Brown University

Division of Applied Mathematics

The Interpolation of Gravitational Waveforms

Author:
Jason Kaye

Advisor:
Professor Jan S. Hesthaven

May 2012

This honors thesis, written by Jason Kaye, has been read and approved by the following faculty members of Brown University's Division of Applied Mathematics:

Jan S. Hesthaven

Björn Sandstede

Date

Contents

1 Preface

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R + g_{\mu\nu}\Lambda = \frac{8\pi G}{c^4}T_{\mu\nu}$$

When considering this system of ten coupled nonlinear partial differential equations, called the Einstein field equations, it becomes clear that describing a law of nature is only one of many steps along the road to practical understanding. Encoded in this system is a broadly applicable description of the relationship between spacetime, matter, and gravitation, but it is notoriously difficult to solve. Acquiring approximate solutions requires many levels of simplification; at each level, some amount of purity is lost, but a great deal of insight into the underlying system may be gained.

In this thesis, we are concerned with gravitational waveforms, which are disturbances in spacetime caused by the motion and interaction of massive objects. The Einstein field equations predict their existence and provide a framework in which to investigate their structure, but a great deal stands between these equations and a useful representation of the whole spectrum of gravitational waveforms. Our goal, in the broadest sense, is to develop techniques by which to obtain accurate approximations of these waveforms without an excessive computational burden. In doing so, we must remember that many layers of complexity separate these approximations from the fact of the matter; therefore, though we will generally settle for solutions to representative special cases of the problem, our intention is to develop methods which are broad enough to be, ideally, applied to its most faithful formulations.

We will begin by offering a brief introduction to the science and mathematics of gravitational wave astrophysics in order to motivative our project. However, this is not our primary focus; we will soon phrase the problem in the language of mathematics, and afterward consider it as such. The techniques on which we focus are built on the findings of related research, which we will discuss after the scientific introduction.

We use interpolation to address the problem of approximating gravitational waveforms, and the bulk of this paper is devoted to introducing the concept of interpolation and investigating various interpolation techniques. Our intention is to provide the reader with some understanding of the theory behind these techniques, and an intuition about its strengths and limitations, so that given certain properties of the waveforms, the results of applying interpolation to our problem will seem natural and can be presented concisely. We will consider two formulations of the problem - in which the waveforms are determined by one and two parameters, respectively - and present results for both.

We reiterate that this paper is primarily concerned with investigating interpolation techniques, and not with the underlying science of our application, which is presented only briefly. In general, we intend to step the reader through the author's process of learning and research, from the very beginning to the present. Thus, the mathematical discussion begins from the most basic foundations, and much of the emphasis is on motivating the techniques which we will use. That said, we begin with a background discussion of gravitational wave astrophysics, much of the content of which is drawn from [1].

2 Gravitational Waves & Their Detection

Imagine that two massive objects in space, like black holes, orbit each other. They spiral closer and closer together until finally they merge. According to Einstein's general theory of relativity, these massive objects alter the structure of spacetime near them. General relativity further predicts that this effect propagates through space in the form of gravitational radiation far from the sources; the change in curvature induced in spacetime by the objects travels like a ripple in a pond.

Gravitational waves, unlike electromagnetic waves, are left mostly unaffected by matter. This makes them faithful to their sources, even if those sources are far away, but it also makes them difficult to detect. That is, on the one hand, the form of a particular gravitational wave provides an observer with a great deal of information about the systems which produced it; accurate measurements could allow the source objects' masses, spins, and orbital patterns, for example, to be deduced. Whereas electromagnetic waves carry information about the atomic properties of systems, the forms of gravitational waves, and therefore the information they carry, are determined by large-scale system dynamics. On the other hand, such an observer must be equipped with highly sensitive and specialized gravitational wave detectors.

The spectrum of gravitational waves spans an enormous frequency range, from 10^{-16} to 10^{4} Hz, and the frequency of a particular wave depends on the sources that created it. These sources include, but are not limited to, systems involving stellar mass black holes, neutron stars, massive black holes, supermassive black holes, in addition to various events from the early universe. Different types of detectors are required to identify these different types of signals.

There are a handful of existing ground-based gravitational wave detectors such as LIGO [2] and VIRGO [3]. There are also plans to build space-based detectors, like eLISA [4], which would be much less susceptible to noise and could therefore detect waves with different frequencies than their ground-based counterparts. It is important to note that no gravitational waves have yet been directly detected, though there is indirect empirical evidence for their existence through observation of the Hulse-Taylor pulsar [5]. There is also a great deal of theoretical evidence from the widely accepted general theory of relativity. In 2015, more advanced and sensitive ground-based detectors will go on line, and these are expected to detect gravitational waves directly [6].

The ground-based detectors use laser interferometers to find passing waves. Imagine two perpendicular arms with a laser beam splitter at their intersection. A laser is fired into the splitter, so that beams travel down both arms. Each arm contains two mirrors, and the beam is reflected off of these mirrors, bouncing back and forth. The beams then return to a photodetector. According to general relativity, if a gravitational wave passes through these arms, one of them will be lengthened slightly, and the other will be shortened. Therefore, one of the lasers will have to travel a shorter distance, and will reach the photodetector faster than the other. Measuring this difference allows an observer to deduce the form of the gravitational wave. The arms are kilometers long, so that small changes in their length cause substantial differences in the arrival times of the lasers. Detectors are placed at different locations on Earth so that they can verify each other's findings. A schematic of the LIGO detector is given in Figure 1.

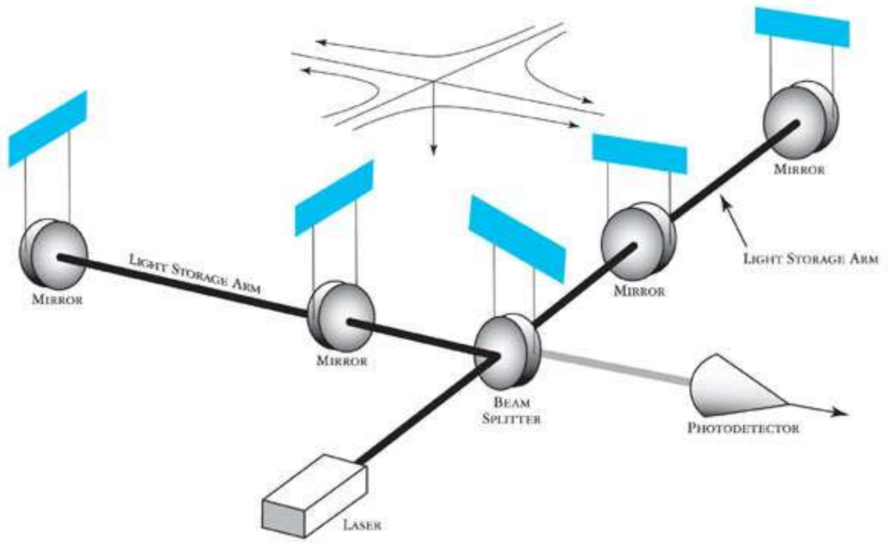

Figure 1: Schematic of the LIGO detector and a passing gravitational wave, taken from [7].

Space-based detectors would use similar techniques to measure gravitational waves. Since gravitational waves interact very little with matter, the signals are very weak, so isolating the system - in particular, the mirrors - from noise is a high priority. Space-based detectors, which use essentially free-falling mirrors, would be less susceptible to noise.

Once signals are detected, it is necessary to determine whether or not they represent gravitational waves. Einstein's field equations, which model the behavior of gravitational waves, provide the key. By obtaining solutions to the equations, signals could be tested against a library of possible gravitational waveforms. A positive detection, then, is a strong match between a signal and some known waveform.

2.1 Gravitational Wave Models & Matched Filtering

However, as stated before, solving Einstein's field equations for general astrophysical sources is, in general, infeasible. Alternatively, we must settle for approximate solutions. Many such formulations exist, and each provides a different sort of approximation [9]. In this research, we use approximate solutions generated using Post-Newtonian formalism, which reduces to Newton's laws in a limiting case. In this framework, approximate solutions are expressed by including only the lowest order nonlinear deviations of the Einstein equations from Newton's laws. For the purposes of this research, then, we then act as if these approximations are the actual solutions.

The gravitational wave models we use have been Fourier transformed, so they are functions of frequency, rather than time. A particular gravitational waveform corresponds to some parameter $\mu = (\mu_1, \dots, \mu_n)$, with $\mu \in M \subset \mathbb{R}^n$, where M is the parameter space which we consider. Some of the parameters that are used were mentioned above; for example, the masses and spins of the sources. As we increase the number of parameters used in the model, we are able to describe a broader range of sources, but we also have more waveforms to consider and each waveform is a more complicated object.

A gravitational waveform, then, is considered for our purposes to be a function $h_\mu : \mathbb{R} \to \mathbb{C}$, where the domain represents the frequency space. The codomain is complex because a Fourier transform has been applied to the waveform. The particular gravitational waveform approximation we use is given later and in [8]. It is important to note that these waveforms h_μ are C^∞ smooth. In fact, if we consider $h : \mathbb{R}^n \times \mathbb{R} \to \mathbb{C}$ as a function of a parameter μ and a frequency f, h is smooth in both variables, so waveforms vary smoothly with their parameter.

Once we have models of gravitational waveforms, we need some way of testing these models for high affinity with a signal. The commonly-used technique is called matched filtering. Matched filtering measures the value of an inner product between two functions; the matched filter is related to the complex inner product given by

$$\langle F, G \rangle := \int_{f_L}^{f_U} \frac{F^*(f)G(f)}{S(f)}\, df \qquad (1)$$

Here, F and G are two waveforms in Fourier space, f_L and f_U are the lower and upper bounds of the frequency domain which we consider, respectively, * represents complex conjugation, and $S(f)$ is called the power spectral density of the gravitational wave detector [10]. S represents the noise of the detectors over different regions in frequency space; it is known that the amount of noise is larger in certain regions, and we want these regions to be weighted less in the inner product, so we make $S(f)$ proportional to the noise of the detector at f. This inner product, then, measures the similarity of F to G; it is maximized, for normalized F and G, when $F = G$.

The matched filter, with respect to h_μ, of data representing some potential underlying signal $s : \mathbb{R} \to \mathbb{C}$ in Fourier space is, then, a function of $\langle s, h_\mu \rangle$. If this inner product is above some threshold, we declare that the signal s was generated by a gravitational waveform with parameter approximately equal to μ. Therefore, a successful detection of a gravitational wave also gives information about the source of that wave; the larger the dimension of the parameter space, the more information we will obtain.

However, a higher-dimensional parameter space requires us to consider many more waveforms in order to resolve some subset of that space. Since the parameter μ is an element of $\mathbb{R}^n$, there is a continuum of gravitational waveforms, so we cannot test a signal against all possible waveforms; we must choose some discrete set of waveforms to test against each signal. A naive solution would be to simply evaluate the inner product $\langle s, h_\mu \rangle$ on a fine grid of parameters μ in parameter space, and check if the maximal such inner product is above the threshold. However, in a high-dimensional parameter space this is simply not feasible, as computing each inner product requires the costly numerical evaluation of an integral over a potentially large frequency domain. Either we accept that there will be holes in the filter, or we devise some method by which to represent the continuum of waveforms with high accuracy using only a few waveforms.

2.2 The Reduced Basis Method of Waveform Representation

One solution, on which the strategies used in this paper are built, is called the reduced basis method, described in [10]. The goal is to approximate the continuum of waveforms with arbitrarily high accuracy by elements from an N-dimensional vector space of particularly representative waveforms. To do so, we must choose the right subset of the continuum as our basis, so that

$$W_N = \text{span}\{e_1, \ldots, e_N\} \tag{2}$$

where W_N is the N-dimensional function space, and $\{e_1, \ldots, e_N\}$ is called the reduced basis of waveforms. In this case, we could hope to approximate an arbitrary waveform h_μ by its projection onto this linear space, given by

$$P_{W_N}(h_\mu) = \sum_{i=1}^{N} \langle h_\mu, e_i \rangle e_i = \sum_{i=1}^{N} \alpha_i e_i \tag{3}$$

for $\alpha_i \in \mathbb{C}$ the complex projection coefficients of the reduced basis expansion.

It has been shown, using a greedy algorithm, that it is possible to select $\{e_1, \ldots, e_n\}$ such that given a dense grid of parameters $\{\mu_1, \ldots, \mu_M\}$, called the training space, in parameter space, and an error tolerance $\epsilon > 0$, we have

$$\|P_{W_N}(h_{\mu_j}) - h_{\mu_j}\| < \epsilon \tag{4}$$

for every j with $1 \leq j \leq M$. Here, the norm $\|\cdot\|$ is induced by the inner product (1). By choosing a dense training space and invoking the smoothness of h in μ, we can infer that the upper bound on the error $\|P_{W_N}(h_\mu) - h_\mu\|$ of the reduced basis approximation is not much higher than ϵ for arbitrary μ over the entire parameter space. Furthermore, it has been shown that the dimension N of the linear space is relatively small given some small ϵ, and that the maximum error over the training space drops off exponentially as the number of reduced basis elements is increased.

In short, we can indeed represent the continuum of waveforms using relatively few samples, and we can find these samples relatively quickly. The greedy algorithm is simple. We select some parameter, and let the span of the corresponding waveform be the linear space W_1. We then compute the error $\|P_{W_1}(h_{\mu_j}) - h_{\mu_j}\|$ between a waveform h_{μ_j} and its projection onto W_1 for every parameter μ_j in the training space. We then select the waveform h_{μ_j} with the largest such error, and add its span to the linear space W_1 to obtain W_2. We repeat this error computation, and keep adding waveforms to the linear space until the maximum error over the training space is less than ϵ. As we add more elements to the linear space, this maximum error is guaranteed to decrease monotonically, and in fact it does so exponentially. A more detailed explanation of the algorithm is given in [10].

The algorithm produces, as an output, an array with entries α_{ij}, where $\alpha_{ij} = \langle h_{\mu_j}, e_i\rangle$ is the projection coefficient of the i^{th} reduced basis element e_i for the waveform corresponding to the j^{th} parameter μ_j in the training space. Suppose, then, that we detect some signal s, and that we would be satisfied to test this signal against every waveform in the dense training space using the matched filter; that is, we wish to compute $\langle s, h_{\mu_j}\rangle$ for M different choices of j, where M is large. The naive solution would be to evaluate M different integrals. However, since $P_{W_N}(h_{\mu_j}) \approx h_{\mu_j}$ with arbitrarily small error, we have

$$\begin{aligned}
\langle s, h_{\mu_j}\rangle &\approx \langle s, P_{W_N}(h_{\mu_j})\rangle \\
&= \left\langle s, \sum_{i=1}^{N} \langle e_i, h_{\mu_j}\rangle e_i \right\rangle \\
&= \sum_{i=1}^{N} \langle s, \langle e_i, h_{\mu_j}\rangle e_i\rangle \\
&= \sum_{i=1}^{N} \langle s, \alpha_{ij}^{*} e_i\rangle = \sum_{i=1}^{N} \alpha_{ij} \langle s, e_i\rangle
\end{aligned}$$

Since the coefficients α_{ij} are stored outputs of the reduced basis algorithm, we now need only evaluate N integrals, rather than M, to compute an accurate approximation to the inner product of a signal s with a waveform h_{μ_j} in the dense training space. M is large and N is small, so since the reduced basis can be constructed without reference to a signal, a large part of the computational burden has been taken offline.

2.3 Applications of Gravitational Wave Approximation

This, of course, supposes that we are satisfied with obtaining the inner products $\langle s, h_\mu\rangle$ only at the training space points $\mu = \mu_j$, which still gives a discrete picture. To obtain a continuous picture would require us to somehow fill in the gaps in the training space. In particular, we see from the above equalities that if we knew the projection coefficients α_i corresponding to all waveforms h_μ, we could accurately approximate $\langle s, h_\mu \rangle$ for any parameter μ.

This process of rapid gravitational wave detection is one of the primary applications of the work which follows. Our general aim is to devise methods by which to approximate some arbitrary waveform h_μ given some set of known waveforms. We will see that this problem can essentially be reduced to the problem of approximating the projection coefficients α_i corresponding to any parameter μ, but we return to this point in detail later. However, approximating the projection coefficients accurately would directly address the first major application of this work, which is fast signal detection over a continuous, rather than discrete parameter space.

Since generating accurate waveform approximations can be costly and difficult - more so in proportion to the accuracy of the approximations - it is useful to have methods by which to quickly generate new waveforms given some known set of waveforms. For example, suppose we wish to generate waveform approximations using only a minor simplification of the original Einstein equations. Some of these simplifications, for example, result in systems of linear PDEs [11] or nonlinear ODEs [12]. To generate several hundred such waveforms, corresponding to different parameters, takes an extraordinary amount of time and computational power. It may then be useful to start with these highly accurate approximations and fill in the many gaps, rather than using a cheaper, less accurate model with fewer gaps. We will show that this problem of waveform generation is closely related to the signal detection problem through the projection coefficients of the reduced basis method, but it can be considered as a separate application.

There are also related parameter estimation problems which rapid waveform generation would help to solve. Given a signal containing a waveform, one might want to know information about the source from which it came, which is encoded in the parameters of the waveform. There are probabilistic, iterative methods which allow one to search for the desired parameters by integrating the signal against many different waveforms. These methods step through parameter space, integrating the signal against a corresponding waveform at each step, in an attempt to maximize the integral over the parameter space. At each step, information is gathered about which direction in parameter space should be explored in order to find a maximal integral. Therefore, this iterative process requires the rapid generation of the next desired waveforms at each step.

It is therefore worthwhile to investigate methods by which to quickly approximate unknown gravitational waveforms. To this end, we will explore various interpolation techniques. In the next section, we define interpolation in generality, and describe how it can be applied to the problem of gravitational waveform approximation. We will depart from our application and discuss interpolation abstractly in order to set our strategies on solid ground, but we will soon after return to the setting of gravitational waveforms.

3 Interpolation and the Problem of Interpolating Waveforms

3.1 The Interpolation Problem

Imagine that there exists some function $f : \mathbb{R}^n \to \mathbb{C}$, of which we may or may not know the form, or have the ability to evaluate without incurring a restrictive cost. At the very least, we postulate the existence of such a function, but cannot feasibly evaluate it directly at just any point. We do, however, possess a set of m data points $\{(x_i, y_i)\}_{i=1}^m$ for $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{C}$, which we imagine to fall exactly on the graph of f, so that

$$f(x_i) = y_i \text{ for } 1 \leq i \leq m. \tag{5}$$

We call $\{x_i\}_{i=1}^m$ the set of interpolation nodes and $\{y_i\}_{i=1}^m$ the set of evaluations of f at the nodes. Using these data points, we aim to approximate f at new points $x \in M \subset \mathbb{R}^n$, where M is the smallest closed n-rectangle containing all of the interpolation nodes. Furthermore, we require that this approximation $\hat{f} : M \to \mathbb{C}$ of f agree with f on all interpolation nodes; that is

$$\hat{f}(x_i) = y_i \text{ for } 1 \leq i \leq m. \tag{6}$$

Having imposed these restriction, we call $\hat{f}$ the interpolant of f on M, and we use $\hat{f}(x)$ as an approximation of $f(x)$ on M for any $x \in M$. A solution of the interpolation problem will entail the construction of such an interpolant $\hat{f}$ which depends only upon the given data points. We note that the problem was here posed for the case of $f : \mathbb{R}^n \to \mathbb{C}$, which will be sufficient for our purposes, but that it could be similarly formulated in a more general setting.

The solution that we seek to some particular interpolation problem will depend directly on the data points, and indirectly upon our knowledge of the underlying function f which we aim to approximate, insofar as the interpolation technique we select is tailored to this function. We begin by presenting two simple examples of solutions to the interpolation problem, where we choose $n = 1$ and restrict the codomain $\mathbb{C}$ to $\mathbb{R}$, so that $f : \mathbb{R} \to \mathbb{R}$.

The example of an interpolation technique - nearest neighbor interpolation - generates a piecewise constant interpolant, such that for any $x \in M$ with $M = [x_1, x_m]$, we let $\hat{f}(x)$ be equal to the evaluation y_i at the node x_i which is closest to x. For example, let the data points be as given in Table 1 below.

i	x_i	y_i
1	0	1
2	1	2
3	3	0

Table 1: Sample data points.

The corresponding nearest neighbor interpolant $\hat{f}$ is given in Figure 2.

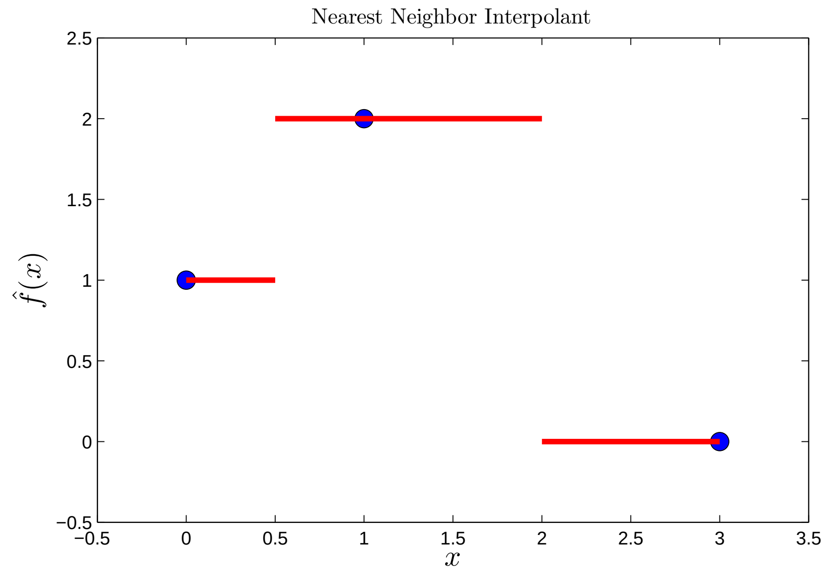

Figure 2: Nearest neighbor interpolant of data in Table 1. Data points are given as blue dots.

We can also generate a piecewise linear interpolant, by simply "connecting the dots". We use the same data points to generate a linear interpolant given in Figure 3.

If we have no information about the underlying function f we wish to interpolate, and we only have a set of seemingly random data points as in the example, we may only be able to do as well in our approximation as we did with the two methods above. However, in practical applications, we will hopefully know something about the structure of the underlying function. The function may be known, and simply expensive to evaluate, so that the interpolant allows us to obtain approximate evaluations of the function more cheaply. Alternatively, a plot of the data points should suggest that the graph of the function has some particular shape; if the data points do not resolve the essential characteristics of the function, we have no hope of generating an accurate interpolant.

Indeed, if the underlying function is smooth and its gradient does not vary much, it will be easy to obtain an accurate interpolant with few data points, but if a function is highly oscillatory or erratic, obtaining an accurate interpolant may require a large number of data points. In general, the more data points we use to generate the interpolant, the more complicated and costly to evaluate the interpolant will be, so we often face a trade off between accuracy and efficiency. However, as we will see in later sections, it is often possible to get away with few data points by choosing an interpolation method which well represents the underlying function.

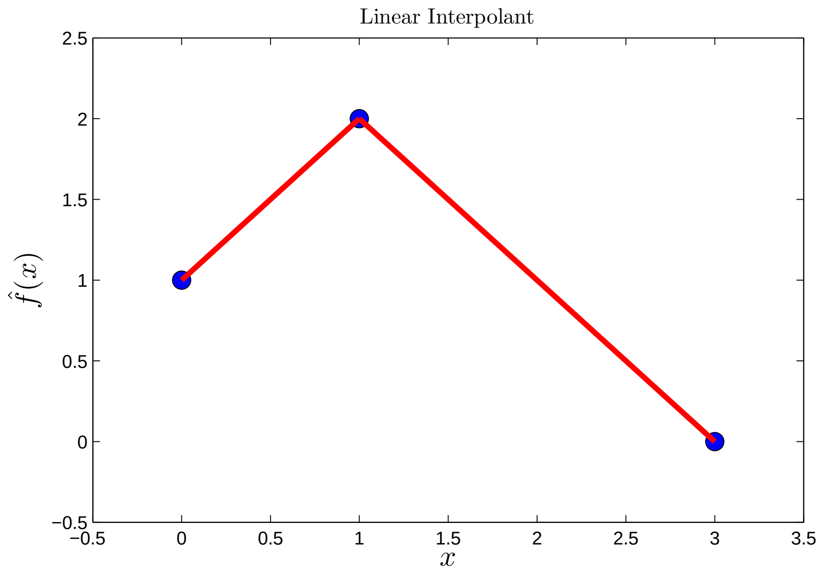

Figure 3: Linear interpolant of data in Table 1.

Before discussing interpolation as it relates to the current application, we will describe an idea that is often used to generate interpolants - it is not a specific technique, like nearest neighbor and linear interpolation, but a general paradigm. We return to the general setting, in which the underlying function is some $f : \mathbb{R}^n \to \mathbb{C}$, and we have the m data points $\{(x_i, y_i)\}_{i=1}^m$. Suppose we have some information about the structure of f; perhaps it is given by trigonometric functions, or it is radially symmetric, or its graph resembles some known surface. We can encode this information into our interpolant by choosing an appropriate interpolation basis, or a set of m functions $\{\phi_i\}_{i=1}^m$ such that $\phi_i : \mathbb{R}^n \to \mathbb{C}$ and the interpolant is a sum

$$\hat{f}(x) = \sum_{i=1}^{m} c_i \phi_i(x) \tag{7}$$

for coefficients $c_i \in \mathbb{C}$. Enforcing the interpolation requirement $\hat{f}(x_i) = y_i$ gives rise to the linear system $\Phi C = Y$, given by

$$\begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_m(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_m) & \phi_2(x_m) & \cdots & \phi_m(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{8}$$

Therefore, solving this system for C by $C = \Phi^{-1} Y$ gives a valid interpolant $\hat{f}$, as long as Φ is invertible. This is not, in general, guaranteed, and depending on the choice of interpolation basis and the data points, Φ may be poorly conditioned or singular. A wide variety of interpolation techniques are special cases of this method. For example, supposing we have $2m+1$ data points, we may choose $\phi_k(x) = e^{ikx}$ and rewrite the sum as

$$\hat{f}(x) = \sum_{k=-m}^{m} c_k e^{ikx} \tag{9}$$

to recover trigonometric interpolation of functions $f : \mathbb{R} \to \mathbb{C}$.

Now that we have defined the interpolation problem, discussed the considerations that must be taken in solving the problem, and provided some examples of solutions, we will define our specific problem, which is to interpolate gravitational waveforms. Note that although we have, thus far, only provided examples of interpolation on underlying functions $f : \mathbb{R} \to \mathbb{C}$, the concepts are easily generalizable to $\mathbb{R}^n$, and we will later introduce techniques which yield interpolants that have $\mathbb{R}^n$ for arbitrary $n \in \mathbb{N}$ as their domain.

3.2 Interpolation of Gravitational Waveforms

Recall that we consider a function $h : (\mathcal{F} \subset \mathbb{R}) \times (M \subset \mathbb{R}^n) \to \mathbb{C}$, where $\mathcal{F} = [f_L, f_U]$ is the frequency domain and M is the parameter space; for our purposes, M is some closed n-rectangle in $\mathbb{R}^n$. A particular waveform, corresponding to some parameter $\mu \in M$, is written as $h_\mu(f)$. The goal is to interpolate the function h in the parameter space M, so as to obtain an approximation of $h_\mu(f)$ for any $\mu \in M$. Assume that we can evaluate $h_{\mu_j}(f)$ for m particular parameters $\{u_1, \ldots, u_m\}$; these will become our interpolation nodes.

It will now be useful to describe the problem we address in less abstract terms. Some of the details which follow are simply included for any reader who wishes to replicate our results, and some are more fundamental to our problem.

The dimensionality of the parameter space M is dependent on the waveform model which we use; a higher-dimensional parameter space will take into account more properties of the waveform and its source. In our waveform models, the source is a system of two massive objects which spiral around each other. We consider two different parameter spaces. In the first, the parameter μ is a function, called the chirp mass $\mathcal{M}$, of the objects' masses m_1 and m_2, given by

$$\mathcal{M}(m_1, m_2) = (m_1 m_2)^{3/5}(m_1 + m_2)^{-1/5} \tag{10}$$

In the second, the parameter $\mu = (m_1, m_2) \in \mathbb{R}^2$ has components given by the masses of the two objects. In both cases, we consider the mass range of $3 - 30$ solar masses. Adding additional parameters would give a higher-dimensional problem, taking into account more features of the sources and the detectors, such as the objects' spins or the detector's orientation.

The waveform models we use are given by the post-Newtonian approximation,

$$h_\mu(f) = \mathcal{A} f^{-7/6} e^{i\left(-\frac{\pi}{4} + \frac{3}{128}(\pi G f \frac{\mathcal{M}}{c^3})^{-5/3} + \cdots\right)} \tag{11}$$

where f denotes frequency, $\mathcal{A}$ denotes an amplitude term, G is the gravitational constant, and c is the speed of light. We indicate that there may be additional terms in the exponential, corresponding to a higher-dimensional waveform. The expressions given above can be found in [8]. We consider a dense grid of values in the frequency domain, across the range $[f_L, f_U]$, where $f_L = 40Hz$ and $f_U \approx 366.338Hz$.

One may wonder why we wish to interpolate the function h if we already have a closed form as given above. However, we simply use these approximate waveforms as models for the problem of astrophysical significance, in which more accurate waveform models are used. The models we use will have characteristics that are similar to the accurate models, but are much easier to generate, and therefore we can easily measure interpolation errors.

We will use the reduced basis algorithm, discussed in Section 2.2, in the process of interpolating the waveforms, so it is necessary to define the Power Spectral Density $S(f)$ which we use for the inner product $\langle \cdot, \cdot \rangle$ defined in (1). It is generated using data from initial LIGO [2], which is fitted by the curve given by

$$S(f) = 9 \times 10^{-46} \left[\left(\frac{4.49}{150} f \right)^{-56} + 0.16 \left(\frac{f}{150} \right)^{-4.52} + 0.52 + 0.32 \left(\frac{f}{150} \right)^{2} \right]. \tag{12}$$

We make one simplification to the problem, based on the information we have about h; we cannot evaluate each known $h_{\mu_j}(f)$ for any $f \in \mathcal{F}$. Instead, the approximate waveforms with which we work are given on some fine but finite and fixed grid $F = \{f_1, \ldots, f_p\}$ with each $f_i \in \mathcal{F}$. That is, for our purposes, h should be considered as a function $M \to \mathbb{C}^p$, so that h_μ is a complex-valued p-vector, with each entry giving the value of h for a fixed $f_j \in F$ at the parameter μ. We could interpolate in the frequency coordinate in order to obtain an approximation of $h_\mu(f)$ at all values of f, but our interest is to obtain an approximation $\hat{h} : M \to \mathbb{C}^p$ of h. In particular, we want $\hat{h}$ to satisfy the interpolation condition

$$\hat{h}_{\mu_j}(f_k) = \hat{h}^k_{\mu_j} = h_{\mu_j}(f_k) \tag{13}$$

for $1 \leq j \leq m$ and every $f_k \in F$; each h_{μ_j} is known on the frequency grid F. Here, the leftmost expression abuses notation slightly in order to resemble the original interpolation condition stated before, and the upper index k in the second expression represents the k^{th} vector component.

This interpolation problem is not, so far, phrased in the language used in the previous section. In particular, we wish to interpolate functions $h : (M \subset \mathbb{R}^n) \to \mathbb{C}^p$, by which we mean that we wish to obtain an approximation of h which satisfies the interpolation condition (13). However, we have only described the interpolation problem on functions mapping subsets of $\mathbb{R}^n$ to $\mathbb{C}$. We must find some way of addressing this discrepancy.

3.2.1 Component-Wise Interpolation

Suppose we break h into p functions $h_j : M \to \mathbb{C}$ for $1 \leq j \leq p$, where each j corresponds to a frequency component; that is, $h_{j_\mu} := h^j_\mu$, where again the upper index j denotes the j^{th} vector component. We can then solve each of these p interpolation problems separately, obtain p interpolants $\hat{h_j}$, and define $\hat{h} : M \to \mathbb{C}^p$ by

$$\hat{h}_\mu = \begin{pmatrix} \hat{h_1}_\mu \\ \hat{h_2}_\mu \\ \vdots \\ \hat{h_p}_\mu \end{pmatrix} \tag{14}$$

so that $\hat{h}^j_\mu = \hat{h_j}_\mu$. This approximation would, in particular, satisfy the interpolation condition (13). For this stitching strategy to be plausible, it is necessary that h be somewhat smooth over the frequency domain, but this is true since h is C^∞ considered as a function $\mathcal{F} \times M \to \mathbb{C}$.

We begin by briefly describing this technique, which we call component-wise interpolation, because it is the most obvious and direct way of reformulating the interpolation problem. It is therefore worth discussing, though it is not the technique on which we will settle. There are two problems with applying component wise interpolation to this problem. Firstly, p is large, so in order to find $\hat{h}_\mu$ for some particular μ, we must solve a large number of interpolation problems at a potentially oppressive computational cost. Indeed, accurate representations of a waveform will be evaluated on a dense grid in the frequency domain, and the cost of the resulting interpolation problem will scale proportionally.

Secondly, we have no guarantee that the functions h_{j_μ} are well-behaved enough to effectively interpolate. In fact, though the graph of h_j over the parameter space M is more manageable for high frequencies, it tends to be highly oscillatory for lower frequencies; so much so that interpolation on these functions might require an unreasonably large number of data points. Figure 4 gives two of these graphs for the one-parameter space; one for a high frequency, and the other for a low frequency. Notice the dramatic oscillations on the low frequency graph for small values of the parameter μ.

In the two-parameter case, the functions h_j are similarly badly behaved for low fixed frequencies. We note that each h_j is a smooth function, so given enough data points, accurate interpolation is possible. However, we would like to obtain an accurate solution of the interpolation problem using as few data points as possible, because the number of data points required in a high-dimensional parameter space can quickly become infeasible, and the generation of data points carries a significant computational cost.

We could select our data points to reflect the location of activity over parameter space; specifically, in the one-dimensional, low frequency plot shown above, we could pack the nodes more densely for lower values of μ. However, though the function is more oscillatory for μ small, there is still a significant amount

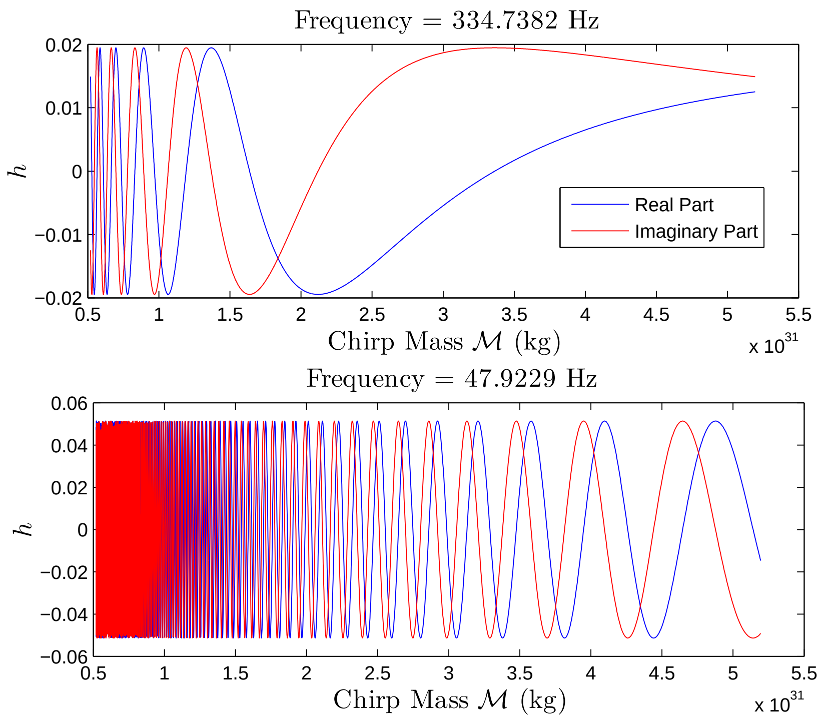

Figure 4: Two waveforms at fixed frequencies over a one-dimensional parameter space.

of activity over the entire parameter space, and we would still require an many data points in the high activity areas. Another potential strategy is to make use of the fact that the functions h_j are better behaved for higher frequencies, and to vary the number of data points we generate for each h_j in proportion to j. We do not explore this strategy of direct component-wise interpolation on the waveforms here, though it may hold some promise if explored; instead, we seek an indirect interpolation technique which makes use of the reduced basis method described above.

3.2.2 Interpolation of Reduced Basis Coefficients

Since the reduced basis method can be used to represent the continuum of waveforms using a small basis, we ask if it is possible to similarly reduce the amount of work we must do to interpolate waveforms by using these basis expansions. Recall that, using the reduced basis technique, we can generate an N-dimensional vector space of waveforms

$$W_N = \text{span}\{e_1, \ldots, e_N\} \tag{15}$$

such that we can accurately approximate an arbitrary waveform h_μ by its projection onto this space. The approximation error can be made arbitrarily small by choosing N sufficiently large, but in practice small error can be obtained with N relatively small. The projection of a particular waveform h_μ is given by

$$P_{W_N}(h_\mu) = \sum_{i=1}^{N} \langle h_\mu, e_i \rangle e_i = \sum_{i=1}^{N} \alpha_i e_i \tag{16}$$

for $\alpha_i = \langle h_\mu, e_i \rangle \in \mathbb{C}$. We can equivalently consider P_{W_N} as a function of the parameter μ to obtain

$$P_{W_N}(\mu) = \sum_{i=1}^{N} \langle h_\mu, e_i \rangle e_i = \sum_{i=1}^{N} \alpha_i(\mu) e_i \tag{17}$$

Here $\alpha : M \to \mathbb{C}^N$ is a function which takes a parameter value and returns a vector of expansion coefficients, so that $\alpha_i(\mu)$, the i^{th} component of $\alpha(\mu)$, is the i^{th} coefficient α_i in the expansion (16). To obtain $\alpha(\mu)$ for some $\mu \in M$, however, requires taking an inner product, which is a costly operation. Our strategy is to perform component-wise interpolation on $\alpha(\mu)$ over parameter space to obtain an interpolant $\hat{\alpha}(\mu)$. N is not too large, so performing N separate interpolation problems is feasible. Furthermore, $\alpha_i(\mu) = \langle h_\mu, e_i \rangle$, so it is smooth for each i, since it is given by a smooth inner product of smooth functions. If we succeed, we can, for arbitrary $\mu \in M$, obtain an approximation of $P_{W_N}(\mu)$, and therefore an approximation $\hat{h}_\mu$ of h_μ, by

$$\hat{h}_\mu = \sum_{i=1}^{N} \hat{\alpha}_i(\mu) e_i \tag{18}$$

By interpolating the waveforms via their projection coefficients, we obtain an approximation of an arbitrary waveform h_μ at every frequency value in the domain, so the method operates independently of the density of the frequency grid. The error in the approximation is a function of the N interpolation errors from each interpolation problem and the error of the reduced basis expansion, the latter of which can be made arbitrarily small. It remains to verify that interpolation of the functions $\alpha_i(\mu)$ is feasible; that is, that the activity of the functions can be resolved and captured using relatively few data points $(\mu, \alpha(\mu))$. To this end, we ought to sample $\alpha(\mu)$ on a fine grid in parameter space, and examine the resulting graphs of $\alpha_i(\mu)$ for all choices of $1 \leq i \leq N$.

We note that, as a result of the greedy selection strategy used in the reduced basis algorithm, the importance of the terms in the expansion (16) tends to decrease as the index i increases; that is, the magnitude of α_i tends to be lower for large i than for small i. In particular, the last few terms of the sum act as small adjustments. To illustrate this, we generate a reduced basis of 178 elements for the one-dimensional problem, and plot the average, over a grid of 3000 choices of μ, of the magnitude of $\alpha_i(\mu)$ for each i. This plot is given in

Figure 5. Note that the greedy algorithm randomly selects the first reduced basis element, corresponding to the coefficient α_1, so we do not necessarily expect its magnitude to be large.

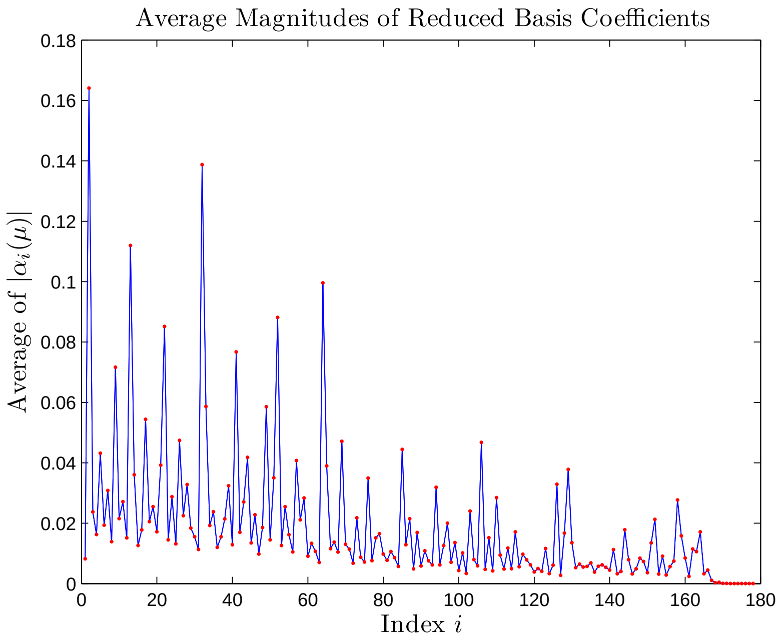

Figure 5: Average magnitudes of reduced basis coefficients $\alpha_i(\mu)$ over 3000 choices of μ.

We see that, indeed, the magnitude of α_i tends to decrease as i increases; in particular, the last few coefficients have magnitude near zero. Therefore, even if the relative interpolation error is higher for the last few coefficients, the absolute interpolation error of the sum will likely not be affected. We note this because the last few coefficients are more poorly behaved as functions of μ than the lower-indexed coefficients.

To give an idea of the structure of the functions we will interpolate, and to show that it is reasonable to interpolate these functions, we first consider the one-parameter problem, generating a reduced basis of 178 elements. We pick several choices of i and observe plots of $\alpha_i(\mu)$. For most choices of i, the activity of $\alpha_i(\mu)$ is highly concentrated; there is a small interval in which the function oscillates with large amplitude, and elsewhere it is near flat with low amplitude oscillations. Presumably the large oscillations occur near those parameters μ for which the i^{th} coefficient plays an important role in the reduced basis of the corresponding waveform - in other words, we see oscillations for parameters μ such that h_μ is somehow well-represented by e_i. We zoom in on locations in parameter space with both high activity and low activity in order to paint a clear picture of the structure of the one-parameter functions $\alpha_i(\mu)$.

For most such plots, the activity is so concentrated that a plot over the entire parameter space is not particularly instructive or clear. However, Figure 6 gives the plot of the 5^{th} coefficient over the entire parameter space $M = [.52 \times 10^{31} kg, 5.19 \times 10^{31} kg]$. Since the index of this coefficient is low, it is quite well-behaved as a function of $\mathcal{M}$, the chirp mass.

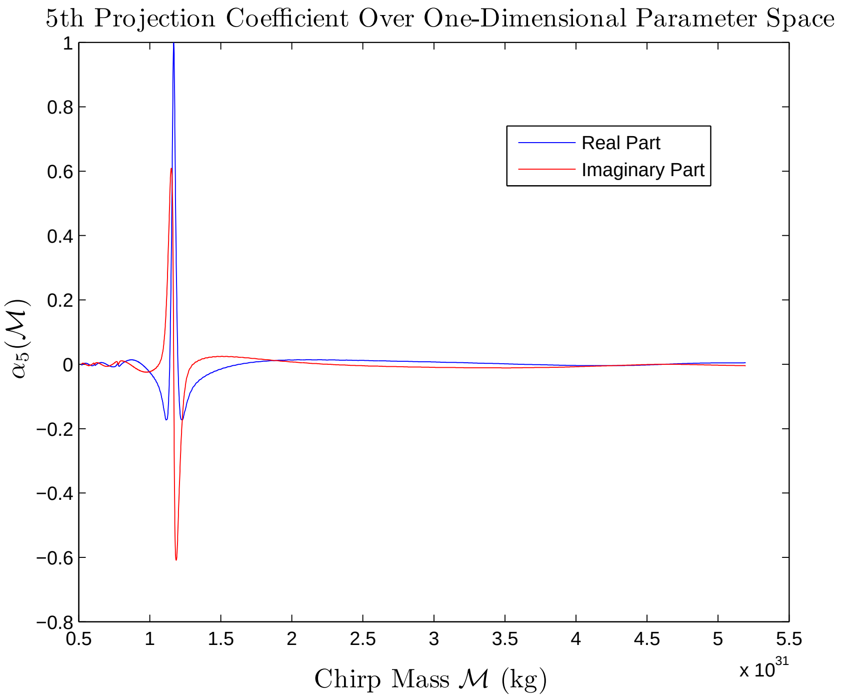

Figure 6: Plot of 5th projection coefficient over the entire one-dimensional parameter space.

It is clear from Figure 6 that in order to accurately interpolate this function, it will suffice to place many nodes in the interval $\approx [.52 \times 10^{31}, 2 \times 10^{31}]$ and few nodes elsewhere. Other choices of i yield similar pictures, though many of them are less well-behaved. Consider the $i = 100$ case. In Figure 7, we give a plot of the 100^{th} projection coefficient over an interval of high variation, and in Figure 8, we give a plot of the same coefficient over an interval of the same length with lower variation.

These plots suggest that it is reasonable to attempt to interpolate the 100^{th} projection coefficient, but we ought to place many more interpolation nodes in the region of higher variation in order to effectively capture the structure of the function. The interval used in Figure 8 still has more activity than we see in most of the function; as we increase $\mathcal{M}$, the oscillations dampen further and the activity decreases. Also notice that as $\mathcal{M}$ increases, the amplitude of the

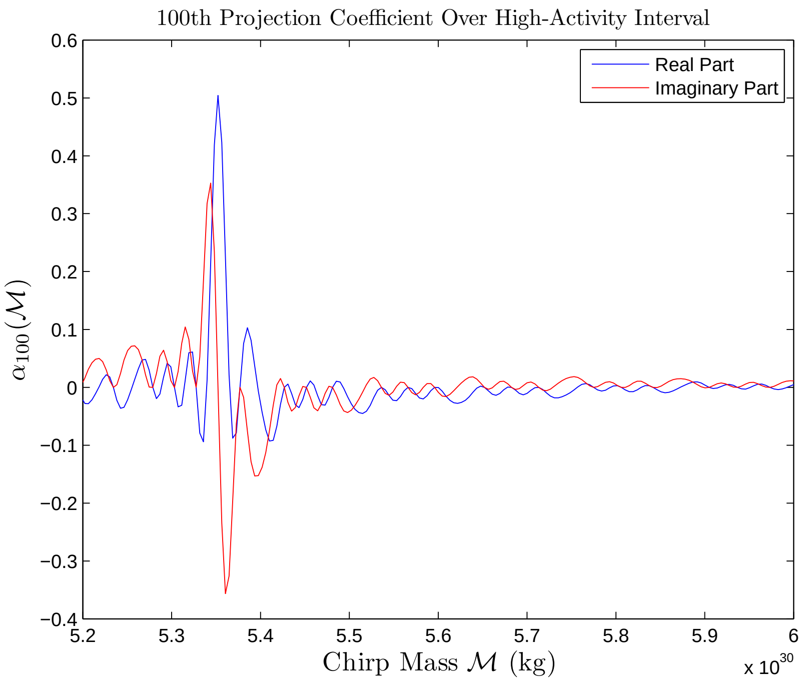

Figure 7: Plot of 100th projection coefficient over the interval $[5.2 \times 10^{30}, 6 \times 10^{30}] \subset M$.

oscillations decrease, so there is only a small contribution to the absolute error of the interpolant made in regions of large $\mathcal{M}$. This further suggests that we can get away with fewer data points in the flatter regions of the function, and since the majority of the function is relatively flat, we should not need an excessive number of data points in order to interpolate accurately.

As discussed before, the projection coefficients with very high index behave more erratically, although they still tend to show more variation for lower $\mathcal{M}$, so they count for less in terms of the final absolute interpolation error. Therefore, instead of including a plot of a higher-indexed projection coefficient, it suffices to note that these functions tend to obey some of the same important structural qualities as the lower-indexed functions, so that they will not throw off our final choice of interpolation nodes too much.

We observe very similar qualities in the functions $\alpha_i(\mu)$ for the two-parameter case in which $\mu = (m_1, m_2) \in M \subset \mathbb{R}^2$. Here, in parallel to the one-parameter case, we observe a ridge-like structure in the region of small m_1 and m_2, with lower-amplitude ridge-like oscillations propagating outward. A surface plot of the real part of $\alpha_{20}(m_1, m_2)$ is given in Figure 9 below; the structure of the imaginary part is similar.

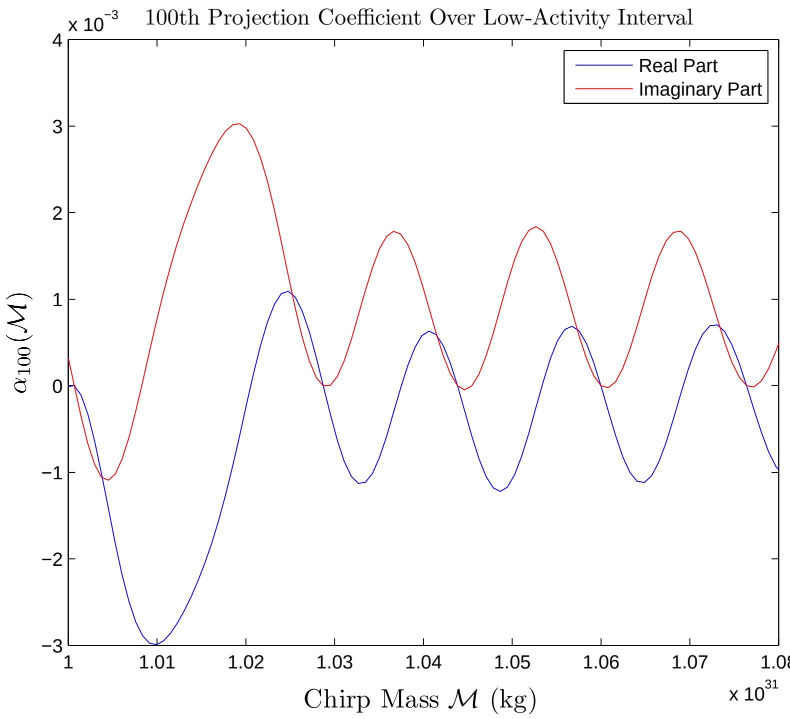

Figure 8: Plot of 100th projection coefficient over the interval $[1 \times 10^{31}, 1.08 \times 10^{31}] \subset M$.

We now have a procedure by which to interpolate the function h at some fixed parameter μ_0. We first generate a reduced basis of orthogonalized waveforms, so that an arbitrary waveform h_μ can be represented by (16) with high accuracy. Then, for each of N interpolation coefficients, we sample m data points $\{\mu_j, \alpha_i(\mu_j)\}_{j=1}^m$ by

$$\alpha_i(\mu_j) = \langle h_{\mu_j}, e_i \rangle \tag{19}$$

for e_i the i^{th} reduced basis vector. We will choose the m nodes μ_j so as to best capture the structure of the function $\alpha_i(\mu)$ with the smallest possible m, clustering nodes around locations of high activity if possible. We will then interpolate each of these N functions at μ_0 to obtain interpolated projection coefficients $\hat{\alpha}_i(\mu_0)$, and then use Equation (18) to obtain the interpolated waveform $\hat{h}_{\mu_0}(f)$ for every $f \in F$, as desired.

3.2.3 Challenges to Solving the Interpolation Problem

We now discuss some challenges to the technique, which are sufficiently general to apply to a wide variety of interpolation problems, but we discuss them in

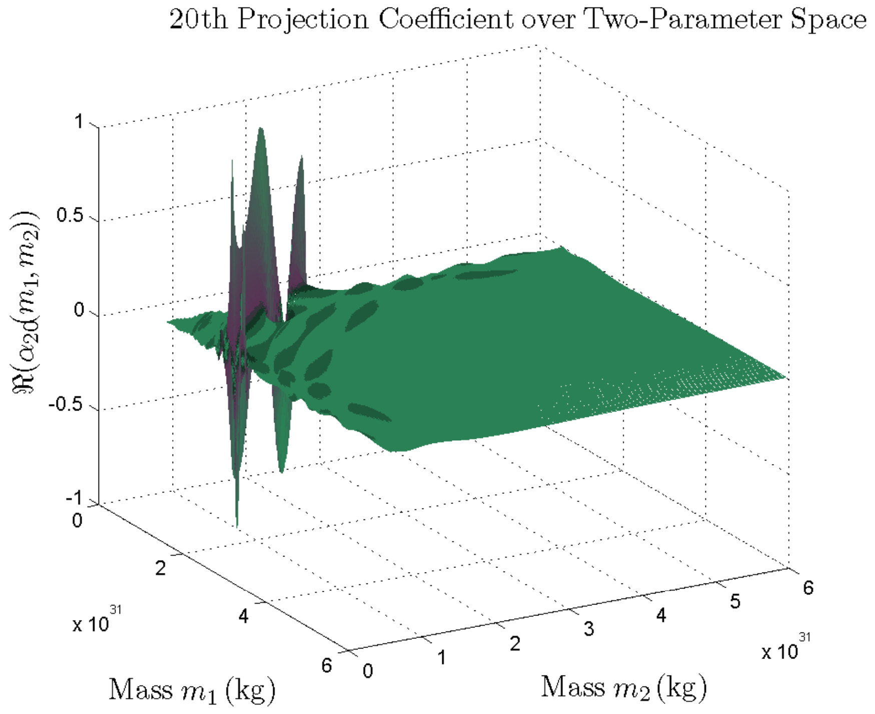

Figure 9: Plot of the real part of the 20th projection coefficient over the two-dimensional parameter space.

the context of our particular strategy. We will construct a checklist of points to consider when filling in the details of our procedure, which will inform our choice of a specific interpolation method.

One of the primary challenges to consider in formulating a solution to this interpolation problem is the curse of dimensionality. For the one-parameter problem, interpolation turns out to be relatively easy using simple techniques and few data points. We simply cluster data points near the concentrated regions of activity, and this is enough to resolve the behavior of the function. Figures 6, 7, and 8 in the previous section illustrate this point. However, consider Figure 9, which gives an example of the functions we wish to interpolate in the two-parameter problem. Clearly, we require far more data points to resolve the regions of high activity. Intuitively, it is easier, using a finite number of dots, to form a detailed picture in the plane than it is in space. Though we cannot yet make a precise statement about the growth of the number of nodes required to resolve the functions, we can expect it to be significant. If the parameter space is, say, 10-dimensional, as it perhaps would be for the astrophysically-relevant problem, we likely cannot hope to generate enough data points to resolve the behavior of the function, because of computational restrictions.

This particular challenge will inform many aspects of our interpolation strategy. For one, we must use as few data points as possible, as the generation of each data point carries a computational cost. We must also make the most out of the m data points which we have the computational power to generate. We would also like to choose an interpolation method which is generalizable to arbitrary dimensions, and whose form does not change much when we change the dimension of the space in which it is applied; we want a method that scales. For example, linear interpolation, as we formulated it, only works on subsets of $\mathbb{R}$, but nearest neighbor interpolation is easily scalable to subsets of $\mathbb{R}^n$.

It is important at this point to make a distinction between computations which occur "off line" and those which occur "on line". Some of the parts of the problem can perfectly well be performed off line, using the most powerful computers available and a reasonably large amount of time. Once we have obtained our outputs, they have many general uses, and we can save them. For example, we can generate a reduced basis of waveforms off line, store it, and have available to us a space in which we can approximate any waveform with high accuracy. However, other aspects of the problem might need to be solved on line, on the go. For example, we might need to quickly generate approximations of waveforms in certain locations in parameter space, based on signals that detectors are reporting at a particular moment. In general, we assume that the generation of interpolants for specific waveforms will be performed on line. However, if we can factor out certain steps and perform them off line at leisure, we can potentially reduce the number of on line computations required.

This point illuminates importance of the distinction between closed-form interpolants and algorithmically-evaluated interpolants, in addition to the related distinction between global and local interpolants. Before now, we have been vague about these distinctions. Suppose we wish to interpolate some function $f : (M \subset \mathbb{R}^n) \to \mathbb{C}$ at a fixed point $x_0 \in M$. A closed-form interpolant is a function $\hat{f} : M \to \mathbb{C}$ which obeys the interpolation conditions and which we can write down and evaluate at any point in M. If we have a closed-form interpolant $\hat{f}$, we simply take $\hat{f}(x_0)$ as our approximation of $f(x_0)$. All of the examples of interpolants discussed in Section 3 are closed-form interpolants.

However, suppose we only care about approximating the value of f at x_0, and not at any other point in M. In this case, we may potentially do more work than is necessary by obtaining a closed-form interpolant which can be used over the whole parameter space M. In particular, an efficient algorithm may exist that takes the data points as an input and produces only the approximation $\hat{f}(x_0)$. In this case, we call $\hat{f}$ an algorithmically-evaluated interpolant. Later, when we discuss polynomial interpolation, we will see an example of a method in which both closed-form and algorithmically-evaluated forms of the interpolant are available, and the choice of which to use is problem-specific.

Related is the notion of global and local interpolants. Suppose we wish to interpolate the value of f at $x_0 \in M$, but that the behavior of the function far away from x_0 is irrelevant to the value of f at x_0. Rather, we may wish to consider a subset U of M, and rephrase the interpolation problem in terms of a new underlying function $f|_U : U \to \mathbb{C}$, using as our data points the subset of data points over M whose nodes fall in U. We call this local interpolation. Alternatively, for every $x_0 \in M$, we may wish to use all of the data points in order to interpolate f at x_0, in which case we say that the interpolation is global.

If we can obtain a global, closed-form interpolant which is accurate, we have a solution to the interpolation problem at every point in M. However, unless the function f is very simple, or the interpolation method is very well suited to f, obtaining such an interpolant may be too costly, and its closed form may still be unreasonably complex. If only limited information is available about the underlying structure of a function, it will be difficult to capture all of the features of this structure in a reasonable amount of time and a concise form. The more data points we use, the more complicated the closed-form interpolant will be, since it is necessarily a function of the data points. We can imagine the difficulty of writing down a global, closed-form interpolant of the two-parameter coefficient plotted in Figure 9.

A global, algorithmically-generated interpolant usually does not make sense; we would only use an algorithmically-generated interpolant if we only wish to interpolate at a point, but the value of a function at a point is often more efficiently approximated using only data points nearby, rather than all data points. This is not necessarily true; if an underlying function has a regular structure which is well-suited to a particular interpolation method, we may throw away important information by using a local technique. A local, closed-form interpolant does not make sense if a faster algorithmically-generated interpolant is available, because it is unnecessary to have a closed form of the interpolant if we will only use it once.

In this paper, we tend to use local, algorithmically-generated interpolants, though some of the local methods we use produce closed-form interpolants which we evaluate at the point of interest and then throw away. We are interested in finding some waveform at a particular parameter μ; therefore, it suffices to interpolate all of the projection coefficients at μ. We consider this to be the on line phase of the problem. If we could construct a global, closed-form interpolant of the projection coefficients, the entire problem could be solved off line, except for the evaluation of the interpolant at a specific point. However, such an interpolant would be so complicated - most likely it would be a sum of many thousands of terms - that generating it would be computationally straining, storing it would be burdensome, and evaluating it at some point μ might take as long as performing a local interpolation at μ. We could construct such an interpolant in theory, but it would be, at best, excessively inelegant, like trying to pack a sphere with thousands of jagged objects of different shapes and sizes.

Instead, we generate a fine, fixed global set of data points off line, and perform on line local interpolations using these data points; that is, given a parameter μ_0, we perform an order k local interpolation by choosing the k nodes closest to μ_0 and solving an interpolation problem using the corresponding data points.

An important challenge, then, lies in making a proper choice of the global interpolation nodes. We should attempt to place the nodes carefully so as to resolve those locations in parameter space in which the plots of the projection coefficients tend to show more activity. By investigating the structure of the underlying waveforms, we will gain clues about how to optimize the global set of data points; this will be discussed at length later.

In choosing the data points, we also must take into account the specific interpolation method that we use, since some methods require the nodes to be arranged in a regular - not necessarily equidistant - grid. In higher dimensions, it may be difficult to generate and store enough data points to fill a grid, and there may be some optimal choice of nodes which captures a great deal of information about the function, but which are scattered. It may be worthwhile, then, to explore interpolation methods which give a high degree of freedom in the choice of interpolation nodes.

We therefore seek a method which is easily scalable to higher dimensions, which allows us to perform accurate local interpolations at a low computational cost, and which gives some freedom in our choice of interpolation nodes. The next section again departs from our specific application in favor of a general discussion of interpolation methods, and in particular, we consider methods which fulfill all or most of these requirements.

4 Interpolation Methods

Given a set of data points, there are many ways that we can draw a curve through these points. The nearest neighbor and linear interpolation methods presented before are two simple possibilities, but do not tend to accurately capture the structure of very many functions. In this section, we present a few different interpolation techniques in generality, and discuss their strong and weak points. We begin with a technique which is simple, but efficient and effective; polynomial interpolation.

4.1 Polynomial Interpolation in One Dimension

Suppose we have m data points $\{(x_i, y_i)\}_{i=1}^m$ for $(x_i, y_i) \in \mathbb{R} \times \mathbb{C}$ which we imagine to be drawn from an underlying function $f : \mathbb{R} \to \mathbb{C}$, so that $f(x_i) = y_i$. Interpolation of these data points via polynomials consists of constructing a polynomial that passes through the data points, and evaluating that polynomial at a new point of interest.

4.1.1 Basic Results Concerning Polynomial Interpolation

Polynomial interpolation is made possible by the following result.

Theorem 1. ***(Lagrange Polynomial)*** *Consider the set $\{(x_i, y_i)\}_{i=1}^m$, where $(x_i, y_i) \in \mathbb{R} \times \mathbb{C}$. Suppose $x_i \neq x_j$ for $i \neq j$. Then there exists a polynomial $p(x)$*

with complex coefficients of degree at most $m-1$ which interpolates $\{(x_i, y_i)\}_{i=1}^m$; that is, for every $1 \leq i \leq m$, we have

$$p(x_i) = y_i \tag{20}$$

Proof. Define m polynomials l_i by

$$l_i(x) := \prod_{\substack{1 \leq j \leq m \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Notice l_i is an $m-1$ degree polynomial for each i, and it is well defined, because $x_i \neq x_j$ for any j in the product. Fixing i, we can check

$$l_i(x_k) = \delta_{ik} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}.$$

Let

$$p(x) := \sum_{i=1}^m y_i l_i(x)$$

which is a polynomial with complex coefficients of degree at most $m-1$. We claim that this polynomial interpolates $\{(x_i, y_i)\}_{i=1}^m$. Indeed, fixing some k with $1 \leq k \leq m$, we have

$$\begin{aligned} p(x_k) &= \sum_{i=1}^m y_i l_i(x_k) \\ &= \sum_{i=1}^m y_i \delta_{ik} \\ &= y_k \delta_{kk} = y_k \end{aligned}$$

as desired. □

The polynomial which was constructed in the proof is called the Lagrange interpolating polynomial. We confirm that the Lagrange polynomial is unique; that is, given any polynomial of degree at most $m-1$ interpolating m data points, this polynomial is precisely the Lagrange polynomial constructed above.

Theorem 2. ***(Uniqueness of the Lagrange Polynomial)*** *Suppose two polynomials p and q of degree at most $m-1$ interpolate the points $\{(x_i, y_i)\}_{i=1}^m$. Then $p = q$.*

Proof. Let $r(x) = p(x) - q(x)$. The degree of $r(x)$ is at most $m-1$, since both $p(x)$ and $r(x)$ have degree at most $m-1$. Furthermore, $r(x)$ has at least m roots, since for every i such that $1 \leq i \leq m$, we have

$$r(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0$$

Therefore, by the Fundamental Theorem of Algebra, we must have $r(x) = 0$, since the degree of r is less than m. This gives $p(x) = q(x)$. □

Notice that polynomial interpolation falls under the framework described at the end of Section 3, in which an interpolant is constructed as a linear combination of some interpolation basis. Indeed, we wish to approximate the underlying function f by

$$\hat{f}(x) = \sum_{i=1}^{m} c_{i-1} x^{i-1} \tag{21}$$

and we can obtain the coefficients c_i by solving the linear system

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_m & x_m^2 & \cdots & x_m^{m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{22}$$

To prove that the Lagrange polynomial is unique, it would also suffice to show that this interpolation matrix, called the Vandermonde matrix, has non-zero determinant, but the proof given above is considerably simpler.

We next give an error bound on the polynomial interpolant; we show that a bound on the interpolation error is determined by the m^{th} derivative of the underlying function f and by the interpolation nodes. Though this result is, for our purposes, of limited practical use, a simple error bound is an important feature of polynomial interpolation, and it helps us to understand one of its primary limitations. The following proof was found in [13].

Theorem 3. ***(Error Bound on Polynomial Interpolation)*** *Suppose that $\{(x_i, y_i))\}_{i=1}^m$ are distinct nodes with $f(x_i) = y_i$ for some $f : \mathbb{R} \to \mathbb{C}$. Let $a = \min\{x_1, \ldots, x_m\}$ and let $b = \max\{x_1, \ldots, x_m\}$. Suppose $f \in C^m[a, b]$, and p is the unique polynomial of degree at most $m - 1$ which interpolates the data points. For every $x \in [a, b]$, there exists $c \in (a, b)$ such that the error $f(x) - p(x)$ is given by*

$$\epsilon(x) := \frac{1}{m!} f^{(m)}(c) \prod_{i=1}^{m} (x - x_i) \tag{23}$$

Notice the similarity between Theorem 3 and the Lagrange form of the remainder in Taylor's Theorem; simply abandon the requirement that all nodes be distinct and set all nodes to be equal.

Proof. Suppose $x = x_i$ for some i with $1 \leq i \leq m$. Then $f(x) = p(x)$ and $\epsilon(x) = 0$ as desired, so we fix some $x \in (a, b)$ that is not a node.

Let $\omega : \mathbb{R} \to \mathbb{R}$ be given by

$$\omega(t) = \prod_{i=1}^{m} (t - x_i)$$

Since x is fixed and $x \neq x_i$ for any i, $\omega(x)$ is fixed and non-zero. Let λ be a constant given by

$$\lambda = \frac{f(x) - p(x)}{\omega(x)}$$

Lastly, define $\phi : \mathbb{R} \to \mathbb{R}$ by

$$\phi(t) = f(t) - p(t) - \lambda\omega(t)$$

Notice ϕ has $m+1$ roots in $[a,b]$; at each of the m nodes x_i we have $f(x_i) = p(x_i)$ and $\omega(x_i) = 0$, and we also have $\phi(x) = 0$. Furthermore, ϕ is m-times differentiable in $[a,b]$, because p and ω are polynomials, and $f \in C^m[a,b]$. We can then apply Rolle's Theorem m times to find that $\phi^{(m)}(t)$ has one root in (a,b); that is, there is some $c \in (a,b)$ with $\phi^{(m)}(c) = 0$. This gives

$$f^{(m)}(c) - p^{(m)}(c) - \lambda\omega^{(m)}(c) = 0$$

However, p is a polynomial of degree at most $m-1$, so $p^{(m)}(t) = 0$. Also, since $\omega(t)$ is a degree m monomial, $\omega^{(m)}(t) = m!$. This gives

$$f^{(m)}(c) = \lambda m! = \frac{f(x) - p(x)}{\omega(x)} m!$$

Rearranging and substituting the original definition of ω into this expression gives the desired result. $\square$

As stated before, this bound will not help us predict the error of a specific interpolation. However, now that we see the connection between polynomial interpolation error and the derivatives of the underlying function, we can describe some of the challenges encountered in using polynomial interpolation.

4.1.2 Runge's Phenomenon & Limitations of Polynomial Interpolation

Runge's function, given by

$$f(x) = \frac{1}{1 + 25x^2} \tag{24}$$

is the classic pathological example in polynomial interpolation. In Figure 10, we give plots of 5th, 9th, and 13th order polynomial interpolants on the interval $[-1,1]$ using equispaced nodes.

We notice that as the order of the interpolant increases, the error near $x = \pm 1$ begins to blow up due to increasingly large oscillations. Indeed for Runge's function with equispaced nodes, as the order of the polynomial interpolant increases, so does the maximum interpolation error.

This observation may be counterintuitive; one might think that as the number of interpolation nodes increases, the interpolation error should decrease. The key to justifying this behavior is in the error bound given in Thereom 3. In Figure 11, we give plots of the functions $\frac{|f^{(m)}(x)|}{m!}$ on a logarithmic scale for the first few values of m. This quantity is proportional to an upper bound on the error of the corresponding polynomial interpolant. We wish to show that the maximum value of the derivatives of f grow fast enough so that this error

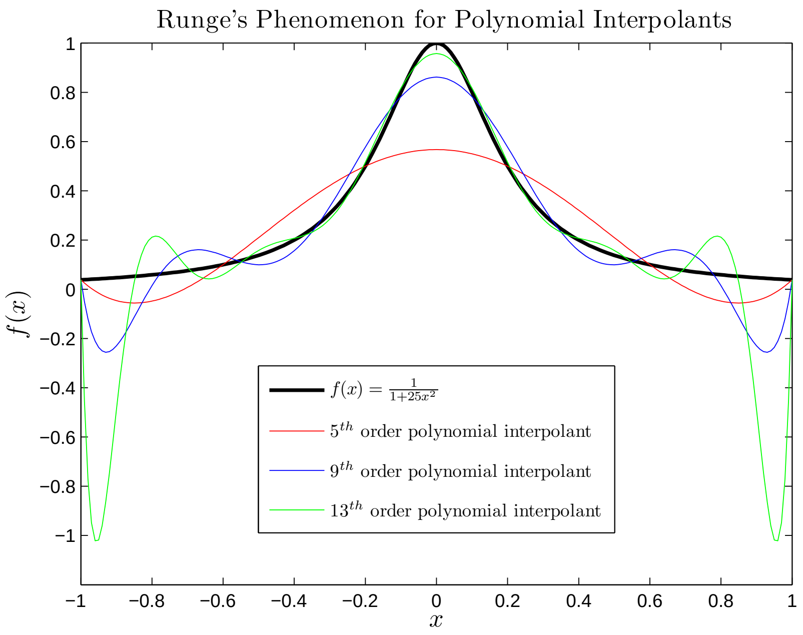

Figure 10: Demonstration of Runge's Phenomenon for $f(x) = \frac{1}{1+25x^2}$; notice the oscillations of the polynomial interpolants near $x = \pm 1$.

bound increases with m. Indeed, Figure 11 shows that for $0 \leq m \leq 4$, this quantity increases quickly; it continues to do so for larger values of m.

Certainly, since the error bound is an upper bound, the rapid growth of the derivatives of f as m increases does not guarantee a corresponding growth in the actual interpolation error. However, Runge's function simply shows that such a blow-up in error is possible, and that a larger-order interpolant does not guarantee a lower interpolation error.

By choosing particular non-equispaced arrangements of interpolation nodes, we can often mitigate Runge's phenomenon. In particular, placing more nodes near the endpoints of the interval can decrease oscillations. There are arrangements of nodes, like Chebyshev nodes, which give a decrease in the interpolation error as the order of the interpolation increases for large classes of functions.

Our strategy is, instead, to use low-order local interpolants, because Runge's phenomenon only becomes noticeable when the number of interpolation nodes becomes large. We could attempt to construct global interpolants, but we would then be forced to select nodes which mitigate Runge's phenomenon, rather than nodes which efficiently represent the underlying function, and in our application, we can not afford to make this sacrifice. Runge's phenomenon then makes the impossibility of global polynomial interpolants clear; for any given choice of interpolation nodes, a polynomial interpolant of such high order would likely

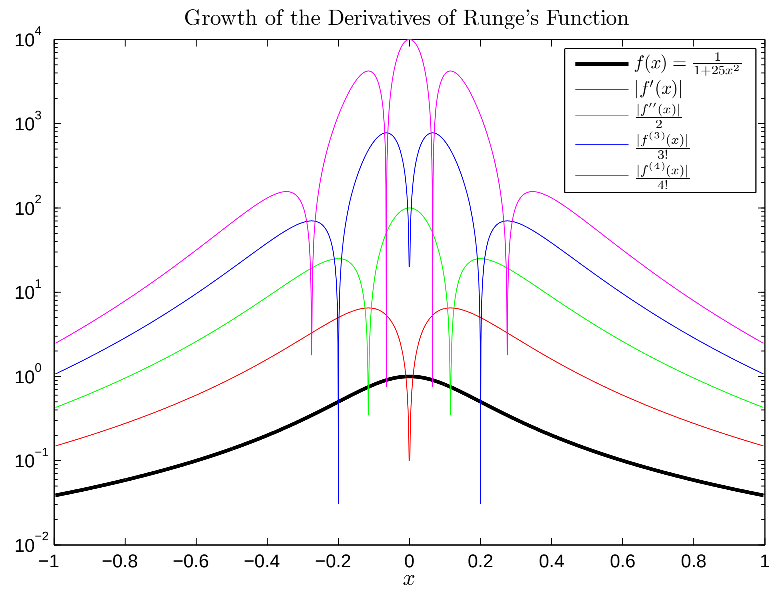

Figure 11: The growth of the derivatives of f is related to the growth of the polynomial interpolant error bound.

contain out-of-control oscillations, and yield an enormous interpolation error.

4.1.3 Implementing Polynomial Interpolation: Neville's Algorithm

There are a number of ways to generate local polynomial interpolants. Of course, we could use the construction of the Lagrange polynomial given in Theorem 1, simply writing down the sum

$$p(x) = \sum_{i=1}^{m} y_i \prod_{\substack{1\leq j\leq m \\ j\neq i}} \frac{x - x_j}{x_i - x_j} \tag{25}$$

This will give a closed-form interpolant, which we may then evaluate at some point of interest. However, in practice, this construction is somewhat inelegant, and evaluating the resulting expression still requires $O(m^2)$ operations.

Alternatively, we could invert the Vandermonde matrix given by Equation (22) to obtain the coefficients of the Lagrange polynomial, construct the polynomial, and evaluate it at some point. There are algorithms that make use of the particular structure of the Vandermonde matrix to invert it using $O(m^2)$ operations [14].

Since we only need to evaluate any given local interpolant at one point, an algorithmically-evaluated interpolant suffices. We use Neville's Algorithm, which is straightforward, easy to code, and requires $O(m^2)$ operations to return the value of the polynomial interpolant at some point given m data points [15]. Since we only construct local interpolants, m is small, so the algorithm is sufficiently fast.

Fix m data points $\{(x_i, y_i)\}_{i=1}^m$ and some point x in the domain of interpolation. Neville's Algorithm makes use of the recurrence relation given by the following Lemma.

Lemma 1. *Let $p_{k,k+n}(x)$ for $k + n \leq m$ be the unique polynomial of degree $n - 1$ which interpolates the points $\{(x_i, y_i)\}_{i=k}^{k+n}$, so in particular $p_k(x) = y_k$. Then for $n \geq 1$, we have*

$$p_{k,k+n}(x) = \frac{(x - x_{k+n})p_{k,k+n-1}(x) + (x_k - x)p_{k+1,k+n}(x)}{x_k - x_{k+n}} \tag{26}$$

Proof. Fix some k. Note that the expression on the right is a polynomial of degree $n - 1$, so by the uniqueness of the Lagrange polynomial, it suffices to show that it interpolates the points $\{(x_i, y_i)\}_{i=k}^{k+n}$. This can be checked directly; the $x = x_k$ and $x = x_{k+n}$ cases follow directly from the expression, and the rest follow since $p_{k,k+n-1}(x_j)$ and $p_{k+1,k+n}(x_j)$ agree for $k + 1 \leq j \leq k + n - 1$. □

If we fix some x, we seek the number $p_{1,m}(x)$, and we can use this recurrence relation to find it. The resulting algorithm is Neville's Algorithm. We simply start with $n = 1$, vary k from $m - 1$ to 1, and use the relation to find all values $p_{k,k+1}(x)$. We then move on to $n = 2$, finding all values $p_{k,k+2}(x)$ for $1 \leq k \leq m - 2$. We repeat until $n = m$, at which point letting $k = 1$ yields the desired interpolant. This amounts to filling in the following $m \times m$ upper triangular array and taking the entry $p_{1,m}(x)$ in the upper-right corner as the interpolant; we fix $m = 4$ for simplicity.

$$P = \begin{pmatrix} p_1(x) & p_{1,2}(x) & p_{1,3}(x) & p_{1,4}(x) \\ 0 & p_2(x) & p_{2,3}(x) & p_{2,4}(x) \\ 0 & 0 & p_3(x) & p_{3,4}(x) \\ 0 & 0 & 0 & p_4(x) \end{pmatrix} \tag{27}$$

Notice that to fill in an entry of P using the recursion relation (26), we must know the entry below it and the entry to the left of it. Therefore, in the example $m = 4$ above, we compute the off-diagonal entries of P in the following order:

$$p_{1,2}(x) \to p_{2,3}(x) \to p_{1,3}(x) \to p_{3,4}(x) \to p_{2,4}(x) \to p_{1,4}(x)$$

We give pseudocode for Neville's Algorithm in Algorithm 1 below. Notice that Neville's Algorithm requires $O(m^2)$ operations; we must step through m choices of n, and for each n, we must step through at most $m - 1$ values of k.

Using this algorithm, we can, given a global grid of data points in one-dimensional parameter space, fix some m, choose m data points around some parameter μ of interest, and carry out a fast order $m-1$ polynomial interpolation at μ. We next discuss a simple way of extending this method to an n-dimensional domain.

Algorithm 1 Neville(X, Y, x)

```
Input: An array of $m$ interpolation nodes $X$, an array of evaluations $Y$ corresponding to the nodes $X$, and a point $x$ at which to interpolate.
Output: The polynomial interpolation at $x$ given the data points $(X, Y)$.
for $j = 1$ to $m$ do
  $P(j, j) = Y(j)$
end for
for $j = 2$ to $m$ do
  for $i = j - 1$ to 1 do
    $P(i, j) = \frac{(X(j)-x)P(i,j-1)+(x-X(i))P(i+1,j)}{X(j)-X(i)}$
  end for
end for
return P(1,m)
```

4.2 Extending Polynomial Interpolation to Higher Dimensions via a Dimension-by-Dimension Approach

Suppose we have an $m \times m$ grid - not necessarily uniform - of data points $\{((x_i, y_j), z_{ij})\}_{i,j=1}^{m}$ corresponding to some underlying function $f : (M \subset \mathbb{R}^2) \to \mathbb{C}$. It is not necessary that the grid be square, but we assume that it is for simplicity. We wish to approximate f at some point (x, y) inside the closed square bounding the grid using polynomial interpolation. We can accomplish this via $m + 1$ one-dimensional interpolations.

We begin by stepping through the m points x_i, and for each x_i, we perform a one-dimensional polynomial interpolation at the point (x_i, y) using the m data points $\{((x_i, y_j), (z_{ij}))\}_{j=1}^{m}$. At each step, we obtain a point $((x_i, y), \hat{f}(x_i, y) = \hat{z}_i)$, where $\hat{f}(x_i, y)$ is the resulting polynomial interpolant. After all m steps, we have a set of points $\{((x_i, y), z_i)\}_{i=1}^{m}$, and we interpolate these points at (x, y) to obtain the desired approximation. A visualization of this process, which we call dimension-by-dimension polynomial interpolation, is given in Figure 12 below for $m = 4$.

In essence, this technique reduces an n-dimensional interpolation problem to a series of $n - 1$-dimensional problems, each of which can then be reduced again, until we are left with $O(m^{n-1})$ one-dimensional problems which we can solve. Thus, a generalization of the two-dimensional technique described above gives rise to a recursive algorithm. Imagine the three-dimensional case, in which the grid of data points is inside some closed $m \times m \times m$ cube. We first consider m planes in the cube, and for each plane, we consider m lines in the plane. We interpolate on each line in the plane, and then once more on the resulting line of approximations, to obtain an approximation in each plane, as described above. Then we solve a final one-dimensional interpolation problem, using each of the plane approximations as data points, to obtain the desired approximation.

Using Neville's Algorithm, the entire method requires $O(m^{n+1})$ operations; we have $O(m^{n-1})$ one-dimensional interpolation problems, each of which re-

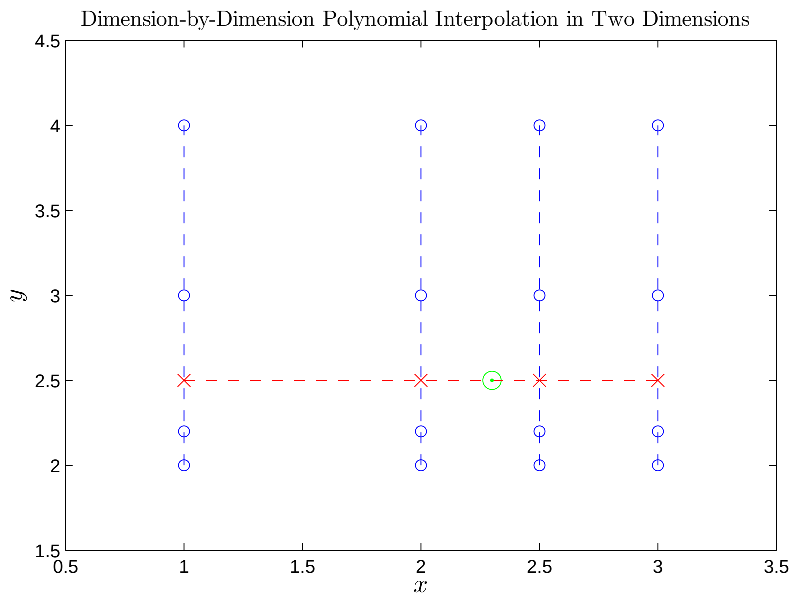
Dimension-by-Dimension Polynomial Interpolation in Two Dimensions
4.5
4
3.5
3
2.5
2
1.5
y
0.5
1
1.5
2
2.5
3
3.5
x

Figure 12: $m = 4$. Original data points are given as blue dots. The first m interpolations are performed along the blue dotted lines to yield data points at the red xs, and the final interpolation is performed along the red dotted line to obtain the desired approximation at the location (x, y) of the green circle.

quires $O(m^2)$ operations to solve. Using local interpolation, m is kept small, so this scaling is not necessarily unmanageable, but may become so for larger n.

The severest limitation of this method, however, is the requirement of a regular grid of data points. Each local interpolation requires such a grid, and since each set of local data points is a subset of the set of global data points, we require the set of global data points to be arranged in a grid. For large n, constructing such a global grid may be infeasible; we may only be able to afford to place nodes at points of high activity, without then filling in the extra nodes required to form a grid. Noting this limitation, we will put this dimension-by-dimension technique to the test in a later section, and find that it is yields positive results in the case $n = 2$. However, we first consider other methods which do not require the data points to be arranged in a grid.

4.3 Interpolation with Radial Basis Functions

A radial function is a function $\phi : \mathbb{R}^n \to \mathbb{R}$ of the form

$$\phi_{x_0}(x) = \psi(\|x - x_0\|) = \psi(r) \tag{28}$$

where $x_0 \in \mathbb{R}^n$ is some fixed center and $\psi : \mathbb{R} \to \mathbb{R}$ [16]. In other words, the value of a radial function depends only on the distance of its argument from the center. A radial basis function interpolant, then, is an interpolant of the form

$$\hat{f}(x) = \sum_{i=1}^{m} c_i \phi_{x_i}(x) = \sum_{i=1}^{m} c_i \psi(\epsilon \|x - x_i\|) \tag{29}$$

Here, $\{x_i\}_{i=1}^m$ is a set of interpolation nodes, ϕ_{x_i} is a radial function with center x_i, and ϵ is some positive constant. As always, $\hat{f}(x)$ must obey the interpolation condition corresponding to the data points $\{(x_i, y_i)\}_{i=1}^m$ for $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{C}$. We obtain the coefficients c_i by solving the symmetric linear system given by

$$\begin{pmatrix} \psi(\epsilon\|x_1 - x_1\|) & \psi(\epsilon\|x_1 - x_2\|) & \cdots & \psi(\epsilon\|x_1 - x_m\|) \\ \psi(\epsilon\|x_2 - x_1\|) & \psi(\epsilon\|x_2 - x_2\|) & \cdots & \psi(\epsilon\|x_2 - x_m\|) \\ \vdots & \vdots & \ddots & \vdots \\ \psi(\epsilon\|x_m - x_1\|) & \psi(\epsilon\|x_m - x_2\|) & \cdots & \psi(\epsilon\|x_m - x_m\|) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{30}$$

Inverting this system to obtain the closed-form interpolant requires $O(m^3)$ operations in general.

A radial basis function interpolant, then, is a sum of m radially symmetric functions which are collocated to the interpolation nodes, such that the sum passes through all data points. At each node, we place some radial basis function, and then take as the interpolant the linear combination of these functions which satisfies the interpolation condition. These interpolants, then, are well suited to underlying functions which exhibit some sort of radial symmetry, either global or local.

We can further expect that the steepness or flatness of the chosen radial basis function will affect the accuracy of the interpolation. Varying this property of the basis functions is the purpose of the constant ϵ in the expansion (29) above. This ϵ is called the shape parameter, and adjusting it properly will turn out to be of the utmost importance in generating accurate radial basis function interpolants.

For certain choices of radial functions, the interpolation matrix above is guaranteed to be non-singular. Notice that in the section on polynomial interpolation, we indirectly proved that the Vandermonde matrix is non-singular by proving the existence of the Lagrange polynomial. However, non-singularity is not in general guaranteed for an arbitrary interpolation basis, so this is an important advantage of the method. There are several commonly used radial functions which guarantee that the interpolation matrix in Equation (30) is non-singular [15]. The multiquadric radial function is given by

$$\psi(r) = \sqrt{1 + (\epsilon r)^2} \tag{31}$$

Notice that since $r \geq 0$, the multiquadric radial function is smooth, and it is equal to 1 at $r = 0$. We also have the inverse multiquadric, given by

$$\psi(r) = \frac{1}{\sqrt{1 + (\epsilon r)^2}} \tag{32}$$

which is also smooth and equal to 1 at $r = 0$. The multiquadric and inverse multiquadric are said to be about equally as effective in applications, even though the multiquadric grows monotonically, whereas the inverse multiquadric decays monotonically. This turns out to be largely unimportant in practice, unless we require that the interpolant decay outside of the rectangle bounded by the data points. A final example is the Gaussian, given by

$$\psi(r) = e^{-(\epsilon r)^2} \tag{33}$$

The Gaussian radial function tends to display a higher sensitivity to changes in the shape parameter ϵ than the multiquadric and inverse multiquadric radial functions, and as a result may not perform as well, though it is well-suited to certain specific applications where the choice of ϵ can be optimized. Note again that each of these radial functions yields a guaranteed non-singular interpolation matrix in (30).

As an example of the method, we interpolate the function $f(x) = e^{-\frac{(x-2)^2}{2}} + e^{-\frac{(x+2)^2}{2}}$ on the interval $[-5, 5]$ using a multiquadric radial basis with $\epsilon = 1$ and 7 equispaced data points. We give a plot of f, the data points, and the interpolant in Figure 13. We also include plots of the individual terms of the sum which forms the interpolant in order to provide intuition about how elements of the interpolation basis might combine to form an accurate interpolant. Since f is smooth and contains radially symmetric structures, we are able to build a decent interpolant with a small set of data points.

Before entering a brief discussion of the sensitivity of the error to the shape parameter ϵ, which will turn out to be a fatal flaw of the radial basis functions for our application, we emphasize its primary advantages. Notice that there is no fundamental difference between the forms of the method for different dimensions n, since the functions in the interpolation basis map all nodes in $\mathbb{R}^n$ onto $\mathbb{R}$ for any n. Since the inversion of the interpolation matrix requires $O(m^3)$ operations in general, the requirement of many more nodes in higher dimensions may cause the computational cost of the problem to scale dramatically. However, the method itself does not require any special generalization to higher dimensions, as polynomial interpolation does. Also notice that the interpolation nodes can be placed anywhere; we do not require them to have any sort of regular structure. This flexibility might allow us to use fewer, intelligently-placed nodes, and is one of the main benefits of interpolation via radial basis functions.

4.3.1 The Shape Parameter ϵ of Radial Basis Functions

Recall the form of the radial basis function interpolant given in Equation (29). The parameter ϵ essentially adjusts the rate of growth of each term in the sum. If we require each term in the sum to reflect localized behavior of the underlying function - for example, if the interpolation nodes are densely packed - then we would choose a large value of ϵ. However, if we require the terms in the sum to have a broader shape - perhaps we believe the form of the radial functions

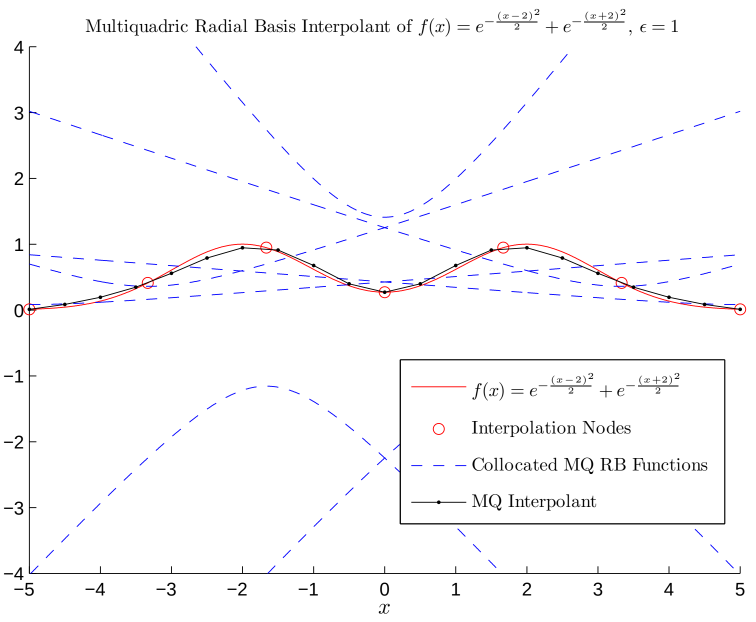

Figure 13: Multiquadric radial basis interpolant of $f(x) = e^{-\frac{(x-2)^2}{2}} + e^{-\frac{(x+2)^2}{2}}$, with $\epsilon = 1$ and 7 equispaced data points.

properly reflects the global behavior of the underlying function - then we would choose a small value of ϵ.

We find that the success or failure of interpolation with radial basis functions is highly dependent on the choice of ϵ; this choice must properly reflect the structure of the underlying function, as well as the choice of interpolation nodes. This property is detrimental to the success of this method in our application. For each waveform we wish to interpolate, we must perform interpolations of many different functions - the projection coefficients over parameter space - each of which has a different, though related, structure. The optimal choice of ϵ for one of these functions in one particular region - assuming that we will use local interpolants; we will discuss this in greater detail later - may be quite different from the optimal choice of ϵ for some other function and some other region. For now it suffices to state that we require a method which is generalizable, and performs well in a variety of contexts.

This begs the question; is there some procedure for choosing ϵ optimally? In [17], it is demonstrated that indeed, using the smooth radial functions listed above, there is some $\epsilon > 0$ for which the error of the radial interpolant is minimized. Various authors have proposed techniques for approximating this optimal ϵ, but no method has been proposed which is efficient, effective, and sufficiently general. One popular method is known as leave-one-out cross validation [18]. We first generate a list of possible choices of ϵ, and fix some ϵ in the list. We then loop through each of the m data points, and for each compute a radial basis interpolant using the $m-1$ other data points. We then compute the error of the interpolant at the "left-out" data point, and store this value in a vector. In the end, we obtain an m vector, and, taking the norm of this vector, obtain a "cost" estimate for the value of ϵ which we have chosen. We then choose the value of ϵ with the lowest associated cost.

If we use this technique to test n choices of ϵ, we incur an $O(nm^4)$ cost. This method is therefore highly inefficient, especially when we are required to solve many different interpolation problems. A proposed improvement to this technique reduces the cost to $O(nm^3)$, but we still only optimize ϵ by testing every possibility in some set of candidates. Thus, this technique is hardly an improvement to tinkering in the context of our application, which requires the interpolation of hundreds of different functions in order to obtain a solution.

Notice that we have, so far, assumed ϵ is fixed for each basis function; that is, that the choice of ϵ is identical for each radial function ϕ_{x_i} in the sum. We could instead choose a different ϵ_i for each i. This would, of course, require some criterion by which to choose each ϵ_i. One author suggests setting the value of ϵ_i to be inversely proportional to the distance of x_i from the nearest node [16]. This way, in regions in which nodes are more densely packed, the corresponding terms of the sum will be steeper and more localized. However, we must choose some proportionality constant, and this choice will affect the interpolation dramatically. We are still faced with an overly-specific method, since now we have no general procedure by which to choose the proportionality constant; the correct choice of such a constant may very well be different for different functions. In light of the absence of a robust and general procedure for optimizing the shape parameter, if the accuracy of the interpolant is overly sensitive to the choice of ϵ, we will likely be forced to reject interpolation via radial basis functions.

In practice, in order to investigate the optimal choice of ϵ for various functions, we simply interpolate the same set of data points over a range of values of ϵ, and take note of which choice of ϵ gives the lowest error. We find that, though the optimal ϵ correlates with the shape of the underlying function - specifically, with the absolute values of its derivatives - it often also depends on other factors. We would expect to see a proportional relationship between the optimal ϵ and the broadness of the underlying function in regions of activity; for example, if we interpolate the functions $\sin(kx)$ for different choices of k, we would expect that the optimal ϵ would increase with k. However, this is not always what we observe; the optimal ϵ appears to depend not only on the shape of the function, but on the resolution of the underlying function by the data points, the placement of those points, and probably other factors.

Furthermore, the range in error over different choices of the shape parameter can be rather large. We consider the relatively well-behaved function $f(x) = x\sin(3x)$. Since $f(x)$ is smooth and somewhat well-resolved by the data points, the sensitivity of the interpolation error to ϵ in this case is typical. Using a multiquadric radial function basis and 30 equispaced data points, we interpolate

f on the interval $[-2\pi, 2\pi]$ using different choices of ϵ; specifically, we choose values $\epsilon \in [\frac{1}{2}, 2]$. We plot the function f with the data points that we use, and the maximum interpolation error over $[-2\pi, 2\pi]$ for each choice of ϵ, in Figure 14. Here, doubling ϵ increases the error by an order of magnitude; for other functions, the slope of this plot may be even steeper.

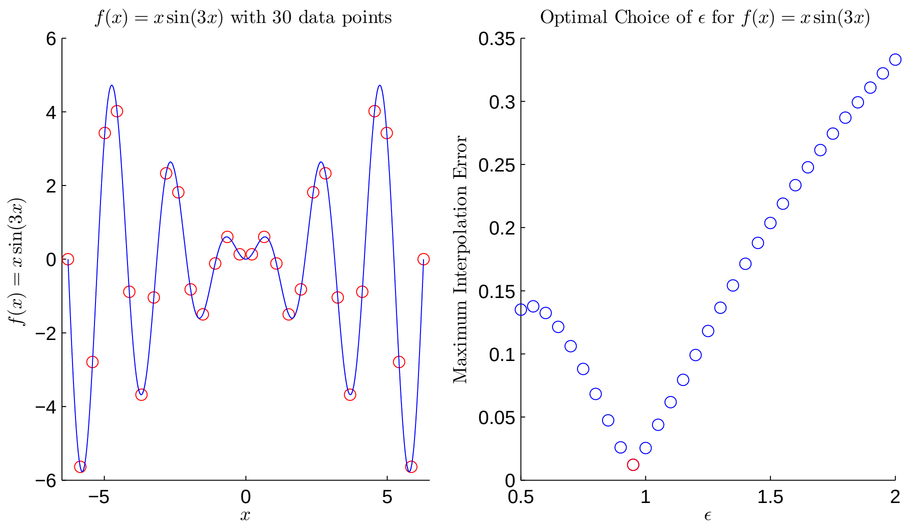

Figure 14: Maximum interpolation error for many choices of $\epsilon \in [\frac{1}{2}, 2]$. The optimal choice of ϵ on this interval is highlighted in red.

For small values of ϵ, the interpolation matrix in (30) can become very ill-conditioned. Intuitively, this is because a smaller value of ϵ yields a flatter radial function, so that the range of values in the entries of the interpolation matrix will be smaller, leading to less linear independence between the rows and columns of the matrix. If we require an interpolant with small ϵ, then, we might need to use other methods, such as singular value decomposition or higher-precision arithmetic, to invert the interpolation matrix. This further increases the computational cost of the interpolation.

A result from [19] shows that in the limit $\epsilon \to 0$, the one-dimensional radial basis function interpolant on some set of data points converges to the corresponding polynomial interpolant. As a result, we see Runge's Phenomenon appear in radial basis function interpolants with small ϵ; this is a further impediment to using radial basis function interpolants with small ϵ. In light of the $\epsilon \to 0$ result, it is natural that using different node distributions, like Chebyshev nodes, effectively addresses this problem if we are to use global interpolants in our application. This phenomenon is explored further in [16].

In light of observation, mentioned above, that in general there exists some optimal choice of ϵ which is greater than 0, this $\epsilon \to 0$ result also implies that the accuracy of radial basis interpolants with an optimal choice of ϵ will generally be better than that of polynomial interpolants. Note that this result is only relevant for one-dimensional problems, although we sometimes notice Runge Phenomenon-like oscillations in higher dimensions.

4.3.2 Global & Local Radial Basis Function Interpolants

Consider the possibility of employing global radial basis function interpolants of the projection coefficients $\alpha_i(\mu)$. A plot of one such function was given in Figure 9. In this particular function, we have a small region of very high oscillation, and then broader oscillations emanating from this region. It may be, then, that different choices of ϵ will better resolve different regions over the parameter space. Furthermore, we must interpolate many different projection coefficients, and since we do not have an automatic procedure to customize our choice of ϵ - and there are potentially hundreds of projection coefficients to interpolate - we must simply fix some ϵ and use it for every interpolation. The sensitivity, then, of radial basis interpolants to the choice of the shape parameter may be incompatible with the variety inherent in such a complex, multi-step interpolation problem considered over the global parameter space.

Most important, however, is that the number of nodes required to resolve the functions, even in one dimension, is likely in the thousands, and in two dimensions, in the tens of thousands. This is exacerbated by the lack of any radial symmetry in the functions of interest; if the structures of the elements in an interpolation basis resemble the underlying function, fewer nodes will be required to resolve that function's behavior. We need only consider Figure 9 to see that a radial basis is not so well-suited to the global problem. Of course, given enough data points, the interpolant will converge to the underlying function, but the marginal error reduction achieved by adding an additional data point will likely level off too soon to obtain a reasonably efficient interpolant. The induced interpolation matrix will be enormous, and inverting it with accuracy will then be a significant challenge. Furthermore, the resulting global closed-form interpolant will take the form of a sum with an unwieldy number of terms.

In short, our problem requires a general method, and a global radial basis interpolant must be tailored to a specific type of underlying function. Instead, we will make use of local interpolants; this strategy eliminates some of the problems associated with global radial basis interpolants which we have pointed out. Since the projection coefficients vary smoothly over parameter space, a small neighborhood of some point $\alpha_i(\mu)$ resembles a hyperplane. In a local neighborhood including only a handful of data points, the underlying function will, in most regions in parameter space, be a surface whose gradient varies mildly. We demonstrate the utility of a local approach by plotting a 9×9 local grid on the front face of the plot in Figure 9, and showing a neighborhood of the grid at three different levels of zoom in Figures 15, 16, and 17 below. These figures should be viewed as a series, beginning with Figure 9, and are intended to provide intuition as to the fundamental difference between local and global interpolation in this context.

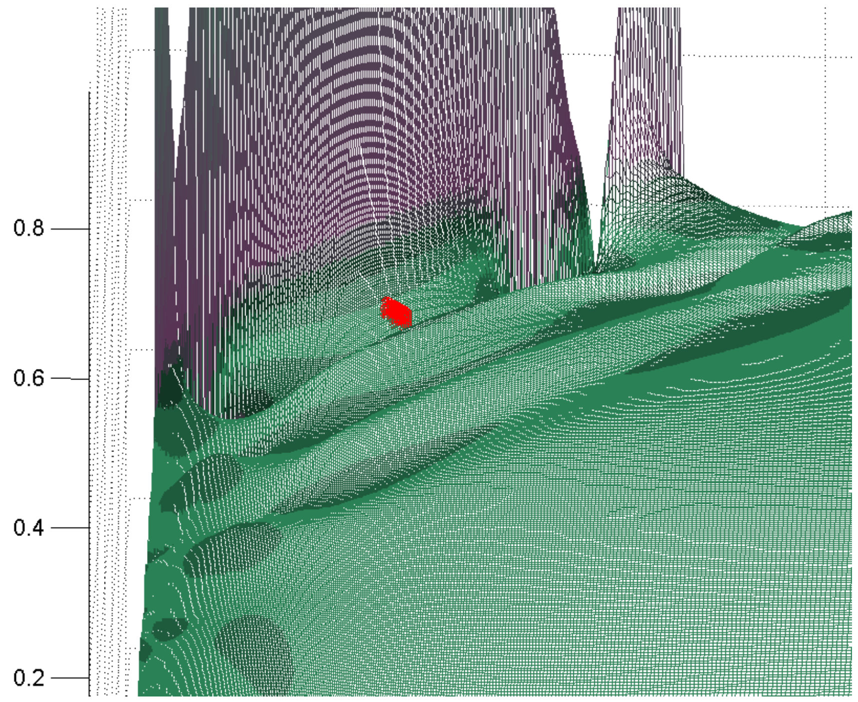

Figure 15: 9×9 local grid, with data points given by red dots, on the plot of the 20^{th} projection coefficient over a two-parameter space. At this first level of zoom, we can not clearly see the points in the local grid, but we may observe the scale of the local grid with respect to the whole parameter space.

This local grid is a subset of a 400×400 global grid which is the Cartesian product of two one-dimensional logarithimically-spaced grids; that is, the nodes in the global grid are clustered towards lower parameter values, so that more nodes are placed in the region of high oscillation. The selection of the global grid will be discussed in a later section. Note that this is a dense grid; the picture might not look as promising if the grid contained fewer nodes, since we would be forced to interpolate on larger neighborhoods of the point of interest, which have less of a resemblance to simple hyperplanes. Furthermore, other regions of the underlying function have more variation in their gradient, or more complex structures.

However, at least for this particular projection coefficient, most regions are locally well-behaved, and other projection coefficients have a similar structure. This is the primary advantage of local interpolation using any method; that the local structure of a smooth function tends to be easier to capture than its global structure, though it is nevertheless sometimes useful in interpolation to maintain information about the global structure of the underlying function. However, there is another related benefit which is more specific to radial basis

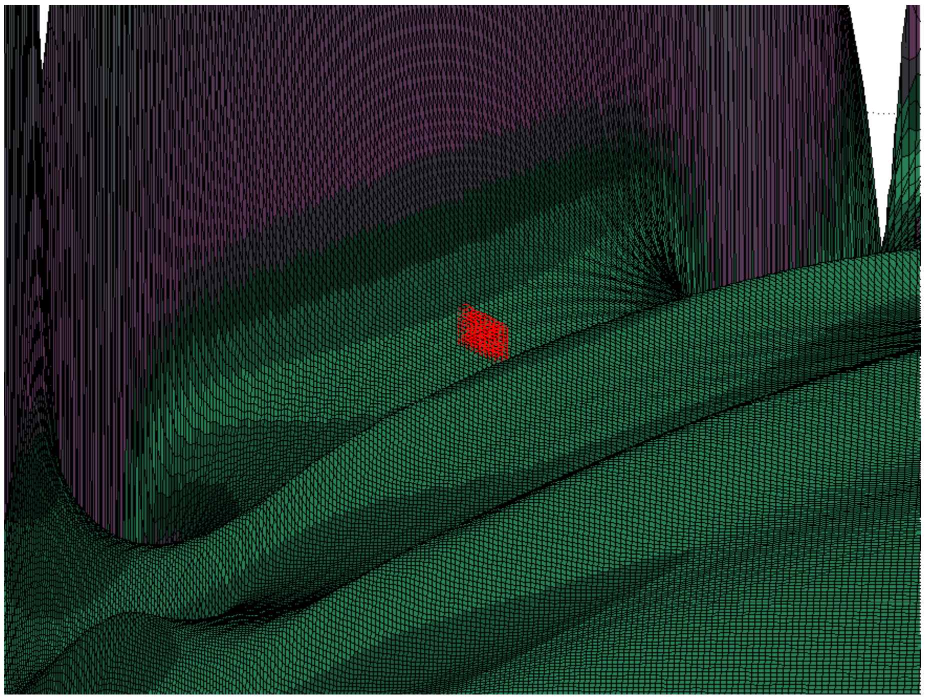

Figure 16: A zoom in of Figure 15 to the local grid. Here, we can observe the local well-behavedness of the underlying function.

interpolants. In our application, attempting to interpolate globally via radial basis functions is awkward, since the underlying functions display very little radial symmetry. On the other hand, a small neighborhood of a point on the graph, in its approximate similarity to a slanted hyperplane, will likely exhibit some sort of radial symmetry. Therefore, a radial basis which fails to represent the underlying function globally may very well do so locally.

Local interpolation with radial basis functions still suffers from the lack of generality caused by the sensitivity of the shape parameter. Up to a rotation, we expect to see similarities between the regions bounded by local grids, as sufficiently small regions resemble hyperplanes. However, the optimization of the shape parameter is related to the underlying function's gradient, in addition to higher derivatives. These quantities will certainly vary from region to region. To elucidate this point, we locally interpolate a Gaussian at a fine grid of points on the interval $[-5, 5]$. We use a global equispaced grid of 100 nodes and perform 99 local interpolations - one between each adjacent pair of points in the global grid. At each point, we generate multiquadric radial basis interpolants for various choices of $\epsilon \in [.05, 1]$, and compute the corresponding interpolation error. In Figure 18, we give plots of a Gaussian with the global data points, in addition to the absolute values of the first two derivatives, and the optimal

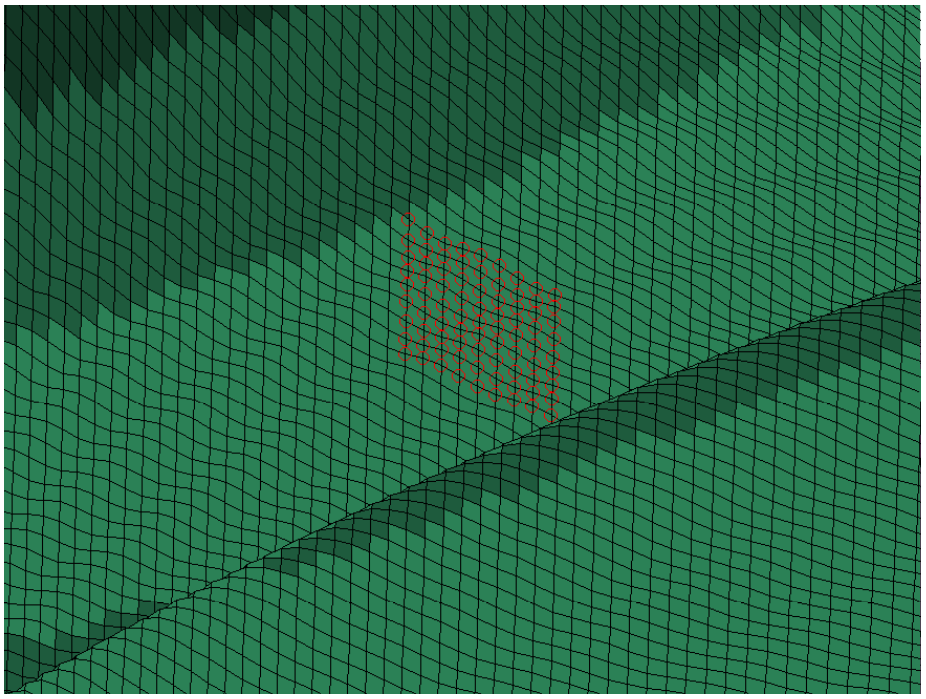

Figure 17: A zoom in of Figure 16. We see that within the bounds of the local grid, the underlying function almost resembles a plane, with mild smooth waves.

choice of ϵ for different local interpolants.

Though there is no simple relationship between the optimal choice of ϵ and the derivatives of the underlying function f, we can see that they are in some way related. In general, steeper regions and regions of greater curvature tend to favor larger choices of ϵ, as expected; when one of these quantities is large and the other is small, the optimal choice of ϵ tends to be somewhere in the middle of the range. It might be worthwhile to use this observation and attempt to construct some procedure by which to select the shape parameter for local interpolants, but the association is likely too vague to exploit. Notice that there are certain unexpected irregularities in the optimal choice of ϵ, which are likely a result of round-off error due to ill-conditioned interpolation matrices for smaller values of ϵ, and could be mitigated by using higher-precision arithmetic. We do not address this issue here because the overall structure of the plot is still clear.

We are left with an interpolation method which is perhaps too specific for our application. However, by using local interpolants, we have some chance, perhaps, of overcoming this specificity. We must also remember that the method works with scattered data points, and is easily generalizable to higher dimensions. Furthermore, since radial basis interpolants converge to polynomial interpolants in the limit $\epsilon \to 0$, and the optimal choice of ϵ usually exists for

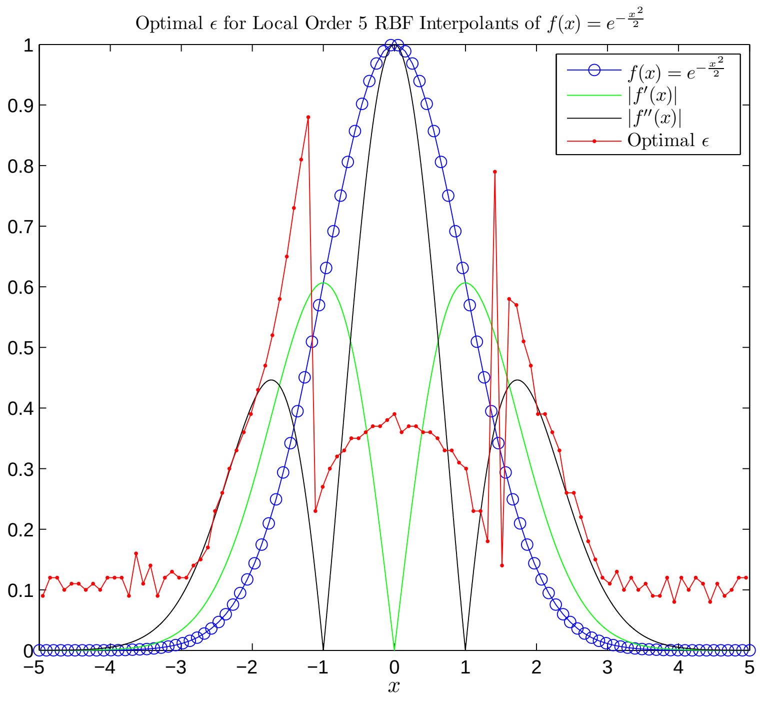

Figure 18: The optimal choice of the shape parameter ϵ for local interpolants is related to the derivatives of the underlying function.

some $\epsilon > 0$, a well-chosen radial basis interpolant will often outperform a polynomial interpolant of the same order. Therefore, it is worthwhile to test the effectiveness of interpolation via radial basis functions on our application.

4.4 Least Orthogonal Interpolation

Theorem 1 establishes the existence of polynomials of degree $m-1$ that interpolate any m data points in one dimension. However, this result does not apply to underlying functions $f : \mathbb{R}^n \to \mathbb{C}$. Given m data points sampled from such a function, what can we say about n-variate polynomial interpolants of those data points?

Let us outline a procedure to generate an n-variate interpolating polynomial of degree at most $m-1$. Choose a one-dimensional subspace such that the projections of the interpolation nodes onto this subspace are distinct. For example, in $n = 2$, if we have data points $((0,0), z_1)$ and $((1,0), z_2)$, we can project onto the x axis, but not onto the y axis. We then obtain data points in this one-dimensional subspace, and write down the Lagrange polynomial interpolant on this subspace. Finally, we extend the resulting polynomial constantly along the

$n-1$-dimensional orthogonal complement of the subspace. For the data points above, the resulting interpolant will be given by $\hat{f}(x,y) = z_1(1-x)+z_2x$; in this case the interpolant is independent of y, but this is only because we were able to project onto the x axis without issue. In general, we would apply a simple coordinate transformation to move the subspace to the x axis, and the resulting interpolant would not then be constant in all but one of the variables.

However, the number of terms required to build this polynomial grows quickly with the number of data points. For example, given 3 data points in a two-dimensional domain, an interpolating polynomial will, in general, be of the form

$$\hat{f}(x,y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 \tag{34}$$

and given 4 data points, we will potentially require a polynomial of the form

$$\hat{f}(x,y) = c_1+c_2x+c_3y+c_4x^2+c_5xy+c_6y^2+c_7x^3+c_8x^2y+c_9xy^2+c_{10}y^3. \tag{35}$$

In general, an n-variate polynomial of degree k has up to $\binom{n+k}{k}$ terms; this expression grows with k like a polynomial of degree n, which is prohibitively fast if we wish to use more than a few data points.

Given m data points, then, we seek a polynomial of minimal degree which interpolates the data points. Such a process will involving choosing which of the $\binom{n+m-1}{m-1}$ possible polynomial terms we require. For example, if the nodes are arranged in a straight line along one particular coordinate axis, the problem is essentially one-dimensional along that coordinate axis, and we will require all of the corresponding single-variable terms to generate an interpolating polynomial. However, depending on the arrangement of the nodes, we may require a polynomial with fewer terms.

In [20], a procedure to construct a polynomial interpolant of minimal degree on an arbitrary grid in an arbitrary number of dimensions is outlined. The procedure uses families of orthogonal polynomials, rather than multivariate monomials as shown above, as elements of the interpolation basis. These polynomial interpolants are not unique, and using different weight functions to generate orthogonal families of polynomials, we may obtain different polynomial interpolants. This allows for greater problem-by-problem variability.

Like radial basis function interpolants, least orthogonal interpolants operate on arbitrary grids in an arbitrary number of dimensions. In principle, we could use this method to compute global interpolants, but these interpolants will suffer from many of the same issues discussed in previous sections. We will implement a local version of the method for the multi-parameter problem.

The theory and strategies of implementation surround this method are complicated, and are not discussed here, but the curious reader can refer to [20] for details.

5 Waveform Interpolation and Results

Now that we have discussed a few relevant interpolation methods, we return to the problem of interpolating waveforms. We begin by restating our general strategy, and then we will compare the effectiveness of different methods, beginning with the one-parameter problem. Recall that we represent all gravitational waveforms in parameter space by a map $h : (M \subset \mathbb{R}^n) \to \mathbb{C}^p$, where p is the size of the grid $F \subset \mathcal{F}$ in the frequency domain on which we sample the waveforms. Fixing a parameter $\mu \in M$, we then obtain a particular gravitational waveform h_μ, which is a p-vector with complex entries.

Given some parameter μ, our aim is to approximate the vector h_μ using local interpolation techniques. We will not, however, approximate h_μ directly, but the projection $P_{W_N}(h_\mu)$ of h_μ onto the reduced basis space. To do so, we will interpolate the N projection coefficients α_i over the parameter space M at the point $\mu \in M$. Our interpolant $\hat{h}$, then, will be of the form

$$\hat{h}_\mu = \sum_{1}^{N} \hat{\alpha}_i(\mu) e_i \tag{36}$$

where $\hat{\alpha}_i(\mu)$ is the value of the interpolant of the i^{th} projection coefficient at μ, and e_i is the i^{th} reduced basis vector. Our data points will be of the form $(\mu_j, \alpha(\mu_j)) \in M \times \mathbb{C}^N$, and we will solve N separate interpolation problems in order to obtain $\hat{h}_\mu$.

We must first choose a global set of data points. To obtain each data point corresponding to the node μ_j, we must take N inner products $\alpha_i(\mu_j) = \langle h_{\mu_j}, e_i \rangle$ and place the result into an N-vector. We do this for a dense set of nodes $\mu_j \in M$. To obtain an order m local interpolant at some point $\mu \in M$, we first choose the m nodes in the global grid, labeled $\{\mu_1, \ldots, \mu_m\}$, which are closest to μ. Then, for each of the N interpolation problems, we have m data points $\{(\mu_j, \alpha_i(\mu_j))\}_{j=1}^m$, and we use one of the interpolation techniques discussed in the previous section to obtain an interpolant $\hat{\alpha}_i : M \to \mathbb{C}$ obeying the condition

$$\hat{\alpha}_i(\mu_j) = \alpha_i(\mu_j) \tag{37}$$

for $1 \leq j \leq m$. Plugging in to Equation (36), we obtain the interpolant $\hat{h}$, and evaluating at μ gives the desired approximation. Notice that we have

$$\hat{h}(\mu_j) = \sum_{i=1}^{N} \hat{\alpha}_i(\mu_j) e_i = \sum_{i=1}^{N} \alpha_i(\mu_j) e_i = P_{W_N}(h_{\mu_j}) \tag{38}$$

for $1 \leq j \leq m$, so, since we consider $P_{W_N}(h)|_M$ as the underlying function in the interpolation problem, the interpolation condition is satisfied.

5.1 The One-Parameter Problem

In this section, we compare only the polynomial and radial basis function interpolation methods, as least orthogonal interpolation reduces to polynomial interpolation in one dimension. As our global set of data points, we will begin by choosing an equispaced grid of nodes, and we will use the results to inform our choice of a better configuration of nodes.

Once we fix the global grid and some order m of our local interpolants, we will sample 100 random parameters $\mathcal{M}$ in the parameter space M. At each sample parameter $\mathcal{M}$, we will compute an interpolant $\hat{h}(\mathcal{M})$ using the strategy described above, and we will also compute the actual projection $P_{W_N}(h_{\mathcal{M}})$ of $h_{\mathcal{M}}$ onto the reduced basis space. As a measure of the success or failure of the interpolation, we consider the mean l^2 error between the interpolant and the underyling function evaluated at $\mathcal{M}$ over the frequency domain F, given by

$$\sqrt{\frac{1}{p}\sum_{i=1}^{p}|P_{W_N}(h_{\mathcal{M}})^i - h_{\mathcal{M}}^i|^2} \tag{39}$$

where $|\cdot|$ denotes the complex modulus, and the upper indices denote the components of the corresponding vectors.

The reduced basis space we use contains 178 elements. It is generated using a logarithmically-spaced training space with 3000 elements, and the maximum error between the projections using this reduced basis space and the actual waveforms in the training space is $O(10^{-6})$. We begin by choosing three different equispaced global grids of interpolation nodes, containing 500, 1000, and 3000 points, respectively. For each choice of global grid, we use local grids containing 5, 9, and 13 points, respectively. We could use larger local grids, and up to some size, this might increase the accuracy of the interpolants. However, for local grids which are too large, we will generally begin to notice Runge's Phenomenon-like oscillations, and the interpolation error will increase. Larger local grids demand a higher computational cost, and since the optimal local grid size is largely problem-dependent, it is not our aim here to optimize this choice; rather, we aim to give the reader some idea of how varying the local grid size effects the interpolation error. We measure the mean l^2 error at each quartile, in addition to the minimum and maximum errors, and the total number of sample waveforms, out of 100, which have interpolation error greater than 10^{-3}.

5.1.1 Preliminary Results for the One-Parameter Problem using Equispaced Global Grids

For each trial, we perform radial basis function interpolation on the projection coefficients with a multiquadric radial function, for many choices of ϵ. We report the lowest error over each choice of ϵ, and compare this to the polynomial interpolation error. We will use these preliminary results to inform any decisions about how to improve the process. In Figures 19, 20, and 21 we compare the two methods for the three different equispaced global grid sizes by plotting the boundaries of the error quartiles for different local grid sizes.

These plots suggest that the performance of the two methods is quite comparable. Though the optimal radial basis function interpolant sometimes outperforms the polynomial interpolant - as the theory suggests it might - their errors tend to fall within the same order of magnitude. These figures seem to show that the error is more a function of the resolution of the underlying function by the grid than of the local interpolation method used. The picture then seems to

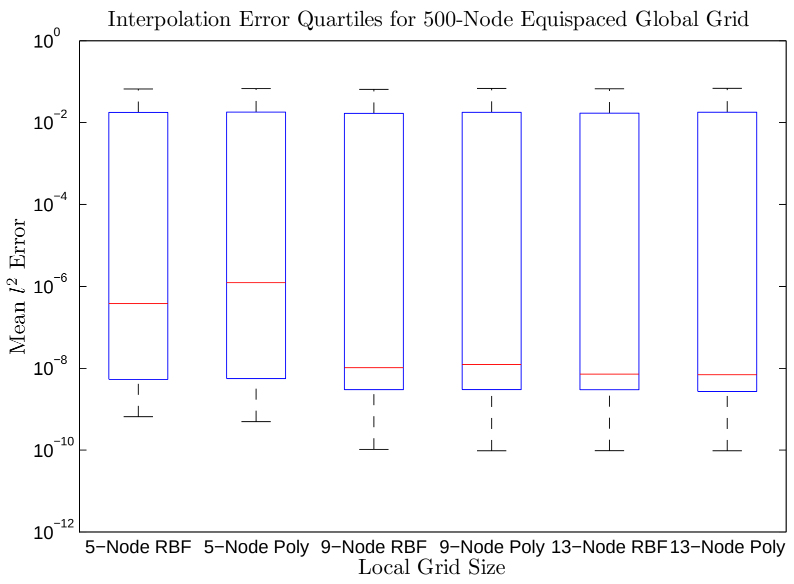

Figure 19: Comparison of l^2 error quartiles of RBF interpolation with optimal ϵ and polynomial interpolation, for a 500-node equispaced global grid and three different local grid sizes. The black lines bound the entire range of errors, the blue box bounds the middle 50% of errors, and the red line is the median error.

favor the simpler and faster polynomial approach, especially if we recall that the plotted error for the radial basis interpolant is, for each interpolated waveform, the lowest error over many choices of ϵ. In a practical setting, we will not know the underyling waveform - this would defeat the purpose of the interpolation - so we cannot choose ϵ optimally. We must then investigate how sensitive the radial basis interpolation error is to changes in the shape parameter; if it is not at all sensitive to ϵ, then the radial basis method is still viable, but if it is, the polynomial method seems more favorable.

5.1.2 Investigation of the Shape Parameter in the One-Parameter Problem

We tested values of ϵ ranging from .4 to 30, with more test values for lower ϵ. To understand the sensitivity of the error to small changes in the shape parameter, we can consider the curvature of the plot of ϵ against the corresponding interpolation error near the optimal value of ϵ. First, however, we check whether or not drastically different choices of ϵ are optimal for different locations in parameter space, and if so, we must determine how well optimal choices of ϵ in one location fare in a location with a different optimal ϵ. We focus on the case of a

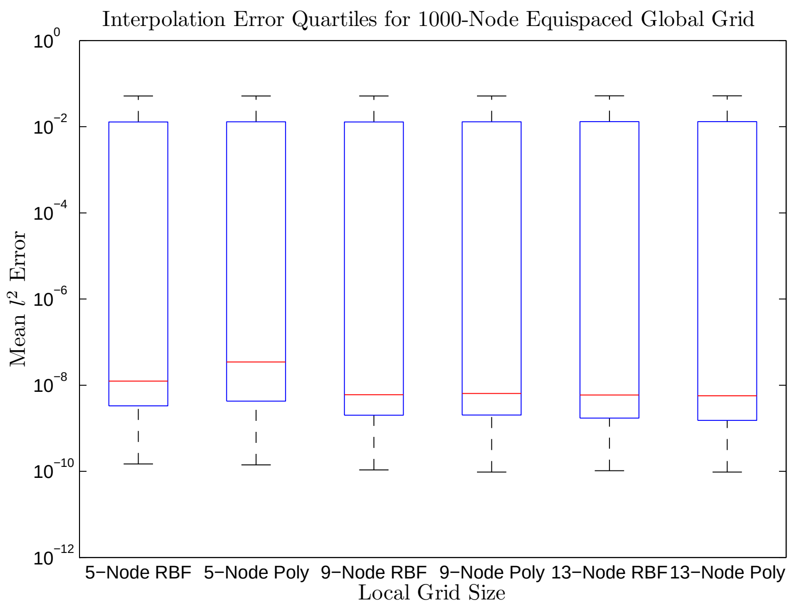

Figure 20: Comparison of l^2 error quartiles of RBF interpolation with optimal ϵ and polynomial interpolation, for a 1000-node equispaced global grid and three different local grid sizes.

3000-node equispaced global grid with 9-node local grids. In Figure 22, we give a histogram of the optimal choices of ϵ over the 100 samples.

We observe that we have a wide range of optimal choices of ϵ for different locations in parameter space, although more of the trials favor smaller ϵ. To obtain a more specific idea of how the optimal shape parameter is chosen, we plot, in Figure 23 the optimal ϵ against the location in parameter space of the waveform for which that ϵ was optimal.

We see that larger ϵ tends to be optimal for lower values of the parameter $\mathcal{M}$, except in the case of the first few choices of $\mathcal{M}$. This will turn out to be a useful observation in optimizing the global grid, and we will discuss it more at length shortly. For now, we simply notice that a wide range of choices of ϵ are optimal for different locations in parameter space. Is there some choice of ϵ that gives small error for most parameters? In Figure 24, we plot the interpolation error as a function of $\mathcal{M}$ and ϵ.

This plot gives us a more complete picture of the behavior of the radial basis function interpolation error corresponding to different locations in parameter space, but it is best viewed as two planar plots. First consider Figure 25. For each choice of $\mathcal{M}$, we see many different errors, which each correspond to some choice of ϵ. This plot shows that given some particular parameter, varying ϵ

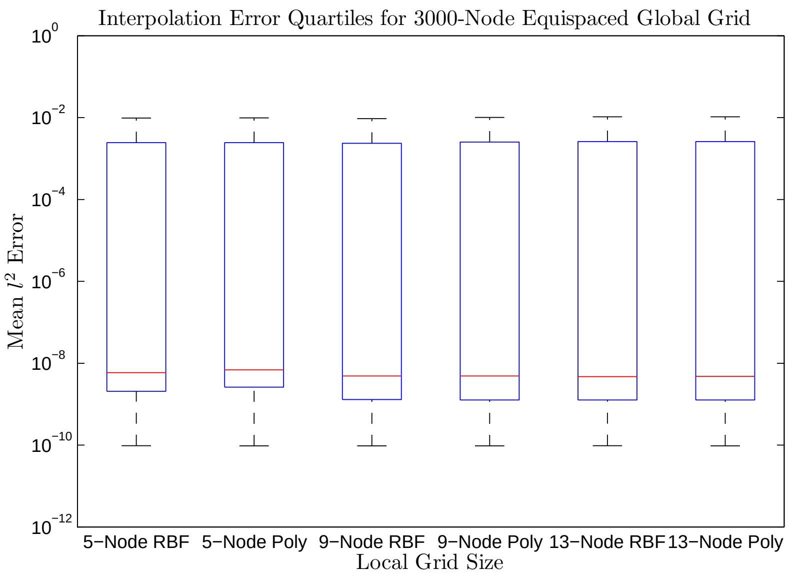

Figure 21: Comparison of l^2 error quartiles of RBF interpolation with optimal ϵ and polynomial interpolation, for a 3000-node equispaced global grid and three different local grid sizes.

and interpolating the waveform corresponding to $\mathcal{M}$ can effect the interpolation error by orders of magnitude.

More important, however, is Figure 26. We would like that for some particular ϵ, the errors corresponding to all of the samples are low. That is, for some ϵ, the top most data points in the corresponding vertical line in the plot in Figure 26 are low. Consider $\epsilon = 25$. Here, all errors, but one, are below 10^{-2}. However, the lowest errors are higher than for smaller ϵ. Thus, we see a trade off; in order to interpolate the tough waveforms better, we must choose an ϵ that results in a poorer interpolation of the easy waveforms. A larger number of the waveforms are better interpolated with smaller ϵ, but for certain waveforms, using small ϵ results in very high error.

For comparison, in this plot, we also give the corresponding data for polynomial interpolation. We see that the polynomial interpolation error is roughly comparable to that of RBF interpolation for $5 < \epsilon < 10$. That is, compared with $\epsilon = 25$, we end up interpolating a few waveforms more poorly in order to decrease the interpolation error by about an order of magnitude for a larger number of waveforms.

Our choice of interpolation method and shape parameter, then, depends on our priorities; for example, do we want a lower median error, or a lower maximum error? However, radial basis interpolants do not seem to beat polynomial

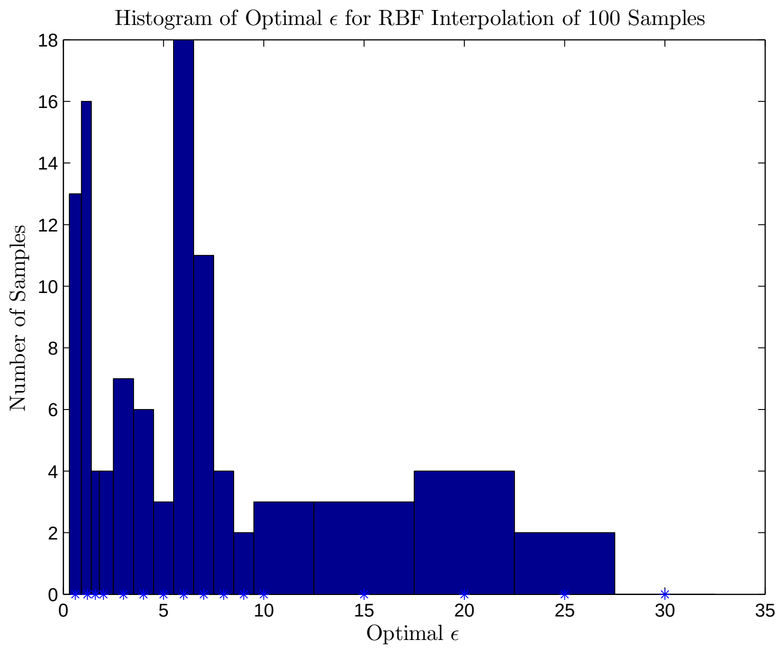

Figure 22: Histogram of optimal ϵ over 100 samples for local RBF interpolation using a 3000-node global grid and 9-node local grids.

interpolants by very much in any case, and the interpolation error is sensitive enough to the choice of shape parameter that there is cause to worry about using radial basis interpolants.

The most effective way to achieve small error for the difficult waveforms, however, is to increase the density of the global grid. In Table 2, we vary the number of nodes in the equispaced global and local grids and report the number of samples with interpolation error larger than 10^{-3}. The numbers for the radial basis interpolants correspond to the best choice of ϵ.

This suggests that we should attempt to adjust the structure of the global grid in order to better resolve locations in parameter space at which interpolation appears difficult. As a starting point, we return to Figure 23, which gives the optimal choice of ϵ for different choices of $\mathcal{M}$ on a 3000-node equispaced global grid with 9-node local grids. We see that larger choices of ϵ are optimal for lower values of $\mathcal{M}$, except for the first few choices of $\mathcal{M}$. From Figure 25, we see that the radial basis interpolation error with the best choice of ϵ tends to be larger for lower values of $\mathcal{M}$; the polynomial interpolation error behaves in the same way.

This first clue suggests that the projection coefficients as functions of the parameter $\mathcal{M}$ have larger gradient, and higher derivatives, for smaller values

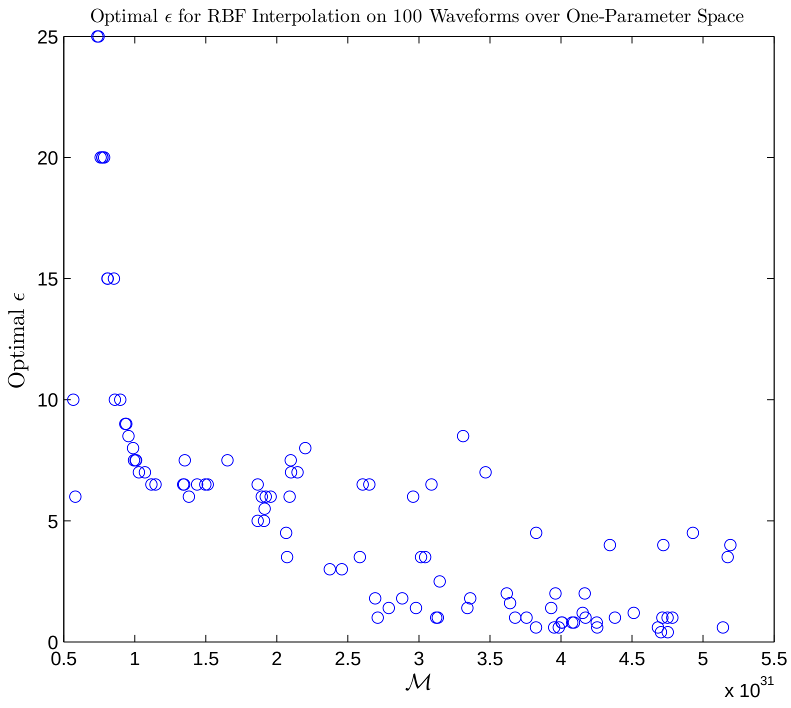

Figure 23: Optimal ϵ over 100 different parameters $\mathcal{M}$ for local RBF interpolation using a 3000-node global grid and 9-node local grids.

of $\mathcal{M}$, except in the case of $\mathcal{M}$ very near the left endpoint of the domain, in which the projection coefficients flatten out. This is reinforced by the second clue, since functions with steep local features are more difficult to interpolate than flat, smoothly varying functions. These hypotheses turn out to be correct; Figures 6 and 7, which show the behavior of typical projection coefficients over one-parameter space, show that we tend to have a spike near some low parameter value, and smaller oscillatory features emanating outward from this spike.

We can attempt to explain this behavior by returning to Figure 4, which shows the values of waveforms for fixed frequency and varying parameter $\mathcal{M}$. The behavior exhibited in this plot is typical of all fixed frequencies, in that we have high oscillation for low parameter values and low oscillation for high parameter values. Recall that the projection coefficients corresponding to some parameter $\mathcal{M}$ are given by inner products of $h_{\mathcal{M}}$ with the elements e_i of the reduced basis. The reduced basis elements are waveforms that are chosen so as to represent the whole space of waveforms accurately, so the reduced basis algorithm will choose for the reduced basis more waveforms from locations in parameter space where there is more activity in the function $h(\mathcal{M})$. In light of Figure 4, it is reasonable to believe that those locations of high activity occur

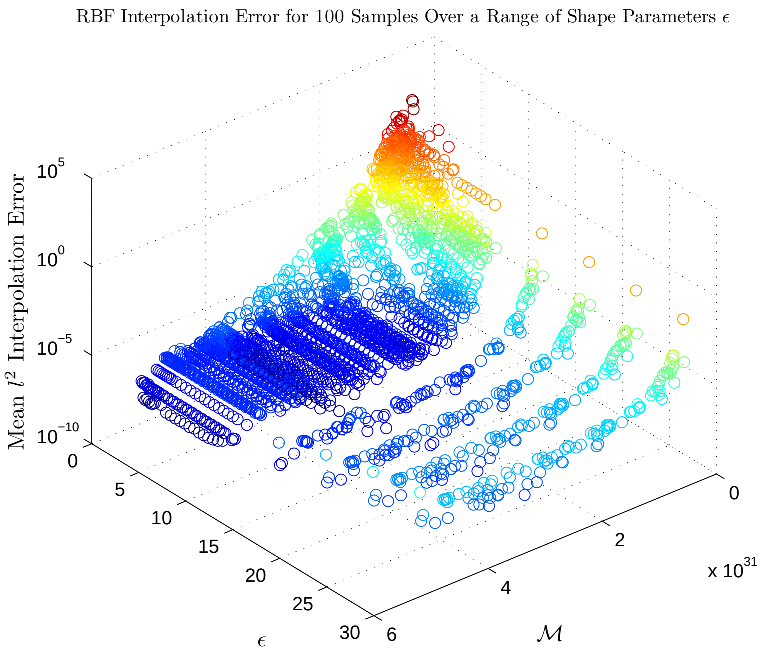

Figure 24: RBF Interpolation Error on a 3000-node global grid and 9-node local grids for 100 samples, plotted as a function of ϵ and μ.

for lower parameters, and indeed, we observe that the reduced basis algorithm chooses more basis elements corresponding to lower parameters. It is to be expected, then, that the projection coefficients will spike near these low parameters; the inner product of $h_{\mathcal{M}}$ with e_i will be larger if $h_{\mathcal{M}}$ is near e_i in parameter space, and many of the waveforms e_i correspond to low parameters. These spikes will be in different places for different projection coefficients, but most tend to occur near lower parameter values.

5.1.3 Adjusting the Global Grid for the One-Parameter Problem

Now that we understand where in parameter space the highest activity of the projection coefficients tends to lie, we can adjust the set of interpolation nodes to take these features into account; specifically, we wish to group nodes near lower parameter values, because the main activity of many projection coefficients occurs in this region. Let a and b denote the minimum and maximum parameter values $\mathcal{M} \in M$, respectively. Consider the function

$$f(x) = a + \frac{10^{\frac{x-a}{b-a}} - 1}{9}(b-a) \tag{40}$$

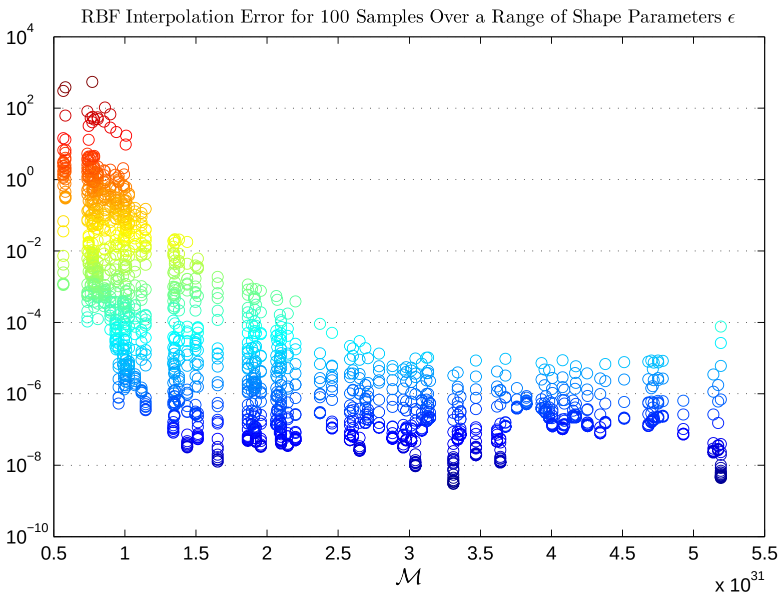

Figure 25: The $\mathcal{M}$-Error cross-section of Figure 24

Notice that f fixes a and b. It also increases monotonically on $[a, b]$ with a positive second derivative. Therefore, if we apply f to an equispaced grid on $[a, b]$, the image will be a grid on $[a, b]$ with points clustered towards a; specifically, the image will be a logarithmically-spaced grid.

We generate logarithmically-spaced global grids of 500, 1000, and 3000 nodes, and interpolate using both methods with 9-node local grids. In Figures 27 and 28, we compare error quartiles for these equispaced and logarithmically spaced global grids, using radial basis function interpolation with optimal ϵ and polynomial interpolation, respectively.

We see the the logarithmically-spaced grids give improvements on the maximum error and the upper quartile. This suggests that using a logarithmically-spaced grid helps to resolve waveforms corresponding to regions in parameter space in which the projection coefficients are badly-behaved; this was the intention of using such a grid. In particular, using this grid gives a large improvement on the 3000-node global grid. We see that 3000 well-placed nodes are sufficient to resolve all of the sample waveforms, but 3000 poorly-placed nodes yields high error on certain waveforms. To reinforce this point, in Table 3 below, we give the number of sample waveforms with polynomial interpolation error larger than 10^{-3} for equispaced global grids and logarithmically-spaced global grids. The analogous numbers for radial basis function interpolation are similar.

It is likely that we can achieve even better error improvements by further adjusting the global grid - for example, by adjusting the base of the logarithmic spacing - but in practice, we may not have the luxury to optimize the grid

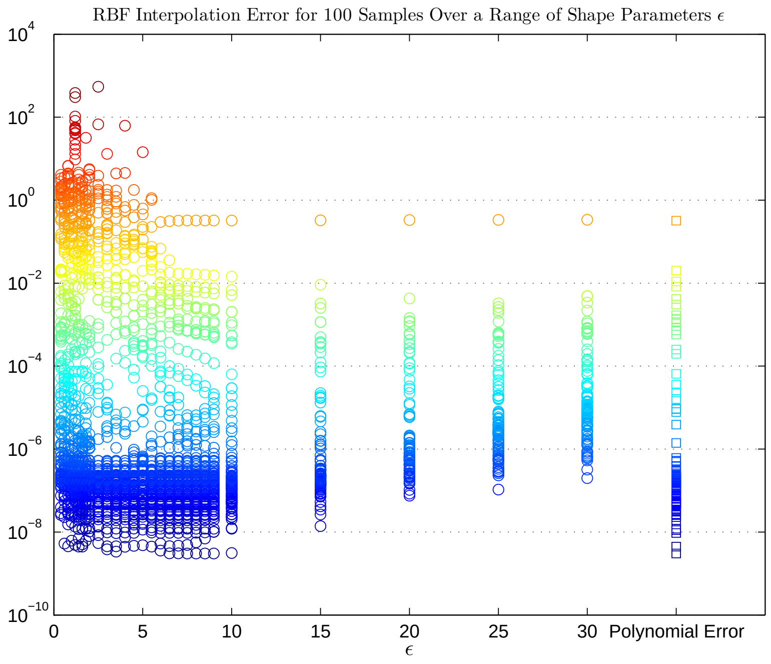

Figure 26: The ϵ-Error cross-section of Figure 24. We supplement this plot with the polynomial error for each sample, given as boxes to the right of the RBF results.

through this sort of tinkering. It suffices to conclude that taking advantage of the underlying structure of the projection coefficients is invaluable to getting the most out of each interpolation node, and that in this one-parameter problem, some grouping of nodes towards lower parameters is more efficient than an equispaced grid of nodes.

In the one-parameter problem, we are able to successfully interpolate a large number of waveforms with small error without an unreasonably large global grid. We find that a logarithmically-spaced global grid is more efficient than an equispaced global grid. We favor the local polynomial interpolation approach, because it is not significantly outperformed by the local radial basis function interpolation approach, it is simpler, it is faster, and it does not require us to adjust any parameters corresponding to different waveforms. By testing various approaches, we have come to understand certain characteristics of the projection coefficients, which are the underyling functions we wish to interpolate. This knowledge will help us to solve the higher-dimensional problem, since these characteristics have higher-dimensional analogues.

Nodes in Global Grid	Nodes in Local Grid	Optimal RBF	Polynomial
500	5	27	30
500	7	25	27
500	9	24	36
1000	5	20	21
1000	7	19	20
1000	9	18	20
3000	5	2	3
3000	7	1	1
3000	9	1	1

Table 2: Number of samples, out of 100, with interpolation error larger than 10^{-3} for equispaced grids.

Nodes in Global Grid	Equispaced Grid	Logspaced Grid
500	27	20
1000	20	3
3000	1	0

Table 3: Number of samples, out of 100, with polynomial interpolation error larger than 10^{-3} for equispaced and logarithmically-spaced global grids. All local grids contain 5 nodes.

5.2 The Two-Parameter Problem

Though the two-parameter problem is qualitatively similar to the one-parameter problem, and we will use many of the same strategies to solve it, certain new challenges are unavoidable. We will require many more data points to interpolate waveforms successfully, as resolving a function of two variables requires many more samples than resolving a function of one variable. The global set of nodes, in particular, will now be very large; the number of local nodes will grow, but each local interpolation is still manageable.

On the other hand, there are more similarities between the two problems than there are differences. Radial basis function interpolation still suffers from the same lack of robustness that it does in one dimension. Though we will not investigate the shape parameter extensively in this section like we did in the last, we have observed that the radial basis function interpolation error is sensitive to the choice of ϵ in the same way. That is, if we are to fix one choice of ϵ for every interpolation, there is a trade off between decreasing the minimum interpolation error and decreasing the maximum interpolation error over a set of sample waveforms. Thus, in this section, we will only consider the interpolation error using radial basis interpolants with optimal ϵ, and we will investigate whether or not, unlike for the one-parameter problem, radial basis interpolants drastically outperform dimension-by-dimension polynomial interpolants. If so, it will be worthwhile to further investigate the shape parameter with respect to

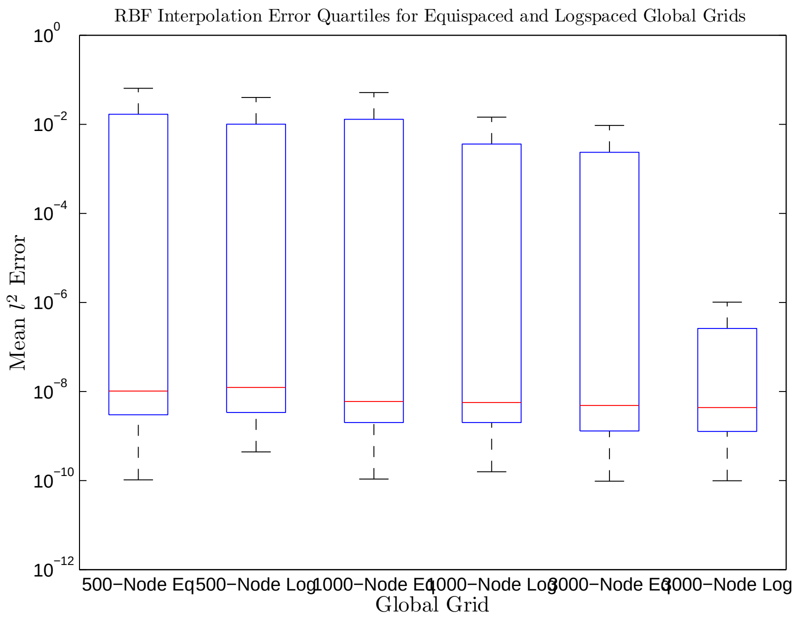

Figure 27: Comparison of mean l^2 error quartiles of RBF interpolation with optimal ϵ for equispaced and logarithmically-spaced global grids. All local grids have 9 nodes.

the two-parameter problem, and if not, we will conclude that the dimension-by-dimension polynomial approach is preferable.

The structures of the one and two-parameter projection coefficients are analogous. This is not surprising, considering the one parameter $\mathcal{M}$ is a function of the two parameters m_1 and m_2. Figures 6 and 9 show typical projection coefficients over one and two-parameter space, respectively. Now, instead of a spike occurring at some, usually low parameter, we have a ridge. We also see smaller oscillations emanating from the ridge, just like in the one-parameter case. For different projection coefficients, the ridge is found in different locations, but it tends to occur when both parameters are small. Therefore, a logarithmically-spaced grid will again outperform an equispaced grid; specifically, we use a Cartesian product of two one-dimensional logarithmically spaced grids. We have observed in practice that this grid choice is preferable, and since the reason is clear, we simply consider logarithmically-spaced grids in this section, and do not present data for equispaced grids.

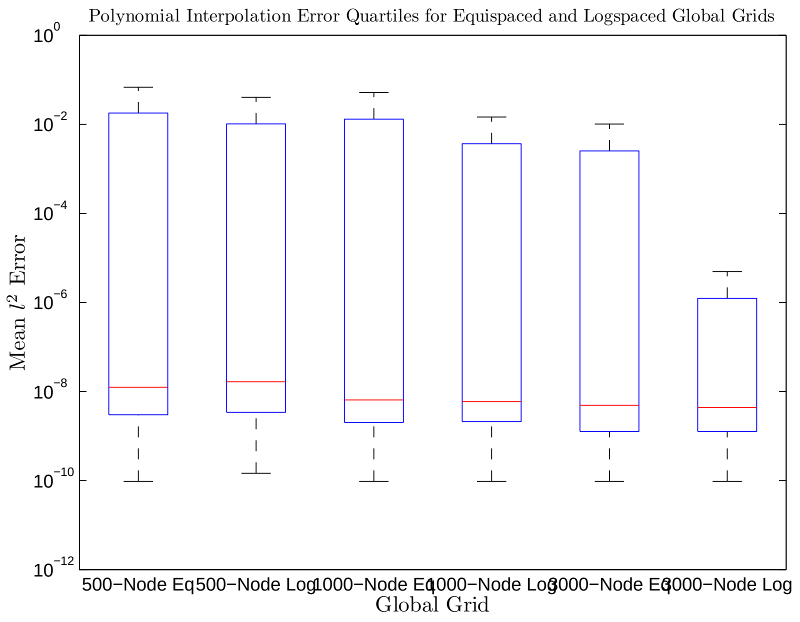

Figure 28: Comparison of mean l^2 error quartiles of polynomial interpolation for equispaced and logarithmically-spaced global grids. All local grids have 9 nodes.

5.2.1 Radial Basis & Dimension-by-Dimension Polynomial Interpolants for the Two-Parameter Problem

We compute interpolation errors for 200 sample waveforms randomly selected over the parameter space. We measure the interpolation error as the mean l^2 error given in 39. The reduced basis space we use is generated using a 200×200 equispaced training space, and is comprised of 161 reduced basis elements. The maximum error tolerance between the reduced basis projections and the actual waveforms over the training space is $O(10^{-6})$. We will first compare local radial basis function interpolation with optimal ϵ with the local dimension-by-dimension polynomial interpolation approach, using 100×100, 250×250 and 400×400 logarithmically-spaced global grids with 5×5 local grids. We again vary ϵ between .4 and 30 in order to select the optimum value. Afterward, we will compare these approaches with least orthogonal interpolation. Figure 29 gives error quartiles for these grid choices.

In Table 4, we give the number of samples with interpolation error greater than 10^{-3} for radial basis interpolants and polynomial interpolants, respectively.

We see that, as in the one-parameter problem, the two methods are comparable. The dimension-by-dimension polynomial technique gives a slightly wider

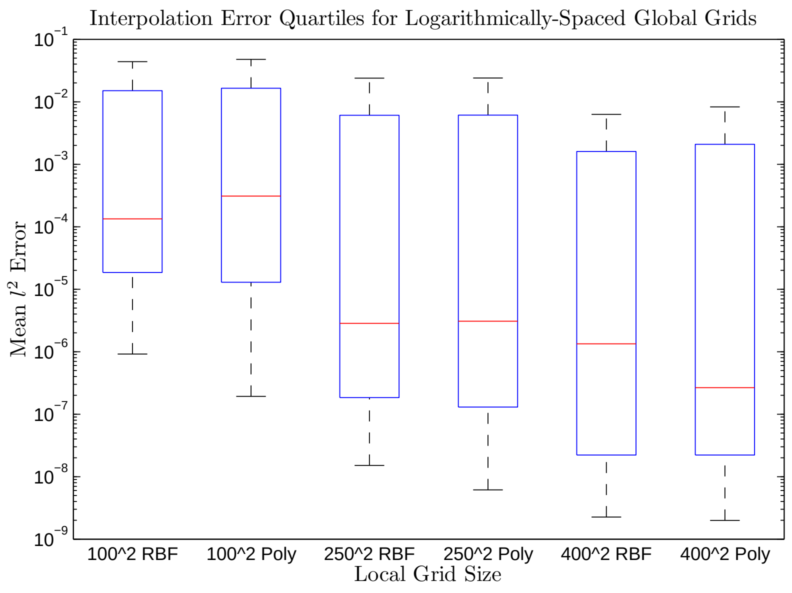

Figure 29: Comparison of mean l^2 error quartiles of RBF and dimension-by-dimension polynomial interpolation for different logarithmically-spaced global grids. All local grids are 5×5.

error range, which is to be expected, since we are viewing the errors of radial basis interpolants with an optimal choice of ϵ. This said, the polynomial technique seems preferable, since it is faster, simpler, and more general. Also notice from Table 4 that for the denser global grids, fewer waveforms were very poorly interpolated by polynomial interpolants than by radial basis interpolants.

In the last section, we showed that logarithmically-spaced global grids outperform equispaced grids, but we did not claim that our choice of global grid was in any way optimal. Indeed, we find that we might benefit further from clustering nodes more densely for lower parameters. In Figure 30, we plot the two-dimensional parameters of the sample waveforms we have chosen, and color-code them by the corresponding dimension-by-dimension polynomial interpolation error. The analogous plot for the radial basis function interpolation error is similar.

Even for the logarithmically-spaced grid generated using the transformation (40), the waveforms corresponding to smaller parameters are less well-resolved. This suggests that we could achieve better results for these waveforms by clustering more points towards lower parameters. This would involve adjusting the transformation (40); we could, for example, change the base of the exponentiation. The goal would be to, unlike in Figure 30, achieve a more even distribution of interpolation errors over parameter space. Note that in exchange for lower-

Global Grid Size	RBF Interpolant	Polynomial Interpolant
100×100	66	75
250×250	25	20
400×400	13	7

Table 4: Number of samples, out of 200, with interpolation error larger than 10^{-3} for radial basis and dimension-by-dimension polynomial interpolants, respectively.

ing the interpolation errors of the lower-parameter waveforms, we would likely increase the errors of the higher-parameter waveforms. We do not investigate more optimal global grids here, as finding one would simply involve problem-specific tinkering, but it might be useful to attempt to formulate a technique by which to choose a more optimal global grid automatically by exploiting the structure of the projection coefficients.

5.2.2 Comparison of the Least Orthogonal Interpolation Methods with the Other Methods

We next proceed to the least orthogonal interpolation method. Least orthogonal interpolants share some of the primary benefits of radial basis interpolants; they do not require structured grids of interpolation nodes, and they are easily generalizable to higher dimensions. Least orthogonal interpolants are multivariable polynomials of minimal degree which satisfy the interpolation condition, so these interpolants do not involve any sort of shape parameter, which could be an improvement over the sensitivity of the radial basis interpolants. The only freedom we have in least orthogonal interpolation is in choosing a weight function; each weight function corresponds to a different interpolation basis of orthogonal polynomials.

We find, using simple test functions, that performing least orthogonal interpolation with structured grids of interpolation nodes, rather than scattered grids, tends to increase the interpolation error. The process of finding an interpolating polynomial of minimal degree relies heavily on the structure of the set of interpolation nodes; changing the arrangement of nodes affects the number of terms required to build a polynomial interpolant. For example, if all m nodes lie on some coordinate axis, say the x axis, the problem essentially reduces to one-dimensional polynomial interpolation, and the interpolant will require all of the m terms $c_0, c_1x, c_2x^2, \ldots, c_{m-1}x^{m-1}$. Such a rigidly-structured polynomial interpolant may not well represent the rest of the underlying function. For example, in the above case, certain weight functions could give an interpolant which is simply the constant extension of the single-variable polynomial along the other coordinates.

The accuracy of the least orthogonal interpolant, then, is related to the structure of the interpolation nodes, even if a structured and unstructured grid resolve the underlying function equally well. We find that a random sampling of

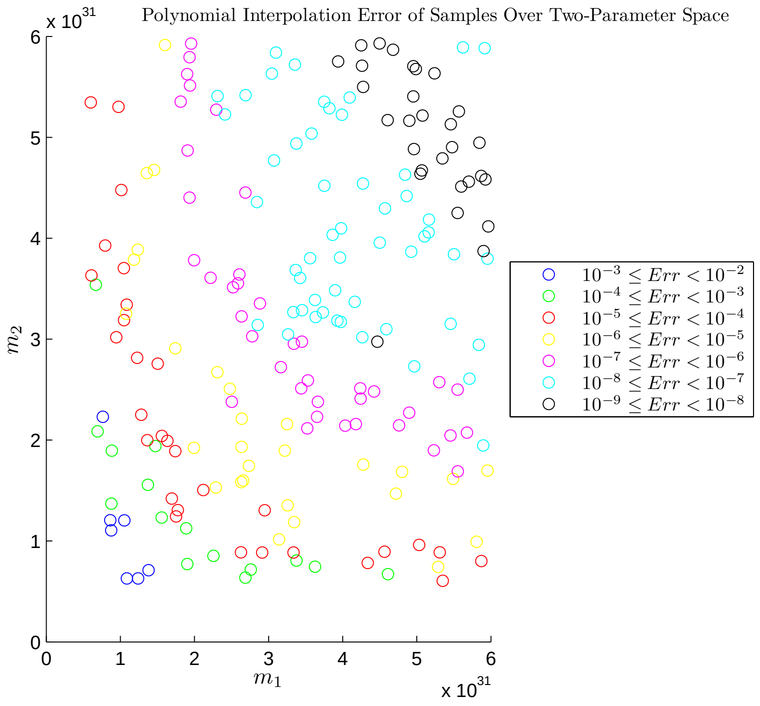

Figure 30: Polynomial interpolation errors of sample waveforms over two-parameter space, using a 400×400 logarithmically-spaced global grid and 5×5 local grids.

nodes yields a more accurate least orthogonal interpolant than any sort of structured grid. Thus, to obtain the global set of nodes, we randomly sample points from a two-dimensional uniform distribution over parameter space, and apply the transformation (40) to the components of these points in order to obtain an unstructured set of nodes which are clustered towards lower parameters.

We test the method using global sets of 100^2, 250^2, and 400^2 nodes generated in this way. We generate local least orthogonal interpolants using the 25 nodes closest to the point at which we wish to interpolate; this corresponds to the 5×5 local grids used to test the radial basis function and polynomial interpolation methods. For each choice of nodes, we interpolate using two different weight functions, corresponding to the Legendre and Hermite polynomial bases.

In Figure 31, we give the least orthogonal interpolation error quartiles for both choices of polynomial basis.

The Hermite polynomial basis slightly outperforms the Legendre polynomial basis. We have also observed that, as expected, the interpolation error is larger for lower parameter values as in Figure 30, so adjusting the global grid could improve the results of least orthogonal interpolation as well. In Figure 32, we

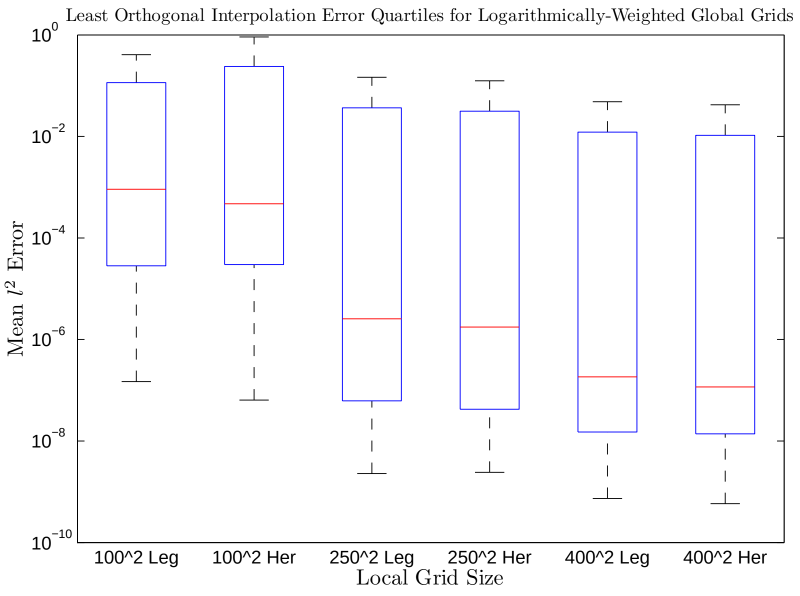

Figure 31: Comparison of mean l^2 error quartiles of least orthogonal interpolants using Legendre and Hermite polynomial bases, respectively. Three different logarithmically-weighted global sets of nodes are tested. All local grids are 5×5.

compare the error quartiles for the least orthogonal method with a Hermite polynomial basis and the dimension-by-dimension polynomial method. Since the performance of polynomial interpolation is comparable to that of radial basis function interpolation, we do not include the error results of the latter in the figure.

Least orthogonal interpolation gives a wider range of errors than dimension-by-dimension polynomial interpolation, suggesting that the former is more sensitive to the location of the waveform in parameter space. That is, least orthogonal interpolation seems to perform better with well-behaved waveforms, but dimension-by-dimension polynomial interpolation seems to perform better with badly-behaved waveforms. To reinforce this point, in Table 5, we give the number of samples with interpolation error greater than 10^{-3} for both methods.

It may be worthwhile to experiment with different weight functions for the least orthogonal interpolation method, since we have only explored two choices here. However, if the application allows for a structured grid of data points, dimension-by-dimension polynomial interpolation appears to provide a fast, simple, and effective solution. The lower minimum errors of the least orthogonal method will not, most likely, be significant in applications, since there are other

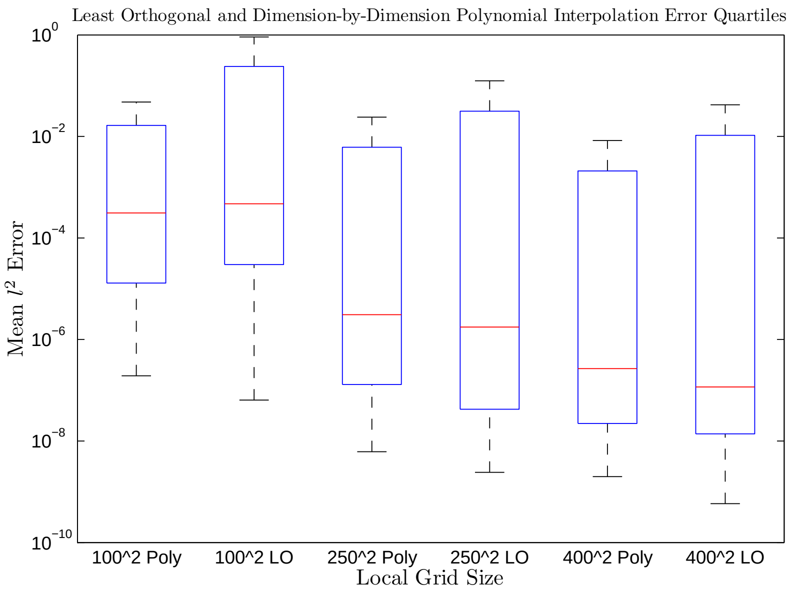

Figure 32: Comparison of mean l^2 error quartiles of dimension-by-dimension polynomial and least orthogonal interpolants with Hermite polynomial basis. Three different logarithmically-weighted global sets of nodes are tested. All local grids are 5×5.

sources of error arising in the process of waveform approximation which may be of a higher order of magnitude than this minimum; in general, we are primarily concerned with interpolating as many waveforms as possible with reasonable accuracy.

Nevertheless, the radial basis function and least orthogonal interpolation techniques should not be discounted. In tackling the higher-dimensional problem, we will likely be forced to interpolate on scattered nodes, so the dimension-by-dimension polynomial approach will no longer be applicable. Both techniques, while more complicated and slower, provide reasonably accurate results, and could potentially be tailored to better solve the problem. In particular, it is possible that a proper choice of weight function could improve the performance of the least orthogonal interpolation method, and in light of the sensitivity of radial basis interpolants to the shape parameter, this might be worth exploring.

Global Nodes	Polynomial Interpolant	Least Orthogonal Interpolant
100^2	75	91
250^2	20	36
400^2	7	19

Table 5: Number of samples, out of 200, with interpolation error larger than 10^{-3} for dimension-by-dimension polynomial and least orthogonal interpolants, respectively.

5.3 Overview of Results & the Higher-Dimensional Problem

We have extensively investigated the one and two-parameter problems, but in these low-dimensional cases, the curse of dimensionality has not quite kicked in. Accurate models of gravitational waveforms could include ten parameters, and storing dense global grids over this sort of space is infeasible. Even in two dimensions, the number of nodes required to resolve the projection coefficients over the whole parameter space is burdensome.

Therefore, constructing an efficient global set of interpolation nodes may be of the utmost importance. In particular, it would be useful to formulate a technique by which to capture the structure of the projection coefficients, and place interpolation nodes in locations where many of the projection coefficients show activity. The improvement in accuracy achieved by using logarithmically-weighted nodes rather than uniformly-spaced nodes is dramatic, and further investigation into an optimal global grid could yield equally dramatic improvements. The difficulty is in automating this process; if we wish to design techniques which are broadly applicable, we must be careful to rely on the information of only those structures which are common to all relevant applications. However, the structure of the waveforms over parameter space seems to determine the structure of the projection coefficients, so it is not unreasonable to imagine that certain fundamental characteristics of the waveforms could be exploited to efficiently place global interpolation nodes.

In higher dimensions, it will likely be impossible to store any sort of regular grid. Therefore, choosing the global nodes efficiently will likely come down to choosing the right probability distribution from which to sample the nodes, unless some iterative or empirical approach is utilized. Furthermore, it will be necessary to use interpolation methods which are compatible with scattered nodes, so although the dimension-by-dimension approach is straightforward and effective in low dimensions, it will no longer be of use when the curse of dimensionality becomes significant.

Unfortunately, there is no hope of constructing global interpolants of the projection coefficients, as their structure is far too fine and irregular to be captured by any reasonable interpolation basis. Therefore, we are restricted to local interpolation methods. On this scale, the underlying function, being smooth, tends to be quite simple. The success or failure of the interpolation, then, depends more on the resolution of the underlying function by the interpolation nodes than on finding an interpolation method which is particularly well-suited to the underlying function. This is why the methods we used tended to give comparable error results, and simple polynomial interpolation was sufficient.

When constructing global interpolants, it is sometimes possible to capture the structure of the underlying function using few nodes and a well-chosen interpolation basis. Here, it seems that the general features of a method, like speed and the ability to operate on scattered nodes, are more important than the structure of the interpolation basis on which the method is built. This suggests that the most significant improvements to the interpolation error may result from adjustments to the arrangement of the interpolation nodes, rather than from finding a better interpolation method.

6 A Brief Conclusion and Discussion of Further Work

In this paper, we have discussed the problem of approximating unknown gravitational waveforms given some set of known waveforms. Successful approximations would be useful for applications in gravitational waveform generation, signal detection, and parameter estimation. To make these approximations, we attempted to combine the reduced basis method of representing the continuum of waveforms by a low-dimensional linear space with interpolation techniques. In doing so, we attempted to interpolate a complicated and subtle function $h : \mathbb{R}^n \times \mathbb{R} \to \mathbb{C}^p$ with p large by solving much fewer than p interpolation problems. We presented introductions to several relevant interpolation techniques - namely polynomial interpolation in one dimension, dimension-by-dimension polynomial interpolation, radial basis function interpolation, and least orthogonal interpolation - and discussed the strengths and weakness of each method. Afterward, we tested these techniques on our application.

Our findings were consistent with the theory which we presented. Optimizing the interpolation grid is of the utmost importance in obtaining accurate results; when the underlying function is complicated, resolving it with interpolation nodes is the first priority. Global interpolants were not feasible, so we opted to use local interpolants. However, it is difficult to tailor a local interpolant to a specific underlying function, so we found that most interpolation methods gave comparable results.

More research should be done, then, into optimizing the global grid of interpolation nodes. It is not unrealistic to imagine that there are some aspects of the structure of the projection coefficients, taking their root in the physics of gravitational waveforms, which can be exploited to determine where the primary activity in these functions lies. If this structure were well-understood, it could lead to the construction of global interpolation grids which could be used to fill in the gaps of many different types of gravitational waveform models. It also may be worthwhile to investigate algorithmic methods of interpolation grid selection, though many of these techniques seem too slow for practical use.

This paper only explored the one and two-parameter problems. It is important to test the methods described here on the higher-dimensional problem. Though we imagine that many of the structures we have found will generalize in obvious ways to higher dimensions, this is not necessarily the case. Furthermore, it is necessary to explore methods of probabilistic node selection in spaces whose dimension is too large to support any sort of regular arrangement of nodes.

There are certainly other strategies that could be used to attempt to interpolate gravitational waveforms. Some of these strategies may use the reduced basis method, and some may not; for example, one could attempt to interpolate the waveforms directly over parameter space. It would be worthwhile to explore entirely different paradigms which could perhaps be applied to more general problems, or which take better advantage of the underlying structure of gravitational waveforms as they are described by Einstein's equations.

An accurate and efficient solution to this problem could help to enable detectors to find waveforms, and discover a great deal of information about their sources. These sources are diverse, scattered through space and time, with countless unsolved mysteries surrounding them. A more complete knowledge of the structure of gravitational waveforms could lead to novel ways of viewing the universe, and of understanding Einstein's confounding equations. Through the study of simplifications and approximations, like interpolants, and the effort to create more faithful approximations, we inch closer to a true understanding of these fundamental laws.

Acknowledgments

I thank Professor Jan S. Hesthaven of Brown University's Division of Applied Mathematics for being an encouraging and patient teacher, and a knowledgeable and inspired research advisor; Scott Field, his former student, for his creativity and spirit; my professors in Brown University's Departments of Mathematics and Applied Mathematics, for their vast knowledge and their eagerness to impart it to others; and my undergraduate peers, in every discipline, for their many passions for their various fields, and for the refreshing insights which those passions produce.

7 References

[1] J. Centrella, AIP Conf. Proc. **1381**, 98 (2011) [arXiv:1109.3492 [astro-ph.HE]].

[2] B. Abbott et al. (The LIGO Scientific), LIGO: The Laser Interferometer Gravitational-Wave Observatory (2007), 0711.3041.

[3] F. Acernese et al., The Virgo status, Class. Quant. Grav. 23, S635 (2006).

[4] P. Amaro-Seoane, S. Aoudia, S. Babak, P. Binetruy, E. Berti, A. Bohe, C. Caprini and M. Colpi *et al.*, arXiv:1201.3621 [astro-ph.CO].

[5] "Press Release: The 1993 Nobel Prize in Physics". Nobelprize.org. 23 Apr 2012. `http://www.nobelprize.org/nobel_prizes/physics/laureates/1993/press.html`

[6] Harry, Gregory M. "Advanced LIGO: The next Generation of Gravitational Wave Detectors." Classical and Quantum Gravity 27.8 (2010): 084006. Print.

[7] "Diagram of LIGO Detector." LIGO. Web. 23 Apr. 2012. `http://www.ligo.caltech.edu/LIGO_web/PR/scripts/facts.html`.

[8] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown and J. D. E. Creighton, gr-qc/0509116.

[9] A. Buonanno, B. Iyer, E. Ochsner, Y. Pan and B. S. Sathyaprakash, Phys. Rev. D **80**, 084043 (2009) [arXiv:0907.0700 [gr-qc]].

[10] S. E. Field, C. R. Galley, F. Herrmann, J. S. Hesthaven, E. Ochsner and M. Tiglio, Phys. Rev. Lett. **106**, 221102 (2011) [arXiv:1101.3765 [gr-qc]].

[11] K. Martel and E. Poisson, Phys. Rev. D **71**, 104003 (2005) [gr-qc/0502028].

[12] A. Buonanno, Y. Pan, J. G. Baker, J. Centrella, B. J. Kelly, S. T. McWilliams and J. R. van Meter, Phys. Rev. D **76**, 104049 (2007) [arXiv:0706.3732 [gr-qc]].

[13] Davis, Lisa G. "Polynomial Interpolation and Error Analysis." Numerical Solution of Differential Equations Course Page. Montana State University, 2007. Web. 2012. `http://www.math.montana.edu/~davis/Classes/MA442/Sp07/Notes/InterpError.pdf`

[14] Eisinberg, A., and G. Fedele. "On the Inversion of the Vandermonde Matrix." Applied Mathematics and Computation 174.2 (2006): 1384-397. Print.

[15] Press, William H. Numerical Recipes: The Art of Scientific Computing. Cambridge [Cambridgeshire: Cambridge UP, 1986. 118-19, 141-142. Print.

[16] Zuev, Julia M. "Recent Advances in Numerical PDEs." Diss. University of Colorado, 2007. Print.

[17] Huang, C.-S., H.-D. Yen, and A.H.-D. Cheng. "On the Increasingly Flat Radial Basis Function and Optimal Shape Parameter for the Solution of Elliptic PDEs." Engineering Analysis with Boundary Elements 34.9 (2010): 802-09. Print.

[18] Fasshauer, Gregory E., and Jack G. Zhang. “On Choosing “optimal” Shape Parameters for RBF Approximation.” Numerical Algorithms 45.1-4 (2007): 345-68. Print.

[19] Driscoll, T., and B. Fornberg. “Interpolation in the Limit of Increasingly Flat Radial Basis Functions.” Computers & Mathematics with Applications 43.3-5 (2002): 413-22. Print.

[20] Narayan, Akil, and Dongbin Xiu. Stochastic Collocation Methods on Unstructured Grids in High Dimensions via Interpolation. Department of Mathematics, Purdue University. Web. 2012.