MATH 590: Meshfree Methods

Chapter 1 — Part 2: Scattered Data Interpolation in $\mathbb{R}^d$

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2014

Outline

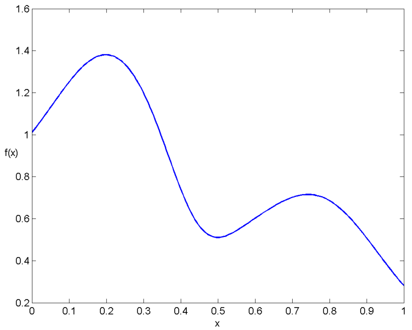
Univariate Functions

Start from Calculus:

$$f : x \mapsto f(x), \quad x \in [a, b], \; f(x) \in \mathbb{R}$$

Example, $[a, b] = [0, 1]$:

$$\begin{aligned} f(x) &= \frac{3}{4} e^{-((9x-2)^2)/4} \\ &+ \frac{3}{4} e^{-((9x+1)^2/49)} \\ &+ \frac{1}{2} e^{-((9x-7)2)/4} \\ &- \frac{1}{5} e^{-((9x-4)^2)} \end{aligned}$$



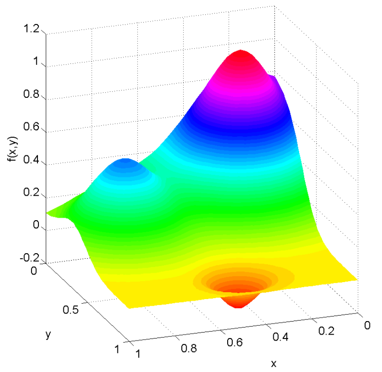
Multivariate Functions

From multivariable Calculus:

$$f : \boldsymbol{x} \mapsto f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d, f(\boldsymbol{x}) \in \mathbb{R}$$

Example, $\Omega = [0,1]^2$, $\boldsymbol{x} = (x, y)$:

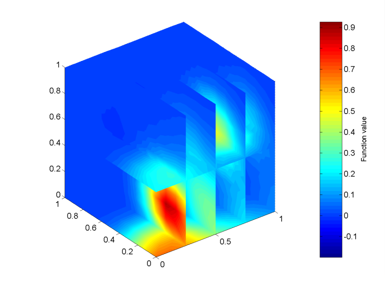
$$\begin{aligned} f(\boldsymbol{x}) \quad = \quad & \frac{3}{4} e^{-((9x-2)^2+(9y-2)^2)/4} \\ & + \frac{3}{4} e^{-((9x+1)^2/49+(9y+1)^2/10)} \\ & + \frac{1}{2} e^{-((9x-7)^2+(9y-3)^2)/4} \\ & - \frac{1}{5} e^{-((9x-4)^2+(9y-7)^2)} \end{aligned}$$



More Multivariate Functions

Example, $\Omega = [0,1]^3$, $\boldsymbol{x} = (x, y, z)$:

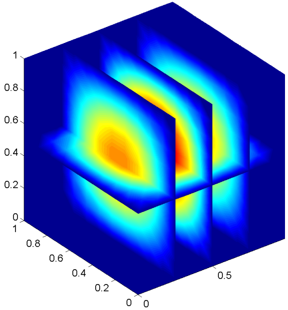
$$
\begin{aligned}
f(\boldsymbol{x}) \;=\; & \frac{3}{4} e^{-((9x-2)^2+(9y-2)^2+(9z-2)^2)/4} \\
& + \frac{3}{4} e^{-((9x+1)^2)/49-((9y+1)^2)/10-((9z+1)^2)/25} \\
& + \frac{1}{2} e^{-((9x-7)^2+(9y-3)^2+(9z-5)^2)/4} \\
& - \frac{1}{5} e^{-((9x-4)^2+(9y-7)^2+(9z-5)^2)}
\end{aligned}
$$



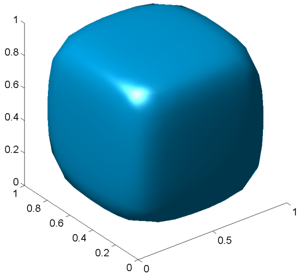
More Multivariate Functions

Example, $\Omega = [0,1]^3$, $\boldsymbol{x} = (x, y, z)$:

$$f(\boldsymbol{x}) = 64x(1-x)y(1-y)z(1-z)$$

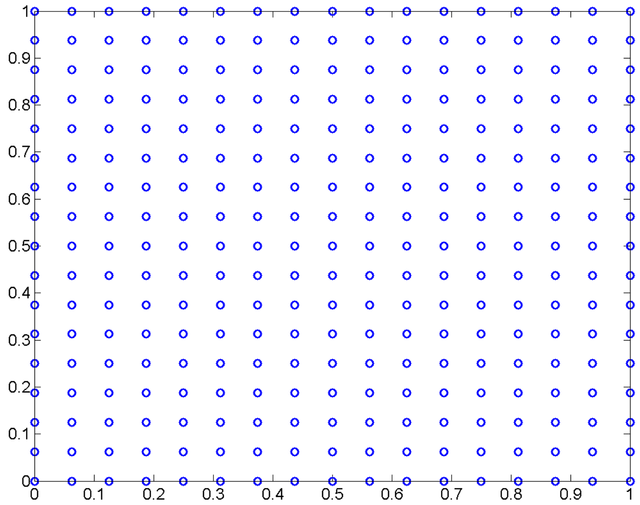


Slice plot

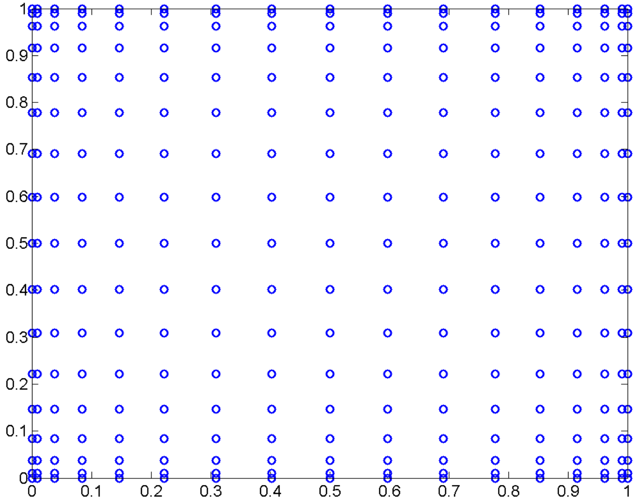


Isosurface plot

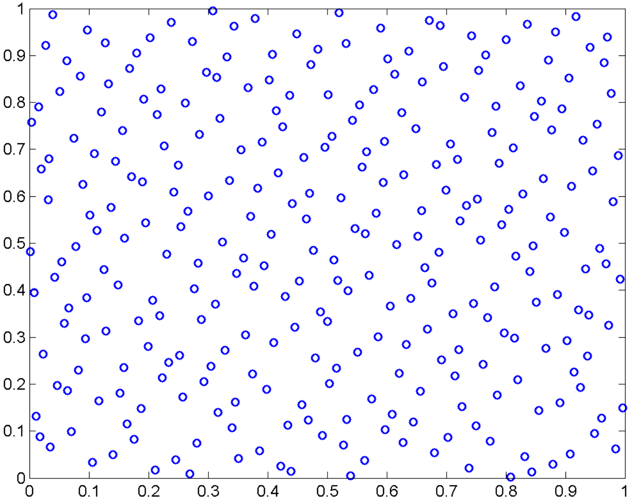
289 uniformly gridded data sites in 2D



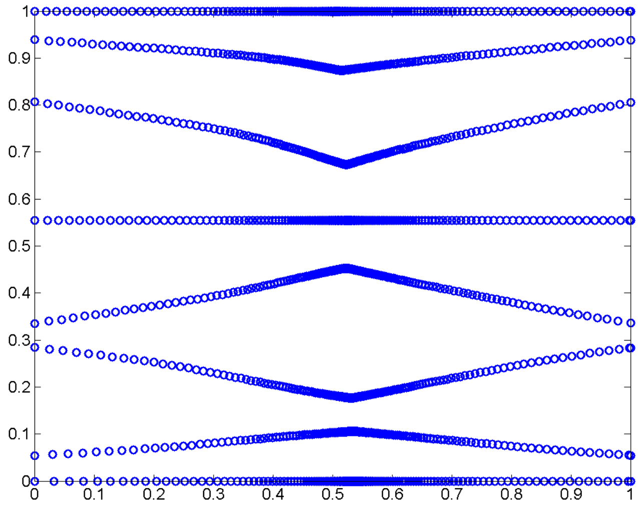
289 Chebyshev data sites in 2D



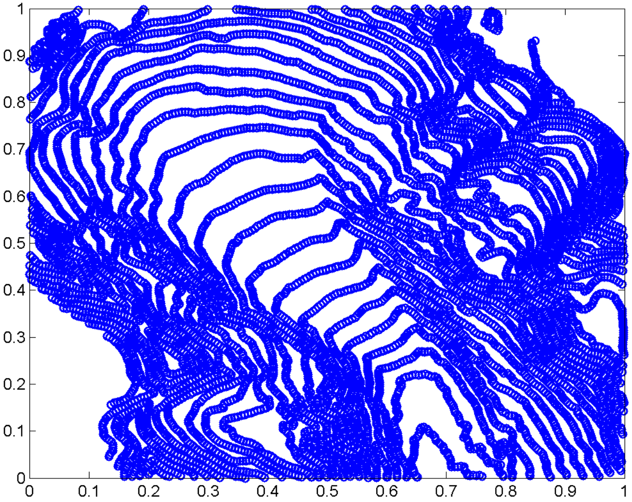
289 Halton data sites in 2D



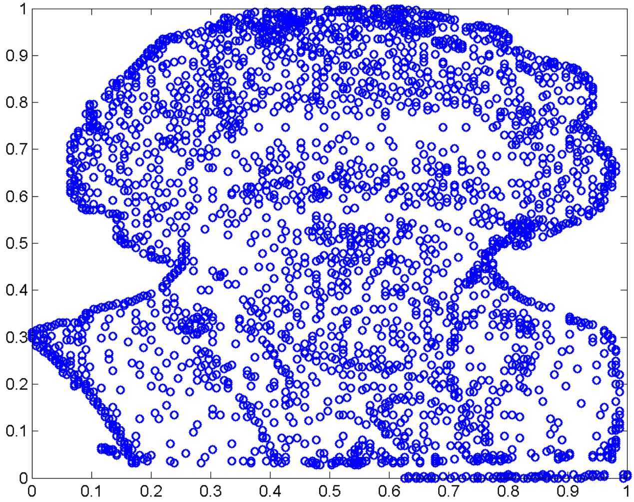
1368 track data sites in 2D



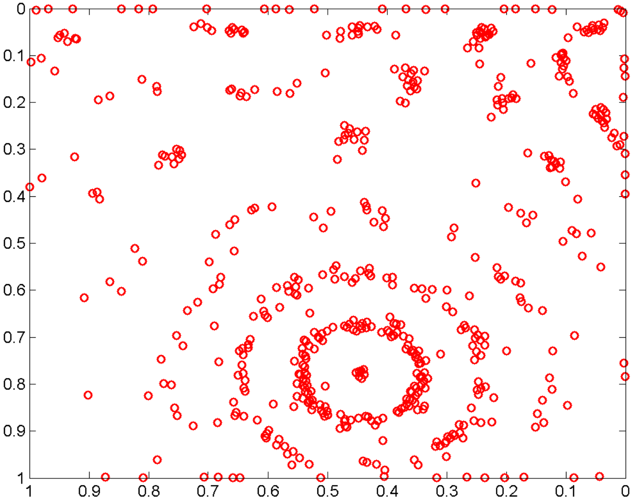
8345 glacier data sites in 2D



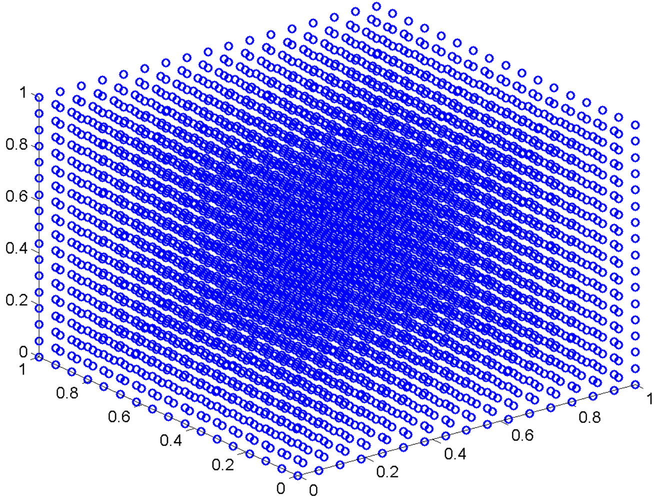
2663 Beethoven data sites in 2D



1000 “optimal” data sites in 2D

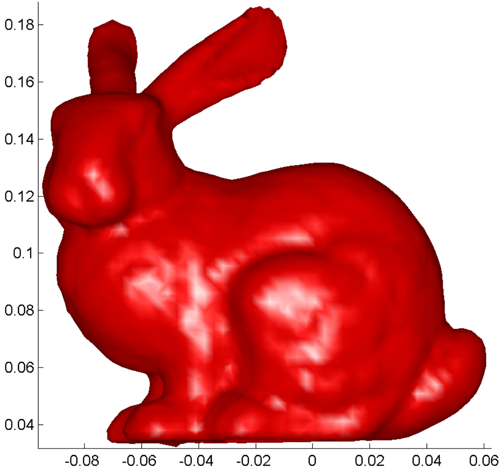


4913 uniformly gridded data sites in 3D

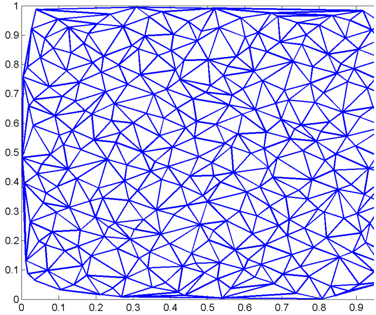


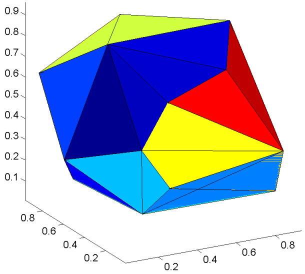
Point Cloud Data

Stanford bunny (simplified): 8171 point cloud data in 3D



Traditional Methods use Meshes





$N = 289$, Delaunay triangulation for bivariate splines, FEMs

$N = 27$, Delaunay tetrahedra for trivariate splines, FEMs

Scattered Data Fitting

- Scattered data fitting is a fundamental problem in approximation theory, statistics and data modeling in general.
- It generalizes the simple (polynomial) interpolation and approximation methods you are familiar with from a basic course in numerical analysis.
- Mathematical challenge: we want a well-posed problem formulation that works
  - for arbitrary space dimensions d and
  - for arbitrary number and location of data points.
- This will naturally lead to distance matrices.
- Later we generalize to radial basis functions or positive definite kernels

The problem in "plain English"

- Given a set of data (measurements, and locations at which these measurements were obtained), we want to find a rule which allows us to deduce information about the process we are studying also at locations different from those at which we obtained our measurements.

Example

- 1D data: a series of measurements taken over a certain time period
- 2D data: produce some sort of weather map based on data collected at weather stations
- 3D data: temperature distribution inside some solid body
- high-D data: often from computer experiments, in finance, optimization, economics, statistics, learning theory.

The problem in "plain English" (cont.)

- We want to find a function s which is a "good" fit to the given data.
- Such a function is often referred to as
  - surrogate model,
  - simulation metamodel,
  - response surface.
- What do we mean by "good"?
  - We may want the function s to exactly match the given measurements at the corresponding locations
    $\longrightarrow$ (scattered data) interpolation
  - We may want the function s to approximately match the given measurements at the corresponding locations
    $\longrightarrow$ (scattered data) approximation such as least squares (with and without noise)
- We will mostly concentrate on the no-noise, interpolation setting.

Mathematical description

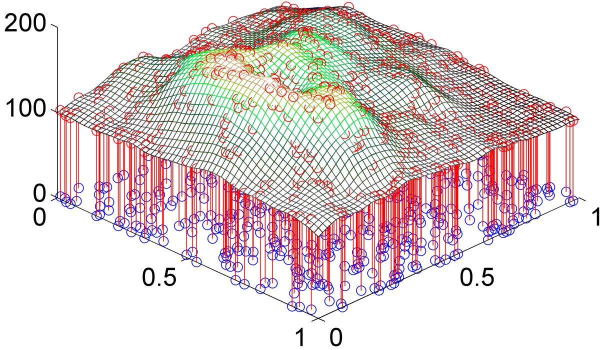
- Assume we are given
  - measurement locations (data sites): $\mathcal{X} = \{\boldsymbol{x}_i,\ i = 1, \ldots, N\} \subset \Omega \subset \mathbb{R}^d$
  - corresponding measurements (data values): $y_i \in \mathbb{R}$
- Later we often assume the data are obtained by sampling some (unknown) function f at the data sites, i.e., $y_i = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$.
- Notation[1] for interpolating function: s

Problem (Scattered Data Interpolation)

Given data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, N$, *with* $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, *find a (continuous) function* s *such that* $s(\boldsymbol{x}_i) = y_i$, $i = 1, \ldots, N$.

[1] Note that this is different from the notation used in [Fas07]. It is in line with [FM15].

Scattered Data Interpolation



Standard setup

A convenient and common approach:
Assume s is a linear combination of certain basis functions B_j, i.e.,

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j B_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

Solving the interpolation problem under this assumption leads to a system of linear equations of the form

$$\mathsf{B}\boldsymbol{c} = \boldsymbol{y},$$

where the entries of the interpolation matrix B are given by $\mathsf{B}_{ij} = B_j(\boldsymbol{x}_i)$, $i, j = 1, \ldots, N$, $\boldsymbol{c} = [c_1, \ldots, c_N]^T$, and $\boldsymbol{y} = [y_1, \ldots, y_N]^T$.

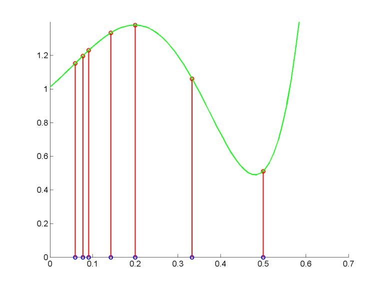
Standard setup (cont.)

The scattered data fitting problem will be well-posed, i.e.,

- a solution to the problem will exist and be unique,

if and only if the matrix B is non-singular.

In 1D it is well known that one can interpolate to arbitrary data at N distinct data sites using a polynomial of degree $N - 1$.



If the dimension is higher, there is the following negative result (see [Haa18, Mai56, Cur59]).

Theorem (Haar–Mairhuber–Curtis)

If $\Omega \subset \mathbb{R}^d$, $d \geq 2$, contains an interior point, then there exist no Haar spaces of continuous functions except for trivial ones, i.e., spaces spanned by a single function.

In order to understand this theorem we need

Definition

Let the finite-dimensional linear function space $\mathcal{B} \subseteq C(\Omega)$ have a basis $\{B_1, \ldots, B_N\}$. Then $\mathcal{B}$ is a *Haar space* on Ω if

$$\det \mathsf{B} \neq 0$$

for any set of distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ in Ω. Here B is the square matrix with entries $\mathsf{B}_{ij} = B_j(\boldsymbol{x}_i)$, $i, j = 1, \ldots, N$.

Existence of a Haar space guarantees invertibility of the interpolation matrix B, i.e., existence and uniqueness of an interpolant of the form (1) to data specified at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from the space $\mathcal{B}$.

Example

Univariate polynomials of degree $N-1$ form an N-dimensional Haar space for data given at $x_1, \ldots, x_N$.

Interpretation of Haar-Mairhuber-Curtis

The HMC theorem tells us that if we want to have a well-posed multivariate scattered data interpolation problem we can no longer fix in advance the set of basis functions we plan to use for interpolation of arbitrary scattered data.

Instead, the basis should depend on the data locations.

Example

It is in general not clear how to perform unique interpolation with (multivariate) polynomials of degree N to data given at arbitrary locations in $\mathbb{R}^2$.
One prescription for obtaining such a unique polynomial is given by the de Boor–Ron least polynomial interpolant (see [BR90, BR92b, BR92a, NX12]).

Proof of Haar–Mairhuber–Curtis

Proof.

Let $d \geq 2$ and assume that $\mathcal{B}$ is a Haar space with basis $\{B_1, \ldots, B_N\}$ with $N \geq 2$.
We need to show that this leads to a contradiction.
By the definition of a Haar space

$$\det\left(B_j(\boldsymbol{x}_i)\right) \neq 0 \qquad (2)$$

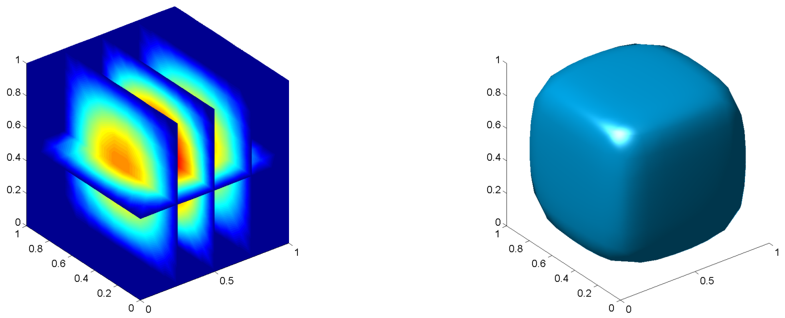
for any distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.

▸ Haar–Mairhuber–Curtis CDF

Since the determinant is a continuous function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we must have had $\det \mathsf{B} = 0$ at some point along the path. This contradicts (2). □

We want to work with a data dependent approximation space as suggested by the Haar-Mairhuber-Curtis theorem.
"Test function"

$$f_d(\boldsymbol{x}) = 4^d \prod_{\ell=1}^{d} x_\ell(1 - x_\ell), \qquad \boldsymbol{x} = (x_1, \ldots, x_d) \in [0,1]^d$$

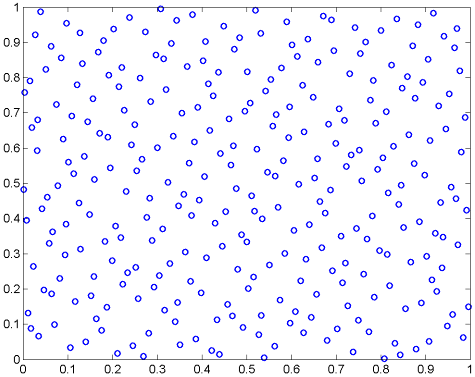


MATLAB code for test function

Program (testfunction.m)

```
% tf = testfunction(d,points)
% Evaluates testfunction
% prod_{l=1}^d x_l*(1-x_l)  (normalized so that max=1)
% at d-dimensional points
function tf = testfunction(d,points)
tf = 4^d*prod(points.*(1-points),2);
```

Sample the testfunction f_d at scattered data sites in the unit cube. We use Halton points (could also use random, Sobol′ or some other space filling design).



We can generate Halton points using `haltonset` from MATLAB's `Statistics Toolbox`.

Halton Points

Halton points (see [Hal60, WLH97]) are created from van der Corput sequences.

Construction of a van der Corput sequence:

Start with unique decomposition of an arbitrary $n \in \mathbb{N}_0$ with respect to a prime base p, i.e.,

$$n = \sum_{i=0}^{k} a_i p^i,$$

where each coefficient a_i is an integer such that $0 \leq a_i < p$.

Example

Let $n = 10$ and $p = 3$. Then

$$10 = 1 \cdot 3^0 + 0 \cdot 3^1 + 1 \cdot 3^2,$$

so that $k = 2$ and $a_0 = a_2 = 1$ and $a_1 = 0$.

Next, define $h_p : \mathbb{N}_0 \to [0, 1)$ via

$$h_p(n) = \sum_{i=0}^{k} \frac{a_i}{p^{i+1}}$$

Example

$$h_3(10) = \frac{1}{3} + \frac{1}{3^3} = \frac{10}{27}$$

$h_{p,N} = \{h_p(n) : \; n = 0, 1, 2, \ldots, N\}$ is called van der Corput sequence

Example

$$h_{3,15} = \{0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \frac{10}{27}, \frac{19}{27}, \frac{4}{27}, \frac{13}{27}, \frac{22}{27}, \frac{7}{27}\}$$

Generation of Halton point set in $[0,1)^d$:

- take d (usually distinct) primes $p_1, \ldots, p_d$
- determine corresponding van der Corput sequences $h_{p_1,N}, \ldots, h_{p_d,N}$
- form d-dimensional Halton points by taking van der Corput sequences as coordinates:

$$H_{d,N} = \{(h_{p_1}(n), \ldots, h_{p_d}(n)) : \; n = 0, 1, \ldots, N\}$$

set of $N+1$ Halton points

Some properties of Halton points

- Halton points are *nested* point sets, i.e., $H_{d,M} \subset H_{d,N}$ for $M < N$
- Can even be constructed sequentially
- In low space dimensions, the multi-dimensional Halton sequence quickly "fills up" the unit cube in a well-distributed pattern
- For higher dimensions ($d \approx 40$) Halton points are well distributed only if N is large enough

We want to construct a (continuous) function s that interpolates the samples obtained from f_d at the set $H_{d,N}$ of Halton points, i.e., want

$$s(\boldsymbol{x}_j) = f_d(\boldsymbol{x}_j), \qquad \boldsymbol{x}_j \in H_{d,N}$$

Assume for now that $d = 1$.

- For small N one can use univariate polynomials
- If N is relatively large it's better to use splines
- Simplest approach: C^0 piecewise linear splines ("connect the dots")

Basis for space of piecewise linear interpolating splines:

$$\{B_j = |\cdot - x_j| : \ j = 1, \ldots, N\}$$

So

$$s(x) = \sum_{j=1}^{N} c_j |x - x_j|, \qquad x \in [0, 1]$$

and c_j determined by interpolation conditions

$$s(x_i) = f_1(x_i), \qquad i = 1, \ldots, N$$

- Clearly, the basis functions $B_j = |\cdot - x_j|$ are dependent on the data sites x_j as suggested by Haar–Mairhuber–Curtis

▸ Haar–Mairhuber–Curtis CDF

- $B(x) = |x|$ is called basic function
  $K(x, z) = |x - z|$ would be the kernel
- The points x_j to which the basic function is shifted to form the basis functions are usually referred to as centers or *knots*.
- Technically, one could choose these centers different from the data sites. However, usually centers coincide with the data sites.
- This simplifies the analysis of the method, and is sufficient for many applications.
- In fact, relatively little is known about the case when centers and data sites differ.
- B_j are (radially) symmetric about their centers x_j
  $\longrightarrow$ radial basis function
- Formal introduction of radial functions later

Now the coefficients c_j in the scattered data interpolation problem are found by solving the linear system

$$\begin{bmatrix} |x_1 - x_1| & |x_1 - x_2| & \dots & |x_1 - x_N| \\ |x_2 - x_1| & |x_2 - x_2| & \dots & |x_2 - x_N| \\ \vdots & \vdots & \ddots & \vdots \\ |x_N - x_1| & |x_N - x_2| & \dots & |x_N - x_N| \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_N) \end{bmatrix} \tag{3}$$

- The matrix in (3) is a distance matrix
- Distance matrices have been studied in geometry and analysis in the context of isometric embeddings of metric spaces for a long time (see, *e.g.*, [Bax91, Blu38, Boc41, Mic86, Sch38, WW76]).
- It is known that the distance matrix based on the Euclidean distance between a set of distinct points in $\mathbb{R}^d$ is always non-singular (more details later).
- Therefore, our scattered data interpolation problem is well-posed.

Since distance matrices are non-singular for Euclidean distances in any space dimension d we have an immediate generalization:
For the scattered data interpolation problem on $[0,1]^d$ we can take

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \|\boldsymbol{x} - \boldsymbol{x}_j\|_2, \qquad \boldsymbol{x} \in [0,1]^d, \tag{4}$$

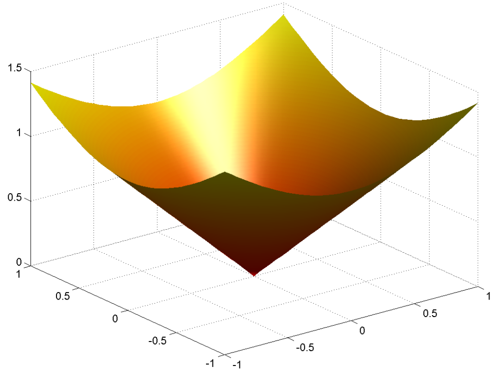
and find the c_j by solving

$$\begin{bmatrix} \|\boldsymbol{x}_1 - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_1 - \boldsymbol{x}_N\|_2 \\ \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_2 - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_2 - \boldsymbol{x}_N\|_2 \\ \vdots & \vdots & \ddots & \vdots \\ \|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_N - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_N - \boldsymbol{x}_N\|_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_d(\boldsymbol{x}_1) \\ f_d(\boldsymbol{x}_2) \\ \vdots \\ f_d(\boldsymbol{x}_N) \end{bmatrix}.$$

- Note that the basis is again data dependent
- For $d > 1$ $\text{span}\{\|\cdot - \boldsymbol{x}_j\|_2,\ j = 1, \dots, N\}$ is not piecewise linear
- Piecewise linear splines in higher space dimensions are usually constructed differently (via a cardinal basis on an underlying computational mesh)

Norm RBF

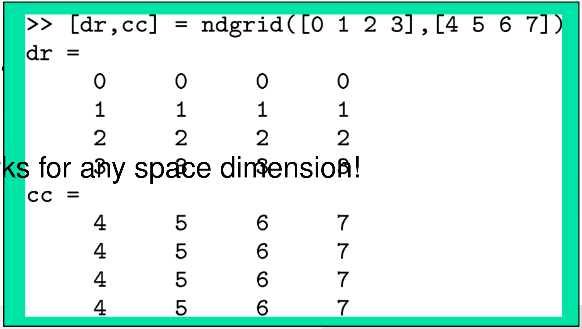
A typical basis function for the Euclidean distance matrix fit, $B_j(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_j\|_2$ with $\boldsymbol{x}_j = \boldsymbol{0}$ and $d = 2$.



One of our main MATLAB subroutines

- Forms the matrix of pairwise Euclidean distances of two (possibly different) sets of points in $\mathbb{R}^d$ (`dsites` and `ctrs`).

```
function DM = DistanceMatrixBook(dsites,ctrs)
[M,d] = size(dsites); [N,d] = size(ctrs);
DM = zeros(M,N);
for l=1:d
   [dr,cc] = ndgrid(dsites(:,l),ctrs(:,l));
   DM = DM + (dr-cc).^2;
end
DM = sqrt(DM);
```



```
>> [dr,cc] = ndgrid([0 1 2 3],[4 5 6 7])
dr =
     0     0     0     0
     1     1     1     1
     2     2     2     2
     3     3     3     3
cc =
     4     5     6     7
     4     5     6     7
     4     5     6     7
     4     5     6     7
```

Works for any space dimension!

Alternate forms of `DistanceMatrix.m`

Program (`DistanceMatrixRepmat.m`)

```
function DM = DistanceMatrixRepmat(dsites,ctrs)
[M,d] = size(dsites); [N,d] = size(ctrs);
DM = zeros(M,N);
for l=1:d
    DM = DM + (repmat(dsites(:,l),1,N) - ...
                   repmat(ctrs(:,l)',M,1)).^2;
end
DM = sqrt(DM);
```

Note: This is more efficient (memory and speed) than the `ndgrid`-based version

Alternate forms of `DistanceMatrix.m` (cont.)

Program (`DistanceMatrixA.m`)

```
function DM = DistanceMatrixA(dsites,ctrs)
M = size(dsites,1); N = size(ctrs,1);
T1 = sum(dsites.*dsites,2);
T2 = -2*dsites*ctrs';
T3 = (sum(ctrs.*ctrs,2))';
DM = sqrt(T1(:,ones(N,1)) + T2 + T3(ones(M,1),:));
```

Note: suggested by a former student of MATH 590 – even faster since no `for`-loop used

Alternate forms of `DistanceMatrix.m` (cont.)

Program (`DistanceMatrix.m`)

```
function DM = DistanceMatrix(dsites,ctrs)
 M = size(dsites,1); N = size(ctrs,1);
DM = repmat(sum(dsites.*dsites,2),1,N) - ...
     2*dsites*ctrs' + ...
     repmat((sum(ctrs.*ctrs,2))',M,1);
DM = sqrt(DM);
```

Note: `repmat` version of previous code (possibly the fastest and most memory efficient)

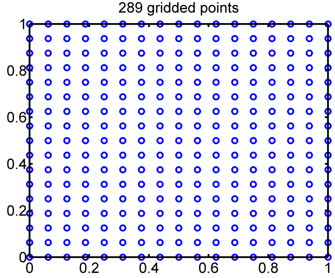
Remark

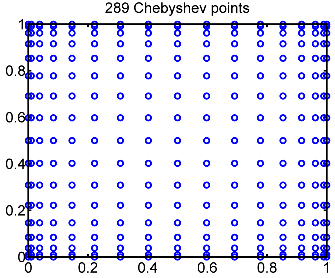
- *Note that the first two subroutines can easily be modified to produce a p-norm distance matrix by making the obvious changes to the code.*
- *The first two subroutines can also be used in a straightforward way to create more general interpolation matrices for non-radial kernels such as* $K(x, z) = \min(x, z) - xz$.
- *We will now use this subroutine to perform distance matrix interpolation.*

Depending on the type of application we are dealing with, we may or may not be able to select where the data is collected, i.e., the location of the data sites or design.

Standard choices in low space dimensions include

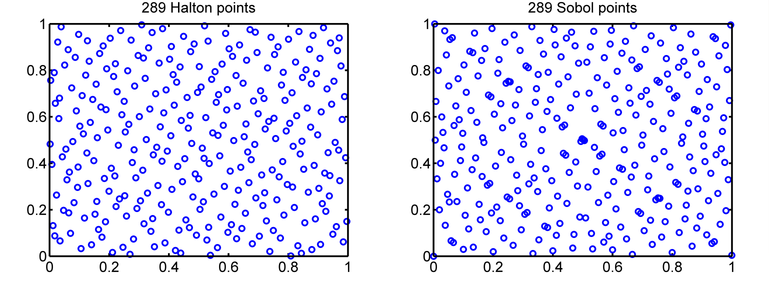
- tensor products of equally spaced points
- tensor products of Chebyshev points



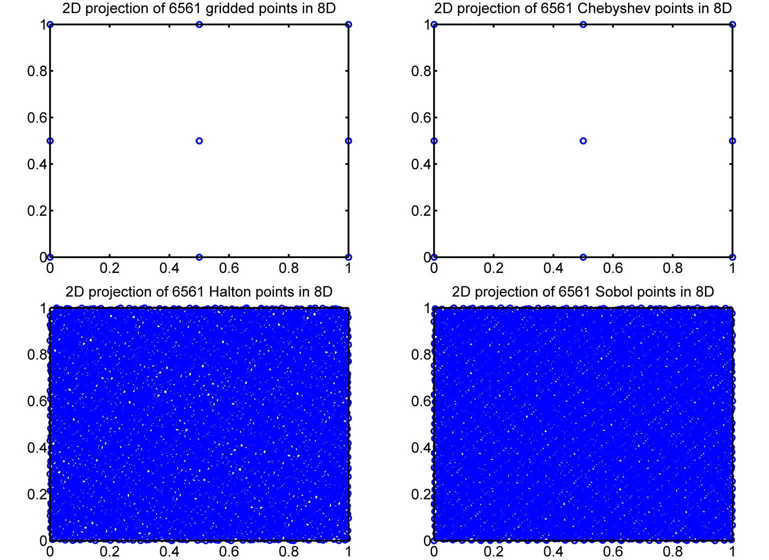


In higher space dimensions it is important to have space-filling (or low-discrepancy) quasi-random point sets. Examples include

- Halton points
- Sobol′ points
- lattice designs
- Latin hypercube designs
- and quite a few others (digital nets, Faure, Niederreiter, etc.)



The difference between the standard (tensor product) designs and the quasi-random designs shows especially in higher space dimensions:



Program (DistanceMatrixFit.m)

```
1   d = 3;
2   k = 2; N = (2^k+1)^d;
3   neval = 10; M = neval^d;
4   dsites = CreatePoints(N,d,'h');
5   ctrs = dsites;
6   epoints = CreatePoints(M,d,'u');
7   rhs = testfunctionsD(d,dsites);
8   IM = DistanceMatrix(dsites,ctrs);
9   EM = DistanceMatrix(epoints,ctrs);
10  s = EM * (IM\rhs);
11  exact = testfunctionsD(d,epoints);
12  maxerr = norm(s-exact,inf)
13  rms_err = norm(s-exact)/sqrt(M)
```

Note the simultaneous evaluation of the interpolant at the entire set of evaluation points on line 10.

Root-mean-square error:

$$\text{RMS-error} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}[s(\xi_i) - f(\xi_i)]^2} = \frac{1}{\sqrt{M}}\|s - f\|_2, \qquad (5)$$

where the ξ_i, $j = 1, \ldots, M$ are the *evaluation points*.

Remark

The basic MATLAB *code for the solution of any kind of RBF interpolation problem will be very similar to* `DistanceMatrixFit`*. Moreover, the data used — even for the distance matrix interpolation considered here — can also be "real" data. Just replace lines 4 and 7 by code that generates the data sites and data values for the right-hand side.*

CreatePoints.m (instead of reading points from files as in the book)

```
function [points, N] = CreatePoints(N,d,gridtype)
% Computes a set of N points in [0,1]^d
% Note: could add variable interval later
% Inputs:
% N: number of interpolation points
% d: space dimension
% gridtype: 'c'=Chebyshev, 'f'=fence(rank-1 lattice),
%    'h'=Halton, 'l'=latin hypercube, 'r'=random uniform,
%    's'=Sobol, 'u'=uniform grid
% Outputs:
% points: an Nxd matrix (each row contains one d-D point)
% N: might be slightly less than original N for
%    Chebyshev and gridded uniform points
% Calls on: chebsamp,lattice,haltonseq,lhsamp,gridsamp
% Also needs: fdnodes,gaussj
% Requires Statistics Toolbox for haltonset and sobolset.
```

The tables and figures below show some examples computed with `DistanceMatrixFit`.

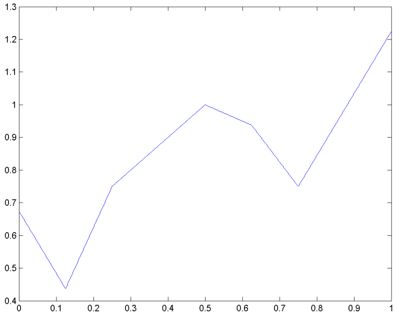
The number M of evaluation points for $d = 1, 2, \ldots, 6$, was 1000, 1600, 1000, 256, 1024, and 4096, respectively (i.e., `neval` = 1000, 40, 10, 4, 4, and 4, respectively).

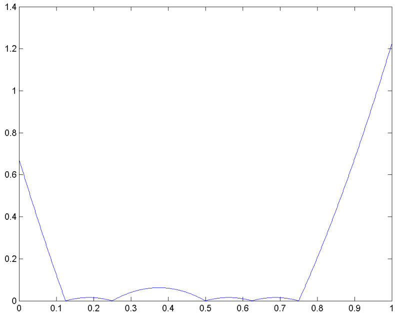
Note that, as the space dimension d increases, more and more of the (uniformly gridded) evaluation points lie on the boundary of the domain, while the data sites (which are given as Halton points) are located in the interior of the domain.

The value k listed in the tables is the same as the `k` in line 2 of `DistanceMatrixFit`.

	1D		2D		3D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	3	5.896957e-001	9	1.937341e-001	27	9.721476e-002
2	5	3.638027e-001	25	6.336315e-002	125	6.277141e-002
3	9	1.158328e-001	81	2.349093e-002	729	2.759452e-002
4	17	3.981270e-002	289	1.045010e-002		
5	33	1.406188e-002	1089	4.326940e-003		
6	65	5.068541e-003	4225	1.797430e-003		
7	129	1.877013e-003				
8	257	7.264159e-004				
9	513	3.016376e-004				
10	1025	1.381896e-004				
11	2049	6.907386e-005				
12	4097	3.453179e-005				

	4D		5D		6D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	81	1.339581e-001	243	9.558350e-002	729	5.097600e-002
2	625	6.817424e-002	3125	3.118905e-002		

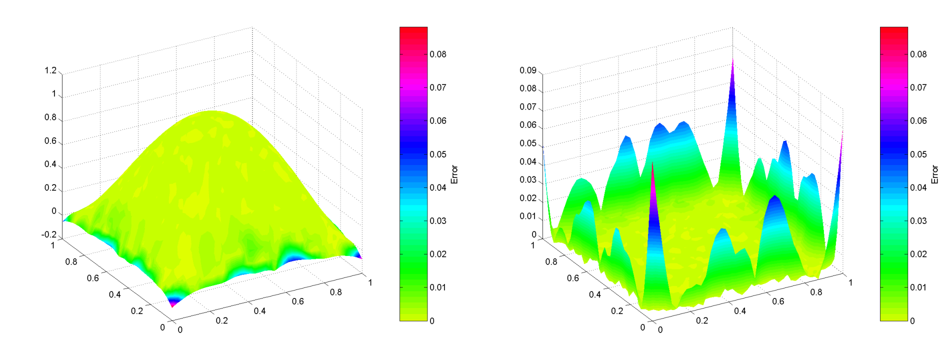




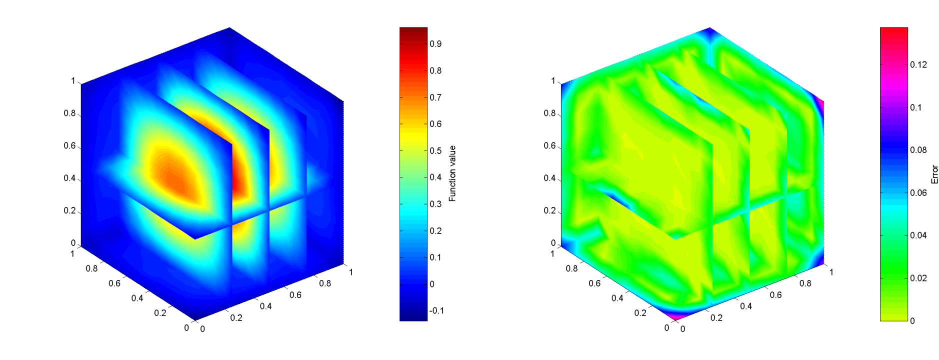
Left: distance matrix fit for $d = 1$ with 5 Halton points for f_1
Right: corresponding error

Remark

Note the piecewise linear nature of the interpolant. If we use more points then the fit becomes more accurate (see table) but then we can't recognize the piecewise linear nature of the interpolant.



Left: distance matrix fit for $d = 2$ with 289 Halton points for f_2
Right: corresponding error
Interpolant is false-colored according to absolute error



Left: distance matrix fit for $d = 3$ with 729 Halton points for f_3 (colors represent function values)
Right: corresponding error

Remark

We can see clearly that most of the error is concentrated near the boundary of the domain.

In fact, the absolute error is about one order of magnitude larger near the boundary than it is in the interior of the domain.

This is no surprise since the data sites are located in the interior.

Even for uniformly spaced data sites (including points on the boundary) the main error in radial basis function interpolation is usually located near the boundary.

Observations

From this first simple example we can observe a number of other features. Most of them are characteristic for radial basis function interpolants.

- The basis functions $B_j = \| \cdot - \boldsymbol{x}_j \|_2$ are radially symmetric.
- As the MATLAB scripts show, the method is extremely simple to implement for any space dimension d.
    - No underlying computational mesh is required to compute the interpolant. The process of mesh generation is a major factor when working in higher space dimensions with polynomial-based methods such as splines or finite elements.
    - All that is required for our method is the pairwise distance between the data sites. Therefore, we have what is known as a meshfree (or *meshless*) method.

Observations (cont.)

- The accuracy of the method improves if we add more data sites.
  - It seems that the RMS-error in the tables above are reduced by a factor of about two from one row to the next.
  - Since we use $(2^k + 1)^d$ uniformly distributed random data points in row k this indicates a convergence rate of roughly $\mathcal{O}(h)$, where h can be viewed as something like the average distance or meshsize of the set $\mathcal{X}$ of data sites.
- The interpolant used here (as well as many other radial basis function interpolants used later) requires the solution of a system of linear equations with a dense $N \times N$ matrix. This makes it very costly to apply the method in its simple form to large data sets.
- Moreover, as we will see later, these matrices also tend to be rather ill-conditioned.

Goals

- present alternatives to this basic interpolation method that address the problems mentioned above such as
  - limitation to small data sets,
  - ill-conditioning,
  - limited accuracy and
  - limited smoothness of the interpolant.

References I

[Bax91] B. J. C. Baxter, *Conditionally positive functions and p-norm distance matrices*, Constructive Approximation **7** (1991), no. 1, 427–440.

[Blu38] L. M. Blumenthal, *Distance Geometries*, Univ. of Missouri Studies **13** (1938), 142.

[Boc41] S. Bochner, *Hilbert distances and positive definite functions*, Ann. of Math. **42** (1941), 647–656.

[BR90] C. de Boor and A. Ron, *On multivariate polynomial interpolation*, Constr. Approx. **6** (1990), 287–302.

[BR92a] ______, *The least solution for the polynomial interpolation problem*, Math. Z. **210** (1992), 347–378.

[BR92b] C. de Boor and Amos Ron, *Computational aspects of polynomial interpolation in several variables*, Mathematics of Computation **58** (1992), no. 198, 705–727.

References II

[Cur59] P. C. Curtis, Jr, *n-parameter families and best approximation*, Pacific J. Math. **9** (1959), no. 4, 1013–1027.

[Fas07] G. E. Fasshauer, *Meshfree Approximation Methods with* MATLAB, Interdisciplinary Mathematical Sciences, vol. 6, World Scientific Publishing Co., Singapore, 2007.

[FM15] G. E. Fasshauer and M. J. McCourt, *Kernel-based Approximation Methods using* MATLAB, World Scientific Publishers, 2015, (in preparation).

[Haa18] A. Haar, *Die Minkowskische Geometrie and die Annäherung an stetige Funktionen*, Math. Ann. **18** (1918), 294–311.

[Hal60] J. H. Halton, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numerische Mathematik **2** (1960), 84–90.

References III

[Mai56] John C. Mairhuber, *On Haar's theorem concerning Chebychev approximation problems having unique solutions*, Proceedings of the American Mathematical Society **7** (1956), no. 4, 609–615, ArticleType: research-article / Full publication date: Aug., 1956 / Copyright 1956 American Mathematical Society.

[Mic86] C. A. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constructive Approximation **2** (1986), no. 1, 11–22.

[NX12] Akil Narayan and Dongbin Xiu, *Stochastic collocation methods on unstructured grids in high dimensions via interpolation*, SIAM Journal on Scientific Computing **34** (2012), no. 3, A1729–A1752.

[Sch38] I. J. Schoenberg, *Metric spaces and completely monotone functions*, The Annals of Mathematics **39** (1938), no. 4, 811–841.

[WLH97] T.-T. Wong, W.-S. Luk, and P.-A. Heng, *Sampling with Hammersley and Halton points*, J. Graphics Tools **2** (1997), 9–24.

References IV

[WW76] J. H. Wells and L. R. Williams, *Embeddings and Extensions in Analysis*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer, Berlin, 1976.