NTNU – Trondheim
Norwegian University of
Science and Technology

The Generalized Empirical Interpolation Method

Martin Arnesen

Master of Science in Physics and Mathematics
Submission date: February 2014
Supervisor: Einar Rønquist, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

A classical way to approximate functions is through interpolation. A classical interpolation method is based on a particular set of interpolation points and associated interpolating functions. For well-chosen points, the interpolation error can be quite small, yet simple to implement. The empirical interpolation method (EIM) was proposed a few years ago as a way to interpolate parameter-dependent function, or more generally, interpolation of functions spanned by a given set of functions. This method generates problem-dependent interpolation points and basis functions.

Very recently, an extension of this method was proposed, denoted as the generalized empirical interpolation method (GEIM). The GEIM considers approximation of linear functionals of a given set of functions. The objective of this thesis is to study the properties of the GEIM and consider potential applications such as parameter estimation.

Assignment given: 2. September 2013
Supervisor: Einar Rønquist, MATH

Preface

This thesis concludes my 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, 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.

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

All implementations are done in MATLAB, 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 Rønquist at the Department of Mathematical Sciences, NTNU. His ideas and feedback have been a great help throughout the work with this thesis.

Martin Arnesen
Trondheim
February 7, 2014

Abstract

The empirical interpolation method is an interpolation scheme with problem dependent basis functions and interpolation nodes, originally developed for parameter dependent functions. It was developed in connection with the reduced basis framework for fast evaluation of output from parameterized partial differential equations, but the procedure may be applicable to a variety of problems, such as image and pattern recognition, numerical integration and data compression. We present the theoretical background and implementation of the method, and give examples to verify exponential convergence for analytic problems.

An extension of the method was proposed recently, denoted as the generalized empirical interpolation method (GEIM). The GEIM considers a parametric manifold of functions, with a set of linear functionals.

Further, we explore how the interpolation points can be used as measurement points in the estimation of parameters from noisy data. We present the statistical framework, and we show how we can identify a set of parameter values that are consistent with our measurements.

Sammendrag

Den empiriske interpolasjonsmetoden en er interpolasjonsmetode med problemavhengige basisfunksjoner og interpolasjonsnoder, opprinnelig utviklet for parameteravhengige funksjoner. Metoden ble utviklet for rask evaluering av parametriserte partielle differensialligninger, men metoden kan være anvendbar innen flere felt, for eksempel bildegjenkjenning, numerisk integrasjon og komprimering av data. Vi presenterer den teoretiske bakgrunnen og implementering av metoden, og gir eksempler for å verifisere eksponentiell konvergens for analytiske problemer.

Nylig har en utvidelse av metoden blitt presentert, betegnet som den generaliserte empiriske interpoleringsmetoden. Den generaliserte versjonen betrakter approksimasjon av lineære funksjonaler av ett gitt sett av funksjoner.

Vi utforsker også hvordan interpolasjonsnodene kan brukes som målepunkter i estimering av parametre fra data med støy. Vi presenterer det statistiske rammeverket, og vi viser hvordan vi kan identifisere et sett av parameterverdier som er konsistente med de målte verdiene.

Contents

Chapter 1

Introduction

In interpolation we want to approximate a given function or fit a function to some numerical data. There are several reasons to do this, for example we can be given a function u that is expensive to evaluate in the form of a computer procedure. In this case, we want another function g that is simpler to evaluate and produce a reasonable approximation to u. Or we can have a table of numerical values and we want to fit a function through those points. Note that interpolation is not to be mistaken for regression. If for instance the values in the table have been contaminated by errors, as might occur if the values came from a physical experiment, we then want a formula that represent the data approximately. This type of technique is called regression analysis [1].

All the interpolation methods have one thing in common: they use a set of points in some domain Ω where the interpolating function and the given function match. We call the points interpolation nodes, and we say that the function interpolate in those nodes. Immediately a question pops up, what happens between the nodes? In the two-dimensional case the easiest choice would just be to use linear interpolation between each pair of adjacent nodes. Another common choice is to use polynomial interpolation between the nodes. This approach is well understood, but have proven rather bad under certain conditions, which brings us to the next question: if we are free to choose where we want the interpolation nodes, where do we want them? The obvious choice is to let the nodes be equidistant over Ω. However, with equidistant nodes, polynomial interpolation have the defect of being highly oscillatory when the number of interpolation nodes increases [2].

We will consider the case when we want to approximate a given parameter dependent function $u(x;\mu)$, where x is in the space domain Ω and μ represent the parameter domain. In one space dimension the location of almost optimal points is provided by Gauss-Lobatto Legendre nodes [3]. In dimensions greater than one more conditions must be fulfilled for a polynomial interpolation to be well defined, and the interpolant may not even be unique [4].

In 2004 the interpolation method called *Empirical Interpolation* was introduced [15]. The method was developed as a tool within the reduced basis framework for parametrized partial differential equations. Since then, work have been done to generalize the method further. We will look at one interpretation, and present the generalized empirical interpolation algorithm.

We will also explore how the interpolation points can be used in estimation of parameters from noisy data. We will present the statistical framework, and see how we can estimate given parameters that are consistent with our experimental data.

The empirical interpolation method is remarkably simple to implement, and has proven relatively competitive to other interpolation methods. In Chapter 2 we present the method, followed with examples to verify exponential convergence for analytic problems.

1.1 Gauss-Lobatto Legendre quadrature

We first consider the case when there are no parameter dependence, i.e. $u = u(x)$. Legendre polynomials are solutions to the Sturm-Liouville problem [6],

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}L_n(x)\right] + n(n+1)L_n(x) = 0,$$

which is in fact an eigenvalue problem,

$$\mathcal{L}L_n(x) = \lambda_n L_n(x).$$

Each Legendre polynomial $L_n(x)$ is an n-th degree polynomial. The polynomials can be expressed by

$$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n],$$

and with $L_0(x) = 1$ and $L_1(x) = x$ the polynomials satisfy the recurrence relation

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x).$$

By using the Legendre polynomials as basis, we write the approximation of a function $u(x)$ as

$$u_N(x) = \sum_{i=0}^{N} a_i L_i(x),$$

where $\{a_i\}$ is the set of basis coefficients. We want to find the best approximation of $u(x)$ by the polynomial $u_N(x)$. With "best" approximation we mean the approximation which minimizes the quadratic error

$$E = \int_\Omega (u-u_N)^2 \mathrm{d}x = \| u - u_N \|^2_{L^2(\Omega)}, \tag{1.1}$$

over all polynomials of degree less than or equal to N. This error can be minimized by adjusting the coefficients $\{a_i\}$. Note that we could let the "best" approximation be defined by the error of some other norm, for example we could consider the L^∞-norm, and minimize $E = \| u - u_N \|^2_{L^\infty(\Omega)}$. In search for good approximations we often consider the Lesbegue constant, Λ_N [7]. The constant give an idea of how good the interpolant is in comparison with the best polynomial approximation.

Assume we want to evaluate the integral

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

for some function g. With Gauss Legendre quadrature [6] the integral is approximated by

$$I \approx \sum_{i=0}^{n} w_i g(x_i),$$

where w_i is the quadrature weight corresponding to the function evaluated at x_i. Here there are $2n+2$ degrees of freedom. We are in principle free to choose where we want to evaluate g, and if we choose x_i to be the roots of the n-th Legendre polynomial L_n, it can be shown that we are able to exactly integrate a polynomial of degree $2n+1$ or less. In Gauss-Lobatto Legendre (GLL) quadrature, however, we fix x_0 and x_n to be the two endpoints -1 and 1, which leaves us with $2n$ degrees of freedom. If we now let x_i, $i = 1, \ldots, n-1$, be the roots of L_n', we are able to integrate a polynomial of degree $2n-1$ or less. In this case we can state that for all $g \in \mathbb{P}_{2n-1}(-1,1)$,

$$\int_{-1}^{1} g(x)\mathrm{d}x = \sum_{i=0}^{n} w_i g(x_i),$$

where $\mathbb{P}_q(-1,1)$ is the space of all polynomials of degree q or less.

A general limitation to the GLL points are that they are only defined on the line, the square or other tensor product domains. If GLL points are to be used in other domains, the problem must be mapped from the actual domain over to a reference domain where the GLL points are defined. This can often be either tedious or impractical.

Later we will use GLL quadrature to approximate several integrals over some domain Ω. If for instance $\Omega = (a,b) \times (c,d)$ and $g = g(x,y)$, the integral we need to consider is

$$I = \int_a^b \int_c^d g(x,y)\mathrm{d}y\mathrm{d}x.$$

We do a change of variables to get the integral on a familiar form, and are left with

$$I = \frac{(b-a)(d-c)}{4} \int_{-1}^{1}\int_{-1}^{1} g\left(\frac{b-a}{2}\xi + \frac{a+b}{2}, \frac{d-c}{2}\eta + \frac{c+d}{2}\right)\mathrm{d}\eta\mathrm{d}\xi,$$

which can be approximated by the tensor product

$$I \approx \frac{(b-a)(d-c)}{4} \sum_{i=0}^{n} \sum_{j=0}^{n} w_i w_j g\left(\frac{b-a}{2}\xi_i + \frac{a+b}{2}, \frac{d-c}{2}\eta_j + \frac{c+d}{2} \right).$$

1.2 Polynomial approximation and interpolation

We consider interpolation through the GLL points introduced in the previous section using Lagrange interpolation. Given a set of interpolation points $\{\xi_0, \ldots, \xi_N\}$ the i-th Lagrange interpolation polynomial is defined by

$$\begin{aligned} \ell_i^N(x) &\in \mathbb{P}_N(-1,1), & i &= 0, \ldots, N, \\ \ell_i^N(x_j) &= \delta_{ij}, & 0 &\leq i, j \leq N. \end{aligned}$$

For example the interpolation polynomial $\ell_2^5(x)$ is equal to 1 at the node (ξ_2) and 0 at the others, see Figure 1.1. If we let $\mathcal{I}_N[u]$ be the interpolant of u through

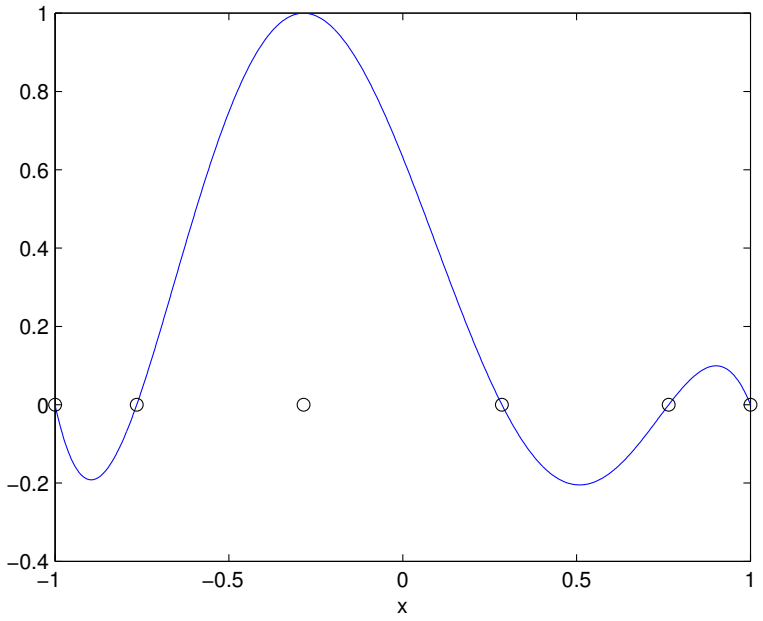

Figure 1.1: Lagrange interpolation polynomial $\ell_2^5(x)$. Note that $\ell_2^5(\xi_2) = 1$, but zero at the other GLL nodes.

$N+1$ GLL points, we can express $\mathcal{I}_N[u]$ as

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

Consider the quadratic error in (1.1). Assume that the function u has a certain regularity. In particular let $u \in H^\sigma(\Omega)$, where

$$H^\sigma(\Omega) = \{u | \int_{-1}^{1} \sum_{j=0}^{\sigma} \left(\frac{d^j u}{dx^j} \right)^2 \mathrm{d}x < \infty \},$$

i.e. the function u and all its derivatives up to order σ are square integrable. The norm induced by $H^\sigma(\Omega)$ is then given by

$$\| u \|^2_{H^\sigma(\Omega)} = \int_{-1}^{1} \left(u^2 + \left(\frac{du}{dx} \right)^2 + \ldots + \left(\frac{d^\sigma u}{dx^\sigma} \right)^2 \right) \mathrm{d}x.$$

This allows us to state an *a priori* error estimate [3],

$$\| u - \mathcal{I}_N[u] \|_{L^2(\Omega)} \leq c N^{-\sigma} \| u \|^2_{H^\sigma(\Omega)},$$

where $c > 0$ is a constant. We see that the error, measured in the L^2-norm, decreases as N increases, and that the rate of convergence is in fact dependent on the regularity of u. In the special case where $\sigma \to \infty$ then u is analytic, and the error decreases exponentially towards zero as N increases. In GLL interpolation the set of basis functions and interpolation nodes is completely detached from the problem, which 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.

Chapter 2

The Empirical Interpolation Method (EIM)

The basic interpolation methods use predefined interpolation points and basis functions. This allows easy implementation and predictable error behaviour. However, this approach does not adapt to the specific problem, and the only information extracted from the underlying function is the nodal values. The EIM uses a greedy algorithm that adapt to the problem by choosing the next interpolation node and forming the next basis function based on the current interpolant. We will see how this adaptive approach may be superior compared to GLL interpolation.

Consider a function $u(x;\mu)$, $x \in \Omega$, which depends smoothly on a parameter μ over some parameter space $\mathcal{D}$. We want to exploit the parametric dependence, which is not done in standard interpolation, i.e. for every different parameter value a new interpolant will have to be constructed separately.

Instead of predefined basis functions, the EIM we sample u at different points in $\mathcal{D}$, so that the interpolation space becomes $X_N = \text{span}\{u(x;\mu_1), \ldots, u(x;\mu_N)\}$. Because of the assumed smooth parameter dependence, we expect a linear combination of these basis functions to approximate $u(x;\mu)$ for any parameter value in $\mathcal{D}$.

Note that the EIM could be realized analytically, but this is often impossible or impractical. Instead we will consider discrete sets for both the parameter space $\mathcal{D}$ and for the space domain Ω. However, we are able to use our interpolant for all parameter values $\mu \in \mathcal{D}$. We say that we train the interpolation space, based on the discrete version of $\mathcal{D}$ and Ω; more on this below.

The EIM approach to approximate u is then

$$u(x;\mu) \approx \mathcal{I}_N[u](x;\mu) = \sum_{i=1}^{N} \varphi_i(\mu) q_i(x),$$

where $q_i(x)$ are the parameter independent basis functions that will span our interpolation space, and $\varphi_i(\mu)$ are parameter dependent basis coefficients. Once the interpolation process is done and the basis functions are constructed, the coefficients can be computed separately for any parameter value $\mu \in \mathcal{D}$. Even though the q_i-s are vectors in the discrete case, we will still refer to them as *basis functions* because it is more consistent, and as mentioned earlier the procedure could in principle be done analytically.

2.1 Interpolation procedure

The interpolation points will be referred to as magic points [4] as suggested by some of the original developers of the EIM. The interpolation procedure relies on a greedy selection of the next magic point, and this choice determines the next basis function. This may in some cases lead to a globally optimal algorithm, but in most cases, as for the EIM, this is not generally true.

Below follows the EIM algorithm, with a quick explanation of the steps. We let Ξ and Ω_d be discrete subsets for $\mathcal{D}$ and Ω respectively for computational purposes. The current interpolant of u through the n magic points is denoted by $\mathcal{I}_n[u]$.

Algorithm 2.1 The Empirical Interpolation Method

$\mu_1 = \arg\max_{\mu\in\Xi} \parallel u(\cdot;\mu) \parallel_{L^\infty(\Omega_d)}$
$u_1(\cdot) = u(\cdot;\mu_1)$
$x_1 = \arg\max_{x\in\Omega_d} |u_1(x)|$
$q_1(\cdot) = u_1(\cdot)/u_1(x_1)$
$B_1 = q_1(x_1)$
for $n = 2, \ldots, N \leq n_{\max}$ **do**
  $\mu_n = \arg\max_{\mu\in\Xi} \parallel u(\cdot;\mu) - \mathcal{I}_{n-1}[u](\cdot;\mu) \parallel_{L^\infty(\Omega_d)}$
  $u_n(\cdot) = u(\cdot;\mu_n)$
  $x_n = \arg\max_{x\in\Omega_d} |u_n(x) - \mathcal{I}_{n-1}[u](x;\mu_n)|$

$$q_n(\cdot) = \frac{u_n(\cdot) - \mathcal{I}_{n-1}[u](\cdot;\mu_n)}{u_n(x_n) - \mathcal{I}_{n-1}[u](x_n;\mu_n)}$$

  $(B_n)_{ij} = q_j(x_i), \quad 1 \leq i, j \leq n$
end for

At stage n we need the current interpolant $\mathcal{I}_{n-1}[u]$ for each parameter value $\mu \in \Xi$. We store the basis functions (which are vectors in the discrete case) in one big matrix $Q_{n-1} = [q_1 \mid q_2 \mid \ldots \mid q_{n-1}]$. The representation of the interpolant can now be expressed as $\mathcal{I}_{n-1}[u] = Q_{n-1}\underline{y}$, where the coefficient vector $\underline{y} \in \mathbb{R}^{n-1}$

is determined by solving the system

$$\sum_{j=1}^{n-1}(B_{n-1})_{ij}y_j = u(x_i), \quad i = 1, \ldots, n-1,$$

for all $\mu \in \Xi$. This construction ensures that we obtain interpolation at x_i, $i = 1, \ldots, n-1$.

The next basis element is chosen to be the parameter value that maximizes the difference between u and the current interpolant $\mathcal{I}_{n-1}[u]$,

$$\mu_n = \arg\max_{\mu\in\Xi} \parallel u(\cdot;\mu) - \mathcal{I}_{n-1}[u(\cdot;\mu)] \parallel_{L^\infty(\Omega_d)} .$$

An alternative way to look at it is that the next basis function is the function corresponding to the paremater value that the current interpolant is least suitable to approximate.

We let the position where the largest error occurs be our next magic point,

$$x_n = \arg\max_{x\in\Omega_d} \mid u_n(x) - \mathcal{I}_{n-1}[u](x;\mu_n) \mid,$$

and add that point to our set of interpolation nodes. As the interpolant $\mathcal{I}_{n-1}[u](x)$ by definition is equal to the underlying function at the magic points, the error at x_i, $i = 1, \ldots, n-1$, is equal to zero. The next normalized basis function is then

$$q_n(\cdot) = \frac{u_n(\cdot) - \mathcal{I}_{n-1}[u](\cdot;\mu_n)}{u_n(x_n) - \mathcal{I}_{n-1}[u](x_n;\mu_n)},$$

and all that is left to do is store the entries in B_n.

The final interpolant is given as

$$\mathcal{I}_N[u](x;\mu) = \sum_{i=1}^{N}\varphi_i(\mu)q_i(x),$$

where for every new $\mu \in \mathcal{D}$, we find the parameter coefficients by solving

$$\sum_{j=1}^{N}(B_N)_{ij}\varphi_j(\mu) = u(x_i;\mu), \quad i = 1, \ldots, N,$$

or in matrix notation

$$\mathcal{I}_N[u](x;\mu) = Q_N B_N^{-1}\underline{u}_N(\mu),$$

where $\underline{u}_N(\mu)$ is a vector containing u sampled at the magic points,

$$\underline{u}_N = \begin{bmatrix} u(x_1;\mu) \\ u(x_2;\mu) \\ \vdots \\ u(x_N;\mu) \end{bmatrix}.$$

It can be shown that this procedure indeed produces a valid interpolation scheme given by the basis functions $q_1, \ldots, q_N$ and interpolation nodes $x_1, \ldots, x_N$ [4].

We make a couple of remarks. First, each new magic point, at step n, does not effect the previous $n-1$ points. This is not surprising given the EIM algorithm, but remember that it is not at all the case for interpolation method based on predefined interpolation points. Often, if we want to extend an interpolant with one more interpolation point, we must change all the previous nodes and construct a whole new interpolant. The second remark is closely linked to the first: the EIM algorithm produces a sequence of hierarchical interpolation spaces $X_1 \subset X_2 \subset \ldots \subset X_N$, where $X_i = \text{span}\{q_1, \ldots, q_i\}$, which is a desired property. If we are forced to limit the number of basis functions to $\tilde{N} < N$, then for EIM the optimal choice is simply given by $q_1, \ldots, q_{\tilde{N}}$.

2.2 EIM error analysis

Since the EIM is problem dependent the general error analysis is both limited and often very pessimistic compared to the observed behaviour. However, the greedy selection in the algorithm gives an advantage as it immediately allows access to an *a posteriori* error estimate,

$$\| u - \mathcal{I}_{n+1}[u] \|_{L^\infty(\Omega)} \leq \| u - \mathcal{I}_n[u] \|_{L^\infty(\Omega)} .$$

This implies that we always know the maximum error for the previous interpolation space, which can be used to end the algorithm at a predefined tolerance, thus avoiding all $n_{\max}$ stages.

If we introduce the Lagrangian functions we can alternatively write the interpolant as

$$\mathcal{I}_N[u] = \sum_{i=1}^{N} \ell_i^M(x) u(x_i; \mu),$$

where $\ell_i^N(x) = \sum_{j=1}^{N} q_j(x)(B_N)_{ij}^{-1}$, and the Lagrangian functions are as usual a *nodal basis* for X_N, i.e. $\ell_i^N(x_j) = \delta_{ij}$. Then we make use of the Lebesgue constant defined as $\Lambda_N = \sup_{x \in \Omega} \sum_{j=1}^{N} \mid \ell_j^N(x) \mid$. For the EIM an upper bound for Λ_N is given by $2^N - 1$ [4]. Again, this is often a very pessimistic estimate, and observed behaviour is usually far better.

Lebesgue's lemma [4], which is a classical result in approximation theory, gives the following bound for the interpolation error

Lemma 2.1. *Assume X is a Banach space, and $X_N \subset X$, $\dim(X_N) = N$. For*

any $u \in X$, the interpolation error satisfies

$$\| u - \mathcal{I}_N[u] \|_X \leq (1 + \Lambda_N) \inf_{g_N \in X_N} \| u - g_N \|_X .$$

Here the *projection error*, $\inf_{g_N \in X_N} \| u - g_N \|_X$, is the best possible approximation of u in the approximation space X_N, measured by the norm $\| \cdot \|_X$ induced by X.

For the EIM Lemma 2.1 can be made more precise if a few conditions are fulfilled. Theorem 2.2 gives an upper bound for the interpolation error from the greedy algorithm,

Theorem 2.2. *Assume $\mathcal{U} \subset X \subset L^\infty(\Omega)$ and there exists a (possibly unknown) sequence of finite dimensional spaces*

$$\mathcal{Z}_1 \subset \mathcal{Z}_2 \subset \ldots \subset \mathcal{Z}_N \subset \text{span}(\mathcal{U}), \quad \dim(\mathcal{Z}_N) = N,$$

such that there exists $c > 0$ and $\alpha > \log(4)$ with for all $u \in \mathcal{U}$

$$\inf_{g_N \in \mathcal{Z}_N} \| u - g_N \|_X \leq c e^{-\alpha N}.$$

Then

$$\| u - \mathcal{I}_N[u] \|_{L^\infty(\Omega)} \leq c e^{-(\alpha - \log(4))N}.$$

A proof is given in appendix B in [4].

The theorem ensures that if there exists a finite dimensional space allowing for an exponentially converging approximation, the EIM will achieve an exponential rate of convergence. Also, if the spaces $\mathcal{Z}_i$ are not predetermined, the greedy algorithm provides such sequence through the EIM space X_N.

2.3 The Runge function

Theorem 2.2 tells us that under reasonable assumptions the EI procedure achieves an exponential rate of convergence. With the example below we present numerical results which verify this behaviour.

To see how the EIM works we will look at an example with a single scalar parameter, $\mu \in \mathcal{D}$, and one space dimension. The function we consider is the Runge function,

$$u(x; \mu) = \frac{1}{1 + \mu x^2},$$

where $\Omega = (-1, 1)$ and $\mathcal{D} = (1, 25)$; see Figure 2.1. This function is a classic in

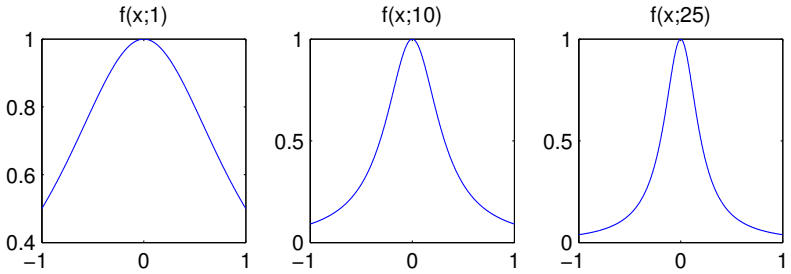

Figure 2.1: Runge function over $\Omega = (-1, 1)$, with $\mu = 1$, $\mu = 10$ and $\mu = 25$.

numerical interpolation, and it was shown by Runge that polynomial interpolation with equidistant (EQ) points will lead to divergence [8].

To compare we will consider Lagrange interpolation through equidistant points and through the GLL points [6]. This is done by creating a test set Q of 100 random μ-values drawn uniformly from the parameter domain $\mathcal{D}$. We then compute the maximum L^∞-error of the interpolant over all $\mu \in Q$,

$$e_N = \max_{\mu \in Q} \parallel u(\cdot;\mu) - \mathcal{I}_N[u](\cdot;\mu) \parallel_{L^\infty(\Omega)} . \tag{2.1}$$

The maximum error for the first 25 iterations can be seen in Table 2.1.

Table 2.1: Maximum L^∞-error over a sample of 100 random parameter values drawn uniformly from [1,25]. The table shows the error for the EIM- (e_{EIM}), GLL- (e_{GLL}) and equidistant (e_{EQ}) interpolation for the 25 first interpolation points.

n	e_{EIM}	e_{GLL}	e_{EQ}
5	1.175e-3	4.381e-1	4.380e-1
10	1.333e-7	2.944e-1	3.000e-1
15	3.256e-11	4.902e-2	7.153
20	5.551e-15	4.097e-2	8.523
25	9.992e-16	7.491e-3	254.495

As seen in Figure 2.2 both GLL interpolation the and EIM converges exponentially, but the EIM converges more rapidly and reach machine precision after 21 iterations. We also see how the the polynomial interpolation with EQ points oscillates and diverges as expected.

Why is the EIM superior to standard polynomial interpolation? For GLL interpolation the maximum error occurs for the biggest parameter value, $\mu \in Q$. The reason for this is that for this particular example the function forms a spike around $x = 0$, which is also where the GLL points have lowest density. The GLL

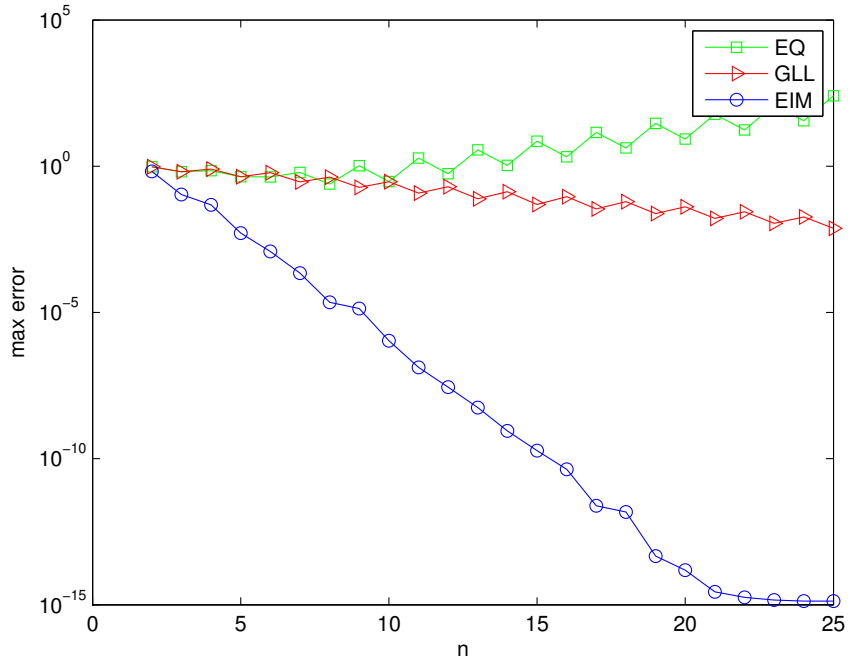

Figure 2.2: Maximum L^{∞}-error as a function of the interpolation points. Both GLL and EIM give an exponential convergence rate.

interpolation fails to pick up the sharp curvature around $x = 0$, and therefore do not achieve high precision. However, since the EIM always uses a greedy choice to determine the next magic point instead of predefined interpolation points, there will never be an area in the interpolation space that the algorithm can not "reach". As the interpolation space expand the maximum error will move around in the parameter space, and then be dealt with immediately. Also, because of the smooth parameter dependence we expect that the basis function for a given μ_n-value will also reduce the error in the neighbourhood of that μ_n. This approach, as seen numerically, gives far superior convergence.

In Figure 2.3 we have plotted the first 20 magic points and the first 5 basis functions the EIM algorithm constructs. Note that the magic points are all negative and have highest density near $x = 0$. This is where the changes in u are the greatest, and it shows the adaptive ability of the method. Another interesting feature of the EIM is that all the basis functions, as the underlying u we are approximating, are symmetric around $x = 0$. This is very desirable, and are not the case for the Lagrange interpolation polynomials based on GLL points.

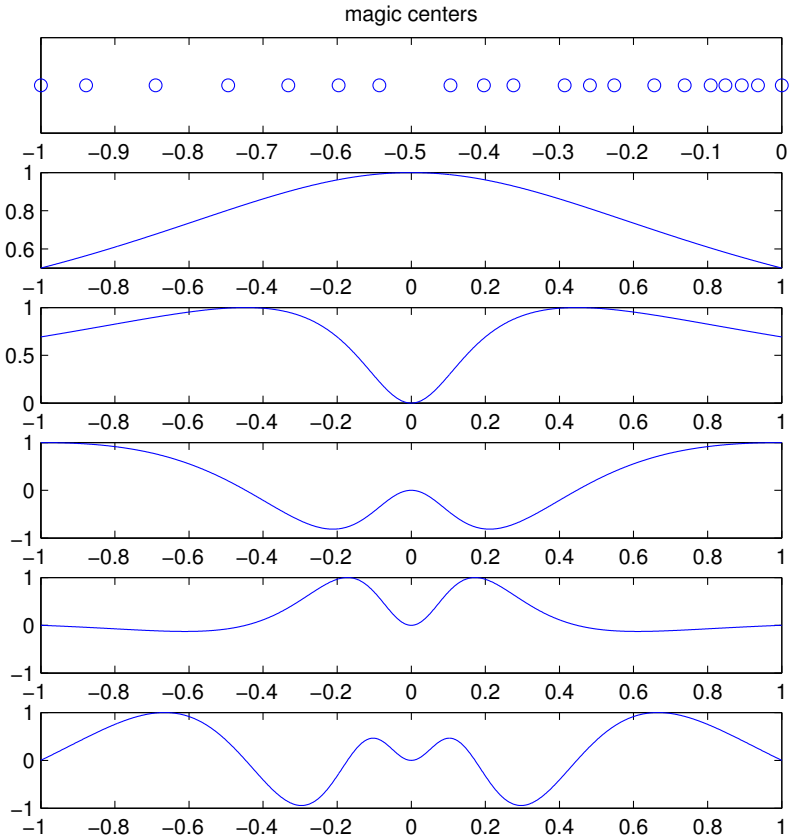

Figure 2.3: Distribution of the first 20 magic points over Ω and the first 5 basis functions. All the basis functions are symmetric around $x = 0$

2.4 The Adaptive cross approximation (ACA)

In [9] an alternative interpolation method is suggested, where Gaussian elimination is considered as an iterative algorithm. The method is equivalent to the well-known *Adaptive cross approximation* [10].

The method considers two space dimensions, which in case transforms to a single scalar parameter $\mu \in \mathcal{D}$ and one space dimension. Given a smooth function $u(x;\mu)$, the method is constructed so that it forms an approximation $\mathcal{I}_n[u]$ after n iterations that matches $u(x;\mu)$ exactly on at least n horizontal and n vertical lines, see Figure 2.4 for an example.

The algorithm for this method is given below, followed with an explanation of each step.

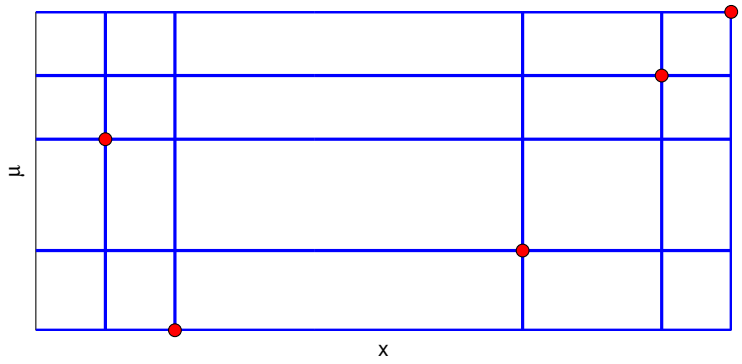

Figure 2.4: The n-th interpolant interpolate the function at n horizontal and n vertical lines through the interpolation space.

Algorithm 2.2 Adaptive cross approximation method

$\mu_1 = \arg\max_{\mu\in\Xi} \parallel u(x;\mu) \parallel_{L^\infty(\Omega_d)}$
$x_1 = \arg\max_{x\in\Omega_d} \mid u(x;\mu_1) \mid$
$p_1(\mu) = u(x_1;\mu)$
$\tilde{q}_1(x) = u(x;\mu_1)$
$c_1 = 1/u(x_1;\mu_1)$
$r_1(x;\mu) = u(x;\mu) - \mathcal{I}_1[f](x;\mu)$
for $n = 2,\ldots,N \leq n_{\max}$ **do**
  $\mu_n = \arg\max_{\mu\in\Xi_t} \parallel r_{n-1}(x;\mu) \parallel_{L^\infty(\Omega_d)}$
  $x_n = \arg\max_{x\in\Omega_d} \mid r_{n-1}(x;\mu_1) \mid$
  $p_n(\mu) = r_{n-1}(x_n;\mu)$
  $\tilde{q}_n(x) = r_{n-1}(x;\mu_n)$
  $c_n = 1/r_{n-1}(x_n;\mu_n)$
  $r_n(x;\mu) = u(x;\mu) - \mathcal{I}_n[f](x;\mu)$
end for

As for the EIM we use a greedy selection to pick μ_1 and x_1, but here we store two functions (instead of one for EIM), $p_1(\mu)$ and $\tilde{q}_1(x)$. We also store the inverse of the function evaluated at the interpolation point, $c_1 = 1/u(x_1;\mu_1)$. The first interpolant is found by

$$\mathcal{I}_1[u] = c_1 p_1(\mu)\tilde{q}_1(x),$$

an "outer product" of two univariate functions (*rank* 1 *functions*), corresponding to the "slices" $p_1(\mu) = u(x_1;\mu)$ and $\tilde{q}_1(x) = u(x;\mu_1)$. With the interpolant we can form the residual error $r_1(x;\mu) = u(x;\mu) - \mathcal{I}_1[u](x;\mu)$. By the same greedy selection on $r_1(x;\mu)$ we determine the next interpolation point, parameter variable, two new functions $p_2(\mu)$ and $\tilde{q}_2(x)$, and c_2.

The interpolant for each iteration is found by summing the last interpolant with the current c_n times a new "outer product" of the new functions,

$$\mathcal{I}_n[u](x;\mu) = \mathcal{I}_{n-1}[u](x;\mu) + c_n p_n(\mu)\tilde{q}_n(x).$$

Note that the algorithm produces the maximum error over the domain at each iteration, which is $r_{n-1}(x_n;\mu_n) = 1/c_n$, so by storing the c_n-s we are also storing information on the maximum error.

The iterations continue until it has reached some predefined $n_{\max}$ or a desired accuracy in the approximation.

The final interpolant after N iterations is given by the sum

$$\mathcal{I}_N[u](x;\mu) = \sum_{i=1}^{N} c_i p_i(\mu)\tilde{q}_i(x),$$

and we have a rank N approximation that matches $u(x;\mu)$ exactly on at least N horizontal and N vertical lines.

We want to compare the ACA to the EIM, so we will apply the algorithm again on the Runge function and compare the maximum error to see which one of them approximates $u(x;\mu)$ the best. Table 2.2 shows the maximum error for the two methods. From the table we see that the two methods approximates the function

Table 2.2: Maximum L^∞-error for two methods.

n	e_{EIM}	e_{ACA}
2	0.1085	0.1085
4	0.0052	0.0052
6	2.221e-4	2.221e-4
8	1.364e-5	1.364e-5
10	1.333e-7	1.333e-7

with the same accuracy. Figure 2.5 shows a plot of the maximum error.

It can be shown that the two methods chooses the same interpolation points. Further, it can also be shown that the EIM and the ACA are in fact equivalent, the EIM also interpolate exactly for all the chosen μ_i-s, i.e. $\mathcal{I}_n[u](x;\mu_i) = u(x;\mu_i)$, for all $x \in \Omega$, $i = 1,\ldots,n$. This property is somewhat surprising since it is not in any way obvious from the EIM algorithm, and has previously not been appreciated.

The two methods use the same interpolation points, and therefore approximate the function with the same accuracy. In implementation, however, there are differences. The EIM stores one vector $q_i(x)$ for each iteration, while the other stores

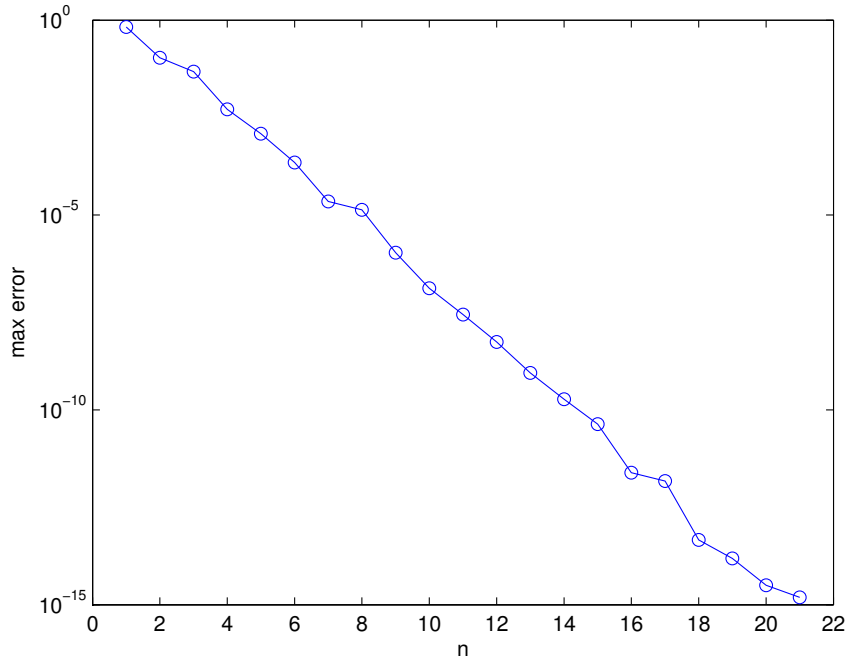

Figure 2.5: The maximum error is the same for the two methods.

two, $p_i(\mu)$ and $\tilde{q}_i(x)$. Also, in the EIM we are required to solve the system

$$\sum_{j=1}^{n}(B_n)_{ij}y_j = u(x_i;\mu), \quad i = 1,\ldots,n,$$

for all $\mu \in \Xi$ in each iteration, while in the ACA we compute an outer vector product.

In the end the EIM has one advantage over the ACA: in the EIM we can compute the final interpolant for all $\mu \in \mathcal{D}$, while for the ACA we are only able to find the interpolant by taking a "slice" of $\mathcal{I}_N[u](x;\mu)$ at say $\tilde{\mu}$, where $\tilde{\mu}$ have to be in the discrete set Ξ.

2.5 Multiple dimensions

Interpolation in the one dimensional setting is quite well documented. There exist many results for the best points and basis functions under different optimality conditions, especially for polynomial interpolation. In multiple dimensions, however, the situation is more open and complex. Some of the results from the scalar case

can be generalized, but it is typically harder to find optimal points as the number of dimensions increase.

As in the one dimensional case the EIM still offers a simple way to determine interpolation nodes and basis functions for higher dimensions. Also, if the conditions in Theorem 2.2 is fulfilled the EIM still achieves exponential convergence. The downside is of course that when the number of dimensions increases we usually require a large number of terms in our EIM expansion to get a satisfying approximation of the underlying function.

We consider the two dimensional function

$$u(x;\mu) = \frac{1}{\sqrt{(x_1-\mu_1)^2 + (x_2-\mu_2)^2}}, \tag{2.2}$$

in $\Omega = (0,1)^2$, where $x = (x_1, x_2)$, $\mu = (\mu_1, \mu_2)$ and $\mathcal{D} = (-1, -0.01)^2$. Figure 2.6 shows the function, with $\mu = (-0.1, -0.1)$. For the test set, Ξ, we use a

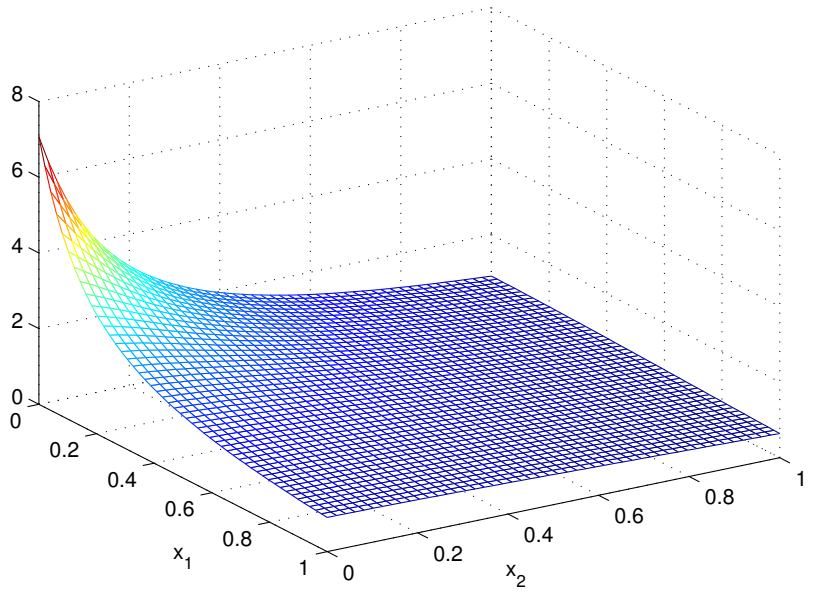

Figure 2.6: $u(x;\mu)$ with $\mu = (-0.1, -0.1)$

uniform distribution of 50 points in each direction which gives a total of $50^2 = 2500$ parameter values. This time we represent each of the EIM basis functions with a tensor product based on 75×75 GLL points.

In Figure 2.7 we have compared the L^2-error associated with the EIM and GLL interpolation. Clearly, the EIM is superior to the GLL approach. The n

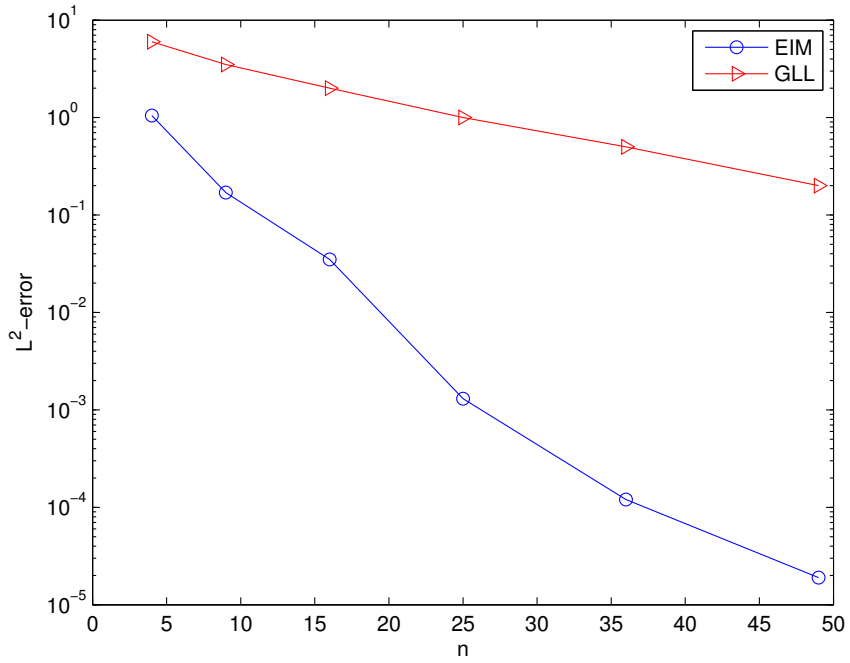

Figure 2.7: L^2-error over a sample of 500 random parameter values drawn uniformly from the parameter domain $\mathcal{D}$ for EIM and GLL. The EIM is clearly superior.

given along the horizontal axis is the total number of interpolation points, and the error is computed for 2 to 7 points in each spatial direction.

Since the interpolation points are not predetermined for the EIM, the nodes can be concentrated in areas where the function is irregular, as seen in Figure 2.8. The parameter values and magic points selected by the greedy selection is clustered at the corners $\mu = (-0.01, -0.01)$ and $x = (0, 0)$. Where the EIM adapts to the specific problem, GLL interpolation will waste resources by covering parts of Ω where only a few nodes are necessary.

Assume we have constructed the EIM by the description over and we now want to evaluate the integral

$$I(\mu) = \int_\Omega u(x_1, x_2; \mu_1, \mu_2) \mathrm{d}x_1 \mathrm{d}x_2.$$

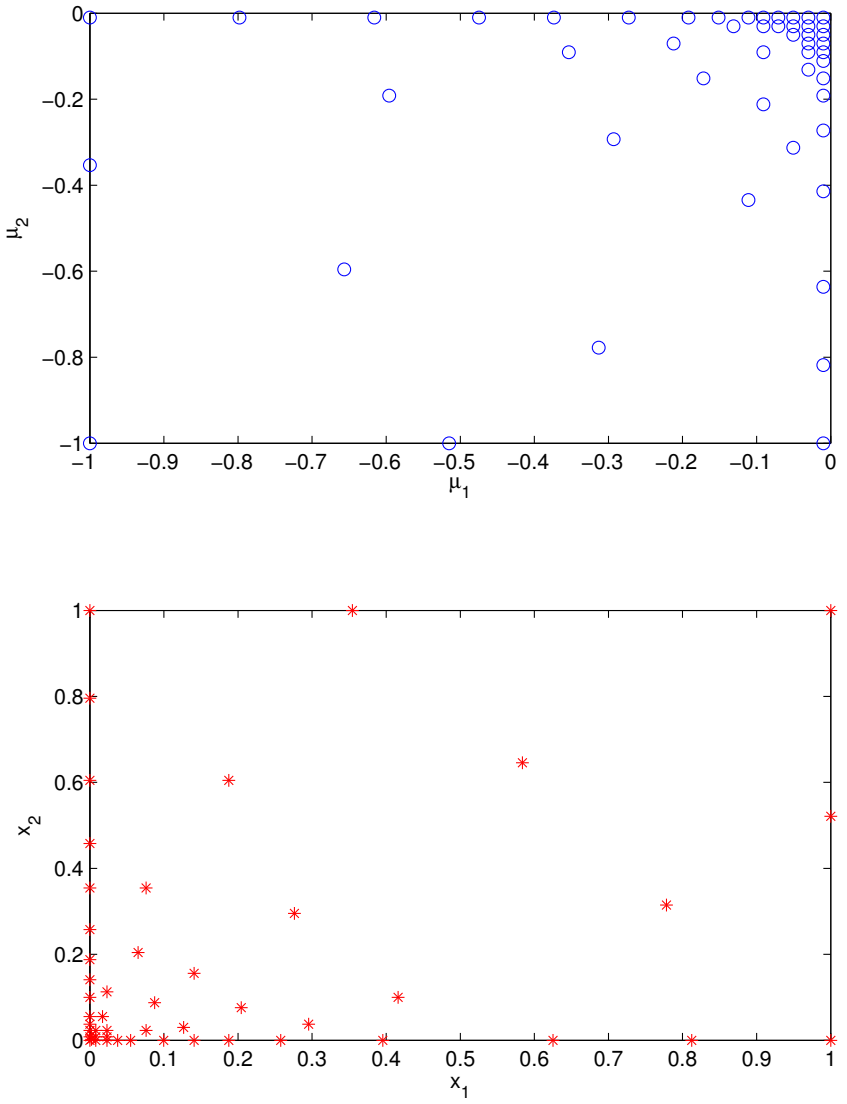

Figure 2.8: Parameter samples (top) and magic points (below) selected by the EIM algorithm. The spatial points are clearly concentrated near the corner points $\mu = (-0.01, -0.01)$ and $x = (0, 0)$.

By the EIM approximation we can approximate the integral by

$$\begin{aligned} I(\mu) &\approx \int_\Omega \sum_{j=1}^{N} \varphi_j(\mu) q_j(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 \\ &= \sum_{j=1}^{N} \varphi_j(\mu) \int_\Omega q_j(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 \\ &= \sum_{j=1}^{N} \varphi_j(\mu) \omega_j, \end{aligned}$$

where $\omega_j = \int q_j(x) \mathrm{d}x$. Since the basis functions are represented on a GLL grid we can now approximate ω_j by the GLL quadrature from Section 1.1. If we use GLL quadrature directly we would need to calculate a new integral approximations for each new parameter value. However, if we first run the EIM algorithm we can precalculate the ω_j from the basis functions, and for each parameter value we need to find the parameter dependent coefficients $\varphi_j(\mu)$ by solving the system

$$\sum_{j=1}^{N} (B_M)_{ij} \varphi_j(\mu) = u(x_i; \mu), \quad i = 1, \ldots, N. \tag{2.3}$$

Next, we can use the EIM algorithm to approximate the integral by

$$I(\mu) \approx \sum_{j=1}^{N} \varphi_j(\mu) \omega_j. \tag{2.4}$$

In Figure 2.9 we have used the EIM to approximate the function from (2.2) given over, and compared it to GLL quadrature. The maximum error given is the differences between our approximations and analytical solution. For GLL the x-axis represents the total number of points, $\mathcal{N} \times \mathcal{N}$, in the GLL grid, while for EIM the lower axis corresponds to the total number of magic points, N, from (2.4).

In terms of computing costs, using GLL directly goes as $\mathcal{O}(\mathcal{N}^2)$ when there are $\mathcal{N}$ GLL points in each spatial direction in the two dimensional case. Alternatively we first use EIM and compute the ω_j-s, which goes as $\mathcal{O}(N\mathcal{N}^2)$. After the ω_j values are found, we only have to find the parameter dependent coefficients in (2.3) for each new parameter value, which goes as $\mathcal{O}(N^3)$, where N is the total number of magic points used. Since typically $N << \mathcal{N}$ this approach may be preferable if we need to evaluate the integral a large number of times for many values of μ.

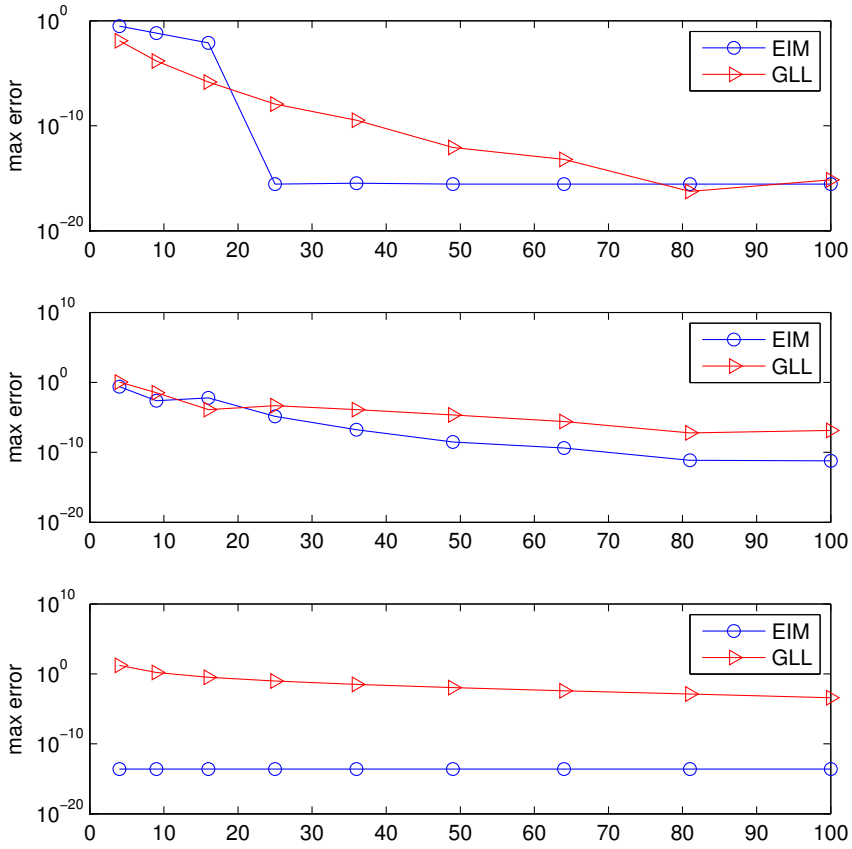

Figure 2.9: Maximum error for GLL quadrature and when we use EIM in interpolation approximation. We have used $\mu = (-1, -1)$ (top), $\mu = (-0.1, -0.1)$ (middle) and $\mu = (-0.01, -0.01)$ (bottom).

Chapter 3

Estimation of parameters from noisy data

Consider the case where we measure a system $u(x;\mu^*)$ in different spatial points which we are free to choose, but our measurements are polluted by some random noise. From the observations we want to estimate the underlying parameter value μ^*. However, we will not be able to determine μ^* exactly because of the noise. Naturally some measurement points will provide more information on the parameter value than others. We are working under the hypothesis that the magic points produced in the EIM algorithm are generally a better choice than for instance just using equidistant points.

We have to make certain assumptions on the introduced noise, which in turn induce the statistical framework we will use. Since the measurements are polluted by noise, there might be several parameter values that could explain the results. We show how we can construct a set of parameter values that are consistent with the experimental data. Further, we expect the size of the set with consistent parameter values to be dependent on where we measure the system, which we will confirm by numerical observations. We will try to explain why some measurement points are better than other by looking at analytical expressions for the Runge function, and the sensitivity of the function at the points where we take our measurements.

3.1 Statistical framework

We assume that the measured experimental data is on the form

$$Z'_k = u(x_k;\mu^*) + \epsilon'_k, \quad 1 \leq k \leq n,$$

where n is the predefined number of measurement points, and $\epsilon'_k \sim \mathcal{N}(0,\sigma^2)$, $1 \leq k \leq n$, is normal, zero-mean, uncorrelated and with standard deviation σ [11].

Throughout the report we are going to use $\sigma = 0.02$ in the numerical experiments. We then define m' independent realizations of Z'

$$Z'_{k;i} = u(x_k; \mu^*) + \epsilon'_{k;i}, \quad 1 \leq k \leq n, \quad 1 \leq i \leq m',$$

so there is a total of nm' measurements available. We will refer to the nm' measurements as one experiment. Before each individual experiment neither μ^* nor σ are known, and we want to estimate these.

We let $\overline{Z} \in \mathbb{R}^n$ be the average measured values in the observation points

$$\overline{Z}_k = \frac{1}{m'} \sum_{i=1}^{m'} Z'_{k;i}, \quad 1 \leq k \leq n$$

Next, the standard deviation can be estimated by

$$\hat{\sigma} = \sqrt{\frac{1}{nm' - n} \sum_{k=1}^{n} \sum_{i=1}^{m'} (Z'_{k;i} - \overline{Z}_k)^2}.$$

We let

$$\hat{\rho} = \hat{\sigma} \sqrt{n \mathcal{F}^{-1}(n, nm' - 1, \gamma)},$$

where $\mathcal{F}^{-1}(d_1, d_2, \gamma)$ is the F-statistic γ quantile for d_1 and d_2 degrees of freedom, and we let V be a vector containing u sampled at the measurement point with the underlying μ^*,

$$V = \begin{bmatrix} u(x_1; \mu^*) \\ u(x_2; \mu^*) \\ \vdots \\ u(x_n; \mu^*) \end{bmatrix}.$$

If we define $\| \cdot \|$ as the Euclidean norm, we can state

Proposition 3.1. *With confidence level γ,*

$$\| V - \overline{Z} \| \leq \frac{\hat{\rho}}{\sqrt{m'}}.$$

A sketch of the proof is given in [11].

Proposition 3.1 is great to test if our numerical implementation is correct. By doing several experiments and checking if the inequality is satisfied, it should hold the same number of times corresponding to our choice of the confidence level γ. Note that the underlying μ^*, and hence the vector V, is in principle unknown.

3.2 Parameter estimation

Let Υ be a fine discrete set of parameter values over the domain $\mathcal{D}$. For simplicity we let $\Upsilon = \Xi$ and we assume that the underlying parameter value μ^* is in Υ. Let

$$\tilde{V}_k = u(x_k; \tilde{\mu}_p), \quad 1 \leq k \leq n, \tag{3.1}$$

where $\tilde{\mu}_p \in \Upsilon$. If we loop over all parameter values $\tilde{\mu}_p \in \Upsilon$, and check if the inequality

$$\| \tilde{V} - \overline{Z} \| \leq \frac{\hat{\rho}}{\sqrt{m'}} \tag{3.2}$$

is satisfied, we get a set Υ_{con} containing parameter values that are consistent with the experimental data. By the statistical framework, the underlying value μ^* is then in the estimation set Υ_{con} the same number of times corresponding to the choice of γ.

Now that the regression framework is set, one question remains: in what points do we want to measure? In the example below we will see that the size of Υ_{con} depends on the measurement points, and how the magic points from the EIM are good candidates. Note that all we use from the EIM are the magic points, and that parameter estimation otherwise has nothing to do with function approximation.

3.3 The Runge function

In Section 2.3 we shown how the EIM was superior to GLL interpolation in function approximation. Now we want to use the statistical framework over to estimate given parameters.

In the numerical experiments we let N_{exp} be the total number of independent experiments, and for each experiment we randomly choose a parameter value $\mu^* \in \Upsilon$ to estimate. Further we let $E(|\Upsilon_{\text{con}}|)$ be the average size of the estimation set,

$$E(|\Upsilon_{\text{con}}|) = \frac{1}{N_{\text{exp}}} \sum^{N_{\text{exp}}} |\Upsilon_{\text{con}}|,$$

so that the quantity $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$ represent the average sharpness of our estimation.

Before we start, the Runge function has one improtant property to notice, the EIM produce $x_1 = 0$, where x_i is the i-th magic point. However, for the Runge function we have

$$u(x_1; \mu) = 1,$$

for all $\mu \in \mathcal{D}$, i.e. that point alone gives no information on the underlying μ^* whatsoever. Since the example is rather simple, containing a single scalar parameter

and one space dimension, we can show analytically that the second magic point is at $x_2 = -1/\sqrt{5}$, so when we measure the system in only one point we will use that one.

First we do measurements at only $x = -1/\sqrt{5}$ and compare this to measurements at only $x = -1$. We will also do measurements at several magic points, $n = 2, \ldots, 10$. The "manually" chosen point $x = -1$ has been used for two reasons: it is the third magic point the EIM produces, and it could have easily been chosen as an observation point if we had no prior knowledge of parameter estimation.

We start off by trying to approximate $\mu^* = 10$ with the two single points; see Table 3.1. The table shows that measuring in $x = -1/\sqrt{5}$ gives a sharper estimate

Table 3.1: The number of parameters that are consistent with the experimental data compared with the total number of elements in the discrete set Υ at two different measurement points, with $m' = 20$, $\sigma = 0.02$ and $\gamma = 0.95$. A total of 1000 independent experiments are preformed.

measurement point	$E(\lvert\Upsilon_{\text{con}}\rvert)/\lvert\Upsilon\rvert$
$x = -1$	0.095
$x = -1/\sqrt{5}$	0.035

than $x = -1$, and note that the precision is approximately three times better for $x = -1/\sqrt{5}$ than for the other.

The question now is why is $x = -1/\sqrt{5}$ a better measurement point than $x = -1$ for this example? We consider the derivative

$$\frac{\partial u(x;\mu)}{\partial \mu} = -\frac{x^2}{(1+\mu x^2)^2}, \tag{3.3}$$

which represent the functions sensitivity. A plot of the absolute value of $\partial u/\partial \mu$ at different points in Ω can be seen in Figure 3.1.

Ideally we want to measure the function where the absolute value of (3.3) realized with μ^* is maximized, but since μ^* in principle is unknown this is not possible. With $\mu^* = 10$ and $x = -1$ we get

$$\left|\frac{\partial u}{\partial \mu}\right|_{x=-1} = \frac{1}{121} \approx 0.008,$$

and with $x = -1/\sqrt{5}$

$$\left|\frac{\partial u}{\partial \mu}\right|_{x=-1/\sqrt{5}} = \frac{1}{45} \approx 0.022.$$

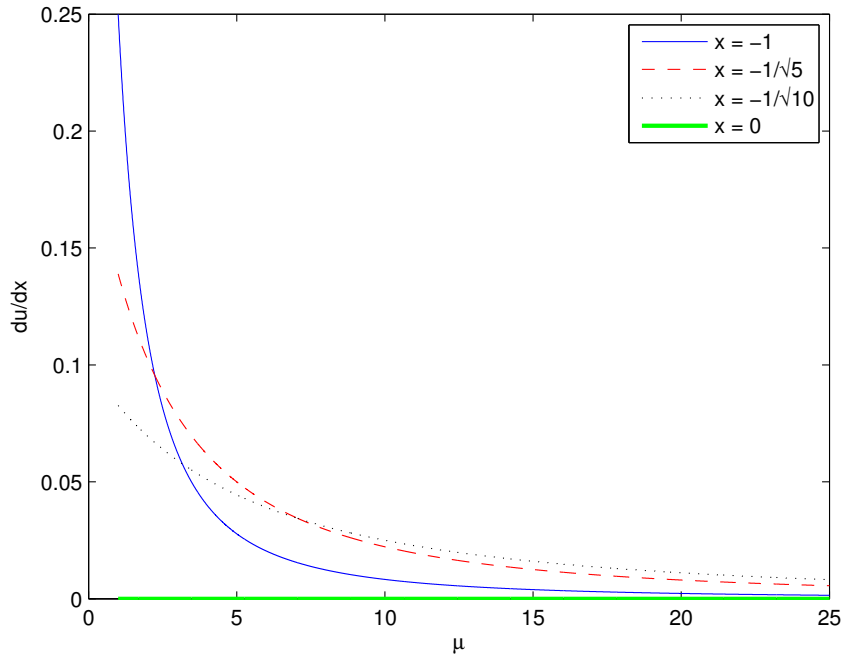

Figure 3.1: The sensitivity of $u(x;\mu)$ at different points in Ω. We see that for $\mu^* = 10$ the function is more sensitive at $x = -1/\sqrt{5}$ than at $x = -1$, and therefore gives higher precision. Also, we see that for $x = 0$ the function does not change as expected.

So, $x = -1/\sqrt{5}$ is a better measurement point than $x = -1$ since the function is more sensitive at that point for $\mu^* = 10$. Remember how $x = -1/\sqrt{5}$ gave approximately three times better precision than $x = -1$ in Table 3.1? We see the same ratio here. The quality of our measurements is a reflection of how sensitive the function is at a given point and for a given parameter value μ^*.

If we where to measure the system at just one point, the absolutely optimal point would be at $x = -1/\sqrt{10}$, since

$$\left|\frac{\partial u}{\partial \mu}\right|_{x=-1/\sqrt{10}} = \frac{1}{40} = 0.025,$$

which is consistent with the plot in Figure 3.1. However, the difference in the sensitivity between $x = -1/\sqrt{10}$ and $x = -1/\sqrt{5}$ are relatively small, and since we do not know the underlying μ^* from the beginning $x = -1/\sqrt{5}$ is without a doubt a pretty good measurement point in this case.

More generally we randomly choose a parameter value $\mu^* \in \Upsilon$ to estimate in each individual experiment. The result is given in Table 3.2. In this case,

Table 3.2: Estimation sharpness using the measuring at the magic points provided by the EIM. Here $m' = 5$ and $\sigma = 0.02$.

n	$E(\lvert\Upsilon_{\text{con}}\rvert)/\lvert\Upsilon\rvert$
2	0.128
3	0.125
4	0.088
5	0.086
6	0.085
7	0.069
8	0.071
9	0.063
10	0.060

when we measure in 10 magic points, we can on average tell the parameter value with an accuracy of 0.060, or 6.0% of the parameter domain. Note that the estimation sharpness here is lower than when we only use the one observation point at $x = -1/\sqrt{5}$ in Table 3.1. The reason is that previously we used $m' = 20$, while now we have $m' = 5$.

Chapter 4

The Generalized EIM (GEIM)

To generalize the EIM assume we are given a parameter dependent function $u(x;\mu)$, and imagine that the only way we are able to observe that function is through some filter. The filter is applied by introducing a linear functional version of EIM [12]. When we are dealing with less regular functions we typically want a filter that smooths it out, with the advantage to more robustly capture the behaviour. Later we will experiment with different filters to see how they effect given functions.

Aside from the filter, the GEIM algorithm is similar to standard EIM. However, the implementation we use is a little different, so we give the algorithm in full. Then we give the details on how to apply our statistical framework from Section 3.1 after the GEIM procedure, in parameter estimation problems.

4.1 A linear functional EIM

We consider the case where we are given a parametric manifold of functions, $\mathcal{M} = \{u(\cdot;\mu)|\mu \in \mathcal{D}\}$, where for a given μ in the domain $\mathcal{D}$ the field $u(\cdot;\mu)$ is a function in $\mathcal{X}(\Omega)$. Then we define a set of linear functionals, $\ell_j : \mathcal{X} \to \mathbb{R}$, $1 \leq j \leq J$, on the form

$$\ell_j(v) = \int_\Omega g(x - x_j)v(x)\mathrm{d}x, \quad v \in \mathcal{X}, \tag{4.1}$$

where the function g is the filter we use to observe the function v. The functionals represent what we actually observe, and we will refer to these as "magic functionals". Further, each magic functional ℓ_j is associated to a functional center $x_j \in \Omega$, which we will denote "magic centers". Examples of the filer g could be

$$g(x - x_j) = \delta(x - x_j), \tag{4.2}$$

and

$$g(x - x_j) = \frac{1}{\sqrt{2\pi}\sigma_m} e^{\frac{-(x-x_j)^2}{2\sigma_m^2}}. \tag{4.3}$$

We denote (4.3) as a "Gaussian filter", which we can adjust by changing the filter width σ_m. Higher σ_m will result in smoother functionals. Note that the Gaussian filter given here corresponds to one space dimension. When we are not able to evaluate the integral in (4.1) exactly, the functionals will be approximated by some quadrature, for example the GLL quadrature described in Section 1.1. Also note that in the special case of (4.2) the GEIM is equivalent to the EIM.

From an approximation perspective the magic functionals will span a space which is in some sense close to the full set of functionals, in other word, from a linear combination of the subset we should be able to construct an interpolant for all $\mu \in \mathcal{D}$. From an estimation perspective the magic functionals will provide good discrimination between members of the the manifold, i.e. different values of $\mu \in \mathcal{D}$. Later we will investigate which σ_m are preferable in parameter estimation for a given example.

We introduce a discrete training set Ξ of P points $\mu_p \in \mathcal{D}$, $1 \le p \le P$. We let $u_p = u(x; \mu_p)$ be the "snapshots" of the function for all points in Ξ, and evaluate $H_{jp} = \ell_j(u_p)$, $1 \le j \le J$, $1 \le p \le P$. We also specify $n_{\max}$ as the number of magic functionals desired. The GEIM algorithm is given below.

Algorithm 4.1 Generalized Empirical Interpolation Method

set $\tilde{j}_1, \tilde{p}_1 = \arg\max_{j,p} |H_{jp}|$
set $(q_1)_j = \ell_j(u_{\tilde{p}_1}) / H_{\tilde{j}_1, \tilde{p}_1}, \quad 1 \le j \le J$
set $B^1 \in \mathbb{R} : \quad B^1_{11} = 1$
for $i = 1, \ldots, n \le n_{\max} - 1$ **do**
  for $p = 1, \ldots, P$ **do**
    find $\alpha^i_p \in \mathbb{R}^i : B^i \alpha^i_p = \begin{pmatrix} H_{\tilde{j}_1, p} \\ \vdots \\ H_{\tilde{j}_i, p} \end{pmatrix}$
    evaluate $(r^i_p)_j = H_{jp} - \sum_{i'=1}^{i} (\alpha^i_p)_{i'} (q_{i'})_j, \quad 1 \le j \le J$
  end for
  set $\tilde{j}_{i+1}, \tilde{p}_{i+1} = \arg\max_{j,p} |(r^i_p)_j|$
  evaluate $\tau^i = (r^i_{\tilde{p}_{i+1}})_{\tilde{j}_{i+1}}$
  define $(q_{i+1})_j = (r^i_{\tilde{p}_{i+1}})_j / \tau^i, \quad 1 \le j \le J$
  construct $B^{i+1} \in \mathbb{R}^{i+1} : B^{i+1}_{kk'} = (q_{k'})_{\tilde{j}_k}$
end for

As for the EIM, at iteration i the algorithm chooses the magic center and functional least well represented by the previous magic centers and functionals measured by the error τ^i. The same error measurement could also be used to stop the iterations after a desired tolerance is achieved, thus avoiding all $n_{\max}$ stages.

After the algorithm is applied we are left with a GEIM system of size n

comprised by the magic centers $x_{\tilde{\jmath}_k}$, the magic functionals $\ell_{\tilde{\jmath}_k}$, $1 \leq k \leq n$, the basis functions q_i, $1 \leq i \leq n$, the associated approximation space $W^n = \text{span}\{q_1, \ldots, q_n\}$, and the interpolation matrix $B \in \mathbb{R}^{n \times n}$.

Note that the nested construction, as for the EIM, gives us a hierarchy of interpolation spaces $W^1 \subset W^2 \subset \ldots \subset W^n$, where $W^i = \text{span}\{q_1, \ldots, q_i\}$, which implies that for any $\tilde{n}$, $1 \leq \tilde{n} \leq n$ we have a GEIM system of size $\tilde{n}$.

Next, assume we are given a function $v \in \mathcal{X}(\Omega)$. We let $L \in \mathbb{R}^J$ so that

$$L_j = \ell_j(v), \quad 1 \leq j \leq J, \tag{4.4}$$

and $V \in \mathbb{R}^n$ so that

$$V_k = L_{\tilde{\jmath}_k}, \quad 1 \leq k \leq n, \tag{4.5}$$

i.e. V contains the observations of the function $v \in \mathcal{X}(\Omega)$. The GEIM system then provides an approximation $\tilde{L}$ to L based on V, by computing $\tilde{\beta} \in \mathbb{R}^n$ from

$$B\tilde{\beta} = V, \tag{4.6}$$

and then

$$\tilde{L}_j = \sum_{i=1}^{n} \tilde{\beta}_i (q_i)_j, \quad 1 \leq j \leq J.$$

By construction B is a lower triangular matrix with $B_{ii} = (q_i)_{\tilde{\jmath}_i} = 1$ on the diagonal, thus it follows that (4.6) has a unique solution. As a result of this construction the GEIM approximation satisfies

$$\tilde{L}_j = L_j, \quad j \in \{\tilde{\jmath}_1, \ldots, \tilde{\jmath}_n\}, \tag{4.7}$$

i.e. we have interpolation in the magic centers, which is the GEIM interpolation property.

Note that a GEIM approximation can be constructed for all $v \in \mathcal{X}(\Omega)$, even though the algorithm trains only on function from the manifold $\mathcal{M}$. So in a sense we may say that B and q_i, $1 \leq i \leq n$, bear a $\mathcal{M}$ "label". However, the interpolation property (4.7) shall still hold for all $v \in \mathcal{X}(\Omega)$.

We make a short remark on the notation. In the EIM the magic centers was denoted x_j, $1 \leq j \leq N$, while in GEIM x_j, $1 \leq j \leq J$ represent the functional centers. The magic centers in the GEIM is denoted $x_{\tilde{\jmath}_k}$, $1 \leq k \leq n$.

4.2 Parameter estimation: GEIM

As mentioned earlier the introduced linear functionals allow us to capture the behaviour of less regular functions, which in turn enable us to estimate given

parameters from the underlying function. When we observe the function we now use the magic centers provided by the GEIM algorithm.

In this case we do measurements on the filtered $u(x;\mu^*)$, and we assume that the measured experimental data this time is on the form

$$Z'_k = V_k + \epsilon'_k, \quad 1 \leq k \leq n,$$

where V is defined in (4.4) and (4.5), and $\epsilon'_k \sim \mathcal{N}(0,\sigma^2)$, $1 \leq k \leq n$, is again normal, zero-mean, uncorrelated and with standard deviation σ. We define m' independent realizations of Z' in the same way,

$$Z'_{k;i} = V_k + \epsilon'_{k;i}, \quad 1 \leq k \leq n, \quad 1 \leq i \leq m'.$$

Here we assume need to assume that V is constructed from a function $u \in \mathcal{M}$, and again we assume that $\mu^* \in \Upsilon = \Xi$. The vector $\tilde{V}$ from Equation (3.1) we use to determine the set Υ_{con} now is constructed from the functionals,

$$\tilde{V}_k = \ell_{\tilde{j}_k}(u(\cdot;\tilde{\mu}_p)), \quad 1 \leq k \leq n,$$

where $\tilde{\mu}_p \in \Upsilon$, $1 \leq p \leq P$.

The rest of the statistical framework and how to estimate parameters is given in the previous chapter.

4.3 Numerical examples

Below we examine different examples with irregular functions, both in one and multiple space dimensions. For each example we apply the GEIM coupled with a given filter.

For each example we first consider the error behaviour after filter use on a given function, by considering the maximum error $|\tau^i|$ provided by the GEIM algorithm.

Second we explore how the magic centers can be used in parameter estimation. Remember that the average sharpness of our estimation is given by $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$, defined by

$$E(|\Upsilon_{\text{con}}|) = \frac{1}{N_{\text{exp}}} \sum^{N_{\text{exp}}} |\Upsilon_{\text{con}}|.$$

Note that in function approximation we use fine grids for both the space domain and the parameter domain, while for parameter estimation we will see that we can make due with a coarse grid for the space domain.

4.3.1 Gaussian filter

The Gaussian filter described in (4.3) smooths out a given function in a relatively intuitive manner, and is therefore great to use as an introduction to how a filter may affect a function.

Example 1: One dimensional step function

For our first example we consider the one dimensional function

$$u(x;\mu) = \begin{cases} 1 & -1 \leq x \leq \mu, \\ 0 & \mu < x \leq 1, \end{cases}$$

where $\Omega = (-1, 1)$ and $\mathcal{D} = (-0.5, 0.5)$; see Figure 4.1. The function is partly

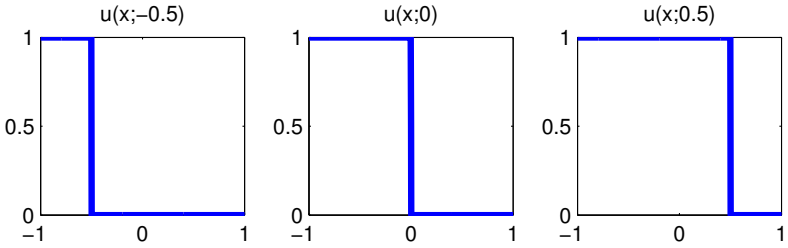

Figure 4.1: $u(x;\mu)$ over $\Omega = (-1, 1)$ with $\mu = -0.5$, $\mu = 0$ and $\mu = 0.5$.

chosen for its simple irregularity and will provide a clear picture to what happens after filter use.

We apply the Gaussian filter (4.3) with $\sigma_m = 0.2$ on a fine grid, the filtered function can be seen in Figure 4.2. The figure shows how the filter smooths out

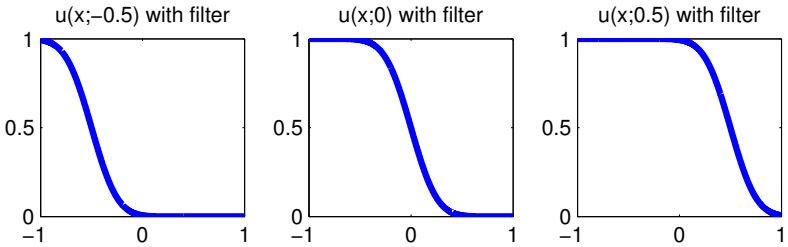

Figure 4.2: $u(x;\mu)$ with Gaussian filter with $\sigma_m = 0.2$, and $\mu = -0.5$, $\mu = 0$ and $\mu = 0.5$. The parameter μ here represent the position of the discontinuity in u.

the function, which can now be approximated by the GEIM. If we increase the filter width σ_m, the resulting functionals after filter use will give an even smoother function. We expect a smoother function to be easier to approximate, which is confirmed in Figure 4.3. The maximum error $|\tau^i|$ is plotted for three different σ_m-s, and the iterations is stopped when $|\tau^i| < 10^{-14}$.

Next we want to see if we are able to estimate a given parameter value μ^* when we add noise according to Section 3.1, and why using a filter could be an advantage on a coarse grid. Imagine we are measuring the function $u(x;\mu^*)$, where $x_j \leq \mu^* < x_{j+1}$. With no filter we get no additional information on μ^*. However,

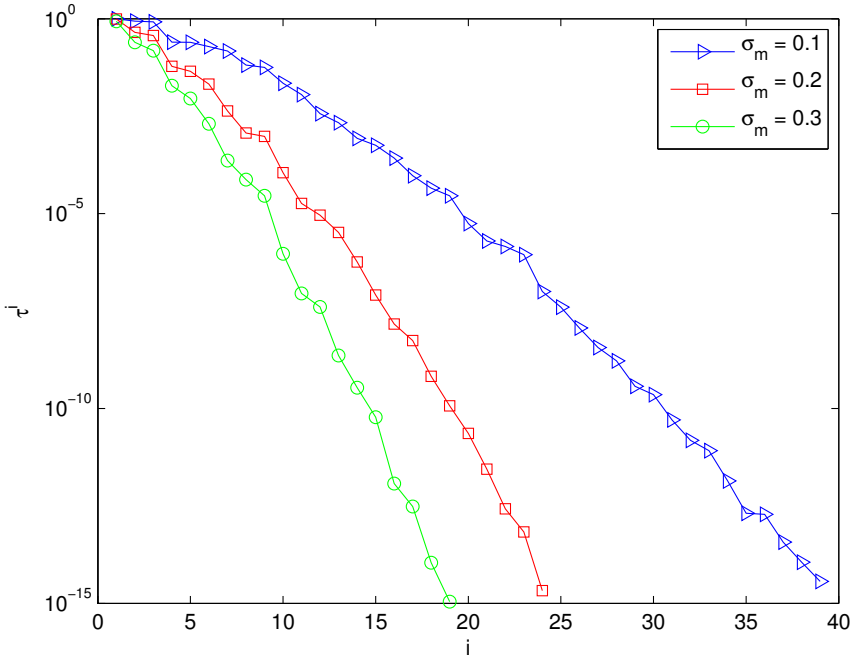

Figure 4.3: With $\sigma_m = 0.10$ the GEIM reaches the tolerance $|\tau^i| < 10^{-14}$ after 39 iterations, $\sigma_m = 0.20$ after 24 iterations and $\sigma_m = 0.30$ after 18 iterations.

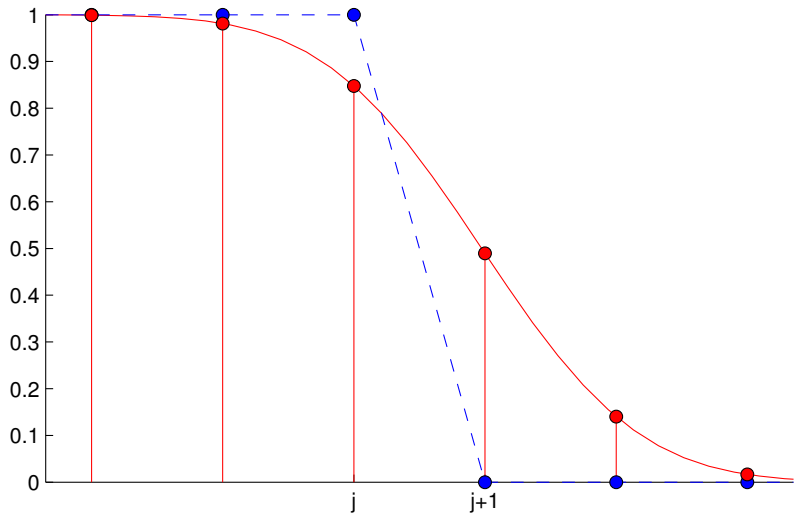

Figure 4.4: Blue dots; $u(x;\mu^*)$ measured without filter and without noise. We can not say anything about the parameter value between x_j and x_{j+1}. Red dots; $u(x;\mu^*)$ measured with filter and without noise.

if we use a Gaussian filter we do, see Figure 4.4 for an example. In this example we see that the resulting functionals is slightly shifted towards x_{j+1} and therefore so is also μ^*. By letting Ω stay coarse and Ξ be fine we should be able to determine the underlying parameter value μ^* with good accuracy by the procedure given in Section 3.1.

When we do parameter estimation (when noise have been added) we want to compare the estimation sharpness with something, so we examine what we can say about the parameter value without filter and without noise. If we let $J = 10$ we get $\Delta x = x_{j+1} - x_j = 0.2222$, and since $\mathcal{D} = (-0.5, 0.5)$ we can tell μ^* with an accuracy of $0.2222 = 22.22\%$ of the parameter domain if we measure in all the points of the coarse space grid.

To determine which parameter values that are consistent with our measurements we are always considering the inequality

$$\| \tilde{V} - \overline{Z} \| \leq \frac{\hat{\rho}}{\sqrt{m'}}.$$

If, for some reason, we could observe the functionals without noise, the right hand side of this equation would be equal to zero. In that case we would be able to tell the exact parameter value with only one obsevation point. However, when we add noise, the quantity $\hat{\rho}/\sqrt{m'} > 0$, and there might be several parameter values that are consistent with the numerical data.

Moving on with the example, we need to fix several values, see Table 4.1. As

Table 4.1: Numerical values for our first example.

$0 \leq j \leq J = 10$	points in space dimension
$0 \leq p \leq P = 500$	points in parameter space
$\Delta x = 0.2222$	grid spacing
$N_{\text{exp}} = 1000$	number of experiments
$m' = 5$	number of magic centers
$\sigma = 0.02$	introduced noise
$\gamma = 0.95$	confidence level

mentioned earlier we let $\Upsilon = \Xi$ for simplicity. In each experiment we randomly choose a $\mu^* \in \Upsilon$ to estimate.

We apply the Gaussian filter for different σ_m-s and magic centers, $2 \leq n \leq 10$. Because of numerical round off error the implemented algorithm starts to choose the same centers multiple times for small σ_m-s, so we set $n_{\max}$ to the maximum number of distinct magic centers chosen. Table 4.2 shows the estimation sharpness

Table 4.2: The table shows $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$, horizontal is n, vertical is σ_m. Note that for $n = J = 10$ we observe the function in all possible centers of our coarse grid over Ω. This disable in a way the greedy choice of the GEIM, so in a sense we can say that on a coarse grid the GEIM only helps for the first few choices of magic centers.

	2	3	4	5	6	7	8	9	10
0.01	0.633	0.395	0.205	0.139	0.140	0.137	-	-	-
0.02	0.608	0.255	0.183	0.093	0.096	0.096	-	-	-
0.03	0.389	0.225	0.135	0.058	0.059	0.059	-	-	-
0.04	0.345	0.177	0.107	0.038	0.035	0.033	0.033	0.035	-
0.05	0.332	0.149	0.076	0.025	0.024	0.020	0.021	0.022	-
0.06	0.300	0.125	0.058	0.021	0.019	0.017	0.017	0.017	-
0.07	0.268	0.101	0.046	0.018	0.017	0.015	0.016	0.016	0.016
0.08	0.255	0.078	0.035	0.017	0.016	0.015	0.015	0.016	0.016
0.09	0.230	0.072	0.030	0.017	0.016	0.015	0.016	0.016	0.016
0.10	0.211	0.060	0.027	0.017	0.017	0.015	0.016	0.017	0.017
0.20	0.091	0.035	0.027	0.023	0.023	0.023	0.023	0.023	0.023

$E(|\Upsilon_{\text{con}}|)/|\Upsilon|$. In the table the red numbers represent estimates that is less sharp than without a filter and without noise. The blue numbers is when our estimate is less or equal to $0.020 = 2.0\%$ of the parameter domain. We see that after filter use we are mostly able to estimate μ^* with high accuracy, if we choose somewhat right σ_m and n.

In Figure 4.5 we have plotted the sharpness $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$ as a function of the number of magic centers, $2 \leq n \leq 10$. The figure shows that the points chosen, after approximately the first five, gives little additional information on the parameter value. This illustrates one of the key points when doing parameter estimation: there are, as stated, a total of nm' measurements available. If we are given the opportunity, it is often better to do more repeated measurements in good measurement points (increase m'), rather that increasing the number of magic centers[1] (increase n).

[1]From Equation (3.2) we expect the estimation sharpness to approximately decrease by a factor of two when m' increase by a factor four, which we will confirm by numerical experiments later.

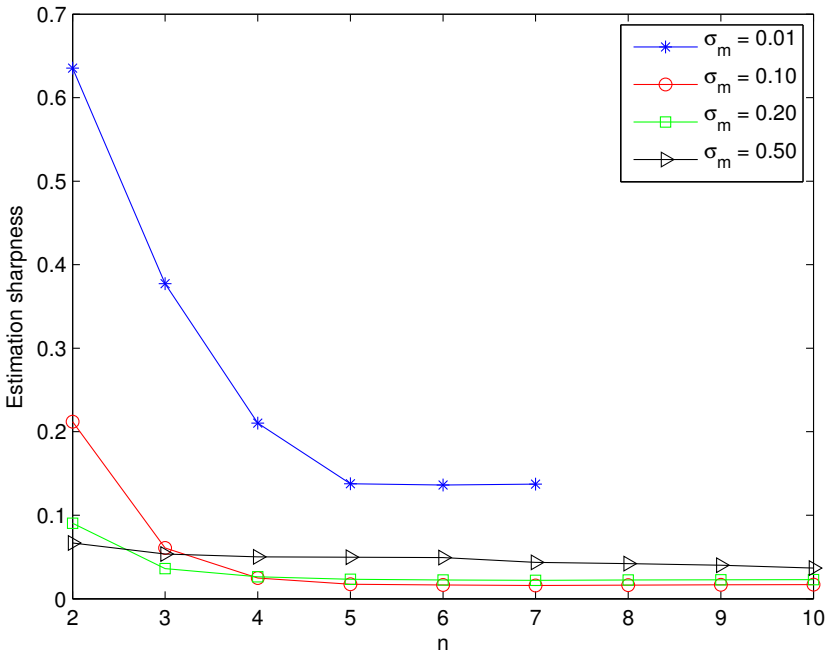

Figure 4.5: Estimation sharpness as a function of n for different σ_m-s.

Example 2: One dimensional box function

For the next example we will look at the one dimensional function

$$u(x;\mu) = \begin{cases} 1 & \mu - b/2 \leq x \leq \mu + b/2, \\ 0 & \text{elsewhere,} \end{cases}$$

where $\Omega = (-1, 1)$ and $\mathcal{D} = (-0.9, 0.9)$. We let $b = 0.2$ so that we are looking at an one dimensional box with unit height and width 0.2 moving inside the domain when μ changes; see Figure 4.6.

Once again we apply the Gaussian filter, this time with different σ_m values; see Figure 4.7. The filter smooths out the function as expected, which enables us again to use the GEIM algorithm to more easily construct a function approximation; see Figure 4.8. Note that we have used a higher filter width for this example than for the previous. This function is in a sense even more irregular, and therefore must be smoothed out even further out to achieve a satisfying approximation.

The first ten chosen magic centers the GEIM picks when $\sigma_m = 0.5$ can be seen in Figure 4.9. Interestingly, the distribution of the chosen centers seems to resemble something between equidistant- and a GLL distribution.

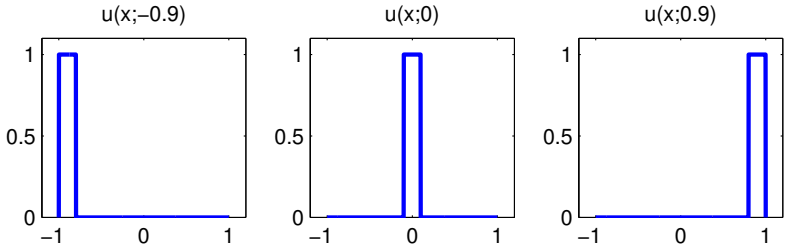

Figure 4.6: $u(x;\mu)$ with $\mu = -0.9$, $\mu = 0$ and $\mu = 0.9$.

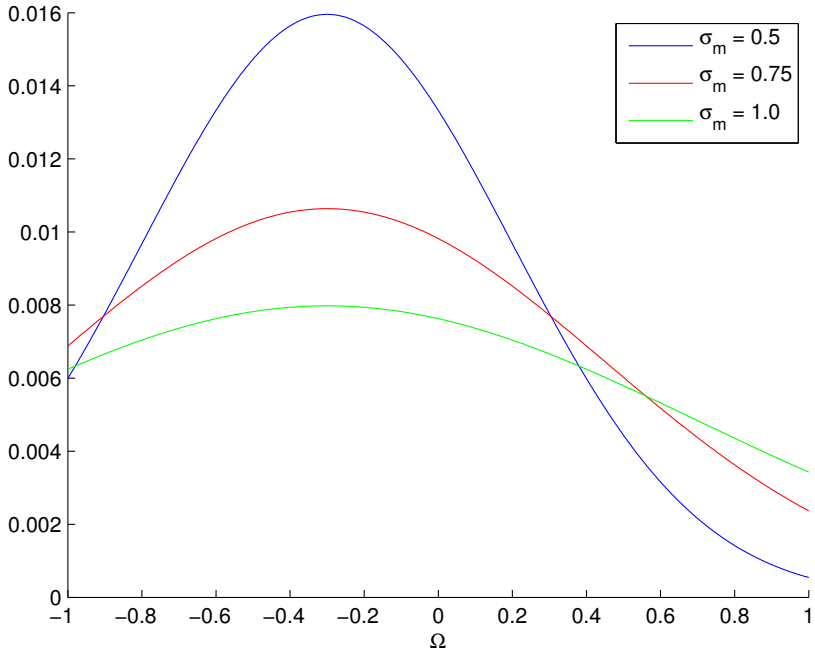

Figure 4.7: $u(x;-0.3)$ filtered with three different choices for σ_m over Ω.

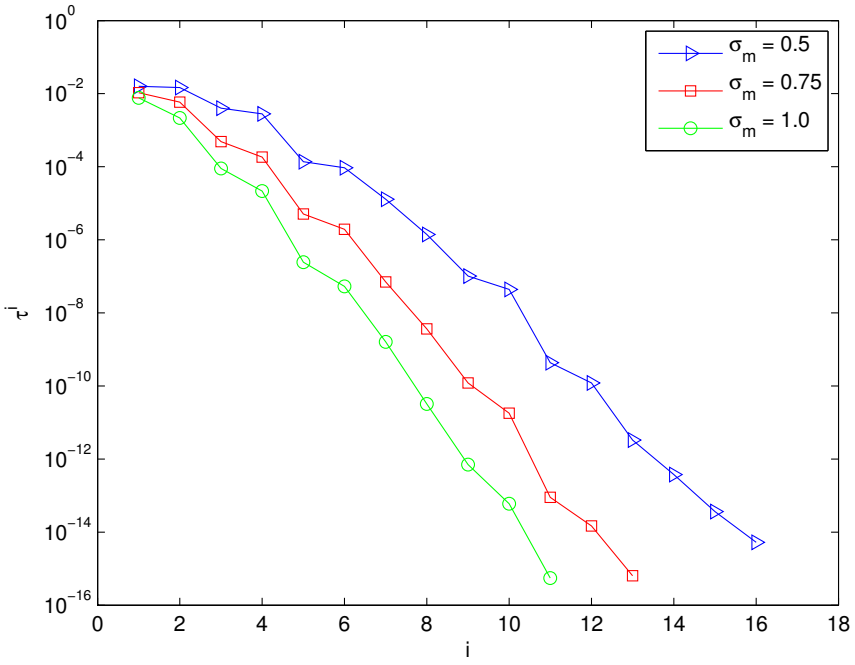

Figure 4.8: With $\sigma_m = 0.5$ the GEIM reaches the tolerance $|\tau^i| < 10^{-14}$ after 16 iterations, with $\sigma_m = 0.75$ after 13 iterations, and with $\sigma_m = 1.0$ after 11 iterations.

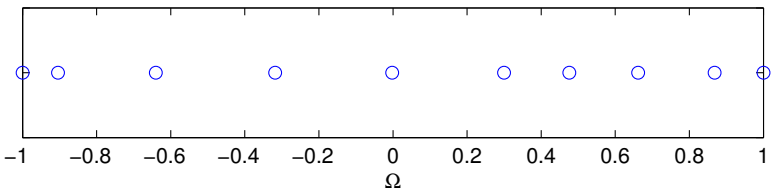

Figure 4.9: Where the GEIM chooses the magic centers when doing function approximation on the box function with $\sigma_m = 0.5$ on a fine grid.

When doing parameter estimation on this example we are estimating where the box is centered in the domain. In the case when σ_m is small we need the grid size to be equal to or smaller than the box width. If the grid spacing is bigger, we might end up with the case where we are given a random chosen parameter value to estimate that "falls between" the grid, and we will never pick up where that value lies. Therefore we let $J = 11$ so that the distance of two adjacent grid points is $\Delta x = 0.2$, the same as the boxwidth.

In the previous example we saw that if we could observe the functionals without noise, we would in principle only need one measurement point to determine the exact parameter value μ^*. In this case, however, we would generally need two observation points to determine the exact value. Figure 4.10 shows an illustration of why this is the case.

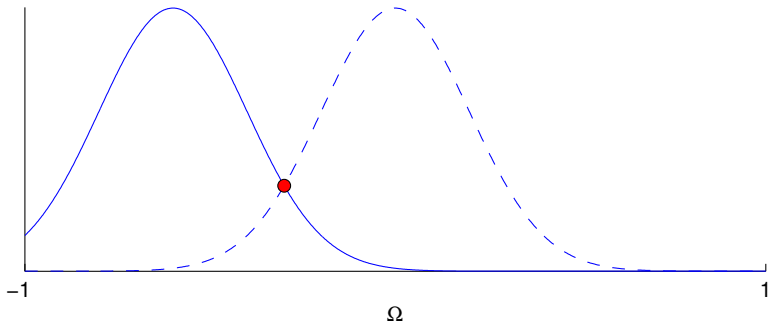

Figure 4.10: If we observe the functionals in one magic point before noise have been added, there are generally two parameter values that may explain the measured value. To find the exact μ^* we need to measure in one additional point.

If we could measure the function in the points on the coarse grid without filter and without noise, we can, with this construction, again tell the parameter value with an accuracy of $0.4/1.8 = 0.2222 = 22.22\%$ of the parameter domain.

The numerical values for the experiment is given in Table 4.3, and the resulting estimation sharpness can be seen in Table 4.4. In the estimation table we have highlighted the σ_m values which gives the best estimation sharpness for each given number of magic centers. We see that when n is smaller we need a higher σ_m. The reason for this is we will more often be given values to estimate that are "far away" from the magic centers, and therefore need a higher σ_m so that the functionals "reach out" to those centers to pick up the given μ^* value. However, when n is larger we have more of the domain covered, so can use a smaller σ_m and still observe where the filtered box function is centered.

Table 4.3: Numerical values for our second example.

$0 \leq j \leq J = 11$	points in space dimension
$0 \leq p \leq P = 500$	points in parameter space
$\Delta x = 0.2$	grid spacing
$N_{\text{exp}} = 1000$	number of experiments
$m' = 5$	number of magic centers
$\sigma = 0.02$	introduced noise
$\gamma = 0.95$	confidence level

Table 4.4: $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$. Horizontal is n, vertical is σ_m. The highlighted numbers shows which σ_m value that gives the best estimation for a given number of magic centers.

	2	3	4	5	6	7	8	9	10
0.04	0.345	0.156	0.055	0.033	0.017	0.014	0.010	0.010	0.010
0.05	0.302	0.116	0.029	0.017	0.012	0.009	0.007	0.007	**0.006**
0.06	0.273	0.102	0.022	0.014	0.011	**0.009**	**0.007**	**0.007**	0.007
0.07	0.253	0.094	0.020	**0.014**	**0.011**	0.009	0.007	0.007	0.007
0.08	0.242	0.082	0.018	0.014	0.011	0.009	0.008	0.007	0.007
0.09	0.223	0.076	**0.018**	0.015	0.013	0.010	0.008	0.008	0.008
0.10	0.181	0.065	0.018	0.016	0.013	0.011	0.008	0.008	0.008
0.20	**0.066**	**0.053**	0.035	0.033	0.024	0.022	0.022	0.020	0.019
0.30	0.097	0.072	0.052	0.046	0.042	0.042	0.039	0.037	0.036

Example 3: Two dimensional box function

We move on by extending the previous example into two dimensions by

$$u(x;\mu) = \begin{cases} 1 & \mu_1 - b_1/2 \leq x_1 \leq \mu_1 + b_1/2, \quad \mu_2 - b_2/2 \leq x_2 \leq \mu_2 + b_2/2, \\ 0 & \text{elsewhere,} \end{cases}$$

where $x = (x_1, x_2)$, $\mu = (\mu_1, \mu_2)$, $b = (b_1, b_2)$ and $\Omega = (-1, 1)^2$. We let $\mathcal{D} = (-0.9, 0.9)^2$ for the parameter domain and $b = (0.2, 0.2)$. A plot of the function with $\mu = (0, 0)$ can be seen in Figure 4.11.

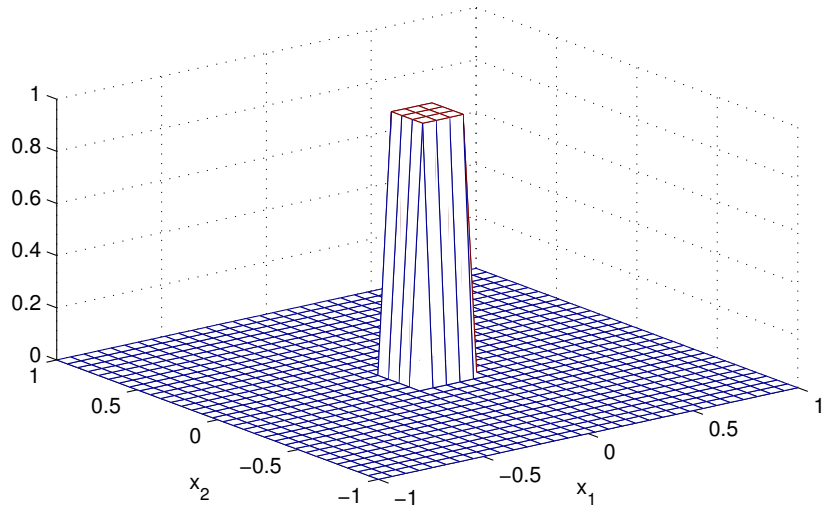

Figure 4.11: $u(x;\mu)$ with $\mu = (0, 0)$.

We apply the Gaussian filter, which in the two dimensional case has the form

$$g(x - x_j) = \frac{1}{2\pi\sigma_m} \exp\left(- \frac{(x_1 - x_{j,1})^2 + (x_2 - x_{j,2})^2}{2\sigma_m} \right).$$

Like in the one dimensional case the function is very hard to approximate, especially for small σ_m-s. Figure 4.12 shows the error $|\tau^i|$ for three σ_m-s. Even though the function is hard to approximate, the GEIM still preform with exponentially convergence.

Again we estimate random chosen parameter values, see Table 4.5 for the experiments numerical values. Without filter and noise, and with $J = 11 \times 11$ and $\mathcal{D} = (-0.9, 0.9)^2$ we can tell the parameter value with a sharpness of

$$\frac{0.4 \times 0.4}{1.8 \times 1.8} = 0.0493,$$

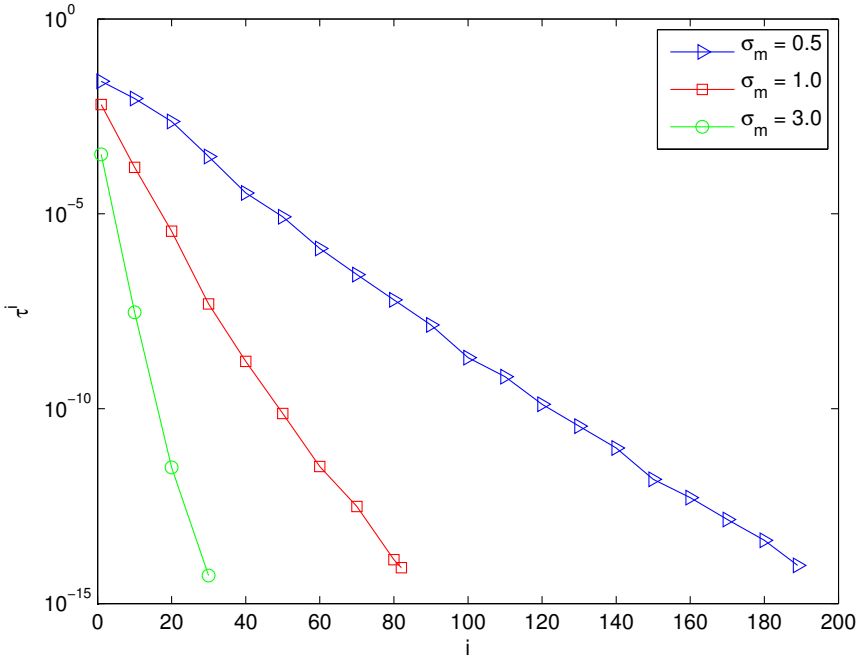

Figure 4.12: With $\sigma_m = 0.50$ the GEIM reaches the tolerance $|\tau^i| < 10^{-14}$ after 189 iterations, $\sigma_m = 1.0$ after 82 iterations and $\sigma_m = 3.0$ after 30 iterations.

Table 4.5: Numerical values for our two-dimensional example.

$0 \leq j \leq J = 11 \times 11 = 121$	points in space dimensions
$0 \leq p \leq P = 100 \times 100 = 10000$	points in parameter space
$N_{\text{exp}} = 1000$	number of experiments
$m' = 5$	number of repeated measurements
$\sigma = 0.02$	introduced noise
$\gamma = 0.95$	our confidence level

or 4.93% of the parameter domain.

We use the Gaussian filter for different σ_m and n values, and look at $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$. We do 1000 experiments for each value, see Table 4.6. The table shows if we choose

Table 4.6: $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$. Horizontal is n, vertical is σ_m.

	10	15	20	25	30	35	40	45	50
0.05	0.407	0.230	0.113	0.050	0.024	0.011	0.005	0.001	0.001
0.06	0.324	0.186	0.071	0.026	0.008	0.004	0.001	0.001	0.001
0.07	0.278	0.129	0.037	0.008	0.004	0.002	0.001	0.001	0.001
0.08	0.253	0.086	0.024	0.008	0.003	0.002	**0.001**	**0.001**	**<0.001**
0.09	0.195	0.060	0.016	0.004	0.002	0.002	0.001	0.001	0.001
0.10	0.180	0.063	**0.011**	**0.003**	**0.002**	**0.001**	0.001	0.001	0.001
0.20	**0.052**	**0.019**	0.012	0.009	0.007	0.007	0.006	0.006	0.005
0.30	0.095	0.056	0.049	0.035	0.036	0.032	0.030	0.027	0.026

an appropriate σ_m we are able to estimate the parameter value with high accuracy.

Note that when σ_m is small the function after filter use is nearly unchanged, i.e. there is little change in the filtered $u(x;\mu)$ when the change in μ is small. This means that to be able to estimate μ^* well, one of the magic centers need to "hit" the box. This affects each individual independent experiments in a couple of ways: if we do not hit the box with a magic center the estimation set Υ_{con} will be rather bad, and when we do hit it, say at magic center i, the rest of the chosen centers $n_{i+1}, \ldots, n_{\max}$ will be redundant. When we do not hit the box we are in a sense removing parameter values that are not consistent with our data, instead of closing in on the right value; see Figure 4.13 for an example. On the other hand, when σ_m becomes too big the resulting function after filter use will be very smooth and contain low values. With low values the introduced noise will affect our measurements relatively more, and we will not be able to estimate the value as well. In that case we need to increase the repeated measurements in each center by m' to get a sharper estimate.

4.3.2 Fourier coefficients filter

In the previous examples we have used the Gaussian filter that smooths out a given function in a fairly intuitive manner. However, there is nothing stopping us from applying any other filter and then use the same framework as before. In the next example we first define our function globally and approximate it by a Fourier series. Then we let the coefficients defined from the resulting Fourier series to be the functionals on a local domain.

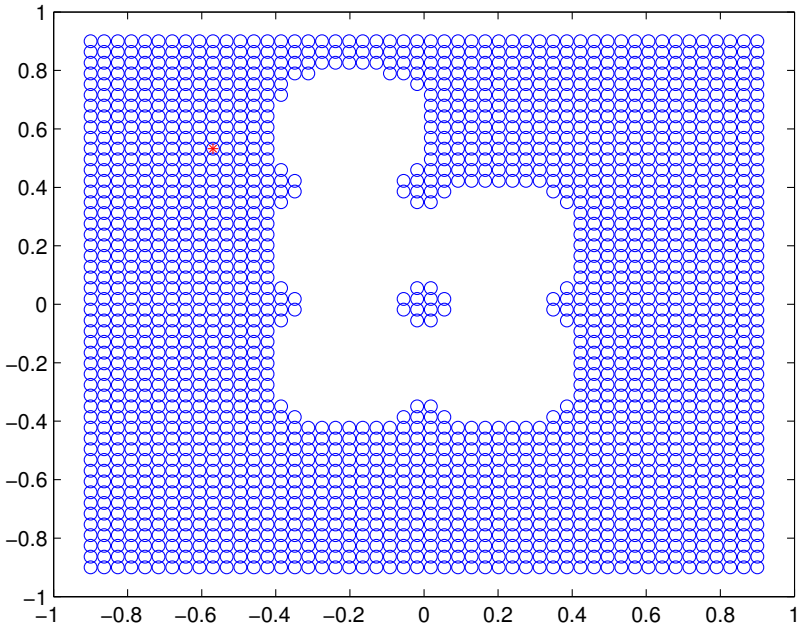

Figure 4.13: The red dot is the parameter value we want to estimate, and the blue is potential points. The five chosen centers have removed some of the points, but fails to find the correct one because the filtered function is too sharp. To get a more accurate estimate we need to either increase n to find the parameter value, or change our filter width σ_m.

We let the global function be

$$u(x;\mu) = \begin{cases} -1 & -\mu - b/2 \leq x \leq -\mu + b/2, \\ 1 & \mu - b/2 \leq x \leq \mu + b/2, \\ 0 & \text{elsewhere,} \end{cases}$$

where b is the box width, $\hat{\Omega} = (-2, 2)$ and $\mathcal{D} = (0.1, 1.9)$. As before we use $b = 0.2$.

We define the filter as

$$g(x) = \frac{1}{2} \sin\left(\frac{j\pi x}{2}\right),$$

so the functionals becomes

$$a_j = \frac{1}{2} \int_{-2}^{2} u(x;\mu) \sin\left(\frac{j\pi x}{2}\right). \tag{4.8}$$

This construction might seems strange, but is in fact the same one dimensional box function as before with an odd extension and the domain shifted from $(-1, 1)$ to $(0, 2)$; see Figure 4.14. The magic funtionals is now the corresponding coefficients

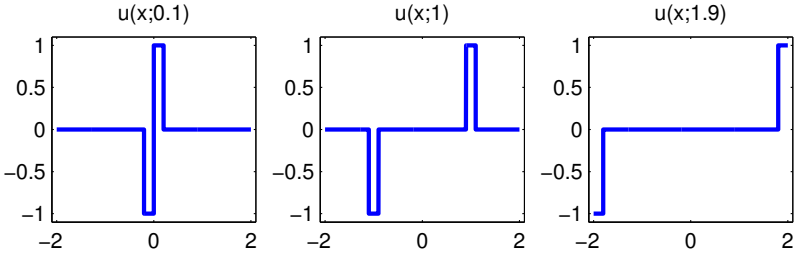

Figure 4.14: The box function $u(x;\mu)$ from example 2 with an odd extension, $\mu = 0.1$, $\mu = 1$ and $\mu = 1.9$.

in a Fourier series of the global periodic function $u(x;\mu)$. The functionals in (4.8) can be found analytically by

$$\begin{aligned} a_j &= \frac{1}{2} \int_{-2}^{2} u(x;\mu) \sin\left(\frac{j\pi x}{2}\right) \\ &= \frac{1}{2} \left(\int_{-\mu-b/2}^{-\mu+b/2} (-1) \sin\left(\frac{j\pi x}{2}\right) \mathrm{d}x + \int_{\mu-b/2}^{\mu+b/2} \sin\left(\frac{j\pi x}{2}\right) \mathrm{d}x \right) \\ &\vdots \\ &= \frac{4}{j\pi} \sin\left(\frac{j\pi\mu}{2}\right) \sin\left(\frac{j\pi b}{4}\right). \end{aligned} \tag{4.9}$$

Note that the functionals is constructed by using the global periodic version of $u(x;\mu)$ defined over $\hat{\Omega} = (-2, 2)$. With the functionals well defined we now let the local domain be $\Omega = (0, 2)$.

If we want we can now write our function as an infinite sum

$$\begin{aligned} u(x;\mu) &= \sum_{j=1}^{\infty} a_j \sin\left(\frac{j\pi x}{2}\right) \\ &= \sum_{j=1}^{\infty} \frac{4}{j\pi} \sin\left(\frac{j\pi\mu}{2}\right) \sin\left(\frac{j\pi b}{4}\right) \sin\left(\frac{j\pi x}{2}\right), \end{aligned}$$

or approximate by a finite sum

$$u(x;\mu) \approx u_J(x;\mu) = \sum_{j=1}^{J} \frac{4}{j\pi} \sin\left(\frac{j\pi\mu}{2}\right) \sin\left(\frac{j\pi b}{4}\right) \sin\left(\frac{j\pi x}{2}\right).$$

To check that a_j is correctly computed we plot u_J over Ω with $J = 50$ and $\mu = 1$; see Figure 4.15.

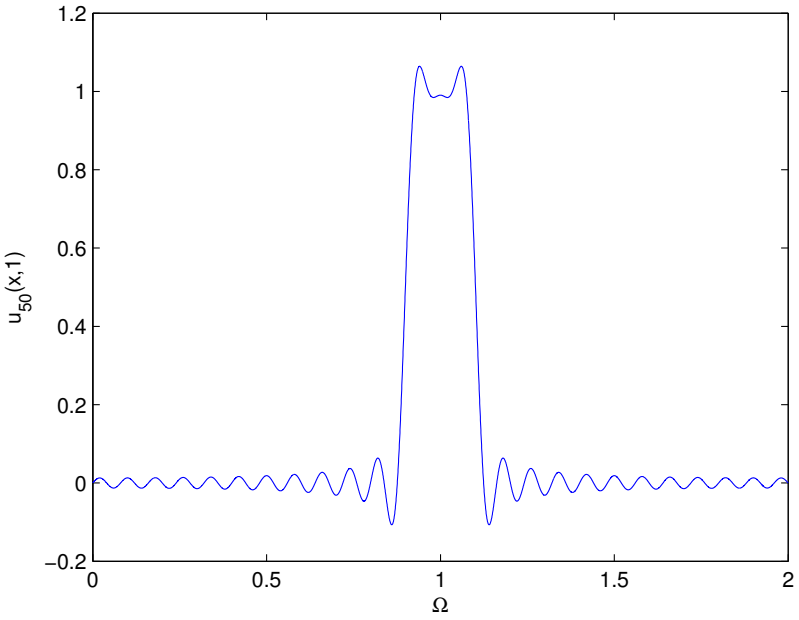

Figure 4.15: The approximation $u_{50}(x;1)$ over $\Omega = (0, 2)$.

Note that in previous examples the functionals was associated with a "center" or measurement site, $x_j \in \Omega$, while now we associate the functionals with coefficients of Fourier series dependent on the box position by μ. Figure 4.16 shows the

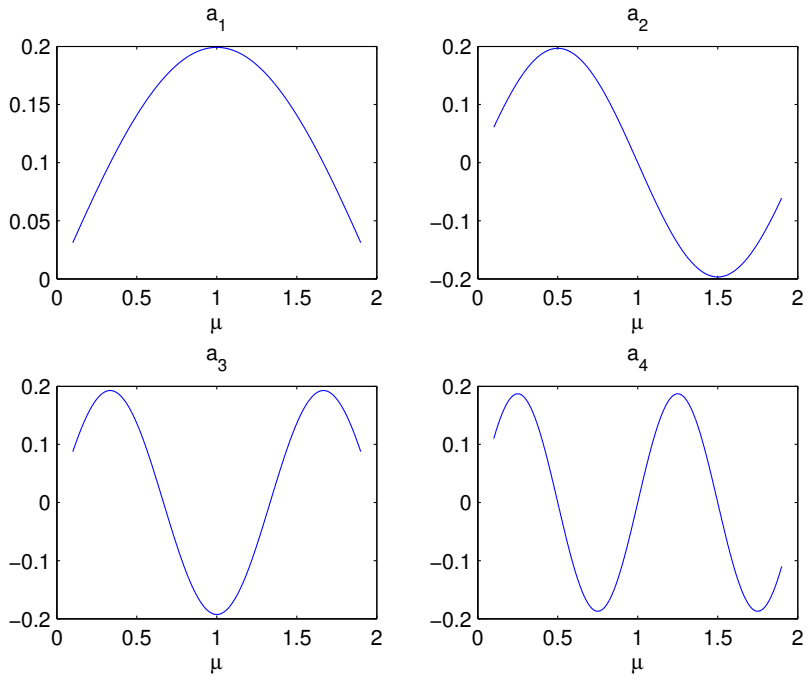

Figure 4.16: The functionals a_j, $j = 1, 2, 3, 4$, over the parameter domain μ.

functionals a_j, $j = 1, 2, 3, 4$, as a function of μ. The functionals after filter use are not in any way smooth as before, and we will see how this affect the results when using the GEIM in function approximation.

From (4.9) we expect a_j to decay with a rate of $1/j$, i.e.

$$a_j \sim C/j, \tag{4.10}$$

or on a logarithmic scale

$$\log a_j \sim C' - \log j.$$

A logarithmic plot of $\max_{\mu \in \mathcal{D}} |a_j|$ is shown in Figure 4.17. The figure shows that a_j decay with a rate of $1/j$. This result have an important impact in parameter estimation, the introduced noise will affect the results relatively more for a_k than a_j, when $k >> j$. This may not be a problem in itself, but for this example it reinforces the argument that it is better to measure multiple times in good measurement points than producing loads of magic centers.

We try to use the GEIM algorithm to approximate. Let $1 \leq j \leq J = 5000$ and set $n_{\max} = 1000$. Figure 4.18 shows the maximum error given by $|\tau^i|$ on a logarithmic scale. The plot shows that the error does not decrease like previous examples, which is of course connected with the shape of the functionals (as seen in Figure 4.16).

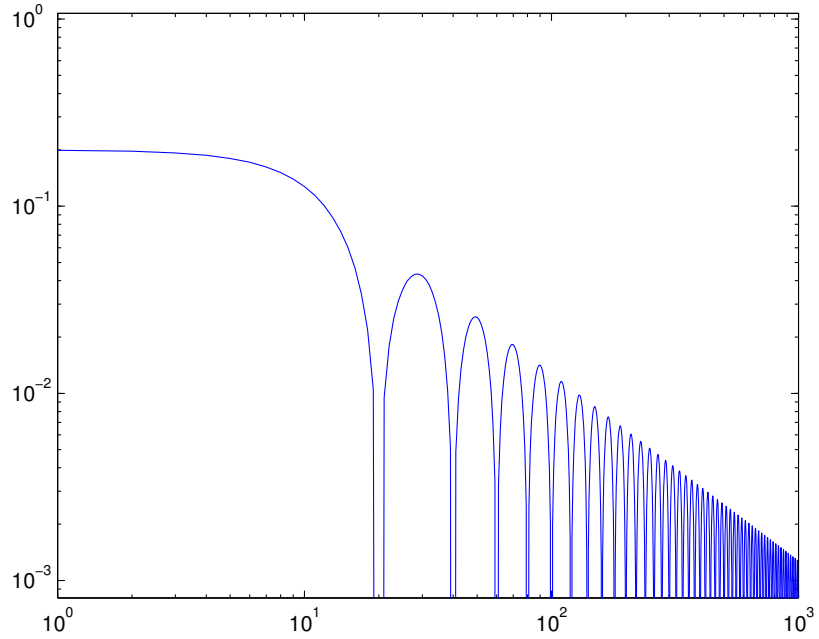

Figure 4.17: Logarithmic plot of $\max_{\mu\in\mathcal{D}} |a_j|$ when j increases.

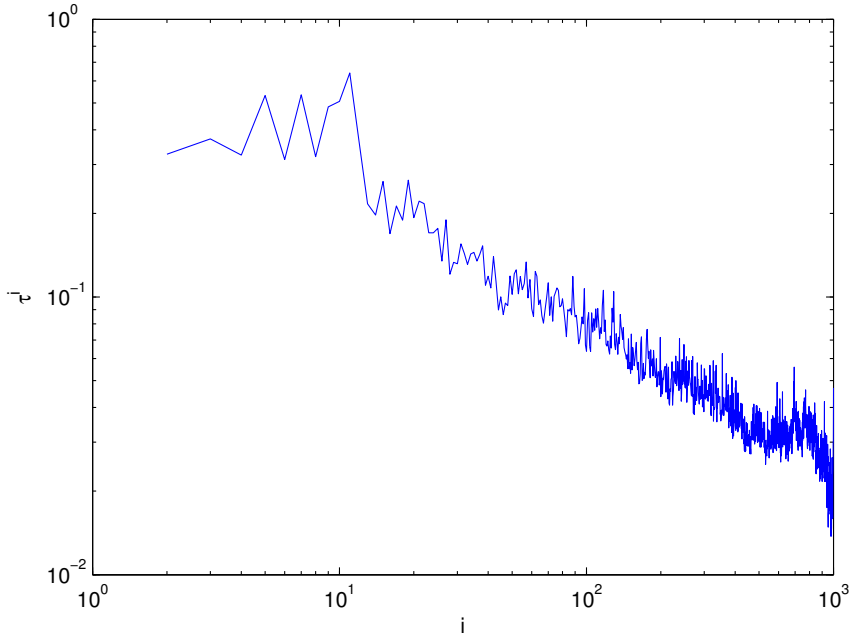

Figure 4.18: The maximum error $|\tau^i|$

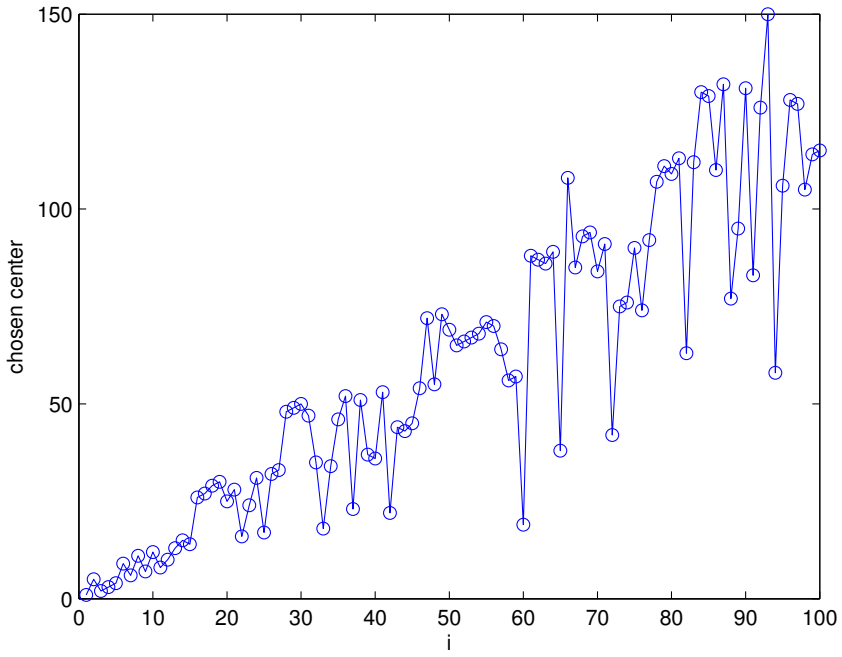

Figure 4.19: The 100 first magic coefficients chosen by the GEIM. A chosen j corresponds to a_j.

Figure 4.19 shows the chosen magic centers (or magic coefficients) for this filter. The GEIM choose the centers in a linear like fashion, which make sense because of the decay rate of the functionals from (4.10). This means that for parameter estimation we gain little by increasing J, meaning that if we want to measure for instance in the first three magic coefficients the algorithm will choose the exact same three whether $J = 10$ or $J = 100$, $1 \leq j \leq J$. Figure 4.20 shows the first six basis functions that the GEIM constructs. Of course, now the q_i-s are not functions in the same sence as before, but rather vectors containing information on the difference between the current interpolant and the filtered function represented by the matrix H from the GEIM.

Since $u(x;\mu)$ can be expressed by a Fourier series it is also integrable over Ω. By Parceval's identity [13] we can express the the L^2-norm of $u_J(x;\mu)$ as

$$\| u_J \|^2_{L^2(\Omega)} = \int_\Omega u_J^2 \mathrm{d}x,$$

where $\Omega = (0, 2)$, which we define as the energy of the approximation. Looking at

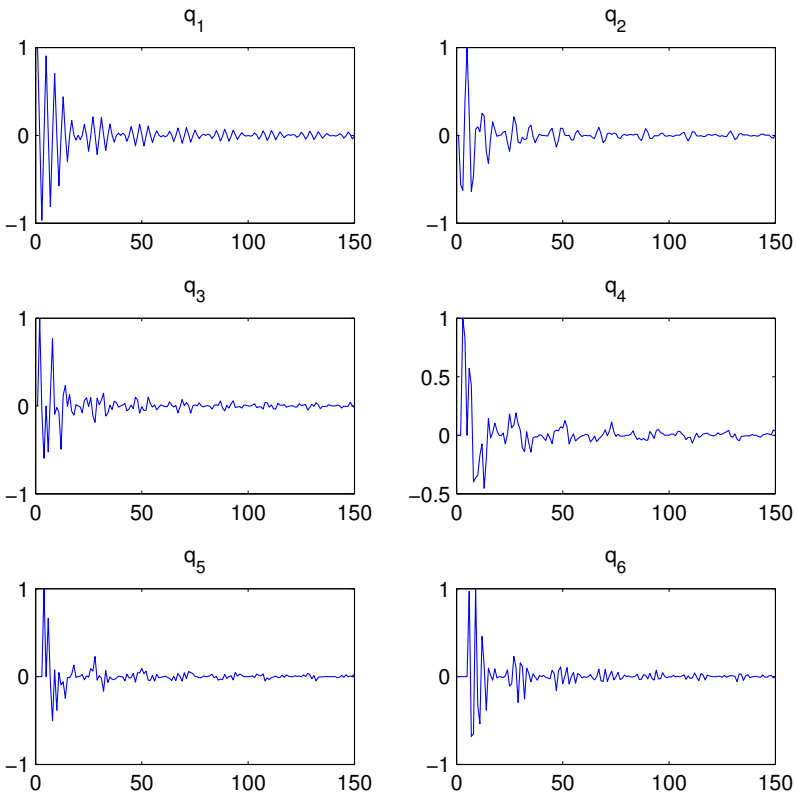

Figure 4.20: Basis functions q_i, $i = 1, 2, 3, 4, 5, 6$, when $J = 150$.

the integral we get

$$\begin{aligned}\int_\Omega u_J^2 \mathrm{d}x &= \int_\Omega \left(\sum_{j=1}^{J} a_j \sin\left(\frac{j\pi x}{2}\right)\right)\left(\sum_{k=1}^{J} a_k \sin\left(\frac{k\pi x}{2}\right)\right)\mathrm{d}x \\ &= \sum_{j=1}^{J}\sum_{k=1}^{J} a_j a_k \underbrace{\int_\Omega \sin\left(\frac{j\pi x}{2}\right)\sin\left(\frac{k\pi x}{2}\right)\mathrm{d}x}_{c}.\end{aligned}$$

To determine the value c there are two cases to consider, $j = k$ and $j \neq k$. When $j = k$ we get

$$\begin{aligned}c &= \int_0^2 \sin^2\left(\frac{j\pi x}{2}\right)\mathrm{d}x \\ &= \left[\frac{x}{2} - \frac{\sin(j\pi x)}{2j\pi}\right]_0^2 = 1,\end{aligned}$$

and when $j \neq k$

$$\begin{aligned}c &= \int_0^2 \sin\left(\frac{j\pi x}{2}\right)\sin\left(\frac{k\pi x}{2}\right)\mathrm{d}x \\ &= \left[\frac{\sin(\pi x(j-k)/2)}{\pi(j-k)} - \frac{\sin(\pi x(j+k)/2)}{\pi(j+k)}\right]_0^2 \\ &= \frac{\sin(\pi(j-k))}{\pi(j-k)} - \frac{\sin(\pi(j+k))}{\pi(j+k)} = 0,\end{aligned}$$

since $j, k = 1, \ldots, J$. So the energy of the signal is simply given by

$$\| u_J \|_{L^2}^2 = \sum_{j=1}^{J} a_j^2.$$

One way to interpret this result is that the GEIM chooses the coefficients that contributes the most to the signal given by $u_J(x;\mu)$.

We apply the statistical framework. In Table 4.7 we have estimated random chosen parameters from the parameter domain, $\mu^* \in \Upsilon$, with $|\Upsilon| = 1000$, $m' = 5, 10, 15, 20$, and as usual $\gamma = 0.95$ and $N_{\text{exp}} = 1000$. As seen, for this example the filtered version of the function looks nothing like the original. However, remember that for each independent experiment the $\mu^* \in \Upsilon$ we estimate still represent the position of the original box in the domain Ω.

Table 4.7: Estimation sharpness given by $E(|\Upsilon_{\text{con}}|)/|\Upsilon|$. Horizontal n, vertical m'. Note that when $m' \to 4m'$ the sharpness goes approximately down by a factor two.

	1	2	3	4	5	6	7	8	9	10
5	0.272	0.076	0.033	0.026	0.021	0.016	0.014	0.012	0.011	0.010
10	0.182	0.049	0.022	0.016	0.014	0.010	0.009	0.008	0.007	0.007
15	0.149	0.039	0.018	0.013	0.011	0.008	0.007	0.006	0.006	0.005
20	0.128	0.035	0.014	0.011	0.010	0.007	0.006	0.005	0.005	0.005

4.3.3 The Runge function: revisited

At the end of this thesis we once more return to the Runge function. Just to recap, the problem contains a single scalar parameter $\mu \in \mathcal{D}$ and one space dimension, and the function is

$$u(x;\mu) = \frac{1}{1+\mu x^2},$$

where $\Omega = (-1, 1)$ and $\mathcal{D} = (1, 25)$.

So far we have only used the magic centers from the EIM or the GEIM in parameter estimation, and discarded the rest of the information provided by the algorithms. Now we let the filter g from (4.1) be the basis functions produced by the EIM, so that the functionals becomes

$$\ell_j(u) = \int_\Omega u(x;\mu) q_j(x) \mathrm{d}x.$$

In a way we are taking the process one step further from the EIM by using the basis function instead of the magic centers. In other words, first we use the EIM to determine the basis functions to use as the filter, then we form the linear functionals by GEIM. Now we have two options, we could let the measurement points either be the same as the magic centers from the EIM, or we can use the GEIM on the resulting system given by the functionals to determine a new set of magic centers to measure at. For the second we need the total number of functionals to be bigger than the number of magic centers, $J > n$, so that the GEIM algorithm have a few options to choose from.

For the first case we let $1 \leq j \leq J = n$ and let $n = 2, \ldots, 10$, with the result given in Table 4.8. Compared with the results with no filter in Table 3.2 from Section 3.3 the estimation sharpness now is better when using a few magic centers. However the sharpness does not improve when n increases, as it does without the basis function filter. The reason for this is that the q_j-s become more oscillatory when j increases, with the result that the numerical values of the functionals ℓ_j

Table 4.8: Estimation sharpness using the basis functions as filter. Here $m' = 5$.

n	$E(\lvert\Upsilon_{\text{con}}\rvert)/\lvert\Upsilon\rvert$
2	0.076
3	0.077
4	0.077
5	0.083
6	0.085
7	0.081
8	0.087
9	0.085
10	0.088

greatly decreases. The noise given by σ then affects the measurements relatively more, just like in "Fourier coefficients filter" example.

Furthermore, we can use the GEIM on the functionals to determine where to do our measurements, with the results given in Table 4.9. Again the results do

Table 4.9: Estimation sharpness using the basis functions as filter and the GEIM applied. Here $m' = 5$.

n	$E(\lvert\Upsilon_{\text{con}}\rvert)/\lvert\Upsilon\rvert$
2	0.103
3	0.078
4	0.084
5	0.081
6	0.084
7	0.087
8	0.087
9	0.088
10	0.085

not improve greatly when n increases, which is no suprise for the same reason as without applying the GEIM on the functionals.

Chapter 5

Conclusions

In this report we have focused on the empirical interpolation method (EIM). The method was first introduced in 2004, developed in connection with the reduced basis framework for parametrized differential equations. We considered one way to generalize the method further, by introducing a linear functional EIM , denoted as the generalized empirical interpolation method (GEIM).

The work done in this report can be divided into two topics. We presented the EIM and GEIM, and looked at the properties of those method. We also investigated whether the interpolation points produced by the interpolation method are good candidates for measurement points in parameter estimation problems.

In Chapter 2 we introduced the EIM and gave a fairly extensive explanation of all the steps of the algorithm. We did a short section on error analysis, and in Theorem 2.2 we stated that for functions with analytic parameter dependence the EIM achieves exponential convergence, which we verified in Section 2.3 and 2.5 for both one- and multiple dimensions. In the experiments we compared the method to interpolation through classical Gauss points, and we saw that EIM was far superior. We also investigated how the EIM can be used in integral approximation.

Furthermore, in Chapter 4 the GEIM was presented by introducing a linear functional EIM. The functionals allowed us to observe functions through some given filter, and consequently more robustly capture the behaviour of less regular functions. We used the GEIM on several examples with irregular functions, and saw how the method approximated a given function with different filters.

For each example we also estimated given parameters, after doing measurements on the functions, with the statistical framework given in Section 3.1. Here we measured in the interpolation points provided by the GEIM algorithm, and we showed that with an appropriate filter we was able to estimate the parameters with high accuracy.

Numerically we observed that repeated measurements in a few good measurement points give a sharper parameter estimation than increasing the number of

points in which we measure. With the one dimensional Runge function we was able to look at analytical expressions, and we showed that the precision on how well we can estimate parameters is a reflection of how sensitive the function is for parameter variation at certain points in the spatial domain.

We have been working undre the hypothesis that the points provided from the interpolation algorithms are generally good pionts to use in a parameter estimation setting [11]. The reason for this, however, is not clear, since there are no other obvious connection between parameter estimation and function approximation. The hypothesis has only been verified by numerical results, so we would like back it up with some theoretical work as well.

The generalized empirical interpolation method have been fairly recently proposed [12, 15], and the amount of literature on the subject is limited. However, it is definitely an interesting extension of the EIM, and it has potential use in a wide range of applications such as image processing and compression, visualization and animation. More research is needed, both theoretical and practical.

Bibliography

[1] G. Konidaris, J. D. Penn, M. Yano, A. T. Patera, *Math, Numerics, and Programming (for Mechanical Engineers)*, lecture notes from Massachusetts Institute of Technology (2012).

[2] W. Cheney, D. Kincaid, *Numerical Mathematics and Computing, 6th edition*, Thomson Brooks/Cole (2008).

[3] E.M. Rønquist, *Approximation and interpolation by higher order polynomials*, NTNU lecture notes (2012).

[4] Y. Maday, N.C. Nguyen, A.T. Patera, G.S.H. Pau, *A general multi purpose interpolation procedure: The magic points*, Communications on pure and applied analysis, Vol. 8, Number 1, pp. 383-404 (2009).

[5] M. Barrault, N.C. Nguyen, Y. Maday, A.T. Patera, *An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations*, C. R. Acad. Sci. Paris, Serie I., pp. 667-672 (2004).

[6] A. Quarteroni, A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer-Verlag (1994).

[7] R. Pasquetti, F. Rapetti, *Spectral element methods on unstructured meshes: which interpolation nodes*, Springer Science (2012).

[8] C. Runge, *Uber empirische funktionen und die interpolation zwischen a quidistanten ordinaten*, Zeit. Math. Phys. (1901).

[9] A. Townsend, L.N. Trefethen, *Gaussian Elimination as an Iterative Algorithm*, SIAM News, Volume 46, Number 2 (2013).

[10] M. Bebendorf, Y. Manday, B. Stamm, *Comparison of some reduced representation approximation*, (2013).

[11] A.T. Patera, E.M. Rønquist, *Regression on parametric manifolds: Estimation of spatial fields, functional outputs, and parameters from noisy data*, C. R. Acad. Sci. Paris, Ser. Vol. 350, pp. 543-547 (2012).

[12] A.T. Patera, E.M. Rønquist, *Regression on Parametric Manifolds: Validation and Prediction from Noisy Data*, work in progress.

[13] N. Young, *An Introduction to Hilbert spaces*, Cambridge University Press, (1988).

[14] T. Aanonsen, *Empirical interpolation with application to reduced basis approximations*, Master of Science Thesis, NTNU Department of Mathematical Sciences (2009).

[15] Y. Maday, O. Mula, *A generalized empirical interpolation method: application of reduced basis techniques data assimilation*, Analysis and Numerics of Partial Differential Equations, Springer pp. 221-235 (2013).

[16] A. Townsend, L.N. Trefethen, *An extension of Chebfun to two dimensions*, work in progress/preprint.

[17] N.R. Draper, H. Smith, *Applied Regression Analysis*, Wiley (1998).