Inverting Non-Linear Dimensionality Reduction with Scale-Free Radial Basis Interpolation

N.D. Monnig[a,*], B. Fornberg[a], F.G. Meyer[b]

[a] *Department of Applied Mathematics, UCB 526, University of Colorado at Boulder, Boulder, CO 80309*
[b] *Department of Electrical Engineering, UCB 425, University of Colorado at Boulder, Boulder, CO 80309*

Abstract

A numerical method is proposed to approximate the inverse of a general bi-Lipschitz nonlinear dimensionality reduction mapping, where the forward and consequently the inverse mappings are only explicitly defined on a discrete dataset. A radial basis function (RBF) interpolant is used to independently interpolate each component of the high-dimensional representation of the data as a function of its low-dimensional representation. The scale-free cubic RBF kernel is shown to perform better than the Gaussian kernel, as it does not require the difficult-to-choose scale parameter as an input, and does not suffer from ill-conditioning. The proposed numerical inverse is shown to be mathematically similar to the eigenvector interpolation known as the Nyström method, a commonly used numerical method for rapid approximation of eigenvectors of a dense weight matrix. Based on this observation, a critique of the Nyström method is provided, with suggestions for improvement.

Keywords: nonlinear dimensionality reduction, graph Laplacian, radial basis function, interpolation, Nyström method

1. Introduction

Generation of low dimensional representations of data in high dimension has recently become an area of intense research. Many nonlinear methods are available such as kernel Principle Component Analysis [1], Laplacian Eigenmaps [2], and Local Linear Embedding [3], among many others. A significant limitation of many nonlinear dimensionality reduction methods is that they are only defined on a discrete set of data. As a result, the inverse mapping is also only defined on the data. There are well known strategies to extend the forward map to a new point—for example the Nyström extension. However, the problem of extending the inverse map has received little attention so far (but see [4, 5, 6]).

*Corresponding author
Email address: E-mail: nathan.monnig@colorado.edu (N.D. Monnig)
URL: http://amath.colorado.edu/student/monnign/ (N.D. Monnig)

We present a method to numerically invert a general smooth bi-Lipschitz nonlinear dimensionality reduction algorithm such as Laplacian Eigenmaps over all points in the image of the forward map. The method relies on interpolation via radial basis functions in high dimension, modeling a smooth manifold in high dimension as a function of the data embedded in a lower dimensional space. A scale-free alternative to the commonly used Gaussian radial basis function kernel is demonstrated to provide computationally stable results.

The proposed numerical inverse is shown to be mathematically similar to the eigenvector interpolation of the Nyström method, a commonly used numerical method for rapid approximation of eigenvectors of a dense weight matrix. Based on this observation, a critique of the Nyström method is provided, with suggestions for improvement.

There are two notable contributions of this paper. The first is the introduction of a numerically-stable scale-free approach to the problem of interpolation of data in high-dimension as a function of its low-dimensional representation. The second is the unambiguous interpretation of the Nyström method as a radial basis function interpolant, and the subsequent observations about the stability of the method.

2. Inverse Mapping

The problem is posed as follows: $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \subset \mathbb{R}^D$, lie on a bounded low-dimensional smooth manifold $\mathcal{M} \subset \mathbb{R}^D$. The data are embedded in $\mathbb{R}^d$ via a nonlinear mapping

$$\boldsymbol{\Phi}_n : \mathbb{R}^D \to \mathbb{R}^d, \boldsymbol{x}^{(i)} \mapsto \boldsymbol{\Phi}_n(\boldsymbol{x}^{(i)}) \quad \text{for} \quad i = 1, \ldots, n. \tag{1}$$

We assume the existence of an underlying continuous operator, $\boldsymbol{\Phi} : \mathcal{M} \to \boldsymbol{\Phi}(\mathcal{M})$, that provides an extension to $\boldsymbol{\Phi}_n$. In the specific case of Laplacian Eigenmaps, for example, the Graph Laplacian converges pointwise to the Laplace Beltrami operator on the manifold, and one can show the convergence of the associated eigenvectors: $\lim_{n\to\infty} \boldsymbol{\Phi}_n(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{M}$ [7].

Like Laplacian Eigenmaps, most common nonlinear dimension reduction algorithms only provide an explicit mapping for a discrete dataset. Therefore, the inverse mapping $\boldsymbol{\Phi}_n^{-1}$ is also only defined on the data. The goal of the present work is to generate a numerical extension of $\boldsymbol{\Phi}_n^{-1}$ to all of $\boldsymbol{\Phi}(\mathcal{M}) \subset \mathbb{R}^d$, the image of the corresponding continuous operator. To simplify the problem, we assume the mapping provided by our dimension reduction algorithm coincides with the limiting continuous operator on the data, i.e. $\boldsymbol{\Phi}_n(\boldsymbol{x}^{(i)}) = \boldsymbol{\Phi}(\boldsymbol{x}^{(i)})$ for $i = 1, \ldots, n$. This notation allows us to discard the notation $\boldsymbol{\Phi}_n$ to denote the data-specific mapping. Instead, we use the notation $\boldsymbol{\Phi}$ to simultaneously denote both the discrete and continuous mapping. With this notation, we propose a method to construct an approximate inverse

$$\boldsymbol{\Phi}^\dagger : \boldsymbol{\Phi}(\mathcal{M}) \to \mathbb{R}^D, \tag{2}$$

such that $\lim_{n\to\infty} \boldsymbol{\Phi}^\dagger(\boldsymbol{y}) = \boldsymbol{\Phi}^{-1}(\boldsymbol{y})$ for all $\boldsymbol{y} \in \boldsymbol{\Phi}(\mathcal{M})$.

Owing to the connections between the method proposed below and the Nyström extension, we use notation specifically suggestive of a dimension reduction method involving a spectral decomposition of the weight matrix of the data in $\mathbb{R}^D$, which is used for instance in Laplacian Eigenmaps, LLE, ISOMAP, etc. However, the numerical extension method has much broader application, and may be applied to numerically invert any smooth bi-Lipschitz embedding of a dataset in $\mathbb{R}^d$.

We denote the weight matrix W, where $W_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ is a measure of the affinity between data points $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)} \in \mathbb{R}^D$. A typical measure is the Gaussian: $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \exp(-\epsilon^2 \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2)$. As noted above, we are specifically interested in dimension reduction methods that involve a spectral decomposition of W. Thus we consider the eigenvalue problem

$$W\boldsymbol{\phi}_l = \lambda_l \boldsymbol{\phi}_l, \tag{3}$$

where $\boldsymbol{\phi}_l$ and λ_l are the eigenvectors and corresponding eigenvalues of W for $i = 1, \ldots, n$. For $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and $\Phi = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{n \times n}$, we consider the full eigenvector decomposition of the weight matrix W:

$$W = \Phi \Lambda \Phi^T. \tag{4}$$

We define the mapping $\phi_l : \mathbb{R}^D \to \mathbb{R}$ as follows: $\phi_l(\boldsymbol{x}^{(i)}) = \phi_{il}$, where ϕ_{il} is the i, l entry of the eigenvector matrix $\Phi \in \mathbb{R}^{n \times n}$. Using this notation,

$$\boldsymbol{\phi}_l = [\phi_l(\boldsymbol{x}^{(1)}), \ldots, \phi_l(\boldsymbol{x}^{(n)})]^T, \tag{5}$$

and we naturally choose to denote the spectral embedding of $\boldsymbol{x}^{(i)} \in \mathcal{M} \subset \mathbb{R}^D$ as

$$\boldsymbol{\Phi}(\boldsymbol{x}^{(i)}) = [\boldsymbol{\phi}_1(\boldsymbol{x}^{(i)}), \ldots, \boldsymbol{\phi}_d(\boldsymbol{x}^{(i)})] \in \boldsymbol{\Phi}(\mathcal{M}) \subset \mathbb{R}^d. \tag{6}$$

(note: typically $d << n$, and dimension reduction is achieved using a partial eigenvector decomposition).

Restating the problem: we would like to generate an approximate inverse $\boldsymbol{\Phi}^\dagger : \boldsymbol{\Phi}(\mathcal{M}) \to \mathbb{R}^D$ that will converge to the true inverse $\boldsymbol{\Phi}^{-1} : \boldsymbol{\Phi}(\mathcal{M}) \to \mathcal{M}$ in the limit at $n \to \infty$. For notational simplicity, we will introduce the following notation for the data embedded in $\mathbb{R}^d$:

$$\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{y}^{(i)} = \boldsymbol{\Phi}(\boldsymbol{x}^{(i)}). \tag{7}$$

We do not consider how the new point $\boldsymbol{y} \in \boldsymbol{\Phi}(\mathcal{M})$ is generated, but simply assume that there exists an $\boldsymbol{x} \in \mathcal{M} \subset \mathbb{R}^D$ such that $\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{y}$. Methods exist for approximating a new point $\boldsymbol{y}$ in the image of $\boldsymbol{\Phi}$ (e.g. the Nyström extension [8]). However, in the present work we focus our attention on the problem of approximating the inverse map, and thus, we assume that we can accurately generate a point $\boldsymbol{y}$ in the range of $\boldsymbol{\Phi}$.

2.1. Linear Inverse Mapping

One method proposed for this type of inverse mapping was proposed by [6]. The authors in [6] approximate $\boldsymbol{x} = \boldsymbol{\Phi}^{-1}(\boldsymbol{y})$ with $\boldsymbol{\Phi}^{\dagger}(\boldsymbol{y})$, defined by linearly interpolating the existing coordinates:

$$\boldsymbol{\Phi}^{\dagger}(\boldsymbol{y}) = \sum_{j:\boldsymbol{y}^{(j)} \in \mathcal{N}_{\boldsymbol{y}}} c_j \boldsymbol{x}^{(j)}, \tag{8}$$

where each $\boldsymbol{x}^{(j)}$ is an original point that is mapped to a neighbor $\boldsymbol{y}^{(j)} = \boldsymbol{\Phi}(\boldsymbol{x}^{(j)})$ of $\boldsymbol{y}$. Here, $\mathcal{N}_{\boldsymbol{y}}$ denotes the set of neighbors of $\boldsymbol{y}$ in $\mathbb{R}^d$. The interpolation coefficients are calculated as follows,

$$c_j = \frac{\exp(-\|\boldsymbol{y} - \boldsymbol{y}^{(j)}\|^2/\sigma)}{\sum_{i:\boldsymbol{y}^{(i)} \in \mathcal{N}_{\boldsymbol{y}}} \exp(-\|\boldsymbol{y} - \boldsymbol{y}^{(i)}\|^2/\sigma)}, \tag{9}$$

where σ is chosen to be the distance between $\boldsymbol{y}$ and its nearest neighbor. The algorithm is justified by the assumption that the interpolation weights should be similar between the two spaces $\mathbb{R}^D$ and $\mathbb{R}^d$. In fact, if we consider each point $\boldsymbol{x}^{(j)} \in \mathbb{R}^D$ to be a function of its coordinates $\boldsymbol{y}^{(j)} \in \mathbb{R}^d$, we observe that this algorithm is more commonly known at *Shepard's method* [9], a moving least squares approximation method. This method is optimal in the following way: at any location $\boldsymbol{y}$, the function $\boldsymbol{\Phi}^{-1}(\boldsymbol{y})$ is approximated by the constant function that minimizes the sum of squared errors within a neighborhood $\mathcal{N}_{\boldsymbol{y}}$, weighted according to their proximity to $\boldsymbol{y}$. In [6], the Gaussian is used as the weight function for Shepard's method.

The first observation is that, despite the similar appearance to a radial basis function (RBF) interpolant (see e.g. Eq (10)), the algorithm does not produce an RBF interpolant. Instead, the algorithm uses a set of linear weights that may not even reproduce the new point $\boldsymbol{y}$ as a linear combination of its neighbors in $\mathbb{R}^d$. Another key observation about this inverse mapping method is that the weights c_j are all positive and $\sum c_j = 1$. As a result, the interpolation is restricted to within the convex hull of the neighbors.

The limitations of Shepard's method [6] are illustrated in figure 1. Here the performance of Shepard's method is compared to the algorithm discussed in this paper. For illustrative purposes, the data are scattered on the unit circle in $\mathbb{R}^D = \mathbb{R}^2$, and mapped to $\mathbb{R}^d = \mathbb{R}^2$ via Laplacian Eigenmaps. Shepard's method, as well as our algorithm, are tested on a known point. We observe that restricting interpolation to within the convex hull of the neighbors limits the algorithm's ability to account for curvature. Shepard's method is tested for varying neighborhood sizes (20, 10, or 5 nearest neighbors), and is found to perform better with fewer nearest neighbors, as the convex hull of the neighbors become more-localized near the true point. However, the method still fails to honor the curvature of the manifold. Our approach performs well, matching the curvature of the manifold.

Provided sufficient data to characterize a curved manifold are available, we believe that a good approximate inverse should be capable of reproducing curvature. A natural way to approach the problem of extending the inverse map, while accounting for curvature, is via interpolation. To accomplish this, we

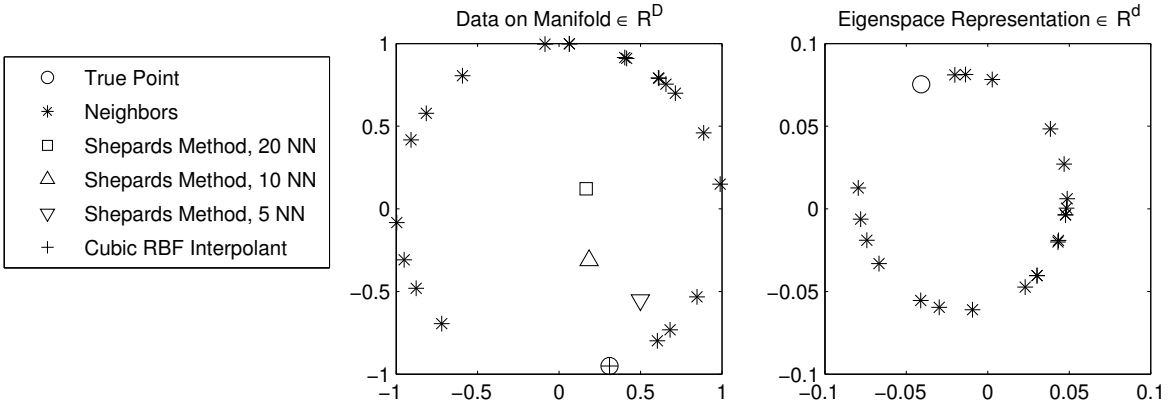

Figure 1: Comparison of Shepard's method [6] with cubic RBF method, for 20 points randomly scattered on the unit circle in $\mathbb{R}^2$. Shepard's method can only interpolate within the convex hull of the neighbors, thus the algorithm fails to match curvature. Shepard's method is tested for 20, 10, and 5 nearest neighbors (NN).

consider $\boldsymbol{x} \in \mathbb{R}^D$ to be a multivariate function of $\boldsymbol{y} \in \mathbb{R}^d$, subject to the constraints $\boldsymbol{x}^{(i)} = \boldsymbol{\Phi}^{-1}(\boldsymbol{y}^{(i)})$, for $i = 1, \ldots, n$. Most interpolation strategies are directly applicable to univariate functions. Thus, we independently interpolate each coordinate in $\mathbb{R}^D$ as a function of all of the coordinates in $\mathbb{R}^d$. To simplify the present analysis, we assume the data are free of noise. The approximation problem is not considered here, and will be the subject of future work. We only mention that the proposed interpolation algorithm is easily generalized to provide an approximation strategy that can accommodate a noisy representation of a manifold. Interpolation with RBFs has the advantages of being algorithmically straightforward in any dimension, and can easily account for curvature. To this end, we begin the next section with a basic introduction to radial basis function interpolation. Abundant literature exists on the topic for the curious reader (e.g. [9, 10, 11] provide surveys at varying levels of detail).

The remainder of the paper is organized as follows. Section 3 introduces the RBF inverse mapping algorithm. Section 4 considers convergence of Gaussian RBF interpolants, followed by section 5 which makes observations about problems arising from ill-conditioning of the interpolation matrix. Section 6 introduces, motivates, and discusses convergence of an alternative scale-free cubic RBF kernel. Section 7 demonstrates performance of the inverse mapping algorithm on several example problems. Section 8 contains a discussion and novel interpretation of the Nyström extension, followed by a final summary in section 9.

3. Radial Basis Function Inverse Mapping

Before introducing our inverse mapping algorithm, we define a basic RBF interpolant. For the RBF kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and the node set $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ with corresponding function values $\{f(\boldsymbol{z}^{(1)}), \ldots, f(\boldsymbol{z}^{(n)})\}$, the RBF interpolant takes the form

$$s(\boldsymbol{z}) = \sum_{j=1}^{n} \alpha^{(j)} k(\boldsymbol{z}, \boldsymbol{z}^{(j)}). \tag{10}$$

The weights, $\{\alpha^{(1)}, \ldots, \alpha^{(n)}\}$, are determined by solving the following system of equations:

$$\begin{bmatrix} k(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(1)}) & \cdots & k(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{z}^{(n)}, \boldsymbol{z}^{(1)}) & \cdots & k(\boldsymbol{z}^{(n)}, \boldsymbol{z}^{(n)}) \end{bmatrix} \begin{bmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(n)} \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{z}^{(1)}) \\ \vdots \\ f(\boldsymbol{z}^{(n)}) \end{bmatrix} \tag{11}$$

which enforces the condition $s(\boldsymbol{z}^{(i)}) = f(\boldsymbol{z}^{(i)})$, for $i = 1, \ldots, n$.

We propose a method to compute the inverse mapping, which better honors the curvature of the underlying manifold by interpolating using RBFs. Similar methods have been explored in [4, 5, 12] to interpolate data on a low-dimensional manifold. In our case, we generate a D-dimensional RBF interpolant: $\boldsymbol{\Phi}^{\dagger} : \mathbb{R}^d \to \mathbb{R}^D$. To accomplish this, we simultaneously generate D independent RBF interpolants to each coordinate in $\mathbb{R}^D$.

Let K denote the RBF kernel matrix for the data embedded in $\mathbb{R}^d$: $K_{ij} = k(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. In [6], the authors assume that $K \approx W$ by using the same kernel in $\mathbb{R}^d$ and $\mathbb{R}^D$. In fact, this assumption is not needed in this work, and we may consider various RBF interpolation kernels, not only the Gaussian (which is typically used to generate the weight matrix in $\mathbb{R}^D$). In order to generate the radial basis function interpolant, we begin by solving the system

$$KA = X, \tag{12}$$

where the i-th row of $X \in \mathbb{R}^{n \times D}$ are the D coordinates of $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$. The j-th column of $A \in \mathbb{R}^{n \times D}$ are the interpolation coefficients for the RBF interpolant of the j-th dimension in $\mathbb{R}^D$. In detail,

$$\begin{bmatrix} k(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(1)}) & \cdots & k(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{y}^{(n)}, \boldsymbol{y}^{(1)}) & \cdots & k(\boldsymbol{y}^{(n)}, \boldsymbol{y}^{(n)}) \end{bmatrix} \begin{bmatrix} \alpha_1^{(1)} & & \alpha_D^{(1)} \\ \vdots & \cdots & \vdots \\ \alpha_1^{(n)} & & \alpha_D^{(n)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & & x_D^{(1)} \\ \vdots & \cdots & \vdots \\ x_1^{(n)} & & x_D^{(n)} \end{bmatrix} \tag{13}$$

where $\alpha_j^{(i)}$ for $i = 1, \ldots, n$ are the RBF interpolation coefficients for the j-th dimension in $\mathbb{R}^D$. The new point $\boldsymbol{x} = \boldsymbol{\Phi}^{-1}(\boldsymbol{y})$ is interpolated using (10), which can be written in matrix form as

$$\boldsymbol{\Phi}^{\dagger}(\boldsymbol{y})^T = \boldsymbol{k}(\boldsymbol{y}, \cdot)^T A = \boldsymbol{k}(\boldsymbol{y}, \cdot)^T K^{-1} X, \tag{14}$$

where $\boldsymbol{k}(\boldsymbol{y}, \cdot) = [k(\boldsymbol{y}, \boldsymbol{y}^{(1)}), \ldots, k(\boldsymbol{y}, \boldsymbol{y}^{(n)})]^T$.

For non-coincident interpolation nodes $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}\} \in \mathbb{R}^d$, and a Gaussian kernel k we are guaranteed K is non-singular [9]. For large data sets, the interpolation need not depend on all the data, as the RBF interpolant is only sensitive to points in the neighborhood of interest. The interpolation algorithm may be implemented locally, providing immense savings in computation time.

4. Convergence of the Interpolant

As we consider interpolation of the data using RBFs, three questions must be addressed: 1) Given a set of interpolation nodes, is the interpolation matrix necessarily non-singular? 2) What types of functions can be approximated? 3) What convergence rate can we expect as we populate the domain with additional nodes?

Convergence criteria for RBF interpolants have been studied extensively in the literature. For a detailed treatment, see [9, 11]. In this section we briefly introduce the necessary concepts to understand convergence of Gaussian RBF interpolants. Before considering the questions posed above, we first study the expected smoothness of the embedding $\mathbf{\Phi}$ and its inverse.

4.1. Properties of the Inverse of the Graph Laplacian

The data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ are a discrete representation of a bounded smooth manifold $\mathcal{M} \subset \mathbb{R}^D$. We model the manifold $\mathcal{M}$ as a function of the eigenvectors of the graph Laplacian: $\mathbf{\Phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \ldots, \phi_d(\boldsymbol{x})]^T$. Thus, the inverse mapping $\mathbf{\Phi}^{-1} : \mathbf{\Phi}(\mathcal{M}) \to \mathbb{R}^D$ will inherit properties from the forward mapping $\mathbf{\Phi} : \mathcal{M} \to \mathbb{R}^d$. We recall that $\boldsymbol{\phi}_l$ is an eigenvector of the graph Laplacian matrix, which is a discrete approximation of the Laplace Beltrami operator on the manifold $\mathcal{M}$. Thus, in the continuous setting the mapping $\mathbf{\Phi} : \mathcal{M} \to \mathbf{\Phi}(\mathcal{M}) \subset \mathbb{R}^d$ must be infinitely differentiable, implying that $\mathbf{\Phi}^{-1} : \mathbf{\Phi}(\mathcal{M}) \to \mathbb{R}^D$ is also infinitely differentiable. By assumption, $\mathcal{M}$ is a bounded subset of $\mathbb{R}^D$. Provided that $\mathbf{\Phi}$ is bi-Lipschitz, then $\mathbf{\Phi}(\mathcal{M})$ is a bounded subset of $\mathbb{R}^d$. As a result, the components of $\mathbf{\Phi}^{-1}$ are in $C^\infty(\mathbf{\Phi}(\mathcal{M}))$.

4.2. Properties of Gaussian RBF Interpolants

To consider the questions posed at the beginning of this section, we must first introduce several definitions. We begin with the definition of a *positive definite kernel*.

Definition 1. *A real-valued continuous function* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *is called* positive definite on $\mathbb{R}^d$ *if*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{z}^{(i)}, \boldsymbol{z}^{(j)}) \geq 0, \tag{15}$$

for n *distinct points* $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\} \subset \mathbb{R}^d$*, and* $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T \in \mathbb{R}^n$*. If in addition, the quadratic form (15) is equal to zero if and only if* $\boldsymbol{\alpha} = 0$*, then* k *is called* strictly positive definite on $\mathbb{R}^d$ *[9].*

Among others, a commonly used strictly positive definite kernel is the Gaussian [9]. We note that this property of the Gaussian implies unique solvability of (13).

We now consider the second question: what types of functions can we approximate to arbitrary precision? As we might expect, an RBF interpolant will converge to functions contained in the closure of the span of all translates of the kernel k, known as the *native space*:

Definition 2. *Given a strictly positive definite reproducing kernel $k : \Omega \times \Omega \to \mathbb{R}$ on a Hilbert space, the* native space $\mathcal{N}_k(\Omega)$ *is defined as the completion of the space of linear combinations of the kernel: $\mathcal{H}_k(\Omega) = span\{k(\cdot, \boldsymbol{z}) : \boldsymbol{z} \in \Omega\}$. This completion is with respect to the k-norm, which is induced by the inner-product given by the reproducing kernel k on the pre-Hilbert space $\mathcal{H}_k(\Omega)$ [9].*

For $\Omega = \mathbb{R}^d$, the native space for the Gaussian RBF is

$$\mathcal{N}_k(\mathbb{R}^d) = \{f \in L_2(\mathbb{R}^d) \bigcap C(\mathbb{R}^d) : \frac{\hat{f}}{\sqrt{\hat{k}}} \in L_2(\mathbb{R}^d)\}, \tag{16}$$

where $\hat{f}$ and $\hat{k}$ are the Fourier transforms of f and k, respectively [9]. We observe that the native space for Gaussians is not large, and is restricted to functions whose Fourier transform decays at least as fast as the Gaussian [9].

To answer the third question, we must be able to consistently quantify the density of interpolation nodes. The commonly used measure is the *fill distance*, the maximum distance from an interpolation node:

Definition 3. *For the domain $\Omega \subset \mathbb{R}^d$ and a set of interpolation nodes $Z = \{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\} \subset \Omega$ the* fill distance*, $h_{Z,\Omega}$, is defined by*

$$h_{Z,\Omega} := \sup_{\boldsymbol{z} \in \Omega} \min_{\boldsymbol{z}^{(j)} \in Z} \|\boldsymbol{z} - \boldsymbol{z}^{(j)}\|. \tag{17}$$

Theorem 15.1 in [9] establishes exponential L^∞ convergence of a Gaussian RBF interpolant to functions in the native space (with respect to decreasing fill distance).

With a basic understanding of the convergence properties of Gaussian RBF interpolants, we may now consider whether the components of $\boldsymbol{\Phi}^{-1}$ can be approximated to arbitrary precision. Despite the regularity of $\boldsymbol{\Phi}^{-1}$ over the domain $\Omega = \boldsymbol{\Phi}(\mathcal{M})$, it appears that there may not be a consistent extension of the function to all of $\mathbb{R}^d$ such that its components are members of the (very restricted) native space of the Gaussian. For example, any extension to a compactly supported function will exhibit a slowly decaying Fourier transform relative to the Gaussian. Any extension with Gaussian decay is the sum of a Gaussian and a compactly supported function, again with slowly decaying Fourier transform. As a result, we are likely restricted to *approximate approximation* when using the Gaussian RBF. However, these issues are ultimately irrelevant, as numerical issues will prevent convergence of Gaussian RBF interpolants, even within the native space.

5. Ill-Conditioning

In this section we discuss ill-conditioning of Gaussian kernel matrices. The condition number is closely related to the choice of scale parameter ϵ. We use the scale parameter convention common to the RBF community: $k(\boldsymbol{z}, \boldsymbol{w}) = \exp(-\epsilon^2 \|\boldsymbol{z} - \boldsymbol{w}\|^2)$. This scale parameter corresponds to the inverse of σ, the commonly used scale parameter within the machine learning community. With the RBF convention, a small

scale parameter corresponds to wide and flat basis functions, while a larger scale parameter corresponds to narrow and localized Gaussians.

In the previous section, it was observed that a Gaussian RBF interpolant will converge exponentially to a function in the native space with respect to fill distance. However, the error bounds are theoretical and do not consider interpolation error resulting from numerical ill-conditioning. It is a well known that if the same scale parameter is used as fill distance is reduced, a Gaussian kernel matrix W will rapidly become ill-conditioned, and the resulting interpolant will exhibit numerical *saturation error*. This issue is common among many RBF interpolants. As shown in [13], the eigenvalues of W follow patterns in the powers of ϵ that increase with successive eigenvalues, which leads to rapid ill-conditioning of W with increasing n. These patterns depend on a number of factors including dimension as well as the geometry of the nodes. The nature of this behavior is not the focus of this paper, and the interested reader is referred to [13] for additional details.

In [14], the authors show that the numerical rank (the number of eigenvalues larger than a certain threshold) of a Gaussian kernel matrix is independent of the number of data points inside a box in $\mathbb{R}^D$, but instead only depends on the scale parameter. As a result, as additional points are added, the additional eigenvalues added are small and rapidly increase the condition number of the matrix.

The relationship between the condition number of W and the spacing of interpolation nodes is explored in figure 2. Owing to the difficulty in precisely establishing the boundary of the domain $\Omega \subset \mathbb{R}^d$ given a discrete set of randomly sampled data, we note that estimating the fill distance $h_{Z,\Omega}$ is somewhat difficult. Additionally, the fill distance is a measure of the "worst case", and may not be representative of the "typical" spacing between nodes. Thus, we consider a proxy for fill distance which depends only on mutual distances between the data points. We use the term *local fill distance*, $\bar{h}_{local}$, to denote the average distance to a nearest neighbor:

$$\bar{h}_{local} := \frac{1}{n} \sum_{i=1}^{n} \min_{j \neq i} \|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\|. \tag{18}$$

The local fill distance is less sensitive to slightly irregular node spacing resulting from random sampling in the domain, and thus more representative of the "typical" spacing of interpolation nodes. Nonetheless, given a regular sampling scheme, the local fill distance should provide a good proxy for fill distance. We note that several authors have recently proposed a similar notion of local fill distance which can be used to derive more accurate pointwise error estimates [15, 16]. In figure 2, we observe rapid ill-conditioning of the weight matrix with respect to decreasing local fill distance.

Conversely, if the fill distance remains constant while the Gaussian kernel scale parameter is reduced, the resulting interpolant improves until ill-conditioning of the W matrix leads to propagation of numerical errors. Figure 3, demonstrates the rapid increase in condition number of W with decreasing values of ϵ. When interpolating with the Gaussian kernel, choice of the scale parameter ϵ is difficult. On one hand,

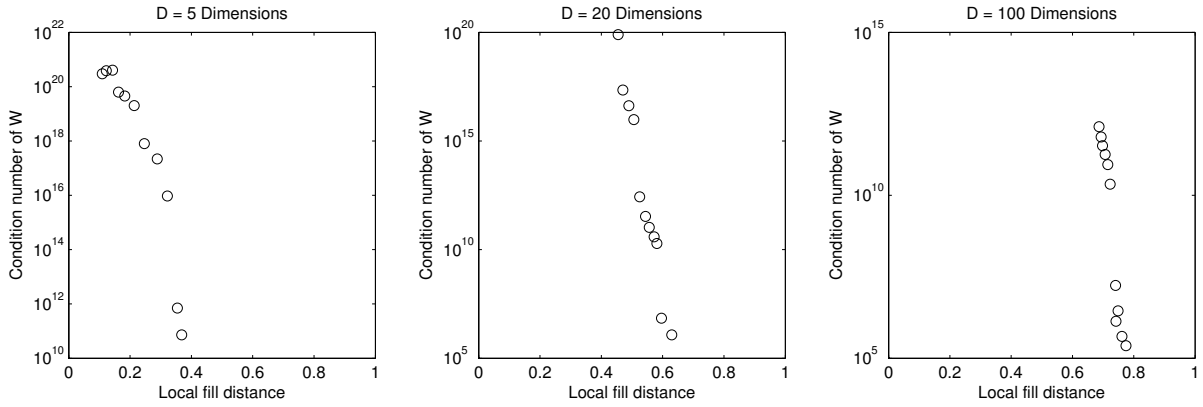

Figure 2: Condition number of the Gaussian kernel matrix W for points randomly scattered on the first quadrant of the unit sphere in various dimensions with scale parameter $\epsilon = 10^{-2}$, for varying local fill distance (18). Note: the same range of n was used in each dimension (10 to 1000). In high dimension, it takes a large number of points to reduce fill distance. However, the condition number of W still grows rapidly for increasing n.

smaller values of ϵ likely lead to a better interpolant. For example: in 1-d, a Gaussian RBF interpolant will converge to the Lagrange interpolating polynomial in the limit as $\epsilon \to 0$ [17]. On the other hand, the interpolation matrix becomes rapidly ill-conditioned for decreasing ϵ. Figure 4 shows the interpolation error versus ϵ for a function in 1-d.

One solution is to generate the RBF interpolant using a stable algorithm. The first such algorithm was the Contour-Pade algorithm [18]. This algorithm relies on contour integration in the complex-ϵ plane to avoid the ill-conditioned region, while calculating the interpolant in the $\epsilon \to 0$ limit. The RBF-QR [19, 20] and RBF-GA [21] algorithms rely on generating a better basis for the same space spanned by the Gaussian RBFs, which allows stable evaluation of the interpolant for small ϵ. All of the stable algorithms are more computationally intensive and algorithmically complex than the RBF-Direct method, making them undesirable for the inverse-mapping interpolation task. In the following section we propose a method that avoids the numerical issues associated with Gaussian RBF interpolation by using a numerically stable, scale-free interpolation kernel.

6. Cubic RBF Interpolation

Saturation error can be avoided by using the scale-free RBF kernel $g(\boldsymbol{z}, \boldsymbol{w}) = \|\boldsymbol{z} - \boldsymbol{w}\|^3$, one instance from the set of RBF kernels known as the *radial powers*:

$$g(\boldsymbol{z}, \boldsymbol{w}) = \|\boldsymbol{z} - \boldsymbol{w}\|^\beta \quad \text{for} \quad \beta = 1, 3, 5, \ldots. \tag{19}$$

Together with the *thin plate splines*:

$$g(\boldsymbol{z}, \boldsymbol{w}) = \|\boldsymbol{z} - \boldsymbol{w}\|^\beta \log\|\boldsymbol{z} - \boldsymbol{w}\| \quad \text{for} \quad \beta = 2, 4, 6, \ldots, \tag{20}$$

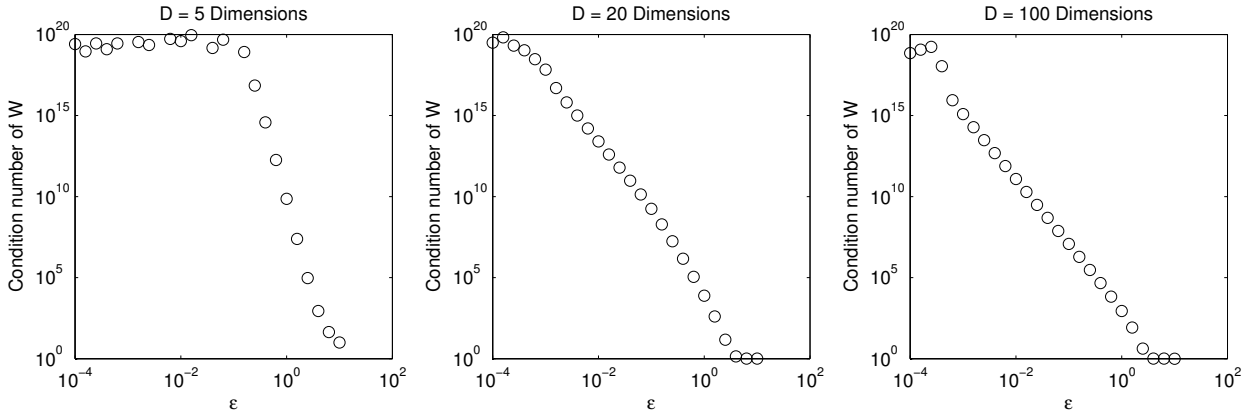

Figure 3: Condition number of the Gaussian kernel matrix W for $n = 200$ points randomly scattered on the first quadrant of the unit sphere in various dimensions, for varying ϵ.

they form the family of RBFs known as the *polyharmonic splines*.

For RBF interpolation, the cubic kernel is less intuitive than the Gaussian, as it is a monotonically increasing function. To motivate the use of this kernel, we begin by considering the one dimensional case, in which the cubic RBF generates a cubic spline interpolant.

6.1. Cubic RBF and Cubic Splines

For interpolations nodes $\{z^{(1)}, \ldots, z^{(n)}\} \subset \mathbb{R}$, the cubic RBF interpolant is of the form

$$s(z) = \sum_{i=1}^{n} \alpha^{(i)} |z - z^{(i)}|^3. \tag{21}$$

Each term in the sum is piecewise cubic, with continuous first and second derivatives. Thus, $s(z)$ inherits the same properties, and is observed to be a cubic spline. The cubic RBF kernel has a discontinuous third derivative at the origin, which translates into discontinuities in ds^3/dz^3 at the interpolation nodes (corresponding to the typical formulation of a cubic spline). In general, the requirements of a cubic spline leaves two additional degrees of freedom that may be chosen to provide additional regularity.

The cubic RBF interpolant automatically makes arbitrary endpoint choices for the additional degrees of freedom. We briefly investigate the endpoint behavior. See [22] for a more detailed treatment. For simplicity, we consider interpolation on the interval $[-1, 1]$: $-1 = z^{(1)} < z^{(2)} < \ldots < z^{(n)} = 1$. All terms in the cubic RBF interpolant change sign between $z^{(1)}$ and $z^{(n)}$:

$$s(z) \quad = \begin{cases} \sum_{i=1}^{n} \alpha^{(i)} (z - z^{(i)})^3 & \text{for} \quad z \geq 1 \\ \sum_{i=1}^{n} -\alpha^{(i)} (z - z^{(i)})^3 & \text{for} \quad z \leq -1. \end{cases} \tag{22}$$

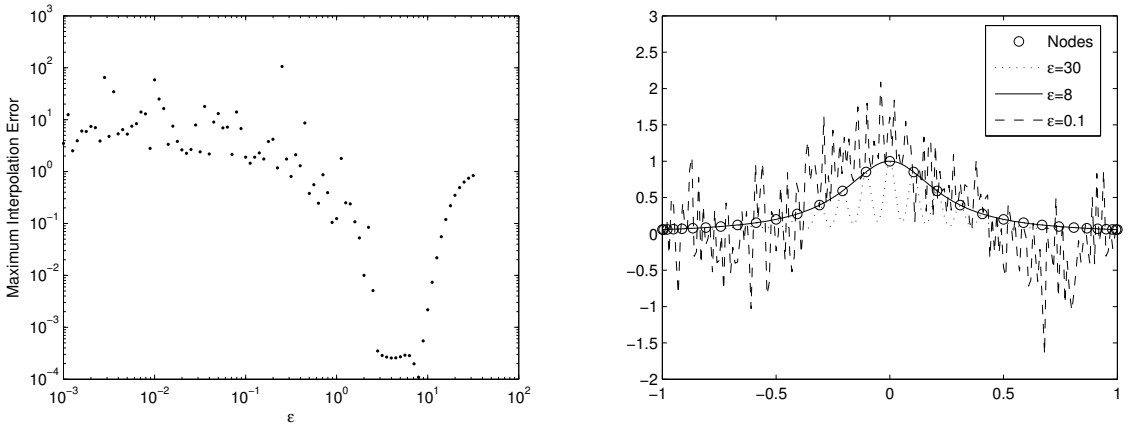

Figure 4: Maximum error versus ϵ (left) for a Gaussian RBF interpolant, and example interpolants for several representative values of ϵ (right). The function $f(x) = 1/(1+16x^2)$ is interpolated using the Chebyshev nodes on the interval $[-1, 1]$ to avoid the Runge phenomenon. For large ϵ, the basis function become too localized, and provide a poor interpolant. For small ϵ, ill-conditioning of the interpolation matrix destroys the interpolant.

A system of equations can be set up and solved to show the following endpoint conditions on the second derivative [22]:

$$\begin{aligned} s''(1) = &\quad 2s'(1) - s'(-1) - \tfrac{3}{2}(s(1) + s(-1)) \\ s''(-1) = &\quad s'(1) - 2s'(-1) - \tfrac{3}{2}(s(1) + s(-1)). \end{aligned} \tag{23}$$

One common set of additional conditions are those of the natural spline, which enforce the conditions $s''(1) = s''(-1) = 0$. These conditions are achieved by adding constant and linear polynomial terms to the cubic RBF interpolant,

$$s(z) = \beta_0 + \beta_1 z + \sum_{i=1}^{n} \alpha^{(i)} |z - z^{(i)}|^3, \tag{24}$$

subject to the additional constraints $\sum_{i=1}^{n} \alpha^{(i)} = \sum_{i=1}^{n} \alpha^{(i)} z^{(i)} = 0$. Then, for $z \geq 1$,

$$s(z) = \beta_0 + \beta_1 z + \sum_{i=1}^{n} \alpha^{(i)} (z - z^{(i)})^3, \tag{25}$$

and we find that

$$s''(z) = 6z \sum_{i=1}^{n} \alpha^{(i)} - 6 \sum_{i=1}^{n} \alpha^{(i)} z^{(i)} = 0. \tag{26}$$

By an identical argument it is shown that $s''(z) = 0$ for $z \leq -1$. Figures 5 and 6 demonstrate the improved behavior of the cubic RBF interpolants near boundaries when constant and linear polynomial terms are included.

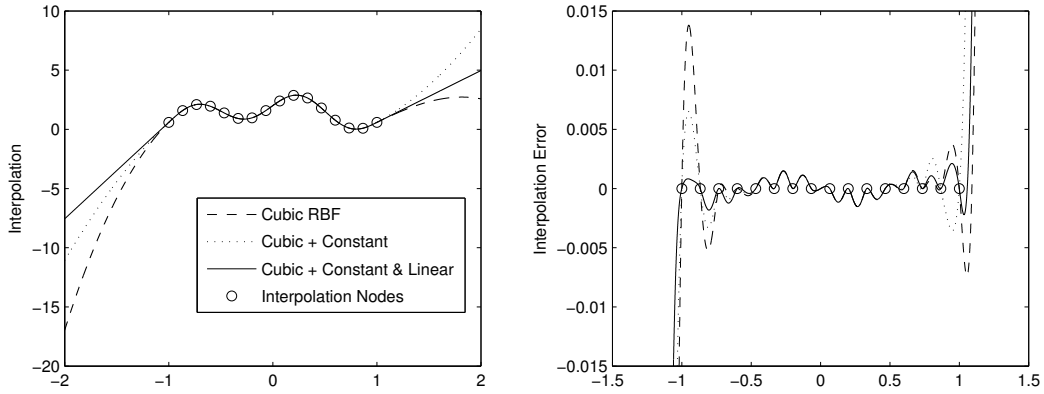

Figure 5: Cubic RBF interpolants with additional constant and linear terms (left) and corresponding error (right) for the test function $f(z) = 1 + \sin(2\pi z) + \cos(2z)$.

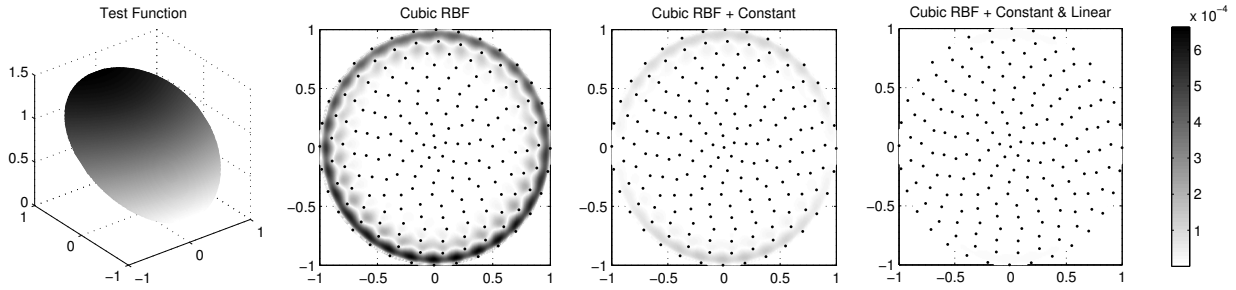

Figure 6: Cubic RBF interpolants with additional constant and linear terms on the test function $f(z, w) = \frac{1}{1+0.2z^2+0.3w^2} + 0.5w$. Color plots represent absolute interpolation error: 0 (white) to 7×10^{-4} (black).

6.2. Multivariate Cubic RBF Interpolation

As we consider multivariate interpolation using the cubic RBF, we must again address the following three questions: 1) Given a set of interpolation nodes, is the interpolation matrix non-singular? 2) What types of functions can be approximated? 3) What convergence rate can we expect as we populate the domain with additional nodes?

In order to address the question of solvability, we must introduce some additional definitions. We begin with the definition of a *conditionally positive definite kernel*, a generalization of positive definite.

Definition 4. *A real-valued continuous function* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *is called* conditionally positive definite of order m on $\mathbb{R}^d$ *if*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{z}^{(i)}, \boldsymbol{z}^{(j)}) \geq 0, \tag{27}$$

for n distinct points $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\} \subset \mathbb{R}^d$, and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T \in \mathbb{R}^n$, subject to

$$\sum_{i=1}^{n} \alpha_i p(\boldsymbol{z}^{(i)}) = 0, \tag{28}$$

where $p(\boldsymbol{z})$ is any real-valued polynomial of degree at most $m-1$. If in addition, the quadratic form (27) is equal to zero if and only if $\boldsymbol{\alpha} = 0$, then k is called strictly conditionally positive definite on $\mathbb{R}^d$ *[9].*

The cubic RBF is conditionally positive definite of order 2 on $\mathbb{R}^d$ [9]. The attentive reader may note that the augmentation of the cubic RBF basis with the constant and linear polynomials will help to provide unique solvability of the interpolation system, given this property of the cubic kernel. Unique solvability will also require a mild condition on the node locations. For this, we define the concept of *m-unisolvency*:

Definition 5. *The set of nodes $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\} \subset \mathbb{R}^d$ is called* m-unisolvent *if the unique polynomial of total degree at most m interpolating zero data on $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ is the zero polynomial.*

Given the properties of the cubic RBF kernel, we require that $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ form a 1-unisolvent set in $\mathbb{R}^d$. For a simple example from [9], 3 collinear points in $\mathbb{R}^2$ do not form a 1-unisolvent set, because different rotations of the zero-plane can interpolate zero data over these nodes. However, 3 non-collinear points in $\mathbb{R}^2$ form a 1-unisolvent set. The requirement of a 1-unisolvent set in $\mathbb{R}^d$ is equivalent to the following condition:

$$\text{span}\{(\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}) \quad \text{for} \quad i, j = 1, \ldots, n\} = \mathbb{R}^d. \tag{29}$$

With the definition of m-unisolvent, and the property of strictly conditionally positive definite, we may now employ the following theorem to guarantee non-singularity of the interpolation matrix:

Theorem 1 (7.2 from [9]). *If the real-valued even function $\gamma : \mathbb{R}^d \to \mathbb{R}$ is strictly conditionally positive definite of order m and the points $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ form an $(m-1)$-unisolvent set, then the following system of linear equations is uniquely solvable:*

$$\begin{bmatrix} G & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}, \tag{30}$$

where $G_{ij} = \gamma(\|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\|)$ for $i, j = 1, \ldots, n$; $\boldsymbol{f} = [f(\boldsymbol{z}^{(1)}), \ldots, f(\boldsymbol{z}^{(n)})]^T$; $P_{kl} = p_l(\boldsymbol{z}^{(k)})$ for $k = 1, \ldots, n$ and $l = 1, \ldots, M$, and the polynomials $p_l(\boldsymbol{z})$ for $l = 1, \ldots, M$ form a basis for the linear space of all polynomials up to degree $(m-1)$.

The proof of the theorem demonstrates the need for the technical conditions on the interpolation nodes as well as the augmentation of the cubic RBF basis with the constant and linear polynomials to guarantee a unique interpolant.

Proof 1. Following the proof in [9], we assume $[\boldsymbol{\alpha}, \boldsymbol{\beta}]^T$ is in the null space of the interpolation matrix in (30), i.e. we let $\boldsymbol{f} = \mathbf{0}$. If we consider the top block, and left multiply by $\boldsymbol{\alpha}^T$ we have

$$\boldsymbol{\alpha}^T G\boldsymbol{\alpha} + \boldsymbol{\alpha}^T P\boldsymbol{\beta} = \mathbf{0}. \tag{31}$$

We observe that $\boldsymbol{\alpha}^T P = \mathbf{0}^T$, since $P^T\boldsymbol{\alpha} = \mathbf{0}$ from the lower block of (30). This leaves

$$\boldsymbol{\alpha}^T G\boldsymbol{\alpha} = \mathbf{0}. \tag{32}$$

We observe that the quadratic form $\boldsymbol{\alpha}^T G\boldsymbol{\alpha} = \mathbf{0}$ if and only if $\boldsymbol{\alpha} = \mathbf{0}$, since γ is strictly conditionally positive definite of order m. Revisiting the top block of (30), we now see that

$$P\boldsymbol{\beta} = \mathbf{0}. \tag{33}$$

The $(m-1)$-unisolvency of $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ enforces that $\boldsymbol{\beta} = \mathbf{0}$. $\square$

In our case ($m = 2$), we use the constant and linear polynomials: $p_1(\boldsymbol{z}) = 1$ and $p_l(\boldsymbol{z}) = z_{l-1}$ for $l = 2, \ldots, d+1$ (z_l denotes the l-th coordinate of $\boldsymbol{z}$ in $\mathbb{R}^d$). Then, given that $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(n)}\}$ forms a 1-unisolvent set of interpolation nodes in $\mathbb{R}^d$, we are guaranteed the existence of a unique interpolant $s : \mathbb{R}^d \to \mathbb{R}$ of the form:

$$s(\boldsymbol{z}) = \sum_{i=1}^{n} \alpha^{(i)} \|\boldsymbol{z} - \boldsymbol{z}^{(i)}\|^3 + \beta_0 + \sum_{j=1}^{d} \beta_j z_j. \tag{34}$$

We must now concern ourselves with the types of target functions to which a cubic RBF interpolant will converge. In particular, we hope to approximate the components of $\boldsymbol{\Phi}^{-1}$ with arbitrary precision. In order to characterize the native space of the cubic RBF, we begin with the definition of the *Beppo Levi space*:

Definition 6. *For $l > d/2$, the linear space*

$$BL_l(\mathbb{R}^d) := \{f \in C(\mathbb{R}^d) : D^\alpha f \in L_2(\mathbb{R}^d), \forall |\alpha| = l\}, \tag{35}$$

equipped with the inner product

$$\langle f, g \rangle_{BL_l(\mathbb{R}^d)} = \sum_{|\alpha|=l} \frac{l!}{\alpha!} \langle D^\alpha f, D^\alpha g \rangle_{L_2(\mathbb{R}^d)}, \tag{36}$$

is called the Beppo Levi space on $\mathbb{R}^d$ of order l*, where D^α denotes the weak derivative of (multi-index) order $\alpha \in \mathbb{N}^d$ on $\mathbb{R}^d$.*

When the dimension, d, is odd, the native space of the cubic RBF is the Beppo Levi space on $\mathbb{R}^d$ of order $l = \frac{d+3}{2}$ [11]. For even dimension, the Beppo Levi space on $\mathbb{R}^d$ of order $l = \frac{d+2}{2}$ corresponds to the native space of the thin plate spline $g(\boldsymbol{z}, \boldsymbol{w}) = \|\boldsymbol{z} - \boldsymbol{w}\|^2 \log\|\boldsymbol{z} - \boldsymbol{w}\|$ [11]. The details of the proof are technical, and

the interested reader is referred to Theorem 10.43 in [11]. We lack a characterization of the native space for the cubic RBF in even dimension.

At this point it is natural to briefly discuss the optimality of polyharmonic spline interpolants. It is a well-known result that the natural cubic spline minimizes the integral of the squared second derivative over the space of interpolants in one-dimension. Likewise, in two-dimensions the thin-plate spline interpolant to scattered data minimizes the so called "bending energy" (hence the term thin-plate spline, or surface spline). These optimality results were generalized to any dimension and order of the iterated Laplacian in [23]. An excellent summary of the results of the variational approach can be found in [24]. The generalized results for the iterated Laplacian have some interesting and initially surprising implications. For example, in three-dimensions the optimal interpolant in terms of minimizing the second-order (generalized) derivatives is the linear radial function [25]. In higher dimension, the family of polyharmonic splines no longer provides an optimal interpolant for minimizing the analog to the two-dimensional bending energy. Instead, appropriate choices from the family of polyharmonic splines (thin plate splines in even dimension and radial powers in odd dimension) are optimal relative to higher orders of the iterated Laplacian. An important observation is that the physical interpretation of bending energy and thus the motivation for optimality with respect to the second order derivatives disappears beyond two-dimensions.

All of our numerical experiments have demonstrated equal or better performance of the cubic RBF relative to the thin plate spline regardless of whether we work in even or odd dimension. Similar results have been observed in [26]. Thus, to promote algorithmic simplicity for practical applications, we have chosen to work solely with the cubic RBF.

For completeness, we will show that the functions we wish to approximate are members of the native space of the cubic RBF provided that we work in odd dimension. We first observe that, $C^\infty(\mathbb{R}^d) \bigcap C_c(\mathbb{R}^d) \subset BL_l(\mathbb{R}^d)$, for any l, where $C_c(\mathbb{R}^d)$ is the linear space of compactly supported functions on $\mathbb{R}^d$. The components of $\mathbf{\Phi}^{-1}$, $\phi_k^{-1} : \mathbf{\Phi}(\mathcal{M}) \to \mathbb{R}$ for $k = 1, \ldots, D$ can be extended to functions in $C^\infty(\mathbb{R}^d) \bigcap C_c(\mathbb{R}^d)$. Thus, the components of $\mathbf{\Phi}^{-1}$ are members of the native space of the cubic RBF, provided that d is odd.

The remaining question is the rate of convergence for a cubic RBF interpolant. Theorem 11.16 in [11] establishes algebraic convergence of at least $\mathcal{O}(h_{Z,\Omega}^{3/2})$ of the cubic RBF interpolant to functions in the native space (we again leave out the technical details here). In practice, we have experienced much higher rates of algebraic convergence, as will be seen in the experimental section. Theorems exist that improve upon the order 3/2 bound for convergence of the cubic RBF, given additional conditions on the smoothness and boundary conditions of the target function, for example in [27, 28, 29]. Any order of algebraic convergence is still slower than the exponential convergence of the Gaussian RBF. Nonetheless, our opinion is that in many applications the benefit gained by the simplicity and numerical stability of the scale-free cubic RBF kernel will offset the reduced convergence rate relative to the Gaussian.

6.3. Far Field Behavior of the Cubic RBF Interpolant

Understanding the behavior of the cubic RBF interpolant near boundaries of the data is more challenging in the multivariate case. Nonetheless, we will observe that inclusion of constant and linear polynomial terms in the interpolant will improve the far field behavior of the interpolant, in addition to providing for unique solvability. To begin the analysis, we translate to spherical coordinates in $\mathbb{R}^d$ (generalizing the argument of [22] from 2 to d-dimensions):

$$\begin{aligned} z_1 &= r\cos\theta_1 = r\gamma_1 \\ z_2 &= r\sin\theta_1\cos\theta_2 = r\gamma_2 \\ &\vdots \\ z_d &= r\sin\theta_1\ldots\sin\theta_{d-1}\cos\theta_d = r\gamma_d. \end{aligned} \tag{37}$$

We proceed to consider the interpolant

$$s(\boldsymbol{z}) = \sum_{i=1}^{n}\alpha^{(i)}\|\boldsymbol{z}-\boldsymbol{z}^{(i)}\|^3 + \beta_0 + \sum_{j=1}^{d}\beta_j z_j \tag{38}$$

$$= \sum_{i=1}^{n}\alpha^{(i)}((z_1-z_1^{(i)})^2+\ldots+(z_d-z_d^{(i)})^2)^{3/2} + \beta_0 + \sum_{j=1}^{d}\beta_j z_j \tag{39}$$

$$= \sum_{i=1}^{n}\alpha^{(i)}((r\gamma_1-z_1^{(i)})^2+\ldots+(r\gamma_d-z_d^{(i)})^2)^{3/2} + \beta_0 + r\sum_{j=1}^{d}\beta_j\gamma_j \tag{40}$$

$$\begin{aligned} = \sum_{i=1}^{n}\alpha^{(i)}(r^2(\gamma_1^2+\ldots+\gamma_d^2) - 2r(\gamma_1 z_1^{(i)}+\ldots+\gamma_d z_d^{(i)}) + ((z_1^{(i)})^2+\ldots+(z_d^{(i)})^2))^{3/2} + \\ \beta_0 + r\sum_{j=1}^{d}\beta_j\gamma_j \end{aligned} \tag{41}$$

$$= r^3\sum_{i=1}^{n}\alpha^{(i)}(1-\frac{1}{r}2(\gamma_1 z_1^{(i)}+\ldots+\gamma_d z_d^{(i)}) + \frac{1}{r^2}((z_1^{(i)})^2+\ldots+(z_d^{(i)})^2))^{3/2} + \beta_0 + r\sum_{j=1}^{d}\beta_j\gamma_j. \tag{42}$$

If we Taylor expand the first sum around $\frac{1}{r}=0$ ($r=\infty$), we find

$$s(\boldsymbol{z}) = r^3\left\{\sum_{i=1}^{n}\alpha^{(i)}\right\} + r^2\left\{-3\sum_{l=1}^{d}\gamma_l\sum_{i=1}^{n}\alpha^{(i)}z_l^{(i)}\right\} + r\{\ldots\} + \{\ldots\} + \frac{1}{r}\{\ldots\} + \ldots \tag{43}$$

$$= r\{\ldots\} + \{\ldots\} + \frac{1}{r}\{\ldots\} + \ldots, \tag{44}$$

where the first two terms are zero as we enforced the constraints $\sum_{i=1}^{n}\alpha^{(i)} = \sum_{i=1}^{n}\alpha^{(i)}z_l^{(i)} = 0$ for $l = 1,\ldots,d$. Thus, we observe that for large r, the cubic RBF with inclusion of constant and linear polynomial terms, behaves linearly in the radial direction. Alternatively, we observe that $\partial^2 s/\partial r^2 = O(1/r^3)$ for large r. By only adding up to linear polynomials, we maintain algorithmic simplicity, and regularize the far field

behavior of the interpolant. The practical consequence of this observation is that we should expect at most linear divergence of the interpolant if we extrapolate outside the convex hull of the data. In particular, we should not experience divergence with r^3 as we might naturally expect based on the form of the interpolant.

7. Experiments

Convergence of RBF interpolants is typically addressed in terms of the L^∞-norm as a function of fill distance. Both of these are measures of the "worst-case" scenarios, and may not represent typical performance over the domain. Additionally, estimation of both of these quantities is difficult, particularly near the boundaries of a discretely sampled domain. For these reasons, we have chosen to measure node spacing and error in a more natural manner. In the first two examples, we synthetically sample a manifold. In order to assess relative performance of the inverse mapping algorithms, we measure error as a function of local fill distance, $\bar{h}_{local}$, from (18). Local fill distance is chosen to provide an appropriate measurement of node spacing under randomized sampling schemes. The relationship between local approximation error and variable local node density has been investigated by [15, 16]. Although we do not address variable node density in the present work, we note that the notion of local fill distance is easily generalized to accommodate this issue which is likely to arise in many applications.

Similarly, error will be quantified with average l^2 error. Given a dataset $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \subset \mathbb{R}^D$ with a corresponding low-dimensional representation $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}\} \subset \mathbb{R}^d$, some random subset $\{\boldsymbol{x}^{(j_1)}, \ldots, \boldsymbol{x}^{(j_m)}\}$ will be selected for testing of the inverse mapping algorithms. For each of these points, $\boldsymbol{x}^{(j_i)} = \boldsymbol{\Phi}^{-1}(\boldsymbol{y}^{(j_i)})$, we will compute the approximate inverse $\boldsymbol{\Phi}^\dagger(\boldsymbol{y}^{(j_i)})$. The average l^2 error, E_{avg}, is calculated as

$$E_{avg} = \frac{1}{m} \sum_{i=1}^{m} \|\boldsymbol{x}^{(j_i)} - \boldsymbol{\Phi}^\dagger(\boldsymbol{y}^{(j_i)})\|. \tag{45}$$

We can think of the average l^2 error as an approximation of the L^1-norm of the Euclidean error over the domain. This measure will appropriately assess the typical performance of the algorithms over the entire dataset.

The following examples demonstrate the rapid convergence of the cubic RBF inverse mapping method. We observe convergence rates substantially faster than the $\mathcal{O}(h_{Z,\Omega}^{3/2})$ bound [11]. We expect increased orders of algebraic convergence provided additional regularity of the target function beyond the minimum requirements of the native space (see theorems in [27, 28, 29]). We begin with a very simple example and proceed to evaluate the performance of the algorithm on problems with increasing complexity.

7.1. Unit Circle

In this example, n points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ are evenly distributed on the unit sphere S^1 (the unit circle in $\mathbb{R}^2 = \mathbb{R}^D$). Each point is separated from its two nearest neighbors by a distance of exactly $2\sin(\frac{\pi}{N})$. One

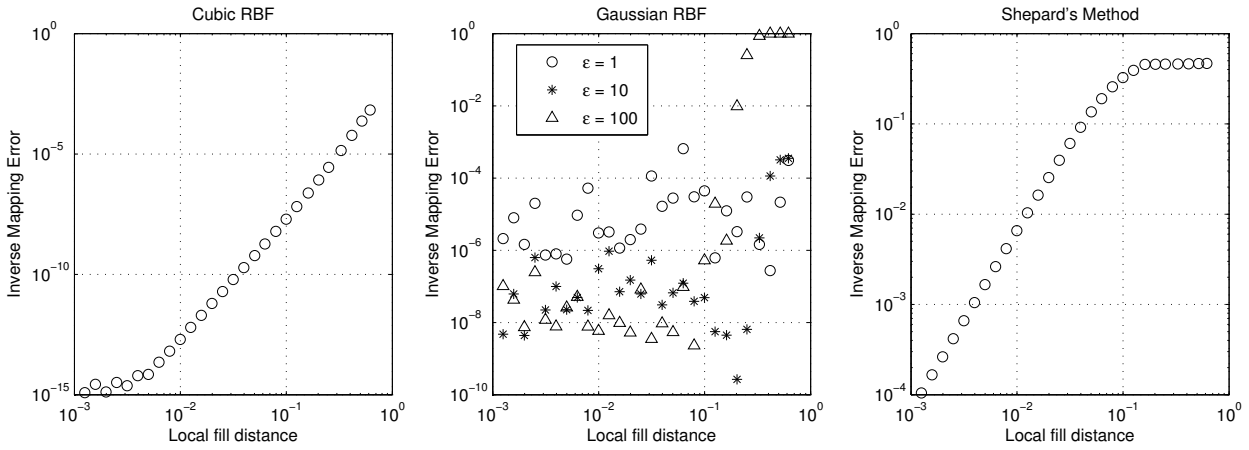

Figure 7: Performance of cubic RBF (left), Gaussian RBF (center), and Shepard's method [6] (right) inverse mapping algorithm on unit circle in $\mathbb{R}^2$. Node spacing is measured by local fill distance, $\bar{h}_{local}$ (18). Error is average l^2 error, E_{avg} (45). Note difference in range of y-axis.

additional point $\boldsymbol{x}$ is placed on the unit circle half-way between $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. The data are mapped to $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}, \boldsymbol{y}\} \subset \mathbb{R}^d = \mathbb{R}^2$ using the first two non-trivial eigenvectors of the graph Laplacian. The inverse mapping algorithms attempt to accurately re-generate the point $\boldsymbol{x} \in \mathbb{R}^D$ from $\boldsymbol{y} \in \mathbb{R}^d$, by interpolation of the data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \subset \mathbb{R}^D$ as a function of their coordinates $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}\} \subset \mathbb{R}^d$.

The performance of the numerical inverse mapping methods versus local fill distance is shown in figure 7. The convergence of cubic RBF inverse mapping is rapid, and appears to scale approximately with $O(\bar{h}^5_{local})$. Interpolation error decreases with fill distance to approximately 10^{-15} when machine precision prevents further improvement in the interpolation. The algorithm captures curvature extremely well, even with sparse data: with only 10 interpolation nodes on the unit circle (the largest fill distance shown in figure 7) the interpolation error is approximately 10^{-3}.

For comparison, the same results are shown also in figure 7 for the Gaussian RBF inverse mapping algorithm as well as Shepard's method [6]. The Gaussian RBF interpolant initially shows rapid convergence, down to the range of 10^{-6} to 10^{-9} depending on choice of ϵ, before ill-conditioning of the interpolation matrix leads to propagation of errors. Shepard's method [6] converges more slowly, and appears to scale approximately with $O(\bar{h}^2_{local})$. In all cases, the inverse mapping method was implemented locally, with the number of nearest neighbors used in the mapping set to either 40 or the total number of data points, whichever is fewer. Shepard's method [6] shows clear evidence of the transition from a global to a local implementation of the algorithm (only demonstrating convergence following the transition to the local regime). The convergence of the cubic RBF algorithm shows no such evidence, suggesting that the local and global implementations are equally effective.

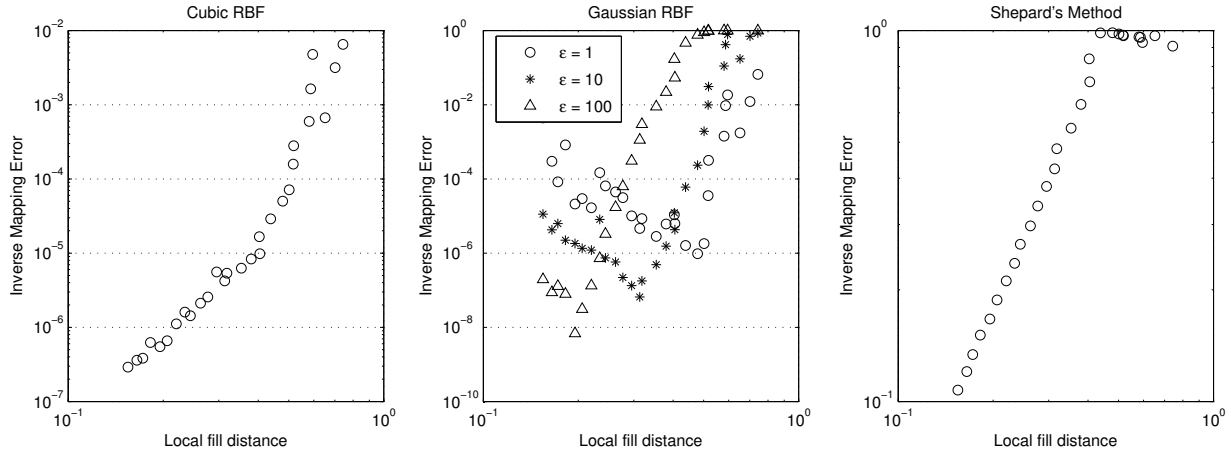

Figure 8: Performance of cubic RBF (left), Gaussian RBF (center), and Shepard's method [6] (right) inverse mapping algorithm on S^4 embedded in $\mathbb{R}^{10}$. Node spacing is measured by local fill distance, $\bar{h}_{local}$ (18). Error is average l^2 error, E_{avg} (45). Note difference in range of y-axis.

7.2. Unit Sphere in $\mathbb{R}^D$

In this example, n points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ are randomly distributed on the unit sphere S^4, then embedded in $\mathbb{R}^{10}$ via a random unitary transformation. The data are mapped to $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}\} \subset \mathbb{R}^d = \mathbb{R}^5$ using the first five non-trivial eigenvectors of the graph Laplacian. The inverse mapping algorithms attempt to accurately re-generate one point $\boldsymbol{x}^{(j)} \in \mathbb{R}^D$ from $\boldsymbol{y}^{(j)} \in \mathbb{R}^d$, by interpolation of the data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(j-1)}, \boldsymbol{x}^{(j+1)}, \ldots, \boldsymbol{x}^{(n)}\} \subset \mathbb{R}^D$ as a function of their coordinates $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(j-1)}, \boldsymbol{y}^{(j+1)}, \ldots, \boldsymbol{y}^{(n)}\} \subset \mathbb{R}^d$. The performance of the cubic RBF, Gaussian RBF, and Shepard's method [6] versus local fill distance is shown in figure 8.

The minimum of the total number of neighbors, and 100 neighbors was used to provide better local coverage in higher dimension. Also, at each fill level, the inverse mapping was repeated on a random subset of the nodes and the errors were averaged to provide a better estimate of average inverse mapping error. In figure 8 we observe that the convergence of cubic inverse mapping is the fastest, and appears to scale approximately with $O(\bar{h}_{local}^5)$. Shepard's method [6] again only convergences in the local regime.

7.3. Handwritten Zeros Dataset

In addition to the previous "artificial" test examples, performance of the inverse mapping algorithm was assessed on a "naturally occurring" high-dimensional data set: a set of digital images of handwritten zeros. The data set consisting of 1000 handwritten zeros was derived from the MNIST database [30]. The images were centered and size-normalized in a fixed size image. Each image in the database is a 28×28 gray scale image. These images were resized to 14×14 pixels, reshaped into a vector, and normalized to have Euclidean norm 1, providing a dataset of 1000 points in $\mathbb{R}^{196}$.

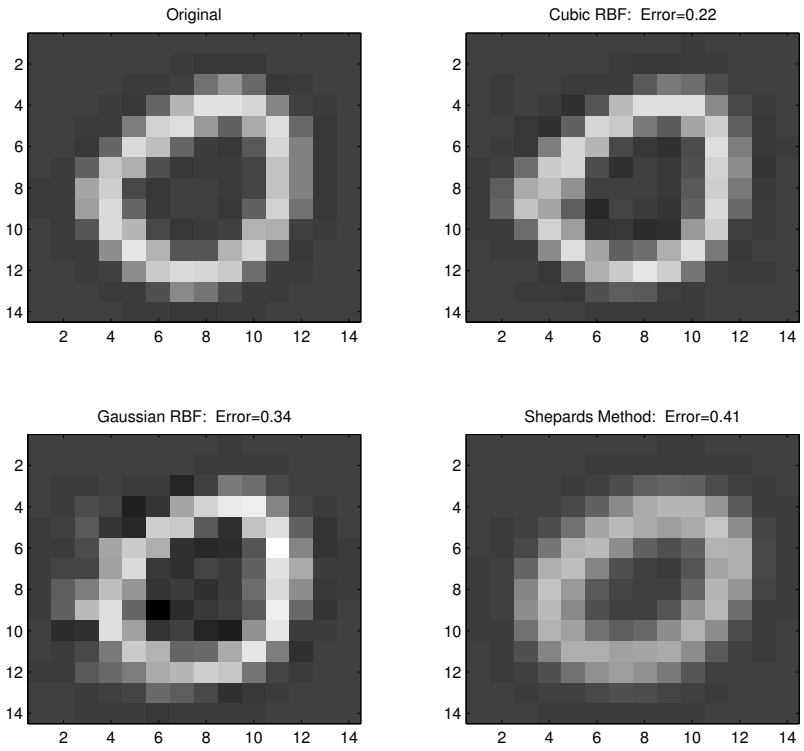

Figure 9: Top row: original image to be reconstructed from the low-dimensional embedding (left), and reconstruction by cubic RBF (right). Bottom row: reconstruction by Gaussian RBF (left), and Shepard's method [6] (right).

A 10-dimensional representation of the 1000 handwritten zeros was generated via Laplacian Eigenmaps. Then the inverse mapping techniques were tested on all images in the set, and compared to the original. Figure 9 shows a representative reconstruction via the various inverse mapping techniques. Figure 10 shows the histogram of errors for the three methods. Shepard's method [6] performs the most poorly, typically generating a very blurry reconstruction of the image, as a simple linear combination of its neighbors. The Gaussian RBF method reduces reconstruction error relative to Shepard's method, but tends to inaccurately reproduce certain pixel values. The cubic RBF performs the best, producing a crisp reconstruction of the image, with the lowest reconstruction error.

8. Discussion

In the previous section we demonstrated excellent performance of a numerical inverse mapping technique involving interpolation using the cubic RBF. We now provide a novel interpretation of the Nyström Method, a scheme commonly used to interpolate the eigenvectors of the weight matrix W. As we will see, the Nyström

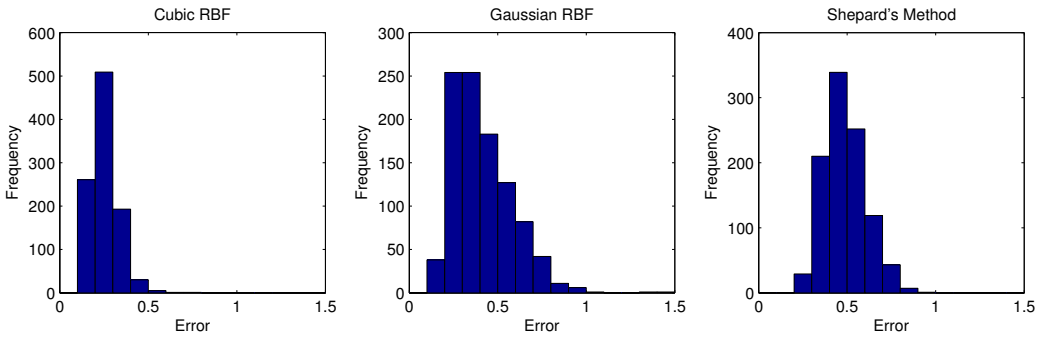

Figure 10: Histogram of reconstruction error for all inverse mapping algorithms on the handwritten zeros dataset. Error is l^2, treating images as vectors in $\mathbb{R}^{196}$.

Method directly generates an RBF interpolant of the eigenvectors of the weight matrix. This interpretation provides new insight into the limitations and potential pitfalls of the Nyström Method.

8.1. Revisiting Nyström

The Nyström method was developed as a technique for numerical approximation of integral eigenfunction problems of the form [8]

$$\int_a^b k(x,y)\phi_l(y)dy = \lambda_l\phi_l(x). \tag{46}$$

If we sample the function at a set of evenly-spaced sample points $x^{(i)}$ for $i = 1,\ldots,n$, we can approximate the integral with a simple quadrature rule,

$$\frac{b-a}{n}\sum_{j=1}^{n} k(x^{(i)},x^{(j)})\phi_l(x^{(j)}) = \lambda_l\phi_l(x^{(i)}). \tag{47}$$

This is equivalent to the matrix eigenvalue problem (absorbing the constant into λ_l),

$$W\boldsymbol{\phi}_l = \lambda_l\boldsymbol{\phi}_l, \tag{48}$$

where $W_{ij} = k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})$ and $\boldsymbol{\phi}_l$ and λ_l are the eigenvectors and corresponding eigenvalues of W for $i = 1,\ldots,n$. Here we generalize the problem to $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$. For $\Lambda = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$ and $\Phi = [\boldsymbol{\phi}_1,\ldots,\boldsymbol{\phi}_n] \in \mathbb{R}^{n\times n}$, we consider the full eigenvector decomposition of the weight matrix W:

$$W = \Phi\Lambda\Phi^T. \tag{49}$$

In the Nyström context, we again define the mapping $\phi_l : \mathbb{R}^D \to \mathbb{R}$ as follows: $\phi_l(\boldsymbol{x}^{(i)}) = \phi_{il}$, where ϕ_{il} is the i,l entry of the eigenvector matrix $\Phi \in \mathbb{R}^{n\times n}$.

Provided that $\lambda_l \neq 0$, the Nyström Extension provides a technique to extend the eigenvector $\boldsymbol{\phi}_l$, defined over a set of sample points, to an arbitrary new point $\boldsymbol{x}$ as [31]

$$\phi_l(\boldsymbol{x}) = \frac{1}{\lambda_l} \sum_{j=1}^{n} k(\boldsymbol{x}, \boldsymbol{x}^{(j)}) \phi_l(\boldsymbol{x}^{(j)}). \tag{50}$$

Perhaps the most common choice for the kernel function k is the Gaussian: $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \exp(-\epsilon^2 \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2)$. In this case, the affinity matrix W is guaranteed non-singular under only the simple condition of non-coincident nodes in $\mathbb{R}^D$ [9] (thus, $\lambda_l \neq 0$ for $l = 1, \ldots, n$). Additionally, since W is a real symmetric matrix, when normalized, $\{\boldsymbol{\phi}_l\}_{l=1}^n$ form an orthonormal basis for $\mathbb{R}^n$ (i.e. $\Phi^T \Phi = I_n$, the identity matrix on $\mathbb{R}^n$).

We now proceed by re-writing $\phi_l(\boldsymbol{x})$ from (50), using the notation $\boldsymbol{k}(\boldsymbol{x}, \cdot) = [k(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, k(\boldsymbol{x}, \boldsymbol{x}^{(n)})]^T$,

$$\begin{aligned} \phi_l(\boldsymbol{x}) &= \lambda_l^{-1} \boldsymbol{k}(\boldsymbol{x}, \cdot)^T \boldsymbol{\phi}_l \\ &= \boldsymbol{k}(\boldsymbol{x}, \cdot)^T \boldsymbol{\Phi} [0 \ldots \lambda_l^{-1} \ldots 0]^T \\ &= \boldsymbol{k}(\boldsymbol{x}, \cdot)^T \boldsymbol{\Phi} \Lambda^{-1} [0 \ldots 1 \ldots 0]^T \\ &= \boldsymbol{k}(\boldsymbol{x}, \cdot)^T \boldsymbol{\Phi} \Lambda^{-1} \boldsymbol{\Phi}^T \boldsymbol{\phi}_l \\ &= \boldsymbol{k}(\boldsymbol{x}, \cdot)^T W^{-1} \boldsymbol{\phi}_l. \end{aligned} \tag{51}$$

If we compare the last line to (14), we observe that in the case of a Gaussian kernel matrix W, the Nyström extension generates a radial basis function interpolation of the eigenvectors of W. This interpretation provides insight into some potential pitfalls of the Nyström Method.

The first important observation about the Nyström interpolation scheme, is the sensitivity to the scale parameter in the case of the typical Gaussian kernel weight matrix. If the scale parameter ϵ is too large, the basis functions will be too localized and provide a poor interpolant. On the other hand, if the scale parameter is too small, the weight matrix W will be very poorly conditioned, and numerical errors may distort the interpolant. A great deal of research has focused on methods to select an appropriate scale parameter (e.g. [14, 32]).

The second observation involves the dangers of sparsifying the weight matrix. In many nonlinear dimensionality reduction applications, it is typical to sparsify the kernel matrix by setting many entries in the weight matrix to zero. Two strategies are typical: thresholding the matrix, and the k-nearest neighbor approach. Thresholding is carried out as follows,

$$W_{ij} = \begin{cases} \exp(-\epsilon^2 \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2) & \text{if} \quad \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\| < \delta, \\ 0 & \text{otherwise.} \end{cases} \tag{52}$$

The k-nearest neighbor approach is as follows,

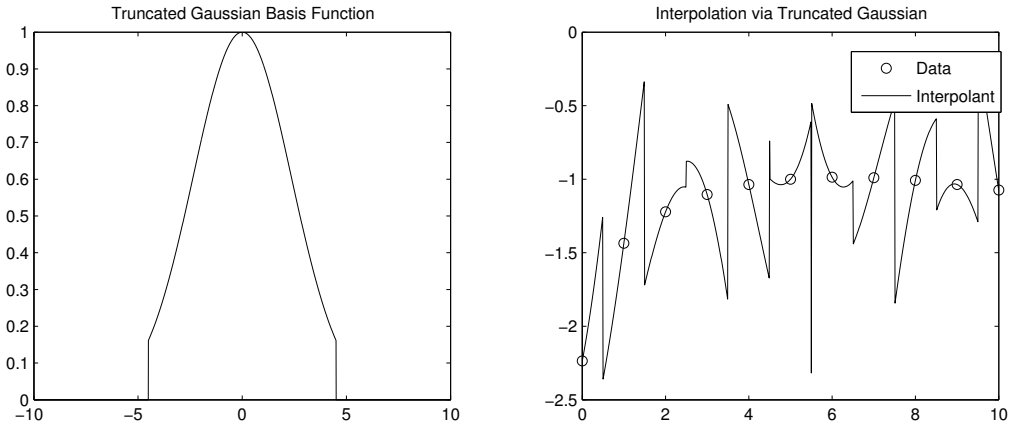

Figure 11: Example RBF interpolation using a truncated Gaussian as basis function.

$$W_{ij} = \begin{cases} \exp(-\epsilon^2 \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2) & \text{if} \quad \boldsymbol{x}^{(j)} \in N_{\boldsymbol{x}^{(i)}}, \\ 0 & \text{otherwise}, \end{cases} \tag{53}$$

where $N_{\boldsymbol{x}^{(i)}}$ denotes the set of k-nearest neighbors of $\boldsymbol{x}^{(i)}$.

If the Nyström extension is applied to a thresholded Gaussian kernel matrix, then the result is an RBF interpolation of the eigenvectors of the submatrix, where the basis function are truncated Gaussians. This is clearly problematic, and will likely generate a very poor (discontinuous) interpolant, as demonstrated in figure 11. In the k-nearest neighbor approach, the Nyström method cannot be interpreted as a consistent RBF interpolant, as the truncation distance varies from node to node. As a result, this method is also likely to perform poorly.

A better alternative to Nyström would simply involve a more intelligent interpolation scheme to extend the eigenvectors to new points in the space. The eigenvectors may be calculated using a (possibly truncated) Gaussian weight matrix over a subset of the nodes. However, these eigenvectors should be extended using a consistent interpolation scheme such as a true (non-truncated) Gaussian RBF, or a cubic RBF interpolant which provides better results as indicated in this paper. Local instead of global implementation of the interpolation algorithm may provide significant computational savings in certain scenarios.

9. Summary and Further Work

A numerical method is proposed to approximate the inverse of a general bi-Lipschitz nonlinear dimensionality reduction mapping, where the forward and consequently the inverse mappings are only explicitly defined on a discrete dataset. An RBF interpolant is used to independently interpolate each component of the high-dimensional representation of the data as a function of its low-dimensional representation. The

scale-free cubic RBF kernel is shown to perform better than the Gaussian kernel, as it does not require the difficult-to-choose scale parameter as an input, and does not suffer from ill-conditioning. Inclusion of additional constant and linear polynomial terms in the cubic RBF basis improves behavior of the interpolant near boundaries, and guarantees non-singularity of the interpolation matrix.

Following exploration of the RBF inverse interpolation scheme, an interpretation of Nyström as an RBF interpolant suggests that this method should not be directly used as an extension scheme when affinities are measured using a truncated Gaussian. Such a scheme will generate a poor (discontinuous) eigenvector interpolation. The present results suggest that reliability of the Nyström method could be improved by using cubic RBF interpolation of eigenvectors instead of the current method which generates the RBF interpolation using the kernel used to build the weight matrix (typically the Gaussian).

The present algorithm generates an exact interpolant to the data. Performance in applications with noisy data will be improved by an approximate interpolation technique. This can be achieved in various ways: for example by using fewer RBF kernel locations than data points, and finding the "best" solution (least squares for example), or by a regularized approach which puts a penalty on the fitting coefficients (e.g. Kernel Ridge Regression). Application of such strategies will be the subject of future work.

Acknowledgments

NDM was supported by National Science Foundation grant DMS 0941476; FGM was partially supported by National Science Foundation grant DMS 0941476, and Department of Energy award DE-SCOO04096.

References

[1] B. Scholkopf, A. Smola, K.-R. Müller, Kernel principal component analysis, in: Advances in Kernel Methods–Support Vector Learning, MIT Press, 1999, pp. 327–352.

[2] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (2003) 1373–1396.

[3] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.

[4] A. Elgammal, C.-S. Lee, Nonlinear manifold learning for dynamic shape and dynamic appearance, Computer Vision and Image Understanding 106 (2007) 31–46.

[5] A. Elgammal, C.-S. Lee, The role of manifold learning in human motion analysis, in: Human Motion, Springer, 2008, pp. 25–56.

[6] D. Kushnir, A. Haddad, R. Coifman, Anisotropic diffusion on sub-manifolds with application to earth structure classification, Applied and Computational Harmonic Analysis 32 (2012) 280–294.

[7] M. Belkin, P. Niyogi, Towards a theoretical foundation for Laplacian-based manifold methods, Journal of Computer and System Sciences 74 (2008) 1289–1308.

[8] E. Nyström, Über die praktische Auflösung von linearen Integralgleichungen mit Anwendungen auf Randwertaufgaben der Potentialtheorie, Commentationes Physico-Mathematicae 14 (1928) 1–52.

[9] G. Fasshauer, Meshfree Approximation Methods With MATLAB, Interdisciplinary Mathematical Sciences, World Scientific, 2007.

[10] M. Buhmann, Radial basis functions, Acta Numerica 9 (2000) 1–38.

[11] H. Wendland, Scattered Data Approximation, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2004.

[12] M. Powell, Radial basis function methods for interpolation to functions of many variables, in: HERCMA, pp. 2–24.

[13] B. Fornberg, J. Zuev, The Runge phenomenon and spatially variable shape parameters in RBF interpolation, Comput. Math. Appl. 54 (2007) 379–398.

[14] A. Bermanis, A. Averbuch, R. Coifman, Multiscale data sampling and function extension, Applied and Computational Harmonic Analysis 34 (2013) 15–29.

[15] R. Devore, A. Ron, Approximation using scattered shifts of a multivariate function, Transactions of the American Mathematical Society 362 (2010) 6205–6229.

[16] T. Hangelbroek, On local RBF approximation, Advances in Computational Mathematics 37 (2012) 285–299.

[17] T. Driscoll, B. Fornberg, Interpolation in the limit of increasingly flat radial basis functions, Comput. Math. Appl. 43 (2002) 413–422.

[18] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (2004) 853–867.

[19] B. Fornberg, C. Piret, A stable algorithm for flat radial basis functions on a sphere, SIAM J. Sci. Comput. 30 (2007) 60–80.

[20] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM Journal on Scientific Computing 33 (2011) 869–892.

[21] B. Fornberg, E. Lehto, C. Powell, Stable calculation of Gaussian-based RBF-FD stencils, Comput. Math. Appl. 65 (2013) 627–637.

[22] B. Fornberg, T. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Comput. Math. Appl. 43 (2002) 473–490.

[23] J. Duchon, Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in: W. Schempp, K. Zeller (Eds.), Constructive Theory of Functions of Several Variables, volume 571 of *Lecture Notes in Mathematics*, Springer Berlin / Heidelberg, 1977, pp. 85–100.

[24] G. Wahba, Spline Models of Observational Data, CBMS-NSF Regional Conference Series in Applied Mathematics, Society of Industrial and Applied Mathematics, 1990.

[25] M. Powell, The theory of radial basis function approximation in 1990, in: W. Light (Ed.), Advances in Numerical Analysis: Volume II: Wavelets, Subdivision Algorithms, and Radial Basis Functions, Oxford University Press, USA, 1992, pp. 105–210.

[26] S. Wild, C. Shoemaker, Global convergence of radial basis function trust region derivative-free algorithms, SIAM Journal on Optimization 21 (2011) 761–781.

[27] T. Gutzmer, J. Melenk, Approximation orders for natural splines in arbitrary dimensions, Mathematics of Computation 70 (2001) 699–703.

[28] R. Schaback, Improved error bounds for scattered data interpolation by radial basis functions, Mathematics of Computation 68 (1999) 201–216.

[29] J. Yoon, L_p-error estimates for shifted surface spline interpolation on Sobolev space, Mathematics of Computation 72 (2003) 1349–1367.

[30] S. Gangaputra, Handwritten digit database, `http://cis.jhu.edu/~sachin/digit/digit.html`, 2012.

[31] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nyström method, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 214–225.

[32] R. Coifman, S. Lafon, Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions,

Applied and Computational Harmonic Analysis 21 (2006) 31–52.