

Reduced-Basis Method for RCS Computations

BY AKIL NARAYAN

August 23, 2007

Table of contents

1	Presentation of the Problem	3
2	The Reduced-Basis Approximation	6
2.1	Motivation	6
2.2	General Framework and Methodology	7
2.2.1	Assumptions on the Spaces X^N	9
2.2.2	The Affine, ‘Compliant’ Hypotheses	10
2.3	The Reduced-Basis Approach	11
2.3.1	<i>A Posteriori</i> Error Bounds	11
2.3.2	Computing the μ_i and N	15
2.3.3	Online-Offline Details	16
2.4	A Special Case: Computing Outputs	18
3	Extensions to Noncoercive Problems	20
3.1	Mathematical Preliminaries	20
3.2	Reduced-Basis Approximation	22
3.2.1	The inf-sup Constant	22
3.2.2	Computation of the Error Bound $\tilde{\beta}(\mu)$	25
3.2.3	Presenting an Algorithm	27
3.2.4	Numerical Example: 1D Helmholtz	28
3.2.5	<i>A Posteriori</i> Error Bound	29
4	Evaluating the Output	30
4.1	The Adjoint Problem	30
4.2	Computing the Output	30
4.3	Application to the RBM	31
4.4	Algorithm Design	33
5	Application to the RCS Problem	34
5.1	The Framework	34
5.2	Formulation: The Helmholtz Equation	35
5.2.1	Implementation: The Operators	37
5.2.2	Results: 1D Problem	37
5.3	Fusion with time-domain solvers	41
5.3.1	Time-to-Frequency Transformation Considerations	41
5.3.2	The selection of \mathcal{R}	41
5.3.3	Bilinear form required	42
5.3.4	Adjoint problem	43
5.3.5	An Alternative For Computing Outputs	45
5.3.6	A Proposed Algorithm	45
Appendix A	The Kolmogorov N-width	47
Appendix B	Finite-Dimensional Representations	47
Appendix C	The Local Discontinuous Galerkin Method	49

C.1 Problem Statement	49
C.2 Discontinuous Finite Element Geometry	49
C.3 DG Strategy	50
C.4 The LDG Formulation	51
Appendix D Elliptic Outflow Boundary Conditions	54
Appendix E The RBM Algorithm	56
Bibliography	59

1 Presentation of the Problem

We begin by presenting some notations and definitions that will be used in the remainder of this write-up. We assume our geometry is some subset $\Omega \subset \mathbb{R}^d$ where $d = 1, 2,$ or 3 . Living on the space $\mathbb{R}_+ \times \Omega$ are the electromagnetic \mathbb{R}^3 -valued field quantities $\mathbf{E}(t, \mathbf{x})$ and $\mathbf{H}(t, \mathbf{x})$ (the electric and magnetic fields). For this work, we shall assume that \mathbf{E} and \mathbf{H} satisfy the free-space nondimensionalized Maxwell's Equations in $\mathbb{R}_+ \times \Omega$ of the form

$$\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} \quad (1)$$

$$\frac{\partial \mathbf{H}}{\partial t} = -\nabla \times \mathbf{E} \quad (2)$$

$$\nabla \cdot \mathbf{E} = 0 \quad (3)$$

$$\nabla \cdot \mathbf{H} = 0 \quad (4)$$

We shall also make use of the temporally-Fourier-transformed version of equations (1) and (2), which we present below as:

$$\nabla \times \tilde{\mathbf{H}} = i\omega \tilde{\mathbf{E}} \quad (5)$$

$$\nabla \times \tilde{\mathbf{E}} = -i\omega \tilde{\mathbf{H}}, \quad (6)$$

where $\tilde{\mathbf{E}}(\omega, \mathbf{x})$ denotes the Fourier transform of $\mathbf{E}(t, \mathbf{x})$. We may take the curl of equation (6) and substitute using (5) to eliminate $\tilde{\mathbf{H}}$. The result is a second order wave equation for $\tilde{\mathbf{E}}$:

$$\nabla \times \nabla \times \tilde{\mathbf{E}} = \omega^2 \tilde{\mathbf{E}} \quad (7)$$

Note that if we assume that equation (3) is satisfied, then we may expand the curl-curl operator and write the Helmholtz equation:

$$\Delta \tilde{\mathbf{E}} + \omega^2 \tilde{\mathbf{E}} = \mathbf{0}, \quad (8)$$

Note that we have opted to derive a Helmholtz equation for the electric field, but that due to the symmetry of (1) - (4), an identical equation can be derived for the magnetic field.

For our interest in computing the radar cross section (RCS) from a target body, we assume that the body $\Gamma \subset \Omega \subset \mathbb{R}^d$ lies entirely enclosed by Ω . Typically, we shall impose some source boundary conditions on $\partial\Gamma$ (though other boundary conditions are certainly possible) and out-flow boundary conditions on $\partial\Omega$. ([1],[2])

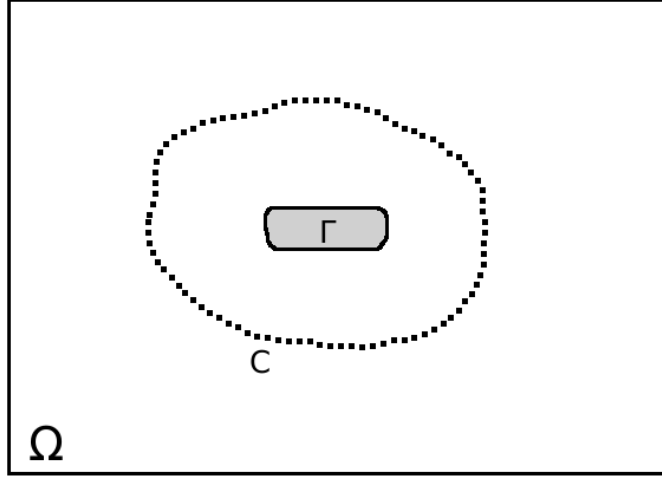


Figure 1. An example of a 2D geometry for the electromagnetics problem.

Definition 1. (*PDE problem definitions*)

We define the following initial/boundary value problems:

1. (Maxwell's Equations) \mathfrak{M} is the collection of equations (1) - (4) along with appropriate (e.g. PEC/source) boundary conditions on $\partial\Gamma$, outflow conditions on $\partial\Omega$, and initial data $[\mathbf{E}(0, \mathbf{x}), \mathbf{H}(0, \mathbf{x})] = [\mathbf{E}_0(\mathbf{x}), \mathbf{H}_0(\mathbf{x})]$.
2. (Curl-curl Equation) \mathfrak{C} is equation (7) with appropriate (e.g. PEC/source) boundary conditions on $\partial\Gamma$ and outflow conditions on $\partial\Omega$.
3. (Helmholtz Equation) \mathfrak{H} is equation (8) with appropriate (e.g. PEC/source) boundary conditions on $\partial\Gamma$ and outflow conditions on $\partial\Omega$.

For all of our problems, we consider the source terms on the boundary $\partial\Omega$ and the angular frequency ω as external parameters. Our ultimate goal is to compute an RCS from the object Γ . Thus, we consider our output as a function of the incident direction of the (monochromatic) field, (θ^{in}, ϕ^{in}) . In all that follows, we shall refer to the pair (θ, ϕ) as the direction φ . To be more precise, we shall define source terms, e.g.,

$$\mathbf{E}_0^{in}(t, \mathbf{x}) = \begin{cases} \tilde{\mathbf{E}}_0^{in} e^{i(\mathbf{k}(\varphi^{in}) \cdot \mathbf{x} - \omega t)} & x \in \partial\Gamma \\ \mathbf{0} & \text{else} \end{cases} \quad (9)$$

$$\mathbf{H}_0^{in}(t, \mathbf{x}) = \begin{cases} \tilde{\mathbf{H}}_0^{in} e^{i(\mathbf{k}(\varphi^{in}) \cdot \mathbf{x} - \omega t)} & x \in \partial\Gamma \\ \mathbf{0} & \text{else} \end{cases} \quad (10)$$

where $\tilde{\mathbf{H}}_0^{in} = \frac{\mu_0}{\eta_0} \hat{\mathbf{k}} \times \tilde{\mathbf{E}}_0^{in}$, η_0 is the free-space impedance ($\mu_0 = \eta_0 = 1$ given our nondimensionalized system (1)-(4)), and $\hat{\mathbf{k}} = \frac{\mathbf{k}}{|\mathbf{k}|}$ is a unit vector in the direction of wave propagation. The superscript *in* denotes 'incident' field (not to be confused with the imaginary unit *i*). We have

explicitly shown that the wavevector \mathbf{k} is a function of the incident direction φ^{in} . Equations (9) and (10) can be applied directly to problem \mathfrak{M} , but for \mathfrak{Z} we must instead use the appropriate phasor quantity for boundary sources.

In order to compute the RCS we shall, as is customary, employ the near-to-far field transformation[3]. As input for this method, we require the Fourier-transformed fields $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{H}}$ on a particular surface surrounding the scatterer Γ . We shall denote this surface C and a sample surface C is shown in figure 1. Note that it is convenient to choose C to conform to the particular discretized geometry.

It is apparent that $\tilde{\mathbf{E}}|_{\mathbf{x} \in C}$ and $\tilde{\mathbf{H}}|_{\mathbf{x} \in C}$ are easily extracted from the solution to \mathfrak{Z} . If we use \mathfrak{M} , then during the time-stepping, we must store $[\mathbf{E}(t, \mathbf{x}), \mathbf{H}(t, \mathbf{x})]|_{\mathbf{x} \in C}$ for the entire computation (or perhaps just for some time interval after transient disturbances have died out), and then employ the Fourier Transform afterwards.

Having computed $\tilde{\mathbf{E}}|_{\mathbf{x} \in C}$ and $\tilde{\mathbf{H}}|_{\mathbf{x} \in C}$, we shall not elaborate on the computation of the RCS, or the associated requisite near-to-far field transformation here, though there are many references that can adequately describe the procedure. The result ([1], [3], [4], [5]) is that the bistatic RCS σ can be computed via

$$\sigma = \frac{\omega^2}{4\pi} \frac{|\mathbf{F}(\varphi^{in}, \varphi^{obs})|^2}{|\tilde{\mathbf{E}}_0^{in}|^2}, \quad (11)$$

where we have introduced the direction of observation $\varphi^{obs} = (\theta^{obs}, \phi^{obs})$. And we have that \mathbf{F} is defined by the following:

$$\mathbf{F}(\varphi^{in}, \varphi^{obs}) = \int_C \left[(\hat{\mathbf{n}} \times \tilde{\mathbf{H}}) - (\hat{\mathbf{n}} \times \tilde{\mathbf{E}}) \times \hat{\mathbf{k}}(\varphi^{obs}) - (\hat{\mathbf{n}} \cdot \tilde{\mathbf{E}}) \hat{\mathbf{k}}(\varphi^{obs}) \right] e^{-i|\mathbf{k}(\varphi^{in})| \hat{\mathbf{k}}(\varphi^{obs}) \cdot \mathbf{x}} d\mathbf{x} \quad (12)$$

In the expression above, $\hat{\mathbf{n}}$ is the outward normal vector to the surface C .

Definition 2. (*RCS Computation*)

We define \mathfrak{R} as the problem of obtaining σ using equations (11) and (12) from the field outputs $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{H}}$ given as solutions to either \mathfrak{M} , \mathfrak{C} , or \mathfrak{Z} .

The ultimate goal is to compute the RCS σ (by any convenient, accurate means) for multiple frequencies ω and incident wave angles φ^{in} . In order to accurately capture the RCS variation, we can require as many as millions of evaluations of (11). The brute-force method to accomplish this is to solve the appropriate PDE system (\mathfrak{M} , \mathfrak{C} , or \mathfrak{Z}) millions of times for each different required value of (φ^{obs}, ω) . In a full 3D system, any PDE system we choose to solve may have upwards of millions of degrees of freedom in itself, which we must solve millions of times. This is computationally infeasible.

A possible solution that has arisen is the application of the Reduced-Basis method (RBM) ([6], [cite more](#)) to this problem. The basic idea of the RBM is to choose our degrees of freedom to adapt to the characteristic shape of the solution. If we can accomplish this, then the number of degrees of freedom required to solve the PDE systems will be dramatically decreased (down to as small as number as 10). Given this reduced system, we can easily solve the PDE millions of times in a small amount of time.

The RBM, which we shall now begin to detail, is a means of dramatically decreasing the computational cost necessary for solving a problem, while maintaining numerical accuracy.

2 The Reduced-Basis Approximation

2.1 Motivation

We shall now discuss things which at first glance will not connect to what was discussed in section 1. Suppose we wish to solve some system $\mathcal{F}(u) = 0$ for some unknown $u \in X$, where X is a complete inner product space (i.e. a Hilbert space). For example, if $X = \mathbb{R}$ we may take \mathcal{F} to be a problem of the form $\mathcal{F}(x) = x^2 - \mu = 0$ for $\mu \in [0, 9]$ unspecified. Also suppose that \mathcal{F} depends on some general collection of parameters $\mu \in \mathcal{D}$. (In the previous example, $\mathcal{D} = [0, 9] \subset \mathbb{R}$.) Typically we shall take $\mathcal{D} \subset \mathbb{R}^p$ for some integer p ; there is no theoretical reason for this restriction, but practical considerations make this restriction desirable, and often we shall ask that $p = 1$ or 2 . Therefore, the problem we are faced with is solving

$$\mathcal{F}(u; \mu) = 0 \tag{13}$$

for $\mu \in \mathcal{D}^p$. From this notation, it is clear that we can view u as a solution dependent on μ , that is, $u(\mu)$. Suppose further that we wish to solve (13) for a great many number of $\mu_i \in \mathcal{D}$, $i = 1, 2, 3, \dots, P$. We could simply solve \mathcal{F} P times and we have our solutions. However, solving \mathcal{F} may not be trivial, and may require a large amount of computing power. Assuming that we know very little about the qualitative shape and regularity of $u(\mu)$, a large number of unknowns will be needed in order to properly resolve $u(\mu_i)$ for each i . In particular, if using a Galerkin method (which is almost always the case), one will need a large number of ‘generic’ basis functions. (By ‘generic’, we mean e.g. the first \mathcal{N}_t grid points, cell averages, monomials, Legendre polynomials, Fourier modes, etc. for some very large number \mathcal{N}_t ; the reason for the notation \mathcal{N}_t will be explained later.)

However, the reduced basis method makes one very critical realization: assuming some regularity of u with respect to μ (which is almost always the case), then if we’ve solved $u(\mu_i)$ for $i = 1, 2, \dots, N \ll P$, then it is *not* the case that we know very little about $u(\mu_i)$ for $i > N$. In fact, we know a lot about the shape characteristics of $u(\mu_i)$ given the first N samples. The basic idea is that if the N -dimensional subspace of $u(\mathcal{D})$ spanned by $\{u(\mu_i)\}_{i=1}^N$ is rich enough, then we can approximate $u(\mu)$ for any μ by using $\{u(\mu_i)\}_{i=1}^N$ as our basis instead of the \mathcal{N}_t generic basis functions (see Appendix A for a more rigorous statement). Since the $u(\mu_i)$ are more representative of the solution than some generic basis functions, they will most likely do a much better job in approximating $u(\mu)$. In this way, we can choose $N \ll \mathcal{N}_t$ while still keeping the same error bounds. Since $N \ll \mathcal{N}_t$, then computing a solution to (13) will be much less expensive.

In the following sections, we hope to make the reduced-basis method more transparent and to nail down some of the details. However, in general there is relatively little theory present. This is due to the relative infancy of the reduced-basis method and the complexity of the RCS problem to be considered. For example, what is the convergence rate of the RB solution? There does exist theory stating that one can get exponential convergence in N of the reduced-basis method, but this relies either on the problem \mathcal{F} having a very simple form, or on an impractical computation to determine the optimal $\{u(\mu_i)\}_{i=1}^N$.

It should also be noted that the reduced basis method in no way means to supplant any traditional discrete solver. Upon closer examination, one can see that the reduced basis method has no means of computing the $\{u(\mu_i)\}_{i=1}^N$ without a traditional e.g. FEM solver. The reduced basis method is merely a method for more efficiently computing solutions to repetitively-solved problems using an existing discrete solver.

One should now realize that if we use some traditional solver to compute $\{u(\mu_i)\}_{i=1}^N$, then we cannot expect the RB solver to do better than the traditional solver. Indeed, as the theory is presented in the following sections, the reader will see that we only have the means of comparing the RB solution to the traditional-solver solution, not to the exact solution.

2.2 General Framework and Methodology

We shall now introduce the language that is accepted in most of the reduced-basis (RB) community. We shall assume that we are attempting to compute an output

$$s^e(u^e, \mu) = l^e(u^e; \mu), \quad (14)$$

where the ‘e’ superscript stands for exact. We insist that the output l^e be linear in the function u^e . The function $u^e \in X^e$ (we adopt the notation of section 2.1: X^e is a Hilbert space) which satisfies some Galerkin-formulated bilinear equation

$$a^e(u^e(\mu), v; \mu) = f^e(v; \mu) \quad \forall v \in X^e \quad (15)$$

This is the (infinite-dimensional) mathematical problem to be solved. Under certain regularity assumptions on a^e and f^e , one can show existence, uniqueness, and well-behavedness (with respect to f^e) of u^e . We shall assume in the rest of this chapter that a^e is symmetric and coercive in the natural norm of X . The reason for assuming symmetry is because such an assumption simplifies some book-keeping, though the symmetry assumption is not essential; in later chapters we shall lift this restriction. The assumption that a^e is coercive in the natural norm of X is again a bit more restrictive than necessary. Strictly speaking, we need only ask for positivity of the affine components of a^e (see section 2.2.2). Again, in later chapters we shall not require a^e to be coercive. Recall that we are actually interested in s^e , while u^e is simply a transitory variable.

We have outlined the continuous, infinite-dimensional problem, but the main thrust of this method is with the discrete form associated with (15). To this end, shall consider some \mathcal{N}_t -dimensional subspace of X^e , and call it $X^{\mathcal{N}_t}$. (The ‘t’ subscript refers to ‘truth’.) All the quantities we have described above indeed have discrete counterparts. It should be noted that the discrete counterparts are *directly dependent on the numerical solver of choice*. With this in mind, in order to refer to the discrete operators associated with the \mathcal{N}_t -dimensional problem, we simply remove the ‘e’ superscript. This is a common convention in this write-up: if the reader sees a quantity and expects that it should have a superscript, but it has none, a superscript of \mathcal{N}_t should be assumed. Thus, we can state our discrete problem as

$$a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in X \quad (16)$$

We refer to the u which satisfies (16) as the ‘truth solution’ (as opposed to the exact solution, u^e) and all quantities/operators/algorithms associated with (16) are referred to as ‘truth’ quantities.

The goal, of course, is to produce some N -dimensional space X^N which is a subset of X^e with $N \ll \mathcal{N}_t$ with which to define a Galerkin problem similar to (15) so that numerically solving it is easier than the \mathcal{N}_t -dimensional problem. All quantities with superscripts ‘N’ hereafter refer to these N -dimensional quantities/operators/algorithms, and are labelled as ‘RB’ quantities.

Definition 3. (*Topology on X*)

Given the inner product space X^m , for $m = e, \mathcal{N}$, or N , and some predetermined value of the parameter $\bar{\mu}$, we define an inner product (and thus an associated norm) on X^m as $(u, v)_{a^m} \doteq a^m(u, v; \bar{\mu})$ for all $u, v \in X^m$. X^m is still endowed with its ‘natural’ inner product (\cdot, \cdot) , but $(\cdot, \cdot)_{a^m}$ induces an equivalent norm.

Definition 3 may not make sense in the general sense: if a is not bounded and coercive, then $a(u, v, \bar{\mu})$ may not qualify to be an inner product. However, the definition of this inner product and associated norm is used in this chapter only, and in this chapter, we do assume that a^m is indeed bounded and coercive. We also note that, assuming affine parameter dependence (see section 2.2.2 for notation), we require $a_q^m(u, u) \geq 0$ for all $u \in X^m$ and $\Theta_q^m(\mu) > 0$ for all $\mu \in \mathcal{D}$ in order for definition 3 to make sense, which is a condition independent of any definition of a norm on X^m . We have not yet said anything about the value of $\bar{\mu}$. We shall comment on this later.

As mentioned before, we assume that a^e and all its incarnations are symmetric bilinear forms. In addition we also expect certain regularity conditions on a^e and its incarnations in order for problem (15) to be well-posed. We list definitions for these below.

Definition 4. (Continuity, coercivity, inf -sup parameters)

We shall define the following parameters:

- The continuity constant of a^e , $\gamma^e(\mu) \doteq \sup_{u \in X^e} \sup_{v \in X^e} \frac{|a^e(u, v; \mu)|}{\|u\|_{X^e} \|v\|_{X^e}}$. Furthermore, we assume $\gamma^e(\mu) \leq \gamma_0^e$ uniformly for all $\mu \in \mathcal{D}$.
- The coercivity constant of a^e , $\alpha^e(\mu) \doteq \inf_{u \in X^e} \frac{|a^e(u, u; \mu)|}{\|u\|_{X^e}^2}$. We also assume $\alpha^e(\mu) \geq \alpha_0^e$ uniformly for all $\mu \in \mathcal{D}$.
- The inf -sup parameter of a^e , $\beta^e(\mu) \doteq \inf_{u \in X^e} \sup_{v \in X^e} \frac{|a^e(u, v; \mu)|}{\|u\|_{X^e} \|v\|_{X^e}}$. We also assume $\beta^e(\mu) \geq \beta_0^e$ uniformly for all $\mu \in \mathcal{D}$.

We briefly pause to note that the elliptic coercivity constant is a special case of the inf -sup parameter and, in particular, the coercivity constant is always less than or equal to the inf -sup parameter. Furthermore, we shall extend these definitions to apply to our truth space $X^{\mathcal{N}_t}$, and to the RB space X^N in the natural way. Therefore, for e.g. the continuity constant γ^e , there are two corresponding constants $\gamma(\mu)$ ($= \gamma^{\mathcal{N}_t}(\mu)$) and $\gamma^N(\mu)$. Because of these definitions, we have in general:

$$\gamma^e(\mu) \geq \gamma(\mu) \geq \gamma^N(\mu) \quad \forall \mu \in \mathcal{D}$$

$$\alpha^e(\mu) \leq \alpha(\mu) \leq \alpha^N(\mu) \quad \forall \mu \in \mathcal{D}$$

$$\beta^e(\mu) \leq \beta(\mu) \leq \beta^N(\mu) \quad \forall \mu \in \mathcal{D}$$

We remind the reader that in this chapter a^m is always symmetric, bounded, and coercive. Thus the inner product $(\cdot, \cdot)_a$ is well-defined, and the norms $\|\cdot\|_a$ and $\|\cdot\|_X$ are equivalent.

Again, the reduced basis idea is that the manifold $\mathcal{M} \doteq \text{span}\{u(\mu_i): i = 1, 2, \dots, \mathcal{N}_t; \mu \in \mathcal{D}\}$ can be very well approximated by a (much) lower-dimensional space $\mathcal{M}^N \doteq \text{span}\{u(\mu_i): i = 1, 2, \dots, N; \mu_i \in \mathcal{D}\}$ for $N \ll \mathcal{N}_t$. Thus, we shall expect to define appropriate spaces X^N (it is for this reason that we shall henceforth drop superscripts on $X^{\mathcal{N}_t}$ and simply call it X , to avoid confusion with X^N).

Finally, we endow the (finite-dimensional) space X ($= X^{\mathcal{N}_t}$) with \mathcal{N}_t basis functions $\{\psi_i\}_{i=1}^{\mathcal{N}_t}$. These are our ‘generic’ basis functions which are e.g. the first M monomials on K nonoverlapping elements (in which case $\mathcal{N}_t = M K$). Of course, they need not be orthonormal, and the underlying assumption is that these are basis functions hard-coded into the pre-existing numerical solver. With these basis functions, it is not hard to show that if one defines the global mass matrix

$$\mathbb{A}_{mn} = \langle \psi_m, \psi_n \rangle_a \tag{17}$$

then for functions f and g in X with modal coefficients (in the ψ_i) given by the vectors \vec{f} and \vec{g} , the inner product on the space X is given by

$$\langle f, g \rangle_a = \vec{f}^T \mathbf{A} \vec{g} \quad (18)$$

In addition, we shall define an energy norm $\|\cdot\|_\mu$ on X which is dependent on the parameter μ as follows:

$$\|f\|_\mu = a(f, f; \mu) \quad (19)$$

This is an energy norm with value dependent on μ , and note that, by definition, $\|\cdot\|_{\bar{\mu}} = \|\cdot\|_a$.

2.2.1 Assumptions on the Spaces X^N

We list here some basic assumptions that we make on the functions spaces X^N . First of all, we assume a heirarchical nature of these spaces, so that $X^{N-1} \subset X^N \subset X^{N+1} \dots \subset X \subset \dots \subset X^e$. The condition $X^N \subset X \subset X^e$ must certainly be satisfied, but we acknowledge that there have been some developments on the spaces such that $X^{N-1} \subset X^N$ is *not* satisfied [6]. While this is certainly an option, we choose instead to opt for the easier-implemented heirarchical choice. The main reason for this is algorithmic simplicity and clarity.

We shall generate the space X^N by making it the span of N ‘snapshots’ $u(\mu_i)$, $i = 1, 2, \dots, N$. We define $\xi_i \doteq u(\mu_i)$. Of course one should assume that the $u(\mu_i)$ are linearly independent with respect to the inner product on X , and this is almost always the case, except in particularly simple and uninteresting problems. This RB space of snapshots is a Lagrange space; we may also take X^N as Taylor or Hermite spaces [7], but these require derivatives of u with respect to μ (the ‘sensitivity derivatives’).

By construction, it is clear that $\dim(X^N) = N$. Of course, we must have some way of numerically implementing a statement like ‘ $a^N(u^N, w; \mu) = f^N(w; \mu) \forall w \in X^N$ ’, and solving for $u^N \in X^N$. The most simple method to do this is to find a basis for X^N , $\{\xi_i\}_{i=1}^N$, and use these as the test functions. However, in order to ensure a well-conditioned system, it is more prudent to orthogonalize the ξ_i to form an orthonormal basis $\{\zeta_i\}_{i=1}^N$ and use the ζ_i as the test functions instead. This may be accomplished using a modified Gram-Schmidt orthogonalization process or whatever orthogonalization algorithm the programmer wishes.

We henceforth assume a heirarchical nesting of the spaces X^N , and that they have associated orthonormal bases $\{\zeta_i\}_{i=1}^N$. In particular, this nesting allows for a large memory savings: if one wishes to implement p -adaptivity in the online portion of the computation, then one need only store $\{\zeta_i\}_{i=1}^{N_{\max}}$ and extract the appropriate basis vectors as necessary.

Finally, we present a proposition relating the truth and RB solutions, which is a standard finite-element Galerkin result.

Proposition 5. (Error between u and u^N , [6])

For any u^N satisfying

$$a^N(u^N, w; \mu) = f^N(w; \mu), \quad \forall w \in X^N \quad (20)$$

then the the error between u and u^N satisfies

$$\|u(\mu) - u^N(\mu)\|_a \leq \sqrt{\frac{\gamma^e(\mu)}{\alpha^e(\mu)}} \inf_{w^N \in X^N} \|u(\mu) - w^N\|_a \quad (21)$$

Proposition 5 essentially gives us a bound on the error of the RB solution with respect to the truth approximation.

We have not yet discussed how we shall form the X^N (i.e. how we choose the μ_i to form $u(\mu_i) = \xi_i$). Indeed this is the core component of the RB method, but we defer this discussion to a later section.

2.2.2 The Affine, ‘Compliant’ Hypotheses

In this section we elaborate on one of the cornerstone assumptions of the current RB method. Although we have not yet described all of the necessary computations for the RB method, the reader may surmise that determining the appropriate ξ_i of section 2.2.1 can be a rather costly procedure, including expensive solves of (16). These computations are referred to as *offline* computations. In order to make up for this, we expect that we can solve a much smaller system, equation (20), with orders-of-magnitude less computational cost than a single solve of (16).

However, in order to achieve such computational savings, we must cut as many corners as possible: suppose we have successfully formed our space X^N with our basis functions $\{\zeta_i\}_{i=1}^N$. We now move on to choosing some value of the parameter μ which interests us and solve (16). However, in order to form a matrix-vector relation to discretize (16), we must evaluate terms of the form $a^N(\zeta_i, \zeta_j; \mu)$ in order to form the matrix \mathbb{A}^N commensurate with (17). Because we have no readily-available, efficient means of forming N -point quadrature formulas to compute this, we must instead compute each of these matrix entries using our \mathcal{N}_t -dimensional truth formulation, which will take $O(N^2\mathcal{N}_t)$ operations. We must do this *every time we choose a different μ* . The ultimate goal is to solve (20) with an order of operations independent of \mathcal{N}_t . In this way, we truly reduce our problem (16) to an N -dimensional problem. One way to ensure an \mathcal{N}_t -independent computation of (20) is by the following assumption.

We assume that the operators a^e and f^e are affine in the parameter μ . That is, we assume that there exist a collection of functions $\Theta_q^a: \mathcal{D} \rightarrow \mathbb{R}$ and $\Theta_q^f: \mathcal{D} \rightarrow \mathbb{R}$ such that

$$a^e(u, w; \mu) = \sum_{q=1}^{Q_a} \Theta_q^a(\mu) a_q^e(u, w) \quad (22)$$

$$f^e(w; \mu) = \sum_{q=1}^{Q_f} \Theta_q^f(\mu) f_q^e(w) \quad (23)$$

$$s^e(w; \mu) = \sum_{q=1}^{Q_s} \Theta_q^s(\mu) s_q^e(v)$$

We assume that the above forms extend to the truth and RB functionals as well. I.e. that the Θ_q are the same regardless of the superscript on the functional. If the functionals take this form, then constructing the appropriate operators to solve (20) becomes much more efficient: one can simply compute and store $a_q^N(\zeta_i, \zeta_j)$, $f_q^N(\zeta_i)$, and $s_q^N(\zeta_i)$ as an offline computation, and then the formation of the operators to solve (20) becomes an $O(N^2)$ process.

We now briefly return to discuss definition 3 and some properties of the a_q . One could consider the case when e.g. a^e is coercive but the individual a_q^e are not all coercive, or when there are some $\Theta_q^a < 0$. This allows the possibility that some of the individual components of our operator noncoercive. However, if this is the case, then our analysis become much more complicated. To simplify our lives, we shall assume for the remainder of this report that $\Theta_q^a > 0$ for every q and that each of the a_q^e is coercive. Note that coercivity of the a_q^e either with respect to the native X^e norm or to the one defined in definition 3 is the same: they are equivalent norms as long as a^e is uniformly on \mathcal{D} coercive and continuous with respect to the native norm of X^e .

We shall now make some remarks on the ‘compliance’ of the problem (15). If $f(v; \mu) = s(v; \mu)$ (recall that s is our output of interest), and the bilinear form a is symmetric, then the system (14) and (15) is said to be ‘compliant’. This term stems from problems of solid mechanics, one of the first areas of application of the RB method. Noncompliant problems suffer from more complex algorithm formulations in order to retain convergence properties. However, noncompliant problems are still very amenable to the RB method; one simply must work a little harder.

2.3 The Reduced-Basis Approach

We now have a good deal of the language down and are ready to speak in a more detailed manner about exactly how the reduced-basis method will work. Below is a very high-level idea of exactly what we plan to do.

Suppose we are given a symmetric, coercive, bilinear form $a^e: X^e \times X^e \times \mathcal{D} \rightarrow \mathbb{R}$, and some bounded linear functional $f^e: X^e \times \mathcal{D} \rightarrow \mathbb{R}$. We wish to solve the Galerkin problem (20), which uses the discrete versions of these operators, for $\mu = \{\mu_i\}_{i=1}^P$. These solutions must be found with an absolute error of ε_{\min} , measured in the norm according to definition 3. We compute the solutions $\{u(\mu_i)\}_{i=1}^P$ as follows:

Algorithm (The Reduced-Basis Method)

1. In an ‘offline’, \mathcal{N}_t -dependent overhead stage, do the following:
 - a. Compute the N necessary to ensure error less than ε_{\min} . (section 2.3.2)
 - b. Compute ‘best fit’ $\{\mu_i\}_{i=P+1}^{P+N}$ and associated $\{\xi_i\}_{i=1}^N$. (sections 2.3.1, 2.3.2)
 - c. Construct the appropriate operators and basis functions. (section 2.3.3)
2. Compute P solutions to (20) using $\{\mu_i\}_{i=1}^P$ and the reduced-basis $\{\zeta_i\}_{i=1}^N$
3. Determine maximal error estimators for the solutions (section 2.3.1)

We thus have our solution

$$u^N(\mu) = \sum_{i=1}^N u_i^N \zeta_i \quad (24)$$

and the associated error bounds. The details of implementing the above algorithm are presented in the following sections.

2.3.1 A Posteriori Error Bounds

In this section we discuss the very important task of quantifying how good of a job we’re doing with our reduced-basis approximation. These error bounds serve two purposes: first they provide an inexpensive method to select the μ_i and N , as detailed in section 2.3.2. Secondly, they provide us with a reliable measure of how close the RB approximations u^N are to the truth approximation, u . Again, we do not develop any framework for comparing the RB solution to the exact solution.

We ask the reader to recall our definitions of the coercivity and continuity constants, as well as the affine parameter assumption notation, as we shall utilize them all now. For any $\mu^* \in \mathcal{D}$, we define a new parameter, $\bar{\Theta}_{\mu^*}^{a,\min} : \mathcal{D} \rightarrow \mathbb{R}^+$ as

$$\bar{\Theta}_{\mu^*}^{a,\min} \doteq \min_{q=1,\dots,Q_a} \frac{\Theta_q^a(\mu)}{\Theta_q^a(\mu^*)} > 0. \quad (25)$$

This parameter will be very useful in our derivation on error bounds for u^N . There will also be need of a similar parameter, $\bar{\Theta}_{\mu^*}^{a,\max}$, where the definition should be clear. In addition, we define the ratio of these quantities

$$\bar{\Theta}_{\mu^*}^a = \frac{\bar{\Theta}_{\mu^*}^{a,\max}}{\bar{\Theta}_{\mu^*}^{a,\min}} \quad (26)$$

We are now in a position to derive an expression for α_0 , the lower bound for the coercivity constant.

Lemma 6. (*Coercivity lower bound, [6]*)

For a parametrically coercive affine operator a , we may define the coercivity lower bound α_0 as $\alpha_0 = \bar{\Theta}_{\bar{\mu}}^{a,\min}$.

Proof. We recall that the topology underlying the space X depends on the choice of a parameter $\bar{\mu}$ for the inner product and norm (see definition 3). The result of this lemma follows straight from various definitions:

$$\begin{aligned} a(u, u; \mu) &= \sum_{q=1}^{Q_a} \Theta_q^a(\mu) a^q(u, u) \\ &= \sum_{q=1}^{Q_a} \left(\frac{\Theta_q^a(\mu)}{\Theta_q^a(\bar{\mu})} \right) \Theta_q^a(\bar{\mu}) a^q(u, u) \\ &\geq \sum_{q=1}^{Q_a} \left(\min_{q=1,\dots,Q_a} \frac{\Theta_q^a(\mu)}{\Theta_q^a(\bar{\mu})} \right) \Theta_q^a(\bar{\mu}) a^q(u, u) \\ &= \bar{\Theta}_{\bar{\mu}}^{a,\min} \sum_{q=1}^{Q_a} \Theta_q^a(\bar{\mu}) a^q(u, u) \\ &= \bar{\Theta}_{\bar{\mu}}^{a,\min} a(u, u; \bar{\mu}) \\ &= \bar{\Theta}_{\bar{\mu}}^{a,\min} \|u\|_a^2, \end{aligned}$$

which follows from our definition of the norm of X . Thus we have

$$\alpha(\mu) = \inf_{u \in X} \frac{|a(u, u; \mu)|}{\|u\|_a^2} \geq \bar{\Theta}_{\bar{\mu}}^{a,\min} > 0$$

□

In the proof above, we derived an expression for the coercivity lower bound in terms of the a -norm and not the natural norm of X . This is acceptable because since a is symmetric, coercive, and bounded, the a -norm is equivalent to the X -norm, and thus any error estimates we derive using this expression for the coercivity lower bound are all valid as long as we measure the error in the a -norm.

We can also derive an upper bound for the continuity constant $\gamma(\mu)$. We shall state this bound without proof because of its similarity to lemma 6.

Lemma 7. (*Continuity upper bound, [6]*)

For a parametrically coercive affine symmetric operator a , we may define the continuity upper bound as $\gamma_0 = \bar{\Theta}_{\bar{\mu}}^{a,\max}$.

The symmetry of the operator is needed in lemma 7 because the Cauchy-Schwartz inequality applied to a is needed in the proof. This upper bound on the continuity constant $\gamma(\mu)$ is not necessary from an algorithmic point of view. However, it is useful for theoretical considerations of the ‘efficiency’ of our error estimators (to be discussed in a later in this section).

With a lower bound for our coercivity constant derived, we have effectively solved half our problem of finding an inexpensive *a posteriori* error estimate for the RB method (although this may not be yet apparent to the reader). The other half comes from deriving an expression for the residual, which we now discuss.

We define the error between the RB and truth approximations as $e(\mu) \doteq u(\mu) - u^N(\mu)$. We also define the residual $r(\cdot, \mu) \in X'$ (the topological dual space to X) as the quantity satisfying

$$r(w; \mu) = f(w; \mu) - a(u^N, w; \mu) \quad \forall w \in X \quad (27)$$

Due to Galerkin orthogonality and the bilinearity of a , the error satisfies

$$a(e(\mu), w; \mu) = r(w, \mu) \quad \forall w \in X \quad (28)$$

We now invoke the Riesz Representation Theorem to produce an element of X , $\hat{e}(\mu)$, which is isometrically isomorphic to $r(w; \mu) \in X'$. I.e.

$$r(w, \mu) = \langle \hat{e}(\mu), w \rangle_a \quad \forall w \in X \quad (29)$$

Coupling this with (28) we obtain

$$a(e(\mu), w; \mu) = \langle \hat{e}(\mu), w \rangle_a \quad \forall w \in X \quad (30)$$

and with our choice for the topology of X , this implies that $e(\bar{\mu}) = \hat{e}(\bar{\mu})$. At the end of the day, what we need is a measure of the error, and either from (29) or the Riesz Theorem, we have

$$\|r(\cdot; \mu)\|_{X'} = \sup_{w \in X} \frac{|r(w; \mu)|}{\|w\|_a} = \|\hat{e}(\mu)\|_a \quad (31)$$

All this theory may be illuminating (or obfuscating, depending on one’s point of view....) but we still do not have a feasible means of molding the computation of $\|\hat{e}(\mu)\|_a$ into an online-offline method. We present a possible solution below [6], again dependent on the affine assumption on the bilinear form a .

Using (27) and (24), we expand our residual out into the following form

$$r(w; \mu) = \sum_{q=1}^{Q_f} \Theta_q^f(\mu) f_q(w) - \sum_{i=1}^N \sum_{q=1}^{Q_a} u_i^N(\mu) \Theta_q^a(\mu) a_q(\zeta_i, w) \quad \forall w \in X \quad (32)$$

We wish to find the norm of this element of X' , but doing so directly using the definition (the middle term in (31)) is prohibitively expensive. Thus, we shall mold this into a more computationally attractive form. It is clear from (32) that we can divide the computation of $r(v; \mu)$ into μ -dependent and μ -independent computations (read ‘online’ and ‘offline’, respectively). To make this more transparent, we introduce some notation:

$$\vec{\Theta}^f(\mu) = \left(\Theta_1^f(\mu), \Theta_2^f(\mu), \dots, \Theta_{Q_f}^f(\mu) \right)^T \quad (33)$$

$$\vec{\Theta}^a(\mu) = \begin{pmatrix} u_1^N(\mu) \Theta_1^a(\mu), & u_1^N(\mu) \Theta_2^a(\mu), & \dots & u_1^N(\mu) \Theta_{Q_a}^a(\mu), \\ u_2^N(\mu) \Theta_1^a(\mu), & \dots & & u_2^N(\mu) \Theta_{Q_a}^a(\mu), \\ \vdots & \ddots & & \vdots \\ u_N^N(\mu) \Theta_1^a(\mu), & \dots & & u_N^N(\mu) \Theta_{Q_a}^a(\mu) \end{pmatrix}^T \quad (34)$$

$$\vec{f}(w) = (f_1(w), f_2(w), \dots, f_{Q_f}(w))^T \quad (35)$$

$$\vec{a}(w) = \begin{pmatrix} -a_1(\zeta_1, w), & a_2(\zeta_1, w), & \dots & a_{Q_a}(\zeta_1, w), \\ a_1(\zeta_2, w), & \dots & & a_{Q_a}(\zeta_2, w), \\ \vdots & \ddots & & \vdots \\ a_1(\zeta_N, w), & \dots & & a_{Q_a}(\zeta_N, w) \end{pmatrix}^T \quad (36)$$

This enables us to write (32) as $r(w; \mu) = \vec{\Theta}^f(\mu)^T \vec{f}(w) + \vec{\Theta}^a(\mu)^T \vec{a}(w)$ and by concatenating the vectors as $\vec{\Theta}(\mu) = \left(\vec{\Theta}^f(\mu)^T, \vec{\Theta}^a(\mu)^T \right)^T$ and $\vec{h}(w) = \left(\vec{f}(w)^T, \vec{a}(w)^T \right)^T$ we can more compactly write

$$r(w; \mu) = \vec{\Theta}(\mu)^T \vec{h}(w) \quad \forall w \in X, \quad (37)$$

and we denote the length of the vectors $\vec{\Theta}(\mu)$ and $\vec{h}(w)$ as Q_N . By our Riesz Representation in (29), we can thus write

$$\langle \hat{e}(\mu), w \rangle_X = \vec{\Theta}(\mu)^T \vec{h}(w) \quad \forall w \in X \quad (38)$$

Finally, by lineary and Q_N applications of the Riesz Representation Theorem, there exist Q_N elements of X which we collect in the vector \vec{g} satisfying

$$\langle g_i, w \rangle_X = h_i(w) \quad \forall w \in X, 1 \leq i \leq Q_N, \quad (39)$$

such that

$$\hat{e}(\mu) = \vec{\Theta}(\mu)^T \vec{g} \quad (40)$$

Still, how does one calculate the elements of \vec{g} ? Because $v \in X$, we can expand $v = \sum_{i=1}^{N_i} v_i \psi_i$. (Notice we have referenced our truth approximation basis ψ_i .) Thus, if we define $\mathbb{H}_{ij} = h_j(\psi_i)$ then we can write

$$\vec{h}(w) = \mathbb{H} \vec{w}, \quad (41)$$

where \vec{w} is the vector of modal coefficients. One can now determine from (39) that $\langle g_n, \psi_j \rangle_a = h_n(\psi_j) = \mathbb{H}_{jn}$ and thus the vector of modal coefficients for g_n is given by $g_{nj} = (\mathbb{X}^{-1} \mathbb{H})_{jn}$. Now we have solved for \vec{g} , but it turns out we can go one step further to calculate the quantities we really care about. We shall use Einstein's summation convention (wherein repeated indices are summed over) and define

$$\begin{aligned} \mathbb{G}_{mn} &= \langle g_n, g_m \rangle_X = \langle g_n, g_{mj} \psi_j \rangle_X \\ &= g_{mj} \langle g_n, \psi_j \rangle_X \\ &= (\mathbb{X}^{-1} \mathbb{H})_{jm} \mathbb{H}_{jn} \\ &= (\mathbb{H}^T \mathbb{X}^{-1})_{mj} \mathbb{H}_{jn} \\ &= (\mathbb{H}^T \mathbb{X}^{-1} \mathbb{H})_{mn}, \end{aligned} \quad (42)$$

where we have used our definition of \mathbb{H} and our derivation for g_{mj} . Also, we've used the fact that \mathbb{A} is symmetric.

Now from (40) we can finally compute the error residual as

$$\begin{aligned} \|\hat{e}(\mu)\|_a &= (\Theta_m(\mu) \langle g_m, g_n \rangle_X \Theta_n(\mu))^{1/2} \\ &= \left(\vec{\Theta}(\mu)^T \mathbb{G} \vec{\Theta}(\mu) \right)^{1/2} \end{aligned} \quad (43)$$

We have not yet motivated why we wished to find a lower bound for the coercivity constant (lemma 6) or the expression (43) for the *a posteriori* residual error. That discussion now begins.

Given the quantities that we have previously computed, we define our error estimator as

$$\Delta_N(\mu) = \frac{\|\hat{e}(\mu)\|_X}{\sqrt{\alpha_0}} \quad (44)$$

I.e. it is the residual error normalized by the lower bound of the coercivity constant. These quantities are readily computed as detailed earlier in this section. We also define the *effectivity* of this estimator as

$$\eta_N(\mu) = \frac{\Delta_N(\mu)}{\|e(\mu)\|_\mu}, \quad (45)$$

which is the error estimator normalized by the norm of the actual error measured in it's own energy norm. In order to prescribe accuracy, we should have $\eta_N \geq 1$, and in order to be efficient, we should make η_N as small as possible. The following result tells us exactly how good our efficiency estimators are.

Proposition 8. (*Effectivity Bounds, [6]*)

For $N = 1, 2, \dots, \mathcal{N}$, and all $\mu \in \mathcal{D}$, the effectivity $\eta_N(\mu)$ satisfies

$$1 \leq \eta_N(\mu) \leq \sqrt{\bar{\Theta}_\mu^a(\mu)} \quad (46)$$

A very desirable property of the above proposition is that the effectivity is bounded independent of N .

In this section, we have described an effective online-offline decomposition to compute an error estimator. This estimator will be used in an algorithm to choose our RB basis functions ζ_i .

2.3.2 Computing the μ_i and N

In this section we shall discuss the important task of choosing our reduced basis ζ_i . There are many details and things to consider in this procedure, and we shall only outline the main method. We shall assume that we are given the first basis function $\zeta_1 = \frac{u(\mu_1)}{\|u(\mu_1)\|_X}$. I.e. we assume that some divine being has imparted unto us the value of the first parameter to form our basis μ_1 . In practice, one can use a random choice from \mathcal{D} , or pick some parameter which clearly is representative of the set \mathcal{D} given the physical nature of the problem.

Following an inductive approach, we now consider the problem of having a collection of basis functions $\{\zeta_i\}_{i=1}^M$ for some $M \geq 1$. There are now two things we should consider:

- Should we set $N = M$ and terminate our search, effectively completing our formation of X^N ?
- If not, what should the parameter μ_{M+1} be?

In order to answer these questions, we describe an algorithm which is termed ‘greedy’ in the literature. The idea is that $\{\zeta_i\}_{i=1}^M$ form a subspace of X , and we wish to design an X^N which best approximates $\mathcal{M} = \{u(\mu) : \mu \in \mathcal{D}\} \subset X$. Therefore, given the X^M we have already constructed, we shall search out the function that we approximate worst in \mathcal{M} and add that to our basis. The determination of ‘worst’ is measured by our error estimator (44).

Algorithm (Selection of the Basis Parameters)

Given $\{\zeta_i\}_{i=1}^M$ and an error tolerance ε_{\min} ,

- Compute $\varepsilon_M = \max_{\mu \in \mathcal{D}} \Delta_M(\mu)$ and $\mu^* = \operatorname{argmax}_{\mu \in \mathcal{D}} \Delta_M(\mu)$
- If $\varepsilon_M < \varepsilon_{\min}$, then set $N = M$ and terminate
- Else, set $\mu_{M+1} = \mu^*$ and $\zeta_{M+1} = \frac{u^*(\mu_{M+1})}{\|u^*(\mu_{M+1})\|_a}$ where u^* is the portion of u orthogonal to X^M
- $M \leftarrow M + 1$, and repeat

There are still some details to flesh out: the computation of ε_M requires one to solve $a^M(u, v) = f^M(u, v) \forall v \in X^M$ over \mathcal{D} . There are many strategies on how to sample \mathcal{D} to accomplish this: one can simply use a fine mesh that is searched over every iteration. (Here the reader should now notice the utility of the affine assumption: if computing $a^M(\zeta_i, \zeta_j)$ was \mathcal{N}_t -dependent, we need $K \mathcal{N}_t$ -computations, where K is the size of our fine mesh of \mathcal{D} , which is quite undesirable.) Or one could have a heirarchical sequence of coarse meshes \mathcal{D}^M such that $\mathcal{D}^M \subset \mathcal{D}^{M+1} \subset \dots \subset \mathcal{D}$ and simply search over the appropriate \mathcal{D}^M each iteration. The computation of $u(\mu_{M+1})$ ($= u(\mu^*)$) requires a *full* $O(\mathcal{N}^3)$ truth approximation solve of (16).

It should be realized that one may be tempted to take ε_{\min} very small. This has two negative impacts, in terms of the efficiency of the RB method: first, N grows inversly to ε_{\min} . It should be no surprise that if one wishes a very high accuracy, one must specify a solution with a large number of unknowns. Second, the process of orthogonalizing $u(\mu_{M+1})$ with respect to X^M becomes an increasingly ill-conditioned problem as ε_{\min} becomes smaller.

We note that our estimator (44) is an estimator for the solution u , and not for our output of interest s . One can define an estimator for s in a manner analogous to (44). However, the formulation is much more tedious. We defer our treatment of this case to a later section.

2.3.3 Online-Offline Details

In this section we provide a few more details on the computational savings and how to structure the online-offline procedure. In the offline step, obviously the basis choices $\{\zeta_i\}_{i=1}^N$ should be chosen. Once this has been done, the online procedure is to solve for $u(\mu) = \sum_{i=1}^N u_i^N \zeta_i$ via the Galerkin formulation

$$a^N(u, \zeta_j; \mu) = f^N(\zeta_j; \mu) \quad 1 \leq j \leq N, \quad (47)$$

or equivalently

$$\sum_{q=1}^{Q_a} \sum_{i=1}^N u_i^N \Theta_q^a(\mu) a_q(\zeta_i, \zeta_j) = \sum_{q=1}^{Q_f} \Theta_q^f(\mu) f_q(\zeta_j) \quad 1 \leq j \leq N \quad (48)$$

Obviously in the offline stage we can compute and store the $a_q(\zeta_i, \zeta_j)$ and the $f_q(\zeta_j)$. This prevents us from needing to do expensive \mathcal{N}_t -point quadratures to compute $a^N(\zeta_i, \zeta_j; \mu)$ during the online stage. Once we have precomputed these quantities, we are ready to do the online computations, but we do not yet have an efficient way to certify the error bounds. This is where (43) can save us a lot of trouble. The computation of $\vec{\Theta}(\mu)$ requires an \mathcal{N} -independent Q_N computations, which can (must) be done online, but \mathbb{G} given by (42) is μ -independent and can (should) be computed offline and stored. Then the residual norm (43) can be computed in an \mathcal{N}_t -independent way online.

With all of the online-offline decomposition specified, we are now in a position to present an algorithm for the RB method. We recall that our parameter space \mathcal{D} has a fine mesh \mathcal{D}_K with K values, and that we orthogonalize the RB snapshots $u(\mu_i)$ in order to form a basis for X^N ; if this step is unstable, we do not wish to continue choosing basis functions.

Offline Procedure

1.) Determine the values of the coercivity and continuity constants for the problem at hand. Determine $N_{\max}, \varepsilon_{\text{tol}}$.	Operation Count $O(?)$
2.) Construct global inner-product (i.e. mass) matrix for the truth approximation, invert it.	$O(\mathcal{N}_t^3)$
3.) Pick μ_1 via some method, compute $u_1 = u(\mu_1)$, set $\zeta_1 = \frac{u_1}{\ u_1\ _X}$, $X^1 = \text{span}(\zeta_1)$, and compute the 1×1 matrices $(a_q^1) = a_q(\zeta_1, \zeta_1)$ and 1×1 vectors $(f_q^1) = f_q(\zeta_1)$.	$O(\mathcal{N}_t^3 + (1 + Q_1)\mathcal{N}_t)$
4.) Construct the $Q_1 \times Q_1$ residual norm matrix \mathbb{G}_1	$O(\mathcal{N}_t Q_1 + \mathcal{N}_t^2)$
5.) for $m = 1$ to $(N_{\max} - 1)$ for $k = 1$ to K Pick $\tilde{\mu}$ as the k 'th gridpoint in \mathcal{D}_K Form matrix $(a_m)_{ij} = \sum_{q=1}^{Q_a} \Theta_a^q(\tilde{\mu}) (a_q^m)_{ij}$ and vector $(f_m)_j = \sum_{q=1}^{Q_f} \Theta_f^q(\tilde{\mu}) (f_q^m)_j$ for $i, j = 1, \dots, m$ Solve $a_m(u; v; \tilde{\mu}) = f_m(v; \tilde{\mu}) \forall v \in X^m$ Compute (truth) residual norm for the solution u (\mathbb{G}) Compute and store the error estimator Δ_m^k end Determine the k which maximizes Δ_m^k , call the associated μ as $\mu^{(m+1)}$. If $\max(\Delta_m^k) < \varepsilon_{\text{tol}}$: set $N = m$, break Solve $a(u_{m+1}, v; \mu^{(m+1)}) = f(v; \mu^{(m+1)}) \forall v \in X$, set $\zeta_{m+1} = \frac{u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j) \zeta_j}{\ u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j) \zeta_j\ }$, $X^{m+1} = \text{span}(\{\zeta_i\}_{i=1}^{m+1})$ If previous step was unstable: pick different $\mu^{(m+1)}$ Compute the $(m+1) \times (m+1)$ matrices $(a_q^{m+1})_{ij} = a_q(\zeta_i, \zeta_j)$ and $(m+1) \times 1$ vectors $(f_q^m)_j = f_q(\zeta_j)$ Construct the $Q_{m+1} \times Q_{m+1}$ residual norm matrix \mathbb{G}_{m+1} end	$O(Q_a m^2 + Q_f m)$ $O(m^3)$ $O(Q_m + Q_m^2)$ $O(1)$ $O(K)$ $O(\mathcal{N}_t^3 + m\mathcal{N}_t)$ $O((m^2 + m)\mathcal{N}_t)$ $O(Q_{m+1}\mathcal{N}_t + \mathcal{N}_t^2)$

Online Procedure

1.) Given μ , form matrix $(a^N)_{ij} = \sum_{q=1}^{Q_a} \Theta_a^q(\mu) (a_q^N)_{ij}$ and vector $(f^N)_j = \sum_{q=1}^{Q_f} \Theta_f^q(\mu) (f_q^N)_j$ for $i, j = 1, \dots, N$	Operation Count $O(Q_a N^2 + Q_f N)$
2.) Solve $a^N(u, v; \mu) = f(v; \mu) \forall v \in X$	$O(N^3)$
3.) Compute truth residual norm for u (optional)	$O(Q_N + Q_N^2)$

'Unstable' in the offline orthogonalization part of the above algorithm refers to the step of computing $\zeta_{m+1} = \frac{u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j) \zeta_j}{\|u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j) \zeta_j\|}$. If ζ_{m+1} happens to be almost in X^m , then this is an ill-conditioned computation. E.g. if $\|u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j) \zeta_j\|$ is small enough, one should consider picking a different $\mu^{(m+1)}$, perhaps instead one which almost maximizes Δ_m^k . The degenerate case where $\{u(\mu): \mu \in \mathcal{D}\}$ is almost completely contained in X^m should be covered by the 'if $\max(\Delta_m) < \varepsilon_{\text{tol}}$ ' step.

The first step in the above algorithm is to compute appropriate bounds for the coercivity constants. Lemmas 6 and 7 provide very easy methods for doing this, and can be derived analytically, or computed computationally. For the affine, coercive case, this is not so difficult. However, for noncoercive problems, which we shall study in the next chapter, this step is very time-consuming. We have presented the bare-bones algorithm above: the parameter search is simply done over a very fine-grid on \mathcal{D} , and the orthogonalization approach is not very robust. However, this algorithm solidifies the basic idea for the reader. The following chapters will be

devoted to applying the RB method to more complicated (read ‘real-world’) problems, which in turn require more complicated procedures.

2.4 A Special Case: Computing Outputs

This entire chapter has dealt with the case when the bilinear form a is affine and coercive. We now add an additional restriction: we recall that we are interested in some output $s(u; \mu)$. It is now assumed that $s(u; \mu) = f(u; \mu)$. I.e., the output of interest is the same as the right-hand-side of our problem. With all of these restrictions, our problem is called ‘compliant’. Additionally, we impose the restriction that the bilinear form is parameter-independent. This is indeed a harsh restriction, but it is very applicable since, as discussed in chapter 5, a fixed-frequency RCS problem has a parameter-independent bilinear form.

In this section we explore some mathematically interesting (but admittedly computationally useless) features of compliant problems with parameter-independent bilinear forms.

We wish to evaluate some output functional $s(u; \mu)$ where $u = u(\mu)$ satisfies

$$a(u, v) = f(v; \mu) \quad \forall v \in X \quad (49)$$

We assume that X is an \mathcal{N}_t -dimensional space for $\mathcal{N}_t < \infty$. Again we consider the simple problem: assume $s = f$. In this case, s (or f) is linear in u and it is thus by the Riesz Representation Theorem, it is possible to find some element $b: \mathcal{D} \rightarrow X$ such that

$$f(v; \mu) = s(v; \mu) = (v, b(\mu))_X \quad \forall v \in X$$

where (\cdot, \cdot) denotes the inner product. We assume that b is not linear in μ . We now attempt to use the RBM framework to evaluate the output $s(u(\mu); \mu) = (u(\mu), b(\mu))_X$. The straightforward way to do this is to solve (49) for our given μ and then compute $f(u; \mu)$. However, in line with the RB theme, we wish to avoid solving the full system (49): we seek an N -dimensional representation ($N \ll \mathcal{N}_t$) $u^N(\mu) = \sum_{n=1}^N \alpha_n(\mu) u_n$, where $u_n := u(\mu_n)$ are solutions to (49). (Note that we have not opted to orthogonalize the RB basis functions as done previously. This is because theoretically, we need not orthogonalize, and it simplifies exposition here if we do not.) We define X^N as the span of the u_n . The choice of the μ_n is the crucial step here. The choice of the α_n is defined as the solution to the linear system $A \alpha = b$, where $A_{mn} = a(u_m, u_n)$, $b_m = f(u_m; \mu)$, $m, n = 1, \dots, N$. This system is invertible as long as a is well-posed and the u_n are linearly independent. Let us look at this system a little more closely:

We are essentially fixing a $\mu \in \mathcal{D}$ and finding the $u^N \in X^N$ which solves

$$a(u^N, v) = f(v; \mu) \quad \forall v \in X^N$$

We recover the $A \alpha = b$ statement mentioned above by setting $v = u_m$ and writing $u^N = \sum_{n=1}^N \alpha_n u_n$. However, let us restate the problem: we write $f(v; \mu) := (v, b(\mu))_X$ and $a(u^N, v) = a(\sum_{m=1}^N \alpha_m u_m, v) = \sum_{m=1}^N \alpha_m a(u_m, v) = \sum_{m=1}^N \alpha_m f(v; \mu_m) = \sum_{m=1}^N \alpha_m (v, b(\mu_m))_X$ and thus derive the statement

$$\sum_{m=1}^N \alpha_m (v, b(\mu_m))_X = (v, b(\mu))_X \quad \forall v \in X^N$$

or, more suggestively,

$$\left(\sum_{m=1}^N \alpha_m b(\mu_m), v \right)_X = (b(\mu), v)_X \quad \forall v \in X^N \quad (50)$$

This is a variational statement about the (already determined) α_m . The α_m are the coefficients which define the solution u^N as a linear combination of the basis functions u_n . I.e. the α_n define the $L^2(X^N)$ (‘least-squares’) projection of $u(\mu)$ onto X^N . What equation (50) says is that if one considers the $b(\mu_n)$ as a basis for $b(\mu)$, then the *same combinatorial coefficients* α_n define the $L^2(W^N)$ (‘least-squares’) projection of $b(\mu)$ onto $W^N := \text{span}\{b(\mu_1), b(\mu_2), \dots, b(\mu_N)\}$. Just as $u^N = u(\mu)$ in the X^N -sense, so also we have $b^N(\mu) = b^N := \sum_{m=1}^N \alpha_m b(\mu_m)$ in the W^N -sense.

This is no happenstance occurrence. The reason for this apparent magic is that $b(\mu)$ is the solution to the adjoint problem of (49) (see chapter 4 for rigorous introduction). The dual problem is defined by: find $b \in W$ solving

$$a(v, b) = s(v; \mu) \quad \forall v \in X$$

Using the fact that $s = f$, it is easy to show that

$$a(v, b(\mu)) = (v, b(\mu))_X = a(u(\mu), v)$$

Thus we have that $b(\mu)$, which is the particular Riesz function corresponding to the functional f , is exactly the adjoint solution. This only happens when $s = f$.

We can then propose the following algorithm: we need to determine the μ_n . Instead of running a greedy algorithm over (49), which involves some expensive linear solves, why not simply run the greedy algorithm over some explicitly calculable form for $b(\mu)$? The short answer is that there is no particular reason to not do this, but there is a caveat. Running the greedy algorithm over $b(\mu)$ minimizes the error in $b^N(\mu)$ relative to $b(\mu)$. However, it does *not* minimize the error over u^N .

Essentially, we are trying to minimize the quantity $|s(u; \mu) - s(u^N; \mu)|^2$. Via our assumptions, we can write this as $|(u(\mu), b(\mu))_X - (u^N(\mu), b(\mu))_X|^2 = |(u(\mu) - u^N(\mu), b(\mu))_X|^2 \leq \|u(\mu) - u^N(\mu)\|_X \|b(\mu)\|_X$. Thus we have a ‘linear’ dependence on the error in u^N . In fact, using this direct estimator, we have absolutely no control over the error in the output! (The functional f , and thus b , may behave wildly with respect to μ .) The reason, of course, is that we have minimized over the error in the adjoint solution, but we have computed a primal solution functional. Another striking problem with this method is that we don’t actually know what $b(\mu)$ is! It is the adjoint truth solution, and is thus as elusive (read ‘expensive to compute’) as $u(\mu)$. (Note that computing $f(u; \mu)$ and $b(\mu)$ are not the same operations. Computing $f(u; \mu)$ is a very easy quadrature. Computing $b(\mu)$ involves a truth-sized matrix inversion, which is, in principle, just as expensive as inverting $a(u, v)$.) Because of this, we can inexpensively compute the output, but we won’t have an error bound in terms of $b(\mu)$ without doing an expensive calculation.

Of course, a smarter solution would be to recognize that $s(u; \mu) = (u(\mu), b(\mu))_X$. Therefore, instead we should compute $s^N(u; \mu) = (u^N, b^N)_X$, which is a calculable quantity. Thus, our error is

$$\begin{aligned} |s - s^N| &= |(u, b) - (u^N, b^N)| \\ &= |(u - u^N, b - b^N) + (u^N, b) + (u, b^N)| \end{aligned}$$

The second and third terms may seem important, but it turns out that one can eliminate them via a smarter choice of output functional. The key point is that our error, using only the first term, is less than or equal to $\|u - u^N\|_X \|b - b^N\|_X$. With our method of choosing the μ_n only to minimize the error in b^N , we can make the second norm small, but we have no control over the first. The fix for this is to use the greedy algorithm to select two sets of μ_i : one to minimize the error in b^N , and the other to minimize the error in u^N . This Petrov-Galerkin approach yields the ‘superconvergence’ phenomenon [8].

Again, the bottom line is that one can simply choose the μ_i to minimize the error in b^N , and if this is done then, choosing the computed output in a smart way, one can get convergence that scales like $C \|b - b^N\|$, where C depends on how accurately u^N is represented, which may be very inaccurate. However, if one chooses the μ_i to minimize both the u^N and b^N , one can get accelerated convergence.

3 Extensions to Noncoercive Problems

We remind the reader that our ultimate goal is to solve a frequency-domain version of \mathfrak{M} (recall from definition (1) that these are Maxwell's Equations). The simplest version of frequency-domain dynamics, the Helmholtz equation \mathfrak{J} , is not coercive. We have developed the ideas of the previous chapter more as a proof-of-concept that the RB ideas do indeed work, and they work *very* well in canonical cases. However, the RCS problem we wish to solve is hardly ordinary, so we shall have to develop more sophisticated ideas. Because of this, the methods presented in this chapter are valuable.

In this chapter we detail some recent advancement into the application of the RB method to noncoercive problems. One of the main problems with the noncoercive case is that we can no longer derive relatively simple *a posteriori* error bounds. Also, for the symmetric, 'compliant' case as we have described above, the convergence of the output of interest goes like the square of the convergence of the function. This is no longer immediately the case for the noncoercive case. In order to circumvent these setbacks, we must unfortunately subject the reader to a further slew of mathematical definitions and notation.

The following are a list of difficulties that noncoercivity introduces, which we shall address in this chapter:

- There is no longer an equivalent norm induced by the bilinear operator a
- The coercivity constant is nonpositive, and thus we cannot use it in our error estimator

The results from this chapter are taken extensively from results in [9], [10], and [11].

3.1 Mathematical Preliminaries

We ask the reader to now forget our previous coercivity assumption on the bilinear, symmetric, bounded operator a . a is no longer coercive, but it *must* satisfy the inf-sup conditions of definition 4. (If it did not, then the problem may not be well-posed.) We also drop the symmetry assumption: we allow the operator a to be nonsymmetric. We repeat and refine our problem statement: given $\mu \in \mathcal{D}$, we seek to find $u \in X$ satisfying

$$a(u, w; \mu) = f(w; \mu) \quad \forall w \in X \quad (51)$$

Equation (51) is called the *primal* problem, and stems from the physics of the system.

We recall that we've assumed X is a Hilbert space. Because of this, the Riesz Representation theorem tells us that X is isometric to its topological dual space. I.e., let X' be the space of continuous linear functionals on X ; that is, for $f \in X'$, $f: X \rightarrow \mathbb{R}$. Then $X = X'$; more rigorously, for any $x \in X$, there exists some element $x' \in X'$ such that

$$(x, w)_X = x'(w) \quad \forall w \in X \quad (52)$$

and $\|x\|_X = \|x'\|_{X'}$, where

$$\|f\|_{X'} = \sup_{w \in X} \frac{|f(w)|}{\|w\|_X} \quad (53)$$

Another way of stating this is that given any $x' \in X$, we can find a corresponding member of X , call it $\rho_{x'}^X$, such that the operation $x'(\cdot): X \rightarrow \mathbb{R}$ is equivalent to $(\rho_{x'}^X, \cdot)_X$. From this, we can readily infer that

$$\rho_{x'}^X = \operatorname{argsup}_{w \in X} \frac{|x'(w)|}{\|w\|_X} \quad (54)$$

As a very important example, we consider the quantity

$$r(v; u, \mu) = a(u, v; \mu) - f(v; \mu) \quad (55)$$

If we assume u and μ fixed, then we can view the object $r(\cdot; u, \mu)$ as an element of X' . I.e., it linearly maps an element $v \in X$ into \mathbb{R} (or more generally, \mathbb{C}). $r(\cdot; u, \mu)$ is the residual of equation (51). Of course, it is zero for all $v \in X$ if u solves (51). We can identify an element in X with the residual. We define $\hat{e} \in X$ as this element, and $(\hat{e}(u, \mu), v)_X = r(v; u, \mu)$ for all $v \in X$. The appellation is suggestive what it represents: we have defined an element of X which is the representative of the residual. Intuition whispers that this should be the exact error e , but this is not the case unless the bilinear form a is the inner product on X . We shall use both the residual r and its simulacrum in X , \hat{e} , in future sections.

For clarity, we restate the inf-sup condition of definition 4:

$$0 \leq \beta_0 \leq \beta(\mu) \stackrel{\circ}{=} \inf_{u \in X} \sup_{w \in X} \frac{|a(u, w; \mu)|}{\|u\|_X \|w\|_X} \quad \forall \mu \in \mathcal{D} \quad (56)$$

This condition can be restated as saying that given any $\mu \in \mathcal{D}$ and $u \in X$, we can find an element of X , $T_\mu u$, such that

$$\beta(\mu) \|u\|_X \|T_\mu u\|_X \leq |a(u, T_\mu u; \mu)| \quad (57)$$

In other words, we can define a supremizer $T_\mu u$ of the object $a(u, \cdot; \mu) \in X'$. Given our discussion of the object ρ^X above, and in light of equation (54), we see that $T_\mu u = \rho_{a(u, \cdot; \mu)}^X$ so that

$$(T_\mu u, w)_X = a(u, w; \mu) \quad \forall w \in X \quad (58)$$

Because of this, we can readily show that T_μ is uniformly bounded in μ (with continuity constant γ_0). Combining (58) with (56), we can write

$$\beta(\mu) = \inf_{u \in X} \frac{\|T_\mu u\|_X}{\|u\|_X} = \frac{\|T_\mu \chi(\mu)\|_X}{\|\chi(\mu)\|_X} \quad (59)$$

where we've introduced $\chi(\mu) \in X$ as the object that infimizes $\frac{\|T_\mu \cdot\|}{\|\cdot\|}$. Therefore, we can write our inf-sup parameter as

$$\beta(\mu) = \frac{a(\chi(\mu), T_\mu \chi(\mu))}{\|\chi(\mu)\|_X \|T_\mu \chi(\mu)\|_X} \quad (60)$$

We now take a bit of a tangent and discuss the following (unmotivated) eigensystem problem: find $(\lambda(\mu), \Lambda(\mu)) \in \mathbb{R} \times X$ with $\|\Lambda(\mu)\|_X = 1$ such that

$$\mathcal{A}(\Lambda(\mu), w) = \lambda(\mu) (\Lambda(\mu), w)_X \quad \forall w \in X \quad (61)$$

where the symmetric bilinear operator $\mathcal{A}(v, w) := (T_\mu v, T_\mu w)_X$. This may seem like a problem only marginally relevant to our current discussion, but the significance becomes clear upon inspection of the minimum of the Rayleigh Quotient:

$$\lambda_{\min}(\mu) := \min_{w \in X} \frac{\mathcal{A}(w, w)}{(w, w)_X} = \min_{w \in X} \frac{(T_\mu w, T_\mu w)_X}{\|w\|_X^2} = \min_{w \in X} \frac{\|T_\mu w\|_X^2}{\|w\|_X^2} = \beta^2(\mu) \quad (62)$$

I.e. the minimum eigenvalue (or the minimum of the Rayleigh quotient) associated with (61) is the square of our inf-sup parameter. Equation (62) will prove integral in our determination of *a posteriori* error estimates.

Finally, we make some assumptions about the bilinear form a : we assume affinity of the truth-approximation bilinear form, as in equation (22), and that there exist seminorms $\{|\cdot|_q\}_{q=1}^Q$, constants $\{\Gamma_q\}_{q=1}^Q$ all nonzero, and a parameter-independent constant C_X such that

$$|a^q(w, v)| \leq \Gamma_q |w|_q |v|_q \quad (63)$$

$$C_X = \sup_{u \in X} \frac{\sum_{q=1}^Q |u|_q^2}{\|u\|_X^2} \quad (64)$$

The first property is continuity of the a^q with respect to the seminorms $|\cdot|_q$, and the second is a statement about uniform continuity. We shall assume that these norms and constants are readily available. It is very often the case that the definitions of the constants C_X , Γ_q , and the seminorms $|\cdot|_q$ are strongly implied by the form of the particular problem. These last few properties are presently unmotivated: we shall utilize them in a future section.

3.2 Reduced-Basis Approximation

Much of the methodology of the reduced-basis method for noncoercive problems mimics the coercive case (see section 2). However, there are a few key difference between that methodology and this one. We shall outline here the basics of using the RB method for the noncoercive case and briefly describe algorithmic development.

3.2.1 The inf-sup Constant

The procedure for computing *a posteriori* error bounds is what differs the most from the coercive case. In particular, our definition of the error bound in equation (44) no longer makes sense without the existence of a coercivity constant. We shall strive now to make up for this defect. Although we shall not return to the consideration of an *a posteriori* error estimate until section 3.2.5, we do wish to state that the developments in the following sections may seem like discussions tangent to the problem at hand, but they are integral in deriving rigorous error bounds. We ask the reader for some patience.

It turns out that we shall define an error estimator very similar to the one presented in (44), except that we shall use a lower bound for the inf-sup parameter β instead of the coercivity constant α . To this end, we shall begin discussion on finding a lower bound for β .

Given our mathematical discussion in section 3.1, in order to compute β_0 , it suffices to compute

$$\beta_0 = \inf_{\mu \in \mathcal{D}} \inf_{w \in X} \frac{\|T_\mu w\|_X^2}{\|w\|_X^2} \quad (65)$$

It should be relatively clear that the procedure implied above is simply far too expensive, especially since the formation of the operator T_μ involves inversion of the operator $a(\cdot, \cdot, \mu)$ from (58). We must search for alternatives. There have been a few techniques ([10], [9], [12], [11]) proposed to determine lower bounds for β . We shall follow the method in [10].

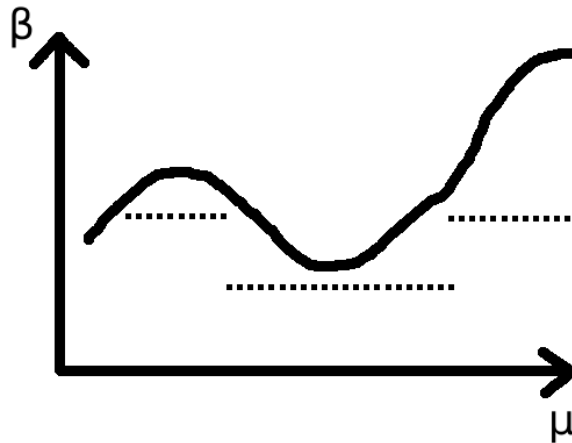


Figure 2. Illustration of the lower-bound technique

The basic idea is the following: we shall separate the parameter space \mathcal{D} into J (possibly) overlapping convex sets \mathcal{P}_j ('polytopes'), and associate some $\bar{\mu}_j \in \mathcal{P}_j$ with each of these sets. We shall use a piecewise-constant approximation of β over these sets \mathcal{P}_j as our lower bound. The graphical illustration when the parameter space \mathcal{D} is one-dimensional is shown in figure 2. To be more precise, we shall use a lower bound

$$\tilde{\beta}(\mu) := \max_{j \in \{1, 2, \dots, J\}; \mu \in \mathcal{P}_j} \varepsilon_\beta \beta(\bar{\mu}_j) \quad (66)$$

where ε_β is some efficiency parameter which we shall discuss later. Of course, we have not yet discussed how to choose the $\bar{\mu}_j$ or the \mathcal{P}_j , how large J should be, or if this is even a good idea at all. We shall elaborate on these features now, drawing extensively from the discussion in [10].

To summarize, we shall approximate $\beta(\mu)$ by a series of piecewise-constant lower-bounds (shown as dotted lines in figure 2). As usual, this is a very simple idea, but requires very sophisticated tools for implementation. Among the ingredients we require are the following

- Given μ , a method for computing/estimating $\beta(\mu)$
- A method for determining the sets over which our lower bound will be constant
- A method for determining the values of the piecewise-constant function

We remark that it is possible to use a piecewise linear lower-bound as well; the formulation is slightly more complex, but has the potential to pay off with a higher estimation efficiency.[10]

We begin by introducing some notation that will be useful. For each polytope \mathcal{P}_j , we associate a collection of vertices \mathcal{V}_j which define the polytope. Recall that $\mu \in \mathcal{D}$ is a P -dimensional quantity, and we denote $\mu_{(p)}$, $p = 1, 2, \dots, P$ as the p 'th component of μ . We make no assumptions on the symmetry of a .

We recall the definition of β and some mathematical forms for it:

$$\begin{aligned} \beta(\mu) &:= \inf_{u \in X} \sup_{v \in X} \frac{|a(u, v; \mu)|}{\|u\|_X \|v\|_X} = \inf_{u \in X} \frac{|a(u, T_\mu u; \mu)|}{\|u\|_X \|T_\mu u\|_X} \\ &\geq \inf_{u \in X} \frac{|a(u, T_{\bar{\mu}} u; \mu)|}{\|u\|_X \|T_{\bar{\mu}} u\|_X} = \inf_{u \in X} \frac{|(T_\mu u, T_{\bar{\mu}} u)_X|}{\|u\|_X \|T_{\bar{\mu}} u\|_X} \end{aligned} \quad (67)$$

We shall essentially want to measure how a changes with respect to μ . In anticipation of this, smoothness of the bilinear form in the parameter is assumed, and we write, for $q = 1, \dots, Q_a$

$$\begin{aligned} \Theta_q^a(\mu) &= \Theta_q^a(\bar{\mu}) + \sum_{p=1}^P \frac{\partial \Theta_q^a}{\partial \mu_{(p)}}(\bar{\mu}) (\mu_{(p)} - \bar{\mu}_{(p)}) + \\ &\quad \left[\Theta_q^a(\mu) - \Theta_q^a(\bar{\mu}) - \sum_{p=1}^P \frac{\partial \Theta_q^a}{\partial \mu_{(p)}}(\bar{\mu}) (\mu_{(p)} - \bar{\mu}_{(p)}) \right] \end{aligned} \quad (68)$$

which is a first-order multivariate Taylor expansion of the Θ_q^a , where we have made use of the sensitivity derivatives $\frac{\partial \Theta_q^a}{\partial \mu_{(p)}}$, which we assume exist and are well-defined for all $\mu \in \mathcal{D}$.

We then write

$$(T_\mu u, T_{\bar{\mu}} v) = (T_\mu u - T_{\bar{\mu}} u, T_{\bar{\mu}} v) + (T_{\bar{\mu}} u, T_{\bar{\mu}} v)$$

Within each polytope centered at $\bar{\mu}$, the last term is a constant. Define $z(u, v, \mu; \bar{\mu}) := (T_\mu u - T_{\bar{\mu}} u, T_{\bar{\mu}} v)$. Using equations (22), (58), and (68), we can write

$$\begin{aligned} z(u, v; \mu; \bar{\mu}) &= a(u, T_{\bar{\mu}} v; \mu) - a(u, T_{\bar{\mu}} v; \bar{\mu}) \\ &= \sum_{q=1}^{Q_a} (\Theta_q^a(\mu) - \Theta_q^a(\bar{\mu})) a_q(u, T_{\bar{\mu}} v) \\ &= \sum_{q=1}^{Q_a} a_q(u, T_{\bar{\mu}} v) \sum_{p=1}^P \frac{\partial \Theta_q^a}{\partial \mu_{(p)}}(\bar{\mu}) (\mu_{(p)} - \bar{\mu}_{(p)}) + \\ &\quad \sum_{q=1}^{Q_a} a_q(u, T_{\bar{\mu}} v) \left[\Theta_q^a(\mu) - \Theta_q^a(\bar{\mu}) - \sum_{p=1}^P \frac{\partial \Theta_q^a}{\partial \mu_{(p)}}(\bar{\mu}) (\mu_{(p)} - \bar{\mu}_{(p)}) \right] \end{aligned} \quad (69)$$

Equation (69) is written only for motivation: the first term is a measure of linearity of a_q with respect to μ . The second term is the deviation from linear behavior. Our lower bound algorithm shall make use of linearity to define a lower-bound value, and shall check the deviation from linearity to determine when a new $\bar{\mu}$ (i.e. a new polytope) needs to be chosen. Now we define

$$\begin{aligned} \mathcal{T}(u, v, \Delta\mu; \bar{\mu}) &:= (T_{\bar{\mu}} u, T_{\bar{\mu}} v)_X + \\ &\quad \sum_{p=1}^P \Delta\mu_{(p)} \sum_{q=1}^{Q_a} \left(\frac{\partial \Theta_q}{\partial \mu_{(p)}}(\bar{\mu}) a_q(u, T_{\bar{\mu}} v) + \overline{\frac{\partial \Theta_q}{\partial \mu_{(p)}}(\bar{\mu}) a_q(u, T_{\bar{\mu}} v)} \right) \end{aligned} \quad (70)$$

where the overbar represents complex conjugate. We associate with (70) a minimum Rayleigh quotient:

$$\mathcal{F}(\Delta\mu; \bar{\mu}) = \min_{w \in X} \frac{\mathcal{T}(w, w, \Delta\mu; \bar{\mu})}{\|w\|_X^2} \quad (71)$$

One can show from (70) and (71) that \mathcal{F} is concave in $\Delta\mu$. Because of this, the set $\mathcal{D}_{\bar{\mu}} := \{\mu \in \mathbb{R}^P: \mathcal{F}(\mu - \bar{\mu}; \bar{\mu}) \geq 0\}$ is convex.

Finally, we define

$$\Phi(\mu, \bar{\mu}) := C_X \max_{q \in \{1, 2, \dots, Q_a\}} \left(\Gamma_q \left| \Theta_q(\mu) - \Theta_q(\bar{\mu}) - \sum_{p=1}^P \frac{\partial \Theta_q}{\partial \mu_{(p)}}(\bar{\mu}) (\mu_{(p)} - \bar{\mu}_{(p)}) \right| \right) \quad (72)$$

which is a bound on that term in (69) which is not included in (70). We shall now state a result without proof:

Proposition 9. (*Piecewise-constant inf -sup bound, [10]*)

Let $\{\mathcal{P}_j\}_{j=1}^J$ be a collection of polytopes satisfying the ‘covering condition’

$$\mathcal{D} \subset \bigcup_{j=1}^J \mathcal{P}_j \quad (73)$$

and the ‘positivity condition’

$$\min_{v \in \mathcal{V}_j} \sqrt{\mathcal{F}(v - \bar{\mu}_j; \bar{\mu})} - \max_{\mu \in \mathcal{P}_j} \Phi(\mu; \bar{\mu}) \geq \varepsilon \beta(\bar{\mu}_j) \quad j = 1, 2, \dots, J \quad (74)$$

Then $\tilde{\beta}(\mu)$ given by (66) satisfies $\tilde{\beta}(\mu) \leq \beta(\mu) \quad \forall \mu \in \mathcal{D}$.

The details of the proof are given in [10] and we refer the reader there for details. A pictorial representation of the covering condition (73) is given in figure 3 for a two-dimensional parameter space.

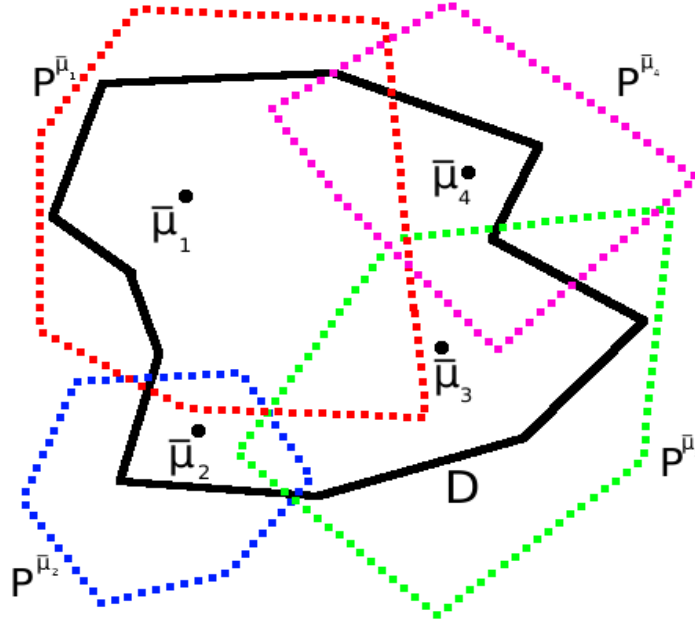


Figure 3. Pictorial representation of covering

Proposition 9 gives us a method of constructing a lower bound for the inf -sup constant. Implementation details will be given in section 3.2.2.

3.2.2 Computation of the Error Bound $\tilde{\beta}(\mu)$

We now turn to the method of constructing the function $\tilde{\beta}(\mu)$ of equation (66). We note that proper construction of the (semi)norms $\|\cdot\|_X$ and $|\cdot|_q$ is integral to the good conditioning of this method. We shall not discuss methods to construct these norms here because for our application, the construction of these norms is natural and robust.

The vast majority of the process for computing $\tilde{\beta}(\mu)$ is an entirely offline procedure, and the online computation requires very little effort. Essentially, offline we compute the \mathcal{P}_j , \mathcal{V}_j , $\bar{\mu}_j$, and $\beta(\bar{\mu}_j)$, and then online all we must do given a μ is to determine which polytope(s) it resides in (i.e. a search algorithm), and then do a very simple maximum calculation described by (66). Once the offline parameters have been computed, then computing $\tilde{\beta}(\mu)$ online is a very fast and straightforward procedure. Therefore, we concentrate on the method of computing the \mathcal{P}_j , \mathcal{V}_j , $\bar{\mu}_j$, and $\beta(\bar{\mu}_j)$.

ε_β is an efficiency parameter with values in $(0, 1)$ for our method. For small $\varepsilon_\beta \ll 1$, we can very easily bound $\beta(\mu)$ with very few polytopes (i.e. J small), but in return we have a very inefficient bound: it will probably be the case that $\tilde{\beta}(\mu) \ll \beta(\mu)$ for a large portion of \mathcal{D} . Conversely, if we choose $\varepsilon_\beta \gg 0$, then we can get very efficient bounds for β , but the price we pay is that we must compute a great many polytopes, which requires considerable offline computational expense. The user may choose whatever ε_β is amenable to her or her tastes.

Given a value for ε_β , we assume at step $j+1$ that we are given the first j polytopes and their respective vertex points. To compute $\bar{\mu}_{j+1}$, we merely select (in some educated fashion) a point in the set $\mathcal{D} \setminus \left(\cup_{i=1}^j \mathcal{P}_i \right)$. Next we compute $\beta(\bar{\mu}_{j+1})$, which is the solution to the eigenproblem (59). This is a nontrivial and costly procedure, but we only need do it J times total. Next, we find the solution v to the nonlinear problem

$$\sqrt{\mathcal{F}(v - \bar{\mu}_{j+1}; \bar{\mu}_{j+1})} - \max_{\mu \in \mathcal{P}_{j+1}} \Phi(\mu; \bar{\mu}_{j+1}) \geq \varepsilon_\beta \beta(\bar{\mu}_{j+1}) \quad (75)$$

where $\tilde{\mathcal{P}}_{j+1}$ is the convex polytope formed by v , any existing members of \mathcal{V}_{j+1} , and $\bar{\mu}_{j+1}$. Note that this problem is extremely costly and complicated to solve. The reasons are

- The value of \mathcal{F} itself requires a minimization solve.
- Finding the max Φ term is an algorithmically nontrivial task to accomplish, especially since the intermediate polytope $\tilde{\mathcal{P}}_{j+1}$ changes based on the choice of v .

However, there are some saving graces for this method, especially applied to our RCS problem:

- Typically, one can use ‘surrogates’ for β and \mathcal{F} . These are inexpensively computed, yet accurate, approximations that depend on the reduced-basis framework. [11], [13].
- For $\mu \in \mathcal{D} \subset \mathbb{R}^P$, one can impose a lexicographic ordering on \mathcal{D} in order to use a binary chop/bisection method to solve (75).
- Most importantly: for our application, $\mu \in \mathbb{R}^3$, but only $\mu_{(1)}$ influences $a(\cdot, \cdot; \mu)$ (see section 5.2.1). Therefore, our polytopes are just line-segment subsets of \mathbb{R} . These observations save us enormous offline computational cost and algorithmic complexity.

Each vertex of \mathcal{V}_{j+1} must be computed using (75). How many vertices to find (i.e. $|\mathcal{V}_{j+1}|$) is open to the implementation. For $\mu \in \mathbb{R}$, convex sets are line segments so that $|\mathcal{V}_j| = 2$ for all j . In higher dimensions, one needs at least $P+1$ points to form a convex set, but there is no upper limit. In practice, once $\bar{\mu}_{j+1} \in \mathcal{P}_{j+1}$, then there is no need to continue.

We begin the process by choosing $\bar{\mu}_1 \in \mathcal{D}$ at random. This defines the ‘centroid’ (in the sense of β) of polytope $\mathcal{P}^{\bar{\mu}_1}$. However, we need to compute the defining vertices \mathcal{V}_j which satisfy the positivity condition (74). To do this, we first solve the eigenvalue problem associated with (62) and compute $\beta(\bar{\mu}_1)$. This is expensive to do. Next, we choose a value of $\varepsilon_\beta \in (0, 1)$, which determines the efficiency of our lower bound. Next we determine the order of the polytope. That is, we fix an integer $|\mathcal{V}_j|$; e.g. for $P=1$, $|\mathcal{V}_j| = 2$ for all j , and for $P=2$ we could (arbitrarily) set $|\mathcal{V}_j| = 3$ for all j .

Now we construct the vertices \mathcal{V}_j such that the positivity condition (74) is satisfied. I.e., for every $\{\mu'_k; k=1, 2, \dots, |\mathcal{V}_1|\}$ we write $|\mathcal{V}_1|$ equations for the μ'_k :

$$\sqrt{\mathcal{F}(\mu'_k - \bar{\mu}_1; \bar{\mu}_1)} - \max_{\mu \in \mathcal{P}_1} \Phi(\mu'_k; \bar{\mu}_1) = \varepsilon_\beta \beta(\bar{\mu}_1), \quad k=1, 2, \dots, |\mathcal{V}_1| \quad (76)$$

The solution of this system gives us the μ'_k . Note that the dominant computational expense is in computing $\mathcal{F}(\mu'_k - \bar{\mu}_1; \bar{\mu}_1)$ because this requires an expensive eigenvalue solve (as is shown in appendix B). The computation of Φ (and even the maximization over the polytope) is quite inexpensive due to the readily-computable form of (72). In particular, many problems have linear parameter dependence, so that $\Phi \equiv 0$ (recall that Φ is a second-order term measuring the deviation from linear parameter dependence). Note that we can efficiently solve this algebraic problem with e.g. binary chop/segmentation algorithms due to the concavity of \mathcal{F} . With this step, we have computed the constant value $\varepsilon_\beta \beta(\bar{\mu}_1)$ and the polytope \mathcal{P}_1 (defined by the vertices \mathcal{V}_j).

We now proceed by induction: given $\{V_j\}_{j=1}^{J-1}$, $\{\bar{\mu}_j\}_{j=1}^{J-1}$, and the associated polytopes $\{\mathcal{P}_j\}_{j=1}^{J-1}$ which all satisfy the positivity condition (74), we choose via some method a $\bar{\mu}_J \notin \bigcup_{j=1}^{J-1} \mathcal{P}_j$. We compute

$$\beta(\bar{\mu}_J) := \sqrt{\min_{w \in X} \frac{\|T_\mu w\|_X^2}{\|w\|_X^2}} \quad (77)$$

by casting it as an eigenvalue problem. We can then solve the system of algebraic equations

$$\sqrt{\mathcal{F}(\mu'_k - \bar{\mu}_J; \bar{\mu}_J)} - \max_{\mu \in \mathcal{P}_J} \Phi(\mu'_k; \bar{\mu}_J) = \varepsilon_\beta \beta(\bar{\mu}_J), \quad k=1, 2, \dots, |\mathcal{V}_J| \quad (78)$$

for the μ'_k , which will define our vertex set $|\mathcal{V}_J|$ and hence our polytope \mathcal{P}_J , which by construction will satisfy the positivity condition. We can then test to see if the covering condition (73) is satisfied. If not, we move on to $J+1$.

Once this step is done, we are essentially finished: we now have a collection of polytopes \mathcal{P}_j and constants $\beta(\bar{\mu}_j)$ which define our inf-sup lower bound (66). The online computation is now merely a task of finding which polytope(s) μ is a member of and then maximizing over the pre-computed $\beta(\bar{\mu}_j)$ associated with those polytopes.

3.2.3 Presenting an Algorithm

We can now write down a full algorithm for computing a lower bound for the the inf-sup parameter β . In order to compute the matrix representations of the operators, we refer the reader to appendix B, whose equations will be referenced in this section. We denote our lower bound as $\hat{\beta}$, as defined by (66). The determination of this function requires us to construct the $\bar{\mu}_j$ and the \mathcal{P}_j , which we do via the following algorithm:

Algorithm Computing the inf-sup parameter $\hat{\beta}$

1. Assume $\varepsilon_\beta \in (0, 1)$, and integers $|\mathcal{V}_j|$ are given
2. Compute \mathcal{P}_1 and $\beta(\bar{\mu}_1)$
 - a. Choose $\bar{\mu}_1$ randomly
 - b. Compute $\beta(\bar{\mu}_1)$ via equation (116)
 - c. Compute the vertices μ'_k via equations (74) and (118)
 - d. Set \mathcal{P}_1 as the polytope defined by the vertices μ'_k , set $j=2$

3. while 1

- a. Choose $\bar{\mu}_j \notin \bigcup_{i=1}^{j-1} \mathcal{P}_i$ via some method
- b. Compute $\beta(\bar{\mu}_j)$ via equation (116)
- c. Compute the vertices μ'_k via equations (74) and (118)
- d. Set \mathcal{P}_j as the polytope defined by the vertices μ'_k , set $j \leftarrow j+1$
- e. If the covering condition (73) is satisfied, **break**

Now we have computed the appropriate polytopes and values and can define our lower bound $\hat{\beta}$. Once this is done, the online computation is very simple: given μ , determine which polytope(s) cover μ , and then take the maximum value of the appropriate $\beta(\bar{\mu}_j)$ as stated in equation (66).

3.2.4 Numerical Example: 1D Helmholtz

To illustrate the lower-bound technique, we apply it to the 1D Helmholtz problem with Dirichlet boundary conditions on the left-hand side and Robin-type outflow boundary conditions on the right-hand side. In this case, the bilinear form $a(u, v; \mu) = a(u, v; \omega)$ takes the form

$$a(u, v; \omega) = a_{\text{lap}}(u, v) + \omega a_{\text{out}}(u, v) + \omega^2 (u, v)_X \quad (79)$$

To see the derivation of this bilinear form for the 1D Helmholtz problem, see section 5.2.1. a_{lap} is the bilinear form associated with the Laplacian, and a_{out} represents the outgoing (Robin-type) boundary conditions. In this case we have $Q_a = 3$, $a_1 = a_{\text{lap}}$, $a_2 = a_{\text{out}}$, $a_3 = (\cdot, \cdot)_X$, and $\Theta_1^a = 1$, $\Theta_2^a = \omega$, and $\Theta_3^a = \omega^2$. We can easily compute the requisite sensitivity derivatives to implement the lower-bound algorithm. Since the parameter is one-dimensional, polytopes are just line segments, and this simplifies the implementation greatly.

With this form for $a(u, v; \omega)$, we implement the lower-bound procedure described in the previous two sections with $\varepsilon_\beta = 0.9$, and this is plotted in figure 4 against the exactly-computed inf-sup parameter.

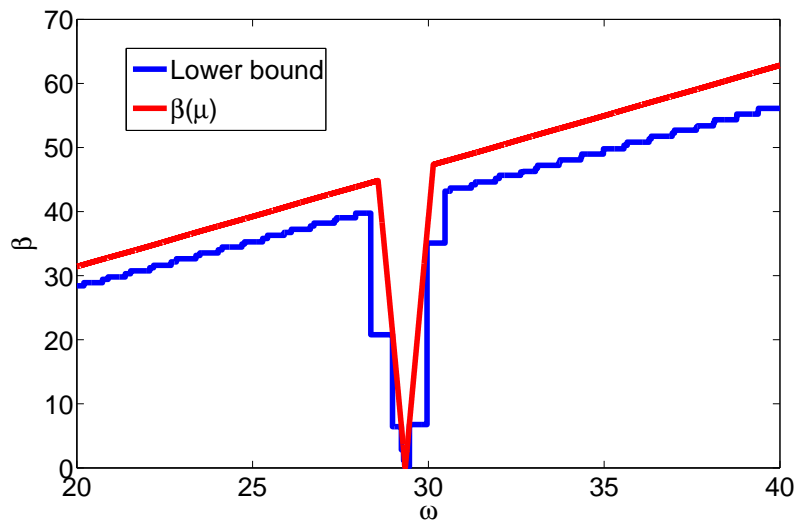


Figure 4. Lower bound for the inf-sup parameter, β .

We see that the lower bound follows the exactly-computed inf-sup parameter quite well, even where $\beta(\mu)$ vanishes near $\omega = 29.5$, a resonance-like phenomenon. $\beta(\mu)$ will be part of our *a posteriori* error bound, as shown in the next section, but where it vanishes, the efficiency of our bound will be very bad. That is, where β vanishes, our estimated error will be much higher than the actual error.

3.2.5 A Posteriori Error Bound

In sections 3.2.1 and 3.2.2 we detailed methods to efficiently compute the inf-sup constant for the bilinear form $a(\cdot, \cdot; \mu)$. The utility of $\tilde{\beta}(\mu)$ will be completely analogous to that of the lower bound of the coercivity constant α_0 in equation (44). We define our *a posteriori* error estimator ([11], [9], [10], [14]) as

$$\Delta_N(\mu) = \frac{\|r(\cdot; u^N, \mu)\|_{X'}}{\tilde{\beta}(\mu)} = \frac{\|\hat{e}(\mu)\|_X}{\tilde{\beta}(\mu)} \quad (80)$$

which is the *a posteriori* estimator for the error in the RB approximation u^N .

Proposition 10. Accuracy of RB error estimator

Assuming well-posedness of the truth-approximation problem (16), then given the form of the error estimator in equation (80), we have

$$\Delta_N(\mu) \geq \|u - u^N\|_X \quad (81)$$

Proof. The proof requires the definition of the inf-sup parameter, its lower bound, and the definition of the residual (55) of the truth-approximation problem:

$$\begin{aligned} \tilde{\beta}(\mu) \leq \beta(\mu) &= \inf_{u \in X} \sup_{v \in X} \frac{|a(u, v; \mu)|}{\|u\|_X \|v\|_X} \\ &\leq \sup_{v \in X} \frac{|a(u - u^N, v; \mu)|}{\|u - u^N\|_X \|v\|_X} \\ &= \frac{1}{\|u - u^N\|_X} \sup_{v \in X} \frac{|f(v; \mu) - a(u^N, v; \mu)|}{\|v\|_X} \\ &= \frac{1}{\|u - u^N\|_X} \sup_{v \in X} \frac{|r(v; u, \mu)|}{\|v\|_X} \\ &= \frac{1}{\|u - u^N\|_X} \|r(\cdot; u, \mu)\|_{X'} \end{aligned}$$

□

As detailed in section 2.3.1, we have an existing online-offline decomposition method for computing $\|\hat{e}\|_X$, and we have just finished explaining the method to calculate $\tilde{\beta}(\mu)$, so we have all the ingredients necessary to calculate Δ_N , which we have just proved is a rigorous upper-bound on the reduced-basis error.

4 Evaluating the Output

In previous chapters, we have been almost exclusively concerned with minimizing the error in our RB solution u^N . However, what we should really be concerned with is how we compute the output s^e . In this chapter we elaborate on methods to compute our actual desired output $s^e(u^e; \mu)$. The reader will note that the greedy algorithm introduced in section 2.3.2 was designed with the goal of minimizing the error in u^N with respect to u . However, it is $s(u; \mu)$ we are actually interested in, not u . While we may be minimizing the error $\|u - u^N\|$, this does not say anything about how small $\|s - s^N\|$ is. This is an unfortunate reality, and we must mitigate this situation.

4.1 The Adjoint Problem

In order to nail down how we can control the error in $s(u, \mu)$, we introduce a new problem: find $v \in X$ such that

$$a(w, v; \mu) = s(w; \mu) \quad \forall w \in X \quad (82)$$

where we recall that $s(u; \mu)$ is our output of interest (e.g. the RCS). Equation (82), the *dual* problem, is a mathematical problem we have introduced; the role of the dual problem is to recover superconvergence of the output functional [8]. We may also write (82) as

$$a^*(v, w; \mu) = s(w; \mu) \quad \forall w \in X$$

where a^* is the formal adjoint operator to a . I.e., a^* is the operator which satisfies $a(v, u; \mu) = a^*(u, v; \mu)$ for all $u, v \in X$.

4.2 Computing the Output

We recall that we've defined primal and dual problems in (51) and (82), respectively. We shall seek to use the dual formulation in order to provide better convergence rates for our output of interest $s(u; \mu)$. Let u^N and v^N be the solutions to the RB primal and dual formulations (51) and (82), respectively. Then the quantities u^N and v^N satisfy

$$\begin{aligned} a(u^N, w; \mu) &= f(w; \mu) \quad \forall w \in X^N \\ a(w, v^N; \mu) &= s(w; \mu) \quad \forall w \in X^N \end{aligned} \quad (83)$$

Furthermore, we define the primal and dual residuals as

$$\begin{aligned} r^{\text{pr}}(w; u^N, \mu) &= a(u^N, w; \mu) - f(w; \mu) \quad \forall w \in X \\ r^{\text{du}}(w; v^N, \mu) &= a(w, v^N; \mu) - s(w; \mu) \quad \forall w \in X \end{aligned} \quad (84)$$

Note that by Galerkin orthogonality, we have

$$\begin{aligned} r^{\text{pr}}(w; u^N, \mu) &= a(u - u^N, w; \mu) \quad \forall w \in X \\ r^{\text{du}}(w; v^N, \mu) &= a(w, v^N - v; \mu) \quad \forall w \in X \end{aligned}$$

Finally, note that by our definitions,

$$r^{\text{pr}}(v^N; u^N, \mu) = a(u^N, v^N; \mu) = r^{\text{du}}(u^N; v^N, \mu) \quad (85)$$

Now consider the following decomposition [8]:

$$\begin{aligned} s(u; \mu) &= s(u^N; \mu) + s(u - u^N; \mu) \\ &= s(u^N; \mu) + a(u - u^N, v; \mu) \\ &= s(u^N; \mu) + a(u - u^N, v^N; \mu) + a(u - u^N, v - v^N; \mu) \\ &= s(u^N; \mu) + r^{\text{pr}}(v^N; u^N, \mu) + a(u - u^N, v - v^N; \mu) \\ &= [s(u^N; \mu) + a(u^N, v^N; \mu) - f(v^N; \mu)] + a(u - u^N, v - v^N; \mu) \end{aligned} \quad (86)$$

The terms in the set of square brackets is computable via our RB primal and dual solutions u^N and v^N , and we define it to be our new estimator of the output:

$$\bar{s}(\mu) = s(u^N; \mu) + a(u^N, v^N; \mu) - f(v^N; \mu) \quad (87)$$

Then assuming well-posedness of the problem, we can state (see section 4.3 for rigorous statement and proof)

$$\|s(u; \mu) - \bar{s}(\mu)\|_X \leq C(\mu) \|u - u^N\|_X \|v - v^N\|_X \quad (88)$$

I.e. if both the primal and dual RB approximations are $O(h^p)$ -accurate, then the desired output is order $O(h^{2p})$ -accurate. We have computed (87) via the primal residual, but due to (85) we could also compute it using the dual residual.

Note that the result (88) as stated applies to the RB solution's error with respect to the truth solution. It can certainly be applied to compare the RB solution to the exact solution, but in order to get bounds on $\|u^e - u^N\|$, we must have a verifiable way of computing quantities related to the exact bilinear formulation.

4.3 Application to the RBM

We have defined a new output estimator in equation (87), and in this section we shall fit this new piece of the puzzle into our reduced-basis methodology. We will use our newly defined dual problem and the new output estimator.

We wish to use the RB method to solve the following problem: find $u, v \in X$ satisfying

$$a(u, w; \mu) = f(w; \mu) \quad \forall w \in X \quad (89)$$

$$a(w, v; \mu) = s(w; \mu) \quad \forall w \in X \quad (90)$$

We note that we have assumed the Galerkin spaces in equations (89) and (90) to be both X . For problems with resonances resulting in spatial singularities, it may be necessary to introduce a primal space X^{pr} and a dual space X^{du} which are refined in different places. For this case, it is possible to present very similar results to what is presented below, but the presentation is much more complex. We shall not deal with this case, and instead refer the interested reader to [14].

Because we have introduced these problems, we must expand our definition of the inf-sup parameter to take these problems into account.

Definition 11. (*The inf-sup parameters for the primal and dual problems*)

We define the primal inf-sup parameter as

$$\beta^{\text{pr}}(\mu) = \inf_{u \in X} \sup_{v \in X} \frac{|a(u, v; \mu)|}{\|u\| \|v\|}$$

and the dual inf-sup parameter as

$$\beta^{\text{du}}(\mu) = \inf_{u \in X} \sup_{v \in X} \frac{|a^*(u, v; \mu)|}{\|u\| \|v\|} = \inf_{v \in X} \sup_{u \in X} \frac{|a(u, v; \mu)|}{\|u\| \|v\|}$$

As before, it is assumed that the primal and bilinear forms have nonzero inf-sup parameters in order to guarantee well-posedness of the problem. Strictly speaking, we should also define the continuity constant $\gamma(\mu)$ for our primal and dual problems as well, but due to the symmetry of the definition of continuity (see definition 4), we have that $\gamma^{\text{pr}} \equiv \gamma^{\text{du}} := \gamma$.

We shall assume that we shall seek reduced-basis solutions $u^N \in X^N$ and $v^M \in X^M$. The corresponding reduced-basis spaces X^N and X^M have dimensions N and M ; problems (89) and (90) are completely different problems; there is no reason to have $X^N = X^M$; indeed there is no apparent reason for them to be similar in any way.

As introduced in section 4.2, we shall use the output estimator

$$\bar{s}(\mu) = s(u^N; \mu) + a(u^N, v^M; \mu) - f(v^M; \mu) \quad (91)$$

and recalling the definitions of the residuals from equation (84), we introduce the error estimators

$$\Delta_N^{\text{pr}} = \frac{\|r^{\text{pr}}(\cdot, u^N; \mu)\|_{X'}}{\beta^{\text{pr}}(\mu)} \quad (92)$$

$$\Delta_M^{\text{du}} = \frac{\|r^{\text{du}}(\cdot, v^M; \mu)\|_{X'}}{\beta^{\text{du}}(\mu)} \quad (93)$$

Lemma 12. *Accuracy of error estimators ([9], [14])*

Assume that both the primal and dual bilinear forms $a(\cdot, \cdot)$ and $a^(\cdot, \cdot)$ induce well-posed truth-approximation problems. Then the primal and dual error estimators satisfy*

$$\begin{aligned} \frac{\Delta_N^{\text{pr}}}{\|u - u^N\|_X} &\geq 1 \\ \frac{\Delta_M^{\text{du}}}{\|v - v^M\|_X} &\geq 1 \end{aligned}$$

Proof. The result is an application of the definition of the inf-sup parameters.

$$\begin{aligned} \beta^{\text{pr}}(\mu) &= \inf_{u \in X} \sup_{v \in X} \frac{|a(u, v; \mu)|}{\|u\| \|v\|} \\ &\leq \sup_{v \in X} \frac{|a(u - u^N, v; \mu)|}{\|u - u^N\| \|v\|} \\ &= \frac{1}{\|u - u^N\|} \sup_{v \in X} \frac{|f(v) - a(u^N, v)|}{\|v\|} \\ &= \frac{\|r^{\text{pr}}(\cdot; u^N, \mu)\|_{X'}}{\|u - u^N\|_X} \end{aligned}$$

Dividing by $\beta^{\text{pr}}(\mu)$ and applying definition (92) yields the result for the primal estimator. The proof for the dual estimator is almost identical. \square

We can now state our main result regarding bounding the convergence of the output in equation (88):

Theorem 13. *Output error bound ([9], [14])*

Assume that the primal and dual bilinear forms are bounded and have nonzero inf-sup parameters. Then

$$|s(u; \mu) - \bar{s}(u^N, v^M; \mu)| \leq \gamma(\mu) \Delta_N^{\text{pr}}(\mu) \Delta_M^{\text{du}}(\mu)$$

where $\bar{s}(u^N, v^M; \mu)$ is defined as in equation (87).

Proof. From equation (86) and application of the definition of continuity we have

$$|s(u; \mu) - \bar{s}(u^N, v^M; \mu)| = |a(u - u^N, v - v^M; \mu)| \leq \gamma(\mu) \|u - u^N\| \|v - v^M\|$$

The result then follows directly from lemma 12. □

We now have the final ingredient necessary for our reduced-basis procedure. We are interested in bounding the error in the output, and theorem 13 provides us a method to do so in terms of the RB error estimators Δ^{pr} and Δ^{du} .

4.4 Algorithm Design

We have defined the adjoint, or dual, problem as in equation (90). We seek to apply the RBM to this adjoint problem, and it should be noted that solving either $a(u, w) = f(w)$ or $a^*(v, w) = s(w)$ is the same abstract problem with a change of notation. Therefore, all the methodology we have developed in chapters 2 and 3 is applicable to the dual problem just as if it were a primal problem.

In particular, computation of the lower bound for the inf-sup parameter is exactly the same, except that the bilinear form for the dual problem is a^* instead of a . Also, the residual-norm matrix \mathbb{G} from equation (42) has a similar definition as the primal problem; one must simply rederive the expression by replacing $a \rightarrow a^*$ and $f \rightarrow s$. (Note that the finite-dimensional equivalent of the adjoint operator $*$ is the Hermitian transpose.)

Using the results of the previous section, we can design a bare-bones algorithm for computing an approximation to the output $s(u; \mu)$ using the RBM. Note that we have presented some details of the RBM in the algorithm in section 2.3.3 and we shall omit those details here. For example, we shall simply say ‘compute Δ^{pr} ’, which entails computing $\|r^{\text{pr}}(\cdot, u^N; \mu)\|_X$, the details of which were presented earlier. We assume the reader is familiar with the basics of the RBM algorithm as presented in section 2.3.3. Also, the following algorithm is not written for optimization; rather it is presented as a rough modus operandi.

Algorithm Output-oriented RBM

Given ε_{tol} , N_{max} , and M_{max} :

- 1.) Compute $\beta^{\text{pr}}(\mu)$, $\beta^{\text{du}}(\mu)$, and $\gamma(\mu)$, and set, e.g., $\varepsilon^{\text{pr}} = \varepsilon^{\text{du}} = \sqrt{\varepsilon_{\text{tol}}}$
 - 2.) Select μ_1^{pr} and μ_1^{du} , and construct the appropriate one-dimensional operators, etc.
 - 3.) For $n = 2: N_{\text{max}}$
 - Search over \mathcal{D} for μ_n^{pr} which maximizes $\sqrt{\gamma(\mu)} \|u(\mu) - u^{n-1}(\mu)\|_X$ using Δ_{n-1}^{pr}
 - If $\sqrt{\gamma(\mu_n^{\text{pr}})} \Delta_{n-1}^{\text{pr}} \leq \varepsilon^{\text{pr}}$ break
 - Add $u(\mu_n^{\text{pr}})$ to $X^{\text{pr}, n-1}$, update operators and estimators
- End
- For $m = 2: M_{\text{max}}$
- Search over \mathcal{D} for μ_m^{du} which maximizes $\sqrt{\gamma(\mu)} \|v(\mu) - v^{m-1}(\mu)\|_X$ using Δ_{m-1}^{du}
 - If $\sqrt{\gamma(\mu_m^{\text{du}})} \Delta_{m-1}^{\text{du}} \leq \varepsilon^{\text{du}}$, break
 - Add $v(\mu_m^{\text{du}})$ to $X^{\text{du}, m-1}$, update operators and estimators
- End

In this way, one finds spaces $X^{\text{pr}, N}$ and $X^{\text{du}, M}$ to find approximations u^N and v^M providing an output estimate $\bar{s}(u^N, v^M; \mu)$ which is guaranteed to be within ε_{tol} of the truth output $s(u; \mu)$.

5 Application to the RCS Problem

In the previous chapters we have discussed the basic ideas of the reduced-basis methods, and in chapter 1 we presented the RCS problem we wish to solve. We shall now describe how we propose to apply the reduced-basis framework to our problem of interest. We shall discuss difficulties and propose some solutions.

We remind the reader that \mathfrak{M} denotes the full homogeneous Maxwell problem given by equations (1)-(4), \mathfrak{C} is the frequency domain curl-curl problem given by (7), and \mathfrak{J} is the Helmholtz equation (8). All these systems have appropriate boundary conditions described by equations (9) and (10). The goal is to compute the RCS σ for many different values of the incident angle φ^{in} , the observed angle φ^{obs} , and the frequency ω . However, it is worth noticing that the computation of the RCS from (12) and (11) involves φ^{obs} only in the output computation: that is, the difficulty in this system is recovering \mathbf{E} and \mathbf{H} , and these quantities have no dependence on the observation angle. Therefore, we shall henceforth restrict our discussion to the computation of the monostatic RCS where $\varphi^{in} = \varphi^{obs} = \varphi$. Computing the bistatic RCS is certainly still a problem to be considered, but almost all of the RB complexity is present in the monostatic case.

The reader will find that we shall make use of most of what has been previously developed. In chapter 2 we discussed the basics of the RBM for the easiest problems in order to familiarize the reader with the method. In chapter 3 we discussed dealing with noncoercive problems, as the Helmholtz problem is if the angular frequency ω is large enough. Finally, in chapter 4 we outlined methods for ensuring accuracy of the desired output (the RCS) versus the numerical solution of the PDE. All of these are crucial components to the solving of the RCS problem.

We also remark that here we are computing the RCS for a fixed object over many angles/frequencies, but using the framework presented in [5], it is also possible to compute the RCS for a family of similarly-structured shapes over many angles and frequencies (as long as the truth approximation retains the same dimension \mathcal{N}_t over the variation in μ). The only difference is the enlarging of the parameter space \mathcal{D} (which, admittedly, does present complications in e.g. computation of the inf-sup lower bound). We shall now state our problems in the language of the reduced-basis methods.

5.1 The Framework

In chapter 1 we presented a number of different formulations of the governing equations of electromagnetic phenomena (namely, \mathfrak{M} , \mathfrak{C} , and \mathfrak{J}). There are two questions which immediately arise:

1. Which system is most amenable to deal with in order to compute the RCS?
2. Which systems can we readily apply the RB method to?

The answer to the first question is relatively clear: because there are already high-order, parallel, unstructured codes in place to compute \mathfrak{M} , (see e.g. [1]) we shall use this solver to make our truth approximations. As regards to the second question, there is very little RB theory at present which applies to time-evolving equations. Therefore, we must restrict our attention to time-independent problems. Thus, either of the Fourier-transformed systems \mathfrak{C} and \mathfrak{J} are applicable; the only question is what the difference between them is and how the RB method can be applied to either of them. The first consideration in this line of thinking is what the easiest thing to do would be. Unfortunately, this is not an easy situation to present a solution: strictly speaking, for given boundary conditions, one can show that when cast into their respective

Galerkin weak forms, systems \mathfrak{C} and \mathfrak{J} exhibit noncoercivity ([15] and [16], respectively). We are thus forced to consider employing the RB method in the noncoercive case. As to the choice between \mathfrak{C} and \mathfrak{J} , the Helmholtz equation assumes a divergence-free electric field, which is true in theory for electrically neutral homogeneous media, as is the case for the RCS. Thus, one may use either the curl-curl formulation, or the Helmholtz formulation as the elliptic problem of interest.

It should be noted that there have been attempts to use the RBM for time-dependent problems ([17], [18], [19]) but they rely on a relatively complicated method of considering the time variable as a parameter, and implementing such a system would require a significant overhaul of existing codes.

The basic strategy then is to use an existing time-domain code as the workhorse for computing frequency-domain ‘truth’ approximations for input into the reduced-basis algorithm. Our methodology for proceeding will be as follows:

1. Use an already-implemented time-domain solver to solve for truth approximations to \mathfrak{M} for the parameter snapshots $\{\mu_i\}_{i=1}^N$.
2. Transform the snapshots of \mathfrak{M} into snapshots of \mathfrak{C} or \mathfrak{J} .
3. Use these transformed snapshots as basis functions ζ_i for our reduced-basis space to solve the time-independent \mathfrak{C} or \mathfrak{J} for the desired range of parameter values.

In order to perform these steps, there are a few ingredients we shall need:

1. A method for determining an error estimate between the solution of \mathfrak{M} transformed to the unknown in \mathfrak{C} or \mathfrak{J} .
2. An RB framework for computing consistently accurate solutions to \mathfrak{C} or \mathfrak{J}

The ultimate goal here is to never be required to discretize or directly solve \mathfrak{C} or \mathfrak{J} , but we shall consider solving these problems in the RBM framework anyway because in one- and two-dimensional test problems, we can verify our algorithm.

The basic, high-level mathematical procedure is the following: we seek a solution u^N , a solution to either \mathfrak{C} or \mathfrak{J} (equations (7) or (8)) using basis functions from the RBM method, and that solution satisfies

$$\|u^N(\mu) - u^e(\mu)\| \leq \underbrace{\|u^N(\mu) - u(\mu)\|}_{\text{RB error}} + \underbrace{\|u(\mu) - u^e(\mu)\|}_{\text{Discretization (DG) error}} \quad (94)$$

The last norm is, in the sense of this analysis, beyond our control: it is the error introduced by the underlying discretization. However, it is for the first quantity that we shall need to compute bounds. The first norm is the error we make with the reduced-basis methods, and as shown in chapters 2 and 3, we can develop sharp *a posteriori* error estimates to certify the accuracy.

In the following sections we shall present more of the details of the RB method applied to this problem. We shall consider the error in our use of the RBM as applied to either \mathfrak{C} or \mathfrak{J} , and the problem of computing outputs. Finally, we shall consider the problem-specific considerations.

5.2 Formulation: The Helmholtz Equation

As discussed in chapter 1, we shall consider our parameters as the angle of incidence (= angle of observation) φ and the angular frequency ω . We thus now define our RB parameter $\mu = (\omega, \varphi) \in \mathbb{R}^+ \times [0, 2\pi) \times [0, \pi] \subset \mathbb{R}^3$ (recall that $\varphi = (\theta, \phi)$). We consider the unknown $\tilde{\mathbf{E}} \in \mathbb{C}^3$ and explicitly write it as a function of μ , while we suppress notation showing dependence on \mathbf{x} . We write down the Helmholtz system \mathfrak{J} as

$$\Delta \tilde{\mathbf{E}}(\mu) + \omega^2 \tilde{\mathbf{E}}(\mu) = 0 \quad \mathbf{x} \in \Omega \quad (95)$$

$$\tilde{\mathbf{E}} \times \hat{\mathbf{n}} = \tilde{\mathbf{E}}_0^{in}(\mathbf{x}) \quad \mathbf{x} \in \partial\Gamma \quad (96)$$

$$\frac{\partial \tilde{\mathbf{E}}}{\partial \hat{\mathbf{n}}} - i\omega \tilde{\mathbf{E}} = 0 \quad \mathbf{x} \in \partial\Omega \quad (97)$$

where $\hat{\mathbf{n}}$ is the outward-pointing normal vector to Γ or $\partial\Omega$ (see figure 1), u_0 is the incident phasor field (see equation (9)), and we assume that ω^2 is not an eigenvalue of the negative Laplacian (i.e. we assume that \mathcal{D} does not contain any eigenvalues of the negative Laplacian). The form of the outflow boundary conditions (97) are discussed in appendix D.

In order to sidestep the necessary (but minimal) augment in notation and definitions for complex-valued functions, we shall opt to solve two versions of (95)-(97): one for $u_0 = \text{Re}[\tilde{\mathbf{E}}_0^{in}]$, and another for $v_0 = \text{Im}[\tilde{\mathbf{E}}_0^{in}]$. Because the system is linear, then we can apply superposition at the end to obtain the full solution. This requires solving two \mathbb{R}^3 -valued systems, which are coupled through the outflow boundary conditions. We shall really only consider solving system (95)-(97) in one and two dimensions. Setting $\mathbf{u} = \text{Re}[\tilde{\mathbf{E}}]$ and $\mathbf{v} = \text{Im}[\tilde{\mathbf{E}}]$, we write the real-valued form of (95)-(97):

$$\left. \begin{aligned} \Delta \mathbf{u}(\mu) + \omega^2 \mathbf{u}(\mu) &= 0 && \text{in } \Omega \\ \mathbf{u} \times \hat{\mathbf{n}} &= \mathbf{u}_0 && \text{on } \partial\Gamma \\ \frac{\partial \mathbf{u}}{\partial \hat{\mathbf{n}}} - \omega \mathbf{v} &= 0 && \text{on } \partial\Omega \end{aligned} \right\} \quad (98)$$

$$\left. \begin{aligned} \Delta \mathbf{v}(\mu) + \omega^2 \mathbf{v}(\mu) &= 0 && \text{in } \Omega \\ \mathbf{v} \times \hat{\mathbf{n}} &= \mathbf{v}_0 && \text{on } \partial\Gamma \\ \frac{\partial \mathbf{v}}{\partial \hat{\mathbf{n}}} + \omega \mathbf{u} &= 0 && \text{on } \partial\Omega \end{aligned} \right\} \quad (99)$$

It is clear that (98) and (99) are coupled through the outflow boundary conditions.

We then discretize equations (98) and (99) with the local discontinuous Galerkin (LDG) method (see appendix C, [20],[21]) and then the entire system takes the form:

$$a((\mathbf{u}, \mathbf{v}), \mathbf{w}; \mu) = f(\mathbf{w}; \mu) \quad \forall \mathbf{w} \in X^3 \times X^3 \quad (100)$$

where the term $f(\mathbf{w}; \mu)$ comes from the Dirichlet boundary conditions (96) and the outflow boundary conditions (97) implemented in the LDG form. The truth space X is the space of piecewise, \mathbb{R} -valued polynomials over the triangulation of the grid. Note that the bilinear operator a in (100) has no dependence on φ , only on ω , as we show in the following section.

The RB method is now implemented almost exactly as outlined in chapter 2. Because the Helmholtz equation is noncoercive (unless ω is small), one must have a way of computing a lower bound for the inf-sup constant (e.g. as done in section 3.2.2) in order to determine *a posteriori* error bounds (section 3.2.5), which are crucial in the offline procedure for choosing the basis functions (see section 2.3.2), as well as the online procedure for accuracy verification. In addition, if we wish to consider computing an output $s((\mathbf{u}, \mathbf{v}); \mu)$, we must also define problems adjoint to (98) and (99) and consider the dual problem as we did in chapter 4.

5.2.1 Implementation: The Operators

As mentioned above, we shall discretize system (98)-(99) using the local discontinuous Galerkin scheme for the Laplacian term. We use this because of the robust stability and accuracy of this method. For the Helmholtz equation, it suffices to just use the appropriate bilinear form for the Laplacian term [21]. We now write $\mathbf{u} = (u_1, u_2, u_3)$ and the test function $\mathbf{w} = (w_1, \dots, w_6) \in X^3 \times X^3$. With this, our bilinear form becomes, e.g. for $\mathbf{w} = (w_1, 0, \dots, 0)$:

$$a((\mathbf{u}, \mathbf{v}), \mathbf{w}; \mu) = a_{\text{lap}}(u_1, w_1) + \omega a_{\text{out}}((\mathbf{u}, \mathbf{v}), w_1) + \omega^2(u_1, w_1)_X \quad (101)$$

where we have explicitly shown dependence on the parameter ω on the right-hand side (recall a does not depend on φ). The operator $a_{\text{lap}}: X \times X \rightarrow \mathbb{R}$ is the symmetric part of the Laplacian LDG discretization. $a_{\text{out}}: (X^3 \times X^3) \times X \rightarrow \mathbb{R}$ is an antisymmetric operator which implements the Neumann/Robin-type boundary conditions specified by outflow. The curious reader wishing for details is encouraged to read appendices C and D.

With the LDG discretization, again with $\mathbf{w} = (w_1, 0, \dots, 0)$ we also have a form for the right-hand-side operator f :

$$f(\mathbf{w}; \mu) = (\alpha(u_{0,1}(\varphi)), w_1)_X \quad (102)$$

where α is some function taking into account both the form of the Dirichlet initial conditions \mathbf{u}_0 and the LDG discretization. It should be noted that \mathbf{u}_0 is a function of the parameter φ , as is explicitly shown, and that α is only nonzero on the scatterer boundary $\partial\Gamma$.

It is evident from equations (101) and (102) that our parameter $\mu = (\omega, \varphi)$ influences the bilinear and right-hand-side operators differently. To be blunt, $a((\mathbf{u}, \mathbf{v}), \mathbf{w}; \mu) = a((\mathbf{u}, \mathbf{v}), \mathbf{w}; \omega)$ and $f(\mathbf{w}; \mu) = f(\mathbf{w}; \varphi)$. This is very beneficial for us: the calculation of the inf-sup parameter lower-bound is prohibitively complex if the μ -dependence in a is more than one-dimensional. Here, ω is one-dimensional, so this computation is simplified.

We can now put this system into the RB framework. The affine decomposition is clear from a generalization of equation (101). We have $Q_a = 3$ with $a_1 = a_{\text{lap}}$, $a_2 = a_{\text{out}}$, and $a_3 = (\cdot, \cdot)_X$. The parameter dependence is $\Theta_1^a(\mu) = 1$, $\Theta_2^a(\mu) = \omega$, and $\Theta_3^a(\mu) = \omega^2$. It is not clear that the right-hand-side operator f has any affine decomposition. However, it is important to notice that since α in equation (102) is a boundary term, it is zero everywhere outside $\partial\Gamma$, and this makes the evaluation of f very cheap overall.

5.2.2 Results: 1D Problem

Using the formulation in section 5.2.1, we implement the one-dimensional Helmholtz equation with using the RBM. As parameter we take the angular frequency ω . Considering φ a parameter in the 1D case is quite trivial and we postpone this until the next section. We solve the one-dimensional version of the Helmholtz formulation presented in section 5.2. The boundary conditions are particularly simple in one dimension:

$$\left. \begin{aligned} u'' + \omega^2 u &= 0 & \text{in } \Omega = [0, 1] \\ u &= 1 & \text{on } x = 0 \\ u' + \omega v &= 0 & \text{on } x = 1 \end{aligned} \right\} \quad (103)$$

$$\left. \begin{aligned} v'' + \omega^2 v &= 0 & \text{in } \Omega = [0, 1] \\ v &= 1 & \text{on } x = 0 \\ v' - \omega u &= 0 & \text{on } x = 1 \end{aligned} \right\} \quad (104)$$

We take $\omega \in \mathcal{D} = [20, 40]$, with $u(x=0) = v(x=0) = 1$ and $x \in [0, 1]$. For a truth approximation, we use the local discontinuous Galerkin method (see appendix C) with 40 equisized cells and a 4th-order polynomial basis on each element. The lower-bound for the inf-sup parameter β is computed as shown in figure 4. There is an exact solution to this problem with the form

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos(\omega x) - \sin(\omega x) \\ \cos(\omega x) + \sin(\omega x) \end{pmatrix}$$

which are the real and imaginary parts of the wave $\sqrt{2}e^{i(\omega x + \pi/4)}$. We take only the primal system (103) and (104); there is no dual system since in this problem we are not computing any output $s(u)$, only the solution itself.

In the RB algorithm, we ask for a maximum error of 10^{-3} before we stop searching for basis functions. The algorithm required 18 basis functions before certifying the error to be less than the desired value.

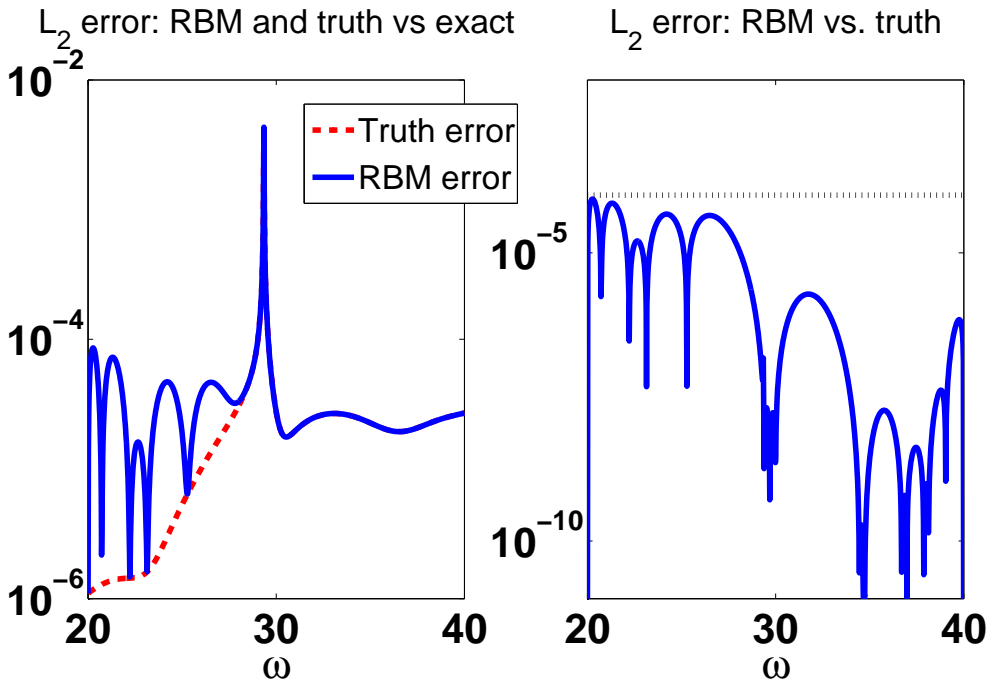


Figure 5. RBM error. Left plot: truth and RBM approximation errors against the exact solution. Right plot: error between truth and RBM solutions; the dotted line represents 10^{-4} , and the maximum error tolerance specified is 5×10^{-3} .

Figure 5 shows the error using the RBM method. The left plot shows the error in the truth solution against the exact solution versus the error in the RB solution against the exact solution, $\|u^e - u\|$ and $\|u^e - u^N\|$. The right plot shows the error between the truth and RB solutions, $\|u - u^N\|$. We can see that $\|u - u^N\|$ is indeed less than 5×10^{-3} as specified. The time necessary to do all the overhead RB processing (computing the lower bound for β , choosing the basis functions) was 98.95 seconds. The plots in figure 5 were generated with 10^4 samples of the parameter space \mathcal{D} . The truth approximation curve (red, dashed line) took 1130 seconds to compute, while the RB method required only 5.48 seconds to produce the blue, solid curve. Thus, taking into account the RB overhead, the total savings was about an order of magnitude. We expect this advantage to grow even more if we move to two or three dimensional problems.

In figure 6 we plot some sample basis functions ζ_j for this problem. Recall that they are orthogonalized against each other (i.e. $\zeta_j \neq u(\mu_j)$), so basis functions corresponding to μ_j for $j > 1$ do not necessarily satisfy the PDE system or the boundary conditions. In figure 1 we show the basis parameters that the algorithm chose.

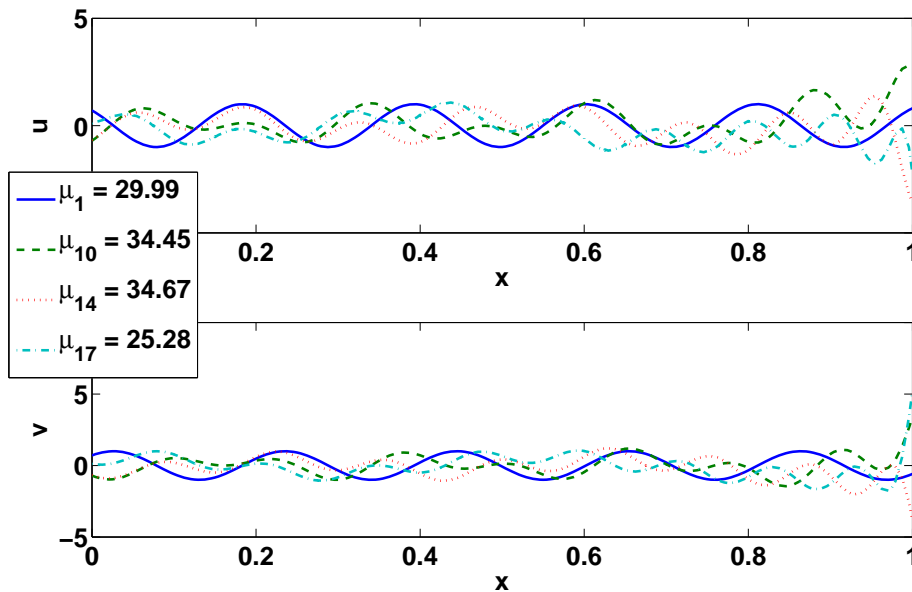


Figure 6. A selection of basis functions chosen by the RB method.

μ_1	29.9990	μ_{10}	34.4494
μ_2	29.3429	μ_{11}	20.0000
μ_3	29.6930	μ_{12}	22.2262
μ_4	40.0000	μ_{13}	23.1423
μ_5	34.7275	μ_{14}	34.6655
μ_6	37.8758	μ_{15}	20.7281
μ_7	36.9757	μ_{16}	36.6997
μ_8	39.0419	μ_{17}	25.2845
μ_9	29.3889	μ_{18}	38.1258

Table 1. Basis parameter choices

In an effort to really test the limits of the RB method, we run the algorithm again on system (103) and (104), but this time we take $\mathcal{D} = [10, 100]$ and again require a maximum error tolerance of 10^{-4} . In this case, the solution varies wildly with the wide range of ω . The sensitivity derivative for ω^2 can be as high as 200. Thus we expect that many basis functions will be necessary to accomplish the same error tolerance. In addition, the inf-sup parameter exhibits three points where it dips down to zero. This will hamper our ability to keep sharp error estimates in those regions.

On this run, the method chose 45 basis functions. The RB overhead ‘offline’ time is 477 seconds with about 80% of that taken up by the computation of the inf-sup lower bound. The online expense is 8.56 seconds for 10^4 solves. The corresponding time required for the truth solver 2419 seconds. We achieve a total computational savings of only about half an order in this case. However, the computational implementation for computing the lower bound is actually relatively inefficient, and since this constitutes the bulk of the required RB time, then optimizing this will yield much more desirable results.

The errors $\|u - u^e\|_X$ and $\|u^N - u^e\|_X$ are plotted in figure 7. For small ω , our error tolerance is so high relative to the truth error, that the RB solution loses many orders of magnitude, but this is completely expected: the RB framework does not seek to bound error relative to the exact solution, only to the truth solution. For large ω , the truth solution error becomes greater than our specified maximum $\|u^N - u\|_X$, so the error curves line up quite well.

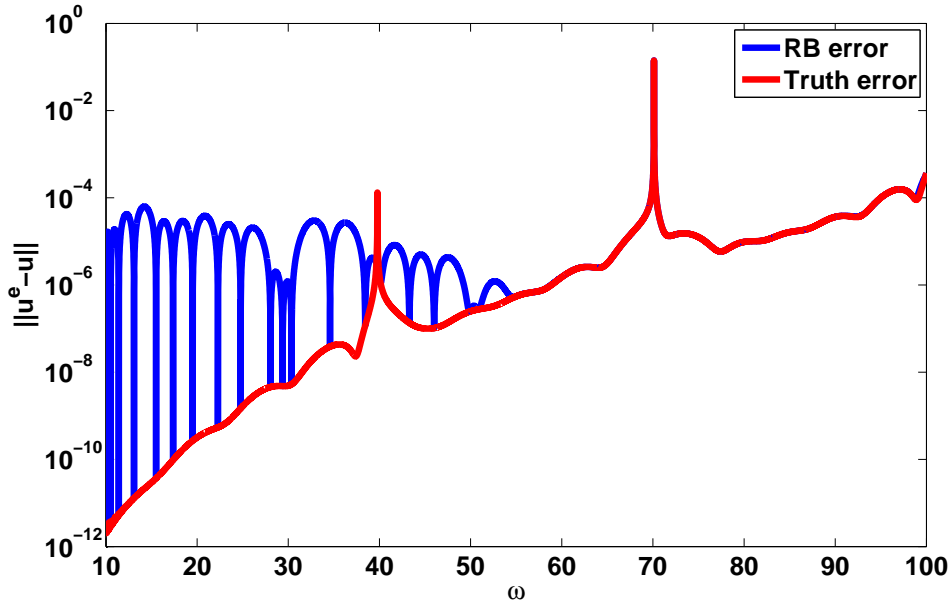


Figure 7. Error between approximations and exact solution for $\mathcal{D} = [10, 100]$.

In figure 8 we show the RB versus truth error, $\|u - u^N\|_X$. The dotted black line represents the specified tolerance. Note how we do not meet the error criterion this time, despite the proof that it should work. The reason for this can be understood in the bottom plot where we plot the calculated lower-bound for β . There are places where β vanishes, but our calculated lower-bound fails to pick this up, and thus our error estimator will not be sharp in these regions.

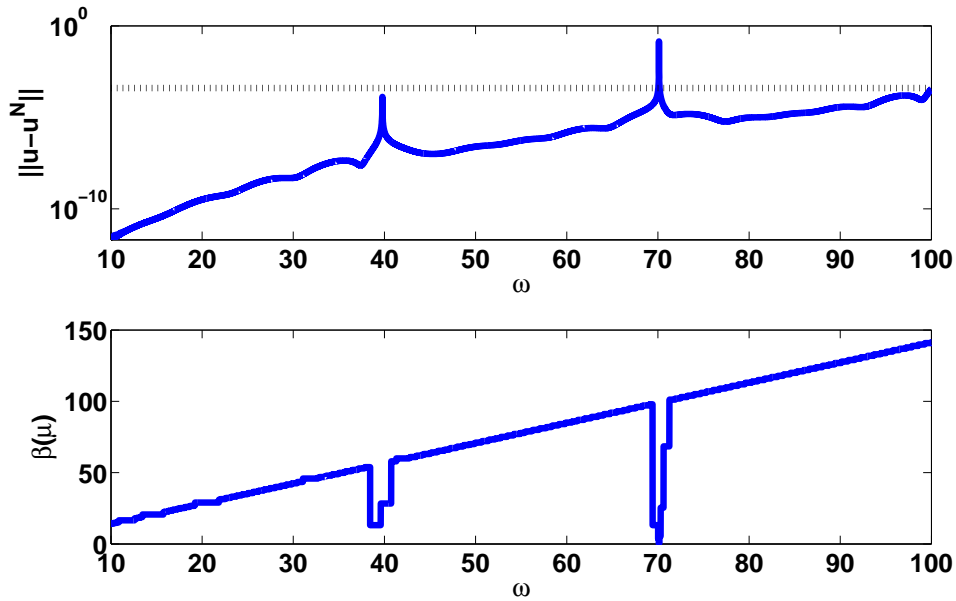


Figure 8. 1D Helmholtz example with $\mathcal{D} = [10, 100]$. Top plot: Error between RB and truth solutions; dotted black line is the 10^{-4} mark, the specified error tolerance. Bottom plot: the calculated lower bound for the inf-sup constant β .

5.3 Fusion with time-domain solvers

In all that we have previously mentioned, we have only dealt with solving a time-independent version of Maxwell's equations in order to calculate the RCS. However, the main goal here is to utilize an already-existing time-domain DG solver to solve this problem. Thus, we shall not actually solve the Helmholtz or curl-curl systems. As mentioned before, the RBM development for time-evolving systems is nascent at best. Therefore, we cannot blindly apply the RBM to this problem if we wish to use a time-domain solver; we have to be a little more clever.

The broad idea is the following: if we wish to use the RBM, *must* mold our problem into a stationary (elliptic) problem. At this point the RB field is too young to allow much else. Thus, we require a method to calculate frequency-domain solutions. We can transform time-domain solutions to frequency-domain solutions using the (fast) Fourier Transform. Of course, this process will not produce results as accurate as if we were actually solving the frequency-domain problem, and we shall have to address this.

5.3.1 Time-to-Frequency Transformation Considerations

We shall generate a time-domain solution to Maxwell's equations (problem \mathfrak{M} with boundary data dependent on φ), which we store for a certain subset $\mathcal{R} \subset \Omega$ and store at discrete time values, $\{t_z\}_{z=1}^Z$, where $t_{z+1} - t_z = \Delta t$ independent of z for all z . Of course, the subset \mathcal{R} could be Ω itself, or a proper subset. The selection of \mathcal{R} shall be considered later. The value of Z depends on a few considerations: we need enough points Z to resolve the particular frequency ω we are interested in. If one waits until a steady-state solution is present, and is relatively confident about the lack of aliasing, then Z can be the minimum number of points necessary to resolve ω (cf. the Nyquist sampling rate).

This provides us with a solution (the electric and magnetic fields), which we shall denote $u_{\mathfrak{M}}^z$. We now seek to transform this solution to the frequency domain and evaluate the result at a particular frequency ω . This should be implemented with an FFT, which is possible under our assumption that Δt be constant. We thus obtain $u(\mu)|_{\mathbf{x} \in \mathcal{R}}$, which is the Fourier transform of the solution to \mathfrak{M} with boundary data dependent on φ evaluated at frequency ω . This solution is what we shall take as our truth approximation to $u(\mu)$, the solution of \mathfrak{C} , or \mathfrak{J} over Ω . These are the solutions we shall use in our global RB approximation (83).

We consider one further step to ensure that the elliptic solution $u(\mu)$ computed via \mathfrak{M} is accurate. An inexpensive way to give additional accuracy to our problem is to use the time-domain fourier-transformed solution as an initial guess to solve the truth-approximation elliptic problem $a(u, v; \mu) = f(v; \mu)$. Because $u(\mu)$ obtained from the time-domain solver is very close to the solution to the elliptic problem, we should get rapid convergence using just a few iterations of an iterative method of choice. This step is not necessary, but it does provide an excellent way to verify that the time-domain solutions solve the problem we want.

5.3.2 The selection of \mathcal{R}

In section 5.3.1 we proposed solving the time-dependent Maxwell's equations system \mathfrak{M} and storing the resulting data for some interval of time over a subset of Ω which we called \mathcal{R} . If we choose $\mathcal{R} = \Omega$, then we must store all the degrees of freedom over all space for a (potentially) long time. This is a major computational burden. In this section we shall consider the possibility of storing the time-dependent solution $u_{\mathfrak{M}}^z$ in a subset \mathcal{R} which is a proper subset of Ω in order to alleviate this burden.

We must determine is what kind of information we actually need about the solution $u(\mu)$ (or equivalently the RB functions ζ_j) in order to proceed with the RBM. We shall only consider the primal RB problem. The following is a rough list of the uses of the solution $u(\mu)$ in the RB algorithm:

- Computing $a_q(\zeta_i, \zeta_j)$, $f_q(\zeta_i)$ to construct a^N and f^N
- Computing $a_q(\zeta_i, \cdot)$ to form residual-norm matrix \mathbb{G} .
- Computing $(u(\mu_i), \zeta_j)_X$ to orthogonalize the basis functions

The first two are absolutely necessary for the RBM. We cannot construct a RB equation, nor can we devise an inexpensive residual estimator without these. The third condition is, theoretically speaking, optional; however, experiments have shown that unless the basis functions are orthogonalized, the resultant bilinear form is very poorly conditioned, or even singular to machine precision when N is even moderately large, e.g. $N > 10$. Therefore we shall take this as a necessary requirement. Can we actually calculate these without knowing the explicit solution $u(\mu_i)$?

Recall that the RB functions are the orthogonalized version of the solutions $u(\mu_i)$. Thus, if we collect the solutions into a matrix \mathbb{U} and the RB functions into another matrix \mathbb{Z} . Then there exists an upper triangular matrix \mathbb{R} such that $\mathbb{Z} \mathbb{R}$ is the QR factorization of \mathbb{U} . In fact, assuming for a moment that we have been orthogonalizing the $u(\mu_i)$ in the RB process, then we can store appropriate values along the RB process to construct \mathbb{R} . Finally, the vector representation of ζ_i is $\vec{\zeta}_i = \mathbb{U} (\mathbb{R}^{-1})_{\cdot i}$, where $\mathbb{D}_{\cdot i}$ represents the i th column of \mathbb{D} . Therefore we can say that $\zeta_i = \sum_{j=1}^N \sigma_{ji} u(\mu_j)$ where σ_{ji} are the matrix components of \mathbb{R}^{-1} .

It is presently inconceivable to be able to explicitly compute $a(\zeta_i, \cdot)$ without knowledge of the entire solution ζ_i . Therefore, we shall attempt instead to take advantage of the fact that $a(u(\mu), v; \mu) = f(v; \mu) \forall v \in X$. The saving grace here is that $f(v; \mu)$ for the RCS is simply a surface integral over $\partial\Gamma$ (see figure 1). Thus, we need only store the values of the solution over $\partial\Gamma$. However, let's take a closer look:

$$\begin{aligned} a(\zeta_i, v; \mu) &= a\left(\sum_{j=1}^N \sigma_{ji} u(\mu_j), v; \mu\right) \\ &= \sum_{j=1}^N \sigma_{ji} a(u(\mu_j), v; \mu) \\ &= \sum_{\{j: \mu_j = \mu\}} \sigma_{ji} f(v; \mu) + \sum_{\{j: \mu_j \neq \mu\}} \sigma_{ji} a(u(\mu_j), v; \mu) \end{aligned} \tag{105}$$

The first term can indeed be computed using only the surface integral over $\partial\Gamma$. However, the second term is more difficult: it is not true that $a(u(\mu_j), v; \mu) = f(v; \mu)$ if $\mu_j \neq \mu$. And even if we try to compute a^q directly, we have no simplifying expression for $a_q(\zeta_i, v)$ or $a_q(u(\mu_i), v)$.

It seems that we must concede that knowledge of the solution everywhere in Ω (i.e. $\mathcal{R} = \Omega$) is necessary as there is at present no foreseeable means to compute some quantities requisite for the RBM without this information.

5.3.3 Bilinear form required

Because we are relying on an already-constructed time-domain solver, it would be quite curious if we were to actually construct the bilinear forms $a_q(\cdot, \cdot; \mu)$ since in that case we would have effectively constructed an elliptic solver for which we would have no need of the time-domain solver. In addition, constructing the matrices takes some extra effort which we are trying to circumvent by using the time-domain solver. In this section we explore methods to avoid explicit construction of the bilinear forms.

It is certainly true that we can construct functions which simply evaluate quantities like $a^q(u, \cdot)$ for an input u which is globally defined over Ω . For a DG-type method, we simply require evaluations of the numerical fluxes, and the rest is quadrature. Can we get by with simply evaluations of the a_q instead of explicit constructions? In contrast to the disheartening conclusion of the last section, the answer here is ‘yes’.

We require a_q for the following operations:

- Computation of the lower bound for the inf-sup parameter β
- Computation of a_q^N

Both of these operations are achievable without explicit construction of a_q . We can create functions to evaluate $a_q(u, \cdot)$, and then use vector inner-products to compute $a_q(u, v)$ for any u, v . The matrices a_q^N are simply evaluations of the form $a_q(\zeta_i, \zeta_j)$, so this is possible.

Regarding computation of the inf-sup parameter, we refer back to sections 3.2.2 and 3.2.3 and to appendix B to conclude that computing such a lower bound requires some eigenvalue solves of matrices which are defined in terms of a . In the notation of appendix B, we need eigenvalues of things like $\mathbb{A}^T \mathbb{X}^{-1} \mathbb{A}$, where \mathbb{A} is the matrix representation of a and \mathbb{X} represents the mass (inner-product) matrix. But doing this only requires us to compute $\mathbb{A}^T \mathbb{X}^{-1} \mathbb{A} u$ for some vector u . Of course, if a routine can be written to evaluate $\mathbb{A} u$, then one can be written to evaluate $\mathbb{A}^T u$, so we should be able to find the appropriate eigenvalues with matrix-vector evaluations.

Thus, we can safely conclude that in order to proceed with the RBM algorithm, it’s sufficient to have routines that evaluate $a(u, \cdot)$ without explicitly constructing that bilinear form.

5.3.4 Adjoint problem

We have one final piece of the puzzle to discuss regarding our solution of the time-evolving problem. In order to achieve the superconvergence phenomenon discussed in chapter 4, we need to define an adjoint or dual problem which is dependent on our output. In section 5.3.5 we described how to achieve this for the RCS in the context of a time-independent elliptic problem. However, we must fit this into our time-evolving algorithm.

Given the monostatic form of the RCS from (11), we seek an output

$$s_{\text{RCS}}(u; \mu) = \frac{\omega^2}{4\pi|u_0|^2} \left| \int_C \left[\frac{i}{\omega} (\hat{\mathbf{n}} \times \tilde{\mathbf{H}}^0) - (\hat{\mathbf{n}} \times u) \times \hat{\mathbf{k}}(\varphi) - (\hat{\mathbf{n}} \cdot u) \hat{\mathbf{k}}(\varphi) \right] e^{-i\mathbf{k}(\omega, \varphi) \cdot \mathbf{x}} d\mathbf{x} \right|^2, \quad (106)$$

where $\tilde{\mathbf{H}}^0 = \nabla \times u$ from (6) and u is the full \mathbb{C}^3 -valued solution to (95) (i.e. the electric field). Note that the $|u_0|^2$ term in the denominator of (106) doesn’t depend on μ due to the form of (9). Any explicit μ -dependence in (106) has been shown (as ω or φ). The problem now becomes approximating s_{RCS} with some reduced-basis solution s_{RCS}^N . The standard way to do this is to introduce the adjoint problem, described in chapter 4. Unfortunately, all of our RB framework hinges on the linearity of the output s . If s is not linear, as is the case with s_{RCS} , then the decomposition provided in (86) is no longer valid and we cannot rigorously bound the error in the output.

For the dual-problem framework, we require a linear output s . Because s_{RCS} is not linear in u (or affine in μ), we must develop another means of estimating the error. We define a linear output $s: X \times \mathcal{D} \rightarrow \mathbb{C}^3$ as

$$s(u; \mu) := \int_C \left[\frac{i}{\omega} (\hat{\mathbf{n}} \times \nabla \times u) - (\hat{\mathbf{n}} \times u) \times \hat{\mathbf{k}}(\varphi) - (\hat{\mathbf{n}} \cdot u) \hat{\mathbf{k}}(\varphi) \right] e^{-i\mathbf{k}(\varphi) \cdot \mathbf{x}} d\mathbf{x} \quad (107)$$

I.e. it is the integral in (106). Also, we define a parameter to measure the maximal strength of the RCS between two solutions:

$$\bar{F}(u, v; \mu) = \max \left\{ \sqrt{s_{\text{RCS}}(u; \mu)}, \sqrt{s_{\text{RCS}}(v; \mu)} \right\}$$

Consider two solutions u and u^N with identical parameters and initial conditions. Then we can calculate:

$$\begin{aligned} |s_{\text{RCS}}(u; \mu) - s_{\text{RCS}}(u^N; \mu)| &= \frac{\omega^2}{4\pi|u_0|^2} \left| |s(u; \mu)|^2 - |s(u^N; \mu)|^2 \right| \\ &= \frac{\omega^2}{4\pi|u_0|^2} \left| (|s(u; \mu)| + |s(u^N; \mu)|) \times \right. \\ &\quad \left. (|s(u; \mu)| - |s(u^N; \mu)|) \right| \\ &\leq \frac{\omega \bar{F}(u, u^N; \mu)}{\sqrt{\pi}|u_0|} \left| |s(u; \mu)| - |s(u^N; \mu)| \right| \\ &\leq \frac{\omega \bar{F}(u, u^N; \mu)}{\sqrt{\pi}|u_0|} |s(u; \mu) - s(u^N; \mu)| \end{aligned} \quad (108)$$

The first inequality utilizes the definition of s in (81) and the second is the ‘reverse’ triangle inequality. We have now expressed the error in s_{RCS} with respect to the error in s . We can now consider our RB problem as one whose desired output is $s(\cdot; \mu)$, and we can make sharp error bounds for s with theorem 13 and the estimator \bar{s} from equation (91), which via (108) we can now extend to s_{RCS} . The superconvergence of the linear output as discussed in chapter 4 is a boon for bounding the error in our desired nonlinear output.

This formulation is quite feasible for the elliptic problems \mathfrak{C} or $\mathfrak{3}$. Unfortunately, for our time-dependent equations \mathfrak{M} , this is much more challenging. Consider what we are doing: we start out with time-dependent Maxwell’s equations and Fourier Transform them. Upon discretization, we are left with the discrete statement: find $u \in X$ satisfying $a(u, w; \mu) = f(w; \mu) \forall w \in X$. However, to guarantee convergence of the output RCS, we must consider the linear functional $s(w; \mu)$ as defined in section 5.3.5 and consider the dual problem: find $v \in X$ satisfying $a^*(v, w; \mu) = s(w; \mu) \forall w \in X$. However, we do not actually solve the elliptic problem, but instead Maxwell’s equations \mathfrak{M} . Now we are left with the puzzle of determining what the corresponding set of time-dependent adjoint equations are.

The adjoint system would be similar to solving an equation of the form

$$\begin{aligned} -\nabla \times \nabla \times \tilde{\mathbf{E}} + \omega^2 \tilde{\mathbf{E}} &= s(\cdot; \mu) \\ &= \int_C \left[\frac{i}{\omega} (\hat{\mathbf{n}} \times \nabla \times \cdot) - (\hat{\mathbf{n}} \times \cdot) \times \hat{\mathbf{k}}(\varphi) - (\hat{\mathbf{n}} \cdot \cdot) \hat{\mathbf{k}}(\varphi) \right] e^{-i\mathbf{k}(\omega, \varphi) \cdot \mathbf{x}} d\mathbf{x} \end{aligned}$$

where $s(\cdot; \mu)$ in the finite-dimensional formulation can be written as some function $g(\mu)$, and can be computed by solving the system

$$s(v; \mu) = (g, v)_X \quad \forall v \in X \quad (109)$$

Then going one step further, one must inverse Fourier-Transform $g(\mu)$ in order to get the contribution to the time-domain system, which perhaps manifests as extra, unphysical right-hand-side inhomogeneities in Maxwell's equations. This is not trivial since $g(\mu)$ depends on ω . This must be explored further in order to determine a proper formulation, or whether this is even possible.

5.3.5 An Alternative For Computing Outputs

In the last section and in chapter 4 we introduced a nonphysical dual problem in order to guarantee convergence of the output, the RCS. This may be undesirable for a large-scale problem and in this section we discuss an alternative method to bound the error in our desired output.

The basic idea is a very simple one: we have a bound on the error in u^N . Can we use this to derive a bound for the output:

$$\|s_{\text{RCS}}(u(\mu); \mu) - s_{\text{RCS}}(u^N(\mu); \mu)\|_X \leq C(\mu) \|u(\mu) - u^N(\mu)\|_X$$

We have already done half of this in section 5.3.4: we've derived a bound of the form

$$\|s_{\text{RCS}}(u) - s_{\text{RCS}}(u^N)\| \leq C(\mu) \|s(u) - s(u^N)\|$$

which is simply a rewrite of equation (108). Now we must bound the error in the output given by equation (107) in terms of the error in u^N , which is a much more manageable task since $s(u; \mu)$ is linear in u .

5.3.6 A Proposed Algorithm

Given the considerations of the previous sections, we can now propose an algorithm which one may be able to use in order to obtain an implementable RB algorithm for the RCS problem utilizing an existing time-domain solver. We shall not dwell on the exact computations as those have been explored in previous sections, and the RBM algorithm is presented in much more detail in appendix E.

We do not consider bounding error in the RCS output in this algorithm since this is still an open problem given the considerations of the previous sections.

Algorithm : RBM Computation of RCS using existing solver – Angle Variation Only

Given $\varepsilon_{\text{tol}}, N_{\text{max}}$:

1. Define the functions to evaluate $a(u, v; \varphi)$ and $f(v; \varphi)$ which form the primal elliptic problem. Explicitly construct $f_q(v)$.
2. Compute the inf-sup parameter once directly since $a(u, v; \varphi) = a(u, v)$ does not depend on the incident angle φ . This will be used to define the error estimator.
3. Determine an appropriate parameter space $\varphi \in \mathcal{D}$ and discretize it appropriately. Set $j = 1$. Choose an appropriate φ_1 .
4. While $j \leq N_{\text{max}}$
 - a. Use the time-domain solver to find and store $u_{\mathfrak{M}}^{\tilde{z}}(\varphi_j)$ on all space over a time interval which exhibits a steady-state solution.
 - b. Fourier Transform $u_{\mathfrak{M}}^{\tilde{z}}(\varphi_j)$ to obtain the elliptic solution $u(\varphi_j)$ over all space.

- c. Explicitly construct $f(v; \varphi_j)$ using $f_q(v)$, and use an iterative method to solve $a(u, v; \omega_j) = f(v; \varphi_j)$ with $u = u(\mu_j)$ as an initial guess. Redefine $u(\mu_j)$ to be this solution.
- d. Orthogonalize it against the previous basis functions $\{\zeta_i\}_{i=1}^{j-1}$. Define the result as ζ_j .
- e. Form the galerkin matrix $(a^j)_{rs}$ by computing $a(\zeta_r, \zeta_s)$. Form the right-hand side vectors $(f_q^j)_r$ by evaluating $f_q(\zeta_r; \varphi)$. Use these to form the residual-norm computing matrix.
- f. Using the residual-norm computing matrix and the inf-sup lower bound, solve $a^j(u, v) = f^j(v; \varphi) \forall v \in X^j$ for each discrete value of φ and find the φ which maximizes the error. If the error is less than ε_{tol} , quit. Otherwise, set φ_{j+1} to be this maximizing value. $j \leftarrow j + 1$.

The algorithm for frequency and angle variation is almost identical, with the exception of needing to use the results in chapter 3 to compute a lower bound for the inf-sup parameter.

Algorithm : RBM Computation of RCS using existing solver – Frequency and Angle Variation

Given $\varepsilon_{\text{tol}}, N_{\text{max}}$:

1. Define the functions to evaluate $a(u, v; \omega)$ and $f(v; \varphi)$ which form the primal elliptic problem. Explicitly construct $f_q(v)$.
2. Determine an appropriate parameter space $\mu = (\omega, \varphi) \in \mathcal{D}$ and discretize it appropriately.
3. Compute a lower bound for the inf-sup parameter. This should only be done over the ω portion of \mathcal{D} since the bilinear form does not depend on φ .
4. Set $j = 1$. Choose $\mu_1 = (\omega_1, \varphi_1)$ in some fashion.
5. While $j \leq N_{\text{max}}$
 - a. Use the time-domain solver to find and store $u_{\mathfrak{M}}^z(\mu_j)$ on all space over a time interval which exhibits a steady-state solution.
 - b. Fourier Transform $u_{\mathfrak{M}}^z(\mu_j)$ to obtain the elliptic solution $u(\mu_j)$ over all space.
 - c. Explicitly construct $f(v; \varphi_j)$ using $f_q(v)$, and use an iterative method to solve $a(u, v; \omega_j) = f(v; \varphi_j)$ with $u = u(\mu_j)$ as an initial guess. Redefine $u(\mu_j)$ to be this solution.
 - d. Orthogonalize it against the previous basis functions $\{\zeta_i\}_{i=1}^{j-1}$. Define the result as ζ_j .
 - e. Form the galerkin matrices $(a^j)_{rs}$ by computing $a_q(\zeta_r, \zeta_s, \omega)$. Form the right-hand side vector $(f_q^j)_r$ by evaluating $f_q(\zeta_r; \varphi)$. Use these to form the residual-norm computing matrix.
 - f. Using the residual-norm computing matrix and the inf-sup lower bound, solve $a^j(u, v, \omega) = f^j(v; \varphi) \forall v \in X^j$ for each discrete value of (ω, φ) and find the (ω, φ) which maximizes the error. If the error is less than ε_{tol} , quit. Otherwise, set $\mu_{j+1} = (\omega, \varphi)_{j+1}$ to be this maximizing value. $j \leftarrow j + 1$.

Appendix A The Kolmogorov N -width

We describe in this section a more rigorous way of explaining exactly why we expect the reduced-basis method to work.

[22], [14], [23],[24], [10], [25], [26], [27], [28], [6],

Appendix B Finite-Dimensional Representations

In this appendix we shall translate some of the notation introduced in sections 3.2.2 and 3.1 into matrix notation so that the implementation becomes more clear. First of all, we assume that X is a finite-dimensional space of dimension \mathcal{N}_t . Suppose that a basis for X is the collection of functions $\{\phi_i\}_{i=1}^{\mathcal{N}_t}$. We define the $\mathcal{N}_t \times \mathcal{N}_t$ matrix \mathbb{X} as

$$\mathbb{X}_{ij} = (\phi_i, \phi_j)_X \tag{110}$$

Then \mathbb{X} represents the inner product of X . I.e., given, $u, v \in X$,

$$(u, v)_X = \mathbf{u}^T \mathbb{X} \mathbf{v} \quad (111)$$

where \mathbf{u} and \mathbf{v} are the $\mathcal{N}_t \times 1$ -sized vectors of modal coefficients for u and v , respectively. I.e.

$$\mathbf{u}_i = (\mathbb{X}^{-1})_{ij} (u, \phi_j)_X \quad i = 1, 2, \dots, \mathcal{N}_t \quad (112)$$

where we use Einstein's summation notation and sum over repeated indices. Next we introduce the finite-dimensional representation of $a(\cdot, \cdot, \mu)$. Let the $\mathcal{N}_t \times \mathcal{N}_t$ matrix \mathbb{A}_μ be defined by

$$(\mathbb{A}_\mu)_{ij} = a(\phi_i, \phi_j; \mu) \quad (113)$$

Then for all $u, v \in X$ we have

$$a(u, v; \mu) = \mathbf{v}^T \mathbb{A} \mathbf{u} \quad (114)$$

We also denote the matrices \mathbb{A}^q as the discrete representations of the operators $a^q(\cdot, \cdot)$. The forms for these can be derived exactly as in (113). Given these definitions, and the form of equation (58), we define

$$\mathbb{T}_\mu = \mathbb{X}^{-1} \mathbb{A}_\mu \quad (115)$$

as the matrix form of the supremizing operator T_μ . Now the eigenvalue problem (62) can be written as

$$\beta^2(\mu) = \lambda_{\min} = \min_{\mathbf{w} \in X} \frac{\mathbf{w}^T \mathbb{T}_\mu^T \mathbb{X} \mathbb{T}_\mu \mathbf{w}}{\mathbf{w}^T \mathbb{X} \mathbf{w}} = \min_{\mathbf{w} \in X} \frac{\mathbf{w}^T \mathbb{A}_\mu^T \mathbb{X}^{-1} \mathbb{A}_\mu \mathbf{w}}{\mathbf{w}^T \mathbb{X} \mathbf{w}} := \min_{\mathbf{w} \in X} \frac{\mathbf{w}^T \mathbb{S}_\mu \mathbf{w}}{\mathbf{w}^T \mathbb{X} \mathbf{w}} \quad (116)$$

where we have used the fact that \mathbb{X} is symmetric (or hermitian), and have defined a new matrix \mathbb{S}_μ . The form of equation (116) tells us that searching for λ_{\min} (or, equivalently, $\beta(\mu)$) amounts to a generalized eigenvalue problem for the matrix \mathbb{S}_μ .

We have thus introduced the discrete, implementable form for computing $\beta(\bar{\mu}_J)$ in equation (77). In order to solve (78), we need to compute the value of \mathcal{F} . This can also be cast as an eigenvalue problem. We recall the form of $\mathcal{T}(w, w, t; \bar{\mu})$ in equation (70) and seek to write it in a discrete form. Using our previous matrix definitions, we can write

$$\mathbb{Q}(t, \bar{\mu}) = \mathbb{S}_{\bar{\mu}} + \sum_{p=1}^P t^{(p)} \sum_{q=1}^Q \left(\frac{\partial \Theta^q}{\partial \mu^{(p)}}(\bar{\mu}) \mathbb{T}_{\bar{\mu}}^T \mathbb{A}^q + \frac{\partial \overline{\Theta^q}}{\partial \mu^{(p)}}(\bar{\mu}) \overline{\mathbb{T}_{\bar{\mu}}^T \mathbb{A}^q} \right) \quad (117)$$

as the discrete form of $\mathcal{T}(\cdot, \cdot, t; \bar{\mu})$. We can then recall the definition of \mathcal{F} in equation (71) and conclude that

$$\mathcal{F}(t; \bar{\mu}) = \rho_{\min} = \min_{\mathbf{w} \in X} \frac{\mathbf{w}^T \mathbb{Q}(t, \bar{\mu}) \mathbf{w}}{\mathbf{w}^T \mathbb{X} \mathbf{w}} \quad (118)$$

I.e., that \mathcal{F} is the result of a minimum generalized eigenvalue problem for the matrix $\mathbb{Q}(t, \bar{\mu})$.

Appendix C The Local Discontinuous Galerkin Method

C.1 Problem Statement

Given a domain $\Omega \in \mathbb{R}^d$ bounded by the set $\partial\Omega = \Gamma^D \cup \Gamma^N \cup \Gamma^r$, we wish to solve the Poisson boundary-value problem for the function $u : \Omega \rightarrow \mathbb{R}$

$$\left\{ \begin{array}{ll} \Delta u = f & x \in \Omega \\ u = g & x \in \Gamma^D \\ \frac{\partial u}{\partial n} = h & x \in \Gamma^N \\ a u + b \frac{\partial u}{\partial n} = k & x \in \Gamma^r \end{array} \right. \quad (119)$$

We assume that Γ^D , Γ^N , and Γ^r are mutually nonoverlapping sets which cover $\partial\Omega$. The quantities g , h , k , a , and b are real-valued functions defined on their appropriate domains. f is real-valued and defined on all Ω . $\frac{\partial u}{\partial n}$ is the directional derivative in the outward-pointing direction from the domain Ω . We denote $\hat{\mathbf{n}}$ as the outward-pointing normal vector. Thus we can write $\frac{\partial u}{\partial n} = \nabla u \cdot \hat{\mathbf{n}}$.

We note that the Neumann boundary condition is just a special case of the Robin boundary condition when $a = 0$ and $b = 1$. We thus define $\Gamma^R := \Gamma^r \cup \Gamma^N$ and extend the definitions of a , b , and k over Γ^r such that $a|_{\Gamma^N} = 0$, $b|_{\Gamma^N} = 1$, and $k|_{\Gamma^N} = h$. It is also the case that the Dirichlet boundary condition is a special case of the Robin condition, but our treatment of the Dirichlet-type and Robin-type conditions is markedly different, so we opt to leave these two separated.

C.2 Discontinuous Finite Element Geometry

We assume that the domain Ω is subdivided into K nonoverlapping subdomains Ω^k covering Ω . We assume that a function u defined on Ω has a discrete representation of the form

$$u_h = \sum_{k=1}^K u_h^k(x) = \sum_{k=1}^K \sum_{n=1}^N u(x_n^k) l_n^k(x) \quad (120)$$

where the $l_n^k(x)$ are the local Lagrange polynomials associated with the node x_n^k . Because the functions u_h^k are entirely local, the function u_h can be possibly discontinuous at element interfaces. We denote the set of all possible realizations of the function u_h as the space V_h , which has dimension KN .

We shall consider the set of all element interfaces as Γ (which includes the boundary edges). The set of interior edges Γ^0 is the collection of points in Γ not on $\partial\Omega$. To be precise, $\Gamma^0 := \Gamma \setminus \partial\Omega$. With this definition and noting that $\partial\Omega = \Gamma^D \cup \Gamma^R$, we have $\Gamma = \Gamma^D \cup \Gamma^R \cup \Gamma^0$.

Inside the element Ω^k at an element interface, we denote the local value of the function u_h as u^- , and the exterior (neighboring) value of the function as u^+ . We then define $\hat{\mathbf{n}}^-$ as the outward-pointing normal vector, and $\hat{\mathbf{n}}^+$ as the inward-pointing normal vector. Inside an element, we take $u^+ \equiv u^-$ and $\hat{\mathbf{n}}^\pm = \mathbf{0}$. We then define the average and jump operators:

$$[[u]] = u^- \hat{\mathbf{n}}^- + u^+ \hat{\mathbf{n}}^+ \quad \{u\} = \frac{1}{2}(u^- + u^+) \quad (121)$$

It will also be necessary to consider the jump and average operators acting on a function $\sigma : \Omega \rightarrow \mathbb{R}^d$. Thus we also define

$$\llbracket \sigma \rrbracket = \sigma^- \cdot \hat{n}^- + \sigma^+ \cdot \hat{n}^+ \quad \{\sigma\} = \frac{1}{2}(\sigma^- + \sigma^+)$$

On the boundary $\partial\Omega$, we define $\llbracket u \rrbracket = u^- \hat{n}^-$, $\llbracket \sigma \rrbracket = \sigma^- \cdot \hat{n}^-$, $\{u\} = u^-$, and $\{\sigma\} = \sigma^-$.

C.3 DG Strategy

In this report, we closely follow the analysis presented in [21] and we shall adopt much of the notation presented there. The general DG formulation for elliptic equations is to define an \mathbb{R}^d -valued auxilliary variable $\sigma := \nabla u$, and use it to aid in solving for the primal variable u . We assume that σ also has a discrete representation, σ_h , where each component can be expanded as in equation (120). We define the space of all possible realizations of σ_h as Σ_h . We define the operator ∇_h as the operator whose action on each individual element exactly mimics the gradient operator. I.e. $\nabla u_h^k = \nabla_h u_h^k$ on Ω^k for every $u_h^k \in V_h$.

As shown in [21], it is possible to write a primal-form local Discontinuous Galerkin bilinear form. For every $v \in V_h$, we define

$$B_h(u_h, v) = \int_{\Omega} \nabla_h u_h \cdot \nabla_h v + \int_{\Gamma} (\llbracket \hat{u} - u_h \rrbracket \cdot \{\nabla_h v\} - \{\hat{\sigma}\} \cdot \llbracket v \rrbracket) + \int_{\Gamma^0} (\{\hat{u} - u_h\} \llbracket \nabla_h v \rrbracket - \llbracket \hat{\sigma} \rrbracket \{v\}) \quad (122)$$

Of course, the main challenge and strength of the DG method is the choice of the numerical fluxes \hat{u} and $\hat{\sigma}$.

Before continuing on to select these fluxes, we first remark that we shall also require use of the somewhat esoteric ‘lifting’ operators $r : \Sigma_h \rightarrow \Sigma_h$ and $l : V_h \rightarrow \Sigma_h$, which are defined in the following way:

$$\int_{\Omega} r(\phi) \cdot \tau = - \int_{\Gamma} \phi \cdot \{\tau\} \quad (123)$$

$$\int_{\Omega} l(v) \cdot \tau = - \int_{\Gamma^0} v \llbracket \tau \rrbracket \quad (124)$$

In addition we define a lifting operator r^D , which is derived from r :

$$\int_{\Omega} r^D(\phi) \cdot \tau = - \int_{\Gamma^0 \cup \Gamma^D} \phi \cdot \{\tau\} \quad (125)$$

If we consider the characteristic function $\chi_A(x) = 1$ for $x \in A$ and $\chi_A(x) = 0$ for $x \in A$, then it is clear that $r(\phi^D) = r(\phi \chi_{\Gamma^0 \cup \Gamma^D}) = r^D(\phi)$ where we have defined ϕ^D as the restriction of ϕ to $\Gamma^D \cup \Gamma^0$. Also, we have that $l(v \chi_{\partial\Omega}) = 0$.

When specifying the fluxes to insert into (122), we shall need is an expression which relates the auxilliary discrete variable σ_h to the primal discrete variable u_h . [21] provides the following relation:

$$\sigma_h = \nabla_h u_h - r(\llbracket \hat{u} - u_h \rrbracket) - l(\{\hat{u} - u_h\}) \quad (126)$$

The final ingredient is the Galerkin statement of determining the solution u_h . This statement is

$$B_h(u_h, v) = \int_{\Omega} f v \quad \forall v \in V_h \quad (127)$$

Following this section, we shall use (122)–(127) to derive an appropriate form to solve for u_h without explicitly requiring an auxilliary variable.

C.4 The LDG Formulation

We now consider the core of the local DG method: the choice of the fluxes and the subsequent derivation of the auxilliary-free primal form. We define $\boldsymbol{\beta} : \Gamma^0 \rightarrow \mathbb{R}^d$ as vector function defined on edges which is constant on each edge. Usually one can take $\boldsymbol{\beta} = \hat{\mathbf{n}}^{\pm}$. Although we shall only require $\boldsymbol{\beta}$ on interior edges, it will simplify future derivations to define $\boldsymbol{\beta} = \mathbf{0}$ on $\partial\Omega$. Additionally, we introduce the penalty term $\mu = \alpha h_e^{-1}$ where h_e is the length of the edge, and α is some positive number, which must be large enough to penalize the discontinuities at interfaces. For our Dirichlet/Robin type problem specified in equation (119), we choose the following fluxes:

$$\left. \begin{aligned} \hat{u} &= \{u_h\} - \boldsymbol{\beta} \cdot \llbracket u_h \rrbracket \\ \hat{\boldsymbol{\sigma}} &= \{\boldsymbol{\sigma}_h\} + \boldsymbol{\beta} \llbracket \boldsymbol{\sigma}_h \rrbracket - \mu \llbracket u_h \rrbracket \end{aligned} \right\} \text{on } \Gamma^0 \quad (128)$$

$$\left. \begin{aligned} \hat{u} &= g \\ \hat{\boldsymbol{\sigma}} &= \boldsymbol{\sigma}_h - \mu \llbracket u_h \rrbracket + \mu g \hat{\mathbf{n}} \end{aligned} \right\} \text{on } \Gamma^D \quad (129)$$

$$\left. \begin{aligned} \hat{u} &= u_h \\ \hat{\boldsymbol{\sigma}} &= \frac{1}{b}(k - a u_h) \hat{\mathbf{n}} \end{aligned} \right\} \text{on } \Gamma^R \quad (130)$$

With these fluxes, we shall seek to write an implementable form for our galerkin statement (127). To this end, we first consider writing a form for $\boldsymbol{\sigma}_h$ now that the fluxes have been defined. Some very useful properties to note are that both the fluxes \hat{u} and $\hat{\boldsymbol{\sigma}}$ are continuous across interior interfaces, and that the jump and average operators, as well as all the lifting operators, are linear.

Since $\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket$ and $\{u_h\}$ are continuous across interfaces, we can write

$$\llbracket \hat{u} - u_h \rrbracket = - \llbracket u_h \rrbracket \chi_{\Gamma^D \cup \Gamma^0} + g \chi_{\Gamma^D} \hat{\mathbf{n}}$$

and we also have that

$$\{\hat{u} - u_h\} = - \boldsymbol{\beta} \cdot \llbracket u_h \rrbracket + \frac{1}{2}(g + u_h) \chi_{\Gamma^D}$$

It turns out that we shall never need $\{\hat{u} - u_h\}$ on $\partial\Omega$, so in the above equation we shall omit the second term on the right-hand-side in all future computations.

Using these expressions in equation (126), we have

$$\boldsymbol{\sigma}_h = \nabla_h u_h + r^D(\llbracket u_h \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) - r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \quad (131)$$

Using this expression, we shall now derive an expression for one of the terms in the bilinear form:

$$\begin{aligned}
\int_{\Gamma^0 \cup \Gamma^D} \{\hat{\boldsymbol{\sigma}}\} \cdot \llbracket v \rrbracket &= \int_{\Gamma^0 \cup \Gamma^D} \{\nabla_h u_h\} \cdot \llbracket v \rrbracket + \int_{\Gamma^0 \cup \Gamma^D} \{r^D(\llbracket u_h \rrbracket)\} \cdot \llbracket v \rrbracket + \\
&\int_{\Gamma^0 \cup \Gamma^D} \{l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket)\} \cdot \llbracket v \rrbracket - \int_{\Gamma^0 \cup \Gamma^D} \{r(g \chi_{\Gamma^D} \hat{\mathbf{n}})\} \cdot \llbracket v \rrbracket + \\
&\int_{\Gamma^0 \cup \Gamma^D} \llbracket \nabla_h u_h \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket + \int_{\Gamma^0 \cup \Gamma^D} \llbracket r^D(\llbracket u_h \rrbracket) \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket + \\
&\int_{\Gamma^0 \cup \Gamma^D} \llbracket l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket - \int_{\Gamma^0 \cup \Gamma^D} \llbracket r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket - \\
&\int_{\Gamma^0 \cup \Gamma^D} \mu \llbracket u_h \rrbracket \cdot \llbracket v \rrbracket + \int_{\Gamma^D} \mu g v \\
&= \int_{\Gamma^0 \cup \Gamma^D} \{\nabla_h u_h\} \cdot \llbracket v \rrbracket - \int_{\Omega} r^D(\llbracket u_h \rrbracket) \cdot r^D(\llbracket v \rrbracket) - \\
&\int_{\Omega} l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) \cdot r^D(\llbracket v \rrbracket) + \int_{\Omega} r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \cdot r(\llbracket v \rrbracket) + \\
&\int_{\Gamma^0} \llbracket \nabla_h u_h \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket - \int_{\Omega} r^D(\llbracket u_h \rrbracket) \cdot l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) - \\
&\int_{\Omega} l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) \cdot l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) + \int_{\Omega} r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \cdot l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) - \\
&\int_{\Gamma^0 \cup \Gamma^D} \mu \llbracket u_h \rrbracket \cdot \llbracket v \rrbracket + \int_{\Gamma^D} \mu g v \\
&= \int_{\Gamma^0 \cup \Gamma^D} \{\nabla_h u_h\} \cdot \llbracket v \rrbracket + \int_{\Gamma^0} \llbracket \nabla_h u_h \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket - \\
&\int_{\Gamma^0 \cup \Gamma^D} \mu \llbracket u_h \rrbracket \cdot \llbracket v \rrbracket + \int_{\Gamma^D} \mu g v - \\
&\int_{\Omega} [r^D(\llbracket u_h \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket)] \cdot [r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket)] + \\
&\int_{\Omega} r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \cdot [r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket)]
\end{aligned}$$

In the second equality we have used the definitions (124) and (125), along with $\boldsymbol{\beta} = \mathbf{0}$ on Γ^D , and in the third equality we have simply rearranged terms. It will be worthwhile to consider the very last term and rewrite it as

$$\begin{aligned}
\int_{\Omega} r(g \chi_{\Gamma^D} \hat{\mathbf{n}}) \cdot [r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket)] &= - \int_{\Gamma} g \chi_{\Gamma^D} \hat{\mathbf{n}} \cdot \{r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket)\} \\
&= - \int_{\Gamma^D} g [r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket)] \cdot \hat{\mathbf{n}}
\end{aligned}$$

Of course, we have computed only the integral over $\Gamma^D \cup \Gamma^0$. The remainder of the integral over Γ^R can be written as

$$\int_{\Gamma^R} \{\hat{\boldsymbol{\sigma}}\} \cdot \llbracket v \rrbracket = \int_{\Gamma^R} \frac{v}{b} (k - a u_h)$$

We also write expressions for some of the other terms in the bilinear form:

$$\int_{\Gamma} \llbracket \hat{u} - u_h \rrbracket \cdot \{\nabla_h v\} = - \int_{\Gamma^D \cup \Gamma^0} \llbracket u_h \rrbracket \cdot \{\nabla_h v\} + \int_{\Gamma^D} g \nabla_h v \cdot \hat{\mathbf{n}}$$

and,

$$\int_{\Gamma^0} \{\hat{u} - u_h\} \llbracket \nabla_h v \rrbracket = - \int_{\Gamma^0} \llbracket \nabla_h v \rrbracket \boldsymbol{\beta} \cdot \llbracket u_h \rrbracket$$

And we have that $\llbracket \hat{\boldsymbol{\sigma}} \rrbracket \equiv 0$ on Γ^0 .

Putting this all together, we can write out an expression for the bilinear form defined in equation (122):

$$\begin{aligned} B_h(u_h, v) &= \int_{\Omega} \nabla_h u_h \cdot \nabla_h v - \int_{\Gamma^0 \cup \Gamma^D} \left[\llbracket u_h \rrbracket \cdot \{\nabla_h v\} + \llbracket v \rrbracket \cdot \{\nabla_h u_h\} \right] - \\ &\int_{\Gamma^0} \left[\llbracket \nabla_h u_h \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket + \llbracket \nabla_h v \rrbracket \boldsymbol{\beta} \cdot \llbracket u_h \rrbracket \right] + \int_{\Gamma^0 \cup \Gamma^D} \mu \llbracket u_h \rrbracket \cdot \llbracket v \rrbracket + \\ &\int_{\Omega} \left[r^D(\llbracket u_h \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) \right] \cdot \left[r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) \right] + \int_{\Gamma^R} \frac{a}{b} u_h v + \\ &\int_{\Gamma^D} g \left[r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) + \nabla_h v \right] \cdot \hat{\mathbf{n}} - \int_{\Gamma^R} \frac{k}{b} v - \int_{\Gamma^D} \mu g v \end{aligned} \quad (132)$$

We now collect all the terms that depend on u_h and define a new operator $\mathcal{A}_h(u_h, v)$:

$$\begin{aligned} \mathcal{A}_h(u_h, v) &= \int_{\Omega} \nabla_h u_h \cdot \nabla_h v - \int_{\Gamma^0 \cup \Gamma^D} \left[\llbracket u_h \rrbracket \cdot \{\nabla_h v\} + \llbracket v \rrbracket \cdot \{\nabla_h u_h\} \right] - \\ &\int_{\Gamma^0} \left[\llbracket \nabla_h u_h \rrbracket \boldsymbol{\beta} \cdot \llbracket v \rrbracket + \llbracket \nabla_h v \rrbracket \boldsymbol{\beta} \cdot \llbracket u_h \rrbracket \right] + \int_{\Gamma^0 \cup \Gamma^D} \mu \llbracket u_h \rrbracket \cdot \llbracket v \rrbracket + \\ &\int_{\Omega} \left[r^D(\llbracket u_h \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket u_h \rrbracket) \right] \cdot \left[r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) \right] + \int_{\Gamma^R} \frac{a}{b} u_h v \end{aligned} \quad (133)$$

and we define a u_h -independent operator $f_h(v)$:

$$f_h(v) = \int_{\Omega} f v - \int_{\Gamma^D} g \left[r^D(\llbracket v \rrbracket) + l(\boldsymbol{\beta} \cdot \llbracket v \rrbracket) + \nabla_h v \right] \cdot \hat{\mathbf{n}} + \int_{\Gamma^R} \frac{k}{b} v + \int_{\Gamma^D} \mu g v \quad (134)$$

And now the solution of our problem from (127) takes the form

$$\mathcal{A}_h(u_h, v) = f_h(v) \quad \forall v \in V_h \quad (135)$$

This form inherits one of the classical properties of the LDG method: the operator \mathcal{A}_h is symmetric, which allows for an easier linear-system solve.

Appendix D Elliptic Outflow Boundary Conditions

Given a domain $\Omega \in \mathbb{R}^d$ bounded by the set $\partial\Omega = \Gamma^D \cup \Gamma^R$, we wish to solve the Helmholtz boundary-value problem for the function $u: \Omega \rightarrow \mathbb{C}^3$

$$\left\{ \begin{array}{ll} \Delta u + \omega^2 u = f & x \in \Omega \\ u = g & x \in \Gamma^D \\ \pm i\omega \sqrt{\mu \varepsilon(x)} u + \frac{\partial u}{\partial n} = 0 & x \in \Gamma^R \end{array} \right. \quad (136)$$

We assume that Γ^D and Γ^R are mutually nonoverlapping sets which cover $\partial\Omega$. The quantities $g(x)$ and $\varepsilon(x)$ are real-valued functions defined on their appropriate domains, and ω and μ are positive constants. f is real-valued and defined on all Ω . $\frac{\partial u}{\partial n}$ is the directional derivative in the outward-pointing direction from the domain Ω . We denote $\hat{\mathbf{n}}$ as the outward-pointing normal vector. Thus we can write $\frac{\partial u}{\partial n} = \nabla u \cdot \hat{\mathbf{n}}$.

This is a reduction of Maxwell's equations while ignoring Gauss's Law. I.e. the function f should actually comprise of a $\nabla(\nabla \cdot u)$ term, but we ignore it here. In simplifications of this problem, that term is 0 anyway.

There is a sign variation in the Robin-type boundary condition. And we explain it here: We wish to simulate an outgoing wave at the boundary Γ^R . The eponymous boundary condition for this, the Sommerfeld radiation condition, is

$$\left(\frac{\partial}{\partial n} \pm i k \right) u = 0$$

where $k = \frac{\omega}{c(x)}$ is the wavenumber, and n is chosen to represent a direction which is 'outgoing'; for our problem, this is the outward-pointing normal vector. This condition stems from assuming u has the form of a plane wave $\exp(i \omega t \pm \mathbf{k} \cdot \mathbf{x})$ which is Fourier-transformed, where $\mathbf{k} = k \hat{\mathbf{k}}$ for some unit vector $\hat{\mathbf{k}}$. Indeed, if we consider a real-valued plane wave $u(t) = \cos(\omega t - \mathbf{k} \cdot \mathbf{x})$, then the frequency-domain version of u has contributions at $\pm \omega$ with the amplitudes $\frac{1}{2} \exp(\mp i \mathbf{k} \cdot \mathbf{x})$. We wish this to be a solution at a boundary. Thus, we shall impose a condition that admits this solution while banning incoming solutions in other directions or with different wavenumbers. A mathematical way of doing this is to impose that $\nabla u \cdot \hat{\mathbf{k}} \pm i k u = \left(\frac{\partial}{\partial k} \pm i k \right) u = 0$. The reason for the unspecified sign in the boundary condition now becomes clear: we have two sets of wavenumbers corresponding to each real-valued wave.

Let us now assume that we wish to use the LDG method for the real- and complex-valued parts of u separately. We denote real- and complex-parts of a variable by superscripts 'R' and 'C', respectively. Let us now suppose that my function $u(t) = \cos(\omega t - \mathbf{k} \cdot \mathbf{x})$. Then $\tilde{u}(\omega) = \frac{1}{2} \delta(\omega) \exp(-i \mathbf{k} \cdot \mathbf{x}) + \frac{1}{2} \delta(\omega) \exp(i \mathbf{k} \cdot \mathbf{x})$. Thus, in order to specify outflow at any particular interface with normal vector n , we've got to specify two boundary conditions for each of my

wavenumbers: $\left(\frac{\partial}{\partial n} \pm i k\right)\tilde{u} = 0$ for wavevectors $\mp \mathbf{k}$. In other words, we have

$$\left(\frac{\partial}{\partial n} + i k\right)(\tilde{u}^R + i \tilde{u}^I) = 0$$

$$\left(\frac{\partial}{\partial n} - i k\right)(\tilde{u}^R + i \tilde{u}^I) = 0$$

Now it turns out that since we're assuming that u is real-valued, the properties of the Fourier transform ensure that if one condition is satisfied, they both will be. (These properties are $\tilde{u}^R(\omega) = \tilde{u}^R(-\omega)$ and $\tilde{u}^I(\omega) = -\tilde{u}^I(-\omega)$.) Therefore we shall only concern ourselves with the first equations. Separating real and complex parts yields

$$\frac{\partial}{\partial n}\tilde{u}^R - k\tilde{u}^I = 0$$

$$\frac{\partial}{\partial n}\tilde{u}^I + k\tilde{u}^R = 0$$

For the LDG formulation, the two unknowns are $(v, w) := (\tilde{u}^R, \tilde{u}^I)$ and $k = \omega \sqrt{\mu \varepsilon(x)}$, and problem (136) can be restated as

$$\left\{ \begin{array}{ll} \Delta \begin{pmatrix} v \\ w \end{pmatrix} + \omega^2 \begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} f^R \\ f^I \end{pmatrix} & x \in \Omega \\ \begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} g^R \\ g^I \end{pmatrix} & x \in \Gamma^D \\ \begin{pmatrix} \frac{\partial}{\partial n} v - k w = 0 \\ \frac{\partial}{\partial n} w + k v = 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} & x \in \Gamma^R \end{array} \right. \quad (137)$$

For the LDG Robin-formulation, the flux used to implement the Robin boundary conditions is used on the auxilliary variable. After doing a lot of algebra (see appendix C), one can write primal-variable Galerkin-statement for the LDG-discretized version of equation (137) as

$$\mathcal{A}_h((v_h, w_h)^T, (r, s)^T) = f_h((r, s)^T) \quad \forall (r, s)^T \in (V_h)^2$$

for some suitable discrete space V_h . We denote \mathcal{A}_h^S as the portion of \mathcal{A}_h which is independent of the form of the Robin boundary conditions; this portion of \mathcal{A}_h is symmetric (or Hermitian). We can then rewrite the Galerkin statement above as

$$\mathcal{A}_h^S((v_h, w_h)^T, (r, s)^T) - \int_{\Gamma^R} k w_h r + \int_{\Gamma^R} k v_h s = f_h((r, s)^T) \quad \forall (r, s)^T \in (V_h)^2$$

And one can see that upon interchange of $(v_h, w_h)^T$ and $(r, s)^T$ that the last two terms on the left-hand side are *anti*-symmetric, and not symmetric, as the operator \mathcal{A}_h^S is.

These boundary conditions are untested in 2D and 3D cases and should be verified to ensure that they perform the tasks for which they were designed. In particular, it's unclear that these conditions will actually work if the outgoing wave vector is not exactly normal to the boundary.

Appendix E The RBM Algorithm

In this appendix we shall give in relatively gory detail the algorithm for implementation of the RBM algorithm for a noncoercive primal-dual system. We assume a system of the form

$$\begin{aligned} a(u, w; \mu) &= f(w; \mu) \quad \forall w \in X \\ a(w, v; \mu) &= s(w; \mu) \quad \forall w \in X \end{aligned} \tag{138}$$

where we also suppose the usual assumptions of affinity of the bilinear form a as well as both the right-hand sides f and s . It is assumed that s is our output of interest and we define the output estimator \bar{s} as in equation (91). Throughout, superscripts ‘pr’ refer to primal-system quantities, and ‘du’ denotes the respective dual quantities. The error estimators are as defined in theorem 13. To make the exposition clearer, the RB space for the primal solution will be X^N , and that for the dual solution will be Y^M . Recall that $N = \dim(X^N)$ and $M = \dim(Y^M)$ and that the residual-norm matrix \mathbb{G}^{pr} defined in section 2.3.1 is of dimension $Q_N^{\text{pr}} = Q_f + N Q_a$, with Q_f and Q_a being the affinity dimensions of the operators a and f . Similarly, $Q_M^{\text{du}} = Q_s + M Q_a$. a^* is the formal adjoint of a .

We assume the following quantities are given:

- ε_{tol} , the desired level of accuracy for the output $s(u; \mu)$ in the X -norm
- N_{max} and M_{max} , the maximum number of reduced-basis functions for the primal and dual problems
- $\varepsilon_\beta \in (0, 1)$, the efficiency parameter for the inf-sup lower-bound calculation
- Integers $|\mathcal{V}_j|$, the number of vertices for polytopes in the inf-sup lower-bound calculation
- \mathcal{D}_K , an appropriate discretization of the parameter space \mathcal{D} , with cardinality $|\mathcal{D}_K| = K$

Offline Procedure	Operation Count
0.) Compute truth approximation matrices and structure affine dependency. Set $\varepsilon^{\text{pr}} = \varepsilon^{\text{du}} = \sqrt{\varepsilon_{\text{tol}}}$.	$O(?)$
1.) Calculate the inf-sup lower bound for both problems. Set $j = 1$ while 1, Choose $\bar{\mu}_j^{\text{pr/du}} \notin \bigcup_{i=1}^{j-1} \mathcal{P}_i$ via some method Compute $\beta^{\text{pr/du}}(\bar{\mu}_j)$ via equation (116) Compute the vertices $\mu_k^{\text{pr/du}}$ via equations (74) and (118) Set $\mathcal{P}_j^{\text{pr/du}}$ as the polytope defined by the vertices $\mu_k^{\text{pr/du}}$, set $j \leftarrow j + 1$. If the covering condition (73) is satisfied, break end Compute $\tilde{\beta}^{\text{pr}}(\mu)$ and $\tilde{\beta}^{\text{du}}(\mu)$ for all $\mu \in \mathcal{D}_K$ using equation (66)	$O(\mathcal{N}_t^3)$
2.) Construct global inner-product (i.e. mass) matrix for the truth approximation, invert it.	$O(\mathcal{N}_t^3)$
3.) Pick $\mu_1^{\text{pr}}, \mu_1^{\text{du}}$ via some method, compute $u_1 = u(\mu_1^{\text{pr}}), v_1 = v(\mu_1^{\text{du}})$, set $\zeta_1^{\text{pr}} = \frac{u_1^{\text{pr}}}{\ u_1^{\text{pr}}\ _X}, \zeta_1^{\text{du}} = \frac{v_1^{\text{du}}}{\ v_1^{\text{du}}\ _X}, X^1 = \text{span}(\zeta_1^{\text{pr}}), Y = \text{span}(\zeta_1^{\text{du}})$ and compute the 1×1 matrices $(a_q^1) = a_q(\zeta_1^{\text{pr}}, \zeta_1^{\text{pr}})$ and $(a_q^{*,1}) = a_q^*(\zeta_1^{\text{du}}, \zeta_1^{\text{du}})$, and 1×1 vectors $(f_q^1) = f_q(\zeta_1^{\text{pr}})$ and $(s_q^1) = s_q(\zeta_1^{\text{du}})$.	$O(\mathcal{N}_t^3 + (1 + Q_1^{\text{pr}})\mathcal{N}_t + (1 + Q_1^{\text{du}})\mathcal{N}_t)$
4.) Construct the $Q_1^{\text{pr}} \times Q_1^{\text{pr}}$ residual norm matrix \mathbb{G}_1^{pr} and the $Q_1^{\text{du}} \times Q_1^{\text{du}}$ residual norm matrix \mathbb{G}_1^{du} .	$O(\mathcal{N}_t Q_1^{\text{pr}} + \mathcal{N}_t Q_1^{\text{du}} + \mathcal{N}_t^2)$
5.) for $m = 1$ to $(N_{\text{max}} - 1)$ for $k = 1$ to K Pick $\tilde{\mu}$ as the k 'th gridpoint in \mathcal{D}_K Form matrix $(a^m)_{ij} = \sum_{q=1}^{Q_a} \Theta_q^a(\tilde{\mu}) (a_q^m)_{ij}$ and vector $(f^m)_j = \sum_{q=1}^{Q_f} \Theta_q^f(\tilde{\mu}) (f_q^m)_j$ for $i, j = 1, \dots, m$ Solve $a_m(u; w; \tilde{\mu}) = f_m(w; \tilde{\mu}) \forall w \in X^m$ Compute (truth) residual norm for the solution u (\mathbb{G}_m^{pr}) Compute and store the error estimator $\Delta_m^{\text{pr}}(k)$ end Determine the k which maximizes $\Delta_m^{\text{pr}}(k)$, call the associated μ as μ_{m+1}^{pr} . If $\sqrt{\gamma(\mu_{m+1}^{\text{pr}})} \max(\Delta_m^{\text{pr}}) \leq \varepsilon^{\text{pr}}$: set $N = m$, break Solve $a(u_{m+1}, v; \mu_{m+1}^{\text{pr}}) = f(v; \mu_{m+1}^{\text{pr}}) \forall v \in X$, set $\zeta_{m+1}^{\text{pr}} = \frac{u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j^{\text{pr}})_X \zeta_j^{\text{pr}}}{\ u_{m+1} - \sum_{j=1}^m (u_{m+1}, \zeta_j^{\text{pr}})_X \zeta_j^{\text{pr}}\ }, X^{m+1} = \text{span}(\{\zeta_i^{\text{pr}}\}_{i=1}^{m+1})$ If previous step was unstable: pick different μ_{m+1}^{pr} Compute the $(m+1) \times (m+1)$ matrices $(a_q^{*,m+1})_{ij} = a_q(\zeta_i^{\text{pr}}, \zeta_j^{\text{pr}})$ and $(m+1) \times 1$ vectors $(f_q^m)_j = f_q(\zeta_j^{\text{pr}})$ Construct the $Q_{m+1}^{\text{pr}} \times Q_{m+1}^{\text{pr}}$ residual norm matrix $\mathbb{G}_{m+1}^{\text{pr}}$ end	$O(Q_a m^2 + Q_f m) O(m^3) O(Q_m^{\text{pr}} + (Q_m^{\text{pr}})^2) O(1) O(K) O(\mathcal{N}_t^3 + m \mathcal{N}_t) O((m^2 + m)\mathcal{N}_t) O(Q_{m+1}^{\text{pr}} \mathcal{N}_t + \mathcal{N}_t^2)$
6.) for $m = 1$ to $(M_{\text{max}} - 1)$ for $k = 1$ to K Pick $\tilde{\mu}$ as the k 'th gridpoint in \mathcal{D}_K Form matrix $(a^{*,m})_{ij} = \sum_{q=1}^{Q_a} \Theta_q^a(\tilde{\mu}) (a_q^{*,m})_{ij}$ and vector $(s^m)_j = \sum_{q=1}^{Q_s} \Theta_q^s(\tilde{\mu}) (s_q^m)_j$ for $i, j = 1, \dots, m$ Solve $a^{*,m}(v; w; \tilde{\mu}) = s_m(w; \tilde{\mu}) \forall w \in Y^m$ Compute (truth) residual norm for the solution v (\mathbb{G}_m^{du}) Compute and store the error estimator $\Delta_m^{\text{du}}(k)$ end	$O(Q_a m^2 + Q_f m) O(m^3) O(Q_m^{\text{du}} + (Q_m^{\text{du}})^2) O(1)$

Bibliography

- [1] A. Kabakian, V. Shankar, and W. Hall, “Unstructured grid-based discontinuous galerkin method for broadband electromagnetic simulations,” *Journal of Scientific Computing*, vol. 20, no. 3, pp. 405–431, 2004.
- [2] W. Hall and A. Kabakian, “A sequence of absorbing boundary conditions for maxwell’s equations,” *Journal of Computational Physics*, vol. 194, pp. 140–144, 2004.
- [3] A. Taflove and S. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House, third ed., 2005.
- [4] C. Chauviere, J. Hesthaven, and L. Lurati, “Computational modeling of uncertainty in time-domain electromagnetics,” *SIAM Journal of Scientific Computing*, vol. 28, no. 2, pp. 751–775, 2006.
- [5] C. Chauviere, J. Hesthaven, and L. Wilcox, “Efficient computation of rcs from scatterers of uncertain shapes,” *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 5, pp. 1437–1448, 2007.
- [6] A. Patera and G. Rozza, *Reduced Basis Approximation and A Posteriori Error Estimation for Parameterized Partial Differential Equations*. MIT Pappalardo Graduate Monographs in Mechanical Engineering, 2007.
- [7] K. Ito and S. Ravindran, “A reduced-order method for simulation and control of fluid flows,” *Journal of Computational Physics*, vol. 143, pp. 403–425, 1998.
- [8] N. Pierce and M. Giles, “Adjoint recovery of superconvergent functionals from pde approximations,” *SIAM Review*, vol. 42, no. 2, pp. 247–264, 2000.
- [9] Y. Maday, A. Patera, and D. Rovas, *A Blackbox Reduced-Basis Output Bound Method for Noncoercive Linear Problems*. Elsevier B.V., 2002.
- [10] N. N. Cuong, *Reduced-Basis Approximations and A Posteriori Error Bounds for Nonaffine and Nonlinear Partial Differential Equations: Applications to Inverse Analysis*. PhD thesis, The Massachusetts Institute of Technology, 2005.
- [11] D. V. Rovas, *Reduced-Basis Output Bound Methods for Parameterized Partial Differential Equations*. PhD thesis, The Massachusetts Institute of Technology, 2002.
- [12] K. Veroy, C. Prud’homme, D. Rovas, and A. Patera, “A posteriori error bounds for reduced-basis approximation of parameterized noncoercive and nonlinear elliptic partial differential equations,” *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, 2003.
- [13] L. Machiels, Y. Maday, I. Oliveira, A. Patera, and D. Rovas, “Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems,” *C. R. Acad. Sci. Paris, S erie I*, vol. 331, no. 2, pp. 153–158, 2000.
- [14] J. Hesthaven Personal Communicae.
- [15] P. Houston, I. Perugia, A. Schneebeli, and D. Schoetzau, “Mixed discontinuous galerkin approximation of the maxwell operator: The indefinite case,” *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 39, no. 4, pp. 727–753, 2005.
- [16] J. Hesthaven and T. Warburton, “High-order nodal discontinuous galerkin methods for the maxwell eigenvalue problem,” *Royal Society A: Mathematical, Physical, and Engineering Sciences*, vol. 362, pp. 493–524, 2004.
- [17] B. Haasdonk and M. Ohlberger, “Reduced basis method for finite volume approximations of parametrized evolution equations,” *Preprint, submitted to Mod ellisation Math matique et Analyse Num rique*, 2006.
- [18] A. T. Y. Kwang, *Reduced Basis Method for 2nd Order Wave Equation: Application to One-Dimensional Seismic Problem*. PhD thesis, The Massachusetts Institute of Technology, 2006.
- [19] A. Tan and A. Patera, “Reduced basis method for 2nd order wave equation: Application to one-dimensional seismic problem,” 2007.
- [20] B. Cockburn and C.-W. Shu, “The local discontinuous galerkin method for time-dependent convection-diffusion systems,” *SIAM Journal on Numerical Analysis*, vol. 35, no. 6, pp. 2440–2463, 1998.
- [21] D. Arnold, F. Brezzi, B. Cockburn, and L. Marini, “Unified analysis of discontinuous galerkin methods for elliptic problems,” *Journal on Numerical Analysis*, vol. 39, no. 5, pp. 1749 – 1779, 2002.
- [22] C. Prud’homme, D. Rovas, K. Veroy, L. Machiels, Y. Maday, A. Patera, and G. Turinici, “Reliable real-time solution of parameterized partial differential equations reduced-basis output bound methods,” *Transactions of the American Society of Mechanical Engineers*, vol. 124, pp. 70–80, 2002.

- [23] Y. Maday, “Reduced basis method for the rapid and reliable solution of partial differential equations.” 2006.
- [24] M. Barrault, Y. Maday, N. Nguyen, and A. Patera, “An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations,” *Comptes rendus Mathématique*, vol. 339, no. 9, pp. 667–672, 2004.
- [25] D. Hunyh, G. Rozza, S. Sen, and A. Patera, “A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants,” *To Appear*, 2006.
- [26] A. Lovgren, Y. Maday, and E. Ronquist, “The reduced basis element method for fluid flows,” 2006.
- [27] I. Oliveira and A. Patera, “Reduced-basis techniques for rapid reliable optimization of systems described by affinely parameterized coercive elliptic partial differential equations,” 2007.
- [28] S. Sen, K. Veroy, D. Huynh, S. Deparis, N. Nguyen, and A. Patera, “‘natural norm’ a posteriori error estimators for reduced basis approximations,” *Journal of Computational Physics*, vol. 217, pp. 37–62, 2006.