High-Fidelity Real-Time Simulation on Deployed Platforms

D.B.P. Huynh[a], D.J. Knezevic[a], J.W. Peterson[b], A.T. Patera[a]

[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

We present a certified reduced basis method for high–fidelity real-time solution of parametrized partial differential equations on deployed platforms. Applications include *in situ* parameter estimation, adaptive design and control, interactive synthesis and visualization, and individuated product specification. We emphasize a new hierarchical architecture particularly well suited to the reduced basis computational paradigm: the expensive Offline stage is conducted pre–deployment on a parallel supercomputer (in our examples, the TeraGrid machine Ranger); the inexpensive Online stage is conducted "in the field" on ubiquitous thin/inexpensive platforms such as laptops, tablets, smartphones (in our examples, the Nexus One Android–based phone), or embedded chips. We illustrate our approach with three examples: a two–dimensional Helmholtz acoustics "horn" problem; a three–dimensional transient heat conduction "Swiss Cheese" problem; and a three–dimensional unsteady incompressible Navier-Stokes low–Reynolds–number "eddy–promoter" problem.

Keywords:
Reduced basis method, *a posteriori* error bounds, parametrized partial differential equations, high-performance computing, deployed platforms, real-time computing

1. Introduction

Many engineering applications require high-fidelity real-time simulation on deployed platforms "in the field." Examples include *in situ* parameter estimation and identification procedures, embedded adaptive design and control systems, virtual reality/synthesis and visualization environments (from music to medicine), and individuated context–dependent product specification frameworks. In all these cases the mathematical model must be sophisticated, the numerical approximation must be accurate, and the response to a query must be rapid — commensurate with real–time decision or interaction requirements — despite the limited processor power and storage capacity available in the field. We shall furthermore be interested in both input–output evaluation and visualization; the latter places additional demands on memory.

We shall suppose that the system input $\mu \in \mathcal{D} \subset \mathbb{R}^P$ enters as a parameter in a partial differential equation (PDE) which describes the relevant physical phenomena over the time interval of interest $0 \le t \le t_f$ and the appropriate spatial domain $\Omega \subset \mathbb{R}^d, d = 2$ or 3. This PDE, say a linear-time-invariant (LTI) parabolic equation, yields (*i*) a field variable over Ω, $u(t;\mu) \in X(\Omega)$ (where $X(\Omega)$ is an appropriate function space), and (*ii*) a scalar output of interest, $s(t;\mu) \in \mathbb{R}$, which can be expressed as a (say) linear functional of the field variable, $s(t;\mu) = \ell(u(t;\mu))$. (In actual practice we may consider many outputs.) Note that the parameter dependence proceeds from the PDE through the field variable and finally to the engineering output.

We shall distinguish between the *pre–deployment* period and the *post–deployment* or equivalently *deployed* period. The pre–deployment period takes place in the laboratory: we prepare the system and associated computational model for subsequent service. The deployed period takes place in the field: we put the system and associated computational model — now implemented on an embedded or more generally "deployed platform" — into service. In the deployed stage the computational task is well–prescribed: given a *query instance* $\mu' \in \mathcal{D}$ we wish to (*a*) predict the output, $\mu' \in \mathcal{D} \to (u(t^k;\mu') \to)\ s(t^k;\mu'), 0 \le k \le K$, and (*b*) visualize the field, $\mu' \in \mathcal{D} \to u(t^k;\mu')|_{\mathcal{R}}, 0 \le k \le K$; here $\mathcal{R} \subset \Omega$ is a region or manifold selected for rendering. (We reserve μ' to denote a query instance – a request post-deployment.) Note the field variable plays an important role both in input–output evaluation and of course in visualization.

To perform this computational task the PDE is typically discretized by a finite difference discretization in time and a finite element (FE) discretization in space. In time we consider a (say) Crank-Nicolson scheme

Email addresses: huynh@mit.edu (D.B.P. Huynh), dknez@mit.edu (D.J. Knezevic), peterson@tacc.utexas.edu (J.W. Peterson), patera@mit.edu (A.T. Patera)
URL: http://augustine.mit.edu/ (A.T. Patera)

associated to time levels $t^k = k\Delta t, 0 \leq k \leq K$, where $\Delta t = t_f/K$; in space we consider Galerkin projection over a FE approximation subspace $X^{\mathcal{N}}$ ($\subset X$) of large dimension $\mathcal{N}$. Our "truth" approximation is then given, for any $\mu \in \mathcal{D}$, by $u^{\mathcal{N}}(t^k;\mu), s^{\mathcal{N}}(t^k;\mu) = \ell(u^{\mathcal{N}}(t^k;\mu)), 0 \leq k \leq K$. We note that, given our restriction to $\mu \in \mathcal{D}$, all solutions of interest perforce reside on the parametrically induced manifold $\mathcal{M}^{\mathcal{N}} \equiv \{u^{\mathcal{N}}(t^k;\mu) \,|\, 0 \leq k \leq K,\ \mu \in \mathcal{D}\}$. We observe that this manifold is relatively low–dimensional; we can further anticipate, and in many cases demonstrate, that this manifold is smooth.

Our truth approximation shall provide, for sufficiently small Δt and in particular for sufficiently large $\mathcal{N}$, the desired accuracy. However, we cannot expect real-time response in particular on deployed platforms typically characterized by limited processor power and memory capacity. We thus pursue the certified reduced basis (RB) approach [1, 2, 3, 4, 5] *as an approximation to the truth approximation*. In time we directly inherit the Crank–Nicolson discretization of the truth; in space we consider Galerkin projection over an RB approximation space X_N ($\subset X^{\mathcal{N}}$) of small dimension N. Our RB approximation is then given, for any $\mu \in \mathcal{D}$, by $u_N(t^k;\mu), s_N(t^k;\mu) = \ell(u_N(t^k;\mu)), 0 < k \leq K$. We also provide rigorous *a posteriori* bounds, $\Delta_N(t^k;\mu)$ and $\Delta^s_N(t^k;\mu)$, for the error in the RB field approximation and the RB output approximation, respectively: for any $\mu \in \mathcal{D}$, $\|u^{\mathcal{N}}(t^k;\mu) - u_N(t^k;\mu)\|_X \leq \Delta_N(t^k;\mu)$, $|s^{\mathcal{N}}(t^k;\mu) - s_N(t^k;\mu)| \leq \Delta^s_N(t^k;\mu), 0 \leq k \leq K$. We may thus say that our RB approximation is *certified*.

The RB approximation space X_N is specifically designed to well approximate functions which reside on the parametrically induced manifold of interest, $\mathcal{M}^{\mathcal{N}}$: indeed, X_N is developed as the span of optimally selected (and combined) snapshots on the manifold $\mathcal{M}^{\mathcal{N}}$. (In contrast, even in a mesh–adaptive context, the truth FE approximation space $X^{\mathcal{N}}$ can represent a large class of functions very distant from $\mathcal{M}^{\mathcal{N}}$.) We can thus expect $N \ll \mathcal{N}$. The latter may in certain simple instances be proven, may in general be confirmed *a posteriori* through our error bounds, and may in practice be observed in a wide variety of problems. (Of course the "constants" will certainly depend on the particular problem under study and especially on the number of parameters, P.) This reduction in dimension *in conjunction* with an Offline–Online computational approach provides the RB advantage in the real–time deployed context. We now discuss the Offline–Online decomposition.

In the Offline stage we develop the RB space: we identify optimal (combinations of) snapshots on $\mathcal{M}^{\mathcal{N}}$ and we "precompute" various parameter–independent functionals of these snapshots implicated in subsequent RB approximations and associated RB error bounds; this Offline stage is expensive — $O(\mathcal{N}^\gamma)$ FLOPs, where γ is a problem-dependent factor related to the computational cost of the truth solves. The Offline stage yields a (problem-dependent) Online Dataset; this dataset is small — $O(Q, N)$ data, where Q measures the parametric complexity of our PDE. In the Online stage we invoke the Online Dataset to perform rapid certified output evaluation: given any $\mu' \in \mathcal{D}$ we calculate the RB output approximation and associated RB output error bound, respectively $s_N(t^k;\mu')$ and $\Delta^s_N(t^k;\mu'), 0 \leq k \leq K$. This Online stage is very inexpensive — $O(Q, N, K)$ FLOPs with $N \ll \mathcal{N}$. (In the next section we also discuss Online certified visualization; in this case the Online Dataset and Online operation count will depend on $\mathcal{N}$ and in particular on the number of FE degrees of freedom associated with $\mathcal{R}$. We note, however, that the visualization is a useful but optional step: the key quantities are the output(s) and associated error bound(s).)

We now associate the Offline stage to the pre–deployment period and the Online stage to the post–deployment period. The expensive Offline stage is conducted prior to deployment and hence the considerable Offline cost is not our principal concern. (Of course, control of the Offline cost is, in practice, very important; we discuss this further in the next section.) Only the inexpensive Online stage is invoked in the deployed period and hence only the very low Online cost will determine our primary performance metric — reliable and rapid response in the field. We may thus achieve our objective of high–fidelity real-time simulation on deployed platforms, as we now describe.

In the Online stage, the response to each query instance, $\mu' \in \mathcal{D} \rightarrow s_N(t^k;\mu'), \Delta^s_N(t^k;\mu'), 0 \leq k \leq K$, requires sufficiently few operations — $O(Q, N, K)$ FLOPs, independent of $\mathcal{N}$ — and sufficiently little data — $O(Q, N)$ storage for the Online Dataset, independent of $\mathcal{N}$ — to achieve *real-time* response on *deployed* ("thin") platforms. Furthermore, our rigorous error bound $\Delta^s_N(t^k;\mu'), 0 \leq k \leq K$, will guarantee the accuracy of the RB output prediction relative to the *high-fidelity* truth. (We emphasize that the error bound does *not* require appeal to $u^{\mathcal{N}}(t^k;\mu'), s^{\mathcal{N}}(t^k;\mu')$.) We thus obtain not just rapid, but also accurate, optimal, and safe decisions in the field.

In Section 2 we describe, for a simple model problem, the reduced basis approach. We emphasize the computational aspects: the RB approximation and associated RB *a posteriori* error estimation "kernels"; the procedure for identification of optimal RB approximation spaces; and the Offline, Online Dataset, and Online decomposition. In Section 3 we elaborate upon the Offline and Online procedures within a hierarchical architecture: we present the Offline procedure from a parallel perspective and describe a particular implementation on the TeraGrid supercomputer Ranger at the Texas Advanced Computing Center (TACC); we present the Online procedure from a deployed/embedded perspective and describe a particular implementation on a Nexus One Android phone "model platform." In Section 4 we present results for three examples: a frequency–domain acoustics problem in a two–dimensional horn configuration Ω — to illustrate the necessity of high–fidelity PDE models and accurate numerical solutions; a transient linear heat conduction problem in a three–dimensional "Swiss Cheese" configuration Ω — to illustrate treatment of many parameters; and a transient incompressible fluid flow problem in a three–dimensional sphere–in–duct configuration Ω — to illustrate extension to (quadratic) nonlinearities.

2. Certified Reduced Basis Formulation

2.1. Model Problem

We shall illustrate the approach for a very simple model problem. We consider steady heat conduction in a (say, polygonal) domain $\Omega = \Omega^1 \cup \Omega^2$: the normalized thermal conductivity in Ω^1 (respectively, Ω^2) is unity (respectively, κ). We apply a uniform unit heat source over the entire domain Ω. We require that the temperature field, u, vanish — zero Dirichlet conditions — on the domain boundary $\partial\Omega$. We consider a single ($P = 1$) parameter: $\mu \equiv \kappa$, the conductivity in Ω^2; $\mathcal{D}$, the parameter domain, is given by (say) the interval $[1, 10]$. We take for our output of interest, s, the integral of the temperature over Ω^1. (Note we may, for example, expand the model to include convection by a prescribed incompressible velocity field. Many other extensions are possible.)

In mathematical terms, $u(\mu) \in X$, where $X = H^1_0(\Omega)$; here $H^1_0(\Omega) = \{v \in H^1(\Omega) \,|\, v|_{\partial\Omega} = 0\}$, $H^1(\Omega) = \{v \in L^2(\Omega) \,|\, \nabla v \in (L^2(\Omega))^2\}$, and $L^2(\Omega)$ is the space of square integrable functions over Ω. We associate to the space X the inner product $(w, v)_X \equiv \int_\Omega \nabla w \cdot \nabla v$ and induced norm $\|w\|_X \equiv \sqrt{(w, w)_X}$ and to the space $L^2(\Omega)$ the inner product $(w, v) \equiv \int_\Omega wv$ and induced norm $\|w\| \equiv \sqrt{(w, w)}$. We then define the continuous and coercive bilinear forms $a^1 \equiv \int_{\Omega^1} \nabla w \cdot \nabla v$, $a^2 \equiv \int_{\Omega^2} \nabla w \cdot \nabla v$, and

$$a(w, v; \mu) \equiv a^1(w, v) + \mu a^2(w, v), \qquad \forall w, v \in X, \quad (1)$$

and the bounded linear forms $f(v) = \int_\Omega v$ and $\ell(v) = \int_{\Omega^1} v$. We can now provide the weak statement of our PDE: given $\mu \in \mathcal{D}$, find $u(\mu) \in X$ such that $a(u(\mu), v; \mu) = f(v)$, $\forall v \in X$; evaluate $s(\mu) = \ell(u(\mu))$. (We may readily accommodate several or even many outputs.)

We note that (1) is a special case of a more general hypothesis. We say that our bilinear form a is "affine in parameter" (or more precisely, "affine in functions of the parameter") if we can write

$$a(w, v; \mu) = \sum_{q=1}^{Q} \Theta^q(\mu) a^q(w, v), \quad (2)$$

where the $\Theta^q : \mathcal{D} \to \mathbb{R}$ are easily evaluated ($O(1)$ FLOPs) parameter–dependent coefficient functions and the a^q are parameter–independent continuous bilinear forms. Similar expansions may be developed for f and ℓ. The assumption (2) is a prerequisite for the crucial Offline–Online decomposition discussed in greater detail below.

In fact many problems admit a representation of the form (2) which can either be identified by inspection or, in the case of certain geometric variations, be constructed in an automated fashion [5]. (Note that the RB approximation is developed on a fixed reference domain Ω; geometric variations thus appear as coefficient functions which arise from the transformation of the actual parameter–dependent domain $\Omega^o(\mu)$ to Ω.) More generally, we can develop an affine representation which *approximates* the bilinear form a [6]; in certain cases we can further develop rigorous bounds to quantify the additional "consistency" errors introduced [7].

We next define the truth finite element (FE) approximation. We introduce a triangulation of Ω, $\mathcal{T}^{\mathcal{N}}$, to which we associate a standard first-order conforming polynomial FE approximation space $X^{\mathcal{N}}$. We can then provide the truth approximation: given $\mu \in \mathcal{D}$, find $u^{\mathcal{N}}(\mu) \in X^{\mathcal{N}}$ such that $a(u^{\mathcal{N}}(\mu), v; \mu) = f(v)$, $\forall v \in X^{\mathcal{N}}$; evaluate $s^{\mathcal{N}}(\mu) = \ell(u^{\mathcal{N}}(\mu))$. For future reference we denote by $\mathcal{V} \subset \{1, \ldots, \mathcal{N}\}$ the set of vertices of $\mathcal{T}^{\mathcal{N}}$ associated to a region or manifold $\mathcal{R} \subset \Omega$ over which we wish to visualize the field $u^{\mathcal{N}}(\mu)$.

2.2. Reduced Basis Kernels

We consider the primal-only reduced basis (RB) formulation. (In actual practice we often pursue primal–dual RB approximations, as described in detail in Section 11 of [5]: primal–dual approaches can be significantly more efficient both in the Offline and Online stages.) We shall assume that we are *given* $N_{\max}$ nested RB approximation spaces $X_1 \subset X_2 \subset \cdots X_N \cdots \subset X_{N_{\max}}$; here X_N is of dimension N. The RB approximation $u_N(\mu) \in X_N$ is expressed in terms of $(\,,\,)_X$-orthonormal basis functions $\{\zeta_i\}_{i=1,\ldots,N_{\max}}$:

$$u_N(x; \mu) = \sum_{j=1}^{N} \omega_{N\,j}(\mu) \zeta_j(x), \quad (3)$$

where x is a point in Ω. The $\zeta_i \in X^{\mathcal{N}}$ — orthonormalized snapshots from $\mathcal{M}^{\mathcal{N}}$ — are in practice piecewise-linear functions over Ω defined by the nodal values $\zeta_j(x_i)$, $1 \le i \le \mathcal{N}$, $1 \le j \le N$, where x_i denotes a vertex of the triangulation $\mathcal{T}^{\mathcal{N}}$. (Note that for visualization purposes $u_N(\mu)|_{\mathcal{R}} \in X^{\mathcal{N}}$ may be reconstructed solely in terms of $\zeta_j(x_i)$, $i \in \mathcal{V}, 1 \le j \le N$.) At the conclusion of this section we shall be in a position to succinctly summarize the algorithm by which these RB spaces — in particular, the underlying snapshots — are optimally selected.

We can now provide the RB Galerkin approximation: given $\mu \in \mathcal{D}$, find $u_N(\mu) \in X_N$ such that $a(u_N(\mu), v; \mu) = f(v)$, $\forall v \in X_N$; evaluate $s_N(\mu) = \ell(u_N(\mu))$. We know by Céa's Lemma that this approximation is optimal (in the energy norm): the Galerkin projection chooses the best linear combination of snapshots. We can further develop the corresponding matrix equations (inserting (3) into our weak form and testing on $v = \zeta_i, 1 \leq i \leq N$): given $\mu \in \mathcal{D}$, find $\omega_N(\mu) \in \mathbb{R}^N$, the solution to $A_N(\mu)\omega_N(\mu) = F_N$, and evaluate $s_N(\mu) = L^T\omega_N(\mu)$. Here $A_{N\,i,j}(\mu) = a(\zeta_j, \zeta_i; \mu), 1 \leq i, j \leq N$, is our stiffness matrix, $F_{N\,i} = f(\zeta_i), 1 \leq i \leq N$, is our load vector, and $L_{N\,i} = \ell(\zeta_i), 1 \leq i \leq N$, is our output vector. Note that A_N, F_N, and L_N are principal submatrices/subvectors of $A_{N_{\max}}$, $F_{N_{\max}}$, and $L_{N_{\max}}$, respectively, thanks to our hierarchical RB spaces.

We could simply calculate, for any given query instance μ', the matrix elements $A_{N\,i,j}(\mu') = a(\zeta_j, \zeta_i; \mu')$, $1 \leq i, j \leq N$, as N^2 integrations over Ω. However this approach is very expensive — $O(\mathcal{N}N^2)$ — and indeed perhaps even more expensive than direct calculation of the truth FE approximation $u^{\mathcal{N}}(\mu'), s^{\mathcal{N}}(\mu')$. Instead, and in anticipation of our Offline-Online strategy, we must partition the computation of $\omega_N(\mu')$ and $s_N(\mu')$ into two complementary components: a "construction" part which will be expensive — operation count $O(\mathcal{N})$ — but which will not depend on the query instance μ' and hence may be performed in the pre–deployment period; an "evaluation" part which will depend on the query instance μ' but which will be inexpensive — operation count $O(N)$, *not* $O(\mathcal{N})$ — and hence can be accommodated in the deployment period.

The key enabler is the "affine in parameter" structure of the bilinear form: (1) implies that $a(\zeta_j, \zeta_i; \mu) = a^1(\zeta_j, \zeta_i) + \mu a^2(\zeta_j, \zeta_i)$, $1 \leq i, j \leq N$, and hence that $A_N(\mu) = A^1_N + \mu A^2_N$, where $A^1_{N\,i,j} = a^1(\zeta_j, \zeta_i)$ and $A^2_{N\,i,j} = a^2(\zeta_j, \zeta_i)$, $1 \leq i, j \leq N$, are *parameter–independent* "proto"–stiffness matrices. (In the operation count and storage estimates developed below we shall extend this argument to the general affine expansions described by (2). In this case we will now have Q proto–stiffness matrices.)

The necessary construction-evaluation partition is now readily identified. In the construction part we compute (and store) the elements of A^1_N and A^2_N — in $O(\mathcal{N}N^2)$ FLOPs. In the evaluation part, for a given query instance μ', we first form $A_N(\mu') = A^1_N + \mu' A^2_N$ from (the stored) A^1_N and A^2_N — in $O(N^2)$ FLOPs; we then solve the small system $A_N(\mu')\omega_N(\mu') = F_N$ — in $O(N^3)$ FLOPs; finally we calculate $s_N(\mu') = L^T_N\omega_N(\mu')$ — in $O(N)$ FLOPs. (Note that $A_N(\mu)$ is small but, unfortunately, full.) The operation count for the evaluation part is independent of the truth resolution, $\mathcal{N}$.

We next provide, now given $u_N(\mu)$ (equivalently $\omega_N(\mu)$), the RB *a posteriori* error estimators. We first define the residual as $r(v; \mu) \equiv f(v) - a(u_N(\mu), v; \mu)$, $\forall v \in X^{\mathcal{N}}$. The Riesz representation of the residual $R(\mu) \in X^{\mathcal{N}}$ is given by

$$(R(\mu), v)_X = r(v), \qquad \forall v \in X^{\mathcal{N}}. \tag{4}$$

We may then introduce our error bounds $\Delta_N(\mu) \equiv \|R\|_X$ and $\Delta^s_N(\mu) \equiv C_\ell\|R\|_X$, where C_ℓ is a calculable constant (the dual norm of ℓ). Note that the X-norm error bound $\Delta_N(\mu)$ has the anticipated form: the dual norm of the residual divided by a lower bound for a stability (here coercivity, more generally inf-sup) constant; for our simple model problem a lower bound for the stability constant is unity, but in general this will not be the case.

It can be readily demonstrated that for any μ in $\mathcal{D}$ (and for any N, $1 \leq N \leq N_{\max}$), $\|u^{\mathcal{N}}(\mu) - u_N(\mu)\|_X \leq \Delta_N(\mu)$ and $|s^{\mathcal{N}}(\mu) - s_N(\mu)| \leq \Delta^s_N(\mu)$. We can further provide an effectivity result: for any μ in $\mathcal{D}$ (and for any N, $1 \leq N \leq N_{\max}$), $\Delta_N(\mu)/\|u^{\mathcal{N}}(\mu) - u_N(\mu)\|_X \leq 10$; our error bound is thus rigorous and "sharp." (The effectivity, here 10, will more generally depend indirectly on the parameter domain $\mathcal{D}$ and directly on the coercivity and continuity constants associated with the bilinear form a.)

In anticipation of our Offline-Online strategy we must now find a partition of the computation of $\|R\|_X$ into construction and evaluation components. Towards that end, we insert the residual expression into (4) and invoke the affine structure of our bilinear form, (1), and our representation (3), to obtain

$$\begin{aligned}(R(\mu), v)_X = f(v) &- \sum_{i=1}^{N} \omega_{N\,i}(\mu) a^1(\zeta_i, v) \\ &- \sum_{i=1}^{N} \mu\omega_{N\,i}(\mu) a^2(\zeta_i, v), \qquad \forall v \in X^{\mathcal{N}}\ .\end{aligned}$$

We then apply linear superposition to express $R(\mu)$ as

$$R(\mu) = \sum_{i=1}^{2N+1} \Phi_i(\mu)\mathcal{G}_i\ , \tag{5}$$

for Φ_i: $\mathcal{D} \to \mathbb{R}$, $\mathcal{G}_i \in X^{\mathcal{N}}$, $1 \leq i \leq 2N + 1$.[1] Note that the "Riesz's pieces" $\mathcal{G}_i$, $1 \leq i \leq N$, are *independent* of μ. Finally, from (5) it follows that

$$\Delta_N(\mu) = \sqrt{\Phi^T(\mu)\Lambda\Phi(\mu)}, \quad \Delta^s_N(\mu) = C_\ell\Delta_N(\mu)\ , \tag{6}$$

where $\Lambda_{i,j} \equiv (\mathcal{G}_i, \mathcal{G}_j)_X$, $1 \leq i, j \leq 2N + 1$.

The construction-evaluation partition is then clear: in the construction part we form and store the μ'-independent quantity Λ (and C_ℓ) — in $O(\mathcal{N}N^2)$

[1] We provide the explicit form: $\Phi_1 = 1$, $\Phi_2(\mu) = \omega_{N\,1}(\mu)$, $\Phi_3(\mu) = \omega_{N\,2}(\mu), \ldots, \Phi_{N+2} = \mu\omega_{N\,1}(\mu), \ldots, \Phi_{2N+1} = \mu\omega_{N\,N}(\mu)$; $\forall v \in X^{\mathcal{N}}$, $(\mathcal{G}_1, v)_X = f(v)$, $(\mathcal{G}_2, v)_X = -a^1(\zeta_1, v)$, $(\mathcal{G}_3, v)_X = -a^1(\zeta_2, v), \ldots$, $(\mathcal{G}_{N+2}, v)_X = -a^2(\zeta_1, v), \ldots$, $(\mathcal{G}_{2N+1}, v)_X = -a^2(\zeta_N, v)$.

FLOPs; in the evaluation part we calculate the error bound from (6) — in $O(4N^2)$ FLOPs. (In general, the stability constant cannot be deduced by inspection but rather must be calculated by an Offline–Online "Successive Constraint Method" (SCM) [8]: the SCM Offline stage is rather onerous; however, the SCM Online cost depends only on Q and is typically negligible compared to the RB Online cost.)

We can encapsulate the computational tasks associated with RB approximation and RB error estimation in the following procedures. Construction is represented as

$$[\mathcal{S}^{\text{eval}}] = \texttt{Construction}(\{\zeta_n\}_{n=1,\ldots,N_{\max}}), \qquad (7)$$

where $\mathcal{S}^{\text{eval}} \equiv \{A^1_{N_{\max}}, A^2_{N_{\max}}, F_{N_{\max}}, L_{N_{\max}}, \Lambda, C_\ell\}$. Evaluation is represented as

$$[\omega_N(\Xi), \Delta_N(\Xi), s_N(\Xi), \Delta^s_N(\Xi), \mu^*] = \texttt{Evaluation}(\Xi; N; \mathcal{S}^{\text{eval}}), \qquad (8)$$

where $\Xi \subset \mathcal{D}$ is either a single value or a set of values of the parameter, and $\mu^* = \arg\max_{\mu\in\Xi} \Delta_N(\mu)$ (which will serve in the Greedy Procedure below).

We now summarize the operation counts under the general affine hypothesis (2) and for both linear elliptic PDEs and LTI parabolic PDEs. In the elliptic case, $K = 1$; in the LTI parabolic case, K denotes the number of time levels (note that `Evaluation` returns the RB approximation and associated error bounds for all time levels, $1 \leq k \leq K$). The operation count for `Construction` is $O(\mathcal{N}QN^2 + \mathcal{N}Q^2N^2 + \mathcal{N}^\gamma QN)$; there is no explicit dependence on K, though certainly for more complex temporal dependence we will require larger N for any given error tolerance. The storage requirement for $\mathcal{S}^{\text{eval}}$ is $O(QN^2 + Q^2N^2)$. The operation count for `Evaluation` is $O(QN^2 + N^3 + KN^2 + Q^2N^2)$; there is no explicit dependence on P, though in general larger P (more parameters) will require larger N for any given error tolerance.

2.3. The Greedy Procedure

To identify our snapshots and hence our RB spaces $X_N, 1 \leq N \leq N_{\max}$, we apply the Greedy Procedure of Algorithm 1. (In the case of parabolic PDEs this Greedy Procedure must be replaced with a POD(t)–Greedy(μ) Procedure [9].) The spaces are constructed sequentially: at each iteration we append the snapshot from $\mathcal{M}^{\mathcal{N}}_{\Xi^{\text{train}}} \equiv \{u^{\mathcal{N}}(\mu) \,|\, \mu \in \Xi^{\text{train}}\}$ which is *least* well represented by the current RB approximation space as measured by the RB field approximation error bound. Note that in Line 8 I is the identity operator and Π_{X_N} is the orthogonal projection onto X_N with respect to the $(\ ,\ )_X$ inner product.

The operation count for the Greedy Procedure (for elliptic PDEs) is $O(\mathcal{N}^\gamma N_{\max}) + O(\mathcal{N}^\gamma QN_{\max}) + O(\mathcal{N}Q^2N^2_{\max}) + O(n^{\text{train}}QN^3_{\max}) + O(n^{\text{train}}Q^2N^3_{\max}) + O(n^{\text{train}}N^4_{\max})$. The first term relates to the calculation of the snapshots — Line 8; note $\gamma \geq 1$ relates to the efficiency of the truth solution procedure. The remainder of the terms relate to Line 10 and Line 11 — the operation counts associated with "integration" of `Construction` and `Evaluation` from $N = 1$ to $N = N_{\max}$. We make two crucial observations: the update snapshot is determined by the error bound and not the true error — the truth solution is calculated, in Line 8, only for the winning "arg max" candidate, deduced in Line 11; as $n^{\text{train}} \to \infty$, the cost of the "many query" calculation $\Delta_N(\Xi^{\text{train}})$ is determined solely by the inexpensive `Evaluation` — the expensive `Construction` is asymptotically negligible. This strategy ensures $O(n^{\text{train}}) + O(\mathcal{N})$ rather than $O(n^{\text{train}}\mathcal{N})$ complexity and thus permits consideration of very large train sets Ξ^{train} for which $\mathcal{M}^{\mathcal{N}}_{\Xi^{\text{train}}}$ will indeed be a good surrogate for the actual manifold of interest $\mathcal{M}^{\mathcal{N}} \equiv \{u^{\mathcal{N}}(\mu) \,|\, \mu \in \mathcal{D}\}$.

In actual practice, the Greedy Procedure can identify very effective RB spaces. Large train samples — enabled by our error bound "importance sampling" strategy and `Construction`–`Evaluation` many–query procedure — are of course imperative in particular for larger P.

Algorithm 1 Greedy Algorithm.

```
1:  specify $\Xi^{\text{train}} \subset \mathcal{D}$ of size $n^{\text{train}}$ and tolerance $\epsilon$.
2:  set $N = 1$, $X_1 = u^{\mathcal{N}}(\mu_1)$ ($\mu_1$ arbitrary in $\mathcal{D}$).
3:  set $\zeta_1 = u^{\mathcal{N}}(\mu_1)$ (normalized: $\|\zeta_1\|_X = 1$).
4:  set $\mathcal{S}^{\text{eval}}$ = Construction($\zeta_1$).
5:  set [.,err,.,.,$\mu^*$] = Evaluation($\Xi^{\text{train}}$; 1; $\mathcal{S}^{\text{eval}}$).
6:  while [err] > $\epsilon$ do
7:      $\epsilon_N$ = err (for "reporting" purposes);
8:      $\zeta_{N+1} = (I - \Pi_{X_N})u^{\mathcal{N}}(\mu^*)$ (normalized);
9:      $N \leftarrow N + 1$;
10:     $\mathcal{S}^{\text{eval}}$ = Construction($\{\zeta_i\}_{i=1,\ldots,N}$) (update);
11:     set [.,err,.,.,$\mu^*$] = Evaluation ($\Xi^{\text{train}}$, $N$, $\mathcal{S}^{\text{eval}}$);
12: end while
13: set $N_{\max} \leftarrow N$.
```

2.4. Offline–Online Decomposition

The Offline–Online Decomposition is now readily identified. In the process, we also now include the visualization aspect of our approach.

The *Offline* stage is simply the Greedy Procedure. The Greedy Procedure (through `Construction`) yields $\mathcal{S}^{\text{eval}}$.

The *Online Dataset* is then $\{\mathcal{S}^{\text{eval}}, \mathcal{S}^{\text{vis}}\}$, where $\mathcal{S}^{\text{vis}} \equiv \zeta_j(x_i), i \in \mathcal{V}, 1 \leq j \leq N$. The Online Dataset, for either linear elliptic PDEs or LTI parabolic PDEs, is of size $O(QN^2 + Q^2N^2) + O(|\mathcal{V}|N)$, where $|\mathcal{V}|$ denotes the cardinality of the vertex set associated with the desired visualization region $\mathcal{R}$. The first contribution is due to $\mathcal{S}^{\text{eval}}$ and is independent of $\mathcal{N}$. The second contribution is due to $\mathcal{S}^{\text{vis}}$ and of course is not independent of $\mathcal{N}$; however, in practice we envision selective rendering and hence $|\mathcal{V}| \ll \mathcal{N}$ — for example $|\mathcal{V}| \approx c\mathcal{N}^{2/3}, c \ll 1$, for $\mathcal{R}$ a small two–dimensional manifold within a three–dimensional domain Ω.

The *Online* stage comprises certified output evaluation and visualization for any given query instance $\mu' \in \mathcal{D}$. Certified output evaluation — calculation of the RB output prediction and associated RB output *a posteriori* error bound — is effected by $[\omega_N(\mu'), \Delta_N(\mu'), s_N(\mu'), \Delta_N^s(\mu'), .] = \texttt{Evaluation}(\mu'; N; \mathcal{S}^{\text{eval}})$; as described in detail earlier, the operation count is independent of $\mathcal{N}$. Certified visualization is then effected by

$$[\texttt{figure}, \Delta_{N,\Gamma}(\mu')] = \texttt{Visualization}(\omega_N(\mu'), \mathcal{S}^{\text{vis}}), \tag{9}$$

where `figure` is a rendering based on

$$u_N(x_i; \mu') = \sum_{i=1}^{N} \omega_{N\,j}(\mu')\zeta_j(x_i), i \in \mathcal{V},$$

and $\Delta_{N,\Gamma}(\mu')$ is an $L^2(\Omega)$ error bound for $u_N(\mu)|_{\mathcal{R}}$. (Under suitable hypotheses on $\mathcal{R}$ we may form $\Delta_{N,\Gamma}(\mu') = C_\Gamma \Delta_N(\mu')$, where C_Γ may be calculated Offline.) The operation count for `Visualization` is $O(|\mathcal{V}|KN)$ (for all time levels K in the LTI parabolic case).

3. Hierarchical Architecture

We have previously identified the pre-deployment period with the Offline stage and the deployment period with the Online stage. Now we further identify the Offline stage with a parallel supercomputer (as warranted) and the Online stage with a "thin" platform: the former has the necessary speed and memory to address the large truth calculations required; the latter — with minimal processing power and memory capacity — is of the reduced physical size and weight, and cost, to permit service (often at many "sites" or "installations") in the field.

The Offline stage admits very efficient treatment on a massively parallel machine [10]. There are two different (heterogeneous) opportunities for data–parallelism. First, in Line 8 of the Greedy Procedure, we can exploit classical parallelism in space for calculation of the truth solution: the domain Ω is broken into n_{proc} smaller subdomains (and associated sets of finite element degrees of freedom) which are in turn distributed to n_{proc} processors. Second, in Line 11 of the Greedy Procedure, we can exploit parallelism in parameter for calculation of the "arg max": the "domain" Ξ^{train} is broken into n_{proc} partitions (and associated sets of parameter values) which are in turn distributed to n_{proc} processors. In the first case, the calculations on different processors are of course coupled and hence communication and granularity will impact parallel efficiency; in the second case, the calculations on different processors are independent — (copies of) the same RB model executing different data — and thus, as in Monte–Carlo simulations, the parallel efficiency will be very high. These algorithms are implemented [10] on (a subset of) the Ranger supercomputer, which has in total 62,976 cores, 123TB of memory, and a theoretical peak performance of 579 TFLOPs.

The Online stage can be hosted on many different small and inexpensive "deployed" platforms from laptops and tablets to smartphones and embedded chips. The "App" software is essentially the `Evaluation` code and the `Visualization` code. The Online Dataset for any particular problem — required by `Evaluation` and `Visualization` — can either be stored on the deployed platform or downloaded over the Internet: the former would be relevant for "in the loop" embedded applications such as detection or control; the latter might be relevant to more general but still "interactive" environments such as education or design. In actual practice more difficult problems (larger N, Q) and in particular more extensive visualization (larger $|\mathcal{V}|$) may not be feasible on very thin deployed platforms since the Online Dataset will be too large. At present the "App" is implemented in the Java programming language and executed on the Nexus One Android–based smartphone — a "model" deployed platform — with a 1GHz processor and 512MB of memory with double-precision accuracy; the OpenGL ES library is invoked for graphical renderings. Note that the App has access to a limited subset of the total memory which certainly restricts the range of problems and parameters that may be considered.

In this context we briefly describe an enhancement to our current approach which simultaneously can extend both Offline parallel performance and Online deployed capability: "hp" reduced basis approaches [11]. In the hp approach, in a new Offline "h" *pre*–preprocessing step we (optimally) decompose the parameter domain $\mathcal{D}$ into smaller parameter subdomains; we then pursue the standard (in fact, "p") certified reduced basis approach in each subdomain — Offline Greedy Procedure, and Online `Evaluation` and `Visualization`. The hp approach offers new opportunities for Offline (embarrassing) parallelism: the standard "p" certified reduced basis procedures can be pursued in parallel on each parameter subdomain. And even more importantly, the hp approach provides a mechanism for consideration of larger problems on deployed platforms: we may download only the data associated with a particular parameter subdomain or subdomains, and implement "parameter caching" notions to anticipate future parameter (subdomain) requests.

Finally, we note that although we implement the Offline and Online stages on vastly different machines with very different capabilities, the end result is in some sense machine–agnostic: the smartphone (low-order reduced basis) prediction is, to within

$\pm\Delta_N^s(\mu)$, indistinguishable from the parallel supercomputer (high–fidelity finite element) calculation. We can thus say that we *effectively* achieve "supercomputing on a phone." Absent rigorous error bounds no such equivalence can be claimed.

4. Numerical Examples

We first consider the Helmholtz equation and in particular the planar acoustics horn problem described in detail in [12]. The full domain Ω^o is depicted in Figure 1(a) and comprises both the horn proper as well as a large circular segment on which second–order radiation conditions [13] are applied. The (symmetric about $y = 0$) horn domain Ω^o is defined by the location of the top wall $y_{\mathrm{horn}}(x), 0 \le x \le 10$: $y_{\mathrm{horn}}(x) = 1/2$ for $0 \le x \le 5$; $y_{\mathrm{horn}}(x) = (1/5)(7.5 - x) + (2\eta/5)(x - 5)$ for $5 \le x \le 7.5$; $y_{\mathrm{horn}}(x) = (2\eta/5)(10-x)+6/5(x-7.5)$ for $7.5 \le x \le 10.0$; here η is a parameter that defines the problem geometry. Note we state the problem in nondimensional form: all lengths are scaled by the "inlet" channel width $\tilde{w}$; all pressures are scaled by a nominal input pressure p_0^o (note we replace our independent variable u^o with p^o to avoid confusion).

The governing equations for the (complex) pressure are then

$$-\nabla^2 p^o - k^2 p^o = 0, \quad \text{in } \Omega^o,$$

where $k = \tilde{\omega}\tilde{H}/\tilde{c}$ for $\tilde{\omega}$ the angular frequency and $\tilde{c}$ the speed of sound. We impose a right–traveling wave condition at the inlet Γ_{in}, $ikp^o + \partial p^o/\partial n = 2ik$, homogeneous Neumann conditions on the horn walls, $\partial p^o/\partial n = 0$, and second order radiation conditions on the circular farfield boundary. (Here n denotes outward normal.) We choose two parameters, $\mu_1 = \eta, \mu_2 = k$, and specify the parameter domain $\mathcal{D} = [1.7, 1.8] \times [0, 2]$. (A larger parameter domain is readily accommodated by the reduced basis approach — but not by our current deployed platform.) The output of interest is the modulus of the reflection coefficient,

$$s(\mu) = \left| \int_{\Gamma_{\mathrm{in}}} p^o dy - 1 \right|;$$

note that $1 - s$ can be related to the fraction of the wave energy transmitted to the surroundings [14]. Finally, for our "truth" we take a second order FE approximation with 14,935 degrees of freedom. The horn problem serves to demonstrate the need for high-fidelity models and accurate numerical calculations: only (accurate solutions to) the PDE can reproduce the detailed acoustic response.

This horn problem stretches the basic formulation of Section 2 in several ways. First, the field variable is now complex. Second, the problem is elliptic but not coercive, which in particular complicates the calculation of the (inf-sup) stability constant required for our rigorous error bounds [15].[2] Third, the output can be derived from a linear functional but in fact is not a simple linear functional of the field. And fourth, the problem is posed on a parameter–dependent domain $\Omega^o(\eta)$: upon piecewise–affine mapping to the reference domain $\Omega \equiv \Omega^o(\eta = 1.75)$ we recover the necessary form (2) with $Q = 11$ terms. Note that visualization is in general performed with respect to the *actual* (possibly parameter–dependent) domain $\Omega^o(\mu)$.

In this two–dimensional case the Offline calculation does not require a large machine and hence the Offline stage is performed on a standard desktop. (Obviously three–dimensional versions of this same problem in particular for higher k would be demanding given the oscillatory structure of the truth solution.) We shall thus focus on the Online results as provided by the Nexus One implementation; in all cases we consider $N = N_{\max} = 53$. We show in Figure 1(a) the `Visualization` of the real part of the pressure for $\eta = 1.5$ and $k = 1.75$. Note that for two–dimensional problems we often can and hence do render the entire field (in which case $|\mathcal{V}| = 3832$: here we interpolate the field onto a first order mesh to reduce $|\mathcal{V}|$) even on relatively thin platforms. We next present (though in practice compute first) in Figure 1(b) the `Evaluation` of our output for Ξ given by $(\eta = 1.5, k_j)_{j=1,\ldots,40}$: the middle curve is $s_N(\Xi)$ while the bottom and top curves represent $s_N(\Xi) - \Delta_N^s(\Xi)$ and $s_N(\Xi) + \Delta_N^s(\Xi)$, respectively. We emphasize the importance of the error bounds in ensuring not just rapid response but also reliable response: *the truth output must reside between the bottom and top curves*. Finally, we note that both Figure 1(a) and Figure 1(b) are direct "screen grabs" from our Nexus One implementation. Also we note that in this example (and the subsequent examples below) the Online computation time on the phone only takes one or two seconds.

We next consider the heat equation — transient thermal conduction — in the complex three–dimensional "Swiss Cheese" configuration. 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 respective centers $[0, 1/2, 1] \times [0, 1/2, 1]$. (Note we directly consider the nondimensional version of the problem based on the standard conduction scalings.) For future reference we describe

$$\overline{\Omega} = \cup_{j=1}^{8} \overline{\Omega}_j,$$

where $\Omega_j, 1 \le j \le 8$, is the intersection of Ω with the 8 octants given by $[0, 1/2]^3, [1/2, 1] \times [0, 1/2]^2, \ldots, [1/2, 1/2]^3$.

The governing equations are most easily described

[2] On the phone we resort to a shortcut: the inf-sup constant for the Online stage is replaced by the minimum inf-sup lower bound over a dense sample in $\mathcal{D}$ calculated in the Offline stage.

in weak form: find $u(t;\kappa)$ for $0 \leq t \leq t_f \equiv 1$ such that

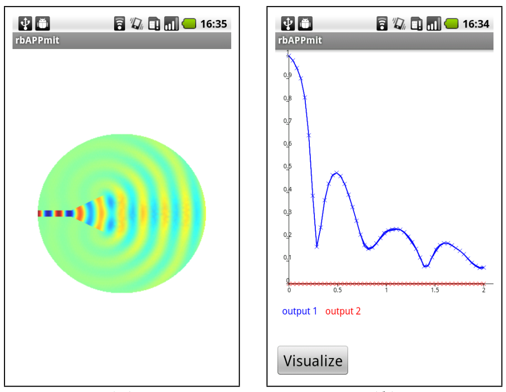
$$\int_\Omega \frac{\partial u}{\partial t} v + \sum_{j=1}^{7} \kappa_j \int_{\Omega_j} \nabla u \cdot \nabla v + \int_{\Omega_8} \nabla u \cdot \nabla v = \int_{\Gamma_\mathrm{b}} v, \forall v \in X, \tag{10}$$

where κ_j is the thermal conductivity of region Ω_j relative to the thermal conductivity of region Ω_8 (note we assume uniform specific heats and hence the κ_j may also be interpreted as scaled thermal diffusivities), $X \equiv \{v \in H^1(\Omega) \,|\, v_{\Gamma_\mathrm{t}} = 0\}$, 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). Note that we impose homogeneous natural, in this case zero flux, conditions on all other boundaries. We choose $P = 7$ parameters, $\mu_j = \kappa_j, 1 \leq j \leq 7$, and specify the parameter domain $\mathcal{D} = [0.5, 2]^7$. We consider several outputs corresponding to averages of the temperature over small cubes; for example, the first output is the average temperature over $[0, 1/4]^3$. Finally, we select a backward Euler temporal discretization with $K = 100$ time levels and a $\mathbb{P}_1$ truth finite element approximation space with $\mathcal{N} = 241{,}520$ degrees of freedom. This example is intended to illustrate the extension of the reduced basis formulation to parabolic PDEs but also the consideration of three–dimensional spatial domains and the treatment of many parameters; the latter increase the burden of both the Offline and Online stages.

The RB treatment of parabolic PDEs parallels quite closely the RB treatment of elliptic PDEs presented in Section 2. We elaborate briefly here on the features particular to RB treatment of LTI parabolic PDEs: The affine hypothesis on bilinear (and linear) forms remains in force; we deduce from inspection of the weak form (10) that for our example we require one proto mass matrix and $Q = 8$ proto stiffness matrices. The RB Galerkin approximation requires little modification from the elliptic case in particular since the RB temporal discretization is inherited from the truth. The RB error bounds are, as in the elliptic case, derived from classical stability arguments, however now more care must be exercised in definition of the proper (spatio-temporal) norms; the latter, in turn, affects the class of output functionals that we may consider — for example, we can provide pointwise bounds in time only for $L^2(\Omega)$ output functionals. The RB spaces are now obtained by a *POD*(t)–Greedy(μ) Procedure [9]: for each μ^* identified by the Greedy(μ), a single POD mode — associated with the *error* in the RB approximation — is appended to the RB space. (Note also that we may "train" on an impulse function in order to treat with a single RB approximation any control function specified in the Online (deployed) stage.) Finally, the operation counts for *LTI* parabolic PDEs will typically scale as *less* than K times the elliptic effort, as summarized in the estimates provided in Section 2.

In this example the Offline burden is considerable and the parallel implementation (here, on Ranger and based on the libMesh [16] library) important both as regards the computation of the truth time series (from which we extract our POD mode) — parallelized over Ω — and the "arg max" over the extensive Ξ_train sample required by the many parameters (large P) — parallelized over Ξ^train. We present in Figure 2 the POD-Greedy convergence, as measured by ϵ_N, for a train sample of size 100,000; computational (wall–clock) time on 512 cores is 3.5 hours. (Note that in this context we may interpret ϵ_N as a bound for the maximum over μ in Ξ^train of the maximum over $k = 1, \ldots, K$, of $\|u^\mathcal{N}(t^k;\mu) - u_N(t^k;\mu)\|_{L^2(\Omega)}$.) We now proceed to the Online stage: we present in Figure 3 the `Visualization` for $\mu' = (2, 2, 0.5, 2, 1.31, 1.745, 1.85)$ at the final time level $k = K$ as a screen grab from the Nexus One implementation. Note that for this three–dimensional configuration we can only afford (given phone memory limitations) to render the field variable over a "small" two–dimensional manifold: the outer surface of the domain.[3]

Finally, we consider the time–dependent incompressible Navier-Stokes equations at relatively low Reynolds number — here we employ methodology from [17]. We consider pressure–driven flow through a duct $(x, y, z) \in [0, 3] \times [0, 1]^2$ which contains a stationary solid sphere S of radius 0.1 with center $(x = 1/2, y = 1/2, z = 1/2)$; our domain Ω is thus given by $[0, 3] \times [0, 1]^2 \setminus S$. (In fact, we impose periodic boundary conditions in x, and hence we implicitly consider



(a) (b)

Figure 1: (a) Visualization on $\Omega^o(\mu')$ of the real part of the pressure for $\mu' = (1.75, 2.0)$. (b) Modulus of the reflection coefficient for $(\eta = 1.75, k_j \in [0, 2])_{j=1,\ldots,50}$. We show the RB outputs (marked with $\times$ symbols) and corresponding upper and lower bounds — here the bounds are sufficiently tight that they cannot be distinguished from the RB output.

[3]In fact, we choose visualization vertices $\mathcal{T}_\mathcal{V}^\mathcal{N}$ corresponding to a uniformly coarsened version of the outer surface of the truth mesh in order to reduce the amount of data loaded onto the phone.

an infinitely long duct with a periodic array of spherical "eddy promoters.") We shall directly consider the problem in nondimensional form; we choose a diffusive scaling in which all lengths are measured in units of the duct width (= height) $\tilde{H}$ and velocities are measured in units of $\tilde{\nu}/\tilde{H}$, where $\tilde{\nu}$ is the kinematic viscosity.

In weak from we wish to find the velocity $u(t;\Pi) \equiv (u_x, u_y, u_z) \in X$ for $0 \le t \le t_f = 0.25$ such that

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v + \frac{1}{2} \int_\Omega (v\nabla \cdot (u\, u) + vu \cdot \nabla u) = \int_\Omega \Pi v_x - \int_\Omega \nabla u \cdot \nabla v, \forall v \in X, \tag{11}$$

where X is the subspace of $H^1(\Omega)^3_{\text{per}}$ of functions which are divergence–free and which vanish (in all components) on all solid walls (note the $_{\text{per}}$ refers to 3–periodicity in x). Here Π is a nondimensional negative pressure gradient in the x–direction; the more usual Reynolds number based on average velocity will scale roughly as $\sqrt{\Pi}$. We choose $P = 1$ parameter, $\mu = \Pi$, and specify the parameter domain $\mathcal{D} \equiv [100, 700]$. We consider two outputs corresponding to averages of the streamwise (x) component of the velocity, u_x, over small cubes with side-length of 0.1 and with centers at ($x = 0.3, y = 1/2, z = 1/2$) and ($x = 0.7, y = 1/2, z = 1/2$) respectively: the outputs can be interpreted as local Reynolds numbers; the difference in these fore-aft outputs can be interpreted as measure of the strength of the nonlinear (inertial) effects. Finally, we select a Crank-Nicolson temporal discretization with $K = 100$ time levels and a truth finite element approximation space with $\mathcal{N} = 623{,}632$ degrees of freedom[4]. This example is intended to illustrate the extension of the RB formulation to (quadratic) nonlinearities.

This treatment of nonlinear parabolic PDEs builds directly on the LTI parabolic foundation. There are, however, significant complications in particular related to the error bounds. First, the stability constants will be significantly negative for higher Reynolds number — indicating exponential growth of the error bounds in time. (In some cases this exponential growth is "real" and inescapable; in some cases this exponential growth is an artifice of our energy arguments — and hence could be mitigated.) We may thus not consider either larger Reynolds numbers or larger final times t_f[5]. Second, these stability constants, albeit pessimistic, will be much more difficult to compute (via the SCM procedure) due to the non-LTI nature of the problem. And third, the dual norm of the residual is much more difficult to compute; as a result, the storage for the Online Dataset and the operation count for `Evaluation` will now scale as Q^2N^4 and not Q^2N^2. (The hp approach can effectively moderate this effect since the division into parameter subdomains reduces N.)

In this case the truth calculation is expensive: each evaluation $\mu \to u^{\mathcal{N}}(t^k;\mu), 0 \le k \le K$, requires 57 wall–clock minutes on Ranger with $n_{\text{proc}} = 256$. The Offline effort on Ranger requires just slightly over 12 wall-clock hours to complete the POD–Greedy for $N_{\max} = 12$. In this case, most of the Offline effort is associated with the truth snapshots; the "arg max" is relatively inexpensive here since $N_{\max}$ is small and also Ξ^{train} is quite small.

We now proceed to the Online stage: we present in Figures 4a and 4b the two outputs as a function of time for $\mu' = 100$ and $\mu' = 700$, respectively, obtained with $N = 12$ basis functions. These plots are screen grabs from our Nexus One implementation — in each case the output data is generated in roughly 2 seconds. We observe that for $\mu' = 700$ the Reynolds based on maximum velocity (at the final time) is roughly 30, indicating significant nonlinear effects; the considerable fore–aft asymmetry at $\mu' = 700$ is further

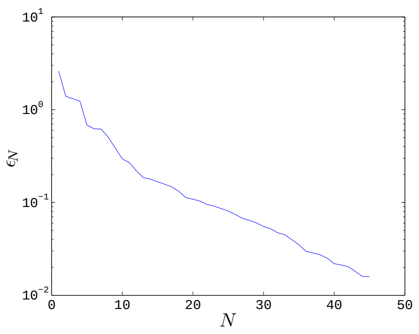


Figure 2: Convergence of the POD-Greedy algorithm for the "Swiss Cheese" problem.

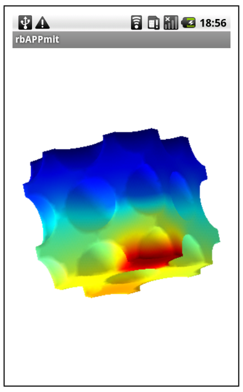


Figure 3: Visualization of the "Swiss Cheese" solution field at the final time for $\mu' = (2, 2, 0.5, 2, 1.31, 1.745, 1.85)$.

[4]Note the latter is in fact based on a Taylor-Hood discretization in which the pressure as Lagrange multiplier is introduced in the usual fashion to impose the divergence-free condition. We choose to write the weak form (11) in div-free form since this is our point of departure for the RB approximation — in which all snapshots are incompressible.

[5]Note for our example here in fact the stability constants are positive: all disturbances monotonically decay. Examples at higher Reynolds number for which the stability constants are negative are presented in [17].

indication of significant deviation from Stokes flow. Again, for each of the two outputs we present both $s_N(t^k;\mu'), 0 \le k \le K$, and also lower and upper bounds $s_N(t^k;\mu') \pm \Delta_N^s(t^k;\mu'), 0 \le k \le K$; the truth $s^{\mathcal{N}}(t^k;\mu'), 0 \le k \le K$, must reside within the bounds provided. We also show plots of the x-component of the velocity field from the Nexus One implementation in Figure 5.[6] In short, we have addressed Hilbert's 25th Problem: "Solve the Navier-Stokes equations on a phone."

Finally, we note that although the emphasis in this paper is on real-time deployed response, in fact our approach is also appropriate for many–query studies (and hence particularly appropriate for real-time many-query applications such as parameter estimation). In our Navier-Stokes example we observe that the break-even point is 12 queries: after 12 queries the reduced basis approach is much less expensive than the classical finite element approach *even if we include in the reduced basis budget the considerable Offline costs*.

[6]We again choose visualization vertices $\mathcal{T}_V^{\mathcal{N}}$ corresponding to a $\mathbb{P}_1$ uniformly coarsened version of the mesh.

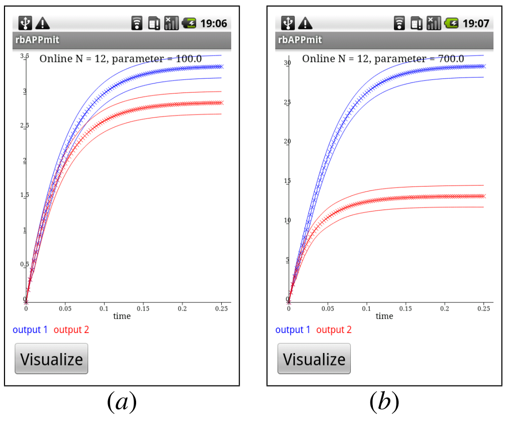

rbAPPmit
Online N = 12, parameter = 100.0
Online N = 12, parameter = 700.0
time
output 1
output 2
Visualize
(a)
(b)

Figure 4: Two RB output plot screenshots from the Nexus One implementation for the Navier-Stokes problem: (a) $\mu' = 100$, (b) $\mu' = 700$. The lines marked with $\times$ symbols are the RB outputs as functions of time, and the corresponding RB output bounds are plotted as solid lines. Note the change in vertical scale between (a) and (b).

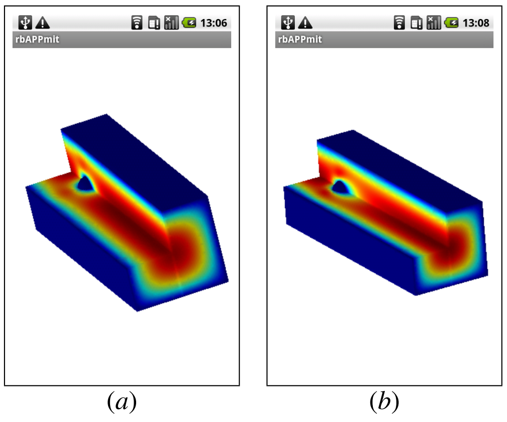


Figure 5: Two plots of the x-component of the Navier-Stokes velocity field from the Nexus One implementation: (a) $\mu' = 100$, (b) $\mu' = 700$. Here we have taken an "L-shaped" manifold as the $\mathcal{R}$ visualization surface in order to show the velocity field in the domain interior.

5. Acknowledgements

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] Porsching, T., Lee, M.. The reduced-basis method for initial value problems. SIAM Journal of Numerical Analysis 1987;24:1277–1287.
[2] Fink, J., Rheinboldt, W.. On the error behavior of the reduced basis technique for nonlinear finite element approximations. Z Angew Math Mech 1983;63(1):21–28.
[3] Almroth, B., Stern, P., Brogan, F.. Automatic choice of global shape functions in structural analysis. AIAA Journal 1978;16:525–528.
[4] Noor, A.. Recent advances in reduction methods for nonlinear problems. Comput Struct 1981;13:31–44.
[5] Rozza, G., Huynh, D., Patera, A.. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations application to transport and continuum mechanics. Archives of Computational Methods in Engineering 2008;15:229–275.
[6] Barrault, M., Nguyen, N.C., Maday, Y., Patera, A.T.. An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations. C R Acad Sci Paris, Série I 2004;339:667–672.
[7] Eftang, J.L., Grepl, M.A., Patera, A.T.. a posteriori error bounds for the empirical interpolation method. CR Acad Sci-Paris, Series I 2010;Submitted.
[8] Huynh, D.B.P., Rozza, G., Sen, S., Patera, A.T.. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. CR Acad Sci Paris Series I 2007;345:473–478.
[9] Haasdonk, B., Ohlberger, M.. Reduced basis method for finite volume approximations of parametrized linear evolution equations. M2AN Math Model Numer Anal 2008;42(2):277–302.
[10] Knezevic, D.J., Peterson, J.W.. A high-performance parallel finite-element framework for the certified reduced basis method. 2010. In preparation.
[11] Eftang, J., Patera, A., Rønquist, R.. An hp certified reduced basis method for parametrized elliptic partial differential equations. SIAM Journal on Scientific Computing 2010;Accepted.
[12] Udawalpola, R., Berggren, M.. Optimization of an acoustic horn with respect to efficiency and directivity. Int J Numer Meth Eng 2008;73:1571–1606.
[13] Medvinsky, M., Turkel, E., Hetmaniuk, U.. Local absorbing boundary conditions for elliptical shaped boundaries. J Comput Phys 2008;:8254–8267.
[14] Blackstock, D.T.. Fundamentals of Physical Acoustics. J. Wiley and Sons, Inc.; 2000.
[15] Huynh, D., Knezevic, D., Chen, Y., Hesthaven, J., Patera, A.. A natural-norm successive constraint method for inf-sup lower bounds. Comput Method Appl M 2010;Http://dx.doi.org/10.1016/j.cma.2010.02.011.

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

[17] Knezevic, D.J., Nguyen, N.C., Patera, A.T.. Reduced basis approximation and a posteriori error estimation for the parametrized unsteady boussinesq equations. Submitted. M3AS.