SIGGRAPH 2014 Course Notes

Scattered Data Interpolation for Computer Graphics v1.0

Ken Anjyo
J.P. Lewis
Frédéric Pighin

Abstract. The goal of scattered data interpolation techniques is to construct a (typically smooth) function from a set of unorganized samples. These techniques have a wide range of applications in computer graphics and computer vision. For instance they can be used to model a surface from a set of sparse samples, to reconstruct a BRDF from a set of measurements, or to interpolate motion capture data. This course will survey and compare scattered interpolation algorithms and describe their applications in computer graphics. Although the course is focused on applying these techniques, we will introduce some of the underlying mathematical theory and briefly mention numerical considerations.

Contents

List of Figures

1 Introduction

Many computer graphics and vision problems involve interpolation. In some cases the sample locations are on a regular grid. For instance this is usually the case for image data since the image samples are aligned according to a CCD array. Similarly, the data for a B-spline surface are organized on a regular grid in parameter space. The well known spline interpolation methods in computer graphics address these cases.

In other cases the data locations are unstructured or scattered. Methods for *scattered data interpolation* (or approximation) are less well known in computer graphics, for example, these methods are not yet covered in most graphics text-books. These approaches have become widely known and used in graphics research over the last decade however. This course will attempt to survey most of the known approaches to interpolation and approximation of scattered data.

1.1 Applications

To indicate the versatility of scattered data interpolation techniques, we list a few applications:

- **Surface reconstruction** [13, 40]. Reconstructing a surface from a point cloud often requires an implicit representation of the surface from point data. Scattered data interpolation lends itself well to this representation.
- **Image restoration and inpainting** [62, 43, 35]. Scattered data interpolation can be used to fill missing data. A particular case of this is inpainting, where missing data from an image needs to be reconstructed from available data.
- **Surface deformation** [44, 39, 29, 34, 59, 30, 8, 54]. Motion capture systems allow the recording of sparse motions from deformable objects such as human faces and bodies. Once the data is recorded, it needs to be mapped to a 3-dimensional representation of the tracked object so that the object can be deformed accordingly. One way to deform the object is to treat the problem as a scattered data interpolation problem: the captured data represents a spatially sparse and scattered sampling of the surface that needs to be interpolated to all vertices in a (e.g. facial) mesh.
- **Motion interpolation and inverse kinematics** [47, 25]. Methods such as radial basis functions and extensions to Gaussian processes have used to interpolate motion to compute inverse kinematics based on actual motion data.
- **Meshless/Lagrangian methods for fluid dynamics** [17]. "Meshfree" methods for solving partial differential equations make use of radial basis function interpolation.

- **Appearance representation** [68, 63, 57]. Interpolation of measured reflectance data or user-specified intensity data is a scattered interpolation problem.

Any mathematical document of this size will contain typos.
Please obtain a corrected version of these notes at:
http://scribblethink.org/Courses/ScatteredInterpolation

1.2 The Interpolation Problem

Interpolation is a fundamental problem that has been studied in several different fields. Some of the concepts and issues are:

- Interpolation could be considered as an *inverse problem*, since the solution potentially involves many more degrees of freedom (for example every point on a curve) than the given data (the known points).
- The type of interpolation (linear, cubic, covariance-preserving, etc.) can be considered as a *prior*, thereby making the inverse problem solvable.
- In scattered data interpolation (SDI), the function is required to to perfectly fit the data. For scattered data *approximation* (SDA), we ask that the function merely passes close to the data. The approximation problem allows handling of noisy data. Although SDI and SDA are different problems, some of the same algorithms can be applied to both problems (e.g. section 3.3 and Figure 15).
- When there is noise in the data, the *bias-variance* issue arises. Fitting a high-order polynomial to the data will exactly interpolate the data, but this is not justified in the presence of noise – sampling the same object again (with different noise) could generate wildly different results, so the interpolation may be portraying the noise more than the data. Fitting a line to the data will average out more of the noise (low variance) but introduces an unavoidable bias if the underlying noise-free data are not on a line.
- Another perspective on these issues is in terms of model complexity, which considers the number of model parameters that are justified by the data. Concepts include the minimum description length principle and model complexity measures such as the Bayes information criterion (BIC) [26].
- High dimensional interpolation suffers from a collection of phenomena nicknamed the *curse of dimensionality* [26]. For example, the amount of

data needed to fit a model rises exponentially with dimension. Alternately, in high dimensions, "everything is far away", so the diameter of a kernel must grow to cover an increasing fraction of the diameter of the whole data set.

1.3 Dimensionality

In the remainder of these notes individual techniques will usually be described for the case of mapping from a multidimensional domain to a single field, $\mathbb{R}^p$ to $\mathbb{R}^1$, rather than the general multidimensional input to vector output ($\mathbb{R}^p$ to $\mathbb{R}^q$) case.

Producing a q-dimensional output can be done by running q separate copies of $\mathbb{R}^p$-to-$\mathbb{R}^1$ interpolators. In most techniques some information can be shared across these output dimensions – for example, in radial basis interpolation the system matrix (section 3) can be inverted once and reused with each dimension.

Throughout the notes we attempt to use the following typesetting conventions:

- Scalar values in lower-case: e.g. the particular weight w_k.
- Vector values in bold lower-case: e.g. the weight vector $\mathbf{w}$ or the particular point $\mathbf{x}_k$.
- Matrices in upper-case: e.g. the interpolation matrix $\mathbf{A}$.

2 Scattered Interpolation Algorithms

2.1 Shepard's interpolation

Shepard's Method [56] is probably the simplest scattered interpolation method, and it is frequently re-invented. The interpolated function is

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} \frac{w_k(\mathbf{x})}{\sum_j w_j(\mathbf{x})} f(\mathbf{x}_k),$$

where w_i is the weight function at site i:

$$w_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|^{-p},$$

the power p is a positive real number, and $\|\mathbf{x}-\mathbf{x}_i\|$ denotes the distance between the query point $\mathbf{x}$ and data point $\mathbf{x}_i$. Since p is positive, the weight functions decrease as the distance to the sample sites increase. p can be used to control the shape of the approximation. Greater values of p assign greater influence to values closest to the interpolated point. Because of the form of the weight function this technique is sometime referred as inverse distance weighting. Notice that:

- For $0 < p \le 1$, $\tilde{f}$ has sharp peaks.
- For $p > 1$, $\tilde{f}$ is smooth at the interpolated points, however its derivative is zero at the data points (Fig. 2), resulting in evident "flat spots".

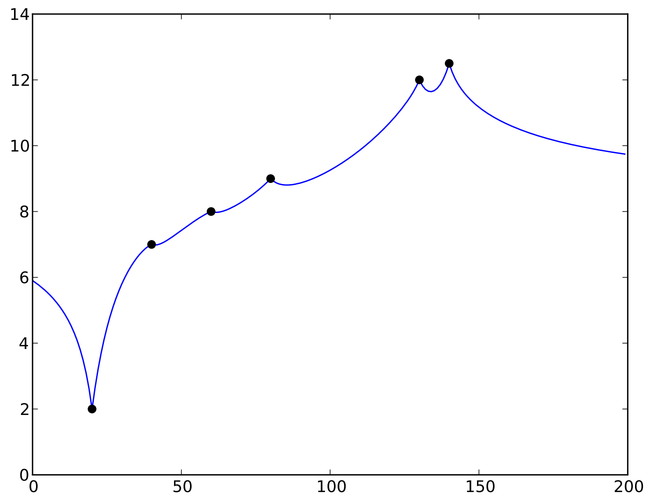


Figure 1: Shepard's interpolation with $p = 1$.

Shepard's method is not an ideal interpolator however, as can be clearly seen from Figs. 1,2.

The *Modified Shepard's Method* [40, 21] aims at reducing the impact of far away samples. This might be a good thing to do for a couple of reasons. First, we might want to determine the local shape of the approximation only using nearby samples. Second, using all the samples to evaluate the approximation does not scale well with the number of samples. The modified Shepard's method computes interpolated values only using samples within a sphere of radius r. It uses the weight function:

$$w_j(\mathbf{x}) = \left[\frac{r - d(\mathbf{x}, \mathbf{x}_i)}{r d(\mathbf{x}, \mathbf{x}_i)} \right]^2 .$$

where $d()$ notates the distance between points. Combined with a spatial data structure such as a k-d tree or a Voronoi diagram [42] this technique can be used on large data sets.

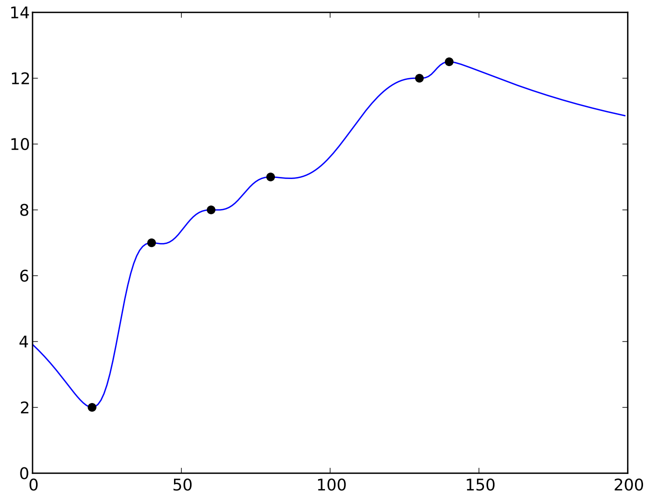


Figure 2: Shepard's interpolation with $p = 2$. Note the the derivative of the function is zero at the data points, resulting in smooth but uneven interpolation. Note that this same set of points will be tested with other interpolation methods; compare Figs. 8, 9, 13, etc.

2.2 Kernel regression

A straightforward generalization of Shepard's interpolation is kernel regression (also called Nadaraya-Watson regression) [10], which generalizes the weight function to an arbitrary "kernel function" $K(\mathbf{x}, \mathbf{x}_i)$:

$$\tilde{f}(\mathbf{x}) = \frac{\sum_k^n K(\mathbf{x}, \mathbf{x}_k) f(\mathbf{x}_k)}{\sum_i^n K(\mathbf{x}, \mathbf{x}_i)}$$

In kernel regression K typically does not go to infinity at the origin, and thus the regression approximates rather than interpolating.

2.3 Moving least-squares

Moving least-squares builds an approximation by using a local polynomial function. The approximation is set to locally belong in Π_m^p the set of polynomials with total degree m in p dimensions. At each point, $\mathbf{x}$, we would like the polynomial approximation to best fit the data in a weighted least-squares fashion, i.e.:

$$\tilde{f}(\mathbf{x}) = \underset{g \in \Pi_m^p}{\arg\min} \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(g(\mathbf{x}_i) - f_i)^2,$$

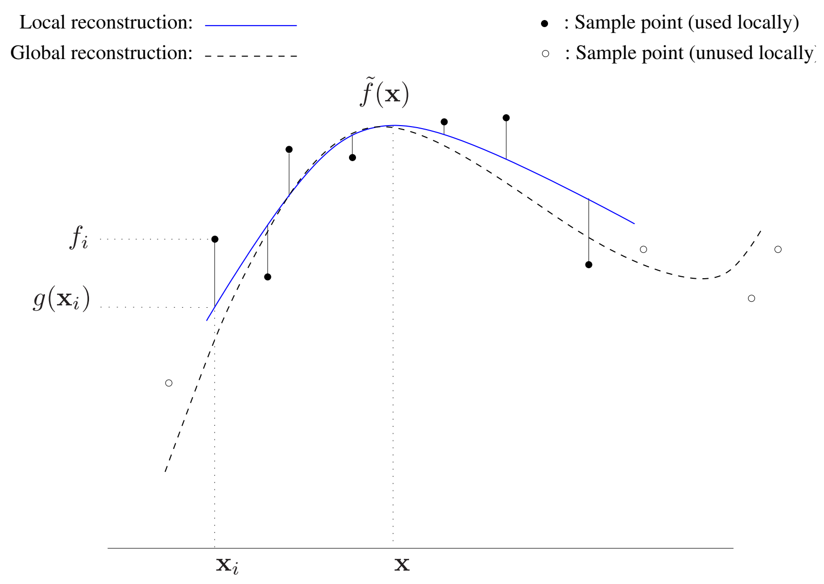


Figure 3: Moving least-squares in 1-D. The function is locally reconstructed using a polynomial of degree 2. The local approximating polynomial is refitted for each evaluation of the reconstructed function.

where w_i is a weighting function used to emphasize the contribution of nearby samples, for instance $w_i(d) = e^{-\frac{d^2}{\sigma^2}}$. Note that this choice of weights will only approximate the data. To interpolate it is necessary to use weights that go to infinity as the distance to a data point goes to zero (as is the case with Shepard interpolation), e.g. $w_i(d) = 1/d$.

Figure 3 illustrates the technique with a 1-dimensional data set reconstructed with a moving second order polynomial.

Using a basis of Π_m^p, $\boldsymbol{b}(\mathbf{x}) = \{b_1(\mathbf{x}), \ldots, b_l(\mathbf{x})\}$ we can express the polynomial g as a linear combination in that basis: $g(\mathbf{x}) = \boldsymbol{b}^t(\mathbf{x})\mathbf{c}$, where $\mathbf{c}$ is a vector of coefficients. If we then call, $\mathbf{a}$, the expansion of $\tilde{f}$ in the basis $\boldsymbol{b}$, we can then write:

$$\tilde{f}(\mathbf{x}) = \boldsymbol{b}^t(\mathbf{x})\mathbf{a}(\mathbf{x})$$

with

$$\mathbf{a}(\mathbf{x}) = \arg\min_{\mathbf{c}\in\mathbb{R}^l} \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{c} - f_i)^2.$$

Computing, $\mathbf{a}(\mathbf{x})$, is a linear least-squares problem that depends on $\mathbf{x}$. We can extract the system by differentiating the expression above with respect to $\mathbf{c}$ and

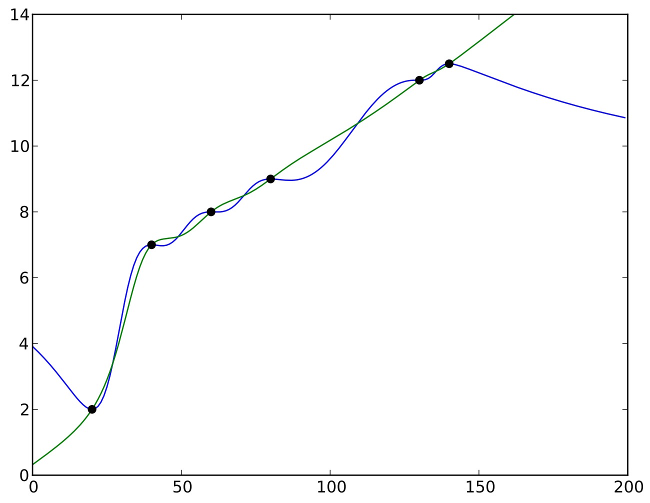


Figure 4: Comparison of Moving Least Squares interpolation using zero-order (constant) and first-order (linear) polynomials. With weights set to a power of the inverse distance, the former reduces to Shepard's interpolation (c.f. Fig. 2).

setting it to 0:

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{c}} \left(\sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{c} - f_i)^2 \right) \Bigg|_{\mathbf{a}} = 0 \\ \Leftrightarrow \quad & \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{a} - f_i) = 0 \\ \Leftrightarrow \quad & \left(\sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i) \right) \mathbf{a} = \sum_i^N f_i w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i). \end{aligned}$$

This last equation can be written in matrix form $\boldsymbol{A}\mathbf{a} = \mathbf{d}$ with:

$$\boldsymbol{A} = \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i)$$

and

$$\mathbf{d} = \sum_i^N f_i w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i).$$

Note that the matrix $\boldsymbol{A}$ is square and symmetric. In the usual case, where $\mathbf{w}$ is

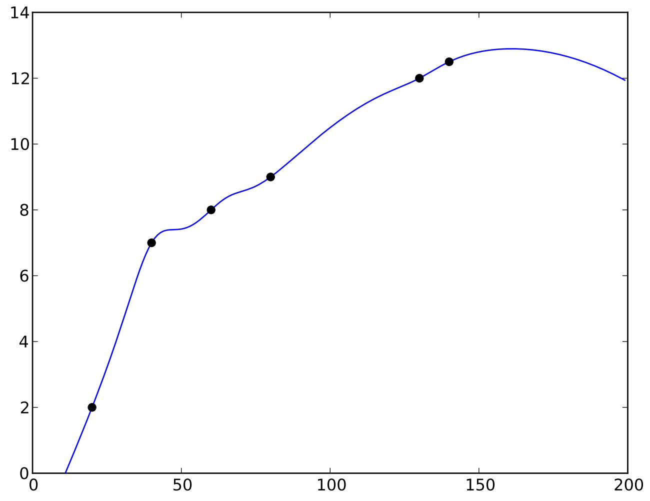


Figure 5: One-dimensional MLS interpolation with second-order polynomials. The interpolation is not entirely satisfactory.

non-negative $\boldsymbol{A}$ is also symmetric and positive semi-definite.

$$\begin{aligned}
\mathbf{x}^t \boldsymbol{A}\mathbf{x} &= \mathbf{x}^t \left(\sum_i^N w_i(\|\mathbf{x}-\mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i) \right) \mathbf{x} \\
&= \sum_i^N w_i(\|\mathbf{x}-\mathbf{x}_i\|)\mathbf{x}^t\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{x}) \\
&= \sum_i^N w_i(\|\mathbf{x}-\mathbf{x}_i\|)(\mathbf{x}^t\boldsymbol{b}(\mathbf{x}_i))^2 \geq 0.
\end{aligned}$$

If the matrix $\boldsymbol{A}$ has full rank the system can be solved using the Cholesky decomposition.

It would seem that the computational cost of moving least-squares is excessive since it requires solving a linear system for each evaluation of the reconstructed function. If the weight functions fall quickly to zero, however, then the size of the system can involve only a few data points. In this case MLS is solving many small linear systems rather than a single large one (versus the case with Radial Basis Functions (section 3), where the system size is usually the number of data points). On the other hand, the resulting interpolation is not ideal (Fig. 5).

Note that if the weight functions are an inverse power of the distance, as with the weights in Shepard's interpolation, and a piecewise constant (zero order polynomial) is chosen for $\boldsymbol{b}(\mathbf{x})$ then MLS reduces to Shepard's interpolation:

$$\begin{aligned}
&\min_a \quad \sum_k^n w_k^{(x)}(a \cdot 1 - f_k)^2 \\
&\frac{d}{da}\left[\sum_k^n w_k^{(x)}(a^2 - 2af_k + f_k^2)\right] = 0 \\
&\frac{d}{da}\left[\sum_k^n w_k a^2 - 2w_k a f_k + w_k f_k^2\right] \\
&= \sum_k^n 2w_k a - 2w_k f_k = 0 \\
&a = \frac{\sum_k^n w_k f_k}{\sum_k^n w_k}
\end{aligned}$$

and

$$\hat{f}(x) = a \cdot 1$$

2.3.1 Applications

Surface reconstruction Cheng *et al.* [14] give an overview of moving least-squares for surface reconstruction.

Fleishman *et al.* [20] uses moving least-squares to reconstruct piecewise smooth surfaces from noisy point clouds. They introduce robustness in their algorithm in multiple ways. One of them is an interesting variant on the moving least squares estimation procedure. They estimate the parameters, β, of their model, f_β, using a robust fitting function:

$$\beta = \arg\min_\beta \operatorname*{median}_i \|f_\beta(x_i) - y_i\|.$$

The sum in the original formulation has been replaced by a more robust median operator.

Image warping Schaefer *et al.* [50] introduces a novel image deformation technique using moving least-squares. Their approach is to solve for the best transformation, $l_v(x)$, at each point, v, in the image by minimizing:

$$\sum_i w_i \|l_v(p_i) - q_i\|$$

where $\{p_i\}$ is a set of control points and $\{q_i\}$ an associated set of displacements. By choosing different transform types for l_v (affine, rigid, ..), they create different effects.

Surface deformation [67] produce an as-rigid-as-possible surface transformation by choosing the local function as a similarity transformation (rotation and uniform scaling) rather than a polynomial.

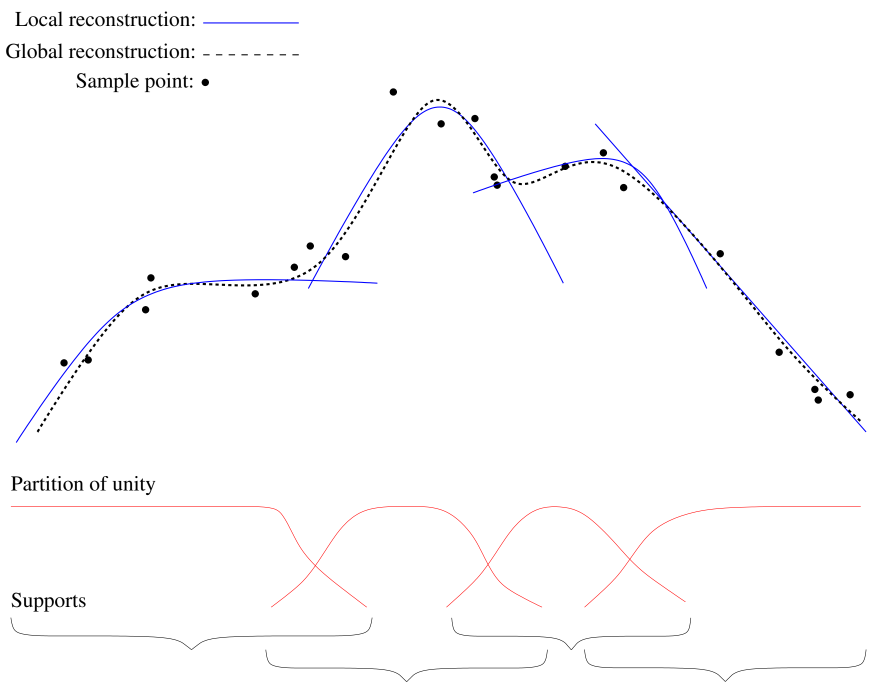


Figure 6: Partition of unity in 1-D. The partition of unity are a set of weighing functions that sum to 1 over a domain. These are used to blend a set of (e.g. quadratic) functions fit to local subsets of the data to create a global reconstruction.

2.4 Partition of unity

The Shephard's and MLS methods are specific examples of a general *partition of unity* framework for interpolation. The underlying principal of a partition of unity method is that it is usually easier to approximate the data locally than to find a global approximation that fits it all. With partition of unity, the construction of the global approximation is done by blending the local approximations using a set of weight functions $\{\phi_k\}$ compactly supported over a domain Ω such that:

$$\sum_k \phi_k = 1 \text{ on } \Omega.$$

We say that the functions $\{\phi_k\}$ form a partition of unity. Now we consider a set of approximations of f, $\{q_k\}$, where q_k is defined over the support of ϕ_k, then we can compute a global approximation as:

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} \phi_k(\mathbf{x}) q_k(\mathbf{x}).$$

In this description the functions, ϕ_k, only need to be non-negative.

To interpret Shepard's method as a partition of unity approach, define

$$\phi_k(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{x}_k\|^p}{\sum_j \|\mathbf{x} - \mathbf{x}_j\|^p}$$

and q_k as the constant function defined by the k-th data point.

2.5 Natural neighbor interpolation

Natural Neighbor Interpolation was developed by Robin Sibson [58]. It is similar to Shepard's interpolation in the sense that the approximation is written as a weighted average of the sampled values. It differs in that the weights are volume-based as opposed to the distance-based weights of Shepard's method.

Natural neighbor techniques uses a Voronoi diagram of the sampled sites to formalize the notion of "neighbor": two sites are neighbors if they share a common boundary in the Voronoi diagram. Using duality, this is equivalent to writing that the sites form an edge in the Delaunay triangulation. By introducing the evaluation point $\mathbf{x}$ in the Delaunay triangulation, the natural neighbors of $\mathbf{x}$ are the nodes that are connected to it. The approximation is then written as:

$$\tilde{f}(\mathbf{x}) = \sum_{k \in N} \alpha_k(\mathbf{x}) f(\mathbf{x}_k),$$

where N is the set of indices associated with the natural neighbors of $\mathbf{x}$ and $\alpha_k(\mathbf{x})$ are weight functions.

$$\alpha_k(\mathbf{x}) = \frac{u_k}{\sum_{j \in N} u_j}$$

where u_j is the volume of the intersection of the node associated with the evaluation point and the node associated with the j-th neighbor in the original Voronoi diagram.

2.5.1 Applications

Surface reconstruction In [11] Boisonnat and Cazals represent surfaces implicitly as the zero-crossings of a signed pseudo-distance function. The function is set to zero at the sampled points and is interpolated to the whole 3D space.

Figure 7: Non-fractal landscapes invented with Wiener interpolation [33]

2.6 Linear interpolation on simplices

A particularly simple approach to scattered interpolation is to form the Delaunay triangulation or its n-dimensional generalization and then interpolate independently and linearly within each simplex (each triangle in the 2D case). Because high-dimensional functions are difficult to visualize and control, [6] introduced a variant of pose space deformation that uses this approach. ([31] may have also used this idea, though it is difficult to say from the one-page description). This approach sacrifices smoothness but prevents overshoot; it was the recommended choice in the evaluation [31]. Further details on interpolation over simplices is given in [15], albeit in a non-graphics context.

2.7 Wiener interpolation and Gaussian Processes

"Wiener interpolation" is a very old technique, evidently invented independently by Wiener and Kolmogorov during the 1940s. It was rediscovered in the 1960s in the geostatistics community where it is known as Kriging, and came to attention again in machine learning in the late 1990s, where the technique is called Gaussian processes. Wiener interpolation differs from polynomial interpolation approaches in that it is based on the expected correlation of the data. Wiener interpolation of discrete data is simple, requiring only the solution of a matrix equation. This section describes two derivations for discrete Wiener interpolation.

Some advantages of Wiener interpolation are:

- The data can be arbitrarily spaced.
- The algorithm applies without modification to multi-dimensional data.
- The interpolation can be made local or global to the extent desired. This is achieved by adjusting the correlation function so that points beyond a desired distance have a negligible correlation.

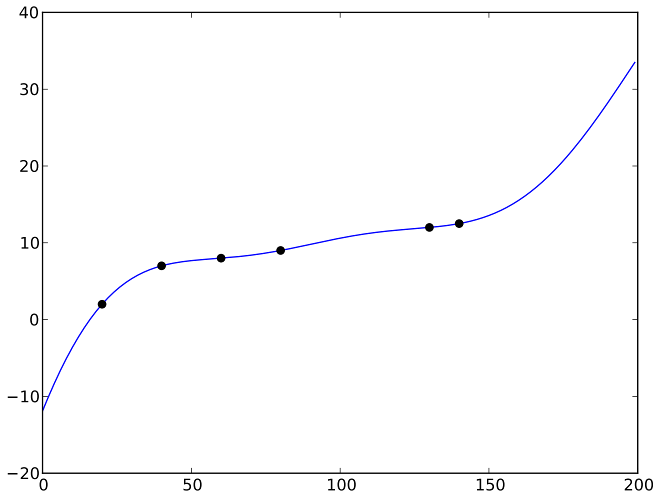


Figure 8: One-dimensional Wiener interpolation, using a Gaussian covariance matrix.

- The interpolation can be as smooth as desired, for example an analytic correlation function will result in an analytic interpolated curve or surface.
- The interpolation can be shaped and need not be "smooth", for example, the correlation can be negative at certain distances, oscillatory, or (in several dimensions) have directional preferences.
- The algorithm provides an error or confidence level associated with each point on the interpolated surface.
- The algorithm is optimal by a particular criterion (below) which may or may not be relevant.

Some disadvantages of Wiener interpolation:

- It requires knowing or inventing the correlation function. While this may arise naturally from the problem in some cases, in other cases it would require interactive access to the parameters of some predefined correlation models to be "natural".
- It requires inverting a matrix whose size is the number of significantly correlated data points. This can be a practical problem if a large neighborhood is used. A further difficulty arises if the chosen covariance is broad, causing the resulting covariance matrix (see below) to have similar rows and hence be nearly singular. Sophistication with numerical linear algebra will be required in this case.

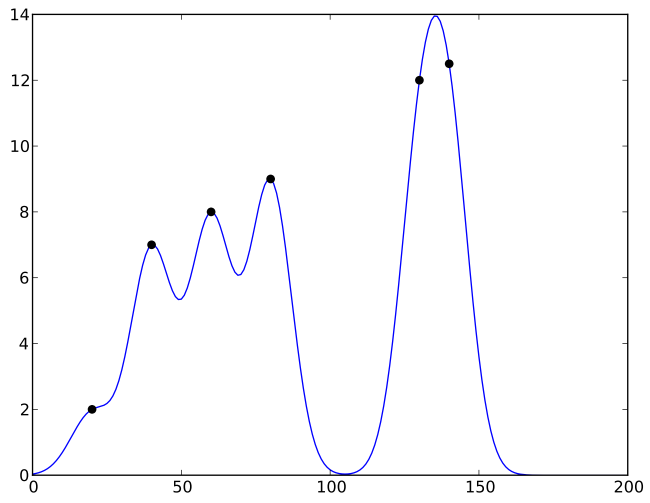


Figure 9: Similar to Fig. 8, but the variance of the Gaussian is too narrow for this data set.

Terminology

Symbols used below:

f value of a stochastic process at time x

$\hat{f}$ estimate of f

f_j observed values of the process at times or locations x_j

The derivations require two concepts from probability:

- The correlation of two values is the expectation of their product, $\mathbf{E}[xy]$. The autocorrelation or autocovariance function is the correlation of pairs of points from a process:

$$C(x_1, x_2) = \mathbf{E}\left\{f(x_1)f(x_2)\right\}$$

  For a stationary process this expectation is a function only of the distance between the two points: $C(\tau) = \mathbf{E}[f(x)f(x+\tau)]$. The variance is the value of the autocorrelation function at zero: $\mathrm{var}(x) = C(0)$. (Auto)covariance usually refers to the correlation of a process whose mean is removed and (usually) whose variance is normalized to be one. There are differences in the terminology, so "Correlation function" will mean the autocovariance function of a normalized process here.

- Expectation behaves as a linear operator, so any factor or term which is known can be moved "outside" the expectation. For example, assuming a and b are known,
$$\mathbf{E}\{af+b\} = a\mathbf{E}f + b$$
Also, the order of differentiation and expectation can be interchanged, etc.

Definition

Wiener interpolation estimates the value of the process at a particular location as a weighted sum of the observed values at some number of other locations:

$$\hat{f} = \sum w_j f_j \tag{1}$$

The weights w_j are chosen to minimize the expected squared difference or error between the estimate and the value of the "real" process at the same location:

$$\mathbf{E}\left\{(f-\hat{f})^2\right\} \tag{2}$$

The reference to the "real" process in (2) seems troublesome because the real process may be unknowable at the particular location, but since it is the *expected* error which is minimized, this reference disappears in the solution.

Wiener interpolation is optimal among linear interpolation schemes in that it minimizes the expected squared error (2). When the data have jointly Gaussian probability distributions (and thus are indistinguishable from a realization of a Gaussian stochastic process), Wiener interpolation is also optimal among nonlinear interpolation schemes.

Derivation 1

The first derivation uses the "orthogonality principle": the squared error of a linear estimator is minimum when the error is 'orthogonal' in expectation to all of the known data, with 'orthogonal' meaning that the expectation of the product of the data and the error is zero:

$$\mathbf{E}\left\{(f-\hat{f})f_k\right\} = 0 \text{ for all } j$$

Substituting $\hat{f}$ from (1),

$$\mathbf{E}\left\{(f-\sum w_j f_j)f_k\right\} = 0 \tag{3}$$

$$\mathbf{E}\left\{ff_k-\sum w_j f_j f_k\right\} = 0$$

The expectation of ff_k is the correlation $C(x - x_k)$, and likewise for $f_j f_k$:

$$C(x - x_k) = \sum w_j C(x_j - x_k)$$

or

$$\mathbf{C}\mathbf{w} = \mathbf{c} \tag{4}$$

This is a matrix equation which can be solved for the coefficients w_j. The coefficients depend on the positions of the data f_j though the correlation function, but not on the actual data values; the values appear in the interpolation (1) though. Also, (4) does not directly involve the dimensionality of the data. The only difference for multi-dimensional data is that the correlation is a function of several arguments: $\mathbf{E}[PQ] = C(x_p - x_q, y_p - y_q, z_p - z_q, \ldots)$.

Derivation 2

The second derivation minimizes (2) by differentiating with respect to each w_k. Since (2) is a quadratic form (having no maxima), the identified extreme will be a minimum (intuitively, a squared difference (2) will not have maxima).

$$\begin{aligned}
&\frac{d}{dw_k}\left[\mathbf{E}\left\{(f - \sum w_j f_j)^2\right\}\right] = \mathbf{E}\left[\frac{d}{dw_k}(f - \sum w_j f_j)^2\right] = 0 \\
&2\mathbf{E}\left\{(f - \sum w_j f_j)\frac{d}{dw_k}(f - \sum w_j f_j)\right\} = 0 \\
&\mathbf{E}\left\{(f - \sum w_j f_j) f_k\right\} = 0
\end{aligned}$$

which is (3).

A third approach to deriving the Wiener interpolation proceeds by expressing the variance of the estimator (see below) and finding the weights that produce the minimum variance estimator.

Cost

From (4) and (1), the coefficients w_j are $\mathbf{w} = \mathbf{C}^{-1}\mathbf{c}$, and the estimate is $\hat{f} = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{c}$. The vector $\mathbf{c}$ changes from point to point, but $\mathbf{x}^T\mathbf{C}^{-1}$ is constant for given data, so the per point estimation cost is a dot product of two vectors whose size is the number of data points.

Confidence

The interpolation coefficients w_j were found by minimizing the expected squared error (2). The resulting squared error itself can be used as a confidence measure for the interpolated points. For example, presumably the error variance would be high away from the data points if the data are very uncharacteristic of the chosen correlation function. The error at a particular point is found by expanding (2) and substituting $\mathbf{a} = \mathbf{C}^{-1}\mathbf{c}$:

$$\begin{aligned}
\mathbf{E}\left\{(f-\hat{f})^2\right\} &= \mathbf{E}\left\{(f-\sum w_j f_j)^2\right\} \\
&= \mathbf{E}\left\{f^2 - 2f\sum w_j f_j + \sum w_j f_j \sum w_k f_k\right\} \\
&= \mathrm{var}(x) - 2\sum w_j C(x,x_j) + \sum w_j w_k C(x_j,x_k)
\end{aligned}$$

(switching to matrix notation)

$$\begin{aligned}
&= C(0) - 2\mathbf{w}^T\mathbf{c} + \mathbf{w}^T\mathbf{C}\mathbf{w} \\
&= C(0) - 2(\mathbf{C}^{-1}\mathbf{c})^T\mathbf{c} + (\mathbf{C}^{-1}\mathbf{c})^T\mathbf{C}\mathbf{C}^{-1}\mathbf{c} \\
&= C(0) - \mathbf{c}^T\mathbf{C}^{-1}\mathbf{c} \\
&= C(0) \;-\; \sum w_j C(x,x_j)
\end{aligned}$$

Applications: terrain synthesis, Kriging, Gaussian processes [33] used Wiener interpolation in a hierarchical subdivision scheme to synthesize random terrains at run time (Fig. 7). The covariance was specified, allowing the synthesized landscapes to have arbitrary power spectra (beyond the fractal $1/f^p$ family of power spectra).

3 Radial basis functions

Radial basis functions ("RBFs") are the most versatile and commonly used scattered data interpolation techniques, and the majority of the remainder of the course will focus on them. They are conceptually easy to understand and simple to implement. From a high level point of view, a radial basis functions interpolation works by summing a set of replicates of a single basis function. Each replicate is centered at a data point and scaled to respect the interpolation conditions. This can be written as:

$$\tilde{f}(\mathbf{x}) = \sum_k^N w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|),$$

where ϕ is a function from $[0,\infty[$ to $\mathbb{R}$ and $\{w_k\}$ is a set of weights. It should be clear from this formula why this technique is called "radial": The influence of a single data point is constant on a sphere centered at that point. Without any further information on the structure of the input space, this seems a reasonable assumption. Note also that it is remarkable that the function, ϕ, be univariate: Regardless of the number of dimensions of the input space, we are interested in distances between points.

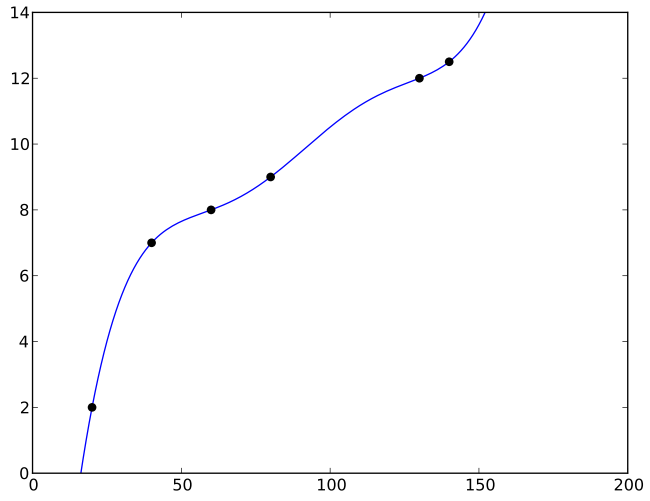


Figure 10: Radial basis interpolation with a Gaussian kernel

The conditions of interpolating the avaliable data can be written as

$$\tilde{f}(\mathbf{x}_i) = \sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) = f_i, \text{ for } 1 \le i \le n.$$

This is a linear system of equations where the unknowns are the vector of weights $\{w_k\}$. To see this, let us call $\phi_{i,k} = \phi(\|\mathbf{x}_i - \mathbf{x}_k\|)$. We can then write the equivalent matrix representation of the interpolation conditions:

$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} & \cdots & \\ \phi_{2,1} & \phi_{2,2} & \cdots & & \\ \phi_{3,1} & \cdots & & & \\ \vdots & & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix}$$

This is a square system with as many equations as unknowns. Thus we can form the radial basis function interpolation by solving this system of equations.

3.1 Radial basis function with polynomial term

In the polyharmonic and thin-plate cases a linear polynomial term is added to the radial basis function expression. Reproducing a polynomial is useful in some applications. For example, thin-plate splines are commonly used for image registration (e.g. [32]), and in that application it would be troubling if an affine

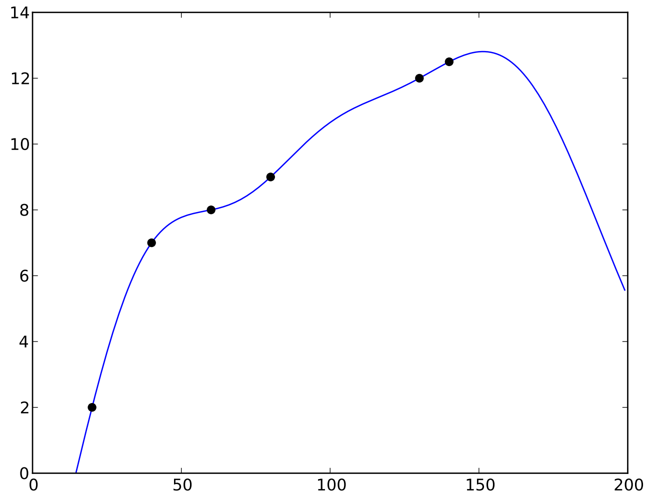


Figure 11: Radial basis interpolation with a Gaussian kernel, varying the kernel width relative to Fig. 10

transformation could not be produced. Also, by regularizing the upper rows of the system matrix, the image warp is forced to become more affine [16].

The addition of a polynomial can be motivated in several ways:

- “Polynomial reproducibility”: it may be useful to have an interpolation that exactly produces some polynomial (e.g. affine) functon if it is known that the data may lie exactly on that function.

- “Filling the null space”: in fact, adding a polynomial is required in cases where the RBF kernel comes from minimizing a roughness, where the roughness is defined as a derivative, squared, integrated over the interpolated function (see section 5.1). The biharmonic and thin-plate kernels are of this form (see section 7.7.2, especially equations (41) and (43)).

  To understand this intuitively, consider the first derivative of a function: the derivative is not changed if a constant is added to the function. Said differently, constant functions are in the null space of the derivative operator. Similarly, for a second derivative, affine functions (linear + constant) are in the null space. Thus, finding a smooth function by minimizing the integrated squared derivative is ambiguous, because one of these functions from the null space can be added without changing the smoothness. The solution is to simultaneously fit the RBF kernels and the polynomial function to the data.

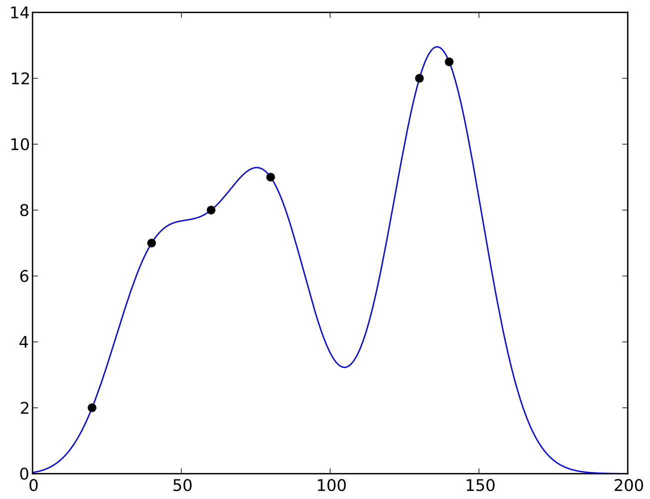


Figure 12: Radial basis interpolation with a Gaussian kernel, varying the kernel width relative to Fig. 10,11. In this figure the kernel width is too narrow to adequately interpolate the data.

As an example, consider RBF estimation in two dimensions with an additional polynomial. A polynomial in one dimension is $a + bx + cx^2 + \cdots$. Truncating after the linear term gives $a + bx$. A similar polynomial in two dimensions is of the form $a + bx + cy$. Adding this to the RBF synthesis equation gives

$$\hat{f}(\mathbf{p}) = \sum_{k=1}^{n} c_k \phi(\|\mathbf{p} - \mathbf{p}_k\|) \; + \; c_{n+1} \cdot 1 \; + \; c_{n+2} \cdot x \; + \; c_{n+3} \cdot y$$

Here $\mathbf{p} = (x, y)$ is an arbitrary 2D point and $\mathbf{p}_k$ are the training points.

The additional polynomial coefficients have to be determined somehow. Since the degrees of freedom in the n points are already used in estimating the RBF coefficients c_k, an additional relation must be found. This takes the form of requiring that the interpolation should exactly reproduce polynomials: If the data $f(x)$ to be interpolated is exactly a polynomial, the polynomial contribution $c_{n+1} \cdot 1 \; + \; c_{n+2} \cdot x \; + \; c_{n+3} \cdot y$ should be exactly that polynomial, and the weights on the RBF kernel $c_k, k <= n$ should be zero.

Doing so results in the additional condition that the weights c_k are in the null space of the polynomial basis. In showing this we use the following notation: $\mathbf{R}$ is the standard RBF system matrix with the RBF kernel evaluated at the distance between pairs of training points, $\mathbf{c}$ are the RBF weights (coefficients), and $\mathbf{f}$ are the desired values to interpolate, so $\mathbf{Rc} = \mathbf{f}$ would be the system to

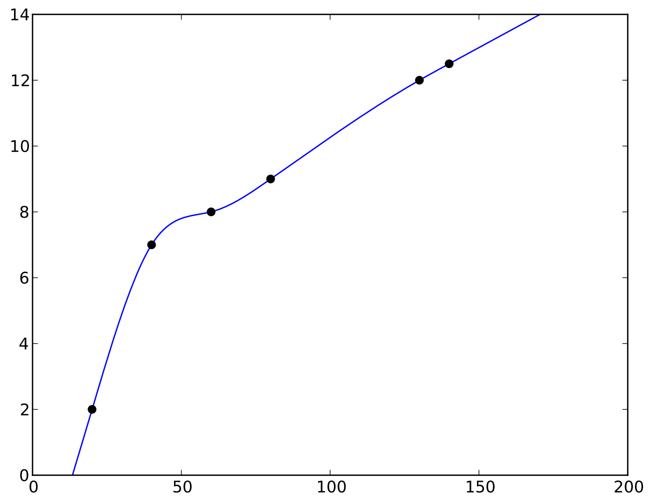


Figure 13: The one-dimensional equivalent of thin-plate spline interpolation is the natural cubic spline, with radial kernel $|r|^3$ in one dimension. This spline minimizes the integrated square of the second derivative (an approximate curvature) and so extrapolates to infinity away from the data.

solve if the extra polynomial is not used. Also let $\mathbf{P}$ be the polynomial basis, and $\mathbf{d}$ be the weights on the polynomial basis. In the 2D case with a linear polynomial $\mathbf{P}$ is a n by 3 matrix with a column of ones, a column of x_k, and a column of y_k, where $(x_k, y_k) = \mathbf{p}_k$ are the training locations. Then,

$$\begin{aligned}
&\mathbf{Rc} + \mathbf{Pd} = \mathbf{f} && \text{the interpolation statement}\\
&\mathbf{Rc} + \mathbf{Pd} = \mathbf{Pm} && \text{fit a polynomial } \mathbf{Pm}\text{, for some unknown coefs } \mathbf{m}\\
&\mathbf{c}^T\mathbf{Rc} + \mathbf{c}^T\mathbf{Pd} = \mathbf{c}^T\mathbf{Pm} && \text{premultiply by } \mathbf{c}^T\\
&\mathbf{c}^T\mathbf{Rc} = \mathbf{c}^T\mathbf{Pm} - \mathbf{c}^T\mathbf{Pd} &&\\
&\mathbf{c}^T\mathbf{P}(\mathbf{m} - \mathbf{d}) = \mathbf{c}^T\mathbf{Rc} &&
\end{aligned}$$

Then if we require that $\mathbf{c}^T\mathbf{P} = \mathbf{0}$, then the left hand side is zero, and so $\mathbf{c}$ must itself be zero since the right hand side is positive. So now going back to the orginal $\mathbf{Rc} + \mathbf{Pd} = \mathbf{Pm}$, we know that if $\mathbf{c}^T\mathbf{P} = 0$, then $\mathbf{c} = 0$, so $\mathbf{Rc} = 0$, so $\mathbf{Pd} = \mathbf{Pm}$.

Restating, this means that if the signal is polynomial and we have required $\mathbf{c}^T\mathbf{P} = \mathbf{0}$, the coefficients $\mathbf{c}$ are zero, and then $\mathbf{Pd} = \mathbf{Pm}$, so $\mathbf{d} = \mathbf{m}$, so the polynomial is exactly reproduced.

Another way of viewing these side conditions is that by imposing an additional relation on $\mathbf{c}$, they reduce the total number of degrees of freedom from $n + p$ (for a p-order polynomial basis) back to n

The additional relation $\mathbf{c}^T\mathbf{P} = \mathbf{0}$ is enough to solve for the RBF and polynomial coefficients as a block matrix system:

$$\begin{bmatrix} \mathbf{R} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}$$

Note that the matrix in this system is symmetric but indefinite, meaning that it has both positive and negative eigenvalues. This prevents the use of some algorithms (such as conjugate gradient).

We showed using linear algebra that the relation $\mathbf{c}^T\mathbf{P} = 0$ is sufficient to reproduce polynomials. In fact, this is called the *vanishing moment condition*, and it is a necessary condition as well. These issues are discussed in more generality and depth in section 7.7 and Theorem 5.

3.2 The choice of RBF kernel

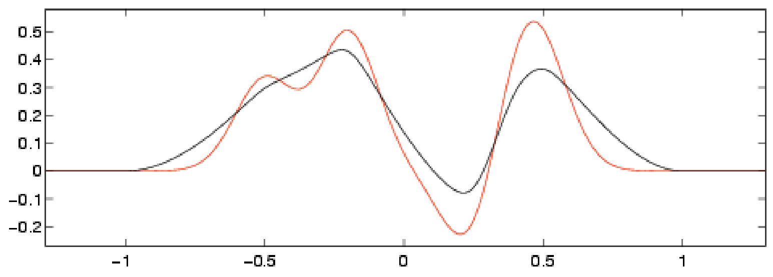


Figure 14: Comparison of the 3D biharmonic clamped plate spline (black line) with Gaussian RBF interpolation (red). Figure reproduced from [46].

A variety of functions can be used as the radial basis kernel $\phi(r)$. Some of the most mentioned choices in the literature are (c.f. [19, Appendix D]):

Gaussian	$\phi(r) = \exp(-(r/c)^2)$	
Hardy multiquadric	$\phi(r) = \sqrt{r^2 + c^2}$	
Inverse multiquadric	$\phi(r) = 1/\sqrt{r^2 + c^2}$	
Thin plate spline	$\phi(r) = r^2 \log r$	(in two dimensions)
Laplacian (or Polyharmonic) splines	$\phi(r) \propto \begin{cases} r^{2s-n} \log r & \text{if } 2s-n \text{ is an even integer,} \\ r^{2s-n} & \text{otherwise} \end{cases}$	

In fact there is not yet a general characterization of what functions are suitable as RBF kernels [19]. Positive definite functions are among those that will generate a non-singular $\mathbf{\Phi}$ matrix for any choice of data locations. There are several alternate characterizations of positive definiteness:

- A function is positive definite if

  $$\sum \sum w_i w_k \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) > 0$$

  for any choice of unique data locations $\mathbf{x}_i$.

- A positive definite function is a Fourier transform of a non-negative function [33]. (Note that "positive definite" does not mean that the function itself is necessarily positive. The sinc function is a counter example, since it is the transform of a box function.)

- The matrix $\mathbf{\Phi}$ generated from a positive-definite kernel function has all eigenvalues positive.

The third characterization indicates that for a positive definite function, the matrix $\mathbf{\Phi}$ will be invertable for any choice of data points. On the other hand, there are useful kernels (such as the polyharmonic family) that are not positive definite.

Several aspects of radial basis kernels are not intuitive at first exposure. The polyharmonic and thin-plate kernels are zero at the data points and *increase* in magnitude away from the origin. Also, kernels that are not themselves smooth can combine to give smooth interpolation. An example is the second-order polyharmonic spline in 3 dimensions, with kernel $\phi(r) = |r|$.

Several of the common RBF kernels have an explicit width parameter (e.g. the Gaussian σ), and the domain can be simply scaled to achieve similar effect with other kernels. The use of a broad kernel can cause surprising effects. This is due to the near-singularity of the $\mathbf{\Phi}$ matrix resulting from rows that are similar. The problem (and a solution) is discussed in section 3.3.

When thinking about the appropriate kernel choice for an application, it is also worth keeping in mind that there is more than one concept of smoothness. One criterion is that a function has a number of continuous derivatives. By this criterion the Gaussian function is a very smooth function, having an infinite number of derivatives. A second criterion is that the total curvature should be small. This can be approxmimated in practice by requiring that the solution should have a small integral of squared second deriviatives – the criterion used in the thin-plate and biharmonic splines.

This second criterion may correspond better to our visual sense of smoothness, see Figures 13, 14. Figure 13 shows a natural cubic spline on scattered data points. The curve extrapolates beyond the data, rather than falling back to zero, which may be an advantage or disadvantage depending on the application. Figure 14 compares two approaches that both fall to zero: Gaussian RBF interpolation (red) and the 3D biharmonic clamped plate spline (black line). The plot is the density of a linear section through an interpolated volume. Note that the control points are distributed through the volume and cannot meaningfully be visualized on this plot. Although the Gaussian function has an infinite number of derivatives, Gaussian RBF interpolation is visually less smooth than the biharmonic spline. The visual smoothness of the Gaussian can be increased by using a broader kernel, but this also increases the overshoots.

For many applications the most important considerations will be the smoothness of the interpolated function, whether it overshoots, and whether the function decays to zero away from the data. For example, artists tend to pick examples at the extrema of the desired function, so overshoot is not desirable. As another example, in the case where a pose space deformation algorithm is layered on top of an underlying skinning algorithm, it is important that the interpolated function fall back to zero where there is no data.

Of course it is not possible to simultaneously obtain smoothness and lack of overshoot, nor smoothness and decay to zero away from the data. It is possible, however, to combine these criteria with desired strengths. The approach is to find the Green's function (see section 5.2) for an objective such as

$$F(f) = \sum (f(x_i) - y_i)^2 \quad + \quad \alpha \int \|\nabla^2 f\|^2 dx \quad + \quad \beta \int \|\nabla f\|^2 dx \quad + \quad \gamma \int \|f\|^2 dx$$

i.e. a weighted sum of the data fidelity and the integral of the squared second derivative, the squared first derivative (i.e. "spline with tension"), and the squared function itself (causing the function to decay to zero).

The kernel choice leads to numerical considerations. The Gaussian kernel can be approximated with a function that falls exactly to zero resulting in a sparser and numerically better conditioned matrix matrix [65], while the polyharmonic kernels are both dense and ill-conditioned. Further numerical considerations will be discussed in section 3.4.

3.3 Regularization

In creating training poses for example-based skinning, the artist may accidentally save several sculpted shapes at the same pose. This results in a singular $\mathbf{\Phi}$ matrix. This situation can be detected and resolved by searching for identical

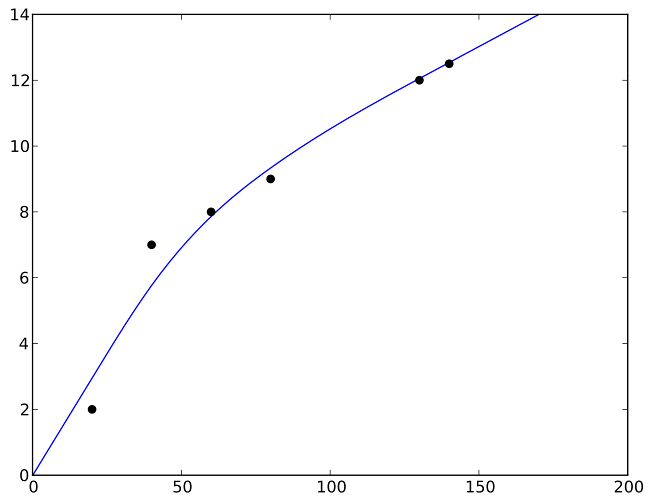


Figure 15: Adding regularization to the plot in Fig. 13 causes the curve to approximate rather than interpolate the data.

poses in the training data. A more difficult case arises when there are several (hopefully similar) shapes at nearby locations in the pose space.

In this case the $\mathbf{\Phi}$ matrix is not singular, but is poorly conditioned. The RBF result will probably pass through the given datapoints, but it may do wild things elsewhere (Figure 16). Intuitively speaking, the matrix is dividing by "nearly zero" in some directions, resulting in large overshoots.

In this case an easy fix is to apply weight-decay regularization. Rather than requiring $\mathbf{\Phi w} = \mathbf{d}$ exactly, weight-decay regularization solves

$$\arg\min_{\mathbf{w}} \|\mathbf{\Phi w} - \mathbf{d}\|^2 \quad + \quad \lambda\|\mathbf{w}\|^2 \tag{5}$$

This adds a second term that tries to keep the sum-square of the weight vector small. However, λ is chosen as a very small adjustable number such as 0.00001. In "directions" where the fit is unambiguous, this small number will have little effect. In directions where the result is nearly ambigous however, the $\lambda\|\mathbf{w}\|^2$ will choose a solution where the particular weights are small.

To visualise this, consider a two-dimensional case where one basis vector is pointing nearly opposite to the other (say, at 179 degrees away). A particular point that is one unit along the first vector can be described almost as accurately as being two units along that vector, less one unit along the (nearly) opposite vector. Considering matrix inversion in terms of the eigen decomposition $\mathbf{\Phi} = \mathbf{U\Lambda U}^T$. of the matrix is also helpful. The linear system is then $\mathbf{U\Lambda U}^T\mathbf{w} = \mathbf{d}$,

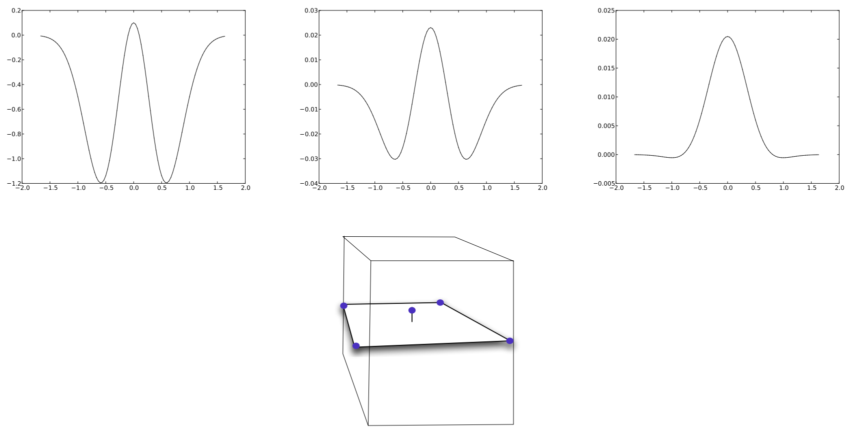


Figure 16: Illustration of ill-conditioning and regularization. From left to right, the regularization parameter is 0, .01, and .1 respectively. Note the vertical scale on the plots is changing. Bottom: The geometry of this 3D interpolation problem. The data are zero at the corners of a square, with a single non-zero value in the center.

and the solution is

$$\mathbf{w} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T\mathbf{d}$$

Geometrically, this is analogous to rotating $\mathbf{d}$ by $\mathbf{U}^{-1}$, stretching by the inverse of the eigenvalues, and rotating back. The problematic directions are those corresponding to small eigenvalues.

Soving (5) for $\mathbf{w}$,

$$\begin{aligned}
&\|\boldsymbol{\Phi}\mathbf{w} - \mathbf{d}\|^2 \quad + \quad \lambda\|\mathbf{w}\|^2 \\
&= \mathrm{tr}(\boldsymbol{\Phi}\mathbf{w} - d)^T(\boldsymbol{\Phi}\mathbf{w} - \mathbf{d}) \quad + \quad \lambda\mathbf{w}^T\mathbf{w} \\
&\frac{d}{d\mathbf{w}} = 0 = 2\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\mathbf{w} - \mathbf{d}) + 2\lambda\mathbf{w} \\
&\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{w} + \lambda\mathbf{w} = \boldsymbol{\Phi}^T\mathbf{d}
\end{aligned}$$

giving the solution

$$\mathbf{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\mathbf{d}$$

In other words, weight decay regularization requires just adding a small constant to the diagonal of the matrix before inverting.

Fig. 16 shows a section through simple a two-dimensional interpolation problem using a Gaussian RBF kernel. The data are zero at the corners of a square, with a single non-zero value in the center. Noting the vertical scale on the plots, the subfigures on the left show wild oscillation that is not justified by the simple data. In the figures we used weight-decay regularization to achieve a more sensible solution. This also has the effect of causing the reconstructed curve to approximate rather than interpolate the data (Fig. 15).

Weight-decay regularization is not the only type of regularization. Regularization is an important and well studied topic in all areas involving fitting a model to data. On a deeper level it is related to model complexity and the bias-variance tradeoff in model fitting. The subject comes up in a number of forms in statistical learning texts such as [26].

3.4 Computational considerations

Algorithms for solving linear systems typically have $O(N^3)$ time complexity. This means that if the number of training points is increased by a factor of 10, the solution time will grow by $10^3 = 1000$. Several more efficient approaches have been developed.

The fast multipole expansion [66] speeds up the computations of $\sum_k^N w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|)$ by splitting the kernel into sums of separable functions, some of which depend only on the training points and thus can be precomuted.

Beatson et al. [5] describe converting a thin-plate problem to an equivalent one with a localized (and thus numerically better behaved) basis. They solve for the localized basis as a cardinal interpolation of (some of) the data points during a preprocessing step.

In the domain decomposition method [4] the data set is subdivided into several smaller data sets and the interpolations equations are solved iteratively. This technique has the advantage of allowing parallel computations.

A different approach is to use only a subset of the available data, leaving the rest for refining or validating the approximation [28]. Unfortunately there is no known means of optimally choosing the subset (it is likely to be "NP"). Some heuristic approach must be used, such as greedily choosing training points that give the greatest reduction to a fitting error. Hatanaka *et al.* [27] uses a genetic algorithm to select the RBF centers.

Most of these techniques will introduce errors in the computation of the solution but since the reconstruction is itself an approximation of the true function, these errors might be acceptable.

In the case of the thin-plate and polyharmonic RBF kernels the system matrix is not sparse and will be increasingly ill conditioned as more points are added.

From a numerical point of view, solving an RBF system is known two depend on two quantities:

- The *fill distance* [64]. The fill distance expresses how well the data, D, fills a region of interest Ω:

  $$h_{D,\Omega} = \sup_{\mathbf{x}\in\Omega} \min_{\mathbf{x}_j \in D} \|\mathbf{x} - \mathbf{x}_j\|_2.$$

  It is the radius of the largest "data-site free ball" in Ω. The fill distance can be used to bound the approximation error.

- The *separation distance* [64]. The separation distance, q_D, measures how close the samples are together. If two data sites are very close then the interpolation matrix will be near-singular.

  $$q_D = \frac{1}{2} \min_{j \neq k} \|\mathbf{x}_j - \mathbf{x}_k\|_2.$$

  It can be shown [64] that the minimum eigenvalue of A, hence its condition number, is related to the separation distance:

  $$\lambda_{min}(A) \geq C q_D^{2\tau - d}$$

3.5 Applications

Mesh deformation. One application of scattered data interpolation is in image-based modeling. The work by Pighin *et al.* [44] describes a system for modeling 3-dimensional faces from facial images. The technique works as follow: after calibrating the cameras, the user selects a sparse set of correspondences across the multiple images. After triangulation, these correspondences yield a set of 3-dimensional constraints. These constraints are used to create a deformation field from a set of radial basis functions. Following their notation, the deformation field f can be written:

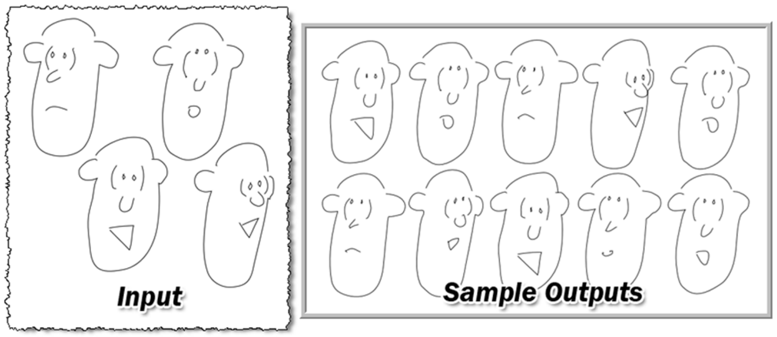
$$f(\mathbf{x}) = \sum_i c_i \phi(\mathbf{x} - \mathbf{x}_i) + M\mathbf{x} + t, ,$$

where ϕ is an exponential kernel. M is a 3×3 matrix and t a vector that jointly represent an affine deformation. The vanishing moment conditions can then be written as:

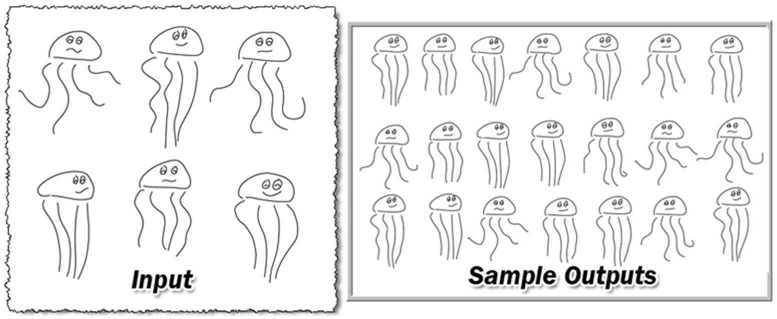
$$\sum_i c_i = 0 \text{ and } \sum_i c_i \mathbf{x}_i^T = 0$$

This can be simplified to just $\sum_i c_i \mathbf{x}_i^T = 0$ by using the notation that $\mathbf{x} \equiv \{1, x, y, z\}$.

Learning doodles by example. Baxter and Anjyo [3] proposed the concept of a latent doodle space, a low-dimensional space derived from a set of input doodles, or simple line drawings. The latent space provides a foundation for



(a) cartoon face



(b) jellyfish drawing

Figure 17: Examples of doodles by [3]. The left images were drawn by an artist; the right images were synthesized using the proposed technique.

generating new drawings that are similar, but not identical to, the input examples, as shown in Figure 17. This approach gives a heuristic algorithm for finding stroke correspondences between the drawings, and then proposes a few latent variable methods to automatically extract a low-dimensional latent doodle space from the inputs. Let us suppose that several similar line drawings are resampled by (1), so that each of the line drawings is represented as a feature vector by combining all the x and y coordinates of each point on each stroke into one vector. One of the latent variable methods in (2) then employs PCA and thin plate spline RBF as follows. We first perform PCA on these feature vectors, and form the latent doodle space from the first two principal components. With the thin plate spline RBF, we synthesize new drawings from the latent doodle space. The drawings are then generated at interactive rates with the prototype system in [3].

Locally controllable stylized shading. Todo et al. [61] proposes a set of simple stylized shading algorithms that allow the user to freely add localized light and shade to a model in a manner that is consistent and seamlessly

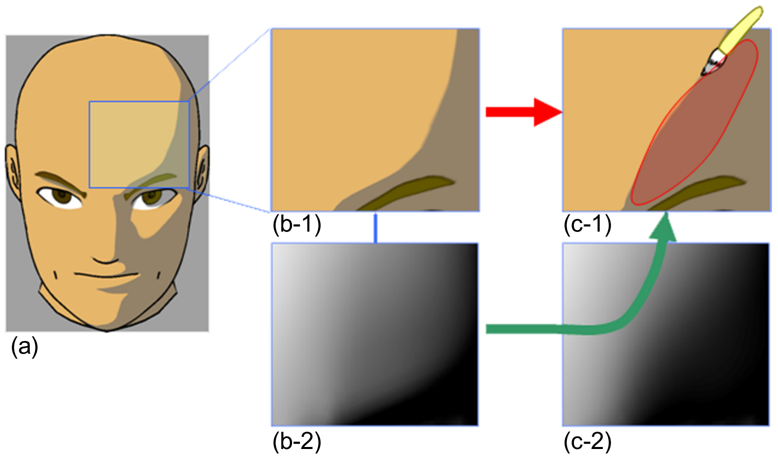


Figure 18: The interpolation problem in [61]. (a) is the original cartoon-shaded face; (b-1) is the close-up view of (a) and (b-2) is the corresponding intensity distribution (continuous gradation); (c-1) is the painted dark area and (c-2) is the corresponding intensity distribution that we wish to obtain.

integrated with conventional lighting techniques. The algorithms provide an intuitive, direct manipulation method based on a paint-brush metaphor to control and edit the light and shade locally as desired. For simplicity we just consider the cases where the thresholded Lambertian model is used for shading surfaces. This means that, for a given threshold c, we define the dark area $\boldsymbol{A}$ on surface $\boldsymbol{S}$ as being $\{\boldsymbol{p} \in \boldsymbol{S} | d(\boldsymbol{p}) \equiv \langle L(\boldsymbol{p}), N(\boldsymbol{p})\rangle < c\}$, where $L(\boldsymbol{p})$ is a unit light vector, and $N(\boldsymbol{p})$ is the unit surface normal at $\boldsymbol{p} \in \boldsymbol{S}$. Consider the cartoon-shaded area as shown in Figure 18, for example. Then let us enlarge the dark area in Figure 18 (a) using the paint-brush metaphor, as illustrated in (c-1) from the original area (b-1). Suppose that the dark area $\boldsymbol{A'}$ enlarged by the paint operation is given in the form: $\boldsymbol{A'} = \{\boldsymbol{p} \in \boldsymbol{S} | f(\boldsymbol{p}) < c\}$. Since we define the shaded area by the thersholded Lambertian, we'd like to find such a continuous function f as described above. Instead of finding f, let us try to get a continuous function $o(\boldsymbol{x}) := d(\boldsymbol{x}) - f(\boldsymbol{x})$, which we may call an *offset* function. The offset function may take 0 outside a neighborhood of the painted area. More precisely, let U be an open neighborhood of the painted area C. The desired function $o(\boldsymbol{x})$ should then be a continuous function satisfying:

$$o(\boldsymbol{x}) = \begin{cases} 0 & \text{, if } \boldsymbol{x} \in \partial U \cup (\partial A - C) \\ c - d(\boldsymbol{x}) & \text{, if } \boldsymbol{x} \in U - \partial(A \cup C). \end{cases} \tag{6}$$

To get $o(\boldsymbol{x})$, we *discretize* the above condition (6), as shown in Figure 19. Suppose that $\boldsymbol{S}$ consists of small triangle meshes. Using a simple linear interpolation method, we get the points $\{\boldsymbol{x}_i\}$ on the triangle mesh edges (or as the nodes)

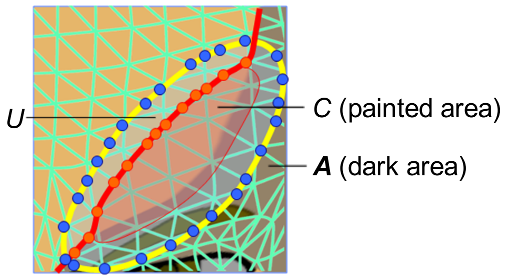


Figure 19: The constraint points for the interpolation problem in Figure 18. The triangle meshes cover the shaded face in Figure 18 (c-1). $\boldsymbol{A}$ is the initial dark area, whereas C and U denote the painted area and its neighborhood, respectively. U is the area surrounded by the closed yellow curve. The blue or red dots mean the boundary points $\{\boldsymbol{x}_i\}$ of these areas.

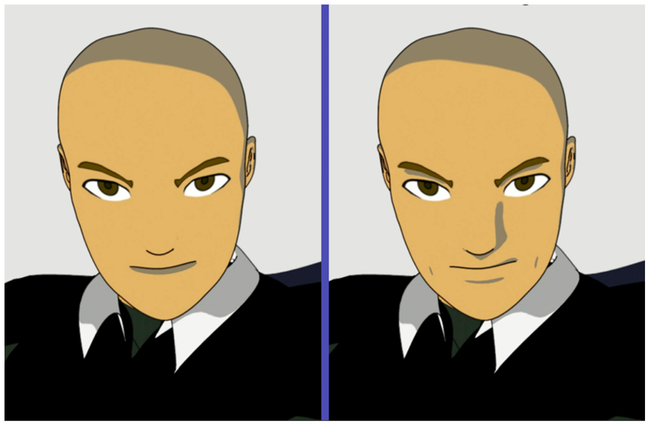


Figure 20: Toon-shaded 3D face in animation. *left*: with a conventional shader; *right* : result by [61].

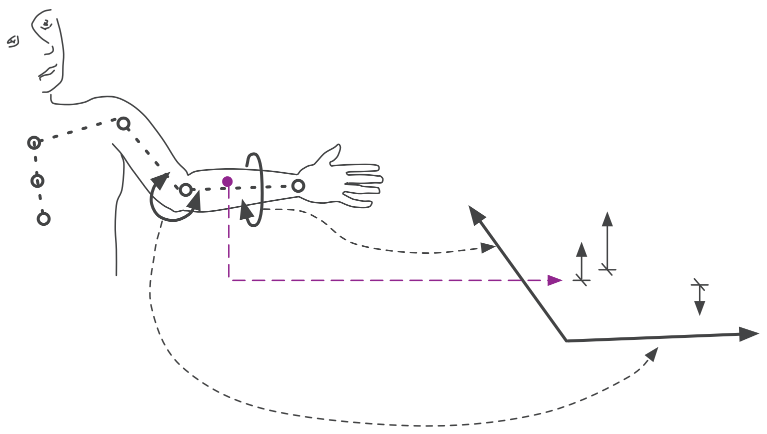


Figure 21: Scattered interpolation of a creature's skin as a function of skeletal pose.

which represent the constraint condition (6) to find an RBF f in the form:

$$f(x) = \sum_{i=1}^{l} w_i \phi(\boldsymbol{x} - \boldsymbol{x}_i) + p(\boldsymbol{x}), \tag{7}$$

where $\phi(\boldsymbol{x}) = \|\boldsymbol{x}\|$. This means that we find f in (7) under the condition (6) for $\{\boldsymbol{x}_i\}_{i=1}^{l}$ as a function defined on $\boldsymbol{R}^3$, rather than on $\boldsymbol{S}$. We consequently set $o(\boldsymbol{x}) := f(\boldsymbol{x})$. Figure 20 demonstrates the painted result. Just drawing a single image, of course, would not need the continuous function f. The reason why we need such an RBF technique as described above is that we wish to make the light and shade animated. This can be performed using a simple linear interpolation of the offset functions at adjacent keyframes.

Example-based skinning. Kurihara and Miyata [30] introduced the weighted pose-space deformation algorithm. This general approach interpolates the skin of character as a function of the degrees of freedom of the pose of its skeleton (Fig. 21). Particular sculpted or captured example shapes are located at particular poses and then interpolated with a scattered intepolation technique. This non-parametric approach allows arbitrary additional detail to be added by increasing the number of examples. For instance a bulging biceps shape might be interpolated as a function of the elbow angle.

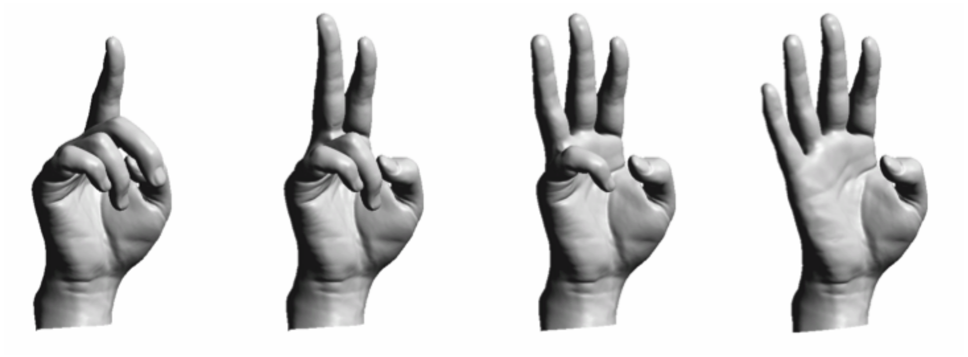


Figure 22: Hand poses synthesized using weighted pose space deformation (from [30]).

In the case of [30], the shapes were captured using a medical scan of one of the authors' hands, resulting in very plausible and detailed shapes (Fig. 22). The video accompanying the paper shows that the interpolation of these shapes as the fingers move is also very realistic.

Kurihara and Miyata used the cardinal interpolation scheme from [59], in combination with normalized radial basis functions. In this scheme, for n sample shapes there are n separate radial basis interpolators, with the kth interpolator using data that is 1 at the k training pose and 0 at all the other poses. The RBF matrix is based on the distance between the poses (in pose space) and so has to be formed and inverted only once.

Another important development in this paper is the way it uses skinning weights to effectively determine a separate pose space for each vertex. That is, the distance between two poses is defined as

$$d_j(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{\sum_k^n w_{j,k}(\mathbf{x}_{a,k} - \mathbf{x}_{b,k})^2}$$

where $w_{j,k}$ are the skinning weights for the kth degree of freedom for the jth vertex, and $\mathbf{x}_{a,k}$ denotes the kth component of location $\mathbf{x}_a$ in the pose space.

Facial animation "Blendshapes" are simple facial models that are composed of weighted linear combinations of a number of individual target shapes, each representing a particular expression. Blendshapes are a popular approach among animators because the targets provide direct control over the space of facial expressions [36]. On the other hand, the simple linear interpolation leaves much to be desired, particularly when several shapes are combined.

One solution involves adding a number of correction shapes [41, 53]. Each such shape is a correction to a pair (or triple, quadruple) of primary shapes. While

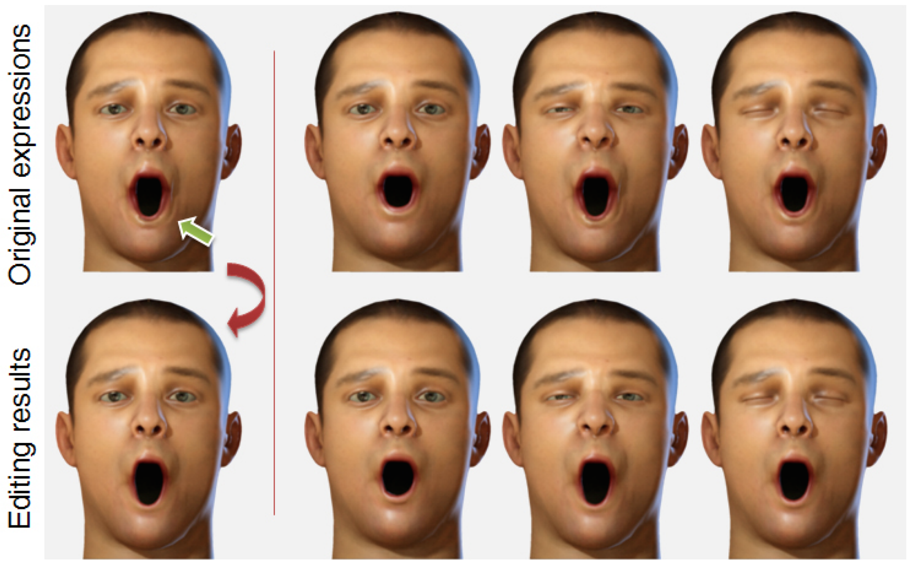


Figure 23: Smoothly propagating an expression change to nearby expressions using Weighted Pose Space Editing [55].

this improves the interpolation, it is a labor intensive solution due to the number of possible combination shapes. Finding the correct shapes also involves a labor-intensive trial and error process. Typically a bad shape combination may be visible at some arbitrary set of weights such as ($w_{10} = 0.3, w_{47} = 0.65$). The blendshape correction scheme does not allow a correction to be associated with this location, however, so the artist must find corrections at other locations such as ($w_{10} = 1, w_{47} = .1$), ($w_{10} = 1, w_{47} = 0$), ($w_{10} = 0, w_{47} = .1$) that together add to produced the desired shape. This requires iterative resculpting of the relevant shapes.

To solve this problem, Seol et al. [55] combine ideas from blendshapes and weighted pose space deformation [30]. Specifically, they use an elementary blendshape model (without corrections) as both a base model and a pose space for interpolation. Then, the weighted PSD idea is used to interpolate additional sculpted shapes. These shapes can be situated at any location in the weight space, and they smoothly appear and fade away as the location is approached (Figure 23). Because of the unconstrained location of these targets and the improved interpolation, both the number of shapes and the number of trial-and-error sculpting iterations are reduced.

Articulated animation for medical imaging MRI medical scans are relatively low resolution, and variants that capture multiple scans over time further compromise this resolution. To address the need to obtain higher quality volume

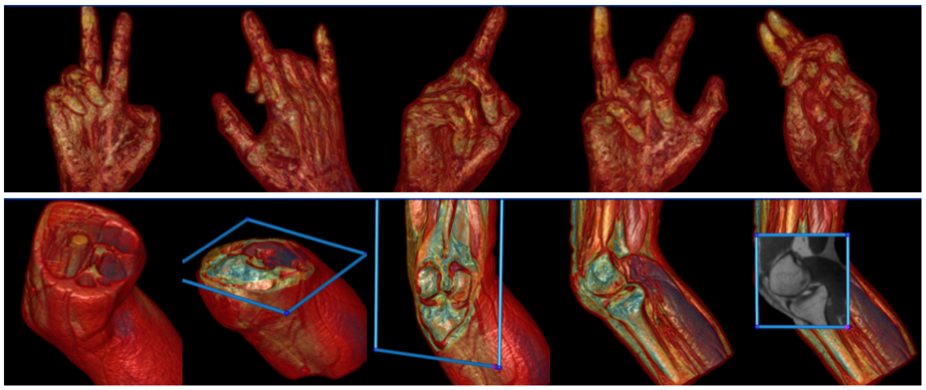


Figure 24: Example-based volumetric interpolation of medical scans.

visualizations of articulated regions such as the hand, Rhee et al. [46] developed an example-based volumetric interpolation system. A generic template hand model is first registered to available scans in a bootstrap step by using a volumetric analogue of linear blend skinning. The system then learns the additional deformations that are not captured by the animated template. This involves a hierarchical volumetric registration using clamped-plate splines. Finally, the detailed deformations at each example pose are interpolated as a function of pose as the underlying skeleton is moved (Figure 24).

4 Scattered interpolation on meshes: Laplace, Thin-plate

The previous subsection on radial basis methods mentioned choices of kernels for using RBFs to produce thin-plate interpolation. There is an equivalent formulation that does not use RBFs. This formulation minimizes a roughness, expressed as squared derivatives, subject to constraints (boundary conditions). In practice, this results in a single linear system solve for the unknowns via discretization of the Laplacian operator on a mesh. In this *mesh-based* approach the problem domain is often on a regular grid, so initially this may not seem like scattered interpolation. However, the unknowns can be scattered at arbitrary sites in this grid, so it is effectively a form of scattered interpolation in which the locations are simply quantized to a fine grid. In addition, forms of the Laplacian (Laplace Beltrami) operator have been devised for irregular meshes [24], allowing scattered interpolation on irregular geometry.

The Laplace equation is obtained by minimizing the integral of the squared first derivative over the domain, with the solution (via the calculus of variations)

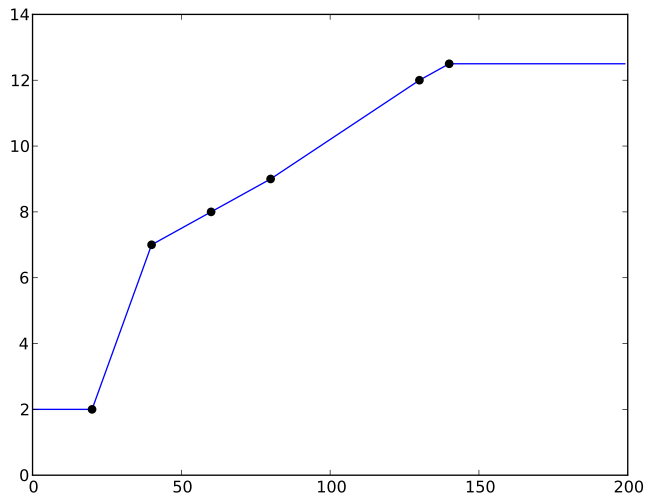


Figure 25: Laplace interpolation in one dimension is piecewise linear interpolation. Note that this figure was obtained by solving the appropriate linear system (rather than by simply plotting the expected result).

that the second derivative is zero:

$$\min_f \int \|\nabla f\|^2 d\Omega \quad \Rightarrow \quad \nabla^2 f = 0$$

Similarly, the biharmonic equation is obtained by minimizing the integral of the squared second derivative over the domain, with the solution that the fourth derivative is zero:

$$\min_f \int \left(\left| \frac{\partial^2 f}{\partial x^2} \right|^2 + 2 \left| \frac{\partial^2 f}{\partial x \, \partial y} \right|^2 + \left| \frac{\partial^2 f}{\partial y^2} \right|^2 \right) dx\, dy \quad \Rightarrow \quad \Delta^2 f = 0$$

The Laplace equation can be solved with a constrained linear system. Some intuition can be gained by considering the Laplace equation in one dimension with regularly spaced samples. A finite-difference approximation for the second derivative is:

$$\frac{d^2 f}{dx^2} \approx \frac{1}{h^2} \left(1 \cdot f(x+1) - 2 \cdot f(x) + 1 \cdot f(x-1)\right)$$

The weight stencil $(1, -2, 1)$ is important. If $f(x)$ is digitized into a vector $f = [f[1], f[2], \cdots, f[m]]$ then the second derivative can be approximated with a matrix expression

$$Lf \propto \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \cdots \\ & & \cdots & & \end{bmatrix} \begin{bmatrix} f[1] \\ f[2] \\ f[3] \\ \vdots \end{bmatrix}$$

In two dimensions the corresponding finite difference is the familiar Laplacian stencil

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \tag{8}$$

These weights are applied to the pixel sample $f(x, y)$ and its four neighbors $f(x, y-1), f(x-1, y), f(x+1, y), f(x, y+1)$. A two-dimensional array of pixels is in fact regarded as being "vectorized" into a single column vector $f(x, y) \equiv f_k$ with $k = y \times \text{xres} + x$.

Regardless of dimension the result is a linear system of the form $Lf = 0$, with some additional constraints that specify the value of $f(x_k)$ at specific positions. When the constraints are applied this becomes an $Lx = b$ system rather than a $Lx = 0$ nullspace problem.

A Poisson problem is similar, differing only in that the right hand side of the equation is non-zero. The problem arises by requiring that the gradients of the solution are similar to those of a guide function, where similar is defined as mininimizing the sum (or integrated) squared error. The Laplace equation is just a special case in which the "guide function" is 0, meaning that the desired gradients should be as small as possible, resulting in a function that is smooth in this sense.

In matrix terms, the corresponding thin-plate problem is of the form

$$L^2 f = b$$

where (again) some entries of f are known (i.e. constrained) and are pulled to the right hand side.

In both cases the linear system can be huge, with a matrix size equal to the number of vertices (in the case of manipulating a 3D model) or pixels (in a tone mapping or image inpainting application). On the other hand, the matrix is sparse, so a sparse solver can (and must) be used. The Laplace/Poisson equations are also suitable for solution via multigrid techniques, which have time linear in the number of variables.

4.1 Applications

Laplacian or Poisson interpolation on regular grids was introduced for image editing in [35, 43], and is now available in commercial tools. [8] used Laplacian and biharmonic interpolation on a mesh to drive a face mask with moving motion capture markers. [54] bring the editing power of [43] to facial motion capture retargeting by formulating a type of Poisson equation in terms of the blendshape facial representation.

5 Where do RBFs come from?

In the previous section we saw that Radial Basis Functions are a simple and versatile method for interpolating scattered data, generally involving only a standard linear system solve to find the interpolation weights, followed by interpolation of the form

$$f(\mathbf{x}) = \sum_k w_k \phi(||\mathbf{x} - \mathbf{x}_k||)$$

where $(\mathbf{x})$ is the resulting interpolation at point $\mathbf{x}$, and ϕ is a radially symmetric "kernel" function.

Several choices for the kernel were mentioned, such as $\exp(-r^2/\sigma^2)$ and some more exotic choices such as $r^2 \log r$.

Where do these kernels come from and how should you choose one? This section provides an intuitive discussion of this question.

5.1 Green's functions: motivation

Although Laplace and thin-plate interpolation is usually done by either sparse linear solves or relaxation/multigrid, it can also be done by radial basis interpolation, and this is faster if there are relatively few points to interpolate.

In these cases the kernel ϕ is the "Green's function of the corresponding squared differential operator". This section will explain what this means, and give an informal derivation for a simplified case. Specifically we're choosing a a discrete one-dimensional setting with uniform sampling, so the problem can be expressed in linear algebra terms rather than with calculus.

The goal is find a function f that interpolates some scattered data points d_k while simultaneously minimizing the roughness of the curve. The roughness is measured in concept as

$$\int |Df(x)|^2 dx$$

where D is a "differential operator" ($D = \frac{d^2}{dx^2}$) for example.

Whereas a *function* takes a single number and returns another number, an *operator* is something that takes a whole function and returns another whole function. The derivative is a prototypical operator, since the derivative of a function is another function. A differential operator is simply an operator that involves some combination of derivatives.

We are working in a discrete setting, so D is a matrix that contains a finite-difference version the operator. For example, for the second derivative, the (second order centered) finite difference is

$$\frac{d^2}{dx^2} \approx \frac{1}{h^2}\left(f_{t+1} - 2f_t + f_{t-1}\right)$$

This finite difference has the weight pattern $1, -2, 1$, and for our purposes we can ignore the data spacing h by considering it to be 1, or alternately by folding it into the solved weights.

The finite difference version of the whole operator can be expressed as a matrix as

$$D = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1\cdots \\ & & \cdots & & \end{bmatrix}$$

(and again the $\frac{1}{h^2}$ can be ignored for some purposes). The discrete equivalent of the integral (5.1) is then $\|Df\|^2 = f^T D^T D f$.

Then our goal (of interpolating some scattered data while minimizing the roughness of the resulting function) can be expressed as

$$\min_f \|Df\|^2 + \lambda^T S(f - d).$$

$$= \min_f f^T D^T D f + \lambda^T S(f - d). \tag{9}$$

S is a "selection matrix", a wider-than-tall permutation matrix that selects elements of f that correspond to the known values in d. So for example, if f is a vector of length 100 representing a curve to be computed that passes through 5 known values, S will be 5×100 in size, with all zeros in each row except a 1 in the element corresponding to the location of the known value. The known value itself is in d, a vector of length 100 with 5 non-zero elements. λ is a Lagrange multiplier that enforces the constraint.

Now take the matrix/vector derivative of eq. (9) with respect to f,

$$\frac{d}{df} = 2D^T D f + S^T \lambda$$

and we can ignore the 2 by folding it into λ.

If we know λ, the solution is then

$$f = (D^T D)^{-1} S^T \lambda \tag{10}$$

- Although continuously speaking the differential operator is an "instantaneous thing", in discrete terms it is a convolution of the finite difference mask with the signal. Its inverse also has interpretation as a convolution.
- If D is the discrete version of $\frac{d}{dx}$ then $D^T D$ is the discrete Laplacian, or the "square" of the original differential operator. Likewise if D is the discrete Laplacian then $D^T D$ will be the discrete biharmonic, etc.

  $(D^T D)^{-1}$ is the Green's function of the squared differential operator.

  In more detail, D^T is the discrete analogue of the adjoint of the derivative, and the latter is $-\frac{d}{dx}$ in the one-dimensional case [10]. The sign flip is evident in the matrix version of the operator: The rows of the original first derivative (forward or backward finite difference) matrix have -1,1 near the diagonal (ignoring the $\frac{1}{h}$). The transposed matrix thus has rows containing 1,-1, or the negative of the derivative.
- $S^T \lambda$ is a vector of discrete deltas of various strengths. In the example $S^T \lambda$ has 5 non-zero values out of 100.

5.2 Green's functions

Earlier we said that the kernel is the "Green's function of the corresponding squared differential operator". A *Green's function* is a term from differential equations. A linear differential equation can be abstractly expressed as

$$Df = b$$

where D is a differential operator as before, f is the function we want to find (the solution), and b is some "forcing function". In the Greens function approach to solving a differential equation, the solution is assumed to take the form of a convolution of the Green's function with the right hand side of the equation (the forcing function). (We are skipping a detail involving boundary conditions). For linear differential equations the solution can be expressed in this form.

The Green's function G is the "convolutional inverse" of the squared differential operator. It is the function such that the operator applied to it gives the delta function,

$$DG = \delta \tag{11}$$

While the theory of Green's functions is normally expressed with calculus, we'll continue to rely on linear algebra instead. In fact, the appropriate theory has already been worked out in the previous subsection and merely needs to be interpreted.

Specifically, in Eq. (10), $(D^T D)^{-1}$ is the Green's function of the squared differential operator. In discrete terms Eq. 11 is

$$D^T D (D^T D)^{-1} = I$$

with the identity matrix rather than the δ function on the right hand side. And $S^T\lambda$ is a sparse set of weighted impulses that is "convolved" (via matrix multiplication) with the Green's function. Thus we see that Eq. 10 is a discrete analogue to a Green's function solution to the original "roughness penalty" goal.

6 Additional Topics

6.1 Interpolating physical properties

Incompressible fluids (water) have divergence-free velocity fields. Using a standard scattered interpolation method on fluid data would violate this divergence-free property. In fact, there is a divergence-free variant of RBF interpolation [38]. There are also techniques for interpolating curl-free phenomena. One way of doing this is to perform a Helmholtz-Hodge decomposition on the kernel to obtain a divergence-free and a curl-free component [23]. E.J. Fuselier [22] also proposes curl-free matrix-valued RBFs.

6.2 Scattered data interpolation in mesh-free methods

Mesh-free approaches are an alternative to finite-element and related integration methods. In these approaches the computation volume is not defined by a static grid but rather the medium is represented by a set of moving particles. Scattered data interpolation is required to interpolate simulated values from the particles to the entire computation volume.

6.3 Scattered data interpolation on a manifold

One nice aspect of RBF interpolation is that it works in any dimension, with the only change being how the distance between points is measured. In skinning, for example, the dimensionality of the "pose space" may be 10 or higher.

In some cases however the data lie on a curved surface, and interpolating in a n-dimensional "flat" Euclidean space is not ideal regardless of the dimensionality. For example, if the data are intrinsically on a sphere, it is possible to formulate the problem as 3D interpolation and then evaluate the results only on the sphere, but this will not give ideal results.

To explain the problem, consider the example of driving a facial mesh with motion capture data. The data takes the form of a number of markers moving in 3D over time. A 3D scattered interpolation will deform the mesh. However, the motion of the upper-lip markers will incorrectly influence the lower-lip geometry, especially if the lips are close. This is because the RBF is based on 3D distance, and the upper lip is adjacent to the lower lip.

Section 4 described methods for interpolating based on generalizations of Laplace's equation, and Section 5 described how the RBF approach is in a sense "dual" to these approaches. The well known "mesh Laplacian" from discrete differential geometry [24] allows the interpolation to occur directly on the mesh surface [8]. In this way, the motion "bleeding" across a hole such as the mouth will be much less of a problem because the influence of the sample gets propagated on the mesh (and around the mouth) rather than cutting across the mouth. Rhee et al. [45] solved this issue using RBFs by introducing a geodesic form of RBF, i.e. the distance between points passed to the RBF kernel is the geodesic distance on the face surface.

The idea of a two-dimensional surface mesh embedded in three-dimensional space generalizes to higher dimensions, where the "surface" is instead (informally) termed a manifold, embedded in a higher dimensional ambient space. The "manifold assumption" in machine learning asserts that most of the data in high-dimensional datasets exists on or close to some (possibly curved) manifold, and a body of *manifold learning* algorithms [60, 48] seek a parameterization of this subspace given only scattered data in the high dimensional space.

Interpolation on the sphere is a special case that is considered separately due to its importance (for describing orientations for example). For a survey of interpolation on the sphere see [18].

7 Guide to the deeper theory

This chapter does not aim at developing a rigorous mathematics behind the techniques described in this course, but at giving intuitive meanings to the mathematical concepts that support those techniques. We do hope that this chapter will give a good introduction to learn the mathematics behind the scene more deeply.

Those mathematical concepts come mainly from *functional analysis*, the mathematical field of modern calculus. The modern calculus introduces a variety of function spaces, such as Hilbert space or Sobolev space, while generalizing concepts in classical calculus. As described below, this gives us a theoretical basis for solving the optimization problems or regularization problems of our practical situations.

7.1 Elements of functional analysis

A function classically means a mapping that gives a real value, denoted by $f(x)$, for a given real number x. One of the motivations for generalizing the concept of function was to give a mathematical validation of Dirac delta function, δ, which has following properties:

$$f(0) = \int f(x)\delta(x)dx \tag{12}$$

or

$$f(t) = \int f(x)\delta(t-x)dx$$

for any ordinary function $f(x)$.

Alternatively the Dirac delta function could be expressed as

$$\delta(t) = \begin{cases} +\infty & (t=0) \\ 0 & (otherwise). \end{cases}$$

But this suggests that δ function is not an ordinary function. The Heaviside function $H(t)$ is also well-known (mainly in signal processing) with its derivative being δ function:

$$H(t) = \begin{cases} 1 & (t \geq 0) \\ 0 & (otherwise). \end{cases}$$

However, $H(t)$ is not differentiable nor continuous at $t = 0$ in a classical sense. How can we explain these things in a mathematically correct way? What we need is therefore to give alternative definitions of functions, derivative, and many more technical terms in classical calculus.

7.2 Brief Introduction to the theory of generalized functions

To achieve a new concept of "derivative", we first take a look at the formula, known as *integration by parts*. For simplicity, we consider a one-dimensional case. Then we have:

$$\int_a^b f'(x)g(x)dx = -\int_a^b f(x)g'(x)dx + [f(t)g(t)]_{t=a}^{t=b}. \tag{13}$$

We derive this formula, supposing that both f and g are smooth (differentiable).

Next let us suppose that g vanishes at the boundary (i.e., $g(a) = g(b) = 0$ in (13)). We then have:

$$\int_a^b f'(x)g(x)dx = -\int_a^b f(x)g'(x)dx. \tag{14}$$

Further, what if the above f is not differentiable (for example, $f = \delta$)? The left hand in (14) is therefore meaningless, but the right hand is still valid. Equation (14) might therefore tell us that, instead of taking the derivative of f, we can think of doing so for g. This could be understood if we consider f as a *functional*, rather than a function, as described later.

Function space and functional Keeping the above things in mind, we next introduce a concept of function space. A function space $\Im$ is a collection of functions defined on a certain domain (typically, region Ω in $\boldsymbol{R}^n$). Here are the examples of function spaces:

Function Space Examples:

1. $\mathcal{P}_m := \{P(x)|P(x)$ is a polynomial of at most m-th order[1] $\}$.
2. $C^m(\Omega)$ is the totality of m-th order smoothly differentiable functions on Ω, where $m = 0$ (the totality of continuous functions), $1, 2, \cdots$, or ∞.
3. $C_0^\infty(\Omega)$ is the totality of infinitely many times differentiable functions on Ω with compact support (i.e., each function of this function space vanishes outside a large ball in $\boldsymbol{R}^n$).
4. $L^p(\Omega) := \{f : \Omega \to R \cup \{\pm\infty\} | \int |f(x)|^p dx < \infty\}$[2], where p is a positive number.

A function space $\Im$ is usually treated as a linear topological space. This means that $\Im$ is a vector space, where convergence of a function sequence is defined (see the section on Hilbert space for details). Next recall a function $f : \Omega \to R$. f is then defined on Ω, and gives a real number $f(x)$ when x is specified. Now we consider mapping F from a function space $\Im$ to R. We call the mapping $F : \Im \to R$ a *functional*. A functional F is defined on a function space, and gives a real number $F(\varphi)$ when an element φ of the function space is specified.

Functional Examples:

1. (*Thin plate spline*) Let Ω be a domain in $\boldsymbol{R}^2$. Let $\boldsymbol{B}_2^2(\Omega)$ be the function space defined as:

$$\boldsymbol{B}_2^2(\Omega) := \{\varphi(x) : \Omega \to R \cup \{\pm\infty\} | \frac{\partial^2\varphi}{\partial x_1^2}, \frac{\partial^2\varphi}{\partial x_1 \partial x_2}, \frac{\partial^2\varphi}{\partial x_2^2} \in L^2(\Omega)\}. \tag{15}$$

[1]This space was denoted Π_{m+1} in subsection 3.1

[2]Rigorously, dx should be denoted by $d\mu(x)$ with Lebesgue measure μ. But we don't have to consider such mathematical details in these course notes.

To get a thin plate spline curve, we then consider the functional F on $\boldsymbol{B}_2^2$ as:

$$F(\varphi) := \iint_\Omega \left(\left| \frac{\partial^2 \varphi}{\partial x_1^2} \right|^2 + 2 \left| \frac{\partial^2 \varphi}{\partial x_1 \partial x_2} \right|^2 + \left| \frac{\partial^2 \varphi}{\partial x_2^2} \right|^2 \right) dx_1 dx_2. \tag{16}$$

2. An ordinary function f can also be identified with a functional. The functional, denoted by T_f, is defined as

$$T_f(\varphi) := \int_\Omega f(x)\varphi(x)dx, \tag{17}$$

for any φ in a certain function space $\Im$[3].

3. (*Dirac delta function*) Consider the function space $C_0^\infty(\Omega)$, where $0 \in \Omega$. For any element $\varphi \in C_0^\infty(\Omega)$, Dirac delta function is defined as:

$$T_\delta(\varphi) := \varphi(0). \tag{18}$$

We then note that the functionals in (17) and (18) are linear. This means, for instance, that we have:

$$T_f(\alpha\varphi + \beta\psi) = \alpha T_f(\varphi) + \beta T_f(\psi),$$

for any φ, ψ and any $\alpha, \beta \in \boldsymbol{R}$.

We will show that, a (continuous) linear functional is the generalized function which gives the theoretical basis on discussions in this course notes. Before doing this, we need to investigate more about the relation between f and T_f.

Functions as functionals Now go back to T_f, for an ordinary function f. Then we want to identify the function f with the functional T_f. To make it, we should investigate whether the following property holds:

$$T_f = T_g \Leftrightarrow f = g \tag{19}$$

For example, it is easy to see that this property holds, if f is a continuous function on Ω, and if the linear functional T_f is defined on $C_0^\infty(\Omega)$. Moreover we can get this identification (19) for a wider class of functions. Let's skip, however, the mathematical proof of (19) and mathematical details in the background. Rather, what we should recognize now is that *an ordinary function f can be identified with a functional T_f on a certain function space* through (19).

[3]We of course assume $T_f(\varphi)$ in (17) takes a finite value for any φ in $\Im$.

Generalized function and its derivative We thus define a *generalized function* in the following way.

Definition: Let $\Im$ be a function space[4]. Let T be a functional: $\Im \to \boldsymbol{R}$. T is called a *generalized function (or distribution)*, if it satisfies the following conditions:

1. T is linear:

$$T(\alpha\varphi + \beta\psi) = \alpha T(\varphi) + \beta T(\psi), \text{for } \varphi, \psi \in \Im, \alpha, \beta \in \boldsymbol{R}. \tag{20}$$

2. T is continuous:

$$\lim_{n\to\infty} \varphi_n = \varphi \text{ in } \Im \Rightarrow \lim_{n\to\infty} T(\varphi_n) = T(\varphi) \text{ in } \boldsymbol{R}. \tag{21}$$

Next we define *derivative* of a generalized function. The hint of making it again lies in the formula of integration by parts in (14). For simplicity we consider one-dimensional case, taking $\Im = C_0^\infty(\Omega)$. Suppose that f is smooth (differentiable). We then note that equation (14) suggests

$$T_{\frac{df}{dx}}(\varphi) = \int \frac{df}{dx}\varphi dx = -\int f\frac{d\varphi}{dx}dx = -T_f\left(\frac{d\varphi}{dx}\right). \tag{22}$$

It therefore seems natural to define the derivative of the generalized function T, as follows:

Definition: Let T be a generalized function: $\Im \to \boldsymbol{R}$.The derivative of T, denoted by T', is defined by

$$T'(\varphi) = -T(\frac{d\varphi}{dx}). \tag{23}$$

We note that T' itself is also a generalized function. The above definition by (23) is reasonable, because, if we consider the case where T is induced by a smooth function (i.e., $T = T_f$), it follows from (22) that $T'(\varphi) = T_{\frac{df}{dx}}(\varphi)$. In the following T' is sometimes denoted by $\frac{dT}{dx}$.

As an exercise, let us calculate the derivative of Heaviside function $H(x)$ ($=1$ if $x \geq 0$, and $=0$, otherwise) in the sense of distribution. Instead of H itself, we therefore consider T_H. Then we can differentiate it as a linear functional.

$$\begin{aligned}\frac{dT_H}{dx}(\varphi) &= -T_H\left(\frac{d\varphi}{dx}\right) = -\int_{-\infty}^{+\infty} h(x)\frac{d\varphi}{dx}dx = -\int_0^\infty \frac{d\varphi(x)}{dx}dx \\ &= -[\varphi(t)]_{t=0}^{t=\infty} = \varphi(0) \equiv T_\delta(\varphi)\end{aligned}$$

[4]$\Im$ could be $C_0^\infty(\Omega), C^\infty(\Omega)$, and other function spaces, which would bring us various generalized functions [52]. However, in our course notes, we mostly set $\Im$ as $C_0^\infty(\Omega)$.

Similarly we can inductively define the n-th order derivative of T.

Definition: The n-order derivative of a generalized derivative of T, denoted by $\frac{d^nT}{dx^n}$ (or by $T^{(n)}$), is defined by

$$\frac{d^nT}{dx^n}(\varphi) := (-1)^n T(\frac{d^n\varphi}{dx^n}). \tag{24}$$

Again $\frac{d^nT}{dx^n}$ is a generalized function. We could therefore say that any generalized function (on $\Im$) is infinitely many times differentiable. Let $\Im'$ be the totality of the generalized functions on $\Im$. We call $\Im'$ the dual space of $\Im$, which again constitutes a vector space.

The dual space $\Im'$ includes δ function, Heaviside function, and regular functions (such as continuous or smooth functions). In practice we can consider most generalized function as being in the form T_f with an ordinary function f. More precisely, if a function f is *locally integrable*[5], T_f is then a generalized function. The regular functions and Heaviside function are locally integrable, while δ function is not. On the other hand, though the definition of Dirac δ in (18) looks a bit artifical, if we *symbolically* use the integral representation like $\int \varphi(x)\delta(x)dx$ instead of $T_\delta(\varphi)$, we still have equation (12) valid in the sense of distribution, which simply means $\int \varphi(x)\delta(x)dx \equiv T_\delta(\varphi) = \varphi(0)$.

Once we get such a mathematical concept of generalized function as described above[6], we can reformulate the problems in classical calculus. For instance the problems of solving ordinary/partial differential equations (ODE/PDE) is described in the following way. Let $P(\xi)$ be an m-th order polynomial with constant coefficients. For a one-dimensional case, $P(\xi) = \sum_{n=0}^m a_n\xi^n$. We then define the differential operator $P(\frac{d}{dx})$ as being $P(\frac{d}{dx})u = \sum_{n=0}^m a_n \frac{d^nu}{dx^n}$. Similarly, in a multi-dimensional case, we consider $P(\xi) \equiv P(\xi_1, \xi_2, \cdots, \xi_n) =$

$$P(\xi) \equiv \sum_{|\alpha|=\alpha_1+\alpha_2+\cdots+\alpha_n \geq 0}^{m} C_{\alpha_1,\alpha_2,\cdots,\alpha_n}\xi_1^{\alpha_1}\cdot\xi_2^{\alpha_2}\cdots\xi_n^{\alpha_n},$$

where $C_{\alpha_1,\alpha_2,\cdots,\alpha_n}$ are constant coefficients. Then we set

$$P(D) \equiv P\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_n}\right)$$

[5]This means that $\int_K |f(x)|dx < +\infty$ holds for any compact set (i.e., bounded and closed) K in Ω.

[6]The theory of hyperfunction [49] gives an alternative theoretical basis on calculus. However, it requires algebraic concepts, such as sheaf and cohomology, so that it is not so elementary, compared to the distribution theory [52], which we discuss in these course notes.

as a partial differential operator. If $P(D)$ is a monomial, for instance, like $D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}$, we put

$$D^\alpha u := \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}.$$

Classically, for a given function f on Ω, we want to find a (smooth) solution u that satisfies $P(D)u = f$ on Ω[7]. Now this is understood as the following problem:

Solving differential equation in $\Im'$:
For a given $F \in \Im'$, find a $T \in \Im'$ such that $P(D)T = F$.

Of course we may take the given F as an ordinary function f, in the sense that we put $F = T_f$. According to the above formulation, we want to find a solution in the much wider space $\Im'$. Even if we can find the solution in $\Im'$, it is not easy to see whether the solution can be represented as an ordinary function. The solution would therefore be called a *weak solution.* In addition, when we *differentiate* a generalized function T, we would then refer to $\frac{dT}{dx}$ as the *weak derivative* of T. In this theoretical approach, what we should do first is to assure the existence of the solution, whereas we need a practical solution. So there still exists a gap between the theory and our practical demand. However, it should be noted that *Finite Element Methods* (FEM) and several related approaches have the theoretical basis from the distribution theory.

7.3 Hilbert Space

We first explain pre-Hilbert space. To make it, we need the following definition.

Definition: Let $\boldsymbol{F}$ be a vector space over $\boldsymbol{R}$. The two-term operation, denoted by $\langle\ ,\ \rangle$ (or $\langle\ ,\ \rangle_{\boldsymbol{F}}$), is called the *inner product*, if it satisfies the following conditions:

- $\langle , \rangle$ is symmetric and bilinear:
  $\langle f, g\rangle = \langle g, f\rangle$,
  $\langle \alpha_1 f_1 + \alpha_2 f_2, g\rangle = \alpha_1\langle f_1, g\rangle + \alpha_2\langle f_2, g\rangle$,
  $\langle f, \beta_1 g_1 + \beta_2 g_2, \rangle = \beta_1\langle f, g_1\rangle + \beta_2\langle f, g_2\rangle$
  for any $f, f_1, f_2, g, g_1, g_2 \in \boldsymbol{F}$, and any $\alpha, \alpha_1, \alpha_2, \beta, \beta_1, \beta_2 \in \boldsymbol{R}$.
- $\langle f, f\rangle \geq 0$, for any$f \in \boldsymbol{F}$, and $\langle f, f\rangle = 0$, if and only if $f = 0$.

The vector space with the inner product is called a *pre-Hilbert space.* It is then noted that any pre-Hilbert space $\boldsymbol{F}$ is a normed space with the norm $\|\ \|$ (or denoted by $\|\ \|_{\boldsymbol{F}}$, if necessary), which is induced by the inner product: $\|f\| := \sqrt{\langle f, f\rangle}$. The following formula always holds, known as Schwarz' inequality:

[7]We skip the discussion on the initial condition, for simplicity.

$$|\langle f,g\rangle| \le \|f\|\|g\|, \text{ for any } f,g \in \boldsymbol{F}. \tag{25}$$

Now, similar to the case of $\Im$ and $\Im'$, we consider $\boldsymbol{F}$ and its dual space $\boldsymbol{F}'$, which is the totality of continuous linear functionals on $\boldsymbol{F}$. Let T be a linear functional on $\boldsymbol{F}$. This means that T satisfies the condition (20) for $\Im = \boldsymbol{F}$). The continuity of T in (21) is then expressed using the norm:

There exists a constant C such that $|T(f)| \le C\|f\|$ holds for any $f \in \boldsymbol{F}$. (26)

Further, if the pre-Hilbert space $\boldsymbol{F}$ is complete, i.e., any Cauchy sequence $(f_n) \subset \boldsymbol{F}$ has its limit in $\boldsymbol{F}$, $\boldsymbol{F}$ is then called a *Hilbert space*.

Hilbert Space Examples:

1. $\boldsymbol{R}^n$. For $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)^T \in \boldsymbol{R}^n$, we have $\langle \boldsymbol{x}, \boldsymbol{y}\rangle := \sum_{i=1}^n x_i y_i$.
2. $l^2 := \{\boldsymbol{c} = (c_j)_{j=1}^\infty | c_j \in \boldsymbol{R}, \sum_{j=1}^\infty |c_j|^2 < \infty\}$. The inner product is given by: $\langle \boldsymbol{c}, \boldsymbol{d}\rangle := \sum_{i=1}^\infty c_i d_i$, where $\boldsymbol{c} = (c_j)_{j=1}^\infty$ and $\boldsymbol{d} = (d_j)_{j=1}^\infty \in l^2$.
3. L^2 *space*: $L^2(\Omega) = \{f : \Omega \to \boldsymbol{R} \cup \{\pm\infty\} | \int_\Omega |f(x)|^2 dx < \infty\}$. For f and $g \in L^2(\Omega)$, we have $\langle f,g\rangle := \int_\Omega f(x)g(x)dx$.
4. *Sobolev space*: $W_2^m(\Omega) := \{f \in L^2(\Omega) | D^\alpha f \in L^2(\Omega), |\alpha| \le m\}$, where $D^\alpha f$ means a weak derivative. The inner product is given by $\langle f,g\rangle := \sum_{|\alpha|\le m} \int_\Omega D^\alpha f(x) D^\alpha g(x)dx$.

We will use the following basic theorem in explaining RKHS in the next section:

Theorem 1 (Riesz): Let $\boldsymbol{H}$ be a Hilbert space, and T be a real-valued continuous linear functional on $\boldsymbol{H}$. Then there exists one and only one function $\boldsymbol{t} \in \boldsymbol{H}$ such that

$$T(f) = \langle \boldsymbol{t}, f\rangle \tag{27}$$

for all $f \in \boldsymbol{H}$.

This theorem says that Hilbert space $\boldsymbol{H}$ is isomorphic to its dual space $\boldsymbol{H}'$ as linear topological spaces[8]

[8]The norm $\| \ \|$ in $\boldsymbol{H}'$ is then given by $\|T\| := sup_{\|f\|\le 1}|T(f)|$, for $T \in \boldsymbol{H}'$.

Fourier Analysis in Hilbert Space We first recall the orthogonal relation between trigonometric functions:

$$\int_{-\pi}^{\pi} \cos nx \cos mx \; dx = \begin{cases} \pi \; (m = n) \\ 0 \; (m \neq n), \end{cases}$$

$$\int_{-\pi}^{\pi} \sin nx \sin mx \; dx = \begin{cases} \pi \; (m = n) \\ 0 \; (m \neq n), \end{cases}$$

$$\int_{-\pi}^{\pi} \cos nx \sin mx \; dx = 0.$$

Let f and g be the elements of a Hilbert space $\boldsymbol{H}$. We say that f is orthogonal to g, if $\langle f, g\rangle = 0$.

Consider now the function space $L^2(-\pi,\pi)$, where the inner product is given by: $\langle f, g\rangle := \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)g(x)dx$. The subset $S := \{1, \cos nx, \sin nx \mid n = 1, 2, \cdots\}$ of $L^2(-\pi,\pi)$ is then known as a *complete orthonormal system* of $L^2(-\pi,\pi)$. This means that the system $S \equiv \{\varphi_1, \varphi_2, \cdots\}$ satisfies the following conditions:

$$\langle \varphi_i, \varphi_j\rangle = \delta_{ij} \text{ for } i, j = 1, 2, \cdots . \tag{28}$$

$$f = \sum_{j=1}^{\infty} \langle f, \varphi_j\rangle \varphi_j, \text{ for any } f \in L^2(-\pi,\pi). \tag{29}$$

The conditions (28) and (29) of course give the definition of complete orthonormal system (CONS) for a general Hilbert space. The coefficients $\langle f, \varphi_j\rangle$ in (29) are then referred to as the Fourier coefficients of f. The right hand of equation (29) is called the Fourier series of f, with regard to the CONS S.

CONS Examples

1. *Legendre polynomials* $\{P_n(x)\}$

$$P_n(x) := \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \; (n = 0, 1, \cdots).$$

Then we have

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{2}{2n+1}\delta_{mn}.$$

Therefore $\{\sqrt{\frac{2n+1}{2}}P_n(x); n = 0, 1, 2, \cdots\}$ constitutes a CONS of $L^2(-1, 1)$.

2. *Laguerre polynomials* $\{L_n(x)\}$

$$L_n(x) := e^x \frac{d^n}{dx^n}(x^n e^{-x}) \; (n = 0, 1, \cdots).$$

This time we have

$$\int_{0}^{\infty} L_m(x)L_n(x)dx = (n!)^2\delta_{mn}.$$

Therefore $\{\frac{1}{n!}L_n(x)e^{-x/2}; n = 0, 1, 2, \cdots\}$ constitutes a CONS of $L^2(0, \infty)$.

For our practical purposes, we may assume that Hilbert space always has a complete orthonormal system[9].

7.4 Reproducing Kernel Hilbert Space

A radial basis function (RBF) is closely related to a reproducing kernel Hilbert space (RKHS). Actually an RBF appear in solving a regularization problem on a certain RKHS $\boldsymbol{H}(\Omega)$, where Ω is a domain in $\boldsymbol{R}^n$ or $\boldsymbol{R}^n$ itself. As mentioned in earlier sections, there are several RBFs that are used in scattered data interpolation, such as *Gaussian*, and *thin plate spline*. The choice of an RBF that is most suited in solving an interpolation/regularization problem depends not only on the smoothness of the function to be desired but also on the dimension n. In this section we will briefly sketch this interesting relation among RBF, smoothness of the function, and the space dimension. Now let us start with the definition of RKHS.

7.4.1 Reproducing Kernel

Let E be an abstract set, and $\boldsymbol{H}$ be a Hilbert space consisting of the (real-valued[10]) functions defined on E, with the inner product $\langle , \rangle$.

Definition The function $K : E \times E \to \boldsymbol{R}$ is called a *reproducing kernel* of $\boldsymbol{H}$, if it satisfies the following conditions[11]:

1. For any fixed $y \in E$, $K(x, y)$ belongs to $\boldsymbol{H}$ as a function of x on E.
2. For any $f \in \boldsymbol{H}$, we have $f(y) = \langle f(x), K(x, y)\rangle_x$.

Definition If Hilbert space $\boldsymbol{H}$ has the reproducing kernel, which satisfies the above conditions (1) and (2), then $\boldsymbol{H}$ is referred to as a *reproducing kernel Hilbert space* (RKHS).

The following proposition will be used in characterizing the reproducing kernel in the next section.

Proposition 1. For the reproducing kernel K, we have:

$$K(y, z) = \langle K(x, y), K(x, z)\rangle_x \tag{30}$$

[9]Rigorously, it is a necessary and sufficient condition for Hilbert space H to have a complete orthonormal system that H is *separable* as a metric space. This means that there exists a dense, and countable subset of H. However we don't have to care about it for our practical situations.

[10]Formally, we can treat complex-valued functions, but the "complex" case may be skipped in this course.

[11]In condition (2), the inner product $\langle , \rangle_x$ means that we get the inner product value of the two functions with variable x.

Proof. From condition (1) of the reproducing kernel, we have $K(y,z) \in \boldsymbol{H}$ for a fixed z. Then, by putting $f(y) = K(y,z)$ in condition (2), we have

$$K(y,z) = \langle K(x,z), K(x,y)\rangle_x = \langle K(x,y), K(x,z)\rangle_x.$$

The next theorem is well known as a classical result in the reproducing kernel theory (N. Aronszajn [2], S. Bergman [7]).

Theorem 2: Hilbert space $\boldsymbol{H}(E)$ has a reproducing kernel, if and only if the following condition is satisfied:
For any $y \in E$, there exists a positive constant $C = C_y$, such that

$$|f(y)| \leq C_y \|f\|, \text{ for any } f \in \boldsymbol{H}. \tag{31}$$

Proof. [*only if* part] The assertion follows from Schwarz' inequality (25). [*if* part] For a fixed $y \in E$, let us consider the linear functional $\delta_y : \boldsymbol{H}(E) \to \boldsymbol{R}$, which is defined as $\delta_y(f) := f(y)$, for $f \in \boldsymbol{H}(E)$. According to (26), condition (31) means that δ_y is continuous. The assertion thus follows from Riesz' Theorem (**Theorem 1** with (27)).

Relating to the above theorem, we note that the reproducing kernel K is uniquely determined for an RKHS $\boldsymbol{H}(E)$ (also see Theorem 3 in the next section).

RKHS Examples

1. $\boldsymbol{R}$: Let E be $\{1\}$. Here we identify $\boldsymbol{H}(E)$ with $\boldsymbol{R}$. An arbitrary element of $\boldsymbol{H}(E)$ is a map $f : E \equiv \{1\} \to \boldsymbol{R}$. Specifying $f \in \boldsymbol{H}(E)$ therefore means specifying a real number x with $f(1) = x$. Let the inner product $\langle , \rangle$ for $\boldsymbol{H}(E)$ be the ordinary multiplication in $\boldsymbol{R} : \langle x, y\rangle = x \cdot y$. We define $K : E \times E \to \boldsymbol{R}$ as $K(1,1) := 1$. It is then easy to see K satisfies condition (1) in the definition of the reproducing kernel. As for condition (2), we have: $\langle f(1), K(1,1)\rangle = f(1)1 = f(1)$.
2. l^2: Let $\boldsymbol{a} = (a_j)_{j=1}^{\infty} \in l^2$. Then $\boldsymbol{a}$ defines a map $\alpha : \boldsymbol{N} \to \boldsymbol{R}$ with $\alpha(i) := a_i (i \in \boldsymbol{N})$. We thus identify $\boldsymbol{a} \leftrightarrow \alpha$. By setting $E = \boldsymbol{N}$, we have $\boldsymbol{H}(\boldsymbol{N}) \equiv \{\alpha : \boldsymbol{N} \to \boldsymbol{R} | \sum_{i=1}^{\infty} |\alpha(i)|^2 < \infty\} \cong l^2$ with its kernel function K as being $K(i,j) = \delta_{ij}$.
3. Let A be an n-th order, symmetric, and positive semi-definite matrix. Then $A(\boldsymbol{R}^n)$ is RKHS and its reproducing kernel is A (see the discussions in the next section):

$$A(\boldsymbol{R}^n) \equiv \{Ax \in \boldsymbol{R}^n \mid x \in \boldsymbol{R}^n\}.$$

7.5 Fundamental Properties

Proposition 2. Let $K : E \times E \to \boldsymbol{R}$ be the kernel function of RKHS $\boldsymbol{H}(E)$. Then K is a symmetric, positive semi-definite function.

Proof. (30) in Proposition 1 says that K is symmetric, since the inner product itself is symmetric. For $(x_1, x_2, \cdots, x_n)^T \in E^n$ and $(a_1, a_2, \cdots, a_n)^T \in \boldsymbol{R}^n$, we have, using (30),

$$\begin{aligned}\sum_{i,J=1}^{n} a_i a_j K(x_i, x_j) &= \sum_{i,j=1}^{n} a_i a_j \langle K(x, x_i), K(x, x_j)\rangle_x \\ &= \left\langle \sum_{i=1}^{n} a_j K(x, x_i), \sum_{j=1}^{n} a_j K(x, x_j) \right\rangle_x \\ &= \left\| \sum_{k=1}^{n} a_k K(x, x_k) \right\|_x^2 \geq 0.\end{aligned}$$

The following theorem says that RKHS is constructed by specifying a symmetric, positive semi-definite function. It should also be noted that the proof is constructive so that it might be useful even in our practical situations.

Theorem 3. Suppose that K is a symmetric, positive semi-definite function on $E \times E$. Then there exists a Hilbert space $\boldsymbol{H}$ that has K as its reproducing kernel.

Sketch of the proof. We put $\boldsymbol{F} := \{\sum_{i=1}^{l} \alpha_i K(x, x_i) |\ l \in \boldsymbol{N}, \alpha_i \in \boldsymbol{R}, x_i \in E\}$. By defining addition and multiplication by constant as usual, we can make $\boldsymbol{F}$ a vector space. Also we can introduce the inner product for $f = \sum_{i=1}^{m} \alpha_i K(x, x_i)$ and $g = \sum_{j=1}^{n} \beta_j K(x, x_j) \in \boldsymbol{F}$ as follows: $\langle f, g\rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j K(x_i, x_j) \in \boldsymbol{R}$. Since K is positive semi-definite, it follows that $\langle f, f\rangle \geq 0$. It is easy to see that F is a pre-Hilbert space. Next we set $\boldsymbol{H}$ as the completion[12] of $\boldsymbol{F}$. Then, with $g(x) \equiv g_y(x) = K(x, y)$, we have $\langle (f(x), K(x, y)\rangle_x = \langle f, h_y\rangle = \sum_{i=1}^{m} \alpha_i K(x_i, y) = \sum_{i-1}^{m} \alpha_i K(y, x_i) = f(y)$, for any $f \in \boldsymbol{F}$. This also holds for any f of $\boldsymbol{H}$, because, for any Cauchy sequence $\{f_m\}$ of $\boldsymbol{F}$, we have

$$\begin{aligned}|f_m(y) - f_n(y)| &\leq |\langle (f_m - f_n)(x), K(x, y)\rangle_x| \\ &\leq \|f_m - f_n\|_x \cdot \|K(x, y)\|_x = \|f_m - f_n\| \cdot K(y, y).\end{aligned}$$

This means that $\{f_m(y)\} \subset \boldsymbol{R}$ converges at any y. $\boldsymbol{H}$ therefore includes $f = \lim_{n\to\infty} f_n$, because f also satisfies condition (2) in the definition of reproducing kernel.

One more remark is about an RKHS that has the complete orthonormal system (CONS). Let us describe the reproducing kernel K with CONS $\equiv \{\varphi_j\}_{j=1}^{\infty}$. From condition (1) of the reproducing kernel, we first get

$$K(x, y) = \sum_{i=1}^{\infty} \alpha_i(y) \varphi_i(x).$$

[12]The completion simply means that all the limits of Cauchy sequences of $\boldsymbol{F}$ are included into its completion. For example, for $\boldsymbol{Q}$, the totality of rational numbers, its completion is $\boldsymbol{R}$, the totality of real numbers.

Then, taking $f(y) = \varphi_i(y)$ in condition (2), we have

$$\begin{aligned}\varphi_i(y) = \langle \varphi_i, K(\cdot, y)\rangle = \langle \varphi_i, \sum_{k=1}^{\infty} \alpha_k(y)\varphi_k\rangle \\ = \sum_{k=1}^{\infty} \alpha_k(y)\langle \varphi_i, \varphi_k\rangle = \sum_{k=1}^{\infty} \alpha_k(y)\delta_{ik} = \alpha_i(y).\end{aligned}$$

We therefore have

$$K(x, y) = \sum_{i=1}^{\infty} \varphi_i(x)\varphi_i(y).$$

7.6 RKHS in L^2 space

We briefly describe an infinite-dimensional RKHS in $L^2(E)$. E is assumed to be a domain in $\boldsymbol{R}^n$, and we say L^2 instead of $L^2(E)$ from now on. Supposing that K is a symmetric, positive semi-definite function on $E \times E$, we first define the real-valued function $\kappa(f)$ on $E : \kappa(f)(y) := \langle K(x, y), f(x)\rangle_x$, for any $y \in E$. Let us further suppose that

$$\iint |K(x, x')|^2 dx dx' < \infty. \tag{32}$$

Then κ can be considered as a linear operator: $L^2 \to L^2$, because, using Schwarz' inequality (25), we have

$$\begin{aligned}\int |\kappa(f)(y)|^2 dy &= \int \langle K(x, x'), f(x)\rangle_x^2 dx' \\ &= \int \left(\int K(x, x')f(x)dx \right)^2 dx' \\ &\leq \iint |K(x, x')|^2 dx \int |f(y)|^2 dy dx' \\ &= \iint |K(x, x')|^2 dx dx' \int |f(y)|^2 dy \\ &= \|f\|_{L^2}^2 \cdot \iint |K(x, x')|^2 dx dx' < \infty.\end{aligned}$$

This yields that $\kappa(f) \in L^2$ and that κ is a continuous linear operator, known as *Hilbert-Schmidt integral operator*. According to Mercer's theorem, we then have the eigen decomposition: $K(x, x') = \sum_{\nu \geq 1} \lambda_\nu \phi_\nu(x)\phi_\nu(x')$ where $\nu \in N$, and λ_ν, ϕ_ν are eigen value and eigen functions of κ, respectively. Assumption (32) yields $\sum_{\nu \geq 1} \lambda_\nu^2 < \infty$, so that we have $\lim_{k \to \infty} \lambda_k = 0$. We now assume that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \cdots > 0$. We then know that $\{\phi_n\}_{n=1}^{\infty}$ is a CONS of L^2, which consequently gives the following result:

Theorem 4. Let $\boldsymbol{H}_\kappa$ be the totality of the functions $f \in L^2$, satisfying $\sum_{k\geq 1} \frac{f_k g_k}{\lambda_k} < \infty$. We then have:

1. $\boldsymbol{H}_\kappa$ is a Hilbert space with the inner product: $\langle f, g\rangle_\kappa = \sum_{k\geq 1} \frac{f_k g_k}{\lambda_k} < \infty$
2. For $g \in \boldsymbol{H}_\kappa$, we have
$$\langle g, \phi_\nu\rangle_\kappa = \frac{\langle g, \phi_\nu\rangle_{L^2}}{\lambda_\nu},\ \langle \phi_\nu, \phi_\nu\rangle_\kappa = \frac{1}{\lambda_\nu}\ \text{(i.e., } \|\phi_\nu\|_\kappa = \frac{1}{\sqrt{\lambda_\nu}})\ (\nu = 1, 2, \cdots).$$
3. K is the reproducing kernel of $\boldsymbol{H}_\kappa$: $f(x) = \langle f, K(\cdot, x)\rangle_\kappa$ for any $f \in \boldsymbol{H}_\kappa$.

We skipped the mathematical details and the rigorous proof of the above theorem. Instead, we should keep in mind the relation between $\boldsymbol{H}_\kappa$ and L^2 through the CONS derived from the Hilbert-Schmidt operator κ. We also note that a similar result is obtained in a finite-dimesional case, where K simply means an n-th order symmetric, positive semi-definite matrix and $L^2 \cong \boldsymbol{R}^n$.

7.7 RBF and RKHS

In this section, though a few theorems are stated, we don't give any rigorous explanations and proofs for them. We would just like to sketch the overall flow from the theory to our practice.

7.7.1 Regularization problem in RKHS

For simplicity, let $E = \Omega = \boldsymbol{R}^n$. Suppose that $(\boldsymbol{x}_i, f_i)$ are given as sample points, where $f_i \in \boldsymbol{R},\ \boldsymbol{x}_i \in \boldsymbol{R}^n (i = 1, 2, \cdots, N)$. Let us consider the following regularization problem: Find a function f defined on $\boldsymbol{R}^n$ such that

$$\min_f \left\{ \sum_{i=1}^N (f_i - f(\boldsymbol{x}_i))^2 + \lambda J_m^n(f) \right\}, \tag{33}$$

where

$$J_m^n(f) := \sum_{\alpha_1+\alpha_2+\cdots+\alpha_n=m} \frac{m!}{\alpha_1!\alpha_2!\cdots\alpha_n!} \|D^\alpha f\|_{L^2}^2, \tag{34}$$
$$D^\alpha f := \frac{\partial^m f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}.$$

The regularization term $\lambda J_m^n(f)$ in (33) prescribes smoothness of a solution. Now where can we find a solution of this problem (33)? According to the definition of $J_m^n(f)$, we should find a solution to (33) in the following space:

$$\boldsymbol{B}_m^n := \{f : \boldsymbol{R}^n \to \boldsymbol{R} \cup \{\pm\infty\} | D^\alpha f \in L^2(\boldsymbol{R}^n), \text{for any } \alpha(|\alpha| = m)\}. \tag{35}$$

So recall the thin plate spline case. We then start with $\boldsymbol{B}_2^2$ in (15) to minimize the energy functional (16).

In the following we want to solve the regularization problem like (33) in RKHS. The main reason for this is the following nice property of RKHS:

Representer Theorem[13] Let $\boldsymbol{H}$ be an RKHS, with its reproducing kernel K and norm $\| \|_{\boldsymbol{H}}$. Consider the regularization problem of the following form: Find $f \in \boldsymbol{H}$ such that

$$\min_{f \in \boldsymbol{H}} \left\{ \sum_{i=1}^{N} (f_i - f(\boldsymbol{x}_i))^2 + \lambda \|f\|_{\boldsymbol{H}}^2 \right\}. \tag{36}$$

The solution f can then be found in the form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i). \tag{37}$$

It would therefore be nice, if we could have $\boldsymbol{B}_m^n$ as the RKHS in the above theorem. However, J_m^n cannot be the squared norm for $\boldsymbol{B}_m^n$, as described next.

The functional J_m^n has the following properties:

1. $J_m^n(f) = 0 \;\Leftrightarrow\; f \in \mathcal{P}_{m-1}$ (the totality of at most (m-1)-th order polynomials).
2. $J_m^n(f) = (-1)^m \langle f, \Delta^m f \rangle_{L^2}$.

Since the null space of J_m^n is equal to $\mathcal{P}_{m-1}$, we first represent $\boldsymbol{B}_m^n$ as being the direct sum of the two function spaces: $\boldsymbol{B}_m^n = \boldsymbol{H}_m^n \oplus \mathcal{P}_{m-1}$. Then let us solve the regularization problem on $\boldsymbol{H}_m^n$:

Theorem 5 [37]. If $m > \frac{n}{2}$, *then* $\boldsymbol{H}_m^n$ is an RKHS with:

$$\langle f, g \rangle_{\boldsymbol{H}_m^n} := \sum_{\alpha_1+\alpha_2+\cdots+\alpha_n=m} \frac{m!}{\alpha_1!\alpha_2!\cdots\alpha_n!} \langle D^\alpha f, D^\alpha g \rangle_{L^2} = \langle (-1)^m \Delta^m f, g \rangle_{L^2}. \tag{38}$$

This also means $\|f\|_{\boldsymbol{H}_m^n}^2 = J_m^n(f)$.

With the above theorem, the regularization problem (33) is restated as:

$$\min_{f \in \boldsymbol{B}_m^n} \left\{ \sum_{i=1}^{N} (f_i - f(\boldsymbol{x}_i))^2 + \lambda J_m^n(f) \right\}$$
$$\Leftrightarrow \min_{g \in \boldsymbol{H}_m^n,\; p \in \mathcal{P}_{m-1}} \left\{ \sum_{i=1}^{N} \{f_i - (g(\boldsymbol{x}_i) + p(\boldsymbol{x}_i))\}^2 + \lambda \|g\|_{\boldsymbol{H}_m^n}^2 \right\}. \tag{39}$$

[13] This is one of the variations of the representer theorem. Please refer to [51].

We thus know the solution to (39) is represented in the form of

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + p_{m-1}(\boldsymbol{x}), \tag{40}$$

where $p_{m-1}(\boldsymbol{x}) \in \mathcal{P}_{m-1}$.

7.7.2 RBF as Green's function

Considering the inner product of $\boldsymbol{H}_m^n$ (38) in Theorem 5, we assume that $K(\boldsymbol{x}, \boldsymbol{y}) = G(\boldsymbol{x} - \boldsymbol{y})$. Let G then be a Green's function in the sense that

$$\Delta^m G(\boldsymbol{x}) = \delta(\boldsymbol{x}), \tag{41}$$

where δ means a generalized function as Dirac δ. We therefore have the solution f in (40) as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x} - \boldsymbol{x}_i) + p(\boldsymbol{x}), \tag{42}$$

where p is a polynomial $\in \mathcal{P}_{m-1}$.

This brings us to our familiar class of radial basis functions:

$$G(\boldsymbol{x}) = \begin{cases} \beta_{mn} |\boldsymbol{x}|^{2m-n} \log |\boldsymbol{x}| & \text{if } 2m - n \text{ is an even integer,} \\ \gamma_{mn} |\boldsymbol{x}|^{2m-n} & \text{otherwise,} \end{cases} \tag{43}$$

where β_{mn} and γ_{mn} are constants.

For example, for the thin plate spline, we have $m = n = 2$ so that $G(\boldsymbol{x}) = |\boldsymbol{x}|^2 \log |\boldsymbol{x}|$ and the polynomial p in (42) is of the first order (linear). Another familiar case is where $m = 2$ and $n = 3$. The we have $G(\boldsymbol{x}) = |\boldsymbol{x}|$ and a linear polynomial for (42).

Regularization with RKHS norm Let us consider another regularization problem, where, instead of $J_m^n(f)$ in (33) and (34), we take $\sum_{m \geq 0} a_m J_m^n(f)$ with a_m being constants and $a_0 \neq 0$. According to the scenario of establishing the above theorems, we have the following result in which the regularization problem is directly solved in an RKHS.

1. We can find a Green's function for the operator $\sum_{m \geq 0} (-1)^m \Delta^m$. The solution is then given by $f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k G(\boldsymbol{x} - \boldsymbol{x}_k)$. Note that this time we don't need a polynomial term like (42).
2. *Gaussian RBF.* In a particular case, where $a_m = \frac{\sigma^{2m}}{m! 2^m} (\sigma > 0)$, we have the Green's function as $G(\boldsymbol{x}) = c \exp(-\frac{\|\boldsymbol{x}\|^2}{2\sigma^2})$.

As described in 2.7, Gaussian Process regression (GPR) can also be used for scattered data interpolation and extrapolation. It should then be noted that we can interpret both of the techniques for GPR and RBF in the common framework of RKHS. Please refer to [1] for this interesting issue.

8 Conclusion

8.1 Comparison and Performance

Several general statements about the performance of the various methods can be made.

- Radial Basis and Wiener interpolation both generally require solving a linear system to determine the weights w_k. This typically requires $O(N^3)$ time (the matrix is symmetric so Choleski can be used), but this can be done as a setup or "training" precomputation.
- Shepard's method is simple and has no setup or training cost. On the other hand it provides rather poor interpolation.
- All methods except for moving least squares have a runtime cost that is linear in the number of data points.
- Moving least squares have no training cost, but require solving a linear system at runtime. On the other hand this system may involve a small subset of the total number of data points.
- For the roughness penalty methods that have equivalent RBF and relaxation formulations, the RBF approach scales with the number of known data points, whereas the direct (Laplace/Poisson) approach scales with the number of unknown points (i.e. locations to be synthesized). In both cases the scaling is typically cubic in the number of points. This can be reduced to linear time with multipole methods in the case of an RBF approach, and multigrid in the case of direct approaches. Laplace interpolation is poor (e.g. it is piecewise linear in the one dimensional case). The iterated Laplacian provides better interpolation but is more difficult to solve with fast (multigrid) methods.

8.2 Further Readings

There are several recent books [12, 64, 19] covering scattered data interpolation topics though unfortunately all of them require a level of mathematics well beyond that required for this course. [19] is perhaps the most accessible. Radial basis and other interpolation methods are also covered in machine learning texts [9] since they can be used for regression and classification.

Acknowledgments

We thank Jean-Marie Aubry, Geoffrey Irving and Armin Iske for discussions of several topics.

9 Appendices

A Python program for Laplace interpolation

```
def solve(d,constrained):

    n = d.size
    nc = constrained.size
    A00 = sparse.lil_matrix((n,n))
    A = sparse.lil_matrix((n+nc,n+nc))

    A00.setdiag(-2.*n_.ones(n))
    koff = 1
    A00.setdiag(n_.ones(n),koff)
    A00.setdiag(n_.ones(n),-koff)
    A00[0,0] = -1.
    A00[n-1,n-1] = -1.

    A[0:n,0:n] = A00

    S = sparse.lil_matrix((nc,n))
    for ir in range(nc):
        S[ir,constrained[ir]] = 1.
    St = S.T

    A[0:n,n:(n+nc)] = St
    A[n:(n+nc),0:n] = S

    A = A.tocsr()

    b = n_.zeros((n+nc,1))
    for i in range(nc):
        b[n+i] = d[constrained[i]]

    f = linsolve.spsolve(A,b)
    f = f[0:n]

    return f
```

```
def test():
    x = n_.array([2.,4., 6.,  8.,  13., 14.])
    y = n_.array([2.,7., 8.,  9.,  12., 12.5])
    n = x.size
    LEN = 200
    x = x * LEN / 20.

    X = n_.arange(float(LEN))
    Y = n_.zeros((LEN,))
    for i in range(x.size):
        ii = int(x[i])
        Y[ii] = y[i]

    f1 = solve(Y,x)
```

References

[1] Ken Anjyo and J.P. Lewis. RBF interpolation and Gaussian process regression through an RKHS formulation. *Journal of Math-for-Industry.*, 3:63–71, 2011.

[2] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[3] W. Baxter and K. Anjyo. Latent doodle space. *Computer Graphics Forum*, 25:477–485, 2006.

[4] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000.

[5] R.K. Beatson, J.B. Cherrie, and C.T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned gmres iteration. *Advances in Computational Mathematics*, 11:253– 270, 1999.

[6] Julien Cohen Bengio and Rony Goldenthal. Simplicial interpolation for animating The Hulk. In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13, pages 7:1–7:1, New York, NY, USA, 2013. ACM.

[7] S. Bergman. The kernel function and the conformal mapping. *Mathematical Surveys*, 5, 1950.

[8] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph.*, 26(3):33, 2007.

[9] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[11] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232, New York, NY, USA, 2000. ACM.

[12] Martin D. Buhmann. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, 2003.

[13] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM.

[14] Z.-Q. Cheng, Y.-Z. Wang, K. Xu B. Li, G. Dang, and S.Y.-Jin. A survey of methods for moving least squares surfaces. In *IEEE/EG Symposium on Volume and Point-Based Graphics*, 2008.

[15] Scott Davies. Multidimensional triangulation and interpolation for reinforcement learning. In *NIPS*, pages 1005–1011, 1996.

[16] Gianluca Donato and Serge Belongie. Approximate thin plate spline mappings. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 21–31, London, UK, UK, 2002. Springer-Verlag.

[17] Gregory Fasshauer. Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11:139–159, 1999. 10.1023/A:1018919824891.

[18] Gregory E. Fasshauer and Larry L. Schumaker. Scattered data fitting on the sphere. In *Proceedings of the international conference on Mathematical methods for curves and surfaces II Lillehammer, 1997*, pages 117–166, Nashville, TN, USA, 1998. Vanderbilt University.

[19] Gregory F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[20] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 544–552, New York, NY, USA, 2005. ACM.

[21] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 15:1691–1704, 1980.

[22] Edward J. Fuselier. *Refined error estimates for matrix-valued radial basis functions*. PhD thesis, Texas A & M University, College Station, Texas, 2006.

[23] Edward J. Fuselier and Grady B. Wright. Stability and error estimates for vector field interpolation and decomposition on the sphere with rbfs. *SIAM J. Numer. Anal.*, 47(5):3213–3239, 2009.

[24] Eitan Grinspun, Mathieu Desbrun, and et al. Discrete differential geometry: An applied introduction. SIGGRAPH Course, 2006.

[25] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[26] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, New York, NY, 2009.

[27] T. Hatanaka, N. Kondo, and K. Uosaki. Multi-objective structure selection for radial basis function networks based on genetic algorithm. In *The 2003 Congress onEvolutionary Computation, 2003. CEC '03.*, pages 1095 – 1100 Vol.2.

[28] Y. C. Hon, R. Schaback, and X. Zhou. Adaptive greedy algorithm for solving large RBF collocation problems. *Numer. Algorithms*, 32:13–25, 2003.

[29] Pushkar Joshi, Wen C. Tien, Mathieu Desbrun, and Frédéric Pighin. Learning controls for blend shape based realistic facial animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 187–192, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[30] Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH Symposium on Computer Animation (SCA-04)*, pages 357–366, 2004.

[31] Gene S. Lee. Evaluation of the radial basis function space. In *ACM SIGGRAPH ASIA 2009 Sketches*, SIGGRAPH ASIA '09, pages 42:1–42:1, New York, NY, USA, 2009. ACM.

[32] J. Lewis, H.-J. Hwang, U. Neumann, and R. Enciso. Smart point landmark distribution for thin-plate splines. In *Proc. SPIE Medical Imaging*, pages 1236–1243, San Diego, 2004.

[33] J. P. Lewis. Generalized stochastic subdivision. *ACM Transactions on Graphics*, 6(3):167–190, July 1987.

[34] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer*

graphics and interactive techniques, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[35] J.P. Lewis. Lifting detail from darkness. In *SIGGRAPH '01: Proceedings of the SIGGRAPH 2001 conference Sketches & applications*. ACM Press, 2001.

[36] J.P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. In *Eurographics*, 2014.

[37] J. Meinguet. Multivariate interpolation at arbitrary points made simple. *Z. Angew. Math.*, 30:292–304, 1979.

[38] Francis J. Narcowich and Joseph D. Ward. Generalized hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comput.*, 63(208):661–687, 1994.

[39] Jun-yong Noh and Ulrich Neumann. Expression cloning. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 277–288, New York, NY, USA, 2001. ACM.

[40] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[41] Jason Osipa. *Stop Staring: Facial Modeling and Animation Done Right, 2nd Ed.* Sybex, 2007.

[42] Sung W. Park, Lars Linsen, Oliver Kreylos, John D. Owens, and Bernd Hamann. Discrete Sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):243–253, 2006.

[43] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[44] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1998. ACM.

[45] Taehyun Rhee, Youngkyoo Hwang, James Dokyoon Kim, and Changyeong Kim. Real-time facial animation from live video tracking. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 215–224, New York, NY, USA, 2011. ACM.

[46] Taehyun Rhee, J.P. Lewis, Ulrich Neumann, and Krishna Nayak. Scan-based volume animation driven by locally adaptive articulated registrations. *IEEE Trans. Visualization and Computer Graphics*, 2011.

[47] C.F. Rose, I. Peter-Pike, J. Sloan, and M.F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. In *EUROGRAPHICS*, 2001.

[48] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[49] M. Sato. Theory of hyperfunctions. I, *Fac. Sci. Univ. Tokyo* Sect. 1, 8, 139-193, 1959; II, ibid, 8, 387-437, 1960.

[50] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 533–540, New York, NY, USA, 2006. ACM.

[51] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[52] L. Schwartz. *Théory des Distributions, vol.I et II.* Hermann, 1950, 1951.

[53] Jaewoo Seo, Geoffrey Irving, J. P. Lewis, and Junyong Noh. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph.*, 30(6):164:1–164:10, December 2011.

[54] Yeongho Seol, J.P. Lewis, Jaewoo Seo, Byungkuk Choi, Ken Anjyo, and Junyong. Noh. Spacetime expression cloning for blendshapes. *ACM Trans. Graph.*, 31(2): Article 14, 2012.

[55] Yeongho Seol, Jaewoo Seo, Paul Hyunjin Kim, J. P. Lewis, and Junyong Noh. Weighted pose space editing for facial animation. *The Visual Computer*, 28(3):319–327, 2012.

[56] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *ACM '68: Proceedings of the 1968 23rd ACM national conference*, pages 517–524, New York, NY, USA, 1968. ACM.

[57] Yuhei Shibukawa, Yoshinori Dobashi, and Tsuyoshi Yamamoto. Interactive editing of volumetric objects by using feature-based transfer function. In *Mathematical Progress in Expressive Image Synthesis I*, pages 55–62, Tokyo, JAPAN, 2014. Springer Verlag.

[58] Robert Sibson. A brief description of natural neighbor interpolation. In V. Barnett, editor, *Interpreting Multivariate Data*, chapter 2, pages 21–36. John Wiley, Chichester, 1981.

[59] Peter-Pike J. Sloan, Charles F. Rose, and Michael F. Cohen. Shape by example. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM Press.

[60] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[61] Hideki Todo, Ken Anjyo, Williams Baxter, and Takeo Igarashi. Locally controllable stylized shading. *ACM Trans. Graph.*, 26(3): Article17, 2007.

[62] Karel Uhlir and Vaclav Skala. Radial basis function use for the restoration of damaged images. In Max Viergever, K. Wojciechowski, B. Smolka, H. Palus, R.S. Kozera, W. Skarbek, and L. Noakes, editors, *Computer Vision and Graphics*, volume 32 of *Computational Imaging and Vision*, pages 839–844. Springer Netherlands, 2006.

[63] R. Peter Weistroffer, Kristen R. Walcott, Greg Humphreys, and Jason Lawrence. Efficient basis decomposition for scattered reflectance data. In *EGSR07: Proceedings of the Eurographics Symposium on Rendering*, Grenoble, France, June 2007.

[64] Holger Wendland. *Scattered Data Approximation.* Cambridge, 2005.

[65] Holger Wendland. Divergence-free kernel methods for approximating the Stokes problem. *SIAM J. Numer. Anal.*, 47(4):3158–3179, 2009.

[66] Lexing Ying. Short note: A kernel independent fast multipole algorithm for radial basis functions. *J. Comput. Phys.*, 213(2):451–457, 2006.

[67] Yuanchen Zhu and Steven J. Gortler. 3d deformation using moving least squares. Harvard Computer Science Technical Report: Tr-10-07. Technical report, Harvard University, Cambridge, MA, 2007.

[68] Todd Zickler, Ravi Ramamoorthi, Sebastian Enrique, and Peter N. Belhumeur. Reflectance sharing: Predicting appearance from a sparse set of images of a known shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1287–1302, 2006.