

Adaptive h -refinement for reduced-order models

Kevin Carlberg

*Harry S. Truman Fellow, Quantitative Modeling & Analysis Department
Sandia National Laboratories, P.O. Box 969, MS 9159, Livermore, CA 94551, USA.*

SUMMARY

This work presents a method to adaptively refine reduced-order models *a posteriori* without requiring additional full-order-model solves. The technique is analogous to mesh-adaptive h -refinement: it enriches the reduced-basis space by ‘splitting’ selected basis vectors into several vectors with disjoint support. The splitting scheme is defined by a tree structure constructed via recursive k -means clustering of the state variables using snapshot data. The method identifies the vectors to split using a dual-weighted residual approach that seeks to reduce error in an output quantity of interest. The resulting method generates a hierarchy of subspaces online without requiring large-scale operations or high-fidelity solves. Further, it enables the reduced-order model to satisfy *any prescribed error tolerance* online regardless of its original fidelity, as a completely refined reduced-order model is equivalent to the original full-order model. Experiments on a parameterized inviscid Burgers equation highlight the ability of the method to capture phenomena (e.g., moving shocks) not contained in the span of the original reduced basis.

KEY WORDS: adaptive refinement, h -refinement, model reduction, dual-weighted residual, adjoint error estimation, clustering

1. Introduction

Modeling and simulation of parameterized systems has become an essential tool across a wide range of industries. However, the high computational cost of executing high-fidelity large-scale simulations is infeasibly high for many time-critical applications. In particular, many-query scenarios (e.g., sampling for statistical inversion) can require thousands of simulations of the system for various input-parameter instances; real-time contexts (e.g., model predictive control) require simulations to execute in mere seconds.

Reduced-order models (ROMs) have been developed to mitigate this computational bottleneck. First, they execute an ‘offline’ stage during which computationally expensive training tasks (e.g., evaluating the high-fidelity model for several instances of the input parameters) compute a representative low-dimensional reduced basis for the system state. Then, during the inexpensive ‘online’, these methods quickly compute approximate solutions for arbitrary input values via a projection process of the high-fidelity full-order-model (FOM) equations onto the low-dimensional subspace spanned by the reduced basis. They also introduce

other approximations in the presence of nonlinearities. See Ref. [1] and references within for a survey of current methods.

While reduced-order models almost always generate *fast* online predictions, there is no guarantee that they will generate sufficiently *accurate* online predictions. In fact, the accuracy of online predictions is predicated on the relevance of the training to the online problem: if the offline stage failed to capture a physical phenomenon, then this feature will be missing from online predictions. In general, the most one can hope for is that the ROM solution error is bounded by a prescribed scalar over a finite set of ‘training points’ in the input-parameter space [2]. The utility of this result is limited by the lack of applicability to online points outside this training set, and by the fact that such bounds often have unknown effectivity: satisfying the bound does not necessarily imply the true solution error is below a desired threshold.

This lack of error control precludes ROMs from being employed in many contexts. For example, PDE-constrained optimization requires the solution to satisfy a prescribed forcing sequence to guarantee convergence [3]. In uncertainty quantification, if the epistemic uncertainty due to the ROM solution error dominates other sources of uncertainty, the ROM cannot be incorporated in a useful manner. When simulating parameterized highly nonlinear dynamical systems, it is unlikely that any amount of training will fully encapsulate the range of complex phenomena that can be encountered online; such problems require an efficient refinement mechanism to generate accurate ROM predictions.

A few methods exist to improve a ROM solution when it is detected to be inaccurate; however, they entail *large-scale* operation counts. The most common approach is to revert to the high-fidelity model, solve the associated high-dimensional equations for the current time step or optimization iteration, add the solution to the reduced basis, and proceed with the enriched reduced-order model [4, 5, 6]. Another approach employs the reduced-order model as a preconditioner for the full-order model [7]; here, the reduced-order model serves to accelerate the full-order solve to any specified tolerance. As our goal is to improve the reduced-order model efficiently, i.e., without incurring large-scale operations, none of these methods is appropriate.

Instead, this work proposes a novel approach inspired by mesh-adaptive *h*-refinement. The main idea is to adaptively refine an inaccurate ROM by ‘splitting’ selected reduced basis vectors into multiple vectors with disjoint discrete support. This splitting technique is defined by a tree structure generated by applying *k*-means clustering to the state variables. The method uses a dual-weighted residual approach to select vectors to split. The resulting method generates a hierarchy of subspaces online without requiring any large-scale operations or high-fidelity solves. Most importantly, the methodology allows the ROM to satisfy *any prescribed error tolerance* online, as a fully refined ROM is equal to the original full-order model.

As a final note, some adaptive methods exist to tailor the ROM to specific regions of the input space [8, 9, 10, 11, 12, 13, 14], time domain [13, 15], or state space [16, 17, 14]. However, these methods are *a priori* adaptive: they construct separate ROMs for each region with the goal of reducing the ROM dimension. Critically, they have no mechanism to improve the ROM *a posteriori* if it delivers a solution with insufficiently accuracy.

In the remainder of this paper, matrices are denoted by capitalized bold letters, vectors by lowercase bold letters, scalars by lowercase letters, and sets by capitalized letters. The columns of a matrix $\mathbf{A} \in \mathbb{R}^{m \times k}$ are denoted by $\mathbf{a}_i \in \mathbb{R}^m$, $i \in \mathbb{N}(k)$ with $\mathbb{N}(a) := \{1, \dots, a\}$ such that $\mathbf{A} := [\mathbf{a}_1 \ \dots \ \mathbf{a}_k]$. The scalar-valued matrix elements are denoted by $a_{ij} \in \mathbb{R}$ such that $\mathbf{a}_j := [a_{1j} \ \dots \ a_{mj}]^T$, $j \in \mathbb{N}(k)$.

2. Problem formulation

2.1. Full-order model

Consider solving a parameterized sequence of systems of equations

$$\tilde{\mathbf{r}}^k(\mathbf{x}^k; \boldsymbol{\mu}) = 0 \quad (1)$$

for $k \in \mathbb{N}(t)$, where $\mathbf{x}^k \in \mathbb{R}^n$ denotes the state at iteration k , $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^{n_\mu}$ denotes the system inputs (e.g., boundary conditions), and $\tilde{\mathbf{r}}^k : \mathbb{R}^n \times \mathbb{R}^{n_\mu} \rightarrow \mathbb{R}^n$ denotes the residual operator at iteration k . This formulation is quite general, as it is appropriate for parameterized systems of linear equations ($t = 1$, $\tilde{\mathbf{r}} : (\mathbf{x}; \boldsymbol{\mu}) \mapsto \mathbf{b}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{x}$) such as those arising from the finite-element discretization of elliptic PDEs, and parameterized ODEs $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}; \boldsymbol{\mu})$ after time discretization by an implicit linear multistep method (e.g., $\tilde{\mathbf{r}}^k : (\mathbf{x}^k; \boldsymbol{\mu}) \mapsto \mathbf{x}^k - \mathbf{x}^{k-1} - \Delta t \mathbf{f}(\mathbf{x}^k; \boldsymbol{\mu})$ for the backward Euler scheme) such as those arising from the space- and time-discretization of parabolic and hyperbolic PDEs. Assume that we are primarily interested in computing outputs

$$z^k = g(\mathbf{x}^k; \boldsymbol{\mu}) \quad (2)$$

for $z^k \in \mathbb{R}$ and $g : \mathbb{R}^n \times \mathbb{R}^{n_\mu} \rightarrow \mathbb{R}$.

When n is ‘large’, computing the outputs of interest z^k by first solving Eq. (1) and subsequently computing outputs via Eq. (2) can be prohibitively expensive. This is particularly true for many-query (e.g., Bayesian inference) and real-time problems (e.g., model-predictive control) that demand a fast evaluation of the input–output map $\boldsymbol{\mu} \mapsto z^k$.

2.2. Reduced-order model

Model-reduction techniques aim to reduce the burden of solving Eq. (1) by employing a projection process. First, they execute a computationally expensive offline stage (e.g., solving Eq. (1) for a training set $\mathcal{D}_{\text{train}} \subset \mathcal{D}$) to construct 1) a low-dimensional trial basis (in matrix form) $\mathbf{V} \in \mathbb{R}^{n \times p}$ with $p \ll n$ that (hopefully) captures the behavior of the state \mathbf{x} throughout the parameter domain \mathcal{D} , and 2) an associated test basis $\mathbf{W} \in \mathbb{R}^{n \times p}$. Then, during the computationally inexpensive online stage, these methods approximately solve Eq. (2) for arbitrary $\boldsymbol{\mu} \in \mathcal{D}$ by searching for solutions in the trial subspace range $(\mathbf{V}) \subset \mathbb{R}^n$ and enforcing the residual $\tilde{\mathbf{r}}^k$ to be orthogonal to the test subspace range $(\mathbf{W}) \subset \mathbb{R}^n$:

$$\mathbf{W}^T \tilde{\mathbf{r}}^k(\bar{\mathbf{x}} + \mathbf{V} \hat{\mathbf{x}}^k; \boldsymbol{\mu}) = 0. \quad (3)$$

When the residual operator is nonlinear in the state or non-affine in the inputs, additional complexity-reduction approximations such as empirical interpolation [18, 19], collocation [20, 21, 6], discrete empirical interpolation [22, 23, 24], and gappy proper orthogonal decomposition [25, 26], are required to ensure that computing the low-dimensional residual $\mathbf{W}^T \tilde{\mathbf{r}}^k$ incurs an n -independent operation count. For simplicity, we do not consider such approximations in the present work; future work will entail extending the proposed methods to such ‘hyper-reduced’ order models.

In many cases, the test basis can be expressed as $\mathbf{W} = \mathbf{A}^n(\mathbf{x}; \boldsymbol{\mu}) \mathbf{V}$. For example, $\mathbf{A}^n(\mathbf{x}; \boldsymbol{\mu}) = \mathbf{I}$ for Galerkin projection; balanced truncation uses $\mathbf{A}^n(\mathbf{x}; \boldsymbol{\mu}) = \mathbf{Q}$, where \mathbf{Q} is the observability Gramian of the linear time-invariant system; the least-squares Petrov–Galerkin projection [20, 25, 26] underlying the GNAT method employs $\mathbf{A}^n(\mathbf{x}; \boldsymbol{\mu}) =$

$\partial \tilde{\mathbf{r}}^k / \partial \mathbf{x}(\mathbf{x}, \boldsymbol{\mu})$. When this holds, the Petrov–Galerkin projection (3) is equivalent to a Galerkin projection performed on the modified residual $\mathbf{r}^k := \mathbf{A}^n(\mathbf{x}; \boldsymbol{\mu})^T \tilde{\mathbf{r}}^k$:

$$\mathbf{V}^T \mathbf{r}^k(\bar{\mathbf{x}} + \mathbf{V} \hat{\mathbf{x}}^k; \boldsymbol{\mu}) = 0. \quad (4)$$

In the remainder of this paper, Eq. (4) will be considered the governing equations for the reduced-order model.

2.3. Objective: adaptive refinement

The goal of this work is as follows: given a reduced basis \mathbf{V} and solution $\hat{\mathbf{x}}^k$ to Eq. (4), 1) detect if the solution is sufficiently accurate, 2) if not, efficiently generate a higher-dimensional reduced basis \mathbf{V}' in a goal-oriented manner that aims to reduce errors in the output z^k such that $\text{range}(\mathbf{V}) \subset \text{range}(\mathbf{V}')$, 3) compute an associated solution $\hat{\mathbf{x}}'^k$, 4) repeat until desired accuracy is reached.

To generate this hierarchy of subspaces efficiently, we propose an analogue to adaptive h -refinement, wherein the support (i.e., element set with nonzero entries) of selected basis vectors \mathbf{v}_i is ‘split’ into disjoint subsets of elements. Like all h -refinement techniques, the proposed method consists of the following components:

1. *Refinement mechanism.* In typical h -refinement, this is defined by the mesh-refinement method applied to finite elements or volumes. The proposed method refines the solution space by splitting the support of the basis vectors using a tree structure constructed via k -means clustering of the state variables. Section 3 describes this component.
2. *Error indicators.* Goal-oriented methods for h -refinement often 1) solve a coarse dual problem, 2) prolongate the adjoint solution to a representation on the fine grid, and 3) compute error estimates of the output using first-order analysis. The proposed method employs an analogous goal-oriented dual-weighted residual approach. Section 4 introduces this.
3. *An adaptive algorithm.* The proposed algorithm identifies when refinement is required and employs error indicators decide on the particular refinement, i.e., which basis vectors should be refined, and how they should be refined. Section 5 provides this algorithm.

Remark. If the reduced-order model (4) is *a priori* convergent in a certain metric, then the proposed strategy guarantees monotonic convergence in that metric. For example, consider the case where the residual is linear in the state and its Jacobian $\partial \mathbf{r}^k / \partial \mathbf{x}(\boldsymbol{\mu})$ is symmetric and positive definite.* Then, it can be shown that

$$\mathbf{V} \hat{\mathbf{x}}^k = \arg \min_{\mathbf{w} \in \text{range}(\mathbf{V})} \|\mathbf{x}^k - \bar{\mathbf{x}} - \mathbf{w}\|_{\partial \mathbf{r}^k / \partial \mathbf{x}(\boldsymbol{\mu})}, \quad (5)$$

The, when the enriched basis \mathbf{V}' satisfying $\text{range}(\mathbf{V}) \subset \text{range}(\mathbf{V}')$ is employed as the search space for (5), it is guaranteed that the resulting solution $\mathbf{V}' \hat{\mathbf{x}}'^k$ will yield a decrease in the objective function

$$\|\mathbf{x}^k - \bar{\mathbf{x}} - \mathbf{V}' \hat{\mathbf{x}}'^k\|_{\partial \mathbf{r}^k / \partial \mathbf{x}(\boldsymbol{\mu})} \leq \|\mathbf{x}^k - \bar{\mathbf{x}} - \mathbf{V} \hat{\mathbf{x}}^k\|_{\partial \mathbf{r}^k / \partial \mathbf{x}(\boldsymbol{\mu})}, \quad (6)$$

as the previous solution remains in the search space $\mathbf{V} \hat{\mathbf{x}}^k \in \text{range}(\mathbf{V}')$. ■

*This can be achieved, for example, by Galerkin projection applied to symmetric coercive elliptic PDEs [2] or by least-squares Petrov–Galerkin applied to a parametrized linear system of equations.

3. Refinement mechanism

3.1. Tree data structure

To begin, we define a tree data structure that characterizes the refinement hierarchy. The tree is characterized by a *child* function $C : \mathbb{N}(m) \rightarrow \mathcal{P}(\mathbb{N}(m))$ and an *element* function $E : \mathbb{N}(m) \rightarrow \mathcal{P}(\mathbb{N}(n))$, where m denotes the number of nodes, \mathcal{P} denotes the powerset. For all $i \in \mathbb{N}(m)$, we have

$$E(j) \cap E(k) = \emptyset, \quad \forall j \neq k \in C(i) \quad (7)$$

$$\bigcup_{j \in C(i)} E(j) = E(i). \quad (8)$$

Finally, we enforce the root node to include all elements $E(1) = \mathbb{N}(n)$, and the leaf nodes to include a single element each

$$\forall l \in \mathbb{N}(n), \exists i \in \mathbb{N}(m) \mid E(i) = l, C(i) = \emptyset. \quad (9)$$

Each basis vector is characterized by a particular node on the tree $d_i \in \mathbb{N}(m)$, $i \in \mathbb{N}(p)$. The set $E(d_i)$ corresponds to the support of the basis vector, i.e., set of nonzero entries. The set $C(d_i)$ corresponds to the set of children defining splits of the basis vector. Due to the requirement in Eq. (9), a completely split basis ($C(d_i) = \emptyset$, $i \in \mathbb{N}(p)$) is equivalent to the full-order model, as $\text{range}(\mathbf{V}) = \mathbb{R}^n$ in this case. This enables the reduced-order model to generate arbitrarily accurate solutions when equipped with the proposed refinement method.

Example. Consider an example with $n = 6$ and an initial reduced basis $\mathbf{V}^{(0)} = \mathbf{v}_1^{(0)}$ of dimension 1. Figure 1 depicts an example of a tree structure for this case. Imagine the basis

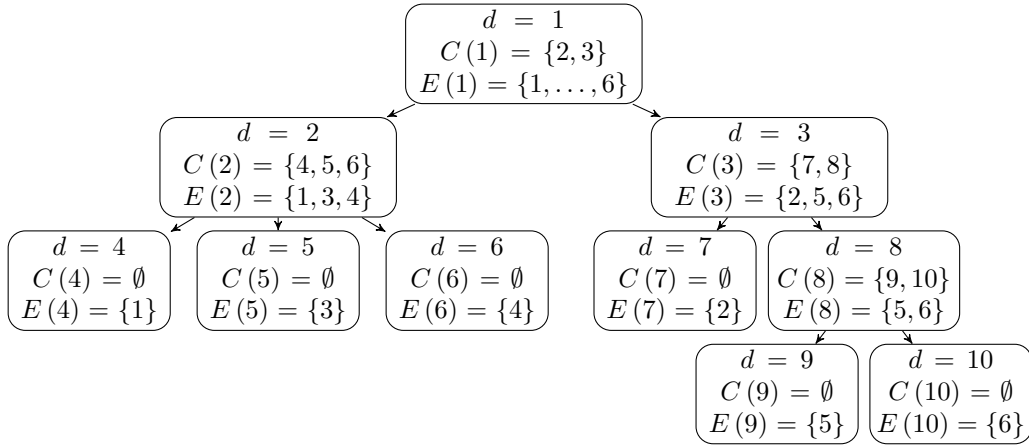


Figure 1. Tree example with $n = 6$

has been split into $p = 4$ according to the tree in Figure 1 with $d_1 = 2$, $d_2 = 7$, $d_3 = 9$, and

$d_4 = 10$; then, the reduced basis is

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_{11}^{(0)} & 0 & 0 & 0 \\ 0 & \mathbf{v}_{21}^{(0)} & 0 & 0 \\ \mathbf{v}_{31}^{(0)} & 0 & 0 & 0 \\ \mathbf{v}_{41}^{(0)} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{v}_{51}^{(0)} & 0 \\ 0 & 0 & 0 & \mathbf{v}_{61}^{(0)} \end{bmatrix}. \quad (10)$$

■

In the sequel, we define the number of total children at node j for tree i to be $q_i := \text{card}(C(d_i))$. We also overload the child function for the two-argument case such that $C(i, j)$ is the j th child node of parent node i , where ordering of the children is implied by the binary relation \leq on the natural numbers. Similarly, the overloaded element function $E(i, j)$ is the j th element for node i ; again, ordering of the elements is implied by the relation \leq on the natural numbers.

3.2. Refinement via basis splitting

We now put the basis-splitting methodology in the framework of typical h -refinement techniques. First, define a ‘coarse’ basis \mathbf{V}^H (initially equal to the nominal basis $\mathbf{V}^{(0)}$). Define also a ‘fine’ basis corresponding to the coarse basis with all vectors split according to the children of the current node. We can express the relationship between the coarse and fine bases as

$$\mathbf{V}^H = \mathbf{V}^h \mathbf{I}_H^h \quad (11)$$

where $\mathbf{V}^h \in \mathbb{R}^{n \times q}$ denotes the fine basis, $\mathbf{I}_H^h \in \{0, 1\}^{q \times p}$ denotes the *prolongation* operator, and q denotes the dimension of the fine reduced subspace. Clearly, $\text{range}(\mathbf{V}^H) \subset \text{range}(\mathbf{V}^h)$. Then, for any generalized coordinates $\hat{\mathbf{w}}^H \in \mathbb{R}^p$ associated with the coarse basis \mathbf{V}^H , we can compute the corresponding fine representation $\hat{\mathbf{w}}^h \in \mathbb{R}^q$ associated with the fine basis \mathbf{V}^h as

$$\hat{\mathbf{w}}_H^h = \mathbf{I}_H^h \hat{\mathbf{w}}^H, \quad (12)$$

which ensures that $\mathbf{V}^H \hat{\mathbf{w}}^H = \mathbf{V}^h \hat{\mathbf{w}}_H^h$. Note this prolongation operator is exact, unlike typical mesh-refinement strategies, where this operator is often defined as a linear or quadratic interpolant of the coarse solution on the fine grid. The *restriction* operator is not uniquely defined, but can be set, e.g., to

$$\mathbf{I}_h^H = (\mathbf{I}_H^h)^+. \quad (13)$$

Here, the superscript $+$ denotes the Moore–Penrose pseudoinverse.

Using the tree structure defined in Section 3.1, we can precisely define these quantities. We first introduce the mapping $f : (i, j) \mapsto k$, which provides the fine basis-vector index k corresponding to the j th child of the i th coarse basis vector. We define it as

$$f(i, j) = \sum_{k < i} q_k + j, \quad j \in \mathbb{N}(q_i), \quad i \in \mathbb{N}(p). \quad (14)$$

In particular, note that if node i in tree is a leaf (i.e., $q_i = 0$), then $f(i, j)$ does not exist for any j . Similarly, the inverse mapping $f^{-1} : k \mapsto (i, j)$ yields the coarse basis-vector index i and child index j corresponding to fine basis vector k .

Now, the dimension of the fine reduced subspace is $q = \sum_{i=1}^p q_i$, and the fine reduced basis is

$$v_{ij}^h = \begin{cases} v_{il}^H, & \exists k \mid j = f(l, k), i \in E(C(d_l, k)) \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

The prolongation operator induced by the proposed splitting scheme is

$$[\mathbf{I}_H^h]_{ij} = \begin{cases} 1, & \exists k \mid i = f(j, k) \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

3.3. k -means clustering of the state variables

This section presents an approach to construct a tree that satisfies the requirements outlined in Section 3.1. This approach uses the following heuristic:

State variables x_i that tend to be strongly positively or negatively correlated can be accurately represented by the same generalized coordinate, and should therefore reside in the same tree node.

Example. To justify this heuristic, consider an example with $n = 6$ degrees of freedom and $n_o = 8$ observations of the state, e.g., from a computed time history. Assume that snapshots can be decomposed as

$$\mathbf{X} = \sum_{i=1}^3 \mathbf{y}_i \mathbf{z}_i^T + 0.1 \mathbf{E} \quad (17)$$

where $\mathbf{E} \in [-1, 1]^{n \times n_o}$ is a matrix of random uniformly distributed noise and the data matrices are

$$\mathbf{Z} = \begin{bmatrix} -2.2083 & -5.1072 & 2.6816 & 9.3277 & -6.4506 & -3.2548 & 4.2237 & -3.2557 \\ -2.9810 & 0.6557 & 3.0474 & 5.5252 & 2.7674 & 2.3311 & 9.6190 & -6.6484 \\ -2.4547 & 5.2676 & -3.6434 & 5.5661 & -7.5449 & 9.3079 & -2.0459 & -0.0728 \end{bmatrix}^T$$

$$\mathbf{Y} = \begin{bmatrix} -3.9885 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8.6843 & 0 & 0 & -1.6393 \\ 0 & -1.7288 & 0 & 6.0559 & 2.2407 & 0 \end{bmatrix}^T.$$

The sparsity structure of \mathbf{Y} implies that following sets of state variables that are strongly correlated or anti-correlated across observations: $\{1\}$, $\{3, 6\}$, and $\{2, 4, 5\}$. This is apparent from computing the matrix of sample correlation coefficients:

$$\mathbf{R} = \begin{bmatrix} 1.0000 & 0.1526 & -0.5698 & -0.1534 & -0.1554 & 0.5705 \\ 0.1526 & 1.0000 & -0.0180 & -1.0000 & -1.0000 & 0.0198 \\ -0.5698 & -0.0180 & 1.0000 & 0.0209 & 0.0212 & -1.0000 \\ -0.1534 & -1.0000 & 0.0209 & 1.0000 & 1.0000 & -0.0227 \\ -0.1554 & -1.0000 & 0.0212 & 1.0000 & 1.0000 & -0.0229 \\ 0.5705 & 0.0198 & -1.0000 & -0.0227 & -0.0229 & 1.0000 \end{bmatrix}. \quad (18)$$

Imagine we start with one-dimensional reduced basis corresponding to the first left singular vector of \mathbf{X}

$$\mathbf{V}^H = \mathbf{v}_1^H = \begin{bmatrix} -0.2609 & -0.0348 & 0.9390 & 0.1240 & 0.0463 & -0.1773 \end{bmatrix}^T.$$

Because the data nearly lie in a three-dimensional subspace of \mathbb{R}^6 , the optimal performance of a refinement scheme would yield small error after splitting this one-dimensional basis into a basis of dimension three. Thus, consider splitting $\mathbf{V}^{(0)}$ into three children, i.e., the tree is characterized by $C(1) = \{2, 3, 4\}$, $E(2) = \{1\}$, $E(3) = \{3, 6\}$, and $E(4) = \{2, 4, 5\}$. The resulting basis becomes

$$\mathbf{V}^h = \begin{bmatrix} -0.2609 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9390 & 0 & 0 & -0.1773 \\ 0 & -0.0348 & 0 & 0.1240 & 0.0463 & 0 \end{bmatrix}^T.$$

The resulting projection error of the data is merely $\|\mathbf{X} - \mathbf{V}^h (\mathbf{V}^h)^+ \mathbf{X}\|_F / \|\mathbf{X}\|_F = 0.0033$. By contrast, generating an alternative three-dimensional fine basis $\bar{\mathbf{V}}^h$ by splitting the basis using a (similar) tree characterized by $E(2) = \{1\}$, $E(3) = \{3, 5\}$, $E(4) = \{2, 4, 6\}$, yields a much larger error of $\|\mathbf{X} - \bar{\mathbf{V}}^h (\bar{\mathbf{V}}^h)^+ \mathbf{X}\|_F / \|\mathbf{X}\|_F = 0.4948$.

One way to identify these correlated variables is to employ k -means clustering after pre-processing the data by 1) normalizing observations of each variable (to enable clustering to detect correlation), and 2) negating the observation vector over the origin if the first observation is negative (to enable clustering to detect anti-correlation). This is visualized in Figure 2 for this example. Note that correlated and anti-correlated variables have a small Euclidean distance between them after this processing; this allows k -means clustering to identify them as a group. ■

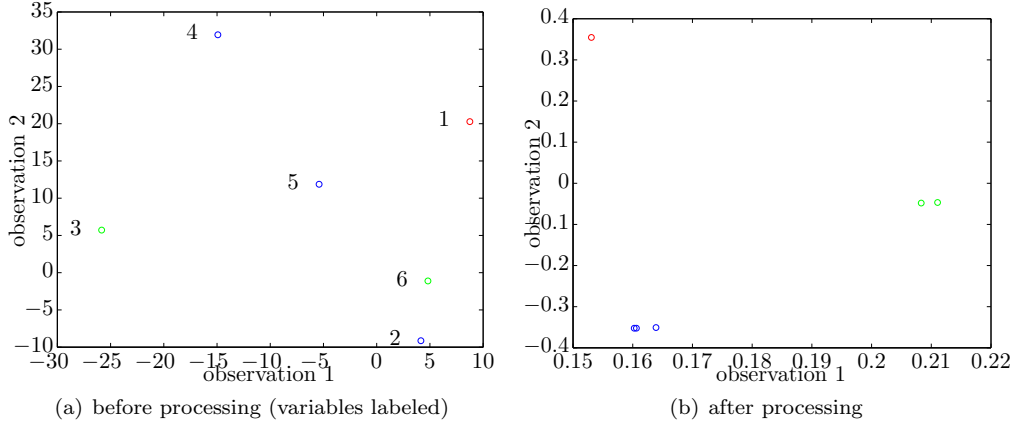


Figure 2. State-variable data projected onto the space of the first two observations for the example in Section 3.3. After processing the observations by normalization and origin flipping, correlated and anti-correlated variables can be easily grouped via clustering.

To this end, we define the tree by recursively applying k -means clustering to observations of the state variables (after reference subtraction, normalization, and origin flipping); Algorithm 1 describes the method. The n_o observations of these variables are obtained from snapshot data; such data are often available, e.g., when \mathbf{V} is constructed via proper orthogonal decomposition.

Algorithm 1 Tree construction via recursive k -means clustering

Input: n_o snapshots of the reference-centered[†] state in matrix form $\mathbf{X} \in \mathbb{R}^{n \times n_o}$, number of means \bar{k}

Output: child function C , element function E , and number of nodes m

```

1: for  $i = 1, \dots, n$  do
2:   Normalize rows of  $\mathbf{X}$  to capture correlation by clustering  $\mathbf{x}_i^T \leftarrow \mathbf{x}_i^T / \|\mathbf{x}_i^T\|$ 
3:   if  $x_{i1} < 0$  then {Flip over origin to capture negative correlation by clustering}
4:      $\mathbf{x}_i^T \leftarrow -\mathbf{x}_i^T$ 
5:   end if
6: end for
7: Set root node to contain all elements  $E(1) = \mathbb{N}(n)$ .
8: Initialize recent-node set  $D \leftarrow \{1\}$  and node count  $m \leftarrow 1$ .
9: while  $\text{card}(D) > 0$  do
10:   $\bar{D} \leftarrow D, D \leftarrow \emptyset$ 
11:  for  $i = 1, \dots, \text{card}(\bar{D})$  do
12:    Set splitting node to the  $i$ th element of the recent-node set  $d \leftarrow \bar{D}(i)$ , where ordering
    is implied by  $\geq$  on the natural numbers.
13:    if  $E(d) = \emptyset$  then {No elements to split}
14:      Continue
15:    end if
16:    Select snapshots of current elements  $\bar{x}_{jk} \leftarrow x_{E(d,j)k}, j \in \mathbb{N}(\text{card}(E(d))), k \in \mathbb{N}(n_o)$ 
17:     $(\bar{E}_1, \dots, \bar{E}_{n_c}) = \text{kmeans}(\bar{\mathbf{X}}, \bar{k})$ , where  $\bar{E}_j \subset \mathbb{N}(\text{card}(E(d)))$  denotes the set of
    elements in cluster  $j$ , and  $n_c$  denotes the number of non-empty clusters.
18:    if  $n_c = 1$  then {Cannot have only one child}
19:      for  $j = 1, \dots, \text{card}(E(d))$  do {Make all children into leaf nodes}
20:         $\bar{E}_j = j$ 
21:      end for
22:    end if
23:    for  $j = 1, \dots, n_c$  do
24:       $m \leftarrow m + 1$ 
25:       $D \leftarrow D \cup m$ 
26:       $E(m) = \{E(d, j) \mid j \in \bar{E}_j\}$ 
27:       $C(d, j) = m$ 
28:    end for
29:  end for
30: end while

```

4. Dual-weighted residual error indicators

To simplify notation in this section, we consider a single solve, i.e., Eq. (4) for a single time and parameter instance, and set $\bar{\mathbf{x}} = 0$. Then, the ROM governing equations can be represented

[†]This implies that the reference state $\bar{\mathbf{x}}$ should be subtracted from the state snapshots.

simply as

$$\mathbf{V}^T \mathbf{r}(\mathbf{V} \hat{\mathbf{x}}) = 0. \quad (19)$$

To compute error indicators for refinement, we propose a goal-oriented dual-weighted residual methodology based on adjoint solves. It can be considered an model-reduction adaptation of duality-based error-control methods developed for differential equations [27, 28], finite-element discretizations [29, 30, 31, 32, 33, 34], finite-volume discretizations [35, 36, 37, 38], and discontinuous Galerkin discretizations [39, 40]. First, we approximate the output due to the (unknown) fine solution $\hat{\mathbf{x}}^h$ to first-order about the coarse solution $\hat{\mathbf{x}}^H$:

$$g(\mathbf{V}^h \hat{\mathbf{x}}^h) \approx g(\mathbf{V}^H \hat{\mathbf{x}}^H) + \frac{\partial g}{\partial \mathbf{x}}(\mathbf{V}^h \hat{\mathbf{x}}^h) \mathbf{V}^h (\hat{\mathbf{x}}^h - \mathbf{I}_H^h \hat{\mathbf{x}}^H), \quad (20)$$

where we have used Eq. (11) to relate the coarse and fine bases.

Similarly, we can approximate the fine residual to first order about the coarse solution as

$$0 = (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^h \hat{\mathbf{x}}^h) \approx (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H) + (\mathbf{V}^h)^T \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H) \mathbf{V}^h (\hat{\mathbf{x}}^h - \mathbf{I}_H^h \hat{\mathbf{x}}^H). \quad (21)$$

Solving for the error yields

$$(\hat{\mathbf{x}}^h - \mathbf{I}_H^h \hat{\mathbf{x}}^H) \approx - \left[(\mathbf{V}^h)^T \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H) \mathbf{V}^h \right]^{-1} (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H) \quad (22)$$

Substituting (22) in (20) yields

$$g(\mathbf{V}^h \hat{\mathbf{x}}^h) - g(\mathbf{V}^H \hat{\mathbf{x}}^H) \approx - (\hat{\mathbf{y}}^h)^T (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H). \quad (23)$$

where the fine adjoint solution $\hat{\mathbf{y}}^h \in \mathbb{R}^q$ is the solution to

$$(\mathbf{V}^h)^T \frac{\partial \mathbf{r}^k}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H)^T \mathbf{V}^h \hat{\mathbf{y}}^h = (\mathbf{V}^h)^T \frac{\partial g}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H)^T \quad (24)$$

Because we would like to avoid solving with the higher-dimensional fine basis \mathbf{V}^h , we approximate $\hat{\mathbf{y}}^h$ as the prolongation of the coarse adjoint solution

$$\hat{\mathbf{y}}_H^h = \mathbf{I}_H^h \hat{\mathbf{y}}^H, \quad (25)$$

where $\hat{\mathbf{y}}^H$ is the solution to

$$(\mathbf{V}^H)^T \frac{\partial \mathbf{r}^k}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H)^T \mathbf{V}^H \hat{\mathbf{y}}^H = (\mathbf{V}^H)^T \frac{\partial g}{\partial \mathbf{x}}(\mathbf{V}^H \hat{\mathbf{x}}^H)^T \quad (26)$$

Substituting the approximation $\hat{\mathbf{y}}_H^h$ for $\hat{\mathbf{y}}^h$ in (23) yields a cheaply computable error estimate

$$g(\mathbf{V}^h \hat{\mathbf{x}}^h) - g(\mathbf{V}^H \hat{\mathbf{x}}^H) \approx - (\hat{\mathbf{y}}_H^h)^T (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H). \quad (27)$$

The absolute value of the right-hand side can be bounded as

$$|(\hat{\mathbf{y}}_H^h)^T (\mathbf{V}^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H)| \leq \sum_{i \in \mathbb{N}(q)} \delta_i^h \quad (28)$$

where the error indicators $\delta_i^h \in \mathbb{R}_+$, $i \in \mathbb{N}(q)$ are

$$\delta_i^h = |[\hat{\mathbf{y}}_H^h]_i (\mathbf{v}_i^h)^T \mathbf{r}(\mathbf{V}^H \hat{\mathbf{x}}^H)|. \quad (29)$$

Meyer and Matties [41] also proposed a dual-weighted residual method for reduced-order models. However, their approach was not applied to adaptive refinement and did not consider a hierarchy of reduced bases; further, their proposed dual solve was carried out on the full-order model, which is infeasibly expensive for most contexts.

Algorithm 2 Error estimates**Input:** coarse reduced basis \mathbf{V}^H , coarse solution $\hat{\mathbf{x}}^H$ **Output:** fine reduced basis \mathbf{V}^h , fine error-estimate vector $\boldsymbol{\delta}^h$

- 1: Solve coarse adjoint problem (26) for $\hat{\mathbf{y}}^H$.
- 2: Define prolongation operator \mathbf{I}_H^h via Eq. (16).
- 3: Define fine reduced basis \mathbf{V}^h via Eq. (11) and fine representation of adjoint solution $\hat{\mathbf{y}}_H^h$ via Eq. (25)
- 4: Compute fine error-estimate vector $\boldsymbol{\delta}^h$ via Eq. (29)

Remark. Some mesh-refinement techniques [35, 36] advocate computing refinement indicators that minimize the error in the computable correction

$$(\hat{\mathbf{y}}^h - \hat{\mathbf{y}}_H^h)^T (\mathbf{V}^h)^T \mathbf{r} (\mathbf{V}^H \hat{\mathbf{x}}^H).$$

To approximate this quantity, they employ prolongation operators of varying fidelity, e.g., linear and quadratic interpolants. Such a strategy is not straightforwardly applicable to the current context, as the prolongation operator \mathbf{I}_H^h is exact. ■

5. Adaptive h -refinement algorithm

We now return to the original objective of this paper: adaptively refine the reduced-order model. Algorithm 3 describes our proposed methodology for achieving this within a time integration scheme. Step 1 first computes the reduced-order-model solution satisfying a tolerance ϵ_{ROM} . Then in Step 2, refinement occurs if the norm of the *full-order residual* is above a desired threshold ϵ . Note that other error indicators could be used to flag refinement, e.g., error surrogates [42]. Refinement continues until this full-order tolerance is satisfied; note that any such tolerance can be reached, as a completely split basis yields a reduced-order model equivalent to the full-order model (see Section 3.1). Finally, Step 7 resets the basis to the original basis every n_{reset} time steps. This ensures 1) the basis does not grow monotonically, and 2) work performed to refine the basis can be amortized over subsequent time steps, where the solution is unlikely to significantly change. Note that if Step 1 entails an iterative solve (e.g., Newton), then the pre-refinement solution can be employed as an initial guess.

Algorithm 4 describes the proposed method for refining the basis using the refinement mechanism and error indicators presented in Sections 3 and 4, respectively. Appendix II describes a more sophisticated approach wherein the basis vectors are not split into all possible children; the children are separated into groups, each of which contributes roughly the same fraction of that vector's error.

First, Step 1 computes error estimates for the fine basis (i.e., current basis with all vectors split into all possible children) using the dual-weighted residual approach. Step 3 identifies the parent basis vectors to refine: those with above-average error contribution from its children. Steps 5–8 split the parent vector i into vectors corresponding to its q_i children according to the defined tree. Steps 9–12 update the reduced basis and tree nodes. Because this split does not guarantee a full-ranks basis, Step 14 performs an efficient QR factorization with column

Algorithm 3 Adaptive h -refinement**Input:** timestep k , basis \mathbf{V} , ROM solver tolerance ϵ_{ROM} , FOM solver tolerance ϵ **Output:** updated basis \mathbf{V} , generalized state $\hat{\mathbf{x}}^k$

- 1: Compute ROM solution $\hat{\mathbf{x}}^k$ satisfying $\|\mathbf{V}^T \mathbf{r}^k(\bar{\mathbf{x}} + \mathbf{V} \hat{\mathbf{x}}^k; \boldsymbol{\mu})\| \leq \epsilon_{\text{ROM}}$.
- 2: **if** FOM not converged $\|\mathbf{r}^k(\bar{\mathbf{x}} + \mathbf{V} \hat{\mathbf{x}}^k; \boldsymbol{\mu})\| > \epsilon$ **then**
- 3: Refine basis via Algorithm 4: $\mathbf{V} \leftarrow \text{Refine}(\mathbf{V}, \hat{\mathbf{x}}^k)$.
- 4: Return to Step 1.
- 5: **end if**
- 6: **if** $\text{mod}(k, n_{\text{reset}}) = 0$ **then**
- 7: Reset basis $\mathbf{V} \leftarrow \mathbf{V}^{(0)}$.
- 8: **end if**

Algorithm 4 Refine**Input:** initial basis \mathbf{V} , reduced solution $\hat{\mathbf{x}}$ **Output:** refined basis \mathbf{V}

- 1: Compute fine error-estimate vector and fine reduced basis via Algorithm 2:
 $(\boldsymbol{\delta}^h, \mathbf{V}^h) \leftarrow \text{Error estimates}(\mathbf{V}, \hat{\mathbf{x}})$.
- 2: Put local error estimates in parent-child format $\eta_{ij} = \delta_{f(i,j)}^h$, $i \in \mathbb{N}(p)$, $j \in \mathbb{N}(q_i)$.
- 3: Identify basis vectors to refine $I = \{i \mid \sum_j \eta_{ij} \geq 1/p \sum_{kj} \eta_{kj}\}$
- 4: **for** $i \in I$ **do** {Split \mathbf{v}_i into q_i vectors}
- 5: **for** $k \in \mathbb{N}(q_i)$ **do**
- 6: $\mathbf{x}_k = \mathbf{v}_{f(i,k)}^h$
- 7: $\bar{d}_k = C(d_i, k)$
- 8: **end for**
- 9: $\mathbf{v}_i \leftarrow \mathbf{x}_1$, $d_i \leftarrow \bar{d}_1$
- 10: **for** $k = 2, \dots, q_i$ **do**
- 11: $\mathbf{v}_{p+k-1} \leftarrow \mathbf{x}_k$, $d_{p+k-1} \leftarrow \bar{d}_k$,
- 12: **end for**
- 13: **end for**
- 14: Compute thin QR factorization with column pivoting $\mathbf{V} = \mathbf{Q}\mathbf{R}$, $\mathbf{R}\bar{\boldsymbol{\Pi}} = \bar{\mathbf{Q}}\bar{\mathbf{R}}$.
- 15: Ensure full-rank matrix $\mathbf{V} \leftarrow \mathbf{V} [\bar{\boldsymbol{\pi}}_1 \cdots \bar{\boldsymbol{\pi}}_r]$, where r denotes the numerical rank of \mathbf{R} .
- 16: Update tree $[d_1 \cdots d_r] \leftarrow [d_1 \cdots d_p] [\bar{\boldsymbol{\pi}}_1 \cdots \bar{\boldsymbol{\pi}}_r]$.

pivoting to identify ‘redundant’ basis vectors. Step 15 subsequently removes these vectors from the basis and Step 16 performs the necessary bookkeeping for the tree nodes.

6. Numerical experiments: parameterized inviscid Burgers’ equation

We assess the method’s performance on the parameterized inviscid Burgers’ equation. While simple, this problem is particularly challenging for reduced-order models. This arises from the fact that ROMs approximate the solution as a linear combination of fixed reduced-basis function; as such, they work well when the dynamics are primarily Eulerian, i.e., are fixed with respect to the underlying grid. However, when the dynamics are Lagrangian in nature

and exhibit motion with respect to the underlying grid (e.g., moving shocks), reduced-order models generally fail to capture the phenomenon at every time step and parameter instance.

We employ the problem setup described in Ref. [43], which was also employed to test the GNAT method in Ref. [26]. Consider the parameterized initial boundary value problem

$$\frac{\partial u(x, \tau)}{\partial \tau} + \frac{1}{2} \frac{\partial (u^2(x, \tau))}{\partial x} = 0.02e^{\mu_2 x} \quad (30)$$

$$u(0, \tau) = \mu_1, \quad \forall \tau > 0 \quad (31)$$

$$u(x, 0) = 1, \quad \forall x \in [0, 100], \quad (32)$$

where μ_1 and μ_2 are two real-valued input variables. Godunov's scheme discretizes the problem, which leads to a finite-volume formulation consistent with the original formulation in Eq. (1). The one-dimensional domain is discretized using a grid with 251 nodes corresponding to coordinates $x_i = i \times (100/250)$, $i = 0, \dots, 250$. Hence, the resulting CFD model is of dimension $n = 250$. The solution $u(x, \tau)$ is computed in the time interval $\tau \in [0, 50]$ using a uniform computational time-step size $\Delta t = 0.05$, leading to $t = 1000$ total time steps.

For simplicity, we employ a POD–Galerkin ROM for this study. During the offline stage, snapshots of the state are collected for the first t_{train} time steps at training inputs. Then, the initial condition is subtracted from these snapshots, and they are concatenated column-wise to generate the snapshot matrix. Finally, the thin singular value decomposition of the snapshot matrix is computed, and the reduced basis \mathbf{V} is set to the first p left singular vectors. During the online stage, a Galerkin projection is employed using this reduced basis. For all experiments, the initial condition is set to the reference condition, i.e., $\mathbf{x} = \mathbf{x}^0$. For h -adaptivity, we set the number of means to $\bar{k} = 10$ in Algorithm 1. For Algorithm 2, the output of interest is set to the residual norm $g(\mathbf{x}^k; \boldsymbol{\mu}) = \|\tilde{\mathbf{r}}^k(\mathbf{x}^k; \boldsymbol{\mu})\|_2^2$. For Algorithm 3, the ROM tolerance is set to $\epsilon_{\text{ROM}} = 5 \times 10^{-3}$ and the FOM tolerance to $\epsilon = 0.05$.[‡] The basis-reset frequency n_{reset} will be varied during the experiments. Step 1, incurs a Newton solve; when refinement has occurred, the initial guess is set to the converged solution from the previous refinement level. Finally, the experiments employ the (more complex) Refine method defined by Algorithm 5 with a child-partition factor $\alpha = 2$.

Note that because the residual operator is nonlinear in the state, a projection alone is insufficient to generate computational savings over the full-order model. Future work will address extending the proposed h -refinement method to ROMs equipped with a complexity reduction mechanism such as empirical interpolation or gappy POD.

6.1. Fixed inputs

For this example, the input parameters are set to $\mu_1 = 3$ and $\mu_2 = 0.02$. However, the problem can be considered to be predictive, as we only collect snapshots in the time interval $\tau_{\text{train}} \in [0, 7.5]$, i.e., for the first $t_{\text{train}} = 150$ time steps.

Table I reports results typical POD–Galerkin ROMs of differing dimensions, as well as results for the proposed h -refinement method with different parameters. Here, the relative

[‡]For the ROMs without adaptivity, the ROM convergence tolerance is set to $\epsilon_{\text{ROM}} = 1 \times 10^{-5}$.

error is defined as

$$\text{relative error} = \frac{1}{t} \sum_{k=1}^t \|u_{\text{FOM}}(\cdot, \tau^k) - u_{\text{ROM}}(\cdot, \tau^k)\|_{L_2} / \|u_{\text{FOM}}(\cdot, \tau^k)\|_{L_2}.$$

Figure 3 compares the solutions predicted by POD–Galerkin with no basis truncation (i.e., $p = 150$) and that of the proposed method with an initial basis size of $p^{(0)} = 10$ with $\mathbf{V}^{(0)} \in \mathbb{R}^{n \times p^{(0)}}$ and a basis-reset frequency of $n_{\text{reset}} = 50$.

	no adaptivity			h -adaptivity				
initial basis dimension $p^{(0)}$	10	45	150	5	10	20	10	10
basis-reset frequency n_{reset}				50	50	50	100	25
average basis dimension per Newton iteration \bar{p}	10	45	150	41.4	44.3	58	73	37
average number of Refine calls per time step				0.20	0.19	0.14	0.13	0.28
relative error (%)	45.8	43.9	8.5	0.3	0.5	0.2	0.2	0.3
online time (seconds)	1.4	2.14	5.77	5.53	4.63	7.27	6.90	7.46

Table I. Comparison between POD–Galerkin ROMs without refinement and with h -adaptive refinement for the fixed-inputs case.

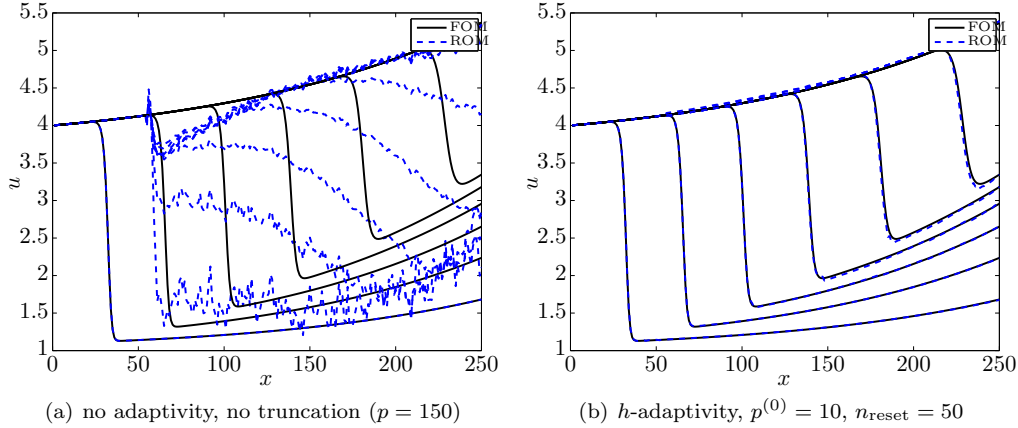


Figure 3. Comparison of solutions computed by POD–Galerkin with and without adaptivity for the fixed-inputs case.

First, note that the reduced-order model is highly inaccurate (even when the basis is not truncated) unless equipped with h -adaptivity. The reason for this is simple: the training has not captured the flow regime with shock locations past approximately $x = 60$. This illustrates a powerful capability of the proposed h -adaptation methodology: it enables ROMs to be incrementally refined to capture previously unobserved phenomena. In fact, the average basis dimension (per Newton iteration) for the best-performing h -adaptive ROM ($p^{(0)} = 10$, $n_{\text{reset}} = 50$) is only $\bar{p} = 44.3$, which is smaller than the basis dimensions for ROMs without adaptivity ($p = 45$ and $p = 150$) that yield much higher errors (43.9% and 8.5%, respectively).

Second, adaptation parameters $p^{(0)}$ and n_{reset} both lead to a performance tradeoff. When $p^{(0)}$ is small, it leads to smaller average basis sizes \bar{p} . However, it increases the number of Refine calls per time step, as the smaller basis must be refined more times to achieve desired accuracy. Similarly, resetting the basis more frequently (smaller n_{reset}) leads to a smaller \bar{p} , but more average refinement steps. As such, an intermediate value of both parameters leads to the shortest online evaluation time.

Finally, notice that the online evaluation time for the adaptive ROM with an average basis size of $\bar{p} = 44.3$ is roughly twice that of a non-adaptive ROM with roughly the same basis size $p = 45$. This discrepancy in evaluation time can be attributed to the overhead in performing the adaptation. For larger problem sizes, one would expect this overhead to be smaller relative to the total online evaluation time.

Next, we assess the performance of the h -refinement method as the full-order tolerance ϵ varies. Table II and Figure 4 report the results. As expected, the proposed method allows the ROM to achieve any of the prescribed tolerances. As the tolerance becomes more rigorous, the ROM solution improves; however, it does so at increased computational cost, as both the average basis dimension \bar{p} and number of Refine calls per time step increase to satisfy the requirement.

	$\epsilon = 0.35$	$\epsilon = 0.05$	$\epsilon = 0.01$
average basis dimension per Newton iteration \bar{p}	33.6	44.2507	53.9
average number of Refine calls per time step	0.115	0.189	0.212
relative error (%)	12.2	0.51	0.078
online time (seconds)	4.61	4.63	7.64

Table II. Effect of full-order-model tolerance ϵ on h -adaptive refinement for $p^{(0)} = 10$ and $n_{\text{reset}} = 50$ for the fixed-inputs case.

6.2. Input variation

For this experiment, we assess the proposed methodology in a parameter-varying scenario. In particular, the offline stage collects snapshots in the time interval $\tau_{\text{train}} \in [0, 2.5]$ for the training set $\{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^3\}$ described in Table III.

Figure 5 and Table IV report the results for this experiment. The same phenomena are prevalent as were apparent in the previous experiment. The primary difference is that the POD–Galerkin model without adaptivity performs better. However, h -adaptivity is still required to drive errors below 1%.

Table III. Offline and online inputs for the inviscid Burgers equation

Input variables	Training point $\boldsymbol{\mu}^1$	Training point $\boldsymbol{\mu}^2$	Training point $\boldsymbol{\mu}^3$	Online point $\boldsymbol{\mu}^*$
μ_1	3	6	9	4.5
μ_2	0.02	0.05	0.075	0.038

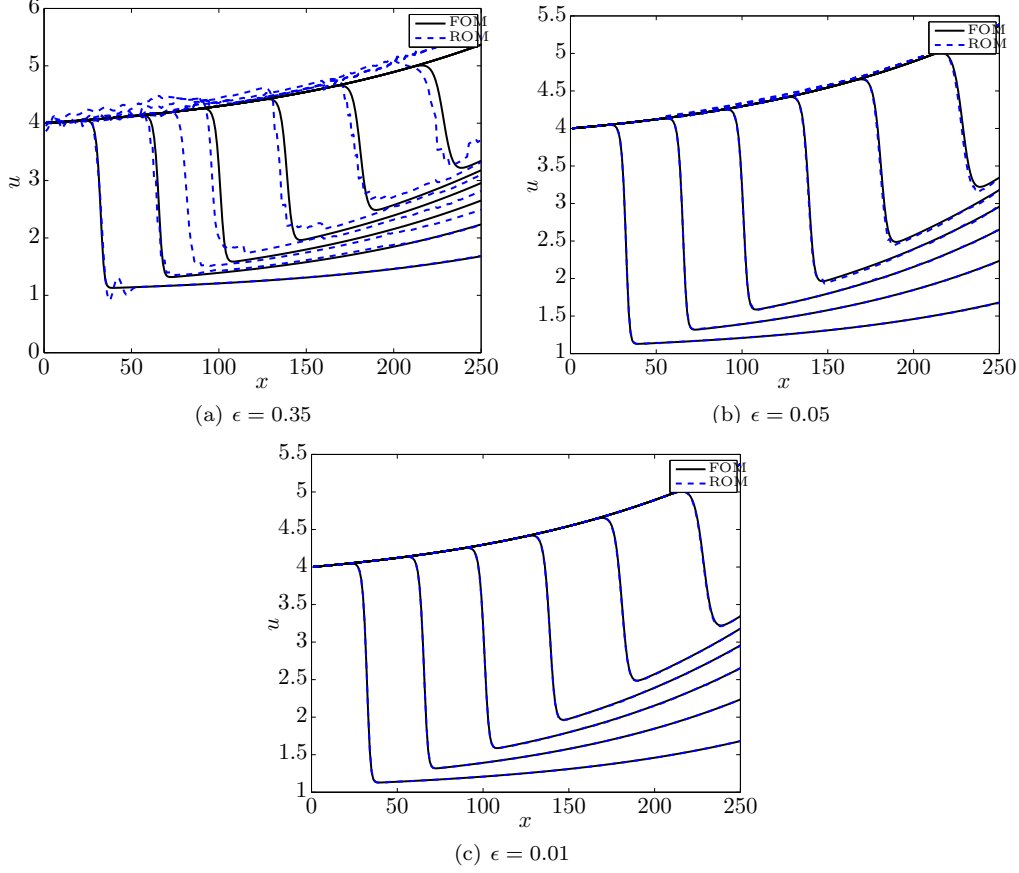


Figure 4. Comparison of solutions computed by h -adaptive POD-Galerkin for different full-order-model tolerances ϵ for the fixed-inputs case.

	no adaptivity			h -adaptivity				
initial basis dimension $p^{(0)}$	10	78	150	5	20	30	20	20
basis-reset frequency n_{reset}				100	100	100	200	50
average basis dimension per Newton iteration \bar{p}	10	78	150	69.8	77.2	87.6	130.6	65.6
average number of Refine calls per time step				0.20	0.072	0.07	0.044	0.11
relative error (%)	41.8	1.7	1.4	0.22	0.14	0.45	0.53	0.70
online time (seconds)	1.75	3.54	8.55	6.41	6.06	8.11	9.11	8.78

Table IV. Comparison between POD-Galerkin ROMs without refinement and with h -adaptive refinement for the input-variation case.

7. Conclusions

This work has presented an adaptive h -refinement method for reduced-order models. Key components include 1) an h -refinement mechanism based on basis splitting and tree structure

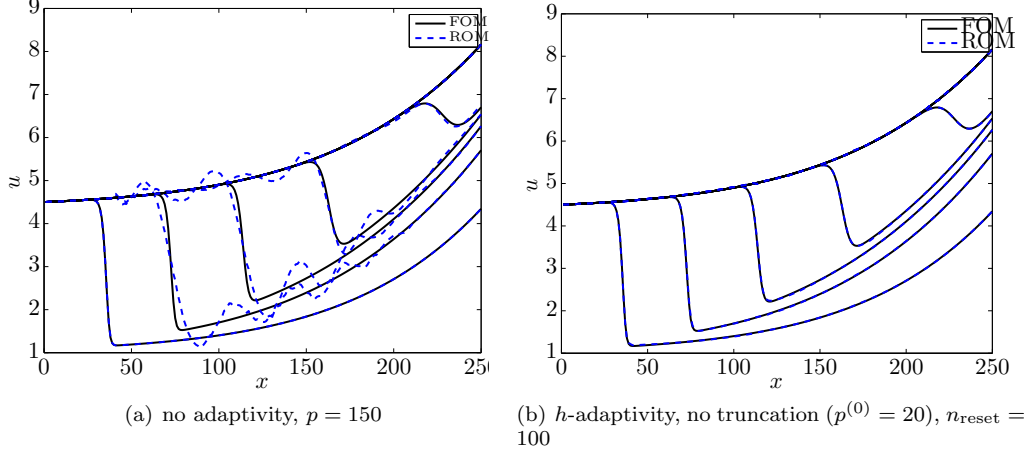


Figure 5. Comparison of solutions computed by POD-Galerkin with and without adaptivity for the varying-inputs case.

constructed via k -means clustering, 2) dual-weighted residual error indicators, and 3) an adaptive algorithm to moderate when and how to perform the refinement. In contrast to existing *a priori* adaptive methods, the proposed technique provides a mechanism to improve the ROM solution *a posteriori*. As opposed to existing *a posteriori* methods, the proposal does so without incurring full-order-model solves. Numerical examples on the inviscid Burgers equation highlighted the method's ability to accurately predict phenomena not present in the training data used to construct the reduced basis.

Future research directions include incorporating complexity reduction (e.g., empirical interpolation, gappy POD) into the refinement process. In particular, as the reduced basis is refined, sample points (and dual reduced-basis vectors) must be added in a systematic way to ensure the reduced-order model remains solvable. In addition, it would be interesting to incorporate a more sophisticated adaptive *coarsening* technique (compared to the simple basis-resetting mechanism in Step 7 of Algorithm 3); for example, one could combine basis vectors whose generalized coordinates are strongly correlated (or anti-correlated) over recent time steps. Further, it would be interesting to pursue adaptive p -refinement methods, wherein other basis vectors (e.g., truncated POD vectors, discrete wavelets) with possibly global support are added from a library to enrich the trial subspace. Finally, it would be advantageous to incorporate Richardson extrapolation in the refinement method to better approximate the outputs of interest; however, this requires knowledge of the convergence rate of the reduced-order model with respect to adding basis vectors.

APPENDIX

II. Refinement algorithm with multiple trees

This section presents a more sophisticated refinement mechanism than that presented in Section 5. In particular, when a vector is flagged for refinement, it is not necessarily split into all its children. Rather, its children are separated into groups, each of which contributes roughly the same fraction α of the total error for that parent vector. This avoids over-refinement when the number of children is relatively large. However, this leads to an increase in required bookkeeping, as the tree structure changes when children merge: the tree must be altered and separately maintained for each vector. Thus, each basis vector \mathbf{v}_i , $i = 1, \dots, p$ will be characterized by its own tree C_i, E_i with m_i nodes, as well as a node on that tree $d_i \in \mathbb{N}(m_i)$.

Algorithm 5 describes the modifications needed to Algorithm 4 to enable this feature. Key modifications include the following. Steps 7–22 separate the children of the parent vector’s tree node d_i into groups; the resulting maintenance of the tree structures is performed in Steps 18–19.[§] In steps 23–26, not only is the basis updated, but the trees are as well. Finally, Step 30 performs the necessary bookkeeping for the tree structures due to the removal of redundant basis vectors.

Acknowledgments

The author acknowledges Matthew Zahr for providing the model-reduction testbed that was modified to generate the numerical results and Seshadhri Comandur for helpful discussions related to tree construction and clustering. This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000.

REFERENCES

1. Benner, P., Gugercin, S., and Willcox, K., “A survey of model reduction methods for parametric systems,” *Max Planck Institute Magdeburg Preprints*, Vol. MPIMD/13–14, 2013.
2. Patera, A. T. and Rozza, G., *Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations*, MIT, 2006.
3. Heinkenschloss, M. and Vicente, L., “Analysis of inexact trust-region SQP algorithms,” *SIAM Journal on Optimization*, Vol. 12, No. 2, 2002, pp. 283–302.
4. Eldred, M. S., Weickum, G., and Maute, K., “A multi-point reduced-order modeling approach of transient structural dynamics with application to robust design optimization,” *Structural and Multidisciplinary Optimization*, Vol. 38, No. 6, 2009, pp. 599–611.
5. Arian, E., Fahl, M., and Sachs, E. W., “Trust-region proper orthogonal decomposition for flow control,” Tech. Rep. 25, ICASE, 2000.
6. Ryckelynck, D., “A priori hyperreduction method: an adaptive approach,” *Journal of Computational Physics*, Vol. 202, No. 1, 2005, pp. 346–366.

[§]Only the lower levels of the tree must be updated, as the current methodology never traverses up a tree.

Algorithm 5 Refine (child grouping)

Input: initial basis \mathbf{V} , reduced solution $\hat{\mathbf{x}}$ **Output:** refined basis \mathbf{V}

- 1: Compute fine error-estimate vector and fine reduced basis via Algorithm 2:
 $(\delta^h, \mathbf{V}^h) \leftarrow \text{Error estimates } (\mathbf{V}, \hat{\mathbf{x}})$.
 - 2: Put local error estimates in parent-child format $\eta_{ij} = \delta_{f(i,j)}^h$, $i \in \mathbb{N}(p)$, $j \in \mathbb{N}(q_i)$.
 - 3: Identify basis vectors to refine $I = \{i \mid \sum_j \eta_{ij} \geq 1/p \sum_{k,j} \eta_{kj}\}$
 - 4: **for** $i \in I$ **do** {Split \mathbf{v}_i into k vectors}
 - 5: $p \leftarrow \dim(\text{range}(\mathbf{V}))$
 - 6: Initialize additional-vector count $k \leftarrow 0$ and handled child-node set $D \leftarrow \emptyset$
 - 7: **while** $D \neq \mathbb{N}(q_i)$ **do** {Divide child nodes into groups with roughly equal error}
 - 8: $D_k = \arg \min_{z \subset K} \text{card}(z)$, where $K = \{z \subset \mathbb{N}(q_i) \setminus D \mid \sum_{j \in z} \eta_{ij} \geq \alpha \sum_j \eta_{ij}\}$, where $\alpha \leq 1$ is a specified child-partition factor.
 - 9: **if** $D_k = \emptyset$ **then**
 - 10: Take all remaining children $D_k = \mathbb{N}(q_i) \setminus D$
 - 11: **end if**
 - 12: $\mathbf{x}_k = \sum_{j \in D_k} \mathbf{v}_{f(i,j)}^h$
 - 13: Update tree: $\bar{C}_k \leftarrow C_i$, $\bar{E}_k \leftarrow E_i$
 - 14: **if** $\text{card}(D_k) = 1$ **then** {Use the same tree}
 - 15: $\bar{d}_k = C_i(d_i, D_k)$
 - 16: **else** {Alter the tree}
 - 17: $\bar{d}_k = d_i$
 - 18: $\bar{C}_k(\bar{d}_k) = \{C_i(d_i, k) \mid k \in D_k\}$
 - 19: $\bar{E}_k(\bar{d}_k) = \bigcup_{k \in \bar{C}_k(d_i)} E_i(k)$
 - 20: **end if**
 - 21: $k \leftarrow k + 1$, $D \leftarrow D \cup D_k$
 - 22: **end while**
 - 23: $\mathbf{v}_i \leftarrow \mathbf{x}_0$, $C_i \leftarrow \bar{C}_0$, $E_i \leftarrow \bar{E}_0$, $d_i \leftarrow \bar{d}_0$
 - 24: **for** $l = 1, \dots, k$ **do**
 - 25: $\mathbf{v}_{p+l} \leftarrow \mathbf{x}_l$, $C_{p+l} \leftarrow \bar{C}_l$, $E_{p+l} \leftarrow \bar{E}_l$, $d_{p+l} \leftarrow \bar{d}_l$
 - 26: **end for**
 - 27: **end for**
 - 28: Compute thin QR factorization with column pivoting $\mathbf{V} = \mathbf{Q}\mathbf{R}$, $\mathbf{R}\bar{\mathbf{\Pi}} = \bar{\mathbf{Q}}\bar{\mathbf{R}}$.
 - 29: Ensure full-rank matrix $\mathbf{V} \leftarrow \mathbf{V} [\bar{\pi}_1 \dots \bar{\pi}_r]$, where r denotes the numerical rank of \mathbf{R} .
 - 30: Update tree $[C_1 \dots C_r] \leftarrow [C_1 \dots C_p] [\bar{\pi}_1 \dots \bar{\pi}_r]$;
 $[E_1 \dots E_r] \leftarrow [E_1 \dots E_p] [\bar{\pi}_1 \dots \bar{\pi}_r]$;
 $[d_1 \dots d_r] \leftarrow [d_1 \dots d_p] [\bar{\pi}_1 \dots \bar{\pi}_r]$.
-

7. Carlberg, K. and Farhat, C., "An Adaptive POD-Krylov Reduced-Order Model for Structural Optimization," *8th World Congress on Structural and Multidisciplinary Optimization, Lisbon, Portugal*, June 1–5 2009.
8. Amsallem, D. and Farhat, C., "An Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity," *AIAA Journal*, Vol. 46, No. 7, July 2008, pp. 1803–1813.
9. Amsallem, D., Cortial, J., Carlberg, K., and Farhat, C., "A method for interpolating on manifolds structural dynamics reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 80, No. 9, 2009, pp. 1241–1258.
10. Eftang, J. L., Patera, A. T., and Rønquist, E. M., "An" hp" certified reduced basis method for parametrized elliptic partial differential equations," *SIAM Journal on Scientific Computing*, Vol. 32, No. 6, 2010, pp. 3170–3200.
11. Eftang, J. L., Knezevic, D. J., and Patera, A. T., "An hp certified reduced basis method for parametrized parabolic partial differential equations," *Mathematical and Computer Modelling of Dynamical Systems*, Vol. 17, No. 4, 2011, pp. 395–422.
12. Haasdonk, B., Dihlmann, M., and Ohlberger, M., "A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space," *Mathematical and Computer Modelling of Dynamical Systems*, Vol. 17, No. 4, 2011, pp. 423–442.
13. Drohmann, M., Haasdonk, B., and Ohlberger, M., "Adaptive Reduced Basis Methods for Nonlinear Convection–Diffusion Equations," *Finite Volumes for Complex Applications VI Problems & Perspectives*, Springer, 2011, pp. 369–377.
14. Peherstorfer, B., Butnaru, D., Willcox, K., and Bungartz, H.-J., "Localized discrete empirical interpolation method," Tech. rep., Technical Report TR-13-1, Aerospace Computational Design Lab, Dept. of Aeronautics & Astronautics, MIT, 2013.
15. Dihlmann, M., Drohmann, M., and Haasdonk, B., "Model reduction of parametrized evolution problems using the reduced basis method with adaptive time partitioning," .
16. Amsallem, D., Zahr, M. J., and Farhat, C., "Nonlinear model order reduction based on local reduced-order bases," *International Journal for Numerical Methods in Engineering*, Vol. 92, No. 10, December 2012, pp. 891–916.
17. Washabaugh, K., Amsallem, D., Zahr, M., and Farhat, C., "Nonlinear model reduction for CFD problems using local reduced-order bases," *42nd AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, AIAA Paper*, Vol. 2686, 2012.
18. Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T., "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathématique Académie des Sciences*, Vol. 339, No. 9, 2004, pp. 667–672.
19. Grepl, M. A., Maday, Y., Nguyen, N. C., and Patera, A. T., "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations," *ESAIM-Mathematical Modelling and Numerical Analysis (M2AN)*, Vol. 41, No. 3, 2007, pp. 575–605.
20. LeGresley, P. A., *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*, Ph.D. thesis, Stanford University, 2006.
21. Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing point estimation in models described by proper orthogonal decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.
22. Chaturantabut, S. and Sorensen, D. C., "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.
23. Galbally, D., Fidkowski, K., Willcox, K., and Ghattas, O., "Non-linear model reduction for uncertainty quantification in large-scale inverse problems," *International Journal for Numerical Methods in Engineering*, Vol. 81, No. 12, published online September 2009, pp. 1581–1608.
24. Drohmann, M., Haasdonk, B., and Ohlberger, M., "Reduced Basis Approximation for Nonlinear Parametrized Evolution Equations based on Empirical Operator Interpolation," *SIAM Journal on Scientific Computing*, Vol. 34, No. 2, 2012, pp. A937–A969.
25. Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, April 2011, pp. 155–181.
26. Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D., "The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows," *Journal of Computational Physics*, Vol. 242, 2013, pp. 623–647.
27. Estep, D., "A posteriori error bounds and global error control for approximation of ordinary differential equations," *SIAM Journal on Numerical Analysis*, Vol. 32, No. 1, 1995, pp. 1–48.
28. Pierce, N. A. and Giles, M. B., "Adjoint recovery of superconvergent functionals from PDE approximations," *SIAM review*, Vol. 42, No. 2, 2000, pp. 247–264.
29. Babuška, I. and Miller, A., "The post-processing approach in the finite element method—part 1:

- Calculation of displacements, stresses and other higher derivatives of the displacements,” *International Journal for numerical methods in engineering*, Vol. 20, No. 6, 1984, pp. 1085–1109.
30. Becker, R. and Rannacher, R., *Weighted a posteriori error control in finite element methods*, Vol. preprint no. 96-1, Universitat Heidelberg, 1996.
 31. Rannacher, R., “The dual-weighted-residual method for error control and mesh adaptation in finite element methods,” *MAFELEAP*, Vol. 99, 1999, pp. 97–115.
 32. Bangerth, W. and Rannacher, R., “Finite element approximation of the acoustic wave equation: Error control and mesh adaptation,” *East West Journal of Numerical Mathematics*, Vol. 7, No. 4, 1999, pp. 263–282.
 33. Becker, R. and Rannacher, R., “An optimal control approach to a posteriori error estimation in finite element methods,” *Acta Numerica 2001*, Vol. 10, 2001, pp. 1–102.
 34. Bangerth, W. and Rannacher, R., *Adaptive finite element methods for differential equations*, Springer, 2003.
 35. Venditti, D. and Darmofal, D., “Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow,” *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204–227.
 36. Venditti, D. A. and Darmofal, D. L., “Grid adaptation for functional outputs: application to two-dimensional inviscid flows,” *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40–69.
 37. Park, M. A., “Adjoint-based, three-dimensional error prediction and grid adaptation,” *AIAA journal*, Vol. 42, No. 9, 2004, pp. 1854–1862.
 38. Nemec, M. and Aftosmis, M., “Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes,” *18th AIAA CFD Conference, Miami. Paper AIAA-2007-4187*, 2007.
 39. Lu, J. C.-C., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.
 40. Fidkowski, K. J., *A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.
 41. Meyer, M. and Matthies, H., “Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods,” *Computational Mechanics*, Vol. 31, No. 1, 2003, pp. 179–191.
 42. Drogmann, M. and Carlberg, K., “The ROMES method for reduced-order-model uncertainty quantification,” *in preparation*, 2014.
 43. Rewienski, M. J., *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2003.