to initial condition $u(\mathbf{x},0;\lambda) = u_0 \in L^2(\Omega)$ and where

$$\begin{aligned}a(u,v;\lambda) &= \int_\Omega k\nabla u\cdot\nabla v d\mathbf{x},\\ f(v) &= \int_{\Gamma_{\text{base}}} kv d\mathbf{x}.\end{aligned} \tag{1.6}$$

Additionally, the quantity of interest (QoI) evaluated $\forall t\in[0,T]$ is compliant (identical to the load functional) and given by

$$Q(u,t;\lambda) = f(u(t;\lambda)). \tag{1.7}$$

If the coefficient field on Ω satisfies Assumption 9.5 of [2, Chapter 9.1] and the linear operator satisfies Assumption 3.19 of [2, Chapter 3.2], then the problem is well posed. In this case, the linear operator is the laplacian, which is known to be negative definite, and the coefficient field is log-normal. That is,

$$k(\mathbf{x};\lambda) = e^{z(\mathbf{x})}, \tag{1.8}$$

where $z(\mathbf{x})$ is a mean-zero Gaussian random field with Gaussian isotropic covariance (i.e square exponential)

$$c(\mathbf{x},\mathbf{y}) = \sigma^2 e^{-\frac{||\mathbf{x}-\mathbf{y}||_2^2}{2l^2}}, \tag{1.9}$$

and thus, by Theorem 9.7 of [2, Chapter 9.1], the problem is well posed.

1.1 Discretization of the Model Problem with Rectangular Finite Elements

Here, the discretization procedure for numerically approximating solutions to the exact problem given by (1.6) and (1.7) is described. First, a conforming approximation space $\mathbb{V}_h \subset \mathbb{V}$ of dimension N_h is constructed:

$$\mathbb{V}_h = \text{span}\{\phi_{\mathbf{i}}\}_{\mathbf{i}=1}^{N_h}, \tag{1.10}$$

where $\phi_{\mathbf{i}}(x, y) \in \mathbb{R}$ are piecewise linear basis functions supported on $[x_i - h, x_i + h] \times [y_i - h, y_i + h]$ and h is the characteristic (here, uniform) mesh width in either direction. Explicitly, the basis functions are

$$\phi_{\mathbf{i}}(x, y) = \phi_{i_1}(x)\phi_{i_2}(y), \tag{1.11}$$

where the 1D functions are

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h}, & x \in I_i \\ \frac{x_{i+1} - x}{h}, & x \in I_{i+1} \\ 0, & \text{otherwise.} \end{cases} \tag{1.12}$$

Under this approximation space, a function $u(\mathbf{x}_i, t_k; \lambda)$ at point x_i and time $t_k = k\Delta t$ can be approximated as $u_h(\mathbf{x}_i, t_k; \lambda) = \sum_{i=1}^{N_h} (u_h^\lambda)_{i,k} \phi_i(\mathbf{x}_i)$. Substituting this expansion into (1.6) and (1.7) and discretizing in time with the backwards Euler scheme induces the following discrete system to solve for the coefficient vector $\mathbf{u}_{h,k}^\lambda$:

$$\mathbf{u}_{h,k}^\lambda = (M + \Delta t A)^{-1} M \mathbf{u}_{h,k-1}^\lambda + \Delta t (M + \Delta t A)^{-1} \mathbf{f}_h, \tag{1.13}$$

where M, the mass matrix, A the stiffness matrix, and $\mathbf{f}_h$ the load vector are given by

$$M_{ij} = (\phi_j, \phi_i)_{L^2(\Omega)}, \ A_{ij} = a(\phi_j, \phi_i; \lambda), \text{ and } (f_h)_i = f(\phi_i), \quad 1 \le i, j \le N_h. \tag{1.14}$$

The output functional is simply evaluated as

$$Q_h^k(\lambda) = (\mathbf{u}_{h,k}^\lambda)^T \mathbf{f}_h. \tag{1.15}$$

1.2 Sampling the Random Coefficient Field

As mentioned, the diffusion coefficient is a log-normal random field. In order to sample this field on the grid space, the Karhunen-Loève (KL) expansion is utilized. Specifically, a square exponential covariance kernel, given in (1.9) is evaluated at each point on the grid. Let the matrix K_{ij} represent this evaluation. Then, one sample λ, corresponding to one operating point, from the standard normal distribution is drawn and the KL expansion G of a Guassian random field on the grid is calculated in the following steps:

1. Find the eigenvalues and eigenfunctions of K via `MATLAB's eig` function, or with extra modification, `svd`.
2. Sort the eigenvalues and get the eigenfunctions in the same order. Let u_1 be the first eigenfunction and v_1 be the corresponding eigenvalue.
3. Compute the KL expansion as $G = u_1 v_1 \lambda$.

Figure 1 depicts the 1st, 10th, 30th and 40th eigenfunctions of the covariance kernel K where the scaling factor $\sigma = 2$ and the correlation length $\rho = 1$.

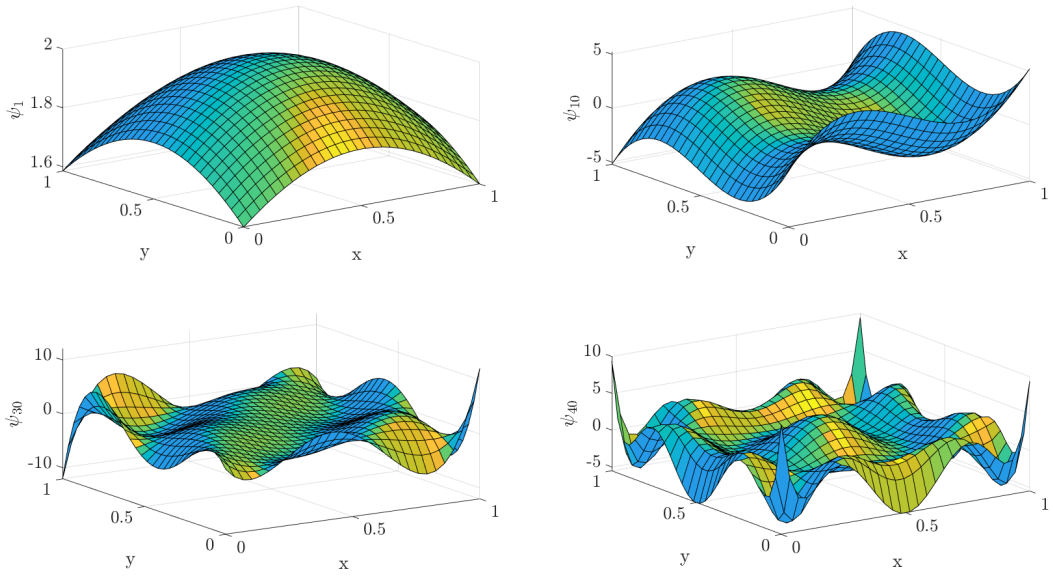

Figure 1: Example eigenfunctions of square exponential covariance kernel. As the mode increases, the eigenfunctions become more oscillatory. The kernel was evaluated on a grid with 900x900 points and $\rho = 1$, $\sigma = 2$.

Given G, the log-normal coefficient field is computed as $k(\lambda) = \exp(G)$. Figure 2 depicts an example KL expansion of a gaussian random field truncated after 1 term, and the corresponding log-normal field.

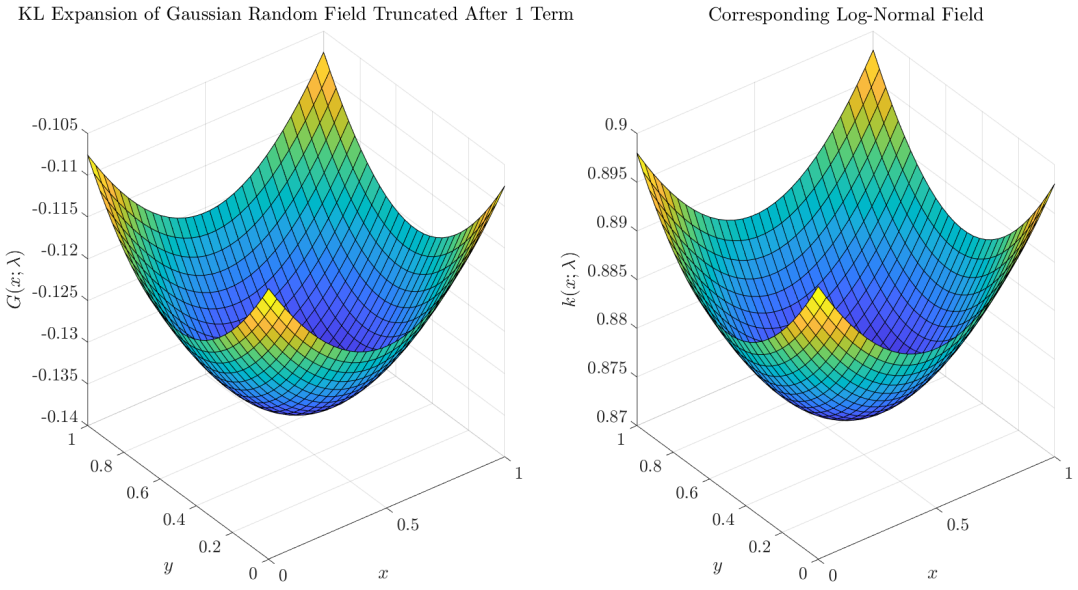

Figure 2: Example KL-expansion of gaussian random field and corresponding log-normal field. The square exponential kernel was evaluated on a grid with 900x900 points and $\rho = 1$, $\sigma = 2$ and $G(x;\lambda)$ was calculated as in step 3 in the description above.

2 Local Basis Interpolation

Rather than expounding the theoretical details of the general RB formulation via proper orthogonal decomposition for the time-dependent diffusion equation, it will presented heuristically and algorithmically. Let $U \in \mathbb{R}^{N_f} \times \mathbb{R}^{N_t}$ denote the "snapshot" matrix of solutions given by (1.13) at each of the N_t time points in the set $\{t_0 = 0, \cdots, t_{N_t} = T\}$ on a grid with N_f points and for a fixed parameter λ. That is, column k of U corresponds to the solution field at time $t = k\Delta t$. Of interest are the dominant modes in the time evolution of the solution; however, as is, the columns of U are correlated. One approach to discovering dominant modes is to perform a rank revealing transformation, namely the singular value decomposition (SVD). The POD basis of U is computed via a thin/economy-sized SVD, and this also "decorrelates" the data. That is,

$$U = \Phi \Sigma V^T, \tag{2.1}$$

and the unitary matrix Φ represents the reduced solution basis. If the singular values of U on the diagonal of S decay sufficiently rapidly, one can discard the basis vectors corresponding to small singular values, i.e. rank truncate, and approximate U to nearly full-fidelity, but with far less computational expense. For example, Figure 2 depicts the fractional singular value spectrum of a snapshot matrix.

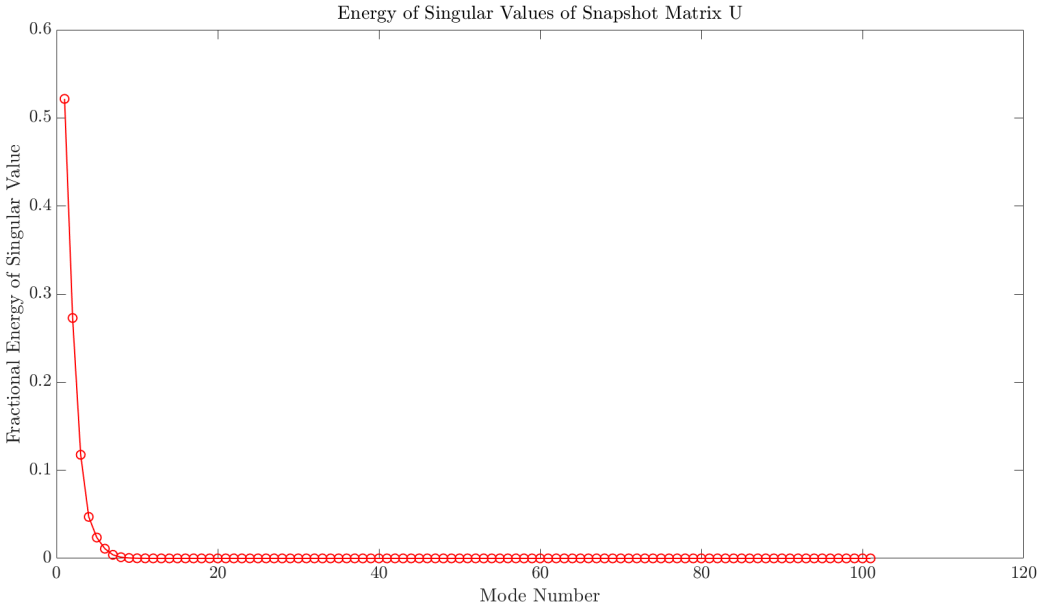

Figure 3: Singular value spectrum of snapshot matrix U, normalized by the sum of singular values. The solution to (1.6) was approximated from $t = 0$ to $t = 1$ with a time step of $\Delta t = 0.01$ starting from normally distributed random initial data, and the solution at each time was collected into U.

Note how the first 5 modes contain over 95% of the total "energy" content. In this case, one can certainly truncate all but the first 5 columns of Φ and expect to recover U to sufficient accuracy.

Assuming the determination has been made that rank truncation is a suitable approach for constructing a RB for a given problem and for several parameters λ, and provided that a RB has been generated by including information from each "local" RB associated to each parameter, the ROM can be constructed by evaluating the reduced solution operators, load vector and initial condition

$$M_{\text{rb}} = \Phi^T M \Phi, \quad A_{\text{rb}} = \Phi^T A \Phi, \quad \mathbf{f}_{\text{rb}} = \Phi^T \mathbf{f}_h, \quad \mathbf{u}_{0_{\text{rb}}} = \Phi^T \mathbf{u}_0, \tag{2.2}$$

and stepped forward in time using the scheme given in (1.13), where Φ is a global RB of sorts. One approach to constructing such a global RB over a range of parameters is to concatenate local bases generated via POD of the snapshot matrix for each parameter into one. However, this may result in a rank-deficient global basis since the local bases may have components in common with each other [3].

The approach introduced in [1] and taken here avoids constructing a global basis. Instead, an attempt is made to construct an interpolant out of the local RBs associated to a collection of parameters $\Lambda = \{\lambda_i\}_{i=1}^{N_R - 1}$ and use this interpolant to evaluate RBs for parameters $\lambda_{N_R} \notin \Lambda$. Then, the reduced system for the new operating point can similarly be evaluated using (2.2) and stepped forward in time using (1.13).

The starting point is recognizing that the columns of a RB, $\Phi \in \mathbb{R}^{N_f \times N_\Phi}$, generated by POD and associated to an operating point, span a subspace $S \in \mathcal{G}(N_\Phi, N_f)$, where

$$\mathcal{G}(N_\Phi, N_f) = \left\{ S \quad | \quad S \subset \mathbb{R}^{N_f},\ \dim(S) = N_\Phi \right\} \tag{2.3}$$

is the Grassmanian manifold, N_Φ is the dimension of the RB and N_f is the spatial dimension of the truth solution. In order to interpolate the RBs to a new operating point, they are first projected to a flat space where interpolating and projecting back to $G(N_\Phi, N_f)$ ensures the columns of the resulting matrix are linearly independent and orthogonal. Specifically, this flat space is that tangent to a given point $S \in G(N_\Phi, N_f)$. The explicit formula for projecting points S_i (represented by matrices Φ_i) onto the tangent space $\mathcal{T}_{S_1}$ of point S_1 (represented by matrix Φ_1) yielding $\Gamma_i \in \mathcal{T}_S$ is given by the logarithm map, which in matrix form is

$$\begin{aligned} \left[(\Phi_i^T \Phi_1)^{-1} (\Phi_i^T - \Phi_i^T \Phi_1 \Phi_1^T) \right]^T &= U_i \Sigma_i V_i^T, \\ \Gamma_i &= U_i \tan^{-1}(\Sigma_i) V_i^T, \end{aligned} \tag{2.4}$$

where the first equality represents an economy-sized SVD. The inverse, exponential map, which is used to project a point $\Gamma_j \in \mathcal{T}_S$ onto $G(N_\Phi, N_f)$ is given in matrix form by

$$\begin{aligned} \Gamma_j &= U_j \Sigma_j V_j^T, \\ \Phi_j &= \Phi V_j \cos(\Sigma_j) + U_j \sin(\Sigma_j), \end{aligned} \tag{2.5}$$

where the second equality represents an economy-sized SVD. An important remark is that the logarithm map is defined in a neighborhood of the point S_1. This implies that the map is not necessarily valid for S_i that is "far" away from S_1, though the notion of large distance is likely problem dependent. Nevertheless, a distance metric is required to determine which points in a set $\{S_i\}_i$ are projected onto $\mathcal{T}_{S_1}$ for the purpose of interpolation. This distance metric is given in [4] in matrix form as

$$\begin{aligned} \Phi_1^T \Phi_i &= U \Sigma V^T, \\ \Theta &= \cos^{-1}(\Sigma), \\ d(\Phi_1, \Phi_i) &= ||\Theta||_2, \end{aligned} \tag{2.6}$$

where, again, the first equality represents an economy-sized SVD. Also, it should be noted that trigonometric operations act only on the diagonals of the arguments. That is, the zeros of the singular value matrices are preserved.

2.1 Implementation of the Local Basis Interpolation Algorithm

Given the projection utilities developed in Section 2, the interpolation algorithm in the context of constructing a ROM for (1.13) can be fully specified. As an addendum, this procedure is meant more to test the accuracy of the interpolation scheme for this specific problem.

Offline Phase: Precomputing operators and building a database of RBs.

1. Precompute the mass matrix M given in (1.14). Since this operator is parameter independent, this only has to be done once.
2. Precompute N_{op} operators A and load vector $\mathbf{f}$ given in (1.14), one for each standard normal parameter $\{\lambda_i\}_{i=1}^{N_{\text{op}}}$. This step is very expensive.
3. Approximate $N_r < N_{\text{op}}$ solutions to (1.6) by way of (1.13) for some prescribed initial data. Each solution corresponds to different parameters λ_i, and the solution at each time $t = k\Delta t$ is stored. That is, compute N_r snapshot matrices.
4. For each solution, compute a solution basis by way of POD on the snapshot matrix.
5. For each solution basis (or some of them), inspect their spectrums to determine the maximum number of columns to discard while still representing a significant fraction of the energy. Store the resulting RBs. Each has dimension N_Φ.
6. Select one of the RBs, Φ_1, as the origin point $S_1 \in G(N_\Phi, N_f)$. Interpolation will occur on its tangent space $\mathcal{T}_{S_1}$.
7. Find N_{interp} nearest neighbors to S_1 from the set of RBs using (2.6). These neighbors and S_1 constitute the training set.
8. Project the RBs in the training set onto the tangent space of S_1 using the logarithm map given in (2.4).

Online Phase: Evaluate RBs at new operating points $\lambda \notin \{\lambda_i\}_{i=1}^{N_r}$.

1. Pick a new operating point $\lambda_{N_r+1} \notin \{\lambda_i\}_{i=1}^{N_r}$.
2. Interpolate (linearly in this case) in the tangent space of S_1 to the new operating point. That is, approximate a given entry of Γ_{N_r+1} using an interpolant constructed from the corresponding entries in $\{\Gamma_i\}_{i=1}^{N_{\text{interp}}}$. Note, it should be ensured the new operating point is within the range of those in the training set so that extrapolation doesn't occur.
3. Use the exponential map (2.5) to project Γ_{N_r+1} onto $G(N_\Phi, N_f)$ to attain Φ_{N_r+1}.
4. Project the full-order model operators (which ideally are precomputed for the new operating point) onto Φ_{N_r+1} using (2.2) to attain the reduced order operators.
5. Solve the ROM with backward Euler and the reduced order operators.
6. Evaluate the QoI.

3 Numerical Experiments

To assess the efficacy of the local basis interpolation scheme, a standard cross-validation procedure was employed. In particular, $N_{\text{op}} = 200$ operators A_λ corresponding to the same number of 1D stochastic parameters $\lambda_i \sim N(0,1)$ were precomputed at a mesh level of 5, and as well a single mass matrix M at the same level. Then, for each of 20 validation cases, a random permuted indexing of $[1, \cdots, N_{\text{op}}]$ was generated. Of the random N_{op} indices, $N_r = 50$ were selected and an $N(0,1)$ initial condition u_0 was supplied to (1.13) to step forward the solution from $t = 0$ to $t = 1$ with a timestep of $\Delta t = 0.01$ for each of the N_r operators A_λ. RBs were computed for each of the N_r solutions via POD with $N_\Phi = 1, \cdots, 10$ basis vectors. Then, 10 new, distinct stochastic parameters were selected using the initial randomly permuted index to serve as test operating points. The ROMs were computed for each of these test operating points through the online procedure described in Section 2.1 and stepped forward with same scheme given in (1.13). Lastly, the QoI evaluated through the ROM was compared for each time with that from the truth model and the errors and speedups for the different number of basis inclusions N_Φ were averaged over the test points and number of validations.

Figure 4 illustrates the solution trajectory at 4 points in time given random initial data and one of the stochastic parameters λ. The solution relaxes towards a steady state solution relatively quickly in this case. Figure 5 compares an instance of the QoI evaluation from the FOM to that from the ROM over time. Figure 6 depicts the average cross validated error in the 2-norm of the QoI evaluation over time as a function of number of basis modes included in the RBs. As expected, including more modes decreases the error, in general. Figure 7 shows the average speedup obtained by evaluating the ROM over the full order model (FOM) as a function of the RB dimension, excluding the time required to construct the operators (which were all precomputed). The general downward trend is expected, though significant gains in efficiency are incurred for RBs of all dimensions considered. Due to time restrictions, scaling with respect to mesh level was not investigated, but it would provide a clearer picture of just how useful the interpolation scheme is for this particular problem with non-affine parameter dependence.

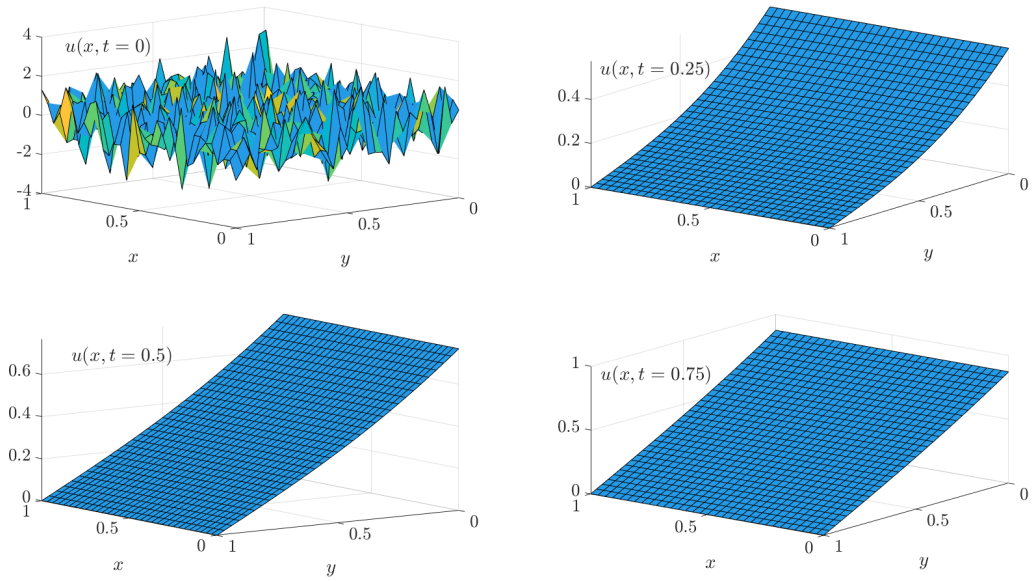

Figure 4: Sample solution trajectory for $t = 0, 0.25, 0.5, 0.75$ and random initial data.

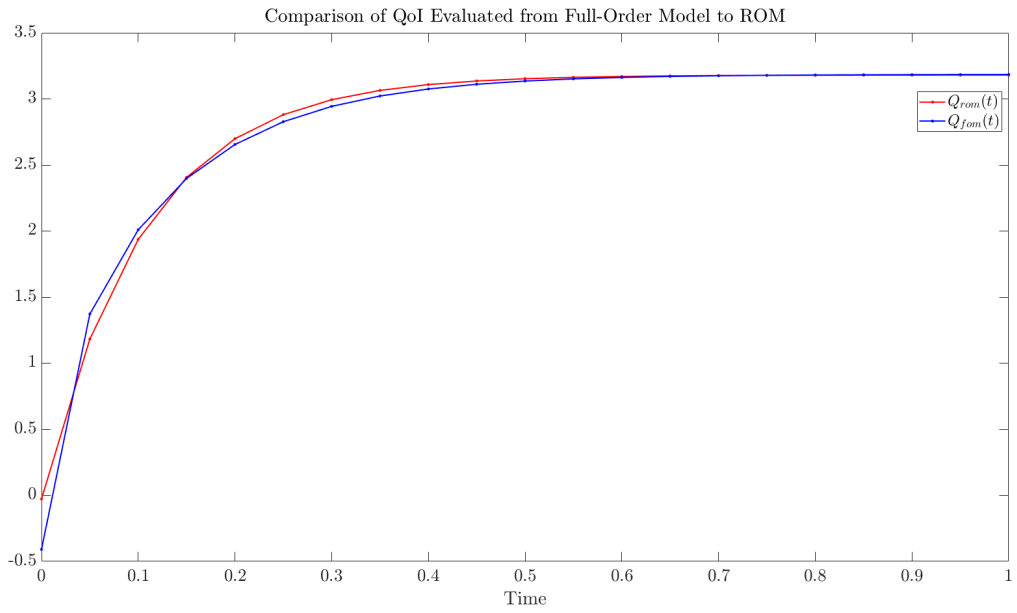

Figure 5: Comparison of the ROM with the FOM in evaluating the QoI over time.

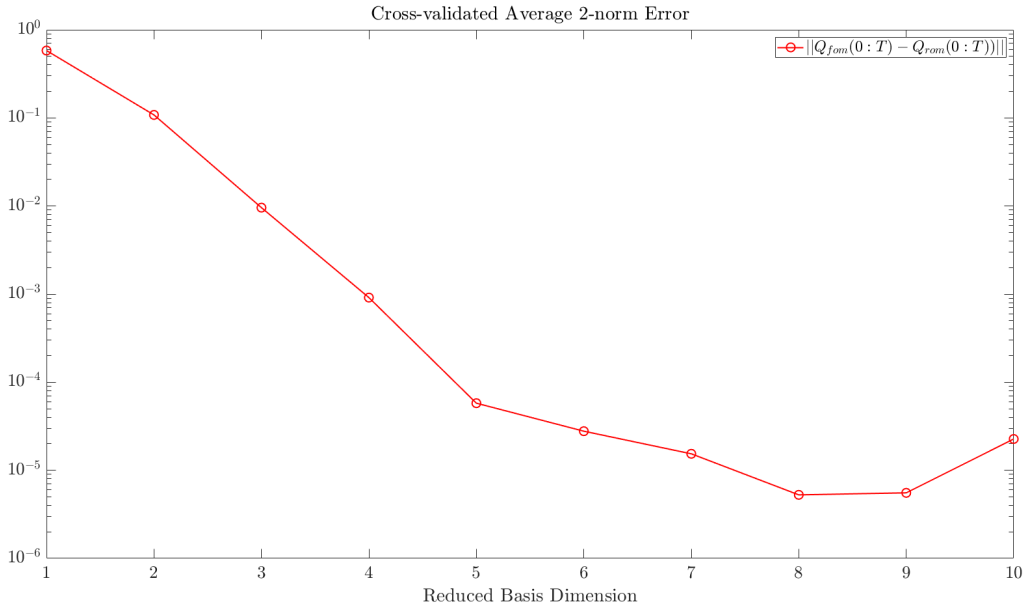

Figure 6: Cross validated average error in the 2-norm of the QoI evaluation from the ROM and the FOM over time.

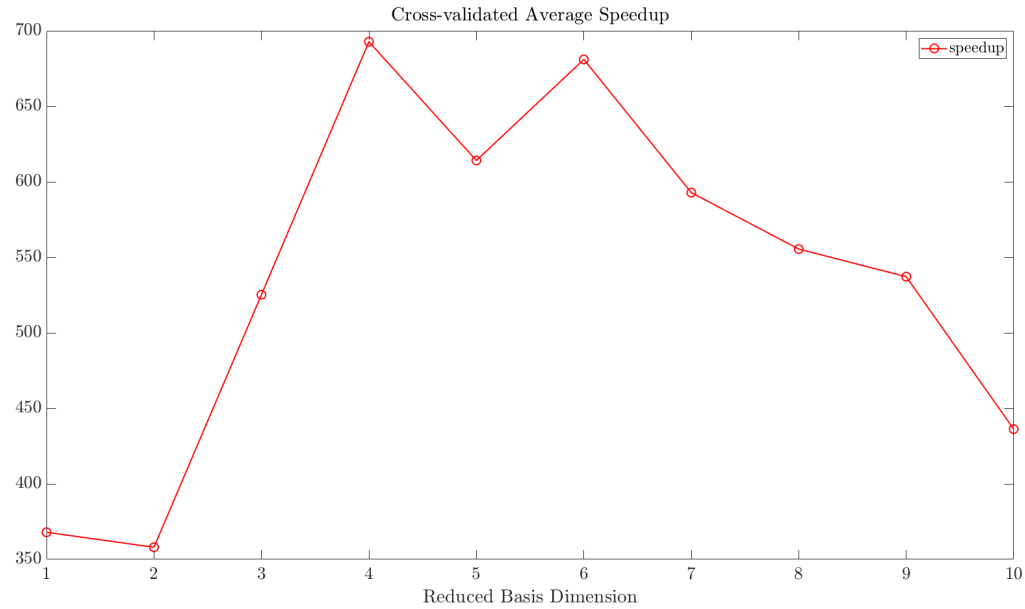

Figure 7: Average cross validated speedup.

4 Disadvantages of Local Basis Interpolation and an Alternative Formulation

In the cross-validation testing, all of the parameter dependent system matrices and load vectors in both the training and test sets were precomputed, while in reality, this would not, in general, be the case. Unfortunately, the approach examined in this work is still burdened in the online phase by the inefficiencies incurred from violating the affine parameter dependence of the bilinear form $a(u, v; \lambda)$ given in (1.6), as evinced by step 4 in the online phase algorithm. That is, the bilinear form cannot be decomposed as

$$a(u, v; \lambda) = \sum_{q=1}^{Q_a} \theta_a^q(\lambda) a_q(u, v). \tag{4.1}$$

For example, suppose the coefficient field $k(\mathbf{x}; \lambda)$ was not log-normal, but rather piecewise constant in "blocks" of the domain Ω. One can efficiently construct the truth matrix A by precomputing it without the coefficient contributions and constructing it for a given parameter by summing up the multiplications of the coefficient on the corresponding block of the operator.

In the log-normal case, however, for each new operating point, one has to compute inner products including the nowhere-constant, spatially varying coefficient field to construct A. This construction can add significant computational overhead to evaluating the ROM, and may diminish any of the speedup over the full order model in real settings.

The authors of [5] present one approach to overcome the issue of non-affine parameter dependence. A matrix is generated by collecting $N_s = N_t N_r$ snapshots of the solution to a transient problem over N_t time steps and for N_r parameters. Note, this is a single snapshot matrix X containing the time evolution of the full-order model for each parameter, as opposed to several snapshot matrices for each parameter. Then, the POD of X is computed to generate a basis Φ of dimension N_Φ, corresponding to the N_Φ largest singular values of X. A ROM is then evaluated using the same projection approach given in (2.2) for each parameter, though with the option of using a different left projection matrix than Φ itself. Markedly, the same projection matrices are used to reduce the dimensionality of the full-order system matrices for each parameter. Then, the reduced system matrices themselves are interpolated to compute that for a new operating point. Similar to the local basis interpolation on the tangent space of a point on the Grassmanian, they use logarithm and exponential maps to project the ROMs onto tangent spaces, interpolate and map back to attain the ROM for a new parameter. While the mappings are similar in concept, they are different in form since the ROM system matrices belong to a different Riemannian manifold than the Grassmanian to which local bases belong. In any case, they manage to avoid querying the full-order system matrices once the initial set of ROMs are generated, thus eliminating the bottleneck inherent to local basis interpolation.

References

[1] D. Amsallem and C. Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *Aiaa Journal*, 2008.

[2] L. Gabriel, C.E. Powell, and Tony Shardlow. *An Introduction to Computational Stochastic PDEs*. Cambridge University Press, 2014.

[3] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 2015.

[4] E. Begelfor and M. Werman. Affine invariance revisited. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[5] J. Degroote, J. Vierendeels, and K. Willcox. Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis. *International Journal for Numerical Methods in Fluids*, 2009.