A High-Performance Parallel Implementation of the Certified Reduced Basis Method

David J. Knezevic[a,*], John W. Peterson[b]

[a] *Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 02139 USA*
[b] *Texas Advanced Computing Center, The University of Texas at Austin, Austin, TX 78758-4497*

Abstract

The Certified Reduced Basis method (herein RB method) is a popular approach for model reduction of parametrized partial differential equations. In this paper we introduce new techniques that are required to efficiently implement the Offline "Construction stage" of the RB method on high-performance parallel supercomputers. This enables us to generate certified RB models for large-scale three-dimensional problems that can be evaluated on standard workstations and other "thin" computing resources with speedup of many orders of magnitude compared to the corresponding full order model. We use our implementation to perform detailed numerical studies for two computationally expensive model problems: a natural convection fluid flow problem and a "many parameter" heat transfer problem. In the heat transfer problem, we exploit the computational efficiency of the RB method to perform a detailed study of "snapshot" selection in the Greedy algorithm, and we also examine statistics of the output sensitivity derivatives to obtain a "global" view of the relative importance of the parameters.

Keywords: Reduced basis method, *a posteriori* error bounds, finite element methods, high-performance computing, Boussinesq equation, heat transfer, sensitivity derivatives

1. Introduction

High-fidelity computational simulation of parametrized partial differential equations (PDEs) is fundamentally important in many scientific and engineering contexts. However, classical high-fidelity numerical discretizations — finite element methods, finite difference methods, spectral methods — are computationally expensive for large-scale problems. This limits their applicability in situations where a PDE needs to be solved for many different parameters (optimization, inverse problems, multiscale analysis), or in which real-time evaluation is desirable (control applications, "in situ" design). To address this limitation, a large body of model reduction literature has emerged [1, 2, 3, 4]. The basic goal in model reduction is to develop a low-dimensional representation that accurately captures the dynamics of a high-fidelity numerical discretization over a specific parameter regime of interest. One well-established approach to model reduction of parametrized PDEs is the certified Reduced Basis (RB) method [5, 6, 7, 8, 9]. In the RB method, a very low-dimensional approximation to the "solution manifold" of a parametrized PDE is constructed, typically resulting in a reduced order model that achieves orders of magnitude savings in computation time and memory.

Throughout this paper we shall assume there exists a sufficiently accurate (for our purposes) finite element discretization of a parametrized PDE, and that we seek to construct a reduced-order model based on this discretization via the RB method. Employing the standard RB nomenclature, we shall henceforth refer to the high-fidelity discretization as the "truth" model. It has been well-established [9] that, for a class of PDEs that satisfy the "affine hypothesis," the RB method allows rapid and highly-accurate approximation of the high-fidelity (truth) model and, moreover, incorporates inexpensive *a posteriori* error bounds that provide a

*Corresponding author.

rigorous measure of the error incurred in the model reduction process. The focus of this paper, however, is to demonstrate that by exploiting opportunities for parallelism in the RB framework, we can realize the full potential of the RB methodology for large-scale PDE simulations. The "pay-off" of model reduction clearly increases in direct proportion to the complexity and expense of the full-fidelity simulation.

The RB framework is naturally separated into an "inexpensive" Online stage and an "expensive" Offline stage. In the inexpensive Online stage, the reduced-order model and corresponding error bounds can be *evaluated* for user-specified input parameters — the cost of the Online calculations are independent of the truth discretization and hence can be performed very rapidly and with limited computational resources [10]. The Offline stage, on the other hand, corresponds to the *construction* of the reduced order model. In this stage we appeal to the truth discretization to develop basis functions that are highly tailored to the solution manifold, as well as to perform other truth model-dependent preprocessing for subsequent Online evaluation steps. In this paper we develop a high-performance implementation of the Offline stage of the RB method. This enables robust treatment of problems with many parameters as well as problems with extremely expensive truth solves, which are out of reach without the ability to exploit parallel computing resources. We explore two large-scale model problems in the numerical results section that demonstrate the effectiveness of our implementation.

1.1. Overview of the RB Method

There are many discussions of the RB method already available in the literature (e.g. [9, 11, 12, 13, 14]), hence for the sake of brevity we do not reproduce this material here. However, we do need to introduce some basic notation that will be required in the subsequent sections of this paper. Following the standard RB method notation [9], we suppose that the system input $\mu \in \mathcal{D} \subset \mathbb{R}^P$ (where $\mathcal{D}$ is a bounded set) enters as a parameter in a PDE which describes the relevant physical phenomena over an appropriate spatial domain $\Omega \subset \mathbb{R}^d, d = 1, 2$ or 3. We may consider either steady-state or time-dependent problems; in the time-dependent case we suppose $0 \leq t \leq t_f$ is the time interval of interest. This PDE yields (i) a field variable over Ω, $u(\mu) \in X(\Omega)$ (where $X(\Omega)$ is an appropriate function space), and (ii) a scalar output (or outputs) of interest, $s(\mu) \in \mathbb{R}$, which can be expressed as a (say) linear functional of the field variable, $s(\mu) = \ell(u(\mu))$. Note that the parameter dependence proceeds from the PDE through the field variable and finally to the engineering output.

We suppose that the parametrized PDE is discretized by finite elements in space, and in the time-dependent case, by a finite difference scheme in time. We then obtain a high-fidelity finite element approximation by computing the Galerkin projection over a finite element approximation subspace $X^{\mathcal{N}}$ $(\subset X)$ of large dimension $\mathcal{N}$. For any $\mu \in \mathcal{D}$, our truth approximation is given by $u^{\mathcal{N}}(\mu) \in X^{\mathcal{N}}$ and $s^{\mathcal{N}}(\mu) = \ell(u^{\mathcal{N}}(\mu)) \in \mathbb{R}$. We note that, given our restriction to $\mu \in \mathcal{D}$, all solutions of interest necessarily reside on the parametrically-induced manifold $\mathcal{M}^{\mathcal{N}} \equiv \{u^{\mathcal{N}}(\mu) \,|\, \mu \in \mathcal{D}\}$. We observe that this manifold is relatively low-dimensional and in many cases of practical interest it can be expected to be smooth. Assuming an RB approximation space (which we shall call X_N $(\subset X^{\mathcal{N}})$ where $\dim(X_N) = N$) has been constructed[1] we then consider Galerkin projection over X_N to obtain, for any $\mu \in \mathcal{D}$, the RB solution $u_N(\mu)$ and consequently the output $s_N(\mu) = \ell(u_N(\mu))$. As part of the RB method, we also compute rigorous *a posteriori* bounds, $\Delta_N(\mu)$ and $\Delta_N^s(\mu)$, for the error in the RB field approximation and the RB output approximation, respectively. Thus for any $\mu \in \mathcal{D}$, we have $\|u^{\mathcal{N}}(\mu) - u_N(\mu)\|_X \leq \Delta_N(\mu)$ and $|s^{\mathcal{N}}(\mu) - s_N(\mu)| \leq \Delta_N^s(\mu)$. The existence of these bounds allows us to claim that our RB approximation is *certified*.

The rest of this paper is arranged in the following way: in Section 2 we focus on new innovations required to provide a flexible and extensible implementation of the RB method that can be employed on high-performance computers. We then exploit this implementation in Section 3 to explore two computationally intensive problems: a natural convection problem in an enclosed three-dimensional cavity, and a thermal conduction problem with parametrized thermal conductivities. The former demonstrates robust

[1] In practice we employ the Greedy algorithm [9, 12] to construct X_N, see Algorithm 1 for details. Additionally, in the time-dependent case, the reduced basis approximation inherits the same temporal discretization as the truth.

model reduction for a nontrivial nonlinear problem while the latter demonstrates the efficacy of our implementation for a problem with "many" parameters. In order to gain insight into the complicated parametric dependencies in the thermal conduction problem, we also use our rapid and accurate RB approximation to perform a statistical study of parameter selection in the Greedy algorithm and of the parametric sensitivity of the output. In each case the Offline stage was performed on the TeraGrid supercomputer Ranger at the Texas Advanced Computing Center (TACC). Finally, in Section 4, we give a few concluding remarks and comment on potential avenues of future research created by the existence of our open, high-performance RB simulation framework.

2. High-Performance Implementation of the RB Framework

Our implementation of the RB framework is available as part of the open source C++ parallel finite element library `libMesh` [15]. We utilize `libMesh`'s interfaces to `PETSc` [16] and `SLEPc` [17] for sparse linear algebra and eigenvalue solver functionality, respectively, and we employ existing `libMesh` functionality for all the $\mathcal{N}$-dependent operations required by the RB method [9]:

(i) PDE solves at "greedily" selected parameter values,
(ii) Truth eigensolves required by the Successive Constraint Method (SCM) [18],
(iii) Poisson solves for computing the Riesz representations of the residual terms and outputs,
(iv) Inner products over Ω to obtain the data arrays required in the Online stage of the RB algorithm.

The object-oriented organization of the RB code is illustrated in Figure 1. The base class `RBBase` implements functionality common across all problem types: it stores the "corners" in $\mathbb{R}^P$ of the parameter domain $\mathcal{D}$, the training set $\Xi^{\text{train}} \subset \mathcal{D}$, and the set of parameter-independent operators and parameter-dependent functions from the affine expansion of the PDE. `RBBase` is extended (via the C++ inheritance mechanism) in two directions which encapsulate, in a broad sense, the two major classes of systems — (i) steady, time-dependent, and nonlinear PDE systems and (ii) SCM eigensystems — which must be solved in the RB method.[2]

The `RBSystem` class (left side of the inheritance tree in Figure 1) represents the RB implementation for elliptic problems. Through its descendants it also provides support for linear parabolic (`TransientRBSystem`) and quadratically-nonlinear parabolic (`QNTransientRBSystem`) PDEs. These classes inherit from `libMesh`'s `LinearImplicitSystem` class, thereby gaining access to matrix assembly and linear system solver interfaces for performing truth finite element solves. Each class provides an appropriate implementation of the Greedy algorithm; for example, `RBSystem` provides the standard Greedy algorithm for steady state problems [12] (see Section 2.2 for details), whereas `TransientRBSystem` implements the POD(t)-Greedy(μ) version [19]. The second direction in which `RBBase` is extended is for the SCM: `RBSCMSystem` inherits from `libMesh`'s `EigenSystem` class in order to access the library's `SLEPc` eigensolver interface, and is responsible for implementing the standard SCM algorithm. `QNTransientSCMSystem` extends `RBSCMSystem` to implement the modified SCM required for quadratically nonlinear time-dependent problems [20]. We also provide `RBEIMSystem`, which is an implementation of the Empirical Interpolation Method that enables efficient RB treatment of non-affine PDEs [21].

2.1. Parallelization Opportunity 1: $\mathcal{N}$-Dependent Operations

We exploit two key opportunities for parallelization in the Offline stage of the certified RB framework. First, we employ `libMesh` to parallelize the $\mathcal{N}$-dependent operations of the RB method enumerated in points (i)–(iv) in the previous section. In addition, `libMesh` also handles the more "mundane" yet critical components required in the truth solve, such as the input and parallel partitioning of the computational mesh, and the output of truth solutions in formats recognized by popular visualization software. `libMesh`

[2] We note the following technical detail: `RBBase` is a template class, hence the two specific instantiations of `RBBase` shown in Figure 1, `RBBase<LinearImplicitSystem>` and `RBBase<EigenSystem>`, represent independent inheritance branches and *not* e.g. a "multiple inheritance" object design.

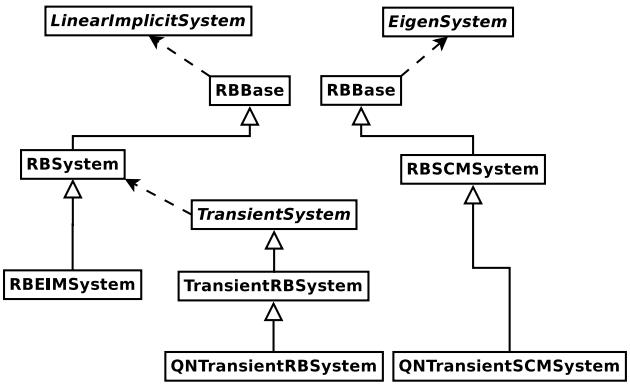


Figure 1: Class diagram for `rbOOmit`. Solid lines with open arrowheads represent C++ inheritance, with arrows pointing from child to parent class. Dashed lines represent specific instantiations of template classes. Class names in italics are part of the `LibMesh` [15] finite element library.

enables us to treat large-scale truth discretizations because it employs domain decomposition with message passing and thread-based parallelism to accelerate matrix assembly and distribute memory consumption among multiple processors.

In a practical RB problem we must define an "affine decomposition" $a(v, w; \mu) = \sum_{q=1}^{Q} \theta_q(\mu) a_q(v, w)$ of a parametrized PDE operator $a(\cdot, \cdot; \cdot) : X^{\mathcal{N}} \times X^{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, where the $a_q(\cdot, \cdot) : X^{\mathcal{N}} \times X^{\mathcal{N}} \to \mathbb{R}$ are parameter-independent operators and the $\theta_q(\cdot) : \mathcal{D} \to \mathbb{R}$ are parameter-dependent functions. To illustrate the way in which `libMesh` facilitates parallelization of the RB method, we provide example assembly code for a simple parameter-independent operator — the Laplace operator on a subdomain of Ω — in Listing 1. The assembly function is passed a `FEMContext` reference which encapsulates all of the necessary element-specific data, and a reference to the `RBSystem` object currently being assembled. The design of the `FEMContext` object allows the library to easily implement thread-based parallelism in the finite element assembly procedures: element-specific data structures can be duplicated as threads are spawned, without any danger of accidentally introducing shared variable lock conditions. An additional level of domain-decomposition parallelism is also obtained as the `assemble_laplacian()` function is called on processor p *only* for the local subset of finite elements "owned" by processor p.

Examples of the finite element data contained in the `FEMContext` object are seen in lines 10, 14, 18, and 21 of Listing 1, where references to element-specific quadrature rule data, shape function gradient values, and degree of freedom counts are obtained from the context object, which is called "`c`" in the code. These data allow the assembly loop which follows in lines 23–26 to be element-*generic*, that is, to work unchanged on any type of geometric finite element. In the body of the loop, on line 26 of Listing 1, the element matrix contribution due to the Laplace operator is accumulated in `c.elem_jacobian` (this contribution is subsequently scaled, in a separate part of the code, by the parameter-dependent function value). The Laplacian example considered here is evidently very simple, but the general structure in Listing 1 can be easily extended to handle much more complicated operators.

As mentioned previously, `libMesh` also leverages the parallel preconditioners and iterative solvers from `PETSc` and `SLEPc` to solve large-scale finite element systems efficiently. While Krylov subspace iterative solvers are sufficiently general and capable of handling the systems which must be solved in the RB method, in some cases it can be advantageous to employ a direct solver. For example, a large number of Poisson problems with fixed matrix and multiple right-hand side vectors need to be solved during the Offline stage to compute terms required for constructing the RB error bound [9]. This is a prototypical scenario in which direct solvers can outperform their iterative cousins. We employ the parallel sparse direct solver `MUMPS` [22] for this purpose, and this predictably leads to significant speedup. For sufficiently large problems, however, direct solvers may still become infeasible due to their large memory footprint.

```
void assemble_laplacian(FEMContext &c, System& system)
{
  if (c.elem->subdomain_id() == 0)
  {
    // Get index of "u" variable from System
    RBSystem& rb_system = libmesh_cast_ref<RBSystem&>(system);
    const unsigned int u_var = rb_system.u_var;

    // Reference to vector of quadrature rule data
    const std::vector<Real> &JxW =
      c.element_fe_var[u_var]->get_JxW();

    // The velocity shape function gradients at q. pts
    const std::vector<std::vector<RealGradient> > &dphi =
      c.element_fe_var[u_var]->get_dphi();

    // The number of local degrees of freedom in each variable
    const unsigned int n_u_dofs = c.dof_indices_var[u_var].size();

    // Now we will build the affine operator
    unsigned int n_qpoints = c.element_qrule->n_points();

    for (unsigned int qp=0; qp != n_qpoints; qp++)
      for (unsigned int i=0; i != n_u_dofs; i++)
        for (unsigned int j=0; j != n_u_dofs; j++)
          c.elem_jacobian(i,j) += JxW[qp] * dphi[j][qp]*dphi[i][qp];
  }
}
```

Listing 1: Sample `libMesh` [15] matrix assembly code demonstrating several techniques which enable parallelization in the Offline stage of the RB framework.

2.2. Parallelization Opportunity 2: The Greedy Algorithm

The other opportunity for parallelization is in the Greedy algorithm (see Algorithm 1), which is employed to select the basis function "snapshots" of $\mathcal{M}^{\mathcal{N}}$, which we denote by $\{\zeta_i, \ldots, \zeta_N\}$. The Greedy algorithm entails evaluation of the RB approximation and associated error bounds on the entire set of training points, Ξ^{train}. There are two key computational tasks in the Greedy algorithm that, for brevity, we encapsulate here in subroutines `Construction` and `Evaluation`. The `Construction` stage takes the current set of basis functions, $\{\zeta_i, \ldots, \zeta_N\}$, as input and develops the Online Dataset(N) needed to evaluate the RB approximation and associated error bounds:

$$[\text{Online Dataset}(N)] = \texttt{Construction}(\{\zeta_n\}_{n=1,\ldots,N}). \tag{1}$$

The `Evaluation` subroutine takes a discrete parameter set $\Xi \subset \mathcal{D}$ as input and returns $\mu^* = \arg\max_{\mu \in \Xi} \Delta_N(\mu)$, and $\texttt{err} = \Delta_N(\mu^*)$:

$$[\mu^*, \texttt{err}] = \texttt{Evaluation}(\Xi; \text{Online Dataset}(N)), \tag{2}$$

Given that the `Evaluation` computations are very inexpensive, we are usually able to utilize relatively large training sets and obtain good "coverage" of $\mathcal{D}$. Nevertheless, there are two situations in which the error bound sampling on Ξ^{train} tends to dominate the computational cost of the Offline stage. The first is problems with many parameters, in which — due to the "curse of dimensionality" — we need to choose *very* large training sets in order to obtain a reasonable coverage of the parameter domain. The other situation is quadratically nonlinear problems in which error bound evaluation requires $O(N^4)$ operations [23, 24]. In this

scenario, when $N \gg 1$, the RB solves required by the "arg max" operation in (2) may require a significant proportion of the overall runtime of the Offline stage.

Algorithm 1 Greedy Algorithm.

specify $\Xi^{\text{train}} \subset \mathcal{D}$ of size n_{train} and tolerance ϵ.
select $\mu^* \in \mathcal{D}$ (arbitrary), set $N = 0$ and $X_0 = \{0\}$.
while $\texttt{err} > \epsilon$ **do**
  $\zeta_{N+1} = (I - \Pi_{X_N}) u^{\mathcal{N}}(\mu^*)$ (normalized);
  $X_N \leftarrow X_N \oplus \text{span}(\zeta_{N+1})$;
  $N \leftarrow N + 1$;
  [Online Dataset(N)] = $\texttt{Construction}(\{\zeta_i\}_{i=1,\ldots,N})$ (*update*);
  $[\mu^*, \texttt{err}] = \texttt{Evaluation}$ (Ξ^{train}, Online Dataset(N));
end while
set $N_{\max} \leftarrow N$.

Fortunately, we can employ an "embarrassingly parallel" implementation of the "arg max" operation. We initialize by partitioning the training set Ξ^{train} into disjoint subsets $\{\mathcal{P}_i, i = 1, \ldots, n_{\text{proc}}\}$, where n_{proc} denotes the number of processors. The parallel algorithm then proceeds with a local "arg max" operation performed simultaneously on all processors. Finally, an inexpensive parallel "arg max" is performed for the n_{proc} local "winning" candidate parameters to find the global maximizer from Ξ^{train}. This simple strategy reduces the computation time for the n_{train} RB solves by a factor of n_{proc}, and (as we demonstrate in Section 3) can lead to very significant Offline speedup.

3. Numerical Results

We now consider two computationally expensive example problems that illustrate the RB techniques discussed above: a nonlinear unsteady Boussinesq problem and a "many parameter" thermal conduction problem. We perform a detailed numerical study of the behavior of the Greedy algorithm for the thermal conduction problem, and we also exploit our reduced basis approximation to perform a statistical study of sensitivity derivatives. All Offline computations were performed on the Ranger supercomputer (located at the Texas Advanced Computing Center), which has in total 62,976 cores, 123TB of memory, and a theoretical peak performance of 579 TFLOPs. We generate reduced order models with rigorous error bounds that can be evaluated on a standard desktop or laptop computer in real-time or near real-time.

3.1. Unsteady Boussinesq Equations

We first consider a three-dimensional version of the unsteady natural convection problem from [24]. The nondimensional, coupled conservation of momentum, energy, and mass equations for this problem are:

$$\frac{\partial \boldsymbol{V}}{\partial t} + \frac{1}{2\sqrt{\text{Gr}\,\text{Pr}}}\big(\boldsymbol{V} \cdot \nabla \boldsymbol{V} + \nabla \cdot \boldsymbol{V}\boldsymbol{V}\big) + \sqrt{\text{Gr}\,\text{Pr}}\nabla P - \nabla^2 \boldsymbol{V} + \sqrt{\text{Gr}\,\text{Pr}} T \hat{\boldsymbol{g}} = 0 \tag{3}$$

$$\frac{\partial T}{\partial t} + \frac{1}{2\sqrt{\text{Gr}\,\text{Pr}}}\big(\boldsymbol{V} \cdot \nabla T + \nabla \cdot T\boldsymbol{V}\big) - \frac{1}{\text{Pr}}\nabla^2 T = 0 \tag{4}$$

$$\nabla \cdot \boldsymbol{V} = 0 \tag{5}$$

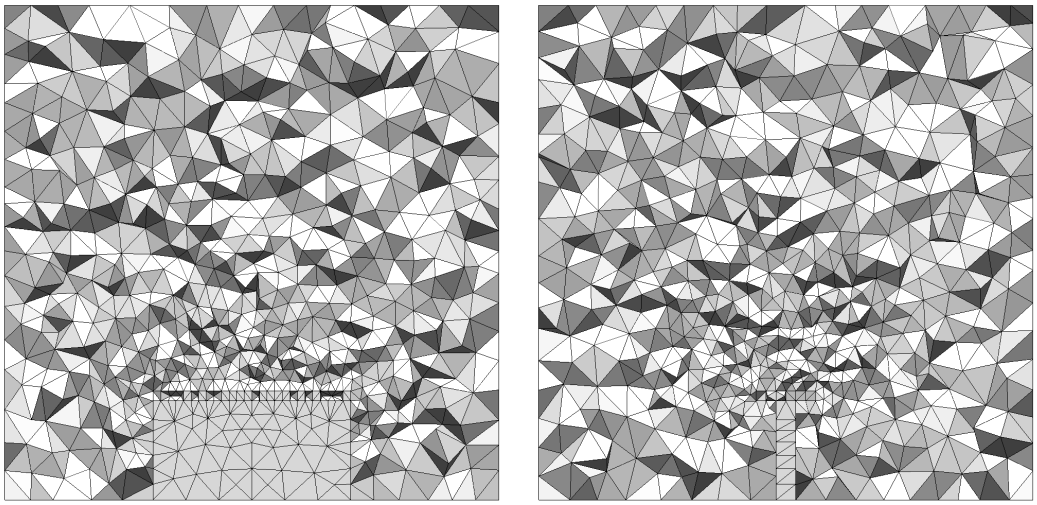
where $\boldsymbol{V} \equiv (V_1, V_2, V_3)$ is the nondimensional velocity vector, P is the nondimensional pressure, T is the nondimensional temperature, and

$$\hat{\boldsymbol{g}} \equiv (-\sin\phi, 0, -\cos\phi)$$

is a unit vector in the direction of gravity. Gr is the Grashof number, a measure of the strength of buoyant effects relative to thermal diffusion, and Pr = 0.71 is the Prandtl number (we assume the fluid is air). Additional details of the nondimensionalization can be found in [24].

The nondimensional domain is $\Omega = (0,5)^3 \setminus \mathcal{P}$ where $\mathcal{P}$ is the pillar $(2.4, 2.6) \times (1.5, 3.5) \times (0, 1)$. We solve for the field variables $\boldsymbol{V}$, P, and T throughout Ω. The "roof" of the cavity is maintained at temperature $T = 0$, the sides and base of the cavity are perfectly thermally insulated, and the top and sides of the pillar are subject to a uniform heat flux of magnitude Gr. We impose no-slip velocity conditions on all boundaries.

The truth weak formulation for (3)–(5) requires only minor modifications from the two-dimensional case in [24], hence we omit it here for the sake of brevity.[3] We consider the time interval $t \in [0, 0.16]$, and employ a Crank-Nicolson temporal discretization with $K = 100$ timesteps. The three-dimensional finite element mesh is composed of quadratic tetrahedral elements and has 71,279 nodes and 50,574 elements; cutaway views showing the internal details of the tetrahedralization are given in Figure 2. The total number of degrees of freedom for the velocity, temperature and pressure is $\mathcal{N}$ = 294,502. The RB and SCM implementations in this case were based on the classes `QNTransientRBSystem` and `QNTransientSCMSystem`, respectively.



(a) $x = 2.5$ (b) $y = 2.5$

Figure 2: x (2a) and y (2b) interior cutaway views of the mesh used in the three-dimensional unsteady Boussinesq calculations. The output regions, which sit just above the heated "pillar," are meshed exactly. The pillar itself is a rectangular cavity at the base of the domain and is not meshed. The mesh is composed of quadratic tetrahedral elements and has 71,279 nodes and 50,574 elements.

We consider a two-tuple parameter $\mu \equiv (\mu_1, \mu_2) \equiv (\mathrm{Gr}, \phi) \in \mathcal{D} \equiv [4000, 6000] \times [0, 0.2]$. Our goal is to study the parametric dependence of the temperature in regions near the top of the heated pillar in the presence of natural convection. Therefore, we consider the following $L^2(\Omega)$-bounded functionals of T

$$s_n(t;\mu) = \ell_n(T(t;\mu),\mu) = \frac{1}{\mu_1 |D_n|} \int_{D_n} T(t;\mu) \; ; \tag{6}$$

as outputs, where $D_1 = [2.2, 2.4] \times [1.5, 3.5] \times [1, 1.1]$, $D_2 = [2.4, 2.6] \times [1.5, 3.5] \times [1, 1.1]$, and $D_3 = [2.6, 2.8] \times [1.5, 3.5] \times [1, 1.1]$ are three small rectangular prisms above the pillar. We denote the RB approximations to the outputs from (6) as $s_{N,n}(t;\mu), n = 1, 2, 3$. A cross-section of the domain geometry and output regions is given in Figure 3, cross-sections of the truth solution at $t = 0.16$ for $(\mathrm{Gr}, \phi) = (6000, 0.2)$ are shown in Figure 4.

[3] Note that we employ skew-symmetric convection operators in (3) and (4) to recover discrete stability properties and simplify RB error bound derivation.

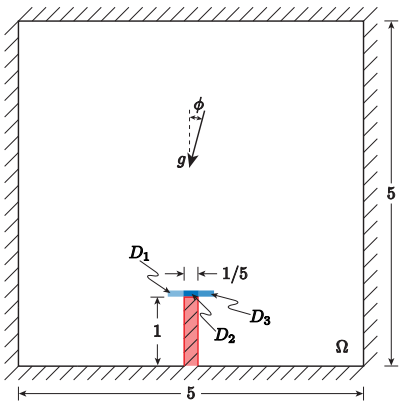


Figure 3: Cross-section at $y = 2.5$ of the computational domain; note that Ω does not include the pillar which is shaded in red. Cross-sections of the output regions D_1, D_2 and D_3 are also indicated.

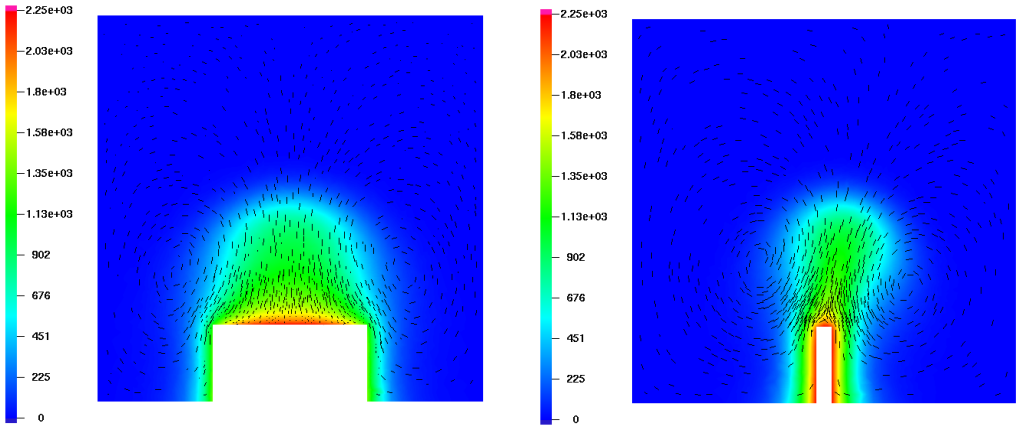
We construct a (primal-only) RB space via the POD(t)-Greedy(μ) algorithm.[4] We specify $\epsilon = 9 \times 10^{-4}$ for the target tolerance and $\delta N = 3$ for the number of POD modes to include at each greedy iteration; we further specify a 32×32 tensor product train sample Ξ^{train} of uniformly spaced points in $\mathcal{D}$. The tolerance is satisfied for $N_{\max} = 90$ and the Greedy convergence is shown in Figure 5. Note that in the quadratically nonlinear case we report a *nominal* error bound during the POD(t)-Greedy(μ) since the SCM cannot be performed until after the RB space has been generated [20]. Nevertheless, this does not compromise the rigor of the RB error bounds in the Online stage. All Offline calculations are performed on 128 processors on Ranger. The total wall time required for the entire Offline stage, including the SCM, is significant: about 42 hours. The "embarrassingly parallel" solves over Ξ^{train} during the greedy algorithm require approximately 24 total minutes in the calculation — a small percentage of the overall Offline runtime. It is important to note, however, that had these solves *not* been parallelized in the manner discussed in this paper, the total wall-clock time for this stage of the calculation would have been approximately 128×24 minutes, or just over 51 hours.

In Figure 6, we show RB output approximations and corresponding RB error bounds obtained with $N = 90$ to the $s_n(t; \mu)$ for parameter values $(\text{Gr}, \phi) = (4000, 0)$, $(5000, 0.1)$, and $(6000, 0.2)$. We note that, as expected, the "left" and "right" outputs (s_1 and s_3, respectively) coincide when $\phi = 0$ and their separation increases as ϕ increases. The Online computation time with $N = 90$ is 20.6 seconds on an AMD Opteron 2382 processor. Most of this time is due to the $O(N^4)$ complexity of the error bounds in the quadratically nonlinear case. Nevertheless, this still represents a speedup factor of approximately 63 with respect to a single truth solve on Ranger, which requires 21.7 minutes on 128 processors.

3.2. Transient Thermal Conduction

We next consider a "many parameter" problem, which is challenging from the point of view of model reduction due to the "curse of dimensionality". We consider transient thermal conduction in a three–dimensional "Swiss cheese" configuration. This is a generalization of the steady-state "thermal block" problem posed on the unit square in two spatial dimensions in [25]; here we consider a time-dependent formulation, a more complicated domain Ω and a much more expensive truth discretization. The domain Ω is given by the unit cube $[0, 1]^3 \setminus S$, where S is a lattice of 27 spheres of radius 1/5 and centers $\{0, 1/2, 1\} \times \{0, 1/2, 1\} \times \{0, 1/2, 1\}$. We subdivide Ω into 27 subdomains, $\overline{\Omega} = \cup_{j=0}^{26} \overline{\Omega}_j$, the Ω_j are shown in Figure 7a, and are numbered sequentially (in a standard "structured" grid pattern) starting with Ω_0 at the origin and

[4] A primal-dual formulation is not appealing in this case since (i) the beneficial effect on output error bounds would be limited due to exponential error bound growth, and (ii) we would need a dual system for each output.



(a) $x = 2.5$ (b) $y = 2.5$

Figure 4: x (4a) and y (4b) interior cutplane views of the truth solution for $(\mathrm{Gr}, \phi) = (6000, 0.2)$. Color contours show the temperature field and the overlaid velocity vectors indicate the circulatory nature of the flow.

then proceeding with x-axis aligned rows, advancing in the y-direction until one layer is complete, followed by the same pattern for the middle and top layers. In particular, the bottom "corners" of the domain correspond to Ω_0, Ω_2, Ω_6, and Ω_8, while the central subdomain is Ω_{13}.

The governing equations are most easily described in weak form: find $u(t;\mu) \in X$ for $0 \le t \le t_f = 1$ such that

$$\int_\Omega \frac{\partial u}{\partial t} v + \sum_{j=1}^{26} \mu_j \int_{\Omega_j} \nabla u \cdot \nabla v + \int_{\Omega_0} \nabla u \cdot \nabla v = \int_{\Gamma_\mathrm{b}} v, \quad \forall v \in X, \tag{7}$$

where μ_j is the thermal conductivity of region Ω_j relative to the thermal conductivity of region Ω_0, $X \equiv \{v \in H^1(\Omega) \,|\, v|_{\Gamma_\mathrm{t}} = 0\}$, Γ_b is the bottom boundary "$z = 0$" (on which we impose inhomogeneous flux boundary conditions), and Γ_t is the top boundary "$z = 1$" (on which we impose zero Dirichlet conditions). We impose homogeneous natural (zero) flux conditions on all other boundaries. We choose $P = 26$ parameters, $\mu_j, 1 \le j \le 26$, and specify the parameter domain $\mathcal{D} = [0.5, 2]^{26}$. We consider one output,

$$s(t;\mu) = \ell(u(t;\mu)) = \frac{1}{|\Omega_0|} \int_{\Omega_0} u(t;\mu),$$

the average temperature over Ω_0, where Ω_0 is the intersection of Ω with the cube $[0, 1/4]^3$.

We pursue a primal-dual approach for this linear time-invariant (LTI) parabolic problem, where the dual problem runs "backwards" in time [11]. The weak formulation for the dual problem is: find $z(t;\mu) \in X$ for $0 \le t \le t_f$ such that

$$-\int_\Omega v \frac{\partial z}{\partial t} + \sum_{j=1}^{26} \mu_j \int_{\Omega_j} \nabla v \cdot \nabla z + \int_{\Omega_0} \nabla v \cdot \nabla z = 0, \quad \forall v \in X, \tag{8}$$

with $\int_\Omega v\, z(t_f) = \ell(v)$. We select a backward Euler temporal discretization with $K = 100$ time levels for both the primal and dual problems and employ a truth finite element approximation space with $\mathcal{N} = 241{,}520$ degrees of freedom. Both the primal and dual systems are implemented via the `TransientRBSystem` class discussed in Section 2, and the SCM is not required here due to the simple "parametrically coercive" [9] structure of (7) and (8). In Figure 8 we show the primal and dual truth solutions at $t = 1$ and $t = 0$, respectively, for the parameter value

$$\mu = (2, 2, 2, 2, 0.5, 2, 0.5, 0.5, 2, 2, 2, 2, 2, 2, 2, 2, 0.5, 2, 2, 2, 2, 2, 2, 2, 2). \tag{9}$$

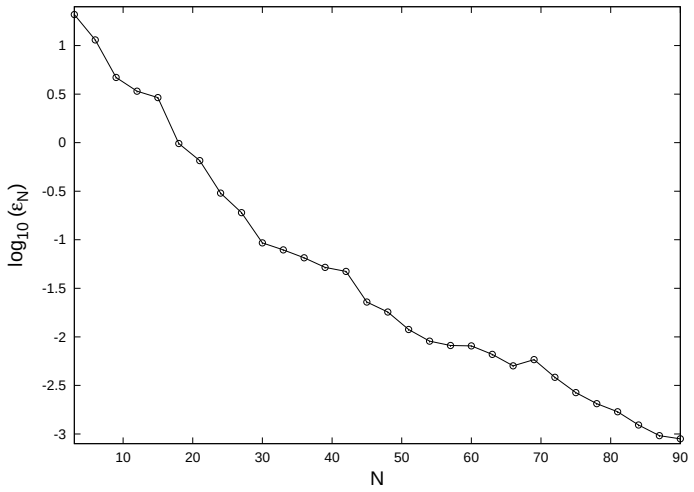


Figure 5: Greedy convergence for the 3D Boussinesq pillar.

This parameter specifies the minimum thermal conductivity in Ω_8 and the adjacent subdomains Ω_5, Ω_7, and Ω_{17}, therefore we observe a "hot corner" in Figure 8a.

We employ the POD(t)-Greedy(μ) for both the primal and dual problems and develop an RB space with $N = 100$ in each case. To investigate the behavior of the Greedy in this "many parameter" problem, we perform the Offline stage three times with log-randomly generated training sets of size $n_{\text{train}} = 10^2,\ 10^4$, and 10^6. The Offline burden due to the expensive truth and (in the $n_{\text{train}} = 10^6$ case) large n_{train} is considerable, and both forms of parallelization from Sections 2.1 and 2.2 are crucial. Here we used 512 processors of Ranger for the Offline computations, and the total Offline wall-clock time in the $n_{\text{train}} = 10^6$ case was approximately 31 hours. The time spent evaluating RB error bounds for the $n_{\text{train}} = 10^6$ calculation was roughly an hour (per processor). The embarrassingly parallel "arg max" operation discussed in Section 2.2 is therefore clearly essential, and in this case saved over 500 hours of computation time compared to an equivalent serial algorithm.

In Figure 9 we show the (normalized) POD(t)-Greedy(μ) convergence,

$$\varepsilon_N \equiv \frac{\Delta_N^K(\mu^*)}{\|u_N(t^K;\mu^*)\|_{L^2(\Omega)}},$$

where $\Delta_N^K(\mu^*)$ denotes the $L^2(\Omega)$-norm RB error bound at time level K, for the primal RB space (the dual convergence plot is analogous and hence is omitted) with the three choices of n_{train}. We observe that the POD(t)-Greedy(μ) convergence reported in the $n_{\text{train}} = 10^2$ case is overly optimistic due to the coarseness of the training set — here Ξ^{train} is a poor surrogate for $\mathcal{D}$. However, the $n_{\text{train}} = 10^4$ and 10^6 cases agree quite well, and this suggests that these larger training sets provide a reasonable surrogate for $\mathcal{D}$. Of course, even the $n_{\text{train}} = 10^6$ training set is extremely sparse in the 26-dimensional parameter domain (e.g. a tensor product grid with only two points in each parameter direction would have $2^{26} \approx 6.7 \times 10^7$ points) but the POD(t)-Greedy(μ) nevertheless performs well due to the highly smooth "parametrically coercive" [9] nature of this problem.

We now proceed to the Online stage. In order to test the accuracy of the primal-dual RB approximation, we perform RB solves on a 5000 point log-random test set, Ξ^{test}, for the RB spaces generated by the $n_{\text{train}} = 10^2$ and $n_{\text{train}} = 10^6$ training sets. We compute the maximum relative output error bound over Ξ^{test}, $\Delta_{\max}^{s,\text{rel}}(\Xi^{\text{test}})$, defined as

$$\Delta_{\max}^{s,\text{rel}}(\Xi^{\text{test}}) \equiv \max_{\mu \in \Xi^{\text{test}}} \frac{\Delta_N^s(t^K;\mu)}{s_N(t^K;\mu) - \Delta_N^s(t^K;\mu)}, \tag{10}$$

N	$\Delta_{\max}^{s,\mathrm{rel}}(\Xi^{\mathrm{test}})$		Online time (seconds)
	$n_{\mathrm{train}} = 10^2$	$n_{\mathrm{train}} = 10^6$	
25	1.13×10^{-1}	1.16×10^{-1}	3.48×10^{-2}
50	2.73×10^{-2}	2.44×10^{-2}	1.01×10^{-1}
75	1.03×10^{-2}	1.05×10^{-2}	2.10×10^{-1}
100	5.22×10^{-3}	4.34×10^{-3}	2.74×10^{-1}

Table 1: Relative output error bounds over the test set Ξ^{test} containing 5000 log-randomly selected points and corresponding Online evaluation timings.

where $\Delta_N^s(t^K;\mu)$ (resp. $s_N(t^K;\mu)$) denotes the primal-dual output bound (resp. dual-corrected RB output) at the final time, $t^K(\equiv t_f)$. Note that the denominator in (10) is a lower-bound for the truth output $s^{\mathcal{N}}(t^K;\mu)$. In Table 1 we present results for $N = 25, 50, 75$, and 100 along with corresponding Online computation times for a single primal-dual RB solve.[5] The data clearly indicate the trade-off between RB accuracy and Online cost. We note from Table 1 that for the $n_{\mathrm{train}} = 10^6$ primal-dual RB spaces with $N = 100$ we obtain an Online output approximation with relative error bound less than 0.5% in 0.274 seconds — compared to approximately 200 seconds for a truth solve on 512 processors of Ranger.

Statistical analysis of the Greedy algorithm

The data generated during the primal and dual basis training procedures for this problem provide a rich dataset and therefore it is instructive to examine the distribution of the parameter values selected by the Greedy algorithm in the high-dimensional space $\mathcal{D}$. We first consider the primal POD(t)-Greedy(μ) calculation with $n_{\mathrm{train}} = 10^6$ (recall that Ξ^{train} is log-uniform randomly distributed) and we generate individual histograms for each of the 26 parameters, $\mu_j, 1 \le j \le 26$, based on the 100 greedily-selected μ values. We specify n_{bins} equally-sized bins within the range $[\mu_{\min}, \mu_{\max}] = [0.5, 2]$ and let $\mu_j(i), 1 \le i \le n_{\mathrm{bins}}$ denote the number of occurrences of μ_j within bin i. To simplify the presentation we then generate a histogram for the "composite" variable $\langle\mu\rangle$,

$$\langle\mu\rangle(i) \equiv \frac{1}{26}\sum_{j=1}^{26}\mu_j(i) \qquad 1 \le i \le n_{\mathrm{bins}}. \tag{11}$$

This composite histogram for $n_{\mathrm{bins}} = 10$ is given in Figure 10, where it is compared to a histogram for a log-uniform random sample of data points over the same parameter range. The greedily-selected parameter values are biased towards the endpoints of the μ domain relative to the log-uniform random sample. The tendency of the Greedy algorithm to select parameter values near the endpoints appears analogous to the endpoint clustering seen in classical interpolation theory, for example the optimal approximation properties provided by Chebyshev points in one dimension [27].

In addition to the statistical distribution of the μ values, it is also instructive to look at the different spatial configurations of the parameters selected by the greedy algorithm, in particular the jumps in thermal diffusivity between adjacent subregions. We define the average jump for a given μ vector as

$$\langle\delta\mu\rangle \equiv \frac{1}{|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}}|\mu_i - \mu_j| \tag{12}$$

where $\mathcal{I}$ is the set of all ordered pairs (i,j) for which $i < j$ and regions Ω_i and Ω_j share a common surface. For the $3 \times 3 \times 3$ configuration considered in the "Swiss cheese" problem (see Figure 7a) there are 54 unique ordered pairs in $\mathcal{I}$. A histogram of $\langle\delta\mu\rangle$ values computed for the $n_{\mathrm{train}} = 10^6$ case is given in Figure 11,

[5] Computation times are averaged over Ξ^{test} and are from an AMD Opteron 2382 processor.

where it is compared to the jumps computed for a large log-uniform random sample of data. The jumps in the greedily-selected dataset are, on average, larger than those from the log-uniform random sample. This suggests that the greedy algorithm detects higher error in μ configurations with larger diffusivity jumps — which in some sense are "higher frequency" modes — thereby preferentially selecting them for inclusion in the reduced basis.

Sensitivity derivative analysis

We have demonstrated that our primal-dual RB method yields a very fast and accurate output approximation for the "Swiss Cheese" problem. Our goal in this final subsection is to demonstrate that the primal-dual RB approximation can be "recycled" in order to provide useful sensitivity derivative information at a fraction of the computational cost of the full-order model.[6] Sensitivity derivatives are crucial in many computational engineering contexts, including control, optimization, parameter estimation and uncertainty quantification [28, 29, 30]. In the present study we exploit the RB method to perform a statistical analysis of the sensitivity derivatives from a large parameter sample to gain a "global" view of the relative importance of the parameters.

We first sketch the derivation of the sensitivity formulation for this problem. Working with the continuous-time weak formulation for the sake of clarity, we note that

$$\int_\Omega \frac{\partial}{\partial t}\frac{\partial u}{\partial \mu_p} v + \sum_{j=1}^{26} \mu_j \int_{\Omega_j} \nabla \frac{\partial u}{\partial \mu_p} \cdot \nabla v + \int_{\Omega_0} \nabla \frac{\partial u}{\partial \mu_p} \cdot \nabla v = -\int_{\Omega_p} \nabla u \cdot \nabla v, \quad \forall v \in X, \tag{13}$$

for $p = 1, \ldots, 26$. Then, from (8) and (13) we obtain

$$\int_0^t \left(-\int_{\Omega_p} \nabla u \cdot \nabla z \right) dt = \int_0^t \left(\int_\Omega \frac{\partial}{\partial t}\frac{\partial u}{\partial \mu_p} z + \sum_{j=1}^{26} \mu_j \int_{\Omega_j} \nabla \frac{\partial u}{\partial \mu_p} \cdot \nabla z + \int_{\Omega_0} \nabla \frac{\partial u}{\partial \mu_p} \cdot \nabla z \right) dt = \frac{\partial s}{\partial \mu_p}(t;\mu). \tag{14}$$

This relationship carries over directly to our fully-discrete truth finite element formulation, so that we obtain:

$$\frac{\partial s^{\mathcal{N}}}{\partial \mu_p}(t^\ell;\mu) = \sum_{k=1}^{\ell} \Delta t \left(-\int_{\Omega_p} \nabla u^{\mathcal{N}}(t^k;\mu) \cdot \nabla z^{\mathcal{N}}(t^{K-\ell+k};\mu) \right). \tag{15}$$

In the context of RB sensitivity derivatives, however, this relationship is only approximate since we employ a Galerkin method and the primal and dual RB spaces do not coincide. Also, we employ Lagrange RB spaces here which are targeted at $u^{\mathcal{N}}$ and $z^{\mathcal{N}}$, not $\frac{\partial u^{\mathcal{N}}}{\partial \mu_p}$ and $\frac{\partial z^{\mathcal{N}}}{\partial \mu_p}$, hence even if the RB residuals are small it is not clear that the primal-dual RB approximation will lead to good sensitivity approximations.[7] Hence, our first task is to determine whether the RB approximation developed for this problem, which is very effective for output approximation, is also satisfactory for sensitivity derivative approximation. We define our RB approximation to $\frac{\partial s^{\mathcal{N}}}{\partial \mu_p}(t^\ell;\mu)$ for $p = 1, \ldots, P$ as,

$$J_N^p(t^\ell;\mu) \equiv \sum_{k=1}^{\ell} \Delta t \left(-\int_{\Omega_p} \nabla u_N(t^k;\mu) \cdot \nabla z_N(t^{K-\ell+k};\mu) \right), \tag{16}$$

and set $J_N(t^\ell;\mu) \equiv (J_N^1(t^\ell;\mu), \ldots, J_N^{26}(t^\ell;\mu))^T \in \mathbb{R}^{26}$. For convenience, we also denote $J^{\mathcal{N}}(t^\ell;\mu) \equiv (\frac{\partial s^{\mathcal{N}}}{\partial \mu_1}(t^\ell;\mu), \ldots, \frac{\partial s^{\mathcal{N}}}{\partial \mu_{26}}(t^\ell;\mu))^T$.

[6] We note that direct and finite-difference approximations for sensitivity approximations both require $O(P)$ forward solves. Since we have $P = 26$ parameters here, dual-based sensitivity calculations are certainly preferable.

[7] To mitigate this effect we could enrich our RB space with derivative information to obtain a Hermite RB space [31], but we do not consider this here.

We examine the accuracy of the RB sensitivity derivative approximation using the $n_{\text{train}} = 10^6$ primal and dual RB spaces, with $N = 100$. We generate a second test set, Ξ_2^{test}, of 30 randomly selected parameters in $\mathcal{D}$ on which to compare $J^{\mathcal{N}}$ and J_N; the sample set is necessarily relatively small due to the computational expense of the truth calculations. We note that the RB sensitivity approximations are obtained at very modest computational cost: each truth sensitivity calculation requires 663 seconds on 512 processors of Ranger (which includes a primal and dual truth solve and evaluation of (15) for $\ell = 1, \ldots, K$), whereas the corresponding RB calculation requires only 2.3 seconds on a single processor.

We consider two error metrics in our comparison: the mean relative error in the RB sensitivity derivatives

$$\mathcal{E}_J \equiv \frac{1}{K\,|\Xi_2^{\text{test}}|} \sum_{\mu_i \in \Xi_2^{\text{test}}} \sum_{k=1}^{K} \frac{\|J^{\mathcal{N}}(t^k;\mu_i) - J_N(t^k;\mu_i)\|_2}{\|J^{\mathcal{N}}(t^k;\mu_i)\|_2},$$

and the mean error in the normalized RB sensitivity derivatives:

$$\widehat{\mathcal{E}_J} \equiv \frac{1}{K\,|\Xi_2^{\text{test}}|} \sum_{\mu_i \in \Xi_2^{\text{test}}} \sum_{k=1}^{K} \|\widehat{J^{\mathcal{N}}(t^k;\mu_i)} - \widehat{J_N(t^k;\mu_i)}\|_2,$$

where $\widehat{J^{\mathcal{N}}(t^k;\mu)} \equiv J^{\mathcal{N}}(t^k;\mu)/\|J^{\mathcal{N}}(t^k;\mu)\|_2$, $\widehat{J_N(t^k;\mu)} \equiv J_N(t^k;\mu))/\|J_N(t^k;\mu)\|_2$ and $\|\cdot\|_2$ denotes the discrete ℓ_2 norm. Our test calculations yielded $\mathcal{E}_J = 0.33$ and $\widehat{\mathcal{E}_J} = 0.015$. This indicates that our RB sensitivity approximation does not capture the truth sensitivities accurately, but it does reproduce the normalized sensitivities very well — to within 2%. We would have to enrich our RB spaces further to reduce the error in $\mathcal{E}_J$, but in the present context we are in fact primarily interested in the normalized sensitivity derivatives since the components of $\widehat{J^{\mathcal{N}}}$ indicate the relative sensitivity of the output to the parameters. Hence, we exploit the fact that we have a cheap and accurate RB approximation, $\widehat{J_N} \approx \widehat{J^{\mathcal{N}}}$, to perform a statistical analysis of the normalized sensitivities in order to obtain a "global" view of the relative importance of the parameters. We compute sensitivity derivatives for a sample of 10,000 parameters in $\mathcal{D}$ and we show the mean, $\langle \widehat{J_N(t^k)} \rangle$, and standard deviation, $\sigma(\widehat{J_N(t^k)})$, at $t = 0.05$, $t = 0.25$ and $t = 1$ in Figures 12 and 13.

All components of the mean sensitivity derivatives in Figure 12 are negative, which is consistent with the fact that an increase in thermal conductivity leads to a decrease in output magnitude. We see from Figure 12a that at early times only 7 of the 26 parameters have a significant effect on the output, and that the output is most sensitive to parameters μ_1, μ_3, μ_4 and μ_9, which determine the conductivities in the regions adjacent to Ω_0. Figures 12b and 12c, however, show that at later times, once heat has diffused through the whole domain, the output sensitivity to parameters in subdomains further from Ω_0 becomes more significant and the sensitivities are overall much closer in magnitude. As steady state is approached, we see from Figure 12c that the output is most sensitive to μ_{13}; this is because subdomain 13 is the central cube in Ω and has the largest volume (see Figure 7a). Finally, we note that over our sample set the mean and standard deviation of $\widehat{J_N^2}$, $\widehat{J_N^5}$, $\widehat{J_N^6}$, $\widehat{J_N^7}$ and $\widehat{J_N^8}$ are small over the whole time interval. This suggests that $\mu_2, \mu_5, \mu_6, \mu_7$ and μ_8 would be the primary candidates to be eliminated from the problem if we sought to reduce model complexity. These parameters correspond to the five subdomains furthest away from Ω_0 in the bottom "layer" of Ω, so it is not surprising that they have relatively little impact on the output.

4. Conclusions

The certified Reduced Basis method provides a computational framework for the development of accurate reduced order approximations with rigorous error bounds for parametrized PDEs. However, the framework is quite elaborate and contains a number of subcomponents in both the Offline and Online stages. We have developed a high-performance object-oriented implementation of the certified Reduced Basis method that is flexible, extensible, efficient and user-friendly. There are many other extensions to the RB method that have not been discussed in detail here. Two notable examples, which are in fact provided in the RB extension to `libMesh`, are the Empirical Interpolation Method [21] for efficient treatment of non-affine

parametrized PDEs, and the *hp*-RB method [32, 33] which adaptively subdivides $\mathcal{D}$ in order to develop a family of "localized" RB spaces that lead to acceleration of the Online stage.

We demonstrated our RB implementation on two computationally expensive problems — a transient Boussinesq problem and a transient thermal conduction problem. In each case we demonstrated that our RB approximation provides a very large speedup and, equally importantly, a vast reduction in hardware requirements. We performed a detailed study of the thermal conduction problem because it is an archetypal example of a "many parameter" model reduction problem. We examined the parameter selection behavior of the Greedy algorithm, and we also gained interesting insights into the non-trivial "global" parametric sensitivities by leveraging the rapid response of the primal-dual RB sensitivity approximation to perform a statistical analysis on a large sample set. These numerical examples demonstrate that our high-performance implementation of the RB method enables efficient analysis of large-scale parametrized PDEs in real-time and many-query contexts, and therefore presents new opportunities for robust reduced order modeling in a range of different application areas.

A key goal of our future work is to employ Online RB approximations in deployed contexts in which we interface PDE-based models with experimental data in real-time. This could enable us to apply high-fidelity, three-dimensional PDE models in areas such as non-destructive testing, real-time control, or *in situ* parameter estimation, which are out of reach with classical PDE discretizations.

Acknowledgements

We would like to thank Prof. Anthony Patera and Dr. Phuong Huynh of MIT, and Jens Eftang of NTNU, for numerous helpful discussions related to this work. This research was supported by AFOSR Grant No. FA9550-07-1-0425, OSD/AFOSR Grant No. FA9550-09-1-0613, and also by the National Science Foundation through TeraGrid resources provided by TACC under Grant No. TG-ASC100016.

References

[1] Z. J. Bai, Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, Applied Numerical Mathematics 43 (1-2) (2002) 9–44.

[2] J. Chen, S.-M. Kang, Model-order reduction of nonlinear MEMS devices through arclength-based Karhunen-Loéve decomposition, in: Proceeding of the IEEE international Symposium on Circuits and Systems, Vol. 2, 2001, pp. 457–460.

[3] M. D. Gunzburger, J. Peterson, J. N. Shadid, Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data, Computer Methods in Applied Mechanics and Engineering 196 (2007) 1030–1047.

[4] M. Meyer, H. G. Matthies, Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods, Computational Mechanics 31 (1-2) (2003) 179–191.

[5] T. A. Porsching, M. Y. L. Lee, The reduced-basis method for initial value problems, SIAM Journal of Numerical Analysis 24 (1987) 1277–1287.

[6] J. P. Fink, W. C. Rheinboldt, On the error behavior of the reduced basis technique for nonlinear finite element approximations, Z. Angew. Math. Mech. 63 (1) (1983) 21–28.

[7] B. O. Almroth, P. Stern, F. A. Brogan, Automatic choice of global shape functions in structural analysis, AIAA Journal 16 (1978) 525–528.

[8] A. K. Noor, Recent advances in reduction methods for nonlinear problems, Comput. Struct. 13 (1981) 31–44.

[9] G. Rozza, D. B. P. Huynh, A. T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, Archives Computational Methods in Engineering 15 (3) (2008) 229–275, http://dx.doi.org/10.1007/s11831-008-9019-9.

[10] D. B. P. Huynh, D. J. Knezevic, J. W. Peterson, A. T. Patera, High-Fidelity Real-Time Simulation on Deployed Platforms, Computers & Fluids Special Issue: Proceedings of the Symposium on High Accuracy Flow Simulations (HAFS), École Polytechnique Fédérale de Lausanne, February 15–16, 2010, DOI: http://dx.doi.org/10.1016/j.compfluid.2010.07.007.

[11] M. A. Grepl, A. T. Patera, *A Posteriori* error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, M2AN (Math. Model. Numer. Anal.) 39 (1) (2005) 157–181.

[12] K. Veroy, C. Prud'homme, D. V. Rovas, A. T. Patera, *A Posteriori* error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations, in: Proceedings of the 16th AIAA Computational Fluid Dynamics Conference, 2003, paper 2003-3847.

[13] T. Lassila, G. Rozza, Parametric free-form shape design with PDE models and reduced basis method, Computer Methods in Applied Mechanics and Engineering 199 (23–24) (2010) 1583–1592, http://dx.doi.org/10.1016/j.cma.2010.01.007.

[14] G. R. Liu, K. Zaw, Y. Y. Wang, B. Deng, A novel reduced-basis method with upper and lower bounds for real-time computation of linear elasticity problems, Computer Methods in Applied Mechanics and Engineering 198 (2) (2008) 269–279, http://dx.doi.org/10.1016/j.cma.2008.07.011.

[15] B. S. Kirk, J. W. Peterson, R. M. Stogner, G. F. Carey, libMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations, Engineering with Computers 23 (3–4) (2006) 237–254.

[16] S. Balay, V. Eijkhout, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.

[17] 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://doi.acm.org/10.1145/1089014.1089019.

[18] D. B. P. Huynh, G. Rozza, S. Sen, A. T. Patera, A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants, C. R. Acad. Sci. Paris, Série I 345 (8) (2007) 473–478.

[19] B. Haasdonk, M. Ohlberger, Reduced basis method for finite volume approximations of parametrized evolution equations, M2AN (Math. Model. Numer. Anal.) 42 (2) (2008) 277–302.

[20] N. C. Nguyen, G. Rozza, A. T. Patera, Reduced basis approximation and a posteriori error estimation for the time-dependent viscous Burgers' equation, Calcolo 46 (3) (2009) 157–185, http://dx.doi.org/10.1007/s10092-009-0005-x.

[21] M. Barrault, Y.Maday, N. C. Nguyen, A. T. Patera, An "Empirical Interpolation" Method: Application to Efficient Reduced-Basis Discretization of Partial Differential Equations, C. R. Acad. Sci. Paris, Série I 339 (2004) 667–672.

[22] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Computer Methods in Applied Mechanics and Engineering 184 (2000) 501–520.

[23] K. Veroy, A. T. Patera, Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations; Rigorous reduced-basis *a posteriori* error bounds, International Journal for Numerical Methods in Fluids 47 (2005) 773–788.

[24] D. J. Knezevic, N. C. Nguyen, A. T. Patera, Reduced basis approximation and a posteriori error estimation for the parametrized unsteady Boussinesq equations, PREPRINT, Accepted to Mathematical Models and Methods in Applied Sciences. http://augustine.mit.edu/methodology/papers/atp_M3AS_preprint_Nov09.pdf.

[25] S. Sen, Reduced basis approximation and *a posteriori* error estimation for many-parameter heat conduction problems, Numerical Heat Transfer, Part B: Fundamentals 54 (5) (2008) 369–389.

[26] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331, http://dx.doi.org/10.1002/nme.2579.

[27] G. W. Stewart, Afternotes on Numerical Analysis, SIAM, Philadelphia, PA, 1996.

[28] B. Adams, W. Bohnhoff, K. Dalbey, J. Eddy, M. Eldred, D. Gay, K. Haskell, P. Hough, L. Swiler, Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual, Sandia technical report SAND2010-2183 (2009).

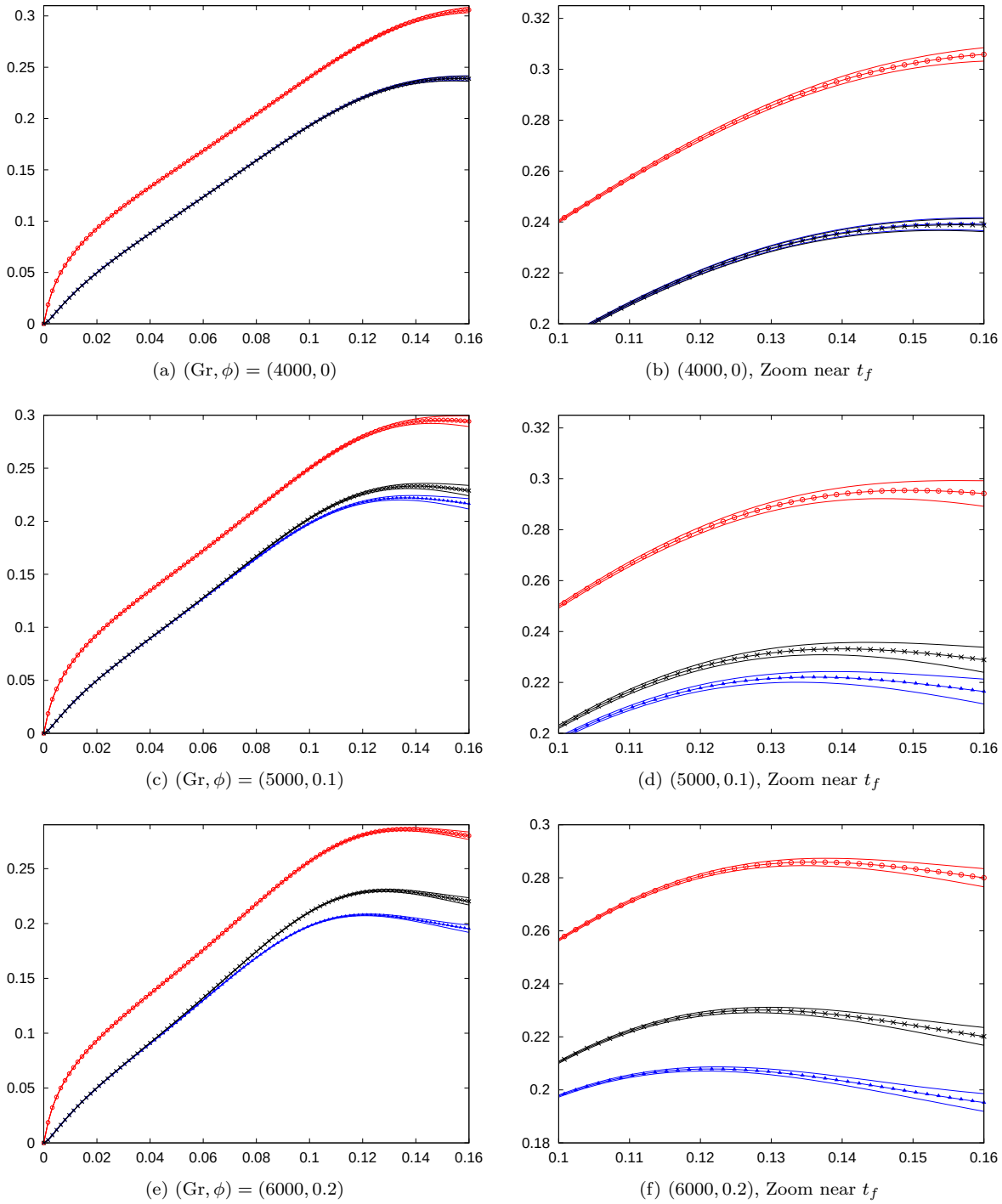
[29] T. R. Bewley, Flow control: new challenges for a new renaissance, Progress in Aerospace Sciences 37 (1) (2001) 21–58.

[30] A. Jameson, Aerodynamic design via control theory, J. of Scientific Computing 3 (1988) 233–260.

[31] K. Ito, S. S. Ravindran, A reduced-order method for simulation and control of fluid flows, Journal of Computational Physics 143 (2) (1998) 403–425.

[32] J. L. Eftang, A. T. Patera, E. M. Rønquist, An *hp* Certified Reduced Basis Method for Parametrized Elliptic Partial Differential Equations, Submitted to SISC (2009).

[33] J. L. Eftang, D. J. Knezevic, A. T. Patera, E. M. Rønquist, An *hp* certified reduced basis method for parametrized time-dependent partial differential equations, Submitted to MCMDS (2010).



(a) $(\mathrm{Gr}, \phi) = (4000, 0)$

(b) $(4000, 0)$, Zoom near t_f

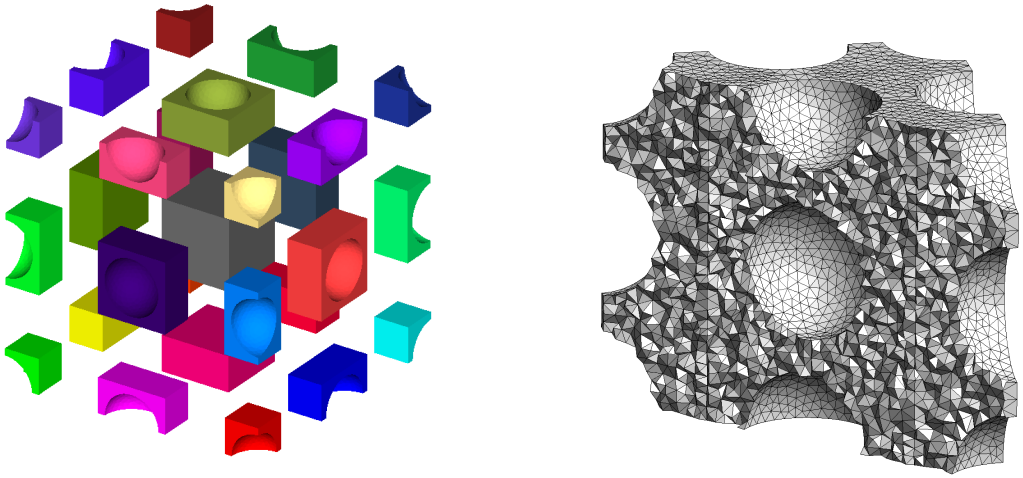
(c) $(\mathrm{Gr}, \phi) = (5000, 0.1)$

(d) $(5000, 0.1)$, Zoom near t_f

(e) $(\mathrm{Gr}, \phi) = (6000, 0.2)$

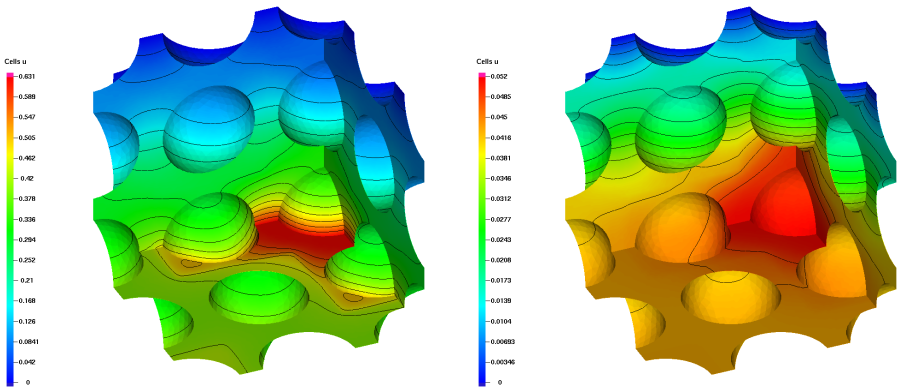
(f) $(6000, 0.2)$, Zoom near t_f

Figure 6: The RB outputs $s_{N,1}(t^k;\mu)$ (blue ▲), $s_{N,2}(t^k;\mu)$ (red ◯), $s_{N,3}(t^k;\mu)$ (black ×), and associated error bounds (solid lines) as functions of time for three values of μ.



(a) (b)

Figure 7: Subdomains (7a) for the "Swiss cheese" problem; each subdomain is associated with a distinct parametrized thermal conductivity. Interior cutaway view (7b) of the "Swiss cheese" mesh which is composed of linear tetrahedra and has 32,307 vertices and 168,926 elements. A once-uniformly-refined version of this grid which was used for the "truth" calculations has 241,520 vertices and 1,351,408 elements. All grids used in this work were created with the Gmsh [26] mesh generator.



(a) (b)

Figure 8: Primal (8a) and dual (8b) truth solutions at $t = 1$ and $t = 0$, respectively, for the parameter in (9).

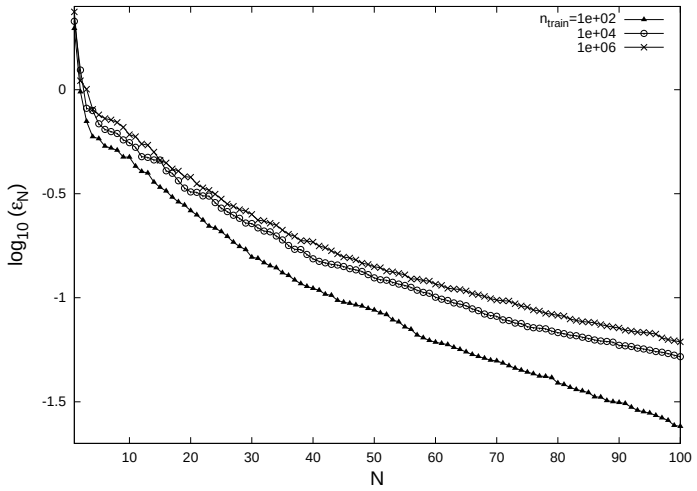
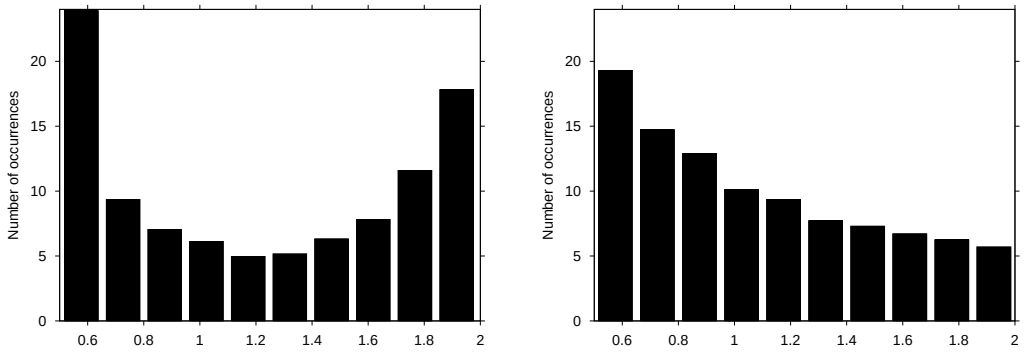
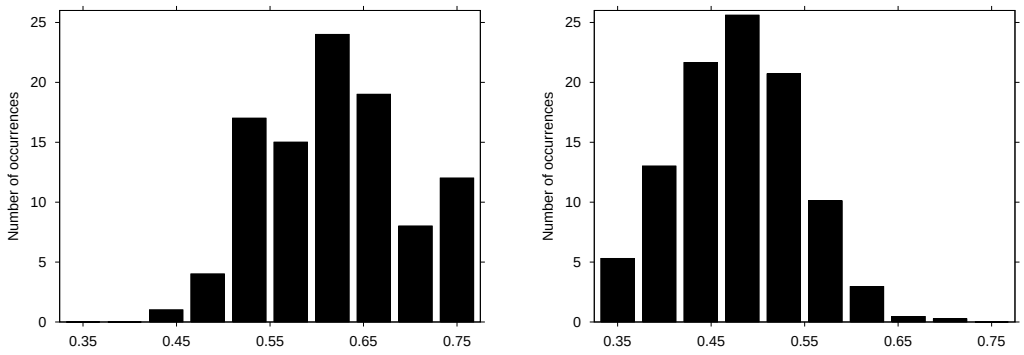


Figure 9: Greedy convergence for the primal basis for three different choices of n_{train}.



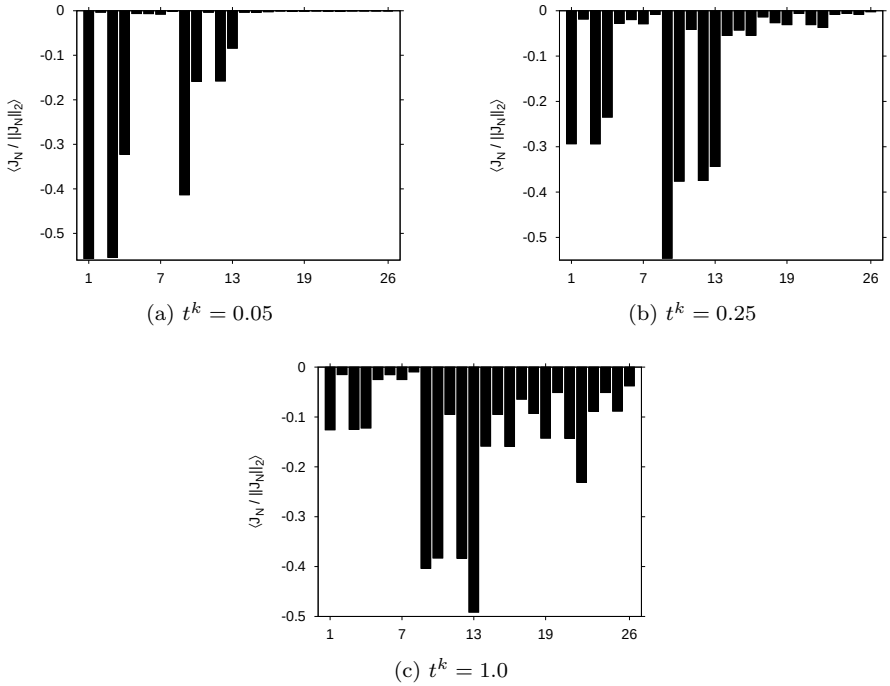
(a) Greedy-selected (b) Log-uniform random

Figure 10: Histograms of averaged greedy-selected (10a) and log-uniform randomly selected (10b) parameter values for the "Swiss cheese" problem with $n_{\text{train}} = 10^6$. The log-uniform histogram was created from a large set of sample data points and scaled to match the averaged greedy-selected $\langle\mu\rangle$ values. The bias of the greedy-selected values towards the endpoints 0.5 and 2 of the parameter domain relative to the log-uniform sample is evident.



(a) Greedy-selected (b) Log-uniform random

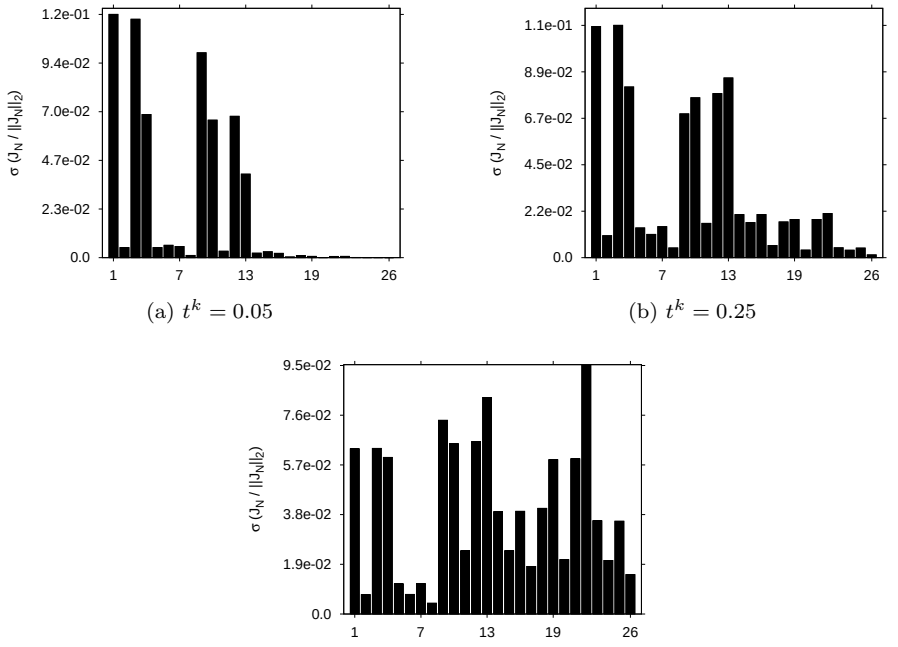
Figure 11: Histograms of average jumps $\langle \delta\mu \rangle$ from (12) in spatially-adjacent greedy-selected (11a) and log-uniform randomly selected (11b) parameter values for the "Swiss cheese" problem with $n_{\text{train}} = 10^6$. The log-uniform histogram was created from a large set of sample data points and scaled to match the averaged greedy-selected $\langle \delta\mu \rangle$ values.



(a) $t^k = 0.05$ (b) $t^k = 1.0$

(c) $t^k = 0.25$

Figure 12: $\langle \widehat{J_N(t^k)} \rangle$ for a sample of 10,000 random parameters in $\mathcal{D}$ at $t^k = 0.05$ (12a), $t^k = 0.25$ (12b), and $t^k = 1$ (12c).



(a) $t^k = 0.05$

(b) $t^k = 0.25$

(c) $t^k = 1.0$

Figure 13: $\sigma(\widehat{J_N(t^k)})$ for a sample of 10,000 random parameters in $\mathcal{D}$ at $t^k = 0.05$ (13a), $t^k = 0.25$ (13b), and $t^k = 1$ (13c).