NTNU
Norwegian University of
Science and Technology

Empirical Interpolation with Application to Reduced Basis Approximations

Tor Øvstedal Aanonsen

Master of Science in Physics and Mathematics
Submission date: June 2009
Supervisor: Einar Rønquist, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

The empirical interpolation (EI) method was recently proposed in the context of the reduced basis method. The original motivation was to allow for an online/offline decomposition even for problems which cannot be described using linear combinations of parameter independent bilinear and linear forms.

Very recently, a deeper understanding and analysis of the EI method has been presented. This project will explore the properties of EI method, primarily by solving selected model problems. Some of the numerical tests will be performed on simple test problems with the aim to gain insight, while some may be applied to more challenging problems derived from a reduced basis setting.

Assignment given: 18. January 2009
Supervisor: Einar Rønquist, MATH

Preface

This thesis concludes my 5-year masters program at the Department of Mathematical Sciences at NTNU, Trondheim, Norway. The course code for this subject is TMA4910 and constitutes 30 ECTS credits.

My field of study is applied mathematics, and in particular, solving partial differential equations using numerical techniques. However, our program (Physics and Mathematics) also gives an introduction to the most common areas of the natural sciences such as mechanics, chemistry and electromagnetism.

This project has touched upon several of the aspects presented in my earlier courses, such as mathematical modeling, numerical analysis and scientific programming. It also combines classical problems in numerics, such as interpolation and problem reduction techniques, with modern approaches.

For the reader to fully appreciate the contents of this report, knowledge of numerical analysis and some finite element theory is an advantage. However, the basic principles and results should be understandable for any graduate level mathematics student.

All implementations are done in Matlab$^{\text{TM}}$, which is a great tool for testing small to medium scale problems, and also for visualizing results.

Finally, I would like to thank my supervisor Einar M. Rønquist at the department of mathematics, NTNU. His advice and ideas have been a great inspiration throughout my work with this thesis.

Tor Øvstedal Aanonsen
Trondheim
June 3, 2009

Summary

The empirical interpolation method is an interpolation scheme with problem dependent basis functions and interpolation nodes developed for parameter dependent functions. It was developed in connection with the reduced basis framework for fast evaluation of output from parametrized partial differential equations, but the procedure may be applicable to a variety of problems.

Here, the theoretical background and implementation of the method is presented. We give examples to verify exponential convergence for analytic problems, and we also investigate the case where the parametric dependence has limited regularity.

Applications such as fast quadrature for parameter dependent functions and rapid solution of nonlinear differential equations are presented.

Finally, empirical interpolation is combined with the reduced basis methodology to efficiently solve an optimization problem. This is modeled as a Laplace equation with mixed boundary conditions, and parameter dependent deformation of the computational domain.

An online/offline computational procedure, along with *a posteriori* error estimates and an efficient greedy sampling strategy is discussed and applied to solve the optimization problem very rapidly.

Contents

List of Figures

Chapter 1

Introduction

The reduced basis (RB) methodology is an efficient procedure for evaluating output functionals, such as e.g. average temperature or pressure, dependent on the solution of a parametrized partial differential equation (μ-PDE). Its use can be particularly advantageous in (i) the "many queries" context, i.e. we want to evaluate the output for a large number of parameter values, or (ii) the "real-time" context, i.e. the value of the parameter is not known until the solution is needed [1].

The specific kind of parameter dependence can vary. Some examples include viscosity [2], frequency [3], stochastic input [4] and geometry [5], as will be the case in this work. However, for all examples, we require that the solution of the PDE varies smoothly with a parameter $\boldsymbol{\mu}$ over some parameter space $\mathcal{D} \subset \mathbb{R}^d$.

If we have this smooth parametric dependence, we expect that a set of solutions sampled at well-chosen parameter values will contain much information about the solution for an arbitrary $\boldsymbol{\mu} \in \mathcal{D}$. We can thus define this set of solutions as a discrete approximation space X_N. Given a new parameter $\boldsymbol{\mu} \in D$ we can then, via a Galerkin procedure, find the best approximation to the corresponding solution as a linear combination of the elements of X_N. This is the principle behind the reduced basis method.

The ability to compute this solution quickly depends on our problem being affine in the parameter. To solve non-affine problems an affine approximation was developed. Based on many of the same principles, the "empirical interpolation" (EI) method was introduced in [6] and expanded on in [7, 8, 9, 10]. Later, the method was studied outside the RB framework [11], a work we will continue in this report.

Both the reduced basis and empirical interpolation methods are realizations of the online/offline computational concept. This we now explain in greater detail.

1.1 The online/offline concept

The decomposition of a problem into a, frequently computationally expensive, generic part, and a more rapid specific part is a common technique in numerics. A method which applies this concept is often called an *online/offline* procedure.

A simple example is the LU-decomposition. If we know that the solution is required for several values of the right-hand side, we need only do the full

Gaussian elimination for the first value. However, for all subsequent evaluations we are left with a simple, triangular system.

Assume that the LU-decomposition is applied to a system of $\mathcal{N}$ equations. The offline complexity is $\mathcal{O}(\mathcal{N}^3)$ from the Gaussian elimination. Online, only $\mathcal{O}(\mathcal{N}^2)$ operations are required. In this case we still have $\mathcal{N}$ dependence in the online stage. What if we were able to reduce the number of equations in the online stage to $N \ll \mathcal{N}$? Even if Gaussian elimination was used for the online calculations, the complexity would only be $\mathcal{O}(N^3)$ and thus independent of $\mathcal{N}$.

The reduced basis method is another realizations of the online/offline concept, where the goal is to achieve $\mathcal{N}$ independence in the online stage. We want to maximize the amount of work done offline. Then, for a new parameter value, we do an inexpensive online evaluation to obtain the solution. Compared to LU-decomposition, the reduced basis method is obviously more complicated. Different values of the parameter will alter not only the right-hand side, but the entire system of equations.

Empirical interpolation was, as mentioned, introduced to achieve an $\mathcal{N}$ independent RB online stage. However, the EI method is in itself based on an online/offline procedure where the interpolation points and basis functions are computed offline. Online, we quickly compute parameter dependent basis coefficients and assemble the interpolant.

What allows us to achieve the online/offline decomposition is that if we have smooth parameter dependence in our problem, a linear combination of precomputed solutions will give an accurate approximation for an arbitrary new parameter value. Of course, the precomputed, or "snapshot", solutions must be sampled at "clever" points in the parameter space $\mathcal{D}$. For empirical interpolation and the reduced basis method, this will be implemented through a greedy sampling strategy. However, this approach is only one way of realizing the online/offline concept.

1.2 Brief outline of the report

The contents of this report can be split into two major parts. The first part, Chapters 3,4 and 5, is devoted to empirical interpolation, and the application of this method. The second part, Chapters 6,7 and 8 deals with the solution of parametrized partial differential equations. However, first we introduce some preliminary mathematical definitions and results in Chapter 2.

Chapter 3 contains the derivation and theoretical discussion of the empirical interpolation method. We also present a number of numerical examples both to verify error estimates and investigate whether additional hypotheses can be made.

A simple application of empirical interpolation in connection with numerical quadrature is presented in Chapter 4. We compare this to standard quadrature procedures for both one-dimensional and multi-dimensional problems.

In Chapter 5 the empirical interpolation method is used in combination with a Runge-Kutta metod, which to our knowledge has never before been atempted, to efficiently solve nonlinear, parameter dependent differential equations.

An optimization problem modeled as a parametrized partial differential equation is introduced in Chapter 6. For this problem, empirical interpolation alone is not enough to achieve an efficient online/offline decomposition.

This problem is first solved using a conventional approach in Chapter 7, the spectral approximation method. However, if the solution is required for a large number of parameters, this method will be to slow.

Finally, in Chapter 8 we present the reduced basis methodology in detail. Error estimates and efficient implementation is discussed. Also, we solve the optimization problem and demonstrate significant improvement in performance over the conventional method.

1.3 Notation

We end the introduction with a few remarks regarding notation. Vectors will usually be denoted by an underline, e.g. $\underline{b} \in \mathbb{R}^n$, however we do not use underline for vector elements. That is, the i'th element of $\underline{b}$ will be denoted by b_i.

The above convention has an important exception. In the case of multi-dimensional parameters we use boldface, usually $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^d$ as is common in the litterature. For scalar parameters we use μ.

Matrices will be denoted by capital letters, but without underline. However, we will frequently have subscripted matrices e.g. B_M. Matrix element i, j of B_M will be denoted by $(B_M)_{ij}$.

Explicit parametric dependence of a function $f(x)$ is referred to as $f(x; \boldsymbol{\mu})$. Similarly for matrices and vectors we use e.g. $A = A(\boldsymbol{\mu})$. The explicit dependence will sometimes be omitted where convenient, but when this is done the parametric dependence is obvious.

Chapter 2

Preliminaries

Before the main topics are introduced we need to describe a few mathematical definitions and techniques. This includes derivation and basic properties of the weak formulation of a partial differential equation, Gauss-Lobatto Legendre quadrature and standard polynomial approximation and interpolation.

2.1 Sobolev and Hilbert spaces

Fundamental to the development of Galerkin based methods is the concept of function spaces. Although this report will not contain much Sobolev- and Hilbert-space theory, a few basic definitions and results are necessary. First, let $H^m(\Omega)$ denote the space of functions, defined on a domain Ω, for which all derivatives up to order m are square integrable, see (2.1). In this notation $L^2(\Omega) = H^0(\Omega)$.

$$\begin{aligned} L^2(\Omega) &= \{v| \int_\Omega v^2 \,\mathrm{d}\Omega < \infty\}, \\ H^1(\Omega) &= \{v| \int_\Omega (v^2 + |\nabla v|^2) \,\mathrm{d}\Omega < \infty\}, \end{aligned} \tag{2.1}$$

with the associated *inner products*

$$\begin{aligned} \forall v, w \in L^2(\Omega), \quad (v,w)_{L^2(\Omega)} &= \int_\Omega vw \,\mathrm{d}\Omega, \\ \forall v, w \in H^1(\Omega), \quad (v,w)_{H^1(\Omega)} &= \int_\Omega (vw + \nabla v \cdot \nabla w) \,\mathrm{d}\Omega. \end{aligned} \tag{2.2}$$

Now define the *norm* of an element $v \in H^m(\Omega)$ as $\|v\|_{H^m(\Omega)} = (v,v)^{1/2}_{H^m(\Omega)}$. This is often called the natural, or induced, norm. A Hilbert-space is by definition complete in its induced norm, a property which is important for theoretical analysis. However, we will not discuss this, and the interested reader is referred to [12, 13, 14]. We also define the *energy norm* as

$$|||v||| = (a(v,v))^{1/2}. \tag{2.3}$$

Here $a(\cdot,\cdot)$ is a symmetric positive definite bilinear form we introduce in the next section. This norm will be involved in several of our following arguments.

We are now ready to express the *weak form* of a partial differential equation.

2.2 Weak formulation

Consider a two-dimensional Poisson problem with mixed boundary conditions

$$\begin{aligned} -\nabla^2 u &= f \quad && \text{in } \Omega, \\ u &= 0 \quad && \text{on } \Gamma_D, \\ \frac{\partial u}{\partial n} &= g \quad && \text{on } \Gamma_N, \end{aligned} \tag{2.4}$$

where $\Gamma_D \cup \Gamma_N = \partial\Omega$. This is the standard differential, or strong, formulation. The spectral approximation, along with other finite element based solution strategies, relies on the weak formulation. This can be derived from (2.4) by performing the following steps:

Define the function space $X^e = \{v | v \in H^1(\Omega),\ v = 0 \text{ on } \Gamma_D\}$. Multiply (2.4) by a test function $v \in X^e$ and integrate

$$\int_\Omega (-\nabla^2 u) v \, \mathrm{d}\Omega = \int_\Omega f v \, \mathrm{d}\Omega.$$

By applying the divergence theorem we get

$$-\int_{\partial\Omega} (\nabla u \cdot n) v \, \mathrm{d}s + \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f v \, \mathrm{d}\Omega.$$

Also, we get from the boundary conditions

$$\cancelto{0}{-\int_{\Gamma_D} (\nabla u \cdot n) v \, \mathrm{d}s} - \int_{\Gamma_N} g v \, \mathrm{d}s + \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f v \, \mathrm{d}\Omega,$$

where the first term disappears due to v being zero on the boundary (remember $v \in X^e$). The weak formulation now reads:

Find $u \in X^e$ such that

$$\underbrace{\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega}_{a(u,v)} = \underbrace{\int_\Omega f v \, \mathrm{d}\Omega + \int_{\Gamma_N} g v \, \mathrm{d}s}_{l(v)}, \quad \forall\, v \in X^e, \tag{2.5}$$

or the more abstract version, find $u \in X^e$ such that $a(u, v) = l(v)$ for all $v \in X^e$. Here $a(u, v)$ is the bilinear form used to define the energy norm in the previous section.

For different PDEs and boundary conditions (e.g. inhomogeneous Dirichlet or Robin), we may get different definitions of X^e, $a(w, v)$ and $l(v)$. The procedure is however the same.

It is now usual to ensure that (2.5) indeed admits a unique solution. If we assume $a(\cdot, \cdot)$ to be *coercive* i.e. $a(v, v) \geq \alpha \|v\|^2_{H^1(\Omega)}$ for all $v \in X^e$, and *bounded*, $|a(w, v)| \leq \beta \|w\|_{H^1(\Omega)} \|v\|_{H^1(\Omega)}$ for all $w, v \in X^e$, the existence of a unique solution follows by the *Lax-Milgram* theorem [15]. Here α and β (often called the coercivity and boundedness constants) are positive. We also need boundedness for the right-hand side, hence there are some requirements on the smoothness of f and g.

For the Poisson problem with Dirichlet boundary conditions (the following is valid also if only parts of the boundary is Dirichlet), establishing coercivity and boundedness, or *continuity*, is straight-forward and relies on the *Cauchy-Schwarz* and *Friedrich* inequalities. In the case of mixed boundary conditions, i.e. Dirichlet and Neumann, as is the case here, we also need the *trace* theorem. For details confer [15].

If our PDE is parameter dependent, it is common to introduce the *parametric weak form*. This is usually done by defining a parameter vector $\boldsymbol{\mu}$ in some parameter space $\mathcal{D}$ and denoting the weak problem as:

Given $\boldsymbol{\mu} \in \mathcal{D}$ find $u(\boldsymbol{\mu}) \in X^e$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = l(v; \boldsymbol{\mu}), \quad \forall v \in X^e.$$

The precise form of parameter dependence for $a(\cdot, \cdot; \boldsymbol{\mu})$ and $l(\cdot; \boldsymbol{\mu})$ will of course be determined by each specific problem.

2.3 Affine parameter dependence

The linear form $l(\cdot; \boldsymbol{\mu})$ and bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ are said to admit *affine parameter dependence* [1] if we can write

$$l(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_l} \Theta_l^q(\boldsymbol{\mu}) l^q(v), \quad \forall v \in X^e, \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.6}$$

and

$$a(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(w, v), \quad \forall w, v \in X^e, \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.7}$$

Where Θ_l^q and Θ_a^q are parameter dependent constants, $l^q(v)$ and $a^q(w, v)$ are parameter independent linear and bilinear forms, and Q_l and Q_a are (preferably small) integers.

A simple example of a bilinear form, with a single parameter, admitting this expansion is the Helmholtz operator which on weak form can be written

$$a(w, v; \mu) = \underbrace{\int_\Omega (\nabla w)^T \nabla v \, \mathrm{d}\Omega}_{a^1(w,v)} + \mu \underbrace{\int_\Omega wv \, \mathrm{d}\Omega}_{a^2(w,v)}.$$

We can express this as $\underbrace{\Theta_a^1(\mu)}_{=1} a^1(w, v) + \underbrace{\Theta_a^2(\mu)}_{=\mu} a^2(w, v)$.

This property allows us to compute all parameter independent terms separately as an offline procedure. When given a new parameter value, only a fast online stage is needed to compute the full linear or bilinear form.

This property will be essential in the development of a fast online/offline reduced basis procedure and is the main motivation for introducing empirical interpolation in that context.

2.4 Gauss-Lobatto Legendre quadrature

Assume we want to evaluate the integral

$$I = \int_{-1}^{1} g(x) \, \mathrm{d}x$$

for some given function $g(x)$. Using Gauss-Lobatto Legendre quadrature to approximate this expression gives us

$$I \approx \sum_{\alpha=0}^{\mathcal{N}} \rho_\alpha g(\xi_\alpha). \tag{2.8}$$

Here we have $2\mathcal{N}+2$ degrees of freedom, ρ_α and ξ_α, $\alpha = 0, \dots, \mathcal{N}$, where ρ_α is the quadrature weight corresponding to the function evaluation in node ξ_α. Notice that we are free to choose where we evaluate g. By choosing ξ_α to be the root of the $\mathcal{N}$th Legendre polynomial, $L_\mathcal{N}$, and

$$\rho_\alpha = \int_{-1}^{1} \prod_{\substack{\beta=0\\ \beta\neq\alpha}}^{\mathcal{N}} \frac{\xi - \xi_\beta}{\xi_\alpha - \xi_\beta}\,\mathrm{d}\xi,$$

it can be shown [16] that we are able to integrate exactly a polynomial of degree $2\mathcal{N}+1$ or less. However the "Lobatto" in Gauss-Lobatto Legendre fixes ξ_0 and $\xi_\mathcal{N}$ to the endpoints -1 and 1 respectively. This leaves us with $2\mathcal{N}$ degrees of freedom. If we now let $\xi_\alpha, \alpha = 1, \dots, \mathcal{N}-1$ be the roots of $L'_\mathcal{N}$ we are able to integrate exactly a polynomial of degree $2\mathcal{N}-1$ or less. If we define $\mathbb{P}_q(-1,1)$ to be the space of all polynomials of degree q or less on $(-1,1)$, we have more formally that for all $g \in \mathbb{P}_{2\mathcal{N}-1}(-1,1)$,

$$I = \int_{-1}^{1} g(x)\,\mathrm{d}x = \sum_{\alpha=0}^{\mathcal{N}} \rho_\alpha g(\xi_\alpha).$$

The approximation (2.8) is readily extended to the multi-dimensional case. For two dimensions let $g = g(\xi,\eta)$ in $\hat{\Omega} = (-1,1)^2$. The approximation rule becomes

$$\int_{\hat{\Omega}} g(\xi,\eta)\,\mathrm{d}\hat{\Omega} \approx \sum_{\alpha=0}^{\mathcal{N}} \sum_{\beta=0}^{\mathcal{N}} \rho_\alpha \rho_\beta\, g(\xi_\alpha, \xi_\beta), \tag{2.9}$$

where we use the same weights and points as for the one-dimensional case. A detailed description of the computation of the GLL points and weights can be found in e.g. [17]. The approximation is still exact when g is a polynomial of degree $2\mathcal{N}-1$ or less in each variable respectively.

2.5 Polynomial approximation and interpolation

The Weierstrass Approximation Theorem [18] states that any continuous function f on an interval $[a,b]$ can be approximated arbitrarily close by a polynomial. In addition, polynomials are easy both to store and evaluate on a computer. This has made polynomials a preferred option for approximation and interpolation. However, the theorem does not give any information regarding what type of polynomial should be used or how we should find it.

As we are dealing with computers, we need a finite set of polynomials. In principle, we could always use the set of mononomials $P_\mathcal{N}^{mono} = \{1, x, x^2, \dots, x^\mathcal{N}\}$ as a basis for the space of polynomials of degree less than or equal to $\mathcal{N}$. This choice will unfortunately often lead to poor numerical qualities due to the increasing degree of colinearity in the basis elements. There exists a number of different *orthogonal* polynomials based on different inner products.

In this section we will focus on the Legendre polynomials and Lagrange interpolation through the GLL points introduced in Section 2.4. Given a set of interpolation nodes $\{x_0, \dots, x_n\}$ the i'th Lagrange interpolation polynomial is defined as follows

$$\begin{aligned} &\ell_i^n(x) \in \mathbb{P}^n, \\ &\ell_i^n(x_j) = \delta_{ij}, \quad 0 \le i, j \le n. \end{aligned}$$

The example $\ell_3^5(x)$, with GLL interpolation nodes, is given in Figure 2.1. We see that the interpolation polynomial is 1 at the corresponding node (x_3), and 0 at all the others.

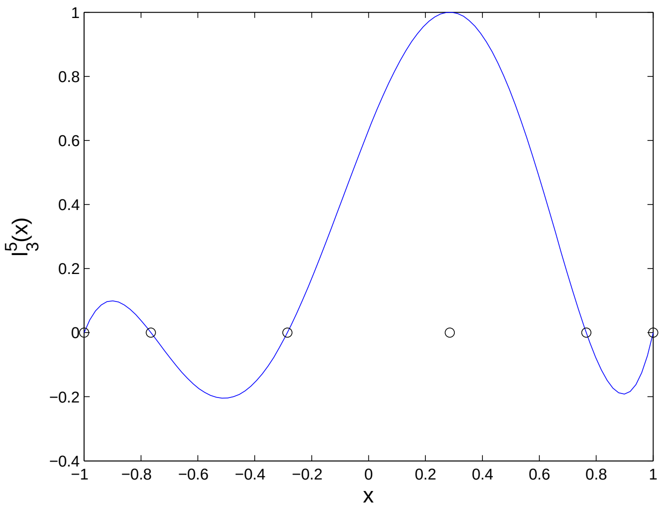


Figure 2.1: Lagrange interpolation polynomial $\ell_3^5(x)$. We see that $\ell_3^5(x_3) = 1$, but that it is zero at the other nodes.

Of particular importance is an error estimate for this type of interpolation when the underlying function f has a certain regularity. The following result was first derived in [19] and further developed in [20]. Let the function f be defined on the interval $[-1, 1]$. Denote the interpolant of f through $\mathcal{N} + 1$ GLL points in this interval by $\mathcal{I}_{\mathcal{N}}[f]$. We then have

Theorem 2.1 *For any $f \in H^\sigma(\Omega)$ the interpolation error satisfies*

$$\|f - \mathcal{I}_{\mathcal{N}}[f]\|_{H^k(\Omega)} \le C\mathcal{N}^{k-\sigma}\|f\|_{H^\sigma(\Omega}. \tag{2.10}$$

This implies that the convergence will depend on the regularity of f in addition to the number of interpolation nodes. We also get exponential convergence in the case of analytic f, i.e. $\sigma \to \infty$.

The interpolant in (2.10) can be expressed using Lagrange interpolation polynomials as

$$\mathcal{I}_{\mathcal{N}}[f](x) = \sum_{i=0}^{\mathcal{N}} f(\xi_i)\ell_i^{\mathcal{N}}(x).$$

Note the similarity to the affine parameter dependence in (2.6). We have here separated the function into a linear combination of f independent, and consequently also parameter independent, polynomials with coefficients determined by the function value at each interpolation node. The set of basis functions and interpolation nodes is thus completely detached from the problem. This makes them easy to compute and analyze. However, as we shall see in the next chapter, there are ways of determining superior, problem dependent basis functions and interpolation nodes. This procedure will be connected even closer to the affine parameter dependence in Section 2.3.

Part I

Chapter 3

Empirical Interpolation (EI)

The polynomial interpolant presented in the previous section was constructed using predefined basis functions and interpolation points. This allows easy implementation and predictable error behavior. However, with this approach the level of adaptability is low. Also, the only information extracted from the underlying interpolated function is the nodal values.

Now, imagine we are given a function $f(x; \boldsymbol{\mu})$, possibly with spatial singularities, which depend smoothly on a parameter $\boldsymbol{\mu}$ over some space $\mathcal{D}$. For each parameter value we have a different GLL interpolant based on the spatial appearance of the function. These have to be constructed separately and we do not exploit the parametric dependence.

The empirical interpolation method uses a different approach. Instead of predefined basis functions, we use f sampled at different points in $\mathcal{D}$. That is, our interpolation space becomes $X_M = \operatorname{span}\{f(x; \boldsymbol{\mu}_1^{EI}), \ldots, f(x; \boldsymbol{\mu}_M^{EI})\}$. As the parametric dependence is smooth, we expect a linear combination of the basis elements to give a good approximation to $f(x; \boldsymbol{\mu})$ for any value of the parameter.

To create the interpolant, we also require that $f(x; \boldsymbol{\mu})$ is matched in M spatial interpolation points. The details on how we build up this interpolation space will be the topic of the next sections, but first we motivate why this approach is advantageous.

3.1 Motivation

Consider the simple one-dimensional linear form

$$l(v; \mu) = \int_a^b f(x; \mu) v(x) \, \mathrm{d}x. \tag{3.1}$$

We here restrict ourselves to the case of a single scalar parameter $\mu \in \mathcal{D} \subset \mathbb{R}$, however in later sections we go back to multiple parameters. For a general function $f(x; \mu)$, l does not admit an affine expansion as in (2.6), and we are unable to exploit the efficient online/offline strategy. To evaluate this integral using a standard GLL quadrature formula with $\mathcal{N}$ points, requires $\mathcal{O}(\mathcal{N})$ operations for each new parameter because no offline computation is possible.

The EI approach is to approximate f as

$$f(x;\mu) \approx f_M(x;\mu) = \sum_{i=1}^{M} \varphi_i(\mu) q_i(x). \tag{3.2}$$

Here, $\varphi_i(\mu)$ are parameter dependent coefficients, and $q_i(x)$ are the parameter independent functions that, as will be demonstrated, span our EI space. It is important to note that the number of terms in the EI expansion have to be much smaller than the number of GLL points, or $M \ll \mathcal{N}$, if this approach is to have any significant effect on performance. However, as will be thoroughly demonstrated in the following sections, this will usually be the case.

The expansion (3.2) clearly admits affine parameter dependence. If we now replace f by f_M in (3.1) we get

$$\begin{aligned} l(v;\mu) &\approx \int_a^b f_M(x;\mu) v(x)\,\mathrm{d}x \\ &= \sum_{i=1}^{M} \varphi_i(\mu) \int_a^b q_i(x) v(v)\,\mathrm{d}x. \end{aligned} \tag{3.3}$$

Our functional l is now replaced by an affine approximation where, if we compare (3.3) with (2.6), $Q_l = M$, $\Theta_l^i = \varphi_i$ and $l^i(v) = \int_a^b q_i(x)v(x)\,\mathrm{d}x$.

We are now able to exploit the potential gains from the online/offline computational strategy. All the parameter independent integrals can be solved very accurately offline, in principle using any numerical integration method. The online stage only consists of finding the coefficients $\varphi_i(\mu)$ and adding all the terms. We will demonstrate that finding the coefficients can be done in $\mathcal{O}(M^2)$ operations, and the sum is only $\mathcal{O}(M)$. The EI method has thus achieved an online stage indeed independent of the offline complexity, $\mathcal{O}(\mathcal{N})$.

In the following section we explain in detail how to derive the basis functions $q_i, \ldots, q_M$. Also, the procedure for finding the coefficients $\varphi_i, \ldots, \varphi_M$ is given. This involves the set of interpolation points $T_M = \{t_1, \ldots, t_M\}$ which we remember were the GLL points in the case of polynomial interpolation. For EI this will no longer be true, and the interpolation points will, as the basis functions, be problem dependent.

3.2 Interpolation procedure

The EI algorithm is remarkably simple. It relies on a greedy selection process, both in the choice of basis functions and interpolation points. In a greedy algorithm, the next step is determined by some local optimality criterion based on the current situation. This may in some cases lead to a globally optimal algorithm, but in most cases, as for the EI method, this is not generally true. For a more thorough explanation of the concept of greedy algorithms, see e.g. [21].

We now present the empirical interpolation algorithm, given as Algorithm 3.1. This is followed by a detailed explanation of each step. We now denote the empirical interpolant of f through the M points of T_M by $\mathcal{I}_M[f]$. Note that T_M contains interpolation points in the spatial domain Ω. The algorithm will also

Algorithm 3.1 Empirical Interpolation

```
$\boldsymbol{\mu}_1^{EI} \leftarrow \arg\max_{\boldsymbol{\mu}\in\Xi_t} \|f(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}$
$\xi_1(\cdot) \leftarrow f(\cdot;\boldsymbol{\mu}_1^{EI})$
$t_1 \leftarrow \arg\max_{x\in\Omega} |\xi_1(x)|$
$q_1(\cdot) \leftarrow \xi_1(\cdot)/\xi_1(t_1)$
$B_1 \leftarrow q_1(t_1)$
for $m = 2, \ldots, M \leq M_{\max}$ do
    $\boldsymbol{\mu}_M^{EI} \leftarrow \arg\max_{\boldsymbol{\mu}\in\Xi_t} \|f(\cdot;\boldsymbol{\mu}) - \mathcal{I}_{M-1}[f](\cdot)\|_{L^\infty(\Omega)}$
    $\xi_m(\cdot) \leftarrow f(\cdot;\boldsymbol{\mu}_M^{EI})$
    $t_m \leftarrow \arg\max_{x\in\Omega} |\xi_m(x) - \mathcal{I}_{m-1}[\xi_m](x)|$
    $q_m(\cdot) \leftarrow \dfrac{\xi_m(\cdot) - \mathcal{I}_{m-1}[\xi_m](\cdot)}{\xi_m(t_m) - \mathcal{I}_{m-1}[\xi_m](t_m)}$
    $(B_m)_{ij} \leftarrow q_j(t_i), \quad 1 \leq i,j \leq m$
end for
```

produce a sequence of points in the parameter space $\boldsymbol{\mu}_1^{EI}, \ldots, \boldsymbol{\mu}_M^{EI}$. These will however only be stored implicitly in the EI basis functions.

In the initial stage of the EI algorithm we choose the parameter value which maximizes our function over $\mathcal{D}$. In general, $\mathcal{D}$ contains infinitely many points, hence for actual computation we use a subset of $\mathcal{D}$ of size $\mathcal{M}$. If the maximum number of EI terms allowed is $M_{\max}$, it is important that we choose $\mathcal{M} \gg M_{max}$. This subset will be denoted by $\Xi_t \subset \mathcal{D}$. Here, t is short for "training". Specifically, Ξ_t contains the parameters we use to prepare, or "train", the empirical interpolant to deal with any new parameter $\boldsymbol{\mu} \in \mathcal{D}$. The use is here analogous to the use of training samples in statistical classification [22, 23].

The identification of the initial basis function is thus performed in the two steps

$$\boldsymbol{\mu}_1^{EI} = \arg\max_{\boldsymbol{\mu}\in\Xi_t} \|f(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)},$$
$$\xi_1(\cdot) = f(\cdot,\boldsymbol{\mu}_1^{EI}).$$

This choice is in accordance with our greedy strategy if we define the "zero-interpolant" $\mathcal{I}_0[f] \equiv 0$, but the algorithm will work for any initial parameter choice. Here, the L^∞-norm is approximated by its discrete version. To do this we sample the function on a very fine grid and compute the maximum value for each of the parameters in our training set.

Our second greedy choice is in the determination of the first interpolation node

$$t_1 = \arg\max_{x\in\Omega} |\xi_1(x)|.$$

The choice of the initial interpolation node is arbitrary. This approach will however be consistent with the selection criterion for the interpolation nodes to follow. In addition, we perform a normalization step, $q_1(\cdot) = \xi_1(\cdot)/\xi_1(t_1)$, for better numerical stability. The initial (one element) interpolation matrix $(B_M)_{11}$ is now given as $(B_M)_{11} = q_1(t_1)$. At this stage it is possible to form the first interpolant as the only function colinear with q_1 that coincides with f at t_1.

$$\mathcal{I}_1[f](\cdot;\boldsymbol{\mu}) = f(t_1;\boldsymbol{\mu})q_1(\cdot).$$

This conludes the initial step.

We now repeat this procedure until a desired accuracy in the approximation is achieved, or we reach the predefined limit $M_{\max}$. How we may limit the required number of interpolation nodes through the use of a posteriori error estimates is discussed in Section 3.3.

Continuing the algorithm, we choose the next basis element to be the function which maximizes the difference between f and our current interpolant $\mathcal{I}_{M-1}[f]$. Again, this is a two step procedure

$$\boldsymbol{\mu}_M^{EI} = \arg\max_{\boldsymbol{\mu}\in\Xi_t} \|f(\cdot;\boldsymbol{\mu}) - \mathcal{I}_{M-1}[f](\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)},$$
$$\xi_M(\cdot) = f(\cdot,\boldsymbol{\mu}_M^{EI}).$$

That is, the next basis function is the function which the current interpolant is least suitable to approximate.

At this stage we need to compute the interpolant, $\mathcal{I}_{M-1}[f]$, for each parameter $\boldsymbol{\mu} \in \Xi_t$. This can be achieved through simple matrix operations in the following manner. Each new basis function q_i is represented by a high order polynomial interpolant based on the GLL points $x_0, \dots, x_P$. Here it is important that the polynomial interpolation error is small compared to the EI error. In this way, we can store all our current basis functions as column vectors in a large matrix $Q_{M-1} \in \mathbb{R}^{(P+1)\times M}$ where $P \gg M$ is the order of our polynomial interpolant.

The discrete representations of the empirical interpolant can now be expressed as $\underline{I}_{M-1} = Q_{M-1}\underline{y}$. Here $\underline{I}_{M-1,i} = \mathcal{I}_{M-1}[f](x_i)$, $i = 0, \dots, P$. The coefficient vector $\underline{y} \in \mathbb{R}^{M-1}$ is determined by solving the system

$$\sum_{j=1}^{M-1} (B_{M-1})_{ij} y_j = f(t_i), \quad i = 1, \dots, M-1,$$

or in matrix notation $B_{M-1}\underline{y} = \underline{f}_{M-1}$, for all $\boldsymbol{\mu} \in \Xi_t$. Again we approximate the continuous infinity norm by its discrete version. That is, we find

$$\max_{0\le i\le P} |f(x_i) - \mathcal{I}_{M-1}[f](x_i)|.$$

The next interpolation node is given as

$$t_M = \arg\max_{x\in\Omega} |\xi_M(x) - \mathcal{I}_{M-1}[\xi_M](x)|.$$

Here, the position where the largest error occurs is added to the set of interpolation nodes. As the interpolant by definition is equal to the underlying function at the interpolation nodes, the error at t_M is removed. We can now compute the new normalized basis function

$$q_M(\cdot) = \frac{\xi_M(\cdot) - \mathcal{I}_{M-1}[\xi_M](\cdot)}{\xi_M(t_M) - \mathcal{I}_{M-1}[\xi_M](t_M)}.$$

All that remains now is to store the values of B_M.

Finally the interpolant is given as

$$\mathcal{I}_M[f](x;\boldsymbol{\mu}) = \sum_{j=1}^{M} \varphi_j(\boldsymbol{\mu}) q_j(x),$$

where, for a new $\boldsymbol{\mu} \in \mathcal{D}$, we find the parameter dependent coefficients $\varphi_j(\boldsymbol{\mu})$ by solving

$$\sum_{i=1}^{M} (B_M)_{ij}\varphi_j(\boldsymbol{\mu}) = f(t_i, \boldsymbol{\mu}), \quad i = 1, \dots, M.$$

It can be shown that this procedure indeed produces a valid interpolation scheme given by the basis functions $q_1, \dots, q_M$ and interpolation nodes $t_1, \dots, t_M$ [11].

Algorithm 3.1 produces a sequence of hierarchical interpolation spaces $X_1 \subset X_2 \subset, \dots, X_M$ where $X_i = \text{span}\{\xi_1, \dots, \xi_i\} = \text{span}\{q_1, \dots, q_i\}$. This is a desirable property. Consider for instance a situation where we are forced to limit the number of basis elements to $\tilde{M} < M$. For the EI method, the optimal choice of elements is simply given as the $q_1, \dots, q_{\tilde{M}}$. For other non-hierarchical interpolation spaces, e.g. the polynomial interpolation from Section 2.5, this is not the case. With the polynomial approach, we would need to recompute the GLL points, and thus the Lagrangian polynomials, to determine the optimal subset.

As the focus of this work is parameter dependent problems, we have presented the EI algorithm within this framework. The use of EI is however not limited to this case. One could consider functions in a general space $\mathcal{U}$. In this case the next basis function would be given by

$$\xi_m = \arg\max_{u \in \mathcal{U}} \|u - \mathcal{I}_{m-1}[f]\|_{L^\infty(\Omega)}.$$

In principle $\mathcal{U}$ can be infinite dimensional, but for actual computation, we need to search a space of finite dimension. Using this formulation, the parameter dependent case can be expressed by $\mathcal{U} = \{f(\cdot;\boldsymbol{\mu}), \boldsymbol{\mu} \in \mathcal{D}\}$, or the finite dimensional version $\mathcal{U} = \{f(\cdot;\boldsymbol{\mu}), \boldsymbol{\mu} \in \Xi_t\}$.

Another example is to let $\mathcal{U}$ be the set of some predefined basis functions. We then use Algorithm 3.1 to compute interpolation points and a "good" ordering of the basis functions for a given region. In this case, we do not approximate any underlying function, but derive a generic interpolation scheme adapted to e.g. a special geometry. Several examples of this procedure, using a basis of mononomials and Legendre polynomials are given in [11].

3.3 Error analysis

If we now introduce the Lagrangian functions to construct the interpolation operator $\mathcal{I}_M$ we can write the interpolant of f as

$$\mathcal{I}_M[f(\cdot)] = \sum_{i=1}^{M} f(t_i)\ell_i^M(\cdot).$$

Here, $\ell_j^M(\cdot) = \sum_{j=1}^{M} q_j(\cdot)(B_M)_{ji}^{-1}$. The Lagrangian functions are a *nodal basis* for X_M, i.e. $\ell_i^M(x_j) = \delta_{ij}$.

In the following error analysis we make use of the Lebesgue constant defined as $\Lambda_M = \sup_{x\in\Omega} \sum_{j=1}^{M} |\ell_j^M(x)|$. For the EI method, an upper bound for Λ_M is given by $2^M - 1$ [11]. This is however a very pessimistic estimate, and in practice the observed behavior is usually far better.

A classical result in approximation theory, also known as Lebesgue's lemma [11], gives the following bound for the interpolation error

Lemma 3.1 *Assume X is a Banach space, and $X_M \subset X$, $\dim(X_M) = M$. For any $f \in X$, the interpolation error satisfies*

$$\|f - \mathcal{I}_M[f]\|_X \leq (1 + \Lambda_M) \inf_{g_M \in X_M} \|f - g_M\|_X. \tag{3.4}$$

Here the *projection error*, $\inf_{g_M\in X_M} \|f - g_M\|_X$, is the best possible approximation of f in the approximation space X_M.

For the EI method, Lemma 3.1 can be made considerably more precise if a few conditions are fulfilled. It can in fact be shown that the upper bound for the interpolation error from the greedy algorithm is given by

Theorem 3.2 *Assume $\mathcal{U} \subset X \subset L^\infty(\Omega)$ and the existence of a (possibly unknown) sequence of finite dimensional spaces*

$$\mathcal{Z}_1 \subset \mathcal{Z}_2 \subset \ldots \subset \mathcal{Z}_M \subset \mathrm{span}(\mathcal{U}), \quad \dim(\mathcal{Z}_M) = M,$$

such that there exists $c > 0$ and $\alpha > \log(4)$ with

$$\forall\, f \in \mathcal{U}, \inf_{g_M \in \mathcal{Z}_M} \|f - g_M\|_X \leq c\mathrm{e}^{-\alpha M}.$$

Then

$$\|f - \mathcal{I}_M[f]\|_{L^\infty\Omega} \leq c\mathrm{e}^{-(\alpha - \log(4))M}. \tag{3.5}$$

Proof: We refer to [11] for the proof of Theorem 3.2.

This theorem has some immediate implications. First of all, if there exists a finite dimensional space allowing for an exponential approximation, the EI method will achieve an exponential rate of convergence. Also, if the spaces $\mathcal{Z}_i$ are not predetermined, the greedy algorithm provides us with such a sequence through the EI space X_M.

A third and final advantage of the greedy algorithm is that it allows easy access to an *a posteriori* error estimate as we know that for any $\boldsymbol{\mu} \in \mathcal{D}$, the arg max definition of ξ_M in Algorithm 3.1 ensures that

$$\|f - \mathcal{I}_M[f]\|_{L^\infty(\Omega)} \leq \|\xi_{M+1} - \mathcal{I}[\xi_{M+1}]\|_{L^\infty(\Omega)}. \tag{3.6}$$

This implies that we always know the maximum error for the previous interpolation space. This can be used to end Algorithm 3.1 at a predefined tolerance level and thus avoid computing all M_{max} stages.

3.4 Numerical examples

As the EI method is a fairly recent development, and the amount of available material on the characteristic properties of the method is limited, we choose to perform a large number of numerical tests.

We will look at examples both where the parametric dependence is analytic, but also less favorable cases where we have limited regularity in the parameter.

To test the quality of the EI method we will compare it to polynomial interpolation through the GLL points. This is done by creating a test set Q_{test} of 100 random $\boldsymbol{\mu}$-values drawn uniformly from the parameter domain $\mathcal{D}$ for each example. We then compute the maximum L^2-error of the interpolant over all $\boldsymbol{\mu} \in Q_{test}$,

$$e_M = \max_{\boldsymbol{\mu} \in Q_{test}} \|f(\cdot;\boldsymbol{\mu}) - \mathcal{I}_M[f](\cdot;\boldsymbol{\mu})\|_{L^2(\Omega)}, \tag{3.7}$$

by GLL quadrature on a very fine grid.

3.4.1 Analytic in the parameter

Theorem 3.2 tells us that under reasonable assumptions the EI procedure achieves an exponential rate of convergence. In this section we present numerical results which verify this behaviour.

Example 1

Our first example is a classical function in interpolation history, and it is the function for which Runge demonstrated that interpolation based on equidistant points will diverge [24].

$$f(x;\mu) = \frac{1}{1+\mu x^2}, \qquad \text{in } \Omega. \tag{3.8}$$

Here $\Omega = (-1,1)$ and $\mathcal{D} = \{\mu, 1 \le \mu \le 25\}$. The results for the first 25

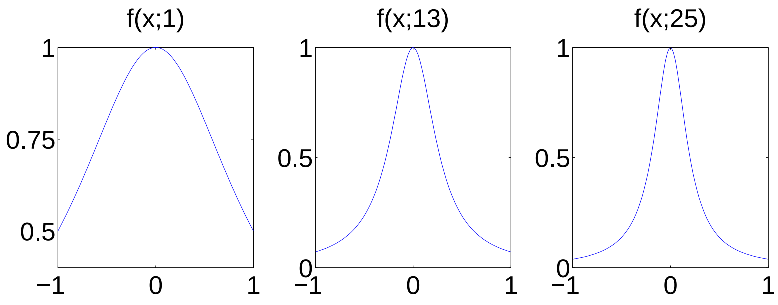


Figure 3.1: $f(x;\mu) = 1/(1+\mu x^2)$ for $\mu = 1$, $\mu = 13$ and $\mu = 25$.

interpolation points are given in Table 3.1. As we see in Figure 3.2 both the GLL and EI interpolation points give an exponential convergence rate, however the EI method is superior and achieves machine precision with only 20 points. The expected divergence of equidistant interpolation is also demonstrated. In the remaining examples we will only compare empirical and GLL interpolation.

Table 3.1: Example 1. Maximum L^2-error over a test sample of 100 random parameter values drawn uniformly from $(1, 25)$. The table displays the error for empirical (e_{EI}), GLL (e_{GLL}) and equidistant (e_{EQ}) interpolation for the 25 first interpolation points.

M	e_{EI}	e_{GLL}	e_{EQ}
5	6.222548e-4	2.070369e-1	2.534919e-1
10	1.712094e-7	5.012065e-2	8.361379e-2
15	6.248218e-12	7.640074e-3	4.194300e-1
20	7.647633e-16	2.186516e-3	3.092732e-1
25	7.610228e-16	3.362375e-4	2.342458

To understand why the EI method performs this much better than standard polynomial interpolation, we take a closer look at the error behavior of the different methods. Remember that we compute the maximum error over a test sample of 100 random parameter values. For M sufficiently large the maximum error for the equidistant and GLL interpolation always occurs for the largest μ in our test sample, i.e. the random parameter closest to 25.

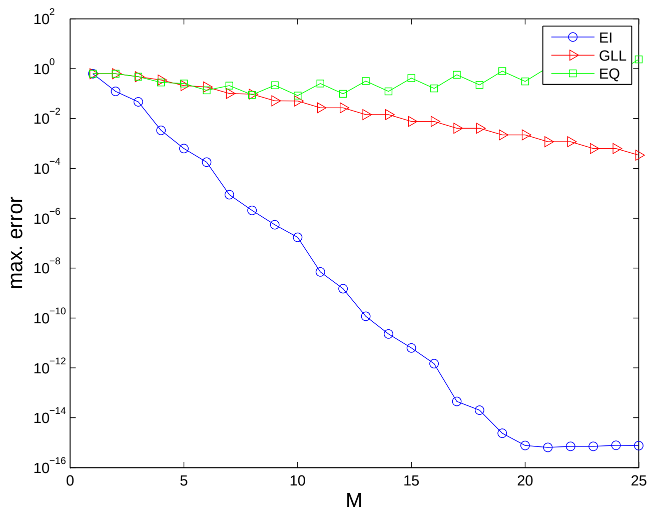


Figure 3.2: Maximum L^2-error over a test sample of 100 random parameter values for $f(x;\mu) = 1/(1 + \mu x^2)$. Both the GLL and empirical interpolation converge at an exponential rate, but the EI method is far superior. As expected, the interpolation based on equidistant points diverges.

This is not surprising. For the equidistant points, $\mu = 25$ gives the largest Runge-oscillations. Also, for large μ our function develops a spike at $x = 0$ (see Figure 3.1). This is where the density of GLL points is lowest, and we thus fail to pick up the sharp gradients created by this spike.

For the EI method however, the error is not dominated by a single, or a few, "bad" parameter values independent of the number of interpolation nodes. This is a result of the greedy algorithm always choosing the parameter value responsible for the largest error as the next function to be included in our interpolation space. As this space is expanded, the "trouble areas" will move around in the parameter space. We can also relate this to the assumption of smooth parameter dependence. We expect that a basis function for a given μ-value will also reduce the error in a neighborhood of this μ. The result, as is clearly demonstrated in this test, is much faster convergence.

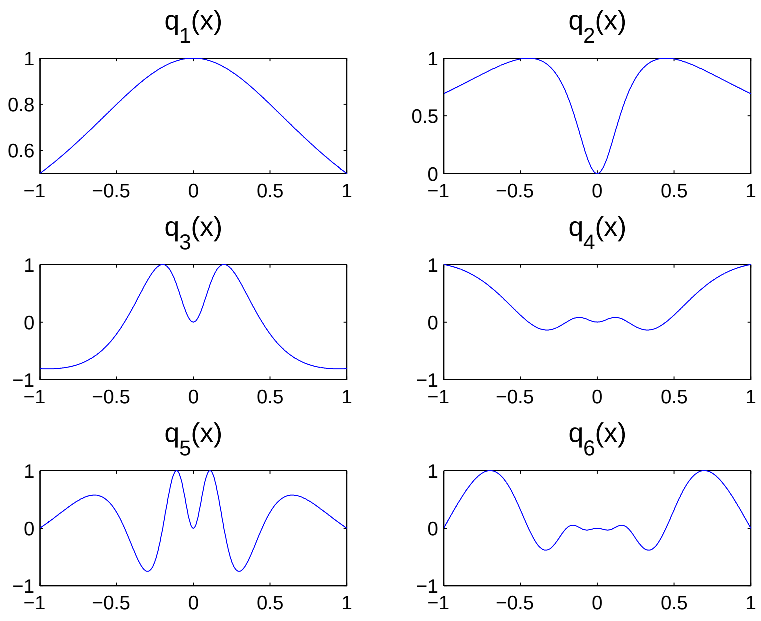


Figure 3.3: The first 6 EI basis functions for $f(x;\mu) = 1/(1+\mu x^2)$.

In Figure 3.3 we have included the first 6 basis functions generated by the EI method for the test function in (3.8). Another interesting feature of the EI method is that all the basis functions, as the underlying f we are approximating, are symmetric around $x = 0$. This is not the case for the Lagrange interpolation polynomials based on the GLL points. An advantage of symmetric basis functions for the interpolation of f is e.g. that if we use only negative (or positive) interpolation nodes, the symmetry will ensure that the interpolant also coincides with f at the mirrored positive (or negative) nodes. This is in fact what happens here. In Figure 3.4 we see the spatial distribution of the interpolation nodes. Notice that all the nodes are negative. Also, the points are clustered towards zero, where the changes in f are greatest. This is a good example of the adaptive ability of the EI method. We could of course achieve something similar with standard polynomial interpolation by distributing the GLL points only on half the interval, in this case [-1,0], and then use the symmetry of f to expand the interpolant to the positive half of Ω as well.

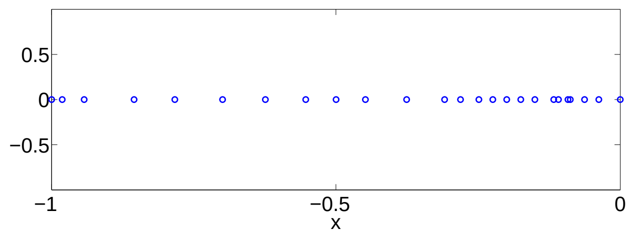


Figure 3.4: The first 25 EI nodes for $f(x;\mu) = 1/(1+\mu x^2)$. Due to symmetry, all the values are negative. Large variations in the basis functions give clustering towards $x = 0$.

Example 2

Our second example is the simple function given as

$$f(x;\mu) = (1+x)^{\mu}, \qquad \text{in } \Omega, \tag{3.9}$$

where $\Omega = (-1, 1)$ and $\mathcal{D} = \{\mu, 1 \le \mu \le 4\}$. In Figure 3.5 we visualized (3.9) for

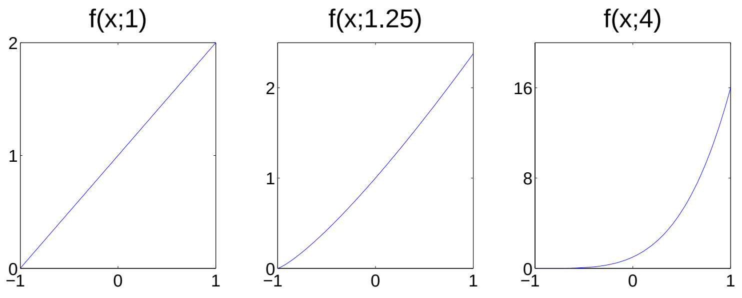


Figure 3.5: $f(x;\mu) = (1+x)^{\mu}$ for $\mu = 1$, $\mu = 1.25$ and $\mu = 4$.

three different values of μ. The underlying function is analytic in the parameter, hence we can expect exponential convergence with the EI method. Notice however that the function will develop a singularity in high order derivatives for fractional μ as x approaches -1. This implies that standard Lagrange interpolation based on the GLL points will only achieve algebraic convergence. The numerical results for this test are collected in Table 3.2.

Again, the EI method clearly demonstrates the best performance. The predicted exponential and algebraic convergence of the EI and GLL interpolation procedures are seen in Figure 3.6. Again, a closer inspection of which parameters is causing the largest errors reveals that for the EI method, this varies as the number of basis functions is increased. For the polynomial interpolation, it is the small fractional values of μ (μ between 1 and 1.25) that are responsible,

Table 3.2: Example 2. Maximum L^2-error over a test sample of 100 random parameter values drawn uniformly from $(1, 4)$. The table displays the error for empirical (e_{EI}) and GLL (e_{GLL}) interpolation for the 25 first interpolation points.

M	e_{EI}	e_{GLL}
5	2.418822e-1	4.336085e+1
10	3.177144e-3	1.350756e-1
15	5.324423e-5	8.820572e-2
20	9.518261e-7	6.530006e-2
25	6.308292e-10	5.174448e-2

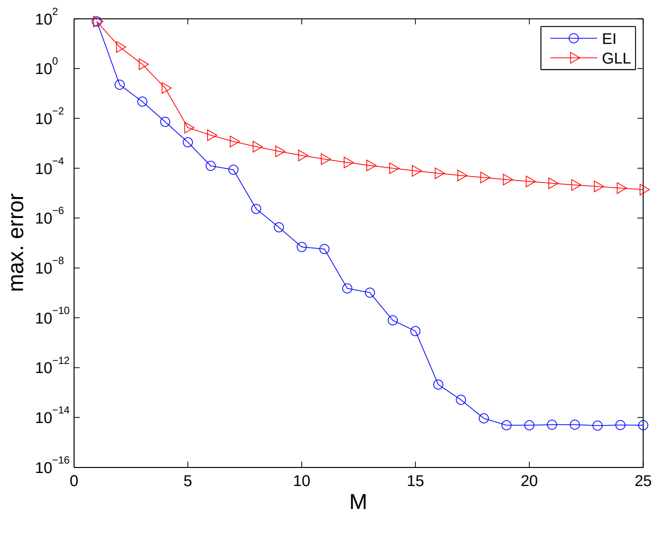


Figure 3.6: Maximum L^2-error over a test sample of 100 random parameter samples for $f(x; \mu) = (1 + x)^{\mu}$. Due to discontinuity in higher derivatives, GLL interpolation only achieves algebraic convergence. EI still converges exponentially as the parameter dependence is analytic.

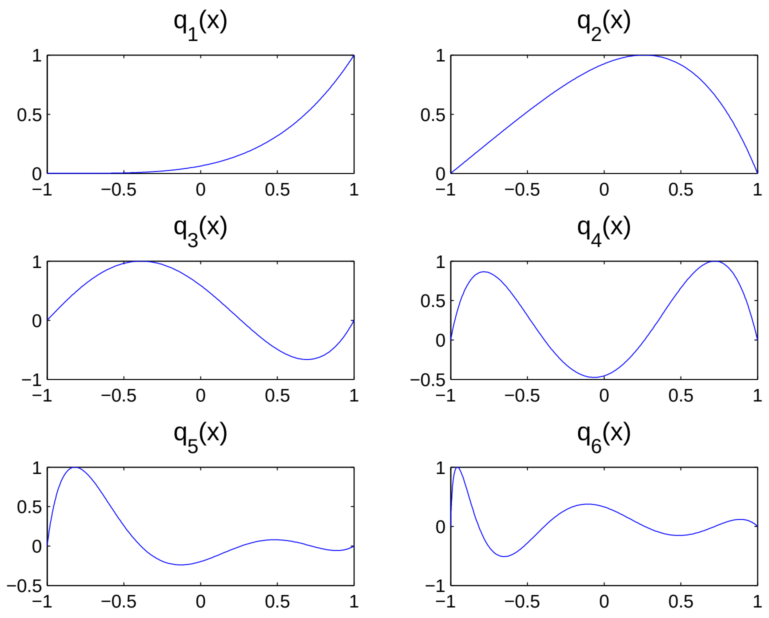


Figure 3.7: The first 6 EI basis functions for $f(x;\mu) = (1+x)^{\mu}$.

as expected. For these values, f will have the lowest regularity and hence also the worst rate of convergence.

Figure 3.7 shows the first 6 EI basis functions for the f in (3.9). We no longer have a symmetric function to interpolate, hence the basis functions are not symmetric. We do however observe that the basis functions adapt to the discontinuity in $x = -1$. In particular, q_5 and q_6 both have sharp gradients at this point to deal with this problem.

Example 3

So far our examples have been of relatively "low spatial frequency". By this we mean that the functions have had a modest degree of oscillation. A direct consequence of Shannon's sampling theorem is that the quality of an interpolant is limited by the number of interpolation points we use per period of the underlying function [25]. Through our final example of this section we will investigate the properties of the EI method when faced with a function of potentially high frequency. To simplify the notation Ω is now shifted to $(-\pi, \pi)$. The function we will interpolate is

$$f(x,\mu) = \cos(\mu x), \qquad \text{in } \Omega. \tag{3.10}$$

Here, we will look at μ such that $0 \le \mu \le 4$. Hence, f will vary from the constant $f(x,0) = 1$ to 4 full cosine periods when $\mu = 4$.

We need some minimal number of interpolation nodes in order to have any hope of representing our underlying function. This is visible in Figure 3.8 where

Table 3.3: Example 3. Maximum L^2-error over a test sample of 100 random parameter values drawn uniformly from $(0, 4)$. The table displays the error for empirical (e_{EI}) and GLL (e_{GLL}) interpolation for the 40 first interpolation points.

M	e_{EI}	e_{GLL}
5	2.968654e-1	1.538479
10	7.367640e-8	1.210420
20	1.054075e-15	1.167325e-3
30	1.101046e-15	1.686058e-9
40	1.187735e-15	2.157651e-14

the error behavior is presented. Before we reach 5 nodes, none of the methods make any progress in reducing the error. However, whilst the GLL interpolation remains ineffective until the number of nodes is about 15, the convergence of the EI method is better. At 15 nodes the function is almost perfectly represented. For $\mu = 4$ this implies only 4-5 nodes on average for each period.

Even though the GLL interpolation also converges quickly when the number of interpolation nodes is sufficient to resolve the periodic function, the EI method's use of adaptive basis functions again results in much faster convergence.

An obvious drawback with the "global" GLL approach used here is that the distribution of points will be clustered towards the ends of the interval. Thus, in the center points will be scarce. A GLL distribution within each period would have been more efficient, but this would require us to calculate this distribution for each new value of the parameter μ (or the frequency).

When we look at the EI basis functions we get a better understanding of why this method performs well with a very limited number of interpolation nodes per period. We see that already with the first 4 $q's$ we are able to represent the highest frequency, $\cos(4x)$ which is given by $q_1(x) - 2q_4(x)$.

For this example we can envision a simple application. Assume you are a scientist listening for a signal. You know the signal will have the form of a cosine, and the frequency is within some band $\mathcal{D}$, but otherwise unknown. If you have e.g. 10 sensors at your disposal, what is the optimal placement of these sensors if you want be able to reconstruct the signal for any $\mu \in \mathcal{D}$. The EI method will not give you the optimal placement, but if you choose the first 10 EI nodes you can expect the error to be very small ($< 10^{-7}$) according to the numerical experiments.

Confident of the performance of the EI method, we now move on to the case where the underlying function is no longer analytic in the parameter.

3.4.2 Limited regularity in the parameter

The numerical examples presented so far have all had analytic dependence in the parameter ensuring exponential convergence. Now we will investigate the properties of the EI method when this is not the case. To do this we can approximate a simple, one-dimensional function where the regularity is easily

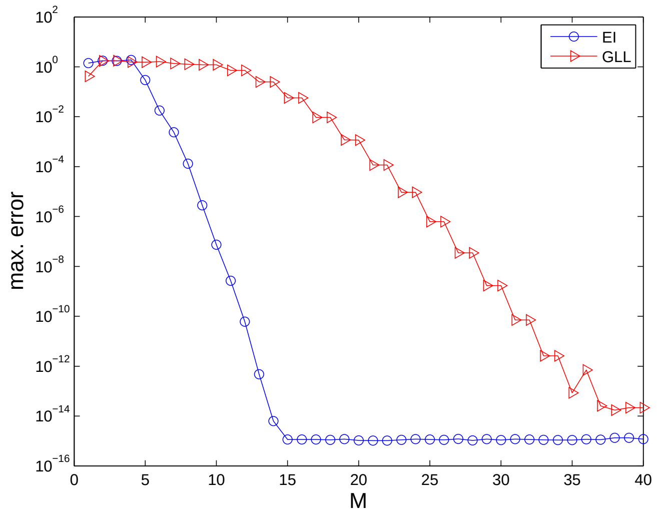


Figure 3.8: Maximum L^2-error over a test sample of 100 random parameter samples for $f(x;\mu) = \cos(\mu x)$ and $0 \leq \mu \leq 4$. As we compute the maximum error, we need quite a few points to capture the oscillation for large μ. Again, the EI method outperforms the polynomial approach.

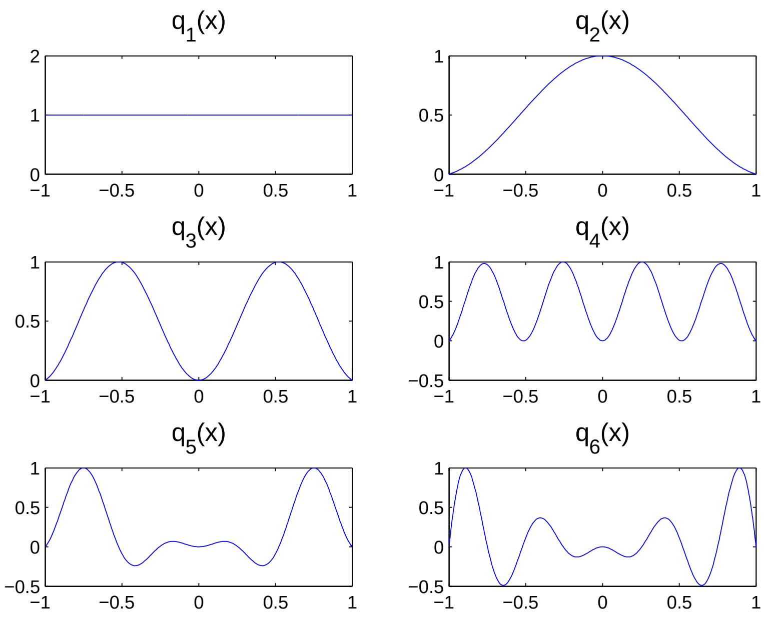


Figure 3.9: The first 6 EI basis functions for $f(x;\mu) = \cos(\mu x)$.

controllable. Here we choose our target function, defined over $\Omega = (-1, 1)$ as

$$f_n(x, \mu) = |x - \mu|^n, \qquad \text{in } \Omega, \tag{3.11}$$

where $n \geq 1$ is an integer and the parameter domain $\mathcal{D} = \{\mu, -1 \leq \mu \leq 1\}$. First note that for n even, f_n will be equal to the polynomial $(x - \mu)^n$. In this case, the EI method will be exact for $M > n$ because $f_n \in X_{n+1}$. As f_n is clearly a polynomial of degree n we must realize that $q_1, \ldots, q_{n+1}$ span the same space. But as each of the q_i's is also by construction a polynomial of degree n, and $\dim(X_{n+1}) = n + 1$, this is obvious. Hence, the function we are interpolating is a member of our approximation space and the projection error is zero. This property is also verified numerically in Figure 3.10.

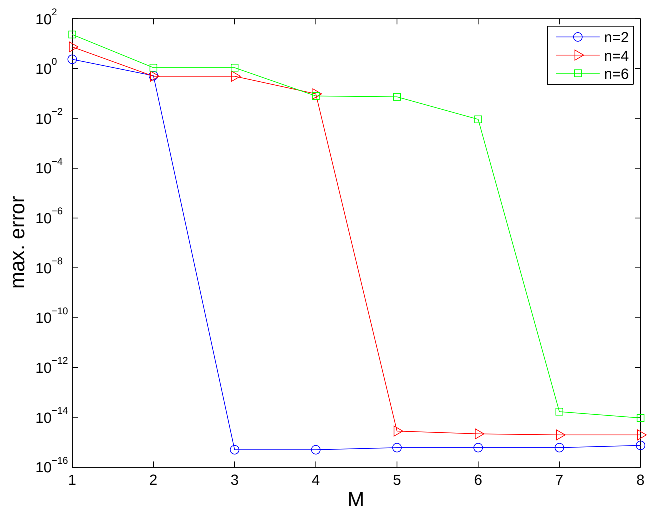


Figure 3.10: Maximum L^2-error over a test sample of 100 random parameter samples for $f_n = |x - \mu|^n$, n even. The error drops to zero as $M = n + 1$ as the approximation space X_M will then contain f_n.

However, for n odd, we will demonstrate that $f_n \in H^{n+1/2-\varepsilon}(\Omega)$ for an arbitrarily small $\varepsilon > 0$. First, let us start with the following result

Lemma 3.3 *Assume the function v on $\Omega = (-1, 1)$ is such that $v_x \in L^2(-1, 1)$. Then $v \in H^1(-1, 1)$.*

Proof: Consider the function $\tilde{v} = v - v(-1)$. Now by the fundamental theorem of calculus and the Cauchy-Schwarz inequality we get

$$\begin{aligned}
\tilde{v}(x) &= v(-1) + \int_{-1}^{x} v_t \,\mathrm{d}t - v(-1) = \int_{-1}^{x} 1 \cdot v_t \,\mathrm{d}t \\
&\le \left(\int_{-1}^{x} 1^2 \,\mathrm{d}t \right)^{\frac{1}{2}} \left(\int_{-1}^{x} v_t^2 \,\mathrm{d}t \right)^{\frac{1}{2}} \\
&\le \left(\int_{-1}^{1} \mathrm{d}t \right)^{\frac{1}{2}} \left(\int_{-1}^{1} v_t^2 \,\mathrm{d}t \right)^{\frac{1}{2}} \\
&= \sqrt{2}\, |v|_{H^1(-1,1)}.
\end{aligned}$$

The assumption $v_x \in L^2(-1,1)$ ensures that $|v|_{H^1(-1,1)}$ is bounded. Also,

$$\int_{-1}^{1} \tilde{v}^2 \,\mathrm{d}x \le 4|v|^2_{H^1(-1,1)} < \infty.$$

As v is only a constant away from $\tilde{v}$, we finally get

$$\|v\|^2_{H^1(-1,1)} = \int_{-1}^{1} (v^2 + v_x^2) \,\mathrm{d}x < \infty,$$

and as claimed $v \in H^1(-1,1)$.

It immediately follows that

Corollary 3.4 *Assume the function v on $\Omega = (-1,1)$ is such that $v_x \in H^\sigma(-1,1)$, $\sigma \ge 0$ ($\sigma = 0$ is $L^2(-1,1)$). Then $v \in H^{\sigma+1}(-1,1)$.*

Now, it can be shown (see Appendix A) that $|x| \in H^{\frac{3}{2}-\varepsilon}$. Hence, by using Corollary 3.4 n times, the regularity of f_n from (3.11) is given as

$$f_n(x,\mu) = |x-\mu|^n \in \begin{cases} H^\infty(-1,1), & n \text{ even} \\ H^{\frac{2n+1}{2}-\varepsilon}(-1,1), & n \text{ odd} \end{cases}$$

For the remainder of this example we will look at odd n. Now that we have established what space f_n belongs to, we may use the result from Section 2.5 to get the *a priori* error estimate for GLL polynomial interpolation

$$\|f_n(\cdot;\mu) - \mathcal{I}_M^{GLL}[f_n](\cdot;\mu)\|_{L^2(\Omega)} < cM^{-\frac{2n+1}{2}} \|f_n\|_{H^{n+1/2-\varepsilon}(\Omega)}. \tag{3.12}$$

There also exist several results in the literature for approximation and interpolation of functions of the type $|x|^\lambda$. In [26] it is demonstrated that the error in polynomial interpolation of this function is bounded by

$$\||x|^\lambda - \mathcal{I}_M^{GLL}[|x|^\lambda]\|_{L^p(\Omega)} < \tilde{c}M^{-(\lambda+1/p)}, \tag{3.13}$$

which for $\lambda = n$ and $p = 2$ is in accordance with (3.12).

From (3.11) it is obvious that the regularity in the parameter μ is the same as in the variable x. An interesting question is now: Will the EI method demonstrate a similar algebraic convergence depending on the parametric regularity? To investigate this we here present numerical results which compare the empirical and polynomial interpolation for the three cases $n = 1$, $n = 3$ and $n = 5$.

Table 3.4: Maximum L^2-error over a test sample of 100 random parameter values drawn uniformly from $(-1,1)$ for $f_n(x;\mu)$. e^i_{EI} and e^i_{GLL} denote the error for $n=i$. We see that the error is very similar for the two methods.

M	e^1_{EI}	e^1_{GLL}	e^3_{EI}	e^3_{GLL}	e^5_{EI}	e^5_{GLL}
1	1.381	9.967e-1	1.969	9.923e-1	3.039	2.153
2	8.165e-1	8.165e-1	3.920e-1	3.099	7.116e-1	1.596e+1
4	2.864e-1	1.559e-1	5.976e-2	1.084e-1	7.729e-2	8.849e-1
8	1.006e-1	5.164e-2	4.246e-3	3.665e-3	1.210e-3	1.647e-3
16	3.471e-2	1.937e-2	3.354e-4	2.532e-4	1.018e-5	1.864e-5
32	1.238e-2	6.311e-3	3.107e-5	2.066e-5	2.857e-7	3.423e-7
64	2.867e-3	2.373e-3	2.413e-6	1.764e-6	4.928e-9	7.080e-9
128	1.121e-3	9.010e-4	1.753e-7	1.550e-7	9.778e-11	1.523e-10
256	4.187e-4	2.969e-4	9.945e-9	1.386e-8	1.574e-12	1.705e-11
512	9.244e-5	1.053e-4	1.073e-9	1.229e-9	4.767e-14	4.837e-12

Table 3.5: Estimated algebraic convergence rate ($e_M \sim M^{-s}$) for the EI method from the last 5 data points in Table 3.4 ($M = 32,\dots,512$).

n	s
1	1.6905
3	3.7538
5	5.6642

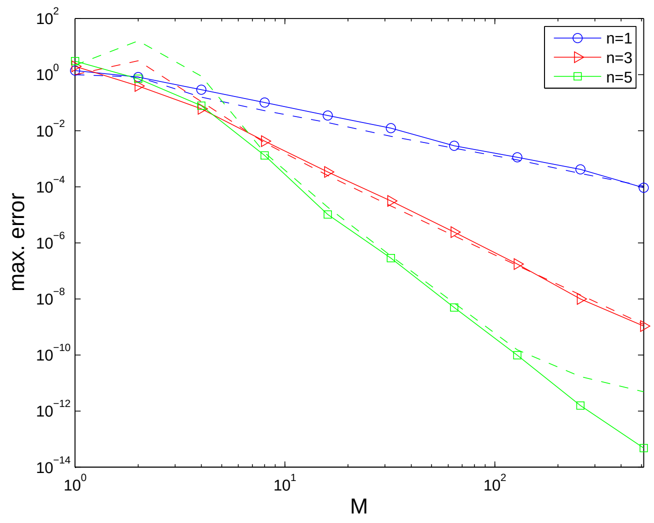


Figure 3.11: Maximum L^2-error for the EI method over a test sample of 100 random parameter samples for $f_n = |x-\mu|^n$, n odd. We now only have algebraic convergence. The dotted line is the corresponding GLL interpolation error with the same number of interpolation nodes.

Table 3.6: Estimated Lebesgue constants for f_n, $n = 1, 3, 5$.

M	Λ_M^1	Λ_M^3	Λ_M^5
1	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000
4	1.0001	1.2782	1.1309
8	1.0014	2.0993	2.5693
16	1.0027	2.2926	3.3534
32	1.0150	2.4709	4.8136
64	1.0294	2.5238	3.7505
128	1.0583	2.8011	4.0686
256	1.1536	3.0913	4.7062
512	1.3556	2.9498	4.8083

The evaluation of the interpolants is the same as in Section 3.4.1. After the interpolants are created, we find the maximum error over a new test set of 100 random parameter values $\mu \in \mathcal{D}$.

The results presented in Table 3.4 and Figure 3.11 show that the empirical and GLL interpolation is remarkably similar for this problem. We indeed observe an algebraic convergence rate. In table 3.5 the estimated rates are given. The *a priori* estimates of (3.12) and (3.13) seem to give quite sharp estimates of the actual error behavior also for the EI method.

To understand why we get this rate of convergence, we must return to the error analysis of Section 3.3. Although the conditions for Theorem 3.2 are no longer fulfilled, we may still use the general result in Lemma 3.1. We know that the projection error will follow (3.12) and the interpolation is thus bounded by

$$\begin{aligned}\|f_n(\cdot;\mu) - \mathcal{I}_M^{EI}[f_n](\cdot;\mu)\|_{L^2(\Omega)} &\leq c(1+\Lambda_M)M^{-(n+\frac{1}{2})}\|f_n(\cdot;\mu)\|_{L^2(\Omega)} \\ &= \bar{c}\Lambda_M M^{-(n+\frac{1}{2})}\end{aligned} \tag{3.14}$$

for some constants c and $\bar{c}$.

As the Lebesgue constant, Λ_M, depends on the choice of interpolation nodes, it will be different for each problem when we use the EI method. Because of this it is difficult to give precise *a priori* error estimates for the case where the parametric regularity is limited. We can however compute the Lebesgue constants for our three cases ($n = 1$, $n = 3$ and $n = 5$) after the interpolation procedure is completed.

If we do this for our example, we see in Table 3.6 that the Lebesgue constant is hardly increasing at all, and we can approximate Λ_M by a constant. This implies that the dominating error term is the projection error and we get an asymptotic error rate given by $e_M \sim M^{-(n+\frac{1}{2})}$.

3.4.3 Multiple dimensions

In the one-dimensional setting, interpolation, and polynomial interpolation in particular, is quite well documented. There exists many results for the best points and basis functions under different optimality conditions. In multiple dimensions however, the situation is more open. Some of the results from the scalar case can be generalized, but it is typically much harder to find optimal points as the number of dimensions increase.

Even though the EI method is in no way claimed to be optimal, it does present a simple and automatic algorithm for determining interpolation nodes and basis functions in higher dimensions. Also, if the conditions of Theorem 3.2 are valid, we still achieve exponential convergence.

As will become apparent in later chapters, the potential gain of a decoupling into an offline parameter independent part, and an online parameter dependent part grows with each added dimension.

The multidimensional procedure is in principle no different from the one-dimensional case. However, the offline complexity will grow very fast for each added dimension as the number of different parameter combinations in our training set will be $\mathcal{N}_{x_1} \cdot \mathcal{N}_{x_2} \cdot \ldots \cdot \mathcal{N}_{x_d}$ in d dimensions.

However, the online cost to calculate the parameter dependent coefficients will still be only $\mathcal{O}(M^2)$. It should be pointed out that for multiple dimensions we usually require a larger number of terms in our EI expansion to get an adequate representation of the underlying function.

Example 1

First we present the two dimensional example

$$f(\mathbf{x};\boldsymbol{\mu}) = \frac{1}{\sqrt{(x_1-\mu_1)^2+(x_2-\mu_2)^2}}, \qquad \text{in } \Omega. \tag{3.15}$$

We have $\Omega = (0,1)^2$ and $\mathcal{D} = (-1,-0.01)^2$. For our test set, Ξ_t, we use a uniform distribution of 50 points in each spatial direction which gives a total number of $50^2 = 2500$ parameter values. Each of the EI basis functions are represented using tensor product GLL interpolants of 75×75 points.

A comparison of the convergence of EI and GLL interpolation is presented in Figure 3.12. Clearly, the EI method is superior to the GLL approach. The M given in the horizontal axis is the total number of interpolation points. As the GLL interpolant is based on a tensor product distribution of points, the error is computed for 1 to 7 points in each spatial direction.

The EI method is not restricted by this tensor product form, and the points can be concentrated in areas where the function is irregular. This behavior is demonstrated in Figure 3.13. The parameter values and interpolation nodes selected by the greedy algorithm is clustered at the corners $\boldsymbol{\mu} = (-0.01,-0.01)$ and $\mathbf{x} = (0,0)$. It can be demonstrated that our example will develop a boundary layer in this region [7]. Where the EI method adapts to this, GLL interpolation will waste resources by covering areas of Ω where only a few points are necessary.

Example 2

Our second example is

$$g(\mathbf{x};\boldsymbol{\mu}) = \frac{1+\frac{\pi^2}{4}(\mu_2-\mu_1-(\mu_1+\mu_2)x_2)^2\sin^2(\frac{\pi}{2}(x_1+1))}{1+(\mu_1+\mu_2)\cos(\frac{\pi}{2}(x_1+1))}, \qquad \text{in } \Omega. \tag{3.16}$$

At first sight this might seem like a strange function, but we choose it because it reappears in Section 8.3.4. For this example we have $\Omega = (-1,1)^2$ and $\mathcal{D} =$

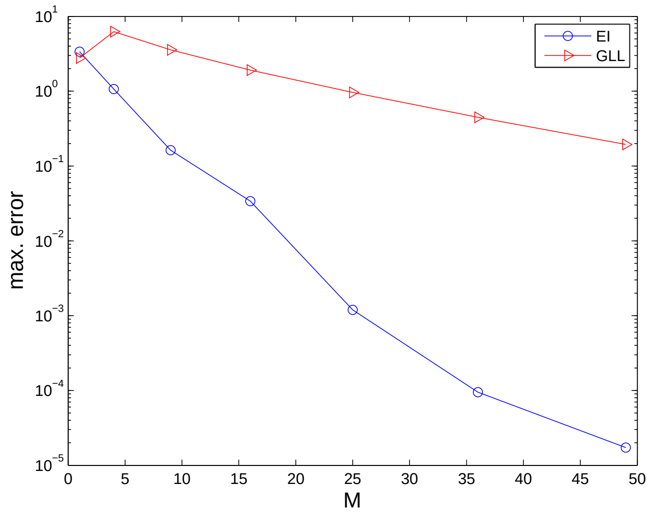
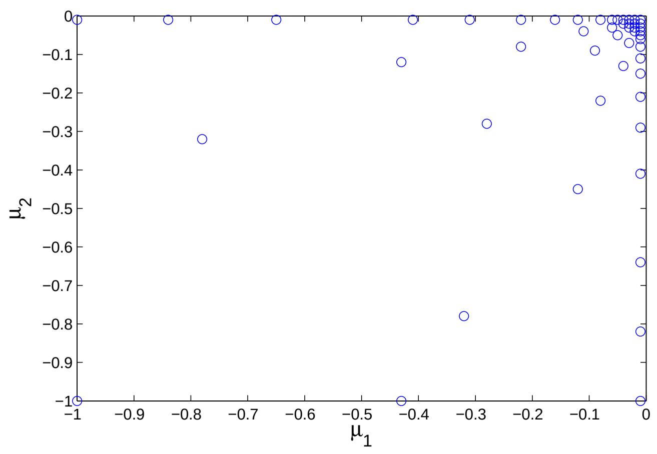


Figure 3.12: EI and GLL error for the two dimensional example in (3.15). The EI method is clearly superior.

$(-0.4, 0.4)^2$. The number of training samples are the same as in the previous example, i.e. 2500 samples, and again the EI basis functions are represented using tensor product GLL interpolants with 75 points in each spatial direction.

Again, the empirical interpolant converges quickly, as seen in Figure 3.14. Also, the selected parameter values and interpolation points is interesting. In addition to selecting parameters at the boundary, we also observe a tendency to pick values where $\mu_1 \approx \mu_2$. These are the parameter values resulting in the greatest spatial variation of g.

The choice of interpolation nodes are distributed in three distinct lines at $x_2 = -1$, $x_2 = 0$ and $x_2 = 1$. This may be connected to where g achieves its maximum and minimum values. This is more thoroughly discussed in the second part of Section 8.3.5.



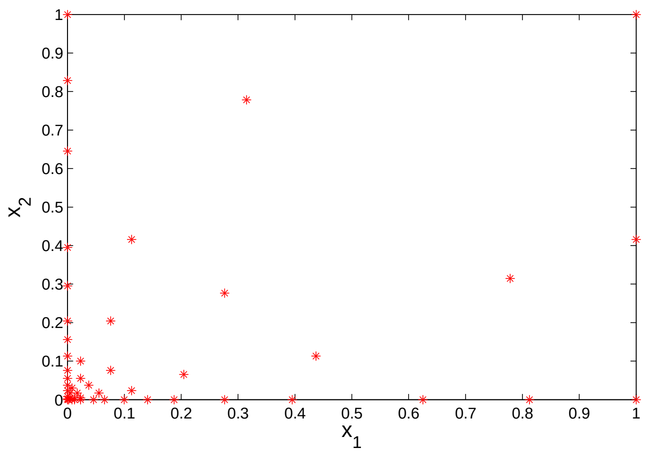


Figure 3.13: Parameter samples (left) and interpolation nodes (right) selected by the greedy algorithm for the function in (3.15). The spatial points are clearly consentrated at the boundary layer at $\boldsymbol{\mu} = (-0.01, -0.01)$ and $\mathbf{x} = (0, 0)$.

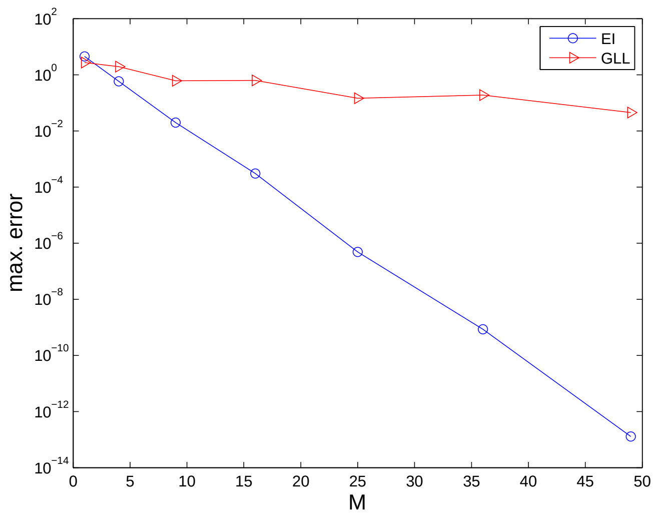


Figure 3.14: EI and GLL error for the two dimensional example in (3.16).

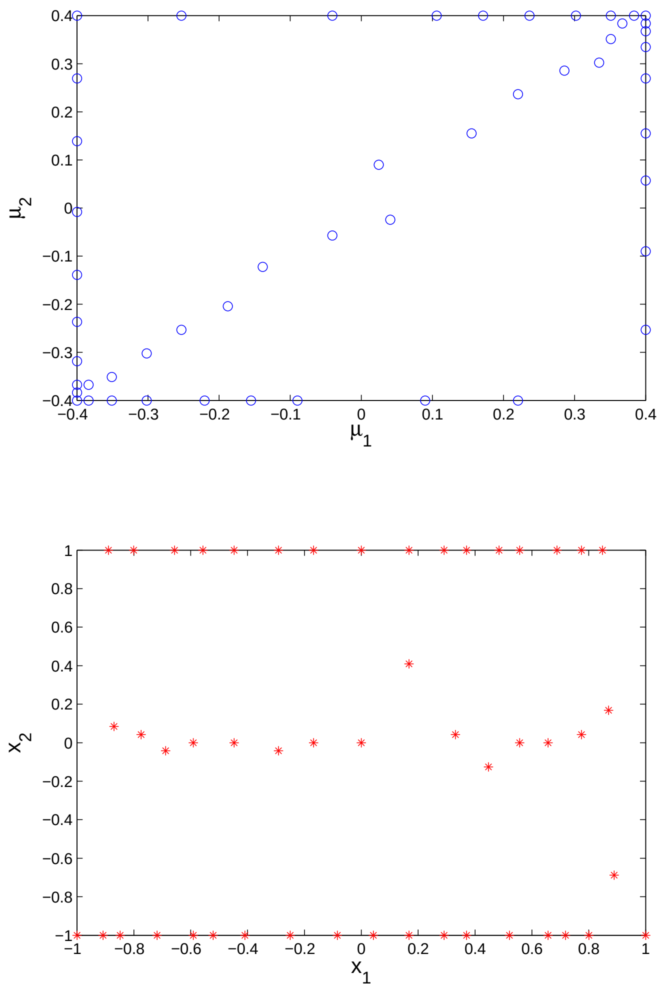


Figure 3.15: Parameter samples (top) and interpolation nodes (bottom) selected by the greedy algorithm for the function in (3.16). The parameters are concentrated around the edges, but many samples are also placed where μ_1 and μ_2 are approximately equal. The spatial points are distributed in three distinct lines.

Chapter 4

Quadrature using Empirical Interpolation

A simple application of the interpolant obtained with the EI method is numerical computation of integrals. We will call this type of numerical integration Empirical Interpolation Quadrature (EIQ).

4.1 Quadrature procedure

Assume we want to evaluate the integral

$$\int_{\Omega} f(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega, \tag{4.1}$$

for a large number of different parameter values. Given the empirical interpolation matrix B_M, nodes z_M and basis functions $q_1,\dots,q_M$, the approximation of the integral (4.1) is simple. First notice that as our interpolant, by construction, is expressed as an affine parameter dependent expansion, we can write

$$\int_{\Omega} f(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega \approx \int_{\Omega} f_M(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega = \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu}) \int_{\Omega} q_i(\mathbf{x})\,\mathrm{d}\Omega.$$

Now, each of the parameter independent integrals $\int_{\Omega} q_i(\mathbf{x})\,\mathrm{d}\Omega$ is computed and stored as elements in a vector $\underline{\Psi} \in \mathbb{R}^M$. For a new parameter $\boldsymbol{\mu}$ we only need to solve the lower triangular system

$$\sum_{j=1}^{M} (B_M)_{ij}\varphi_j(\boldsymbol{\mu}) = f(z_i;\boldsymbol{\mu}), \quad i = 1,\dots,M.$$

We then compute the sum

$$\sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu})\Psi_i \tag{4.2}$$

to obtain our integral approximation.

So far we have used high order GLL interpolants to represent our EI basis functions. This makes the evaluation of the offline, parameter independent integrals simple using standard GLL quadrature as described in Section 2.4. Note however, that the offline integrals may in principle be evaluated using any type of numerical integration method given that it is sufficiently accurate.

Also, as the output of the offline stage is only a single number for each term in the EI expansion, the online computation will be very fast. In the next section we discuss both the offline and online computational complexity in detail.

4.2 Computational complexity

As the EI method relies on decomposing the problem into a parameter independent offline stage and a parameter dependent online stage, we give a detailed presentation of the computational complexity for both stages.

As a part of the offline discussion we also give the computational complexity of performing the greedy algorithm. For later applications the analysis will not be as complete as the algorithm will be very similar in these cases.

We also study the storage requirement for both stages and compare the EI method to other alternative numerical integration schemes.

4.2.1 Offline complexity

The offline computational complexity will of course depend both on the way we represent our EI basis functions (and thus implicitly on the way the greedy algorithm is implemented) and on the choice of numerical integration scheme for the offline integrals.

We will look at the case where the basis functions are represented using high order GLL polynomial interpolants using $\mathcal{N}+1$ points. With this approach, each of the basis functions $q_1, \dots, q_M$ is represented by vectors $\underline{q}_1, \dots, \underline{q}_M$ containing the value of the function at all the GLL points. This is stored as a matrix $Q_M = [\underline{q}_1, \dots, \underline{q}_M] \in \mathbb{R}^{(\mathcal{N}+1)\times M}$ and we thus get a storage requirement of $\mathcal{O}(\mathcal{N}M)$.

Using this representation we can easily evaluate all the parameter independent integrals with high accuracy using GLL quadrature. This is done by calculating

$$\Psi_i = \sum_{\alpha=0}^{\mathcal{N}} \rho_\alpha (Q_M)_{\alpha i}, \quad i = 1, \dots, M.$$

The offline integration is thus performed in $\mathcal{O}(\mathcal{N}M)$ operations.

We now turn to the actual computation of the EI basis functions and interpolation points. The dominating term in Algorithm 3.1 is clearly

$$\boldsymbol{\mu}_m(x) = \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} \|f(\cdot;\boldsymbol{\mu}) - \mathcal{I}_{m-1}[f](\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega}.$$

As mentioned, in the actual implementation we use a discrete approximation $\Xi_t \subset \mathcal{D}$. Recall that the size of Ξ_t is $\mathcal{M}$. For each $\boldsymbol{\mu} \in \Xi_t$ we have to compute the interpolant and the corresponding error. The interpolant is found by solving an $M \times M$ lower triangular system and then computing the matrix vector product of Q_m and this solution. The error is approximated by the maximum error of all the GLL points. Hence the total complexity of this step is $\mathcal{O}(\mathcal{N}\mathcal{M}m)$. If we add up all M steps we get the total complexity $\mathcal{O}(\mathcal{N}\mathcal{M}M^2)$.

The offline cost is clearly substantial, but as will become apparent in the next section, if we are willing to invest the resources to do this work, the potential savings when faced with a new parameter are great.

4.2.2 Online complexity

As mentioned in Section 4.1 the online evaluation of the EI quadrature will be very fast. As we have stored all the parameter independent integrals, the solution is obtained in two steps. First by solving a lower triangular system of equations, which is done in $\mathcal{O}(M^2)$ operations. Second, we compute the sum in (4.2) which is only $\mathcal{O}(M)$ operations. The total number of floating point operations is thus $\mathcal{O}(M^2)$.

In addition to fast evaluation, for the online stage we only need to store the vectors $\underline{\Psi}$, $\underline{\varphi}(\boldsymbol{\mu})$ and $\underline{f}$ and the interpolation matrix B_M. This gives a total of $3M + \frac{1}{2}M(M+1) = \mathcal{O}(M^2)$ double precision numbers.

4.3 Error analysis

We denote the EI quadrature error by $e_M^{\mathcal{N}}$. This error will contain two components (if we disregard rounding error). The first is the error from approximating f by the empirical interpolant f_M. The second is the quadrature error from evaluation of the offline integrals.

The first we have already introduced as e_M. If we now denote the offline quadrature errors by $e_i^{\mathcal{N}}, i = 1, \dots, M$ we get

$$\begin{aligned}
e_M^{\mathcal{N}} &= \left| \int_\Omega f(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega - \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu}) \sum_{\alpha=0}^{\mathcal{N}-1} \rho_\alpha q_i(\xi_\alpha) \right| \\
&= \left| \int_\Omega f(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega - \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu}) \left(\int_\Omega q_i(\mathbf{x})\,\mathrm{d}\Omega - e_i^{\mathcal{N}} \right) \right| \\
&= \left| \int_\Omega f(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega - \int_\Omega f_M(\mathbf{x};\boldsymbol{\mu})\,\mathrm{d}\Omega + \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu}) e_i^{\mathcal{N}} \right| \\
&\le \left| \int_\Omega e_M\,\mathrm{d}\Omega + e^{\mathcal{N}} \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu}) \right| \\
&\le c_1 e_M + |c_2(\boldsymbol{\mu})| e^{\mathcal{N}}.
\end{aligned}$$

Here we have introduced $e^{\mathcal{N}} = \max_i(e_i^{\mathcal{N}})$, $c_1 = |\Omega|$ and $c_2(\boldsymbol{\mu}) = \sum_{i=1}^{M} \varphi_i(\boldsymbol{\mu})$. As expected, the total error contains one term from the interpolation error and one term from the offline quadrature error.

We can now state the following about the EI quadrature error

Theorem 4.1 *Assume $f(\mathbf{x};\boldsymbol{\mu}) \in H^\sigma(\Omega)$ satisfies the conditions of Theorem 3.2. Then there exists constants $\tilde{c}_1$, $\tilde{c}_2(\boldsymbol{\mu})$ and $\tilde{\alpha}$ such that the error in the empirical interpolation quadrature procedure is bounded by*

$$e_M^{\mathcal{N}} \le \tilde{c}_1 \exp\{-\tilde{\alpha} M\} + \tilde{c}_2(\boldsymbol{\mu}) \mathcal{N}^{-\sigma}. \tag{4.3}$$

Proof: This follows directly from Theorem 3.2 and Theorem 2.1.

This shows that we should seek a balance where none of the terms are completely dominant. If either $\mathcal{N}$ or M is to big we waste computational power without achieving any significant reduction in error. Also, if the function we are integrating is analytic both in the variable $\mathbf{x}$ and the parameter $\boldsymbol{\mu}$ the EI quadrature achieves exponential convergence both in M and $\mathcal{N}$.

Even though we run the risk of doing a little too much work we should however be conservative when choosing $\mathcal{N}$. This is because it will only affect the offline stage, and the reliability of the EI results is dependent on sufficient offline accuracy. Choosing $\mathcal{N}$ large also ensures exponential convergence in M even when the function has limited regularity in $\mathbf{x}$.

4.4 Numerical examples

We present three numerical examples of varying complexity to demonstrate when the EIQ approach can be preferable.

4.4.1 Simple engineering application

The EI quadrature can also be applied to engineering challenges. We here present a very simple, one dimensional example, which of course could be solved in a much easier fashion. However, for the sake of demonstration we will use the EIQ approach.

Consider the following problem. Imagine you are an engineer designing a roof. The architect demands the roof to have the shape of a trigonometric function. Also, the project manager is trying to minimize cost. Assume that the amount of reinforcement required is proportional to the length of the roof and inverse proportional to the square-root of the height. In addition, the cost of the material used for the reinforcement depends linearly on the amount.

If we let the height of the roof, i.e. the amplitude of our trigonometric function be our parameter μ the mathematical definition of the problem becomes

$$\min_{\mu\in\mathcal{D}} C(\mu) = \frac{P}{\sqrt{\mu}} \int_0^L \sqrt{1+\mu^2\cos^2(t)}\,\mathrm{d}t. \tag{4.4}$$

Here, P is the cost per unit reinforcement material, and L the length of the building. In our numerical experiments we let $L=\pi$ and $\mathcal{D}=\{\mu, 1\le\mu\le 5\}$. The value of P does not influence the problem and can be set equal to 1.

To solve (4.4) we first create an affine parameter dependent expansion of the integration kernel using the greedy algorithm. We then continue by evaluating the offline, parameter independent integrals. To solve the minimization problem we create a very fine uniformly distributed grid over $\mathcal{D}$ using 10000 values. Finally, we can quickly (about 0.3 seconds)[1] compute the integral for each of these values to find the minimum.

The evaluated cost function is given in Figure 4.1. We see that the optimal value is $\mu_{opt} = 1.5325$ which gives cost of $C_{opt} = 2.3354$.

In this example, the kernel function is analytic also in the variable. Thus we will get very good results with GLL quadrature as well. An estimate for

[1]As we are mainly interested in relative improvements of computational speed in this work, we do not give any spesific hardware information regarding timing results.

Table 4.1: Maximum quadrature error for the simple engineering problem. The "exact" integrals are computed using built-in quadrature procedures in MATLAB$^{\mathrm{TM}}$ with a tolerance of 10^{-14}. It is thus no point in including more than 20 EI nodes.

M	e_{EI}	e_{GLL}
2	6.214595e-02	3.355060
4	1.132111e-03	5.197304e-1
6	3.491698e-06	1.255407e-1
8	1.856523e-07	4.415687e-2
10	2.766356e-10	1.798382e-2
15	1.509903e-14	2.599176e-3
20	8.881784e-15	4.478141e-4
30	-	1.836354e-5

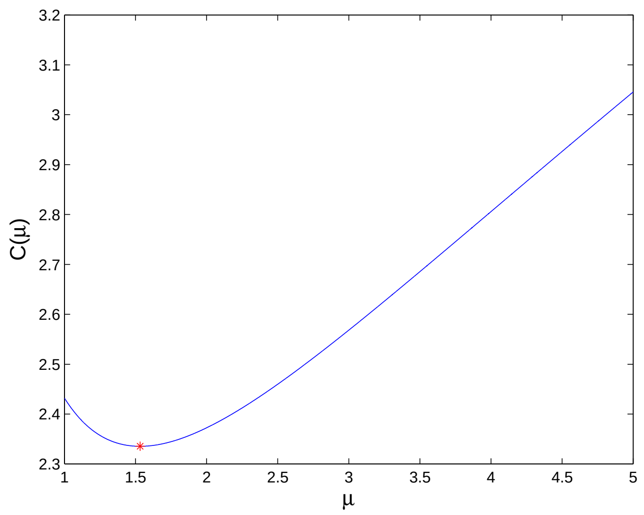


Figure 4.1: Cost function $C(\mu)$ computed over a very fine grid. The minimum $(*)$ is $C_{opt} = 2.3354$ for $\mu_{opt} = 1.5325$.

the maximum quadrature error as a function of interpolation nodes is given in Table 4.1. Immediately it looks as if the EI quadrature again outperforms the GLL quadrature. However, it is important to notice that for a new parameter the amount of online work for M nodes is $\mathcal{O}(M^2)$ for EI quadrature, but only $\mathcal{O}(M)$ for GLL quadrature. If we compare the error for M EI points and M^2 GLL points, which would result in roughly the same amount of work, the error results are more similar.

4.4.2 Moments of the β-distribution

Let us now look at a typical parameter dependent function; a probability density function (pdf). A possible application of the EI quadrature procedure is the computation of statistical moments. For a one dimensional, continuous probability distribution $p(x;\boldsymbol{\mu})$ the n'th moment is defined as the the expected value of X^n where $X \sim p(x;\boldsymbol{\mu})$. This is computed as

$$E(X^n) = \int_{-\infty}^{\infty} x^n p(x;\boldsymbol{\mu})\,\mathrm{d}x. \tag{4.5}$$

We will here use EI quadrature to approximate the first moment, or mean, of the β-distribution. The same approach can also be applied to the second moment. If we know these quantities we can also easily compute the variance $Var(X) = E(X^2) - (E(X))^2$.

Following standard notation, the parameter $\boldsymbol{\mu} = (\alpha,\beta)^T$. A stochastic variable $X \sim beta(\alpha,\beta)$ if

$$p(x;\alpha,\beta) = \begin{cases} \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, & x \in (0,1) \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where the parameters $\alpha,\beta > 0$ and the beta function B is defined as

$$B(u,v) = \int_0^1 w^{u-1}(1-w)^{v-1}\,\mathrm{d}w.$$

The exact expression for the mean is known and given as

$$E(X) = \frac{\alpha}{\alpha+\beta}.$$

This enables us to easily evaluate the EIQ error.

In our first numerical test, where we only investigate the error caused by the EI expansion, we use $\mathcal{N} = 1000$. This is chosen very high as p will develop a singularity in higher derivatives similar to Example 2 in Section 3.4.1. Our parameter sample space is $\mathcal{D} = (1,4)^2$. We discretize $\mathcal{D}$ using a uniformly spaced grid of size $\mathcal{M} = 200^2$.

Recall from section 4.2.1 that the offline complexity is $\mathcal{O}(\mathcal{N}\mathcal{M}M^2)$, hence the required work will in this case be substantial. The error for the EI quadrature and GLL quadrature as a function of the number of interpolation nodes M is given in Table 4.2. The EIQ converges much faster than the conventional quadrature.

Let us now change the problem. Assume we want to approximate the mean for a large number of new parameters with a maximum error tolerance of 10^{-6}.

Table 4.2: Maximum quadrature error for the mean of a β-distribution over 1000 random (α, β) pairs.

M	e_{EI}	e_{GLL}
5	4.093635e-02	1.803959e-01
10	3.084039e-03	3.206775e-02
20	2.377752e-05	1.222629e-02
30	7.814867e-06	6.231308e-03
40	1.968040e-07	3.708694e-03

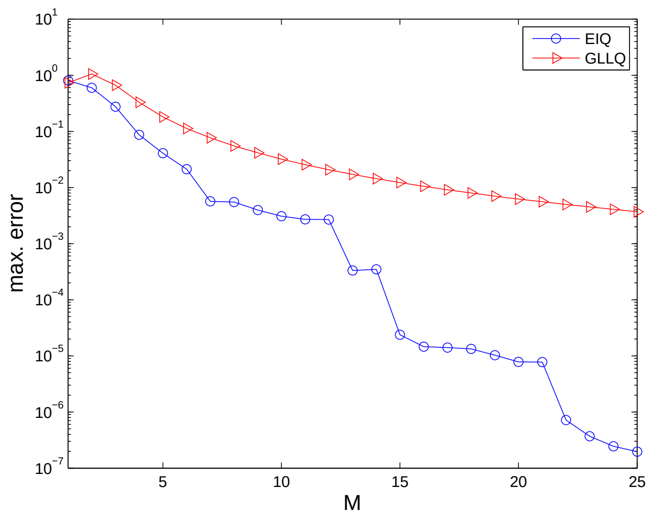


Figure 4.2: Maximum quadrature error for the mean of a β-distribution over 1000 random (α, β) pairs.

With the EI method, this would require $M = 22$ EI nodes. To achieve a similar tolerance using GLL quadrature would require approximately 2500 points. This number is estimated from the convergence rate in Figure 4.2.

If we need the mean for say 100 new parameter values, this is achieved in about 0.012 seconds using EI quadrature. The same results obtained by GLL quadrature requires approximately 0.15 seconds. Hence, for this example the EI quadrature is roughly 12-13 times faster than GLL quadrature when we only consider online computations.

4.4.3 Weighing a pyramid

To demonstrate the increased effect of the EI method as the number of dimensions grow, we conclude this chapter with a three-dimensional example.

Consider a square-based pyramid with the following weight distribution

$$d(x, y, z; \mu) = \frac{\mathrm{e}^{-\mu(x^2+y^2)}}{1+z}. \tag{4.7}$$

We have parametrically dependent exponential increase, or decrease, in the density in the radial direction and linear decrease in the vertical direction. We look at parameters in the region $\mathcal{D} = [-1, 1]$. The spatial domain is given by the base $(x, y) \in [-1, 1]^2$ and apex in $(x, y, z) = (0, 0, 2)$. A graphical presentation of the weight distribution and pyramid for $\mu = 1$ is given in Figure 4.3.

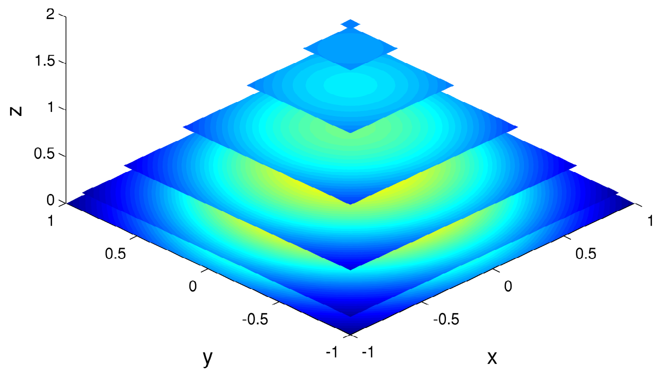


Figure 4.3: Example of pyramid with the density function given in (4.7). The density is increasing from dark to bright. Here $\mu = 1$.

Assume now that we are interested in the total mass of the pyramid for the entire parametric domain. Given the density, the mass $m(\mu)$, can be calculated as

$$\begin{aligned} m(\mu) &= \iiint_\Omega d(x,y,z;\mu)\,\mathrm{d}\Omega \\ &= \int_0^2 \int_{-1+\frac{z}{2}}^{1-\frac{z}{2}} \int_{-1+\frac{z}{2}}^{1-\frac{z}{2}} \frac{\mathrm{e}^{-\mu(x^2+y^2)}}{1+z}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \end{aligned} \tag{4.8}$$

The straight-forward approach to evaluate (4.8) is to discretize the parameter domain and compute the approximate mass for each parameter value using three-dimensional GLL quadrature. If we for simplicity assume the same number of GLL points in each vertical layer, this approximation becomes

$$m(\mu) \approx m_{\mathcal{N}}(\mu) = \sum_{\alpha=0}^{\mathcal{N}} \sum_{\beta=0}^{\mathcal{N}} \sum_{\gamma=0}^{\mathcal{N}} \rho_\alpha \rho_\beta \rho_\gamma \left(\frac{2-\xi_\alpha}{2}\right)^2 d(\xi_\alpha, \xi_\beta, \xi_\gamma; \mu). \tag{4.9}$$

Here we have to take into consideration the decreasing cross-section of the pyramid as the height increases. The complexity for (4.9) is $\mathcal{O}(\mathcal{N}^3)$ for each parameter.

We have a density that is very smooth in the parameter, and we thus expect rapid convergence with the EI method. For this example it is sufficient with only 13 EI snapshots to achieve machine precision accuracy. If we use a spatial grid of 35 GLL points in each direction, and discretize $\mathcal{D}$ using 1000 uniformly distributed points the online EI procedure is computed for all the parameter values in under 0.05 seconds. The brute force GLL approach requires almost 6.9 seconds. The EI method is thus approximately 140 times faster. This is a dramatic speedup.

For more complex density functions and domains we would expect even better results. GLL quadrature requires knowledge of d in $\mathcal{N}^3$ points. The EI quadrature however, uses only M points, even in three-dimensions. A more complex density would thus favor the EI quadrature.

Chapter 5

Nonlinear problems

To complete our investigation of the properties of the empirical interpolation method we apply it to the solution of nonlinear problems. We present two examples later in the chapter, but first we make some remarks regarding the greedy algorithm in this type of setting.

5.1 Greedy algorithm for nonlinear problems

In principle, no modification of Algorithm 3.1 is needed for nonlinear applications. However, where we in our previous examples only required a function evaluation to obtain f, we now need to solve a nonlinear problem, e.g. a nonlinear differential equation. The amount of offline work is thus even greater in this setting, especially as we want the offline solutions to be very accurate.

In previous versions of the algorithm [6], where the next basis element was determined by maximizing the projection error, this offline cost became prohibitively large. This was first solved by substituting the infinity norm with the L^2 norm [7]. A better solution, which is also the approach we have adopted, is to use the interpolation error instead of the projection error [11]. This enables us to compute the error relatively quickly, even though we have to solve the nonlinear system for each of the parameters in our discrete training sample.

5.2 Numerical examples

We here present two examples where the EI algorithm is applied to nonlinear problems. First, a simple one dimensional initial-value problem. Second, we introduce EI in a setting where a large number of evaluations is necessary, namely in combination with the shooting method [16].

5.2.1 Simple initial value problem

Consider the simple, nonlinear initial-value problem (IVP) given by

$$y'(x) = (\mu - 2x)y^2, \quad y(x_0) = y_0. \tag{5.1}$$

Given the initial condition $y(x_0) = y_0$ this can be solved easily using a number of numerical techniques. However, assume that we *a priori* only know a range for the parameter, i.e. $a \leq \mu \leq b$ for some constants a and b. Also, we are interested in the solution up to some point $x_N > x_0$.

We can use the empirical interpolation technique to express the solution as an affine parameter dependent expansion. This will enable us to quickly find an approximation of $y(x)$ for a given parameter value μ. To see this, note that

$$y(x;\mu) \approx y_M(x;\mu) = \sum_{i=1}^{M} \varphi_i(\mu) q_i(x).$$

Again, the parameter independent values $q_i(x_M)$, $i = 1, \ldots, M$ can be computed offline. The coefficients $\varphi_i(\mu)$ are still given as

$$\sum_{j=1}^{M} (B_M)_{ji} \varphi_j(\mu) = y(x_i;\mu), \quad i = 1, \ldots, M.$$

In our previous examples the right-hand side was simply a function which we could easily evaluate in $\mathcal{O}(1)$ operations for a given μ. Now, the situation is quite different. We need the solution to the original problem in each of the EI nodes. Computing the entire solution is not an option as this is what we are trying to avoid. We can however use a Runge-Kutta (RK) scheme to obtain the solution only in the EI nodes. This will introduce additional error, but if the tolerance requirements are not to strict it may still be a favorable option.

It is straightforward to demonstrate that the solution of (5.1) for the initial condition $y(0) = 1$ is

$$y(x;\mu) = \frac{1}{x^2 - \mu x + 1}. \tag{5.2}$$

For our numerical tests, we look at x in the interval $(0, 5)$ and parameters $\mu \in \mathcal{D} = (0, 1)$. With this $\mathcal{D}$ we avoid singularities due to the denominator being zero. This would for instance happen for $\mu = 2$ at $x = 1$.

The numerical forward integration of our IVP (5.1) is done with the implicit midpoint method (IMM) as this is stable also for nonlinear problems [27]. This method determines the next step $y_{n+1} \approx y(x_{n+1})$ for the problem $y'(x) = g(x, y)$ by

$$\begin{aligned} y_{n+1} &= y_n + hk_1, \\ k_1 &= g(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \end{aligned} \tag{5.3}$$

where h is the step length. We thus need to solve a nonlinear equation at each iteration to find k_1. The forward integration, (5.3), is done offline on a fine grid with $\mathcal{N} = 250$ points to compute the basis elements q_i, $i = 1, \ldots, M$ and interpolation matrix B_M.

Online, we use IMM to obtain the solution in the EI nodes and compute the coefficients $\varphi_i(\mu)$. As the number of forward RK steps is only M this will be much faster than solving the entire system of $\mathcal{N}$ steps.

If we are interested in the solution on the entire interval, we will still have $\mathcal{N}$ dependence as the basis functions are represented using $\mathcal{N}$-dimensional vectors. However, if we are only interested in the value at the endpoint, $x_N = 5$, this only requires $\mathcal{O}(M)$ operations, and we achieve a significant reduction in computational effort.

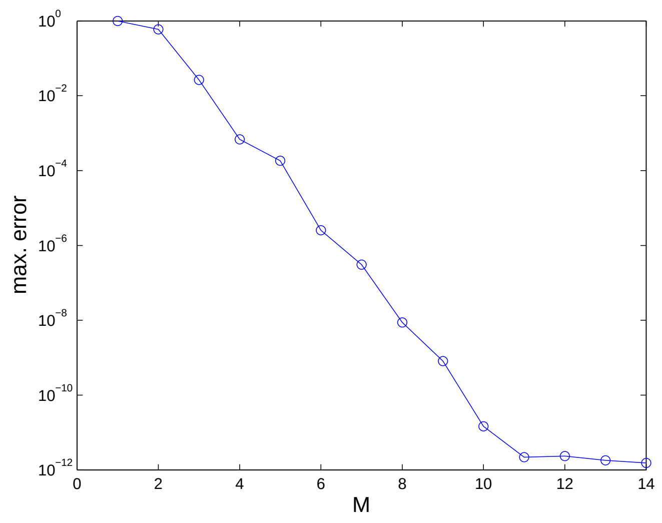


Figure 5.1: Maximum interpolation error for the nonlinear IVP $y'(x) = (\mu - 2x)y^2$, $y(0) = 1$.

Table 5.1: Maximum error in the endpoint $x = 5$ over a test set of 100 random parameter values. We achive offline solution accuracy ($< 10^{-5}$) with only 6 EI points.

M	e_M
1	9.9674e-3
2	6.8572e-4
3	2.8019e-4
4	4.9481e-5
5	4.3953e-5
6	9.9520e-6
7	1.0878e-5
8	9.2305e-6
9	9.3303e-6
10	6.7700e-6

The maximum interpolation error for the first 14 EI samples is shown in Figure 5.1. Again, the convergence is exponential. The maximum endpoint error for a test set of 100 random parameter values is given in Table 5.1. If we compare the results for the endpoint error with the interpolation error from the EI procedure, we see that the convergence is very rapid. We reach the EI error (10^-5) with only 6 samples. As this is the tolerance for the offline solution of the IVP we can not expect a smaller error in the EI approximation.

We see that for this example, the error will be dominated by the offline error at only $M = 10$ EI iterations. If we solve (5.1) for 100 new parameter values, using 10 terms in our EI expansion, this is achieved in about 9 seconds. Solving the full system for all 100 parameters for comparison takes about 271 seconds. This implies that the online EI calculation is roughly 30 times faster than the brute force approach. Also, the loss in accuracy is minimal.

In Figure 5.2 we see that for only 10 EI samples we cannot distinguish the EI approximation from exact solution for the case $\mu = 0.5$. However, if we include too many samples, the final samples will not really contribute to reducing the error, but may cause some instability in the results.

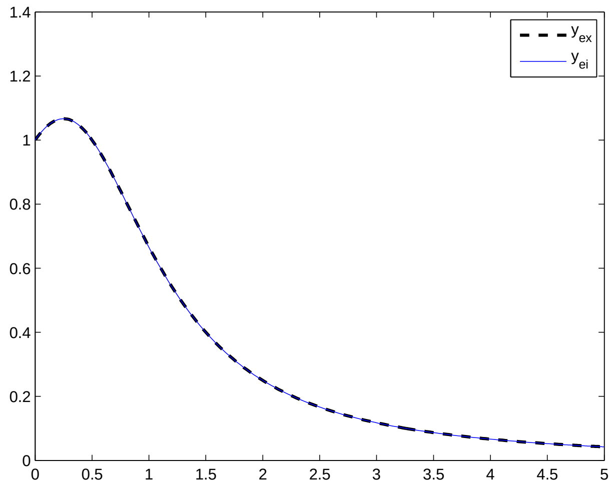


Figure 5.2: Comparison beteween EI approximation and exact solution for the IVP in (5.1) for $\mu = 0.5$. With only 10 EI samples the two functions are practically indistiguishable.

5.2.2 The Empirical Shooting Method (ESM)

A common, and often effective, method used for solving boundary-value problems (BVPs) is the shooting method [16, 28, 29]. This works by transforming the BVP into an initial value problem which is much easier to handle. We here give the general procedure and apply it to a small example. Consider the

Table 5.2: Initial endpoint values computed for the IVP (5.7). The values indicate solutions ($y(1; \mu) = 1$) in the intervals $(-40, -30)$ and $(-10, -1)$.

μ	-1	-10	-20	-30	-40
$y(1; \mu)$	59.4134	-2.4008	-4.8370	-1.4668	2.8610

nonlinear second-order BVP

$$y''(x) = g(x, y, y'), \quad a \le x \le b, \quad y(a) = y_a, \quad y(b) = y_b. \tag{5.4}$$

The idea behind the shooting method (SM) is to approximate the solution of (5.4) by a sequence of IVPs such that

$$y''(x; \mu) = g(x, y, y'), \quad a \le x \le b, \quad y(a) = y_a, \quad y'(a) = \mu_k. \tag{5.5}$$

The parameters μ_k are chosen such that $\lim_{k\to\infty} y(b; \mu_k) = y(b)$. The name of the method thus relates to the analogy of shooting at a stationary target. The barrel has a fixed initial height, but we are allowed to change the trajectory by adjusting the initial firing angle.

For nonlinear problems there are a number of ways to compute this sequence of parameters, and the choice of a good method will be problem dependent. However, what is certain is that we need to solve the IVP (5.5) for a large number of parameters.

Let us now look at an example. Consider the second-order nonlinear BVP

$$y''(x) = \frac{3}{2}y^2, \quad 0 \le x \le 1, \quad y(0) = 4, \quad y(1) = 1. \tag{5.6}$$

Equation (5.6) can be transformed into a first-order IVP in two variables given by

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} v \\ \frac{3}{2}u \end{pmatrix}, \quad u(0) = 4, \quad v(0) = \mu. \tag{5.7}$$

Here we have introduced the new variables $u = y$ and $v = y'$. Again the forward integration is easily performed using any desired RK method.

From the differential equation we see that the curvature is always positive. As the boundary-value at $x = 1$ is lower than the boundary-value at $x = 0$, the firing angle μ have to be negative. A few initial "shots" give the values in Table 5.2. For $\mu < -40$ we get increasing end-values. This indicates parameter values in the intervals $(-10, -1)$ and $(-40, -30)$.

To avoid two separate EI expansions we choose the parameter space $\mathcal{D} = (-40, -1)$. We use an offline resolution of $\mathcal{N} = 1500$ points in the RK solutions. In Figure 5.3 we see that we still have rapid exponential convergence.

To obtain rough estimates of the correct initial parameters we compute the solution of (5.7) for $\mu = -40, -39, \ldots, -1$. We can then use Newton's method, with the closest integer solution as initial guess, to obtain accurate solutions. That is, we compute the sequence of parameters by

$$\mu_{k+1} = \mu_k - \frac{y(1; \mu_k) - y(1)}{y'(1; \mu_k)}.$$

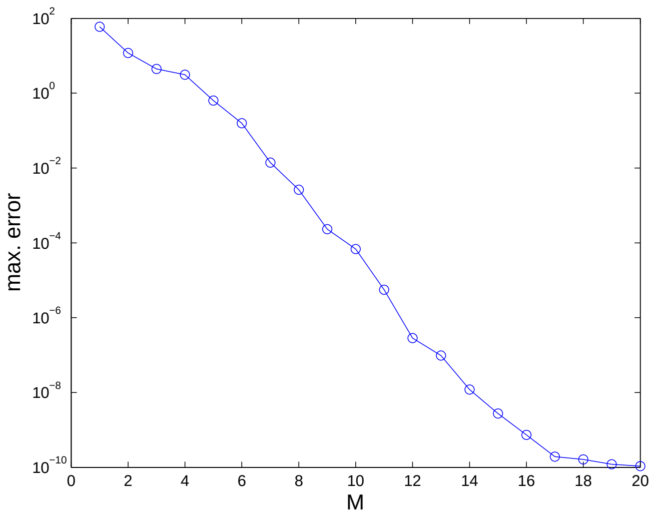


Figure 5.3: Maximum interpolation error for the nonlinear IVP $y''(x) = \frac{3}{2}y^2$, $y(0) = 4$, $y'(0) = \mu$.

If we introduce u and v from (5.7) and y(1)=1 this can be written as

$$\mu_{k+1} = \mu_k - \frac{u(1;\mu_k) - 1}{v(1;\mu_k)}.$$

From Figure 5.4 we get the starting values $\mu_0^1 = -36$ and $\mu_0^2 = -8$. The second value, $\mu_0^2 = -8$, is actually one of the solutions. We thus only need to perform the iterative procedure for the first initial value.

When we are sufficiently close to the solution, the Newton iteration scheme converges very fast. This however, requires exact knowledge of the function and its derivative. In this case we only have approximate values as we compute the u and v on a relatively coarse grid. With 1200 iterations the error is almost down to 10^{-8} as seen in Table 5.3.

The fast online evaluation with the EI method enables us to perform this iteration in less than 0.66 seconds. Using the RK solution of the full system for each Newton iteration is much slower and is completed in roughly 38.4 seconds. Even if we include the offline EI time of about 18.3 seconds, the ESM method is faster. That the ESM approach is about twice as fast is expected. We do 15 iterations over a test set of 40 values. This involves the RK solution of 600 full systems, that is half of the number required in the Newton iteration. The solution to the initial BVP in (5.4) is thus computed twice as fast with ESM as the with the standard shooting method with the tolerance requirement of 10^{-8}.

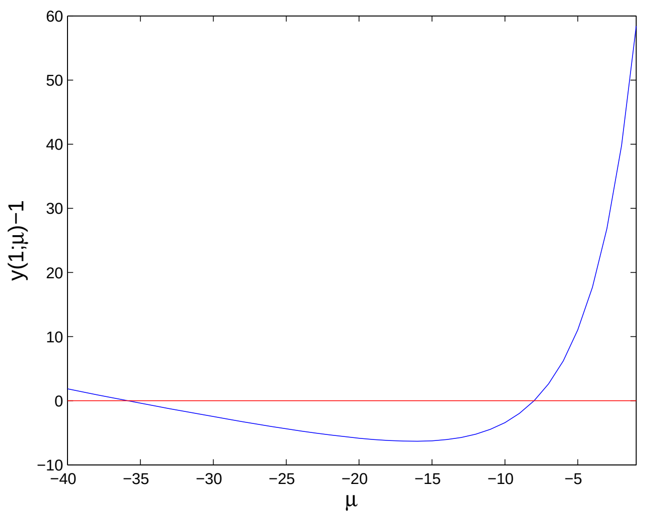


Figure 5.4: Value at endpoint, $y(1;\mu)$, for IVP (5.7) calculated for $\mu = -40, \dots, -1$. The closest integer solutions are $\mu_0^1 = -36$ and $\mu_0^2 = -8$.

Table 5.3: Endpoint error for the Newton iteration.

Iterations	Endpoint error
100	1.009e-2
200	2.877e-3
300	8.202e-4
400	2.338e-4
500	6.667e-5
600	1.901e-5
700	5.418e-6
800	1.545e-6
900	4.404e-7
1000	1.256e-7
1100	3.579e-8
1200	1.020e-8

Part II

Chapter 6

An optimization problem

So far we have successfully applied the empirical interpolation method to achieve both efficient numerical quadrature and solution to nonlinear differential equations. The rapid evaluation was made possible by the online/offline computational strategy where the majority of the work is performed in a parameter independent preprocessing stage. In addition, the construction of a triangular interpolation matrix allowed us to perform the online stage in $\mathcal{O}(M^2)$ operations independent of the offline complexity.

EI is however not the only method which takes advantage of the online/offline approach. In the remainder of this report, our main focus will be on another of these methods, the reduced basis (RB) approximation technique. In many ways it is the predecessor of empirical interpolation as it was the inability of the RB method to handle non-affine problems which motivated the development of this interpolation scheme.

To present the RB method we shift our attention to partial differential equations, more specifically the Laplace equation. We also introduce parameter dependence, and an output related to the solution of the Laplace problem.

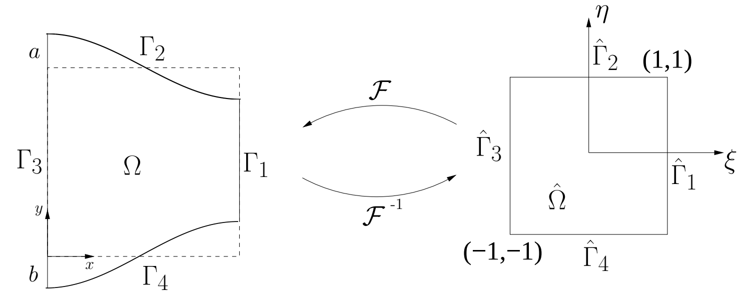


Figure 6.1: Problem setup for the optimal flow problem. The parameters a and b can be changed independently. For the undeformed case (- - -) Ω is simply $(0,1)^2$.

Imagine the following situation: Given an amount of material, e.g. to build a "pipe" in two dimensions, what shape of the pipe will minimize resistance, or pressure drop from, the inlet to the outlet for a given flow through Γ_3? This problem can be modeled as the solution of the Laplace equation in Ω with Γ_3 as the inlet, and Γ_1 as the outlet (see Figure 6.1). We also get an output of interest as the mean value of u over Γ_3. The intuitive solution is that the minimal resistance occurs for the undeformed pipe, but nevertheless it may serve as a model problem for our numerical analysis.

We are interested in the pressure drop per unit length in the x-direction. For easier comparison between solutions, the length of Ω in the x-direction is thus always equal to 1. To test different pipe configurations we introduce variation on the upper and lower boundary, here interpreted as the pipe walls. We let each of the walls independently admit a deformation corresponding to a half cosine, where the degree of deformation is given by two amplitude parameters, a and b. Here we get a parameter dependent partial differential equation, or μ-PDE. In our case the parameter vector $\boldsymbol{\mu} = (a, b)^T$.

Now, suppose we want to determine the optimal parameter values giving the lowest possible pressure drop. For this problem we would expect a high degree of regularity. An efficient method for solving the problem, given a single value of the parameter $\boldsymbol{\mu}$, is the spectral method, which we introduce in Chapter 7.

Again, one possible strategy for the optimization problem would be to construct an EI basis space for the solution offline. Then, for a new parameter we assemble the interpolant to obtain the solution. This approach has one major problem. We need the solution to the underlying problem to compute the EI coefficients, as we have already discussed in the previous chapter. For those examples, which were solved using RK methods, we solved this by using only the EI interpolation points in the solution. With the spectral method, this is not possible, and we need a different approach.

In the next chapter, we present the reduced basis approximation framework which is tailored for this context. The solutions corresponding to a large number of different parameters need to be evaluated in a short amount of time. We can thus afford a computationally expensive offline stage if the online stage is very fast.

The RB approach is very similar to the EI method, but also different. Both methods build up an approximation space containing "snapshot" solutions, but instead of interpolation coefficients we find parameter dependent coefficients via a Galerkin procedure. We thus do not need the solution to compute the approximation, as is required for the interpolation strategy.

Chapter 7

The Spectral Method (SM)

For the problem presented in the previous chapter we expect a smooth solution. We could of course solve the problem, for a given set of parameters, with a standard low-order finite element approximation. However, when the solution admits a high degree of regularity, this may be exploited.

Here we present an approach where the rate of convergence will in fact depend on the regularity of the underlying exact solution: the spectral method [30, 31, 32]. To illustrate this method, we formulate and discretize a mixed boundary condition Poisson problem. In addition, we solve this problem on a deformed geometry, as this is what we will need to handle the optimization problem from Chapter 6.

An efficient solution of a deformed Poisson problem with the spectral method involves a number of different elements such as grid-generation, a specialized fast algorithm for the undeformed problem, an iterative solution procedure, in our case the conjugate gradient algorithm (CG) with preconditioning and a global numbering scheme for the unknowns. We will not discuss all these issues in detail, and for a complete and thorough presentation the reader is referred to [33].

7.1 The Poisson problem in a general domain

The abstract weak formulation of the Poisson problem with mixed boundary conditions is still: Find u in X^e such that

$$a(u,v) = l(v), \quad \forall v \in X^e, \tag{7.1}$$

where $a(\cdot,\cdot)$ and $l(\cdot)$ are as defined in Section 2.2. However, for the actual computation of the linear and bilinear forms we want to use GLL quadrature. This is defined over the square $\hat{\Omega} = (-1,1)^2$. We thus need to perform a transformation of our problem from the physical domain Ω to a *reference domain* $\hat{\Omega}$, as depicted in Figure 7.1. The mapping between $\hat{\Omega}$ and Ω we will denote by $\mathcal{F}$, and its inverse by $\mathcal{F}^{-1}$.

First note that generally $\mathcal{F}$ can be defined by the relations

$$\begin{aligned} x &= x(\xi,\eta), \\ y &= y(\xi,\eta). \end{aligned}$$

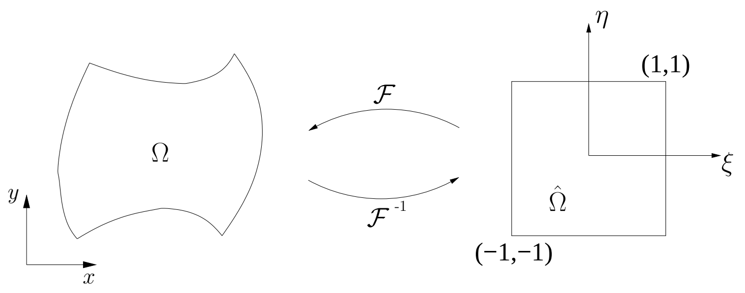


Figure 7.1: Illustration of a deformed domain Ω. We use the reference domain $\hat{\Omega} = (-1,1)^2$ with $\mathcal{F}$ mapping $\hat{\Omega}$ onto Ω.

We will discuss a method of realizing these relations in Section 7.3.1, but for the moment assume them to be known. Let us now define the Jacobian matrix J as

$$J = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix}. \tag{7.2}$$

Using J we can express the differentials $\mathrm{d}x$ and $\mathrm{d}y$ in terms of the reference differentials $\mathrm{d}\xi$ and $\mathrm{d}\eta$.

$$\begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix} = J \begin{pmatrix} \mathrm{d}\xi \\ \mathrm{d}\eta \end{pmatrix}.$$

To simplify our expressions we introduce the common notation $\frac{\partial v}{\partial x} = v_x$ so that $\nabla v = \binom{v_x}{v_y}$. Also, if we denote $\det(J)$ by $\mathcal{J}$, we can define the geometry matrix

$$G = \frac{1}{\mathcal{J}^2} \begin{pmatrix} y_\eta^2 + x_\eta^2 & -y_\xi y_\eta - x_\xi x_\eta \\ -y_\xi y_\eta - x_\xi x_\eta & y_\xi^2 + x_\xi^2 \end{pmatrix}. \tag{7.3}$$

This is a symmetric positive definite (SPD) 2×2 matrix [33]. Using (7.2) and (7.3) the weak form given in (7.1) becomes: Find $u \in X^e$ such that

$$\int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \tilde{G} \hat{\nabla}\hat{u} \, \mathrm{d}\xi \mathrm{d}\eta = \int_{\hat{\Omega}} \hat{f}\hat{v}\mathcal{J} \, \mathrm{d}\xi\mathrm{d}\eta + \int_{\hat{\Gamma}_3} \hat{g}\hat{v}\mathcal{J}^S \, \mathrm{d}\eta, \quad \forall v \in X^e \tag{7.4}$$

for $\tilde{G} = \mathcal{J}G$ and $\mathcal{J}^S = \frac{\partial y}{\partial \eta}\big|_{\Gamma_3}$.

7.2 Discretization and algebraic equations

The discretization procedure for the spectral method is very similar to standard finite element discretizations. We will only give a brief discussion of the computations and present the final result. We will approximate the solution as a high-order polynomial on the reference domain, i.e. the discrete space $X_\mathcal{P}$ is defined as

$$X_\mathcal{P} = \{v \in X^e, \, \hat{v} \in \mathbb{P}_\mathcal{P}(\hat{\Omega})\}. \tag{7.5}$$

Here, $\mathbb{P}_\mathcal{P}$ denotes the space of polynomials of degree $\mathcal{P}$ or less in two variables. Note that dependent on the mapping $\mathcal{F}$, v will in general not be a polynomial on the physical domain Ω.

Introducing the discrete solution $u_\mathcal{P}$ gives us the discrete form of (7.4) given as: Find $u_\mathcal{P} \in X_\mathcal{P}$ such that

$$\int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \tilde{G} \hat{\nabla} \hat{u}_\mathcal{P} \, \mathrm{d}\xi \mathrm{d}\eta = \int_{\hat{\Omega}} \hat{f} \hat{v} \mathcal{J} \, \mathrm{d}\xi \mathrm{d}\eta + \int_{\hat{\Gamma}_3} \hat{g} \hat{v} \mathcal{J}^S \, \mathrm{d}\eta, \quad \forall v \in X_\mathcal{P}. \tag{7.6}$$

We now use GLL quadrature to approximate these integrals and choose the test functions $v \in X_\mathcal{P}$ as the product of two Lagrange interpolation polynomials, ensuring a nodal basis, or $\hat{v}(\xi, \eta) = \ell_i(\xi)\ell_j(\eta)$ for $i = 1, \dots, \mathcal{P}, j = 0, \dots, \mathcal{P}$. This gives us the approximation for the bilinear form $a(\cdot,\cdot)$ and the right-hand side $l(\cdot)$ as

$$a_\mathcal{P}(u_\mathcal{P}, v) = \sum_{\alpha=0}^{\mathcal{P}} \sum_{\beta=0}^{\mathcal{P}} \rho_\alpha \rho_\beta \underbrace{\begin{pmatrix} D_{\alpha i}\delta_{\beta j} \\ \delta_{\alpha i} D_{\beta j} \end{pmatrix}^T}_{(\hat{\nabla}\hat{v})^T} \underbrace{\begin{pmatrix} \tilde{g}_{11} & \tilde{g}_{12} \\ \tilde{g}_{21} & \tilde{g}_{22} \end{pmatrix}_{\alpha\beta}}_{\tilde{G}} \underbrace{\begin{pmatrix} u_{\mathcal{P},\xi} \\ u_{\mathcal{P},\eta} \end{pmatrix}_{(\xi_\alpha, \xi_\beta)}}_{\hat{\nabla}\hat{u}_\mathcal{P}} \tag{7.7}$$

and

$$\begin{aligned} l(v)_\mathcal{P} &= \sum_{\alpha=0}^{\mathcal{P}} \sum_{\beta=0}^{\mathcal{P}} \rho_\alpha \rho_\beta f_{\alpha\beta} v_{\alpha\beta} \mathcal{J}_{\alpha\beta} + \sum_{\alpha=0}^{\mathcal{P}} \rho_\alpha g_\alpha v_\alpha \mathcal{J}_\alpha^S \\ &= (\rho_i \rho_j J_{ij}) f_{ij} + \rho_i g_i \mathcal{J}_i^S, \quad \forall\, i, j. \end{aligned} \tag{7.8}$$

Expression (7.7) could, using global representation of our unknowns, be written as a matrix-vector product $A^{2D}\boldsymbol{u}$, with $A^{2D} \in \mathbb{R}^{(\mathcal{P}-1)\mathcal{P} \times (\mathcal{P}-1)\mathcal{P}}$ and $\boldsymbol{u} \in \mathbb{R}^{(\mathcal{P}-1)\mathcal{P}}$, which represents the action of the discrete two-dimensional Laplace operator upon $u_\mathcal{P}$. Performing this matrix-vector product in the standard way would have a computational complexity of $\mathcal{O}(\mathcal{P}^4)$, but using the above expression along with a local data representation, we are able to achieve the same result in $\mathcal{O}(\mathcal{P}^3)$ operations. We will return to this in Section 7.3.2 where we analyze the complexity of solving this system using the conjugate gradient algorithm.

As $X_\mathcal{P} \subset X^e$ we know, by Galerkin orthogonality [15], that the approximation $u_\mathcal{P}$ will be the best possible approximation of u in $X_\mathcal{P}$, measured in the energy norm. Combining this with the coercivity and continuity of $a(\cdot,\cdot)$ we can derive what is known as *Céa's* lemma [31]

$$\|u - u_\mathcal{P}\|_{H^1(\Omega)} \le \frac{\beta}{\alpha} \inf_{v \in X_\mathcal{P}} \|u - v\|_{H^1(\Omega)}. \tag{7.9}$$

If we now choose v to be a high-order interpolant of u, and use the error estimate from Section 2.5, the *a priori* error estimate for $e_\mathcal{P} = u - u_\mathcal{P}$ becomes

$$\|e_\mathcal{P}\|_{H^1(\Omega)} \le c\mathcal{P}^{1-\sigma} \|e_\mathcal{P}\|_{H^\sigma(\Omega}, \tag{7.10}$$

for $u \in H^\sigma(\Omega)$, where c is a constant. It can in fact be shown [34] that for u analytic, i.e. $\sigma \to \infty$, we get exponential convergence.

7.3 Solution procedure

The solution of the deformed Poisson problem is somewhat involved. However, the procedure is completed in two major steps. Computing the mapping $\mathcal{F}$ and solving the resulting algebraic equations using CG.

7.3.1 Grid genereation – The Gordon-Hall algorithm

In 1973 William Gordon and Charles Hall presented a simple, yet effective, way of generating a grid for a general domain [35]. Here we will only give a short description of the key aspects of the algorithm and demonstrate the different steps using a small graphical example.

The Gordon–Hall algorithm is actually a way of distributing the grid-points in the interior of the computational domain. The boundaries must be given by the user. First we need to distribute our boundary nodes along each edge, including the corners. The idea is now to match the edges in pairs by drawing straight lines between corresponding nodes and finally subtracting a matching of the four corners. This procedure is much easier to present graphically which we do in Figure 7.2.

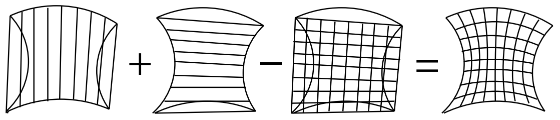


Figure 7.2: Visualization of the Gordon–Hall steps. The final grid is created by connecting the remaining intersections.

We will use this approach to generate a representation of our mapping $\mathcal{F}$ taking us from reference to physical variables. To do this we distribute GLL points along each edge according to a chordial distribution. Another alternative would be to distribute the points according to the arch length. When the GLL points are distributed, we can create high-order interpolant approximations to each edge. These interpolants are what we use for computations.

We have used the same order for interpolating $\mathcal{F}$ and computing $u_{\mathcal{P}}$. This is called an *isoparametric* representation of the geometry [15].

This algorithm may fail to return a unique one-to-one mapping if the deformation becomes too severe, e.g. if one of the internal angles approaches 180°. Generally it is not common to use a single domain to represent the geometry, but we can still use a spectral discretization within each subdomain. This is known as the *spectral element method* [36], and it has been demonstrated that the a priori error estimates from Section 7.2 are also valid for this method [37].

7.3.2 Iterative solution with Conjugate Gradients

Since our discrete Laplace operator given in (7.7) is symmetric and positive definite, we can solve the system of equations using CG. The standard implementation of this algorithm is dominated by a matrix-vector product [38, 39, 40].

If we were to explicitly form the two-dimensional, discrete Laplace operator A^{2D}, this operation would require $\mathcal{O}(\mathcal{P}^4)$ operations as mentioned earlier. However, all that is really needed in the algorithm is the ability to evaluate this product. Consider now a function $\mathcal{A} : \mathbb{R}^{\mathcal{P}\times(\mathcal{P}+1)} \to \mathbb{R}^{\mathcal{P}\times(\mathcal{P}+1)}$ such that $\mathcal{A}(W)$ contains the same information as $A^{2D}w$, for all $w \in \mathbb{R}^{\mathcal{P}(\mathcal{P}+1)}$, only we use a local instead of a global numbering. If we exploit this local numbering and use one-dimensional matrix-matrix products, the evaluation of $\mathcal{A}(w)$ can be performed in $\mathcal{O}(\mathcal{P}^3)$ operations using a tensor product sum-factorization technique [41].

Another advantage of this approach is the reduction in memory requirement from $\mathcal{O}(\mathcal{P}^4)$ to $\mathcal{O}(\mathcal{P}^2)$. If we denote the number of CG iterations by n_{it} we get a total computational complexity of $\mathcal{O}(n_{it}\mathcal{P}^3)$ for $\mathcal{P}(\mathcal{P}+1)$ unknowns.

It is also possible to reduce the number of iterations by preconditioning. For the undeformed, rectangular case we can construct a tailored algorithm with the same complexity as one iteration of the CG iteration. This is based on tensor product representation of A^{2D} and the solution of a generalized eigenvalue problem [32]. The solution to the undeformed problem can then be used as a preconditioner for the deformed solution.

7.4 Numerical example

We now solve the parametric PDE introduced in Chapter 6 for a set of parameters picked at random from the parameter domain $\mathcal{D}$. With this example, the rapid convergence of the Spectral Method for a problem with high regularity is verified.

For this problem, we have $f(x, y) = 0$. The right-hand side, (7.8), will thus only contain the term from the Neumann boundary condition. We choose g such that the flow through Γ_3 is equal for all values of $\boldsymbol{\mu}$. This implies $g = 1/|\Gamma_3|$.

Figure 7.3 show the spectral solution for $\boldsymbol{\mu} = (-0.311, 0.098)$ with a polynomial degree of $\mathcal{P} = 25$.

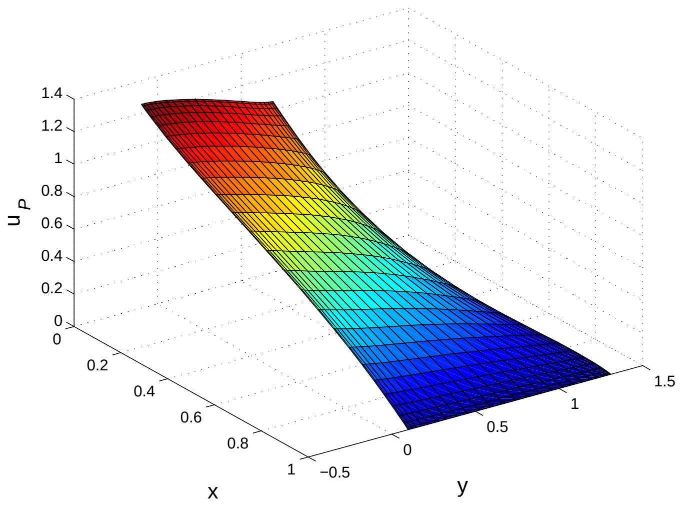


Figure 7.3: Spectral solution of optimization PDE with $\boldsymbol{\mu} = (-0.311, 0.098)$. The polynomial degree is $\mathcal{P} = 25$.

In addition, the convergence is given in Figure 7.4. For the L^2-error and H^1-error we have approximately

$$\|u - u_{\mathcal{P}}\|_{L^2(\Omega)} \sim \mathcal{P}^{-6},$$
$$\|u - u_{\mathcal{P}}\|_{H^1(\Omega)} \sim \mathcal{P}^{-5}.$$

If we compare this to the error estimate given in Section 7.2, this indicates that the exact solution u is an element of $H^6(\Omega)$, and the convergence is very fast.

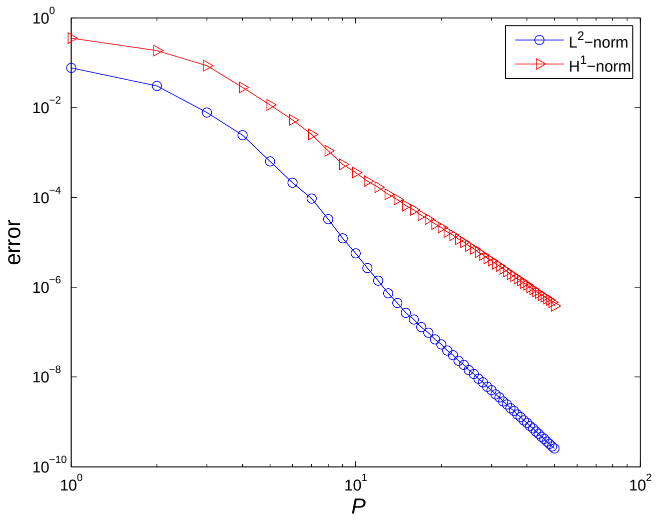


Figure 7.4: Convergence of the Spectral Method for $\boldsymbol{\mu} = (-0.311, 0.098)$.

In the previous section, we demonstrated that the complexity was $\mathcal{O}(n_{it}\mathcal{P}^3)$ for n_{it} CG iterations for this example. For a small number of parameter values, computing the spectral approximation for each value is a feasible option. Imagine now that we need the solution for a hundred, or even a thousand times as many values. And in each case we require a strict tolerance level to be met.

This will be the case if we are to solve the actual optimization problem of finding the optimal values for a and b which minimize the pressure drop. Faced with such a problem, computing the full spectral approximation for each choice of a and b will be very time consuming.

For this we will use the reduced basis methodology, mentioned in Chapter 6. In the next chapter we present this approach in detail, and finally use the framework to solve the optimization problem with a rapid online computational procedure.

Chapter 8

Reduced Basis approximations

The problem posed in Chapter 6, along with many other problems, belongs to the class of so-called parametrized partial differential equations, or μ-PDEs for short. Typically, the problem will be dependent on a parameter vector $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^p$, and there is a corresponding output of interest $s(\boldsymbol{\mu})$. Here, $\mathcal{D}$ is the space of possible parameter values. Of course, the solution will also be parameter dependent, hence we write $u(x, y) = u(x, y; \boldsymbol{\mu})$. This again implies that if we want the solution for many parameter values, we must solve the problem equally many times. For a large real-world simulation, this may not be a feasible option, and we thus need a different approach.

The parametric dependence of the solution is often quite regular. Hence we expect that a set of solutions for parameters *sampled* at a few carefully chosen points in the parameter domain $\mathcal{D}$, will give a good representation of the solution for an arbitrary new parameter value. This is indeed the motivation for the reduced basis methodology, where for a new parameter the problem is reduced to finding the optimal linear combination of previously computed solutions.

Obviously, this is not suited for a single query setting as it requires the solution to several full-scale problems. However, if we are in the many queries context on an optimization problem, or in any other situation depending on fast evaluation for a large number of new parameters, we will see that the additional initial work is well justified.

8.1 Important concepts

Here we introduce some of the concepts needed to describe and understand the reduced basis framework.

8.1.1 Truth solution

As we shall see, our reduced basis approximation space will be built up by so called *truth* solutions. Now, each truth solution will in turn be a member of a high order approximation space. Here we denote this space by $X^{\mathcal{N}} \subset X^e$, where $\mathcal{N} = \dim(X^{\mathcal{N}})$, and X^e is the exact solution space to the continuous problem introduced in Section 2.2. Note that $X^{\mathcal{N}} = X_{\mathcal{P}}$ from the previous chapter. The actual derivation of these solutions can be based upon a number of different approximation techniques [1], such as finite element/volume, spectral element or pure spectral, as will be used here.

It is now important that $\mathcal{N}$ is chosen large enough. We need to ensure that the error in going from X^e to $X^{\mathcal{N}}$ is smaller than the RB error, and also that we achieve the desired accuracy in our output. As a consequence of this, it is essential to develop a computational procedure where the solution for a new parameter $\boldsymbol{\mu}$ can be obtained independent of $\mathcal{N}$. We will return to this online/offline strategy in Section 8.3.4.

8.1.2 Parameter independent inner products and norms

Associated with our exact solution space X^e we may use the usual energy inner product and induced norm given as

$$(((w,v)))_{\boldsymbol{\mu}} = a(w,v;\boldsymbol{\mu}), \quad \forall\, w,v \in X^e, \tag{8.1}$$

$$|||w|||_{\boldsymbol{\mu}} \equiv \sqrt{a(w,w;\boldsymbol{\mu})}, \quad \forall\, w \in X^e. \tag{8.2}$$

These are obviously parameter dependent. Also note that due to the coercivity and continuity of $a(\cdot,\cdot;\boldsymbol{\mu})$ these expressions are well-defined.

For our reduced basis approximation however, we need a *parameter independent* inner product and norm, e.g. to orthogonalize our RB basis, a point we will return to later on. To this end, we choose an inner product and norm associated with a specific parameter value $\bar{\boldsymbol{\mu}}$.

$$(((w,v)))_{\bar{\boldsymbol{\mu}}} = a(w,v;\bar{\boldsymbol{\mu}}), \quad \forall\, w,v \in X^e, \tag{8.3}$$

$$|||w|||_{\bar{\boldsymbol{\mu}}} \equiv \sqrt{a(w,w;\bar{\boldsymbol{\mu}})}, \quad \forall\, w \in X^e. \tag{8.4}$$

As $X^{\mathcal{N}} \subset X^e$, our discrete truth approximation space inherits the norm from the exact space,

$$\|w\|_{X^{\mathcal{N}}} = \|w\|_{X^e} = |||w|||_{\bar{\boldsymbol{\mu}}}, \quad \forall\, w \in X^{\mathcal{N}}.$$

The particular choice of $\bar{\boldsymbol{\mu}}$ will not influence the RB solution. However it will have impact on the sharpness of the *a posteriori* error estimate which we return to in Section 8.2.3.

For the optimization problem presented in Chapter 6 a reasonable choice is for $\bar{\boldsymbol{\mu}}$ to be the parameters giving an undeformed, i.e. rectangular, computational domain. This implies that $\bar{\boldsymbol{\mu}} = (0,0)^T$.

We also define the coercivity constant

$$\alpha(\boldsymbol{\mu}) = \inf_{w \in X^{\mathcal{N}}} \frac{a(w,w;\boldsymbol{\mu})}{a(w,w;\bar{\boldsymbol{\mu}})}, \tag{8.5}$$

for all $\boldsymbol{\mu} \in \mathcal{D}$ and the continuity constant

$$\beta(\boldsymbol{\mu}) = \sup_{w \in X^{\mathcal{N}}} \sup_{v \in X^{\mathcal{N}}} \frac{a(w,v;\boldsymbol{\mu})}{a(w,v;\bar{\boldsymbol{\mu}})}, \tag{8.6}$$

for all $\boldsymbol{\mu} \in \mathcal{D}$ in the parametric case.

8.2 The Reduced Basis Methodology

The reduced basis methodology is built up by three basic ingredients:

1. Derivation and solution of an algebraic set of equations.
2. Efficient selection of snapshot parameters.
3. Rigorous and sharp error estimates.

We here present the general procedure for completing each of these steps. In Section 8.4 we give a more detailed description for a concrete problem, the two-dimensional Laplace equation on deformed geometry.

Intuitively it would seem more reasonable to perform step 2 before step 1. However, as will become apparent in the following sections, we need to compute the RB solution for a large number of parameters each time we expand our parameter space.

8.2.1 Derivation of algebraic equations

We present the methodology of the Lagrange Reduced Basis recipe [1]. Let N_{max} be the maximum dimension of our RB space X_N. Now introduce a set of parameters

$$\boldsymbol{\mu}^n \in \mathcal{D}, \quad n = 1, \dots, N_{max}.$$

Each of these is associated with a RB sample. The set of the N first parameter samples we denote by

$$S_N = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^N\}, \quad 1 \le N \le N_{max}.$$

The idea is now to calculate the truth approximation, or *snapshot*, for each of the corresponding parameter values in S_N, $u^{\mathcal{N}}(\boldsymbol{\mu}^1), \dots, u^{\mathcal{N}}(\boldsymbol{\mu}^N)$. These snapshots we again use to build our RB approximation space X_N. Assuming linearly independent snapshots, X_N is given as

$$X_N = \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^1), \dots, u^{\mathcal{N}}(\boldsymbol{\mu}^N)\}.$$

From here on we will for convenience denote $u^{\mathcal{N}}(\boldsymbol{\mu}^j)$ simply by u^j. Notice the hierarchical property $X_1 \subset X_2 \subset, \dots, \subset X_N$. By construction, we also have $X_N \subset X^{\mathcal{N}} \subset X^e$.

Numerically it is preferable to work with an orthogonal basis. To achieve this we introduce the orthogonal basis vectors $\zeta^1, \dots, \zeta^N$ which we obtain from $u^1, \dots, u^N$ through a modified Gram Schmidt (MGS) procedure. The basis elements are here orthogonalized with respect to the $\bar{\boldsymbol{\mu}}$ inner product given in (8.3). We can now represent the approximation $u_N \in X_N$ as

$$u_N(\boldsymbol{\mu}) = \sum_{n=1}^{N} u_{Nn}(\boldsymbol{\mu}) \zeta^n.$$

The unknown coefficients $u_{Nn}(\boldsymbol{\mu}), n = 1, \dots, N$ are found using a Galerkin procedure. Hence our reduced problem becomes: Given a new parameter value $\boldsymbol{\mu} \in \mathcal{D}$, find $u_N(\boldsymbol{\mu}) \in X_N$ such that

$$a(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = l(v; \boldsymbol{\mu}), \quad \forall\, v \in X_N, \tag{8.7}$$

and evaluate the output given as a linear *output functional* l^o.

$$s_N(\boldsymbol{\mu}) = l^o(u_N(\boldsymbol{\mu}); \boldsymbol{\mu}). \tag{8.8}$$

That is, we find the best linear combination – in the energy norm – spanned by the previously computed snapshot solutions to satisfy the new parameter $\boldsymbol{\mu}$.

To arrive at a system of algebraic equations, we need to realize the condition "$\forall v \in X_N$". As usual, we do this by successively choosing v to be the basis functions ζ^p, $p = 1, \dots, N$. Again by bilinearity of $a(\cdot, \cdot; \boldsymbol{\mu})$ and linearity of $l(\cdot; \boldsymbol{\mu})$, this gives us the N conditions

$$\sum_{m=1}^{N} u_{Nn} a(\zeta^n, \zeta^p; \boldsymbol{\mu}) = l(\zeta^n; \boldsymbol{\mu}), \quad p = 1, \dots, N, \tag{8.9}$$

and the output is evaluated as

$$s_N(\boldsymbol{\mu}) = \sum_{n=1}^{N} u_{Nn} l^o(\zeta^n; \boldsymbol{\mu}). \tag{8.10}$$

If the output functional $l^o(\cdot; \boldsymbol{\mu}) = l(\cdot; \boldsymbol{\mu})$ the output is said to be *compliant* [1].

How we solve this system will depend both on the dimension N and the representation of unknowns and basis functions. Also it is of great importance how we evaluate $a(\cdot, \cdot; \boldsymbol{\mu})$, $l(\cdot; \boldsymbol{\mu})$ and $l^o(\cdot; \boldsymbol{\mu})$. We will return to these issues in Sections 8.3.2 and 8.3.4.

8.2.2 Parameter sampling

In Section 8.2.1 we assumed the set of RB parameters $S_N = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^N\}$ to be known. In general, this will not be the case, and we need some procedure to select the $\boldsymbol{\mu}$'s.

Of course, one way is to simply pick new parameters at random. This approach is easy to realize and very quick. However, the quality of our approximation space, X_N, is hard to predict. Also, we take no advantage of the properties of our underlying problem in the parameter selection.

A second possibility is to manually determine the samples. For a small number of parameter values this is feasible, but if $N \sim \mathcal{O}(10)$ or larger, this process becomes tedious. It also requires extensive knowledge of the characteristics of our problem to pick good parameters.

Finding the optimal choice of parameters is difficult. We will, as in the EI algorithm, use a greedy approach which finds the locally optimal parameter choice at each step. As our parameter domain $\mathcal{D}$ contains infinitely many points we will again settle for a finite "training" sample $\Xi_t \subset \mathcal{D}$. We assume this subset to be rich enough to give a good description of how $u(\boldsymbol{\mu})$ behaves over $\mathcal{D}$.

The goal of the greedy algorithm is to find the parameter which maximizes the current error in the output over Ξ_t. That is

$$\boldsymbol{\mu}^N = \arg \max_{\boldsymbol{\mu} \in \Xi_t} |s^{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|.$$

However, computing the error in the output would require the solution $u(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_t$. Assume that we are able to compute an estimate for the output error,

$$\Delta_N^{out}(\boldsymbol{\mu}) \geq |s^{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|, \tag{8.11}$$

much cheaper (that is, independent of $\mathcal{N}$). This is known as the *a posteriori* error estimate [1] and will be discussed in the next section. The greedy parameter selection algorithm is given in Algorithm 8.1. Note that the orthogonalization step is performed as a part of the algorithm to ensure numerical stability. The

Algorithm 8.1 Greedy parameter selection

```
Given inital parameter $\boldsymbol{\mu}^1$, compute $u^{\mathcal{N}}(\boldsymbol{\mu}^1)$.
$S_1 \leftarrow \{\boldsymbol{\mu}^1\}$
Compute $\zeta^1$, $X_1 \leftarrow \text{span}\{\zeta^1\}$.
for $2 \leq N \leq N_{max}$ do
    $\boldsymbol{\mu}^N \leftarrow \arg\max_{\boldsymbol{\mu} \in \Xi_t} \Delta_N^{out}(\boldsymbol{\mu})$.
    Compute $u^{\mathcal{N}}(\boldsymbol{\mu}^N)$.
    $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}^N\}$.
    Compute $\zeta^N$, $X_N \leftarrow X_{N-1} \cup \text{span}\{\zeta^N\}$.
end for
```

greedy algorithm is automatic and can also be decoupled into a computationally expensive, parameter independent offline stage and a fast online parameter dependent stage. How this is achieved will depend on the particular problem and can be far from trivial. This decoupling will be the topic of Section 8.3.4, but now we take a closer look on the error estimate $\Delta_N(\boldsymbol{\mu})$.

8.2.3 *A posteriori* error estimator

As mentioned in the previous section, the development of an efficient *a posteriori* error estimator is essential for the greedy algorithm to work. To obtain the upper bound for the output error in (8.11) we first find an upper bound for the error in the field-variable $u(\boldsymbol{\mu})$, $\Delta_N(\boldsymbol{\mu})$. If we assume the output to be compliant, the output error bound will then be $\Delta_N^{out}(\boldsymbol{\mu}) = \Delta_N^2(\boldsymbol{\mu})$.

To develop this bound we need what is known as the error residual relationship [1]. Namely that the error $e^{\mathcal{N}}(\boldsymbol{\mu}) \equiv u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})$ satisfies

$$a(e^{\mathcal{N}}, v; \boldsymbol{\mu}) = r_N(u; \boldsymbol{\mu}), \quad \forall\, v \in X^{\mathcal{N}}. \tag{8.12}$$

Here the residual $r(v; \boldsymbol{\mu})$ is defined as

$$r(v; \boldsymbol{\mu}) \equiv l(v; \boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}), \quad \forall\, v \in X^{\mathcal{N}}, \tag{8.13}$$

and is an element in the dual space of $X^{\mathcal{N}}$, $(X^{\mathcal{N}})'$.

Now, by the Riesz representation theorem [12], we know that there exists a unique function $\hat{e}^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ such that

$$(\hat{e}^{\mathcal{N}}(\boldsymbol{\mu}), v)_{X^{\mathcal{N}}} = r(v; \boldsymbol{\mu}), \quad \forall\, v \in X^{\mathcal{N}}. \tag{8.14}$$

In addition

$$\|r(\cdot; \boldsymbol{\mu})\|_{(X^{\mathcal{N}})'} \equiv \sup_{v \in X^{\mathcal{N}}} \frac{r(v; \boldsymbol{\mu})}{\|v\|_{X^{\mathcal{N}}}} = \|\hat{e}^{\mathcal{N}}(\boldsymbol{\mu})\|_{X^{\mathcal{N}}}. \tag{8.15}$$

This dual norm relation is central to the online/offline decoupling of our error estimator.

We can now present the error bounds for the energy norm and output as

$$\Delta_N(\boldsymbol{\mu}) \equiv \frac{\|\hat{e}^{\mathcal{N}}(\boldsymbol{\mu})\|_{X^{\mathcal{N}}}}{\sqrt{\alpha_{LB}(\boldsymbol{\mu})}}, \tag{8.16}$$

$$\Delta_N^{out}(\boldsymbol{\mu}) \equiv \frac{\|\hat{e}^{\mathcal{N}}(\boldsymbol{\mu})\|^2_{X^{\mathcal{N}}}}{\alpha_{LB}(\boldsymbol{\mu})}. \tag{8.17}$$

Recall that $\Delta_N^{out}(\boldsymbol{\mu}) = \Delta_N^2(\boldsymbol{\mu})$ which follows from compliance. Here we have introduced $\alpha_{LB}(\boldsymbol{\mu})$, a lower bound on the coercivity constant. This bound has to satisfy

$$\alpha_{LB}(\boldsymbol{\mu}) \leq \alpha(\boldsymbol{\mu}) = \inf_{v \in X^{\mathcal{N}}} \frac{a(v,v;\boldsymbol{\mu})}{a(v,v;\bar{\boldsymbol{\mu}})}. \tag{8.18}$$

Deriving an explicit expression for $\alpha_{LB}(\boldsymbol{\mu})$ can be challenging and is highly problem dependent. For the error estimates (8.16) and (8.17) to be effective, we need the ability to evaluate $\alpha_{LB}(\boldsymbol{\mu})$ independent of $\mathcal{N}$. The simplest case we have for *parametrically coercive* problems, i.e. all $\Theta^q(\boldsymbol{\mu})$ and $a^q(\cdot,\cdot)$ are strictly positive. Here the lower bound is given as [1]

$$\alpha_{LB}(\boldsymbol{\mu}) \equiv \min_{q=1,\dots,Q_a} \frac{\Theta^q(\boldsymbol{\mu})}{\Theta^q(\bar{\boldsymbol{\mu}})}.$$

For general coercive and non-coercive problems, this approach will not work. Advanced methods based on a greedy offline stage and online solution of a linear program have been developed [42, 43], but for special problems it is also possible to "manually" compute the lower bound as we demonstrate in Section 8.3.3.

8.3 Application to deformed geometries

For our reduced basis example we will look at the two-dimensional Laplace equation. The derivation of the weak form will thus be very similar to the more general Poisson problem presented in Chapter 7. We will have the same exact solution space X^e defined in Section 2.2. Also, the linear and bilinear forms are equal, with $f = 0$. However, we have now introduced an output of interest evaluated through the output functional $l^o(\cdot;\boldsymbol{\mu})$.

In the next section we present the weak formulation, along with details regarding the Neumann boundary conditions and output evaluation.

8.3.1 Weak formulation

In Section 2.2 we derived the weak form of the homogeneous Dirichlet Poisson problem. This was continued in Section 7.1 where we introduced the reference domain $\hat{\Omega}$ and the mapping $\mathcal{F}$. If we now include the output evaluation the weak form is: Given $\boldsymbol{\mu} \in \mathcal{D}$, find $u(\boldsymbol{\mu}) \in X^e$ such that

$$a(u,v;\boldsymbol{\mu}) = l(v;\boldsymbol{\mu}), \quad \forall\, v \in X^e, \tag{8.19}$$

and evaluate the output of interest

$$s(\boldsymbol{\mu}) = l^o(u;\boldsymbol{\mu}). \tag{8.20}$$

From (7.4) we have

$$a(u,v;\boldsymbol{\mu}) = \int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \tilde{G}(\boldsymbol{\mu}) \hat{\nabla}\hat{u} \, \mathrm{d}\xi \mathrm{d}\eta, \tag{8.21}$$

and

$$l(v;\boldsymbol{\mu}) = \int_{\hat{\Gamma}_3} \hat{g}\hat{v}\mathcal{J}^S(\boldsymbol{\mu}) \, \mathrm{d}\eta. \tag{8.22}$$

In principle g may be a function $g(x,y)$. However, in our case Γ_3 will be a straight, vertical line and we choose g such that the flow Q through this border remains fixed and equal to 1. More formally

$$1 \equiv Q = \int_{\Gamma_3} g \, \mathrm{d}y,$$

which gives us $g = 1/|\Gamma_3|$.

A closer look at the output, given by the mean value of u over Γ_3, reveals that

$$s = l^o(u) = \frac{1}{|\Gamma_3|} \int_{\Gamma_3} u \, \mathrm{d}y = \frac{1}{g|\Gamma_3|} \int_{\Gamma_3} gu \, \mathrm{d}y = l(u).$$

We thus have a compliant problem.

From (8.21) and (8.22) it is clear that the parametric dependence is implicit in the geometric factors. However, if we insert $\mathcal{J}^S = \frac{\partial y}{\partial \eta}\big|_{\Gamma_3} = \frac{|\Gamma_3|}{2}$ we get

$$l(v;\boldsymbol{\mu}) = \int_{\hat{\Gamma}_3} \hat{v} \, \mathrm{d}\eta, \quad \forall\, v \in X^e. \tag{8.23}$$

Hence our linear form is in fact independent of the parameter.

8.3.2 Reduced Basis model

Notice that the snapshots used for creating the RB approximation space are not the solutions on the physical domain, but that the u^j's of Section 8.2.1 are here in fact $\hat{u}^{\mathcal{N}}(\boldsymbol{\mu}^j)$. This gives us the reduced basis approximation space

$$X_N = \mathrm{span}\{\hat{u}^{\mathcal{N}}(\boldsymbol{\mu}^1), \ldots, \hat{u}^{\mathcal{N}}(\boldsymbol{\mu}^N)\} = \mathrm{span}\{\zeta^1, \ldots, \zeta^N\}$$

where, as previously mentioned, we orthogonalize our space through a MGS procedure for better numerical stability.

If we now proceed as in Section 8.2.1 and let $u_N(\boldsymbol{\mu}) = \sum_{m=1}^N u_{Nm}\zeta^m$ and choose our test-functions accordingly, we get the same equations as in (8.9) which we represent in matrix form as

$$A_N(\boldsymbol{\mu})\underline{u}(\boldsymbol{\mu}) = \underline{l}_N(\boldsymbol{\mu}).$$

Due to compliance, the output from (8.10) is now easily evaluated as

$$s_N(\boldsymbol{\mu}) = (\underline{u}(\boldsymbol{\mu}))^T \underline{l}_N(\boldsymbol{\mu}).$$

Here, if we recall (8.9), each element of the reduced basis matrix, solution and right-hand side is given as

$$\begin{aligned} A_N(\boldsymbol{\mu})_{mn} &= a(\zeta^m, \zeta^n; \boldsymbol{\mu}), \quad 1 \le m,n \le N, \\ \underline{u}(\boldsymbol{\mu}) &= (u_{N1}, \ldots, u_{NN})^T, \\ \underline{l}_N(\boldsymbol{\mu})_m &= l(\zeta^m; \boldsymbol{\mu}), \quad 1 \le m \le N. \end{aligned} \tag{8.24}$$

This technique seems simple enough, but there are still a few issues to resolve:

1. How is $A_N(\boldsymbol{\mu})$ and $\underline{l}_N(\boldsymbol{\mu})$ generated (efficiently)?
2. How do we solve the RB system when $A_N(\boldsymbol{\mu})$ and $\underline{l}_N(\boldsymbol{\mu})$ have been obtained?

These questions we address in Section 8.3.4

8.3.3 *A priori* error analysis

To develop error bounds for the Reduced Basis approximation we need to demonstrate that the parameter dependent bilinear form $a(v, w, \boldsymbol{\mu})$ is still coercive and continuous. As $\tilde{G}$ is symmetric we may write $a(w, v; \boldsymbol{\mu})$ as

$$\begin{aligned} a(w, v; \boldsymbol{\mu}) &= \int_{\hat{\Omega}} (\hat{\nabla}\hat{w})^T \tilde{G} \hat{\nabla}\hat{v} \,\mathrm{d}\xi \mathrm{d}\eta \\ &= \int_{\hat{\Omega}} (\hat{\nabla}\hat{w})^T Q \Lambda Q^T \hat{\nabla}\hat{v} \,\mathrm{d}\xi \mathrm{d}\eta, \end{aligned}$$

where Q and Λ are 2×2 matrices containing the eigenvectors and eigenvalues of $\tilde{G}$. Upper and lower bounds for a can now be constructed by considering the maximum and minimum eigenvalues.

$$\begin{aligned} a(w, v; \boldsymbol{\mu}) &\leq \int_{\hat{\Omega}} \lambda_{max}(\xi, \eta; \boldsymbol{\mu}) (\hat{\nabla}\hat{w})^T \underbrace{Q^T Q}_{=I} \hat{\nabla}\hat{v} \,\mathrm{d}\xi \mathrm{d}\eta \\ &\leq \lambda^+(\boldsymbol{\mu}) \int_{\hat{\Omega}} (\hat{\nabla}\hat{w})^T \hat{\nabla}\hat{v} \,\mathrm{d}\xi \mathrm{d}\eta \\ &= \lambda^+(\boldsymbol{\mu}) a(w, v; \bar{\boldsymbol{\mu}}). \end{aligned} \tag{8.25}$$

Here we have used the parameter independent bilinear form $a(w, v; \bar{\boldsymbol{\mu}})$ introduced in Section 8.1.2 which corresponded to an undeformed domain. In addition we have defined $\lambda^+(\boldsymbol{\mu})$ as

$$\lambda^+(\boldsymbol{\mu}) = \max_{(\xi,\eta)\in\hat{\Omega}} \lambda_{max}(\xi, \eta; \boldsymbol{\mu}).$$

Similarly we have for the lower bound

$$a(w, v; \boldsymbol{\mu}) \geq \lambda^-(\boldsymbol{\mu}) a(w, v; \bar{\boldsymbol{\mu}}), \tag{8.26}$$

with

$$\lambda^-(\boldsymbol{\mu}) = \min_{(\xi,\eta)\in\hat{\Omega}} \lambda_{min}(\xi, \eta; \boldsymbol{\mu}).$$

We can use $\lambda^+(\boldsymbol{\mu})$ and $\lambda^-(\boldsymbol{\mu})$ as bounds for the parametric continuity and coercivity constants defined in Section 8.1.2. In other words we have $\lambda^+(\boldsymbol{\mu}) = \beta_{UB}(\boldsymbol{\mu}) \geq \beta(\boldsymbol{\mu})$ and $\lambda^-(\boldsymbol{\mu}) = \alpha_{LB}(\boldsymbol{\mu}) \leq \alpha(\boldsymbol{\mu})$.

Finding $\lambda^+(\boldsymbol{\mu})$ and $\lambda^-(\boldsymbol{\mu})$ is not trivial and can in general not be computed independent of $\mathcal{N}$. However, for special cases this is possible, as will be demonstrated later. The bounds developed above will also be important in the online/offline computation of the *a posteriori* error estimator.

Existence and uniqueness now follows from the arguments of section 2.2. We also have the *a priori* results collected in Theorem 8.1.

Theorem 8.1 *For $\boldsymbol{\mu} \in \mathcal{D}$ and $u_N(\boldsymbol{\mu}), s_N(\boldsymbol{\mu})$ satisfying (8.7) and (8.8) the following results hold*

$$|||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})|||_{\bar{\mu}} \leq \frac{\beta_{UB}(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})} \, ||| \, u^{\mathcal{N}}(\boldsymbol{\mu}) - v \, |||_{\bar{\mu}} \, . \tag{8.27}$$

And

$$|s^{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})| \leq \beta_{UB}(\boldsymbol{\mu}) \, ||| \, u^{\mathcal{N}}(\boldsymbol{\mu}) - v \, |||^2_{\bar{\boldsymbol{\mu}}} \, . \tag{8.28}$$

Proof: The first statement, (8.27), follows from the Galerkin orthogonality principle and Céa's lemma (7.9), as a concequence of (8.25) and (8.26). The output error follows from compliance and the continuity of the bilinear form.

It should be pointed out that the RB error is bounded only relative to the truth approximation. However, by the triangle inequality we have

$$\|u - u_N\| = \|u - u_N + u^{\mathcal{N}} - u^{\mathcal{N}}\| \leq \|u^{\mathcal{N}} - u_N\| + \|u - u^{\mathcal{N}}\|.$$

The final term, the error in the truth approximation, we assume to be negligible.

Now we present the approach for an efficient online/offline decoupling in the construction and solution of our Reduced Basis problem (8.7)–(8.8).

8.3.4 Online/offline decoupling using EI

To achieve an efficient online/offline decoupling of the RB problem we need affine expressions for the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ and linear functional $l(\cdot; \boldsymbol{\mu})$. Under these assumpions, the assembly of $A_N(\boldsymbol{\mu})$ and $\underline{l}_N(\boldsymbol{\mu})$ in (8.24) can be computed as

$$A_N(\boldsymbol{\mu})_{mn} = a(\zeta^m, \zeta^n; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(\zeta^m, \zeta^n), \text{ for } m, n = 1, \ldots, N, \tag{8.29}$$

and

$$l_N(\boldsymbol{\mu})_n = l(\zeta^n; \boldsymbol{\mu}) = \sum_{q=1}^{Q_l} \Theta_l^q(\boldsymbol{\mu}) l^q(\zeta^n), \text{ for } n = 1, \ldots, N. \tag{8.30}$$

Now the decomposition is clear. Offline we compute the parameter independent bilinear forms $a^q(\zeta^m, \zeta^n)$ and parameter independent linear functionals $l^q(\zeta^n)$ for all basis functions. The results are stored as Q_a $N \times N$ matrices and Q_l vectors of length N. In the online stage we are only left with the summations (8.29) and (8.30). The online complexity for constructing the linear set of equations is thus $\mathcal{O}(Q_a N^2 + Q_l N)$, which is independent of $\mathcal{N}$. As we expect N to be small, we can use a direct solver for $A_N(\boldsymbol{\mu})\underline{u}_N(\boldsymbol{\mu}) = \underline{l}_N(\boldsymbol{\mu})$ giving a total online complexity of $\mathcal{O}(N^3 + Q_a N^2 + Q_l N)$.

If we look at the linear form given in (8.23) we see that the right-hand side satisfies (8.30) with $Q_l = 1$, $\Theta_l^1(\boldsymbol{\mu}) = 1$ and $l^1(w) = \frac{1}{2} \int_{\Gamma_3} \hat{w} \, d\eta$. The bilinear form is not so obvious and requires some work.

First note that if we dissolve the matrix notation and exploit the symmetry of $\tilde{G}$ we can write

$$\begin{aligned} a(w,v;\boldsymbol{\mu}) &= \int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \tilde{G} \hat{\nabla}\hat{w} \, \mathrm{d}\xi \mathrm{d}\eta \\ &= \int_{\hat{\Omega}} \tilde{g}_{11}(\xi,\eta;\boldsymbol{\mu}) \frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \xi} \, \mathrm{d}\xi \mathrm{d}\eta \\ &+ \int_{\hat{\Omega}} \tilde{g}_{12}(\xi,\eta;\boldsymbol{\mu}) \left(\frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \eta} + \frac{\partial \hat{v}}{\partial \eta}\frac{\partial \hat{w}}{\partial \xi} \right) \mathrm{d}\xi \mathrm{d}\eta \\ &+ \int_{\hat{\Omega}} \tilde{g}_{22}(\xi,\eta;\boldsymbol{\mu}) \frac{\partial \hat{v}}{\partial \eta}\frac{\partial \hat{w}}{\partial \eta} \, \mathrm{d}\xi \mathrm{d}\eta. \end{aligned} \tag{8.31}$$

The geometric coefficient functions are given by (a detailed derivation can be found in [33])

$$\begin{aligned} \tilde{g}_{11}(\xi,\eta;\boldsymbol{\mu}) &= \frac{1}{\mathcal{J}}(x_\eta^2 + y_\eta^2), \\ \tilde{g}_{12}(\xi,\eta;\boldsymbol{\mu}) &= \frac{-1}{\mathcal{J}}(x_\xi x_\eta + y_\xi y_\eta), \\ \tilde{g}_{22}(\xi,\eta;\boldsymbol{\mu}) &= \frac{1}{\mathcal{J}}(x_\xi^2 + y_\xi^2). \end{aligned}$$

Here, $\mathcal{J} = x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian determinant introduced in Chapter 7. Also, we need the mapping $\mathcal{F}$ from the physical domain Ω to the reference domain $\hat{\Omega}$, which we denote by $x(\xi,\eta;\boldsymbol{\mu})$ and $y(\xi,\eta;\boldsymbol{\mu})$. This mapping is obtained by the Gordon-Hall algorithm as described earlier.

If we combine the expressions for the geometric coefficient functions with the decomposition of the bilinear form in (8.31), we see that we have three, potentially non-affine, bilinear forms. The obvious solution is to use empirical interpolation on the $\tilde{g}$'s. The resulting approximate bilinear forms will thus be affine by construction, and they can be computed independet of eachother. This is however not as straight-forward as it seems.

One of the advantages of the Gordon-Hall approach is that we can quickly find the mapping $\mathcal{F}$ for all the points in our grid using matrix-matrix multiplication. The problem is that we are not interested in all the grid points. We only want the points corresponding to the interpolation nodes given by the EI algorithm. Otherwise the online calculation of the EI coefficients would depend on the total number of points, $\mathcal{N}$.

For many problems the physical boundaries will only be given as a set of points. The boundary is then usually represented using some form of interpolation or approximation. Assuming isoparametric representation of the geometry, a complete online/offline decoupling will not be possible using the Gordon-Hall algorithm. However, if explicit expressions for the boundaries are available (and differentiable), as they are for our problem, we can find explicit expressions also for $x(\xi,\eta;\boldsymbol{\mu})$ and $y(\xi,\eta;\boldsymbol{\mu})$ and thus for $\tilde{g}_{11}$, $\tilde{g}_{12}$ and $\tilde{g}_{22}$.

The trigonometric deformation for the optimal parameter problem in Chapter 6 results in the following parametric description of the edges $\Gamma_1, \ldots, \Gamma_4$

$$\begin{aligned}
\Gamma_1 : \quad & \begin{pmatrix} x_1(\eta) \\ y_1(\eta) \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2}(1-a-b)(\eta+1)+b \end{pmatrix}, \\
\Gamma_2 : \quad & \begin{pmatrix} x_2(\xi) \\ y_2(\xi) \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(\xi+1) \\ 1+a\cos(\frac{\pi}{2}(\xi+1)) \end{pmatrix}, \\
\Gamma_3 : \quad & \begin{pmatrix} x_3(\eta) \\ y_3(\eta) \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2}(1+a+b)(\eta+1)-b \end{pmatrix}, \\
\Gamma_4 : \quad & \begin{pmatrix} x_4(\xi) \\ y_4(\xi) \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(\xi+1) \\ -b\cos(\frac{\pi}{2}(\xi+1)) \end{pmatrix}.
\end{aligned}$$

From these we obtain (details of the calculation is given in Appendix B)

$$\tilde{g}_{11}(\xi,\eta;\boldsymbol{\mu}) = 1+(a+b)\cos\left(\frac{\pi}{2}(\xi+1)\right), \tag{8.32}$$

$$\tilde{g}_{12}(\xi,\eta;\boldsymbol{\mu}) = \frac{\pi}{2}\left(a-b+(a+b)\eta\right)\sin\left(\frac{\pi}{2}(\xi+1)\right), \tag{8.33}$$

$$\tilde{g}_{22}(\xi,\eta;\boldsymbol{\mu}) = \frac{1+\frac{\pi^2}{4}(b-a-(a+b)\eta)^2\sin^2(\frac{\pi}{2}(\xi+1))}{1+(a+b)\cos(\frac{\pi}{2}(\xi+1))}. \tag{8.34}$$

We see that $\tilde{g}_{11}$ and $\tilde{g}_{12}$ admit affine decompositions. Also, $\tilde{g}_{22}$ can be recognized as the second example of Section 3.4.2. We thus know that it can be efficiently approximated by the EI method.

Introducing the EI expansion of $\tilde{g}_{22}$,

$$\tilde{g}_{22,M}(\xi,\eta;\boldsymbol{\mu}) = \sum_{m=1}^{M} \varphi_m(\boldsymbol{\mu}) q_m(\xi,\eta),$$

we present the complete online/offline procedure for evaluating $a(w,v;\boldsymbol{\mu})$. The parameter dependent coefficients are given as

$$\begin{aligned}
\Theta_a^1(\boldsymbol{\mu}) &= 1, \\
\Theta_a^2(\boldsymbol{\mu}) &= a+b, \\
\Theta_a^3(\boldsymbol{\mu}) &= a-b, \\
\Theta_{a,EI}^m(\boldsymbol{\mu}) &= \varphi_m(\boldsymbol{\mu}), \quad m=1,\ldots,M.
\end{aligned} \tag{8.35}$$

In addition we get a total of $Q_a = 3+M$ parameter independent bilinear forms to evaluate.

$$\begin{aligned}
a^1(w,v) &= \int_{\hat{\Omega}} \frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \xi}\,\mathrm{d}\xi\mathrm{d}\eta, \\
a^2(w,v) &= \int_{\hat{\Omega}} \Bigg(\cos(\frac{\pi}{2}(\xi+1))\frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \xi} \\
&\quad + \frac{\pi}{2}\eta\sin(\frac{\pi}{2}(\xi+1))\left(\frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \eta} + \frac{\partial \hat{v}}{\partial \eta}\frac{\partial \hat{w}}{\partial \xi}\right)\Bigg)\,\mathrm{d}\xi\mathrm{d}\eta, \\
a^3(w,v) &= \int_{\hat{\Omega}} \frac{\pi}{2}\eta\sin(\frac{\pi}{2}(\xi+1))\left(\frac{\partial \hat{v}}{\partial \xi}\frac{\partial \hat{w}}{\partial \eta} + \frac{\partial \hat{v}}{\partial \eta}\frac{\partial \hat{w}}{\partial \xi}\right)\,\mathrm{d}\xi\mathrm{d}\eta, \\
a_{EI}^m(w,v) &= \int_{\hat{\Omega}} q_m(\xi,\eta)\frac{\partial \hat{v}}{\partial \eta}\frac{\partial \hat{w}}{\partial \eta}\,\mathrm{d}\xi\mathrm{d}\eta, \quad m=1,\ldots,M.
\end{aligned} \tag{8.36}$$

This completes the online/offline decomposition of the RB equations. With $Q_f = 1$ and $Q_a = 3 + M$ the total online complexity is $\mathcal{O}(N^3 + MN^2 + M^2)$, where the final term, M^2, is required for calculating the M EI coefficients.

The first ingredient of our RB approach, construction and solution of the algebraic equations, is now complete. For the second and third ingredients we also need an efficient online procedure for evaluating the error estimate $\Delta_N(\boldsymbol{\mu})$. This follows in the next section.

8.3.5 Computational procedure for the error estimate

Recall the definition of $\Delta_N(\boldsymbol{\mu})$,

$$\Delta_N(\boldsymbol{\mu}) = \frac{\|\hat{e}(\boldsymbol{\mu})\|_{X^{\mathcal{N}}}}{\alpha_{LB}(\boldsymbol{\mu})}.$$

We thus need efficient procedures for computing both the dual residual norm $\|\hat{e}(\boldsymbol{\mu})\|_{X^{\mathcal{N}}}$ and the lower bound on the coercivity constant $\alpha_{LB}(\boldsymbol{\mu})$.

Dual residual norm

In Section 8.2.3 we introduced the dual residual representation $\hat{e}(\boldsymbol{\mu})$. This will now be used to develop an online/offline procedure of the *a posteriori* error estimate.

First note that by affine parameter dependence we can expand the residual as

$$r(v;\boldsymbol{\mu}) = \frac{1}{2}l^1(v) - \sum_{q=1}^{Q_a}\sum_{n=1}^{N} \Theta_a^q(\boldsymbol{\mu}) u_{Nn}(\boldsymbol{\mu}) a^q(\zeta^n, v), \quad \forall\, v \in X^{\mathcal{N}}. \tag{8.37}$$

Note that as we have introduced an EI expansion of one of the geometric factors, this is really an approximate residual. However we will assume the EI error to be small and adopt the notation of [1]. Now define $Q_N \equiv Q_f + Q_a N = 1 + (3+M)N$. It will prove convenient to write (8.37) as

$$r(v;\boldsymbol{\mu}) = \sum_{n=1}^{Q_N} \mathcal{E}_{Nn}(\boldsymbol{\mu}) h_{Nn}(v), \quad \forall\, v \in X^{\mathcal{N}}. \tag{8.38}$$

Here we have introduced $\underline{\mathcal{E}}_N(\boldsymbol{\mu}) \in \mathbb{R}^{Q_N}$ and $\underline{h}_N(v) \in \mathbb{R}^{Q_N}$ as

$$\begin{aligned}\underline{\mathcal{E}}_N(\boldsymbol{\mu}) = \Big(&\tfrac{1}{2},\\ &u_{N1}(\boldsymbol{\mu}), (a+b)u_{N1}(\boldsymbol{\mu}), \ldots, \varphi_M(\boldsymbol{\mu})u_{N1}(\boldsymbol{\mu}),\\ &u_{N2}(\boldsymbol{\mu}), (a+b)u_{N2}(\boldsymbol{\mu}), \ldots, \varphi_M(\boldsymbol{\mu})u_{N2}(\boldsymbol{\mu}),\\ &\vdots\\ &u_{NN}(\boldsymbol{\mu}), (a+b)u_{NN}(\boldsymbol{\mu}), \ldots, \varphi_M(\boldsymbol{\mu})u_{NN}(\boldsymbol{\mu})\Big)^T\end{aligned} \tag{8.39}$$

and

$$\begin{aligned}\underline{h}_N(v) = \Big(&l^1(v),\\ &-a^1(\zeta^1,v), -a^2(\zeta^1,v),\dots,-a^M_{EI}(\zeta^1,v),\\ &-a^1(\zeta^2,v), -a^2(\zeta^2,v),\dots,-a^M_{EI}(\zeta^2,v),\\ &\qquad\vdots\\ &-a^1(\zeta^N,v), -a^2(\zeta^N,v),\dots,-a^M_{EI}(\zeta^N,v)\Big)^T.\end{aligned} \tag{8.40}$$

If we combine (8.14) and (8.38) it follows that

$$\hat{e}(\boldsymbol{\mu}) = \sum_{n=1}^{Q_N} \mathcal{E}_{Nn}\hat{g}_{Nn},$$

where

$$(\hat{g}_{Nn}, v)_{X^{\mathcal{N}}} = h_{Nn}(v), \quad \forall\, v \in X^{\mathcal{N}}.$$

This implies that we can compute $\|\hat{e}(\boldsymbol{\mu})\|^2_{X^{\mathcal{N}}}$ as

$$\|\hat{e}(\boldsymbol{\mu})\|^2_{X^{\mathcal{N}}} = \sum_{n=1}^{Q_N}\sum_{m=1}^{Q_N} \mathcal{E}_{Nn}(\boldsymbol{\mu})\mathcal{E}_{Nm}(\boldsymbol{\mu})(\hat{g}_{Nn}, \hat{g}_{Nm})_{X^{\mathcal{N}}}. \tag{8.41}$$

The final inner products are parameter independent and can thus be calculated offline. Now define $G_N \in \mathbb{R}^{Q_N \times Q_N}$ as

$$(G_N)_{mn} = (\hat{g}_{Nm}, \hat{g}_{Nn})_{X^{\mathcal{N}}} = (H_N^T (A^{2D}(\bar{\boldsymbol{\mu}}))^{-1} H_N)_{mn}, \tag{8.42}$$

where $H_N \in \mathbb{R}^{\mathcal{N}\times Q_N}$ is given by

$$(H_N)_{in} = h_{Nn}(\phi_i), \quad i = 1,\dots,\mathcal{N}, n = 1,\dots,Q_N$$

and $X^{\mathcal{N}} = \text{span}\{\phi_1,\dots,\phi_{\mathcal{N}}\}$. Recall that these basis functions are constructed by multiplying two one-dimensional Lagrangian polynomials.

The online/offline procedure for $\|\hat{e}(\boldsymbol{\mu})\|_{X^{\mathcal{N}}}$ is thus as follows. Offline compute G_N from (8.42), online assemble $\underline{\mathcal{E}}_N$ and compute the sum (8.41). The online complexity is $\mathcal{O}(Q_N^2) = \mathcal{O}(M^2N^2)$.

Note that for the actual implementation of (8.42) we do not explicitly construct $A^{2D}(\boldsymbol{\mu})$ as this is of dimension $\mathcal{N}\times\mathcal{N}$. Instead the system $A^{2D}(\boldsymbol{\mu})Y = H$ is solved comlumn wise using CG and the fast operator evaluation from Section 7.3.2 resulting in an offline complexity of $\mathcal{O}(Q_N n_{it}\mathcal{N}^{3/2} + Q_N^2\mathcal{N})$. The first term is due to the calculation of Y and the second to the multiplication $H_N^T Y$.

Coercivity lower bound

In Section 8.3.3 we demonstrated that the lower bound for the coercivity constant is given by

$$\alpha_{LB}(\boldsymbol{\mu}) = \lambda^-(\boldsymbol{\mu}) = \min_{(\xi,\eta)\in\hat{\Omega}} \lambda_{min}(\xi,\eta;\boldsymbol{\mu}),$$

but no procedure for calculating $\lambda^-(\boldsymbol{\mu})$ was given. To do this, we again exploit the exact expressions for the geometric coefficients $\tilde{g}_{11}$, $\tilde{g}_{12}$ and $\tilde{g}_{22}$ given in (8.32)-(8.34). For a general 2×2 matrix A, it is easily demonstrated that the eigenvalues of A are given by

$$\lambda(A) = \frac{1}{2}\left(\mathrm{tr}(A) \pm \sqrt{\mathrm{tr}^2(A) - 4\det(A)}\right). \tag{8.43}$$

For $\tilde{G}$ we can simplify this expression considerably. A closer look at the determinant of $\tilde{G}$ reveals that

$$\begin{aligned}\det(\tilde{G}) &= \tilde{g}_{11}\tilde{g}_{22} - \tilde{g}_{12}^2 \\ &= \frac{1}{\mathcal{J}^2}\left((x_\eta^2 + y_\eta^2)(x_\xi^2 + y_\xi^2) - (x_\xi x_\eta + y_\xi y_\eta)^2\right) \\ &= \frac{1}{\mathcal{J}^2}\underbrace{(x_\xi y_\eta - x_\eta y_\xi)^2}_{=\mathcal{J}^2} \\ &= 1.\end{aligned}$$

We know that $\det(\tilde{G}) = \lambda_1\lambda_2$. This implies that $\lambda_{min} = 1/\lambda_{max}$, hence we only need to find λ_{max}. In addition, $\tilde{G}$ is SPD and real. The eigenvalues thus have to be positive, and the square-root in (8.43) cannot be negative. This means that λ_{max} is given by

$$\lambda_{max}(\tilde{G}) = \frac{1}{2}\left(\mathrm{tr}(\tilde{G}) + \sqrt{\mathrm{tr}^2(\tilde{G}) - 4}\right),$$

but by the same argument, λ_{max} is maximized when $\mathrm{tr}(\tilde{G})$ is maximized. It is thus sufficient to find the maximizer of the trace.

Introducing the variable $t = \frac{\pi}{2}(\xi + 1)$, the explicit expression for $\mathrm{tr}(\tilde{G})$ is

$$\mathrm{tr}(\tilde{G}) = 1 + (a+b)\cos(t) + \frac{1 + \frac{\pi^2}{4}(b - a - (a+b)\eta)^2\sin^2(t)}{1 + (a+b)\cos(t)}.$$

We immediately see that we must have $\eta = \pm 1$. Finding the optimal value for t, and thus ξ, requires some work, but it is also possible to calculate this analytically (see Appendix C). The extreme values of $\mathrm{tr}(\tilde{G})|_{\eta=\pm 1}$ are given by a quadratic equation in $\cos(t)$, which degenerates to a linear equation in the case $a + b = 0$. In addition, we also need to check the endpoints $\xi = \pm 1$.

In total, we get eight possible locations for the maximum (or seven if $a + b = 0$), all of which can be computed in $\mathcal{O}(1)$ operations. Hence, the online cost for computing the lower bound $\alpha_{LB}(\boldsymbol{\mu}) = \lambda^-(\boldsymbol{\mu}) = 1/\lambda^+(\boldsymbol{\mu})$ is independent of $\mathcal{N}$ (and even N and M).

The online/offline computation of $\Delta_N(\boldsymbol{\mu})$ is now clear and can be incorporated into the greedy parameter selection algorithm.

8.4 Optimization problem results

We here present the numerical results obtained for the optimization problem stated in Chapter 6. Details regarding the truth approximation, greedy parameter selection, RB error and finally the optimal parameter choice is given.

8.4.1 Spectral approximation

To obtain the high accuracy snapshots for building our RB solution space, we use the spectral method for deformed geometries presented in Chapter 7. Based on the observed performance of this technique, a polynomial degree of $\mathcal{P} = 30$ in each spatial direction is sufficient to ensure negligible errors from the truth approximation. This actually gives us $\mathcal{P}+1$ degrees of freedom in the y-direction and $\mathcal{P}$ in the x-direction, i.e. a total of $\mathcal{N} = (\mathcal{P}+1)\mathcal{P} = 930$ unknowns.

8.4.2 Parameter domain and samples

For the RB solution of the optimization problem, we will consider deformations in the range $\mathcal{D} = \{(a, b), -0.4 \leq a, b \leq 0.4\} \subset \mathbb{R}^2$. For the discrete surrogate Ξ_t we use a uniformly distributed grid of 25 points in each parameter, giving a total of 625 different combinations.

The parameters are chosen using the greedy approach given in Algorithm 8.1 with the initial parameter $\boldsymbol{\mu}^1 = (0, 0)^T$. The first 25 values are shown in Figure 8.1. Clearly the algorithm has a tendency to pick parameter values along the boundary of $\mathcal{D}$. This is however not surprising. For smooth two-dimensional problems, the Gauss-Lobatto points are also more densely distributed along the boundary, albeit not as extreme as observed here.

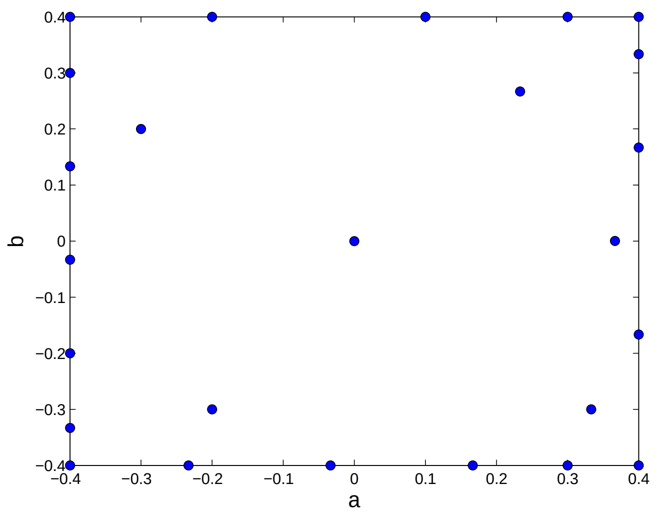


Figure 8.1: The first 25 RB parameter samples chosen by the greedy sampling strategy. The initial sample is the undeformed domain $(a, b) = (0, 0)$. The majority of samples are on the boundary of $\mathcal{D}$.

The first 5 geometries are shown in Figure 8.2. These correspond to the initial, undeformed case and the four corners of $\mathcal{D}$. This behavior is common when using the greedy parameter selection strategy. The initial samples will try to "cover" as much of $\mathcal{D}$ as possible.

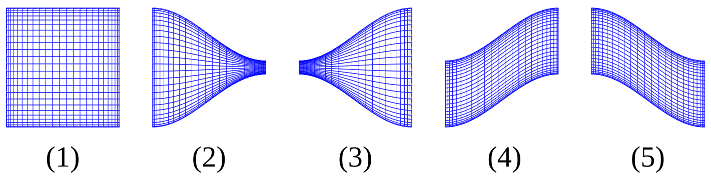


Figure 8.2: The first 5 snapshot geometries used in the construction of the RB approximation space. The geometries correspond to the initial undeformed case and each of the four corners of $\mathcal{D}$.

Again, this is reasonable considering the smooth parameter dependence. When a RB parameter is chosen, we also expect the approximation to be good in a neighborhood of this parameter value.

8.4.3 Reduced basis solution and error

For a problem of this type we expect rapid convergence in the RB solution as we increase the number of basis functions [1]. However, as EI is used to approximate one of the geometric factors, an additional error is introduced. Thus if we keep the number of interpolation nodes, M, fixed there is no point in increasing the size of the RB space beyond a certain value as the total error will be dominated by the EI error.

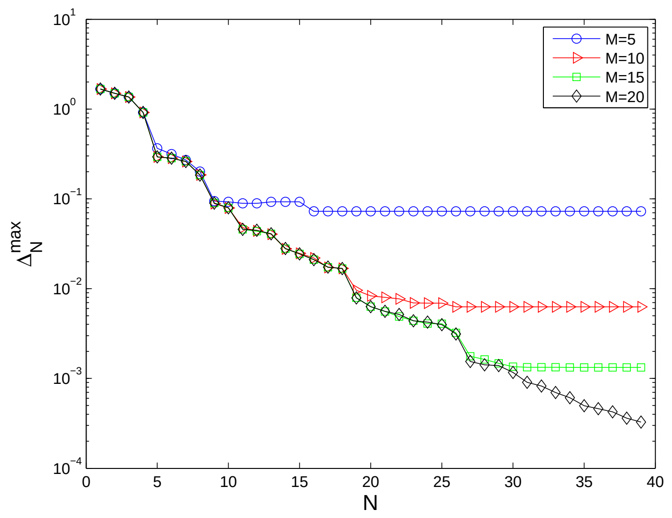


Figure 8.3: Maximum error estimate for the RB solution for different values of M. We have exponential convergence in N, but when the accuracy in the interpolation is reached, the EI error becomes dominant.

This behavior is clearly demonstrated in Figure 8.3 which shows the maximum error estimate,

$$\Delta_N^{max} = \max_{\boldsymbol{\mu} \in \Xi_t} \Delta_N(\boldsymbol{\mu}), \tag{8.44}$$

as a function of N for different values of M. We observe exponential convergence in N, but at a certain point the error stays fixed. If we compare the results to the EI error from Section 3.4.3, Figure 3.14, we see that the value of the error estimate is in agreement with the EI error for a given M.

In this kind of situation it is important to have an *a posteriori* error estimate which takes into account both the RB and EI error. Otherwise we could risk ending up in a situation where we waste resources by choosing either N or M to large or even fail to achieve a predefined required tolerance because the EI error is not taken into consideration.

8.4.4 Optimal flow parameters

To determine the optimal choice of a and b we compute the output, via our RB solution, for a grid with 33 points in each direction. Thus we need to compute the output for a total of $33^2 = 1089$ different values of the parameters.

If we require a maximum error in the output of 10^{-4}, we see from Figure 8.3 that $N = 19$ RB samples will be sufficient (remember that the output error is less than the field variable error squared. A field variable error of 10^{-2} will thus suffice). The number of EI samples used is $M = 15$.

The entire online computation for all 1089 parameter choices is completed in on average 0.21 seconds. For comparison, the evaluation of a "truth" solution for a single parameter value takes on average 0.20 seconds. The online evaluation is thus approximately 1000 times faster than a brute force approach. And this is only for a simple two-dimensional problem, where fast algorithms are available also for the solution of the full PDE.

In Figure 8.4 we show a contour plot of the output as a function of a and b. As expected, the minimal pressure drop occurs when $a = b = 0$. It is also clear from Figure 8.5 that the assumption of smooth parameter dependence is justified. This was also established through the exponential convergence demonstrated in Section 8.4.3.

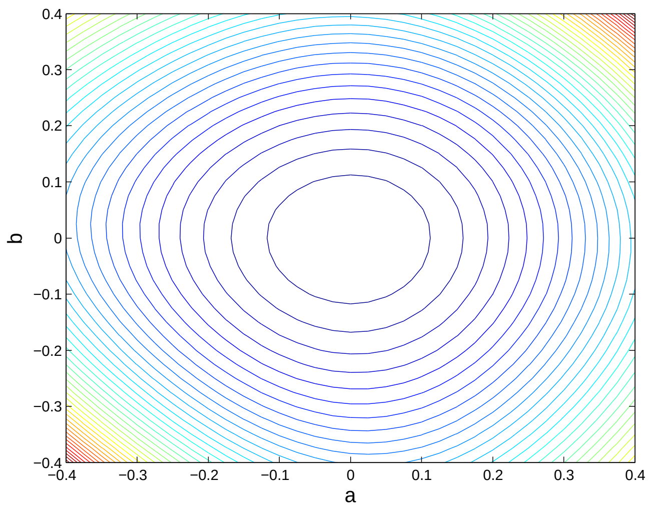


Figure 8.4: Contour plot for the optimal parameter choice. We see that $(a, b) = (0, 0)$ gives the lowest value.

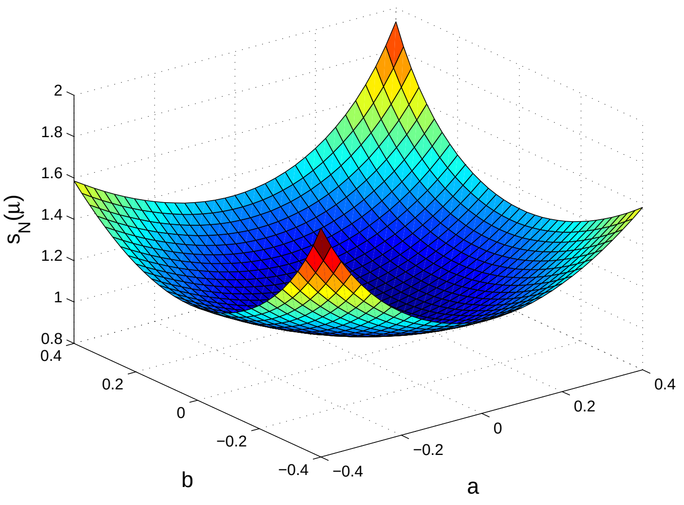


Figure 8.5: Surface plot for the optimal parameter choice. The parametric dependence is very smooth, as expected.

Chapter 9

Conclusions

The results in this report fall into two main categories. In the first part we investigated the properties of the empirical interpolation procedure as an isolated method. The second part connected empirical interpolation to the reduced basis framework to achieve an efficient online/offline decomposition for a non-affine problem.

In Section 3.4.1 we verified exponential convergence for functions with analytic parameter dependence. Also, we obtained results similar to standard polynomial interpolation for functions with limited regularity in the parameter in Section 3.4.2. The latter results were explained by very moderate increase in the Lebesgue constant, even for large values of M.

Chapters 4 and 5 were devoted to applications of the empirical interpolation procedure. Without loss of accuracy, we demonstrated significant computational gains for the EI quadrature, especially in higher dimensions. This follows from the $\mathcal{O}(M^2)$ complexity for calculating the EI coefficients, regardless of the dimension of the underlying problem.

Empirical interpolation was also used as a tool in the solution of nonlinear diferential equations. The standard EI method requires the solution at the interpolation nodes to compute the parameter dependent coeffients. For this type of problem that information is not readily available. We approximated the solution only in the interpolation points using a Runge-Kutta method, thus obtaining approximate EI coefficients. The additional error can be difficult to control, and the approach should be used with care. However for our examples the results were promising, and the basic idea is definately worthy of further research.

The second part of the report is, as mentioned, devoted to efficient solution of a non-affine problem. This involved the evaluation of an ouput functional to a parameter dependent partial differential equation. Using a conventional approach such as the spectral method to compute the output for a large number of parameter values would be very time consuming.

As a more feasible alternative, we introduced the reduced basis methodology in Chapter 8. Online/offline decoupling of the RB equations was discussed, and we were able to achieve complete independence of the truth solution complexity in the online stage. This was made possible by introducing EI to approximate the only non-affine geometric factor.

We also achieved an online/offline decomposition of the *a posteriori* error estimator by computing a coercivity lower bound based on exact calculation of the eigenvalues of the geometry matrix. This was incorporated into a greedy algorithm to automatically increase the number of basis elements in our RB approximation space X_N.

Using the error estimate we solved the optimization problem to a predefined required accuracy. The solution was obtained roughly 1000 times faster than would have been the case with a brute force conventional method, thus providing a good example of the power of the reduced basis method under the right circumstances.

The empirical interpolation method is a fairly new development, and the amount of literature on the subject is very limited. However, it is definitely an interesting method, and it has potential use in a wide range of applications such as image processing and compression, visualization, animation and inverse problems. More research is needed, both theoretical and practical, especially for the case with limited parametric regularity.

Also, our use of EI to approximate geometric factors was done on a simple deformation. A more general procedure would be preferable, as in most cases the analytic expressions for the elements of $\tilde{G}$ will not be readily available.

Bibliography

[1] A.T. Patera and G. Rozza. *Reduced Basis Approximation and A Posteriori Error Estimation for Parametrized Partial Differential Equations.* Version 1.0, Copyright MIT 2006, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering.

[2] Cuong N. Nguyen, Gianluigi Rozza, and Anthony A. Patera. Reduced Basis Estimation and A Posteriori Error Estimation for the Time-Dependent Viscous Burgers Equation. *Calcolo*, 2008. (submitted), work in progress.

[3] K. Veroy, C. Prud'homme, DV Rovas, and AT Patera. A Posteriori Error Bounds for Reduced-Basis Approximation of Parametrized Noncoercive and Nonlinear Elliptic Partial Differential Equations. *AIAA*, Paper 2003-3947, 2003. Proceedings of the 16th AIAA Computational Fluid Dynamics Conference.

[4] Sebastien Boyaval, Claude Le Bris, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. A Reduced Basis Approach for Variational Problems with Stochastic Parameters: Application to Heat Conduction with Variable Robin Coefficient. *Computer Methods in Applied Mechanics and Engineering*, 2008. Submitted.

[5] Alf Emil Løvgren, Yvon Maday, and Einar Rønquist. A Reduced Basis Element Method for the Steady Stokes Problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 40:529–522, 2006.

[6] M. Barrault et al. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. Paris*, Ser. I 339:667–672, 2004.

[7] M.A. Grepl, Y. Maday, N.C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *M2AN Math. Model. Numer. Anal.*, 41(3):575–605, 2007.

[8] Nguyen Ngoc Cuong. *Reduced-Basis Approximations and A Posteriori Error Bounds for Nonaffine and Nonlinear Partial Differential Equations: Application to Inverse Analysis.* PhD thesis, Singapore-MIT Alliance, 2005.

[9] Martin A. Grepl. *Reduced-Basis Approximation and A Posteriori Error Estimation for Parabolic Partial Differential Equations.* PhD thesis, MIT, 2005.

[10] Gianluigi Rozza. *Shape Design by Optimal Flow Control and Reduced Basis Techniques: Applications to Bypass Configurations in Haemodynamics*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2005.

[11] Yvon Maday, Ngoc Cuong Nguyen, Anthony T. Patera, and George S. H. Pau. A general multipurpose interpolation procedure: the magic points. *Communication on pure and applied analysis*, 8(1), 2009.

[12] Nicholas Young. *An Introduction to Hilbert spaces*. Cambridge University Press, 1988.

[13] Erwin Kreyzig. *Introductory Functional Analysis with applications*. Wiley, 1989.

[14] Luc Tartar. *An Introduction to Sobolev Spaces and Interpolation Spaces*. Springer, 2007. Lecture Notes of the Unione Matematica Italiana 3.

[15] Dietrich Braess. *Finite elements, Theory, fast solvers, and applications in solid mechanics, Third Edition*. Cambridge University Press, 2007.

[16] Richard L. Burden and J. Douglas Faires. *Numerical Analysis, 8th Edition*. Thomson, Brooks/Cole, 2005.

[17] Alfio Quarteroni and Alberto Vialli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, 1994.

[18] K. Weierstrass. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen vernderlichen. *Sitzungsberichte der Akademie zu Berlin*, pages 633–639 and 789–805, 1885.

[19] C. Canuto and A. Quarteroni. Approximation Results for Orthogonal Polynomials in Sobolev Spaces. *Mathematics of Computation*, 38(157):67–86, 1982.

[20] Christine Bernardi and Yvon Maday. Polynomial interpolation results in Sobolev spaces. *J. Comp. Appl. Math*, 43:53–80, 1992.

[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2003.

[23] Peter Bühlmann. Computational Statistics. Lecture notes, ETH Zürich, 2004.

[24] C. Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeit. Math. Phys.*, 46:224–243, 1901.

[25] C.E. Shannon. Communication in the Presence of Noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, 1949.

[26] M. Ganzburg. Moduli of Smoothness and Best Approximation of Functions with Singularities. *Comp. Math. Applic.*, 40:91–93, 2000.

[27] E. Harrier and G. Wanner. *Solving Ordinary Differential Equations II.* Springer, 2nd rev. edition, 2004.

[28] M.R. Mataušek. Direct Shooting Method for the Solution of Boundary-Value Problems. *Journal of Optimization Theory and Applications*, 12(2):152–172, 1973.

[29] Richard Weiss. The Convergence of Shooting Methods. *BIT Numerical Mathematics*, 13(4):470–475, 1973.

[30] C. Bernardi, Y. Maday. Edited by P.G. Ciarlet, and J.L Lions. *Handbook of Numerical Analysis. Volume V: Techniques of Scientific Computing (Part 2).* Elsevier, 1997. Spectral Methods.

[31] C.G. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid dynamics.* Springer, 2007.

[32] M.O. Deville, P.F. Fisher, and E.H. Mund. *High-Order Methods for Incompressible Fluid Flows.* Cambridge University Press, 2002].

[33] Einar M. Rønquist. Lecture Notes. MA8502 Numerical solution of partial differential equations. NTNU, 2008.

[34] Christoph Schwab. *P- and HP Finite Element Methods.* Oxford University Press, 1998.

[35] William Gordon and Charles Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Int. J. for Numer. Meth. in Engineering*, 7:461–477, 1973.

[36] A.T. Patera. A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion. *Journal of Computational Physics*, 54:458–488, 1984.

[37] Yvon Maday and Anthony T. Patera. Spectral Element Methods for the Incompressible Navier-Stokes Equations. *State-Of-The-Art Surveys on Computational Mechanics*, H00410, 1989.

[38] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The John Hopkins University Press, 3 edition, 1996.

[39] Yousef Saad. *Iterative Methods for Sparse Linear Systems, 2nd Edition.* Siam, 2003.

[40] Jorge Nocedal & Steven J. Wright. *Numerical Optimazation, Second Edition.* Springer, 2006.

[41] S.A. Orszag. Spectral Methods for Problems in Complex Geometries. *J. Comput. Physics*, 37:70–92, 1980.

[42] D.B.P Huynh et al. A successive constraint optimization method for lower bounds of parametric coercivity and inf-sup stability constants. *C. R. Acad. Sci. Paris*, Ser I(345), 2007.

[43] Yanlai Chen, Jan S. Hesthaven, Yvon Maday, and Jeriónimo Rodríguez. Improved Successive Constraint Method Based A Posteriori Error Estimate for Reduced Basis Approximation of 2D Maxwell's Problem. Submitted to Elsevier Science, 2008.

Appendix A

Regularity of $f(x) = |x|$

We will here demonstrate that for $x \in \Omega = [-1, 1]$, $f(x) = |x| \in H^{\frac{3}{2}-\varepsilon}(\Omega)$. To this end, consider an even expansion of $f(x)$ outside Ω as depicted in Figure A.1. To simplify the notation we will also assume a transformation of variables such that $\Omega \to [-\pi, \pi]$.

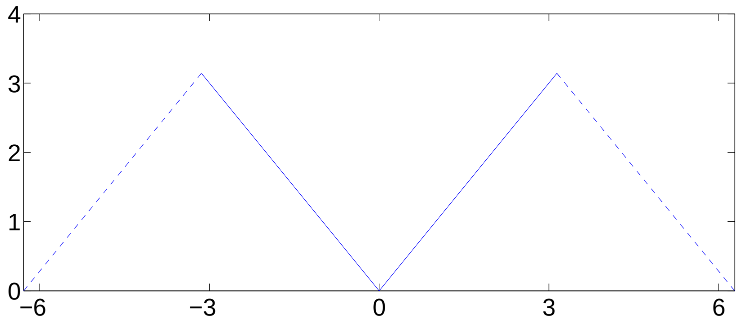


Figure A.1: Even expansion of $|x|$.

Now, f is clearly integrable on $[-\pi, \pi]$, hence we may express f as a Fourier series. In general

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{a}_k e^{ikx},$$

where $\hat{a}_k$ are complex numbers. By Parceval's identity [12] we can thus express the L^2-norm of f as

$$\|f\|^2_{L^2(\Omega)} = \int_{-\pi}^{\pi} f^2 \, dx = \sum_{k=-\infty}^{\infty} |\hat{a}_k|^2.$$

Similarly we get, by taking the derivative,

$$|f|^2_{H^1(\Omega)} = \sum_k |ik\hat{a}_k|^2 = k^2 |\hat{a}_k|^2,$$

and the full H^1-norm becomes

$$\|f\|^2_{H^1(\Omega)} = \sum_k (1+k^2)|\hat{a}_k|^2.$$

By continuing this procedure we can write the H^σ-norm in terms of the Fourier coefficients as

$$\|f\|^2_{H^\sigma(\Omega)} = \sum_{k=-\infty}^{\infty} \sum_{l=0}^{\sigma} k^{2l}|\hat{a}_k|^2.$$

This leads to the regularity of f being given by $f \in H^\sigma(\Omega)$ where σ is the maximal number satisfying (only the leading order of k will be important)

$$\sum_k k^{2\sigma}|\hat{a}_k|^2 < \infty. \tag{A.1}$$

Now let us use (A.1) to determine the regularity of $f(x) = |x|$. To do this we need to find the Fourier coefficients. As f is a real and even function we get

$$f(x) = \sum_k a_k \cos kx,$$

where the a_k's are determined by

$$\begin{aligned} a_k &= \int_{-\pi}^{\pi} f(x)\cos kx \,\mathrm{d}x = 2\int_0^\pi x\cos kx\,\mathrm{d}x \\ &= \cancelto{0}{\frac{2}{k}[x\sin kx]_0^\pi} - \frac{2}{k}\int_0^\pi \sin kx\,\mathrm{d}x \\ &= \frac{2}{k^2}\left[\cos kx\right]_0^\pi = \frac{2}{k^2}(1-(-1)^k). \end{aligned}$$

The condition thus essentially becomes

$$\sum_k k^{2\sigma}|a_k|^2 = \sum_k k^{2\sigma-4} < \infty.$$

For some sufficiently large $k^\star$ we have

$$\sum_k k^{2\sigma-4} < \infty \Leftrightarrow \int_{k^\star}^{\infty} k^{2\sigma-4}\,\mathrm{d}k < \infty$$

or

$$\sigma < \frac{3}{2}.$$

Hence, it follows that

$$f \in H^{\frac{3}{2}-\varepsilon}(\Omega).$$

Appendix B

Derivation of geometric factors

Consider a two-dimensional Poisson problem defined on some general domain Ω. When solving such a problem numerically, it is common to do all calculations on a reference domain $\hat{\Omega} = (-1,1)^2$, and transfer the results back to Ω via some mapping $\mathcal{F}$. In general, this mapping is represented by two functions $x(\xi,\eta)$ and $y(\xi,\eta)$, where (x,y) are the physical variables, and (ξ,η) are the reference variables. This mapping can be constructed with the Gordon-Hall algorithm [35].

When we transform the weak Laplace operator to reference variables,

$$\int_{\Omega} (\nabla v)^T \nabla u \, \mathrm{d}\Omega = \int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \tilde{G} \hat{\nabla}\hat{u} \, \mathrm{d}\xi \mathrm{d}\eta,$$

we use a geometry matrix $\tilde{G}$ given by

$$\tilde{G} = \frac{1}{\mathcal{J}} \begin{pmatrix} y_\eta^2 + x_\eta^2 & -y_\xi y_\eta - x_\xi x_\eta \\ -y_\xi y_\eta - x_\xi x_\eta & y_\xi^2 + x_\xi^2 \end{pmatrix}.$$

Here, $\mathcal{J}$ is the Jacobian determinant defined as

$$\mathcal{J} = x_\xi y_\eta - x_\eta y_\xi.$$

Obviously, $\tilde{G}$ is symmetric. To see how we derive exact expressions for the geometric coefficient functions $\tilde{g}_{11}$, $\tilde{g}_{12}$ and $\tilde{g}_{22}$ we must take a closer look at the Gordon-Hall algorithm. We will here assume a geometry with four edges.

The Gordon-Hall mapping $\mathcal{F}$, is constructed via three "sub-mappings". Two which connects the boundaries (left-right and top-bottom) and one which connects the four corners. Each of these are constructed using the linear shape functions

$$\begin{aligned} \phi_0(r) &= \frac{1-r}{2}, \\ \phi_1(r) &= \frac{1+r}{2}, \end{aligned} \tag{B.1}$$

for $-1 \leq r \leq 1$.

In a addition we need a representation of the boundaries. We denote these representations, one for each edge, by $\underline{\gamma_1}, \ldots, \underline{\gamma_4}$ where $\underline{\gamma_1} = (x(\xi,-1), y(\xi,-1))^T$

and so on. The three sub-mappings mentioned above can now be constructed as

$$\begin{aligned}
\mathcal{F}^{\xi}(\xi,\eta) &\equiv \phi_0(\xi)\underline{\gamma_4}(\eta)+\phi_1(\xi)\underline{\gamma_2}(\eta),\\
\mathcal{F}^{\eta}(\xi,\eta) &\equiv \phi_0(\eta)\underline{\gamma_1}(\xi)+\phi_1(\eta)\underline{\gamma_3}(\xi),\\
\mathcal{F}^{\xi\eta}(\xi,\eta) &\equiv \phi_0(\xi)\phi_0(\eta)\underline{\gamma_1}(-1)+\phi_0(\xi)\phi_1(\eta)\underline{\gamma_3}(-1)+\\
&\quad\ \phi_1(\xi)\phi_0(\eta)\underline{\gamma_1}(1)+\phi_1(\xi)\phi_1(\eta)\underline{\gamma_3}(1),
\end{aligned}$$

with the final mapping $\mathcal{F}\equiv\mathcal{F}^{\xi}+\mathcal{F}^{\eta}-\mathcal{F}^{\xi\eta}$.

To compute the Jacobian and geometry matrix $\tilde{G}$, we need the partial derivatives $\frac{\partial x}{\partial\xi}$, $\frac{\partial x}{\partial\eta}$, $\frac{\partial y}{\partial\xi}$ and $\frac{\partial y}{\partial\eta}$. We only compute $\frac{\partial x}{\partial\xi}$ in detail as the others are very similar. In the following calculations γ_i^x denotes the x-component of $\underline{\gamma_i}$.

$$\begin{aligned}
\frac{\partial x}{\partial\xi} &= \frac{\partial}{\partial\xi}\left(\mathcal{F}^{\xi,x}+\mathcal{F}^{\eta,x}-\mathcal{F}^{\xi\eta,x}\right)\\
&= \frac{\partial\phi_0}{\partial\xi}\gamma_4^x(\eta)+\frac{\partial\phi_1}{\partial\xi}\gamma_2^x(\eta)+\phi_0(\eta)\frac{\partial\gamma_1^x}{\partial\xi}+\phi_1(\eta)\frac{\partial\gamma_3^x}{\partial\xi}\\
&- \gamma_1^x(-1)\frac{\partial\phi_0}{\partial\xi}\phi_0(\eta)-\gamma_3^x(-1)\frac{\partial\phi_0}{\partial\xi}\phi_1(\eta)\\
&- \gamma_1^x(1)\frac{\partial\phi_1}{\partial\xi}\phi_0(\eta)-\gamma_3^x(1)\frac{\partial\phi_1}{\partial\xi}\phi_1(\eta)\\
&= \frac{1}{2}\Big(\gamma_2^x(\eta)-\gamma_4^x(\eta)\Big)+\phi_0(\eta)\frac{\partial\gamma_1^x}{\partial\xi}+\phi_1(\eta)\frac{\partial\gamma_3^x}{\partial\xi}\\
&- \frac{1}{2}\Big(\gamma_1^x(1)\phi_0(\eta)+\gamma_3^x(1)\phi_1(\eta)-\gamma_1^x(-1)\phi_0(\eta)-\gamma_3^x(-1)\phi_1(\eta)\Big).
\end{aligned}$$

If we do this also for the other partial derivatives, we can compute $\mathcal{J}$ and the elements of $\tilde{G}$ without difficulty. This procedure requires explicit, differentiable expressions for $\underline{\gamma_1},\ldots,\underline{\gamma_4}$.

Appendix C

Exact eigenvalues for the geometry matrix

Transforming the weak Laplace operator from physical to reference variables results in a geometry matrix $\tilde{G}$ given by

$$\tilde{G} = \frac{1}{\mathcal{J}} \begin{pmatrix} y_\eta^2 + x_\eta^2 & -y_\xi y_\eta - x_\xi x_\eta \\ -y_\xi y_\eta - x_\xi x_\eta & y_\xi^2 + x_\xi^2 \end{pmatrix}.$$

In addition, the assumption

$$\mathcal{J} = x_\xi y_\eta - x_\eta y_\xi > 0$$

is made. Here, $x = x(\xi, \eta)$ and $y = y(\xi, \eta)$ denote the functions taking us from reference variables (ξ, η) to physical variables (x, y). These will also depend on a parameter $\boldsymbol{\mu}$. Hence we get $\tilde{G} = \tilde{G}(\xi, \eta; \boldsymbol{\mu})$.

We are interested in the minimal eigenvalue of the geometry matrix $\tilde{G}$ with respect to the spatial variables ξ and η, $\lambda^-(\boldsymbol{\mu})$. This we may find as we have exact expressions for all the elements of $\tilde{G}$ given by

$$\begin{aligned} \tilde{g}_{11}(\xi, \eta; \boldsymbol{\mu}) &= 1 + (a+b)\cos\left(\frac{\pi}{2}(\xi+1)\right), \\ \tilde{g}_{12}(\xi, \eta; \boldsymbol{\mu}) &= \frac{\pi}{2}\left(a - b + (a+b)\eta\right)\sin\left(\frac{\pi}{2}(\xi+1)\right), \\ \tilde{g}_{22}(\xi, \eta; \boldsymbol{\mu}) &= \frac{1 + \frac{\pi^2}{4}(b - a - (a+b)\eta)^2 \sin^2(\frac{\pi}{2}(\xi+1))}{1 + (a+b)\cos(\frac{\pi}{2}(\xi+1))}. \end{aligned}$$

For a general 2×2 matrix A the eigenvalues are given by

$$\lambda(A) = \frac{1}{2}\left(\mathrm{tr}(A) \pm \sqrt{\mathrm{tr}^2(A) - 4\det(A)}\right).$$

If we let our matrix be $\tilde{G}$ it is easily demonstrated that $\det(\tilde{G}) = 1$. In addition, $\tilde{G}$ is SPD and real. From this we can deduce that $\lambda^-(\boldsymbol{\mu}) = 1/\lambda^+(\boldsymbol{\mu})$ and the maximum eigenvalue must be

$$\lambda^+(\boldsymbol{\mu}) = \max_{\xi,\eta}\left\{\frac{1}{2}\left(\mathrm{tr}(\tilde{G}(\xi,\eta;\boldsymbol{\mu})) + \sqrt{\mathrm{tr}^2(\tilde{G}(\xi,\eta;\boldsymbol{\mu})) - 4}\right)\right\}. \tag{C.1}$$

To find the maximizer of (C.1) it is clearly sufficient to find the maximizer of $\mathrm{tr}(\tilde{G})$ as these will coincide.

In our case, $\mathrm{tr}(\tilde{G})$ becomes

$$\mathrm{tr}(\tilde{G}) = 1 + (a+b)\cos(t) + \frac{1 + \frac{\pi^2}{4}(b - a - (a+b)\eta)^2 \sin^2(t)}{1 + (a+b)\cos(t)}.$$

For convenience, we have introduced $t = \frac{\pi}{2}(\xi + 1)$. We see that a maximum must occur at either $\eta = -1$ or $\eta = 1$. For t it is not as obvious, and we must check both the endpoints ($\xi = \pm 1$) and the extreme values. Hence we need to solve

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}\mathrm{tr}(\tilde{G})\Big|_{\eta=-1} \\ &= \left(-(a+b) + \frac{\frac{\pi^2}{2}(2b)^2\cos(t)}{1 + (a+b)\cos(t)} + \frac{(1 + \frac{\pi^2}{4}(2b)^2 \sin^2(t))(a+b)}{(1 + (a+b)\cos(t))^2} \right) \sin(t) \end{aligned}$$

and

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}\mathrm{tr}(\tilde{G})\Big|_{\eta=1} \\ &= \left(-(a+b) + \frac{\frac{\pi^2}{2}(2a)^2\cos(t)}{1 + (a+b)\cos(t)} + \frac{(1 + \frac{\pi^2}{4}(2a)^2 \sin^2(t))(a+b)}{(1 + (a+b)\cos(t))^2} \right) \sin(t). \end{aligned}$$

Here, $\sin(t)$ will be zero only at the endpoints, which we have to check for maxima anyway. If we use the relation

$$\sin^2(t) + \cos^2(t) = 1,$$

and simplify we get the following second order equation in $\cos(t)$ (we only show $\eta = -1$)

$$\underbrace{((a+b)^2 - (\pi b)^2)(a+b)}_{\alpha_-}\cos^2(t) + \underbrace{2((a+b)^2 - (\pi b)^2)}_{\beta_-}\cos(t) - \underbrace{(a+b)(\pi b)^2}_{-\gamma_-} = 0$$

with

$$\cos(t) = \frac{1}{2\gamma_-}\left(-\beta_- \pm \sqrt{\beta_-^2 - 4\alpha_-\gamma_-} \right). \tag{C.2}$$

In the degenerate case $a + b = 0$ (and $a, b \neq 0$) the equation reduces to

$$2(\pi b)^2 \cos(t) = 0 \Rightarrow \cos(t) = 0 \Rightarrow \xi = 0.$$

The number of potential maxima, and corresponding minima, is thus 8 (or 7 for $a + b = 0$). For each value of η (± 1) we have 4 possibilities for ξ, two endpoints and the two solutions of (C.2).