Motivation: Scattered Data Interpolation in $\mathbb{R}^d$

Mathematical Description of Scattered Data Fitting

Example: Interpolation with Distance Matrices

Using Different Designs
Outline

1. Motivation: Scattered Data Interpolation in $\mathbb{R}^d$
2. Mathematical Description of Scattered Data Fitting
3. Example: Interpolation with Distance Matrices
4. Using Different Designs
Motivation: Scattered Data Interpolation in $\mathbb{R}^d$

What are multivariate scattered data?

Univariate Functions

Start from Calculus:

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

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

\[ f(x) = \frac{3}{4} e^{-((9x-2)^2)/4} + \frac{3}{4} e^{-((9x+1)^2)/49} + \frac{1}{2} e^{-((9x-7)^2)/4} - \frac{1}{5} e^{-((9x-4)^2)} \]
Motivation: Scattered Data Interpolation in $\mathbb{R}^d$

What are multivariate scattered data?

Multivariate Functions

From multivariable Calculus:

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

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

$$f(\mathbf{x}) = \frac{3}{4} e^{-(9x-2)^2+(9y-2)^2)/4}$$
$$+ \frac{3}{4} e^{-(9x+1)^2/49+(9y+1)^2/10)}$$
$$+ \frac{1}{2} e^{-(9x-7)^2+(9y-3)^2)/4}$$
$$- \frac{1}{5} e^{-(9x-4)^2+(9y-7)^2)}$$
More Multivariate Functions

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

$$f(\mathbf{x}) = \frac{3}{4} e^{-((9x-2)^2 + (9y-2)^2 + (9z-2)^2)/4}$$
$$+ \frac{3}{4} e^{-((9x+1)^2)/49 - ((9y+1)^2)/10 - ((9z+1)^2)/25}$$
$$+ \frac{1}{2} e^{-((9x-7)^2 + (9y-3)^2 + (9z-5)^2)/4}$$
$$- \frac{1}{5} e^{-((9x-4)^2 + (9y-7)^2 + (9z-5)^2)}$$
More Multivariate Functions

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

$$f(\mathbf{x}) = 64x(1 - x)y(1 - y)z(1 - z)$$
What are multivariate scattered data?

289 uniformly gridded data sites in 2D
289 Chebyshev data sites in 2D
Motivation: Scattered Data Interpolation in $\mathbb{R}^d$

What are multivariate scattered data?

289 Halton data sites in 2D
Motivation: Scattered Data Interpolation in $\mathbb{R}^d$

What are multivariate scattered data?

1368 track data sites in 2D
What are multivariate scattered data?

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
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.
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.
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
    \[ \rightarrow \] (scattered data) interpolation
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
    $\rightarrow$ (scattered data) interpolation
  - We may want the function $s$ to approximately match the given measurements at the corresponding locations
    $\rightarrow$ (scattered data) approximation such as least squares (with and without noise)
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
    \( \rightarrow \) (scattered data) interpolation
  - We may want the function \( s \) to approximately match the given measurements at the corresponding locations
    \( \rightarrow \) (scattered data) approximation such as least squares (with and without noise)
- We will mostly concentrate on the no-noise, interpolation setting.
Outline

1. Motivation: Scattered Data Interpolation in $\mathbb{R}^d$
2. Mathematical Description of Scattered Data Fitting
3. Example: Interpolation with Distance Matrices
4. Using Different Designs
Mathematical description

- Assume we are given
  - measurement locations (data sites):
    \[ \mathcal{X} = \{ \mathbf{x}_i, \; i = 1, \ldots, N \} \subset \Omega \subset \mathbb{R}^d \]
  - 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., \( y_i = f(\mathbf{x}_i), \; i = 1, \ldots, N \).

- Notation\(^1\) for interpolating function: \( s \)

---

\(^1\)Note that this is different from the notation used in [Fas07]. It is in line with [FM15].
Assume we are given

- measurement locations (data sites): $\mathcal{X} = \{x_i, \ i = 1, \ldots, N\} \subset \Omega \subset \mathbb{R}^d$
- 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., $y_i = f(x_i), \ i = 1, \ldots, N$.

Notation\(^1\) for interpolating function: $s$

---

Problem (Scattered Data Interpolation)

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

\(^1\)Note that this is different from the notation used in [Fas07]. It is in line with [FM15].
Scattered Data Interpolation
Scattered Data Interpolation
Scattered Data Interpolation
Scattered Data Interpolation
Scattered Data Interpolation
Scattered Data Interpolation
Standard setup

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

\[
s(\mathbf{x}) = \sum_{j=1}^{N} c_j B_j(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d.
\] (1)
Standard setup

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

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

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

$$Bc = y,$$

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

fasshauer@iit.edu
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.
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 $N - 1$. 
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 $\Omega$ if

$$\det B \neq 0$$

for any set of distinct $x_1, \ldots, x_N$ in $\Omega$. Here $B$ is the square matrix with entries $B_{ij} = B_j(x_i)$, $i, j = 1, \ldots, N$. 
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 $\Omega$ if

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

for any set of distinct $x_1, \ldots, x_N$ in $\Omega$. Here $B$ is the square matrix with entries $B_{ij} = B_j(x_i), i, j = 1, \ldots, N$.

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 $x_1, \ldots, x_N$ from the space $\mathcal{B}$. 

fasshauer@iit.edu

MATH 590 – Chapter 1
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 \( \Omega \) if

\[
\det B \neq 0
\]

for any set of distinct \( x_1, \ldots, x_N \) in \( \Omega \). Here \( B \) is the square matrix with entries \( B_{ij} = B_j(x_i), i, j = 1, \ldots, N \).

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 \( x_1, \ldots, x_N \) 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.
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.
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 $\mathbb{R}^2$. 
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 $N \geq 2$. 

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.
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

$$\det (B_j(x_i)) \neq 0 \quad (2)$$

for any distinct $x_1, \ldots, x_N$. 


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

$$\det (B_j(x_i)) \neq 0$$  \hspace{1cm} (2)

for any distinct $x_1, \ldots, x_N$. 

\begin{center}
\text{Haar–Mairhuber–Curtis CDF}
\end{center}
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

\[
\det(B_j(x_i)) \neq 0
\]

for any distinct \( x_1, \ldots, x_N \).

Since the determinant is a continuous function of \( x_1 \) and \( x_2 \) we must have had \( \det B = 0 \) at some point along the path. This contradicts (2).
Outline

1. Motivation: Scattered Data Interpolation in $\mathbb{R}^d$
2. Mathematical Description of Scattered Data Fitting
3. Example: Interpolation with Distance Matrices
4. Using Different Designs
We want to work with a data dependent approximation space as suggested by the Haar-Mairhuber-Curtis theorem. “Test function”

\[ f_d(x) = 4^d \prod_{\ell=1}^{d} x_\ell (1 - x_\ell), \quad x = (x_1, \ldots, x_d) \in [0, 1]^d \]
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.
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 \( 0 \leq a_i < p \).
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 $h_p : \mathbb{N}_0 \rightarrow [0, 1)$ via

$$h_p(n) = \sum_{i=0}^{k} \frac{a_i}{p^{i+1}}$$
Next, define $h_p : \mathbb{N}_0 \rightarrow [0, 1)$ via

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

Example

$$h_3(10) = \frac{1}{3} + \frac{1}{3^3} = \frac{10}{27}$$
Next, define $h_p : \mathbb{N}_0 \rightarrow [0, 1)$ via

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

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

$$s(x_j) = f_d(x_j), \quad x_j \in H_{d,N}$$
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

$$s(x_j) = f_d(x_j), \quad x_j \in H_{d,N}$$

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”)
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

$$s(x_j) = f_d(x_j), \quad x_j \in H_{d,N}$$

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

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

and $c_j$ determined by interpolation conditions

$$s(x_i) = f_1(x_i), \quad i = 1, \ldots, N$$
Clearly, the basis functions $B_j = | \cdot - x_j |$ are dependent on the data sites $x_j$ as suggested by Haar–Mairhuber–Curtis.
Clearly, the basis functions $B_j = |\cdot - x_j|$ are dependent on the data sites $x_j$ as suggested by Haar–Mairhuber–Curtis.

- $B(x) = |x|$ is called **basic function**
- $K(x, z) = |x - z|$ would be the **kernel**
Clearly, the basis functions $B_j = | \cdot - x_j |$ are dependent on the data sites $x_j$ as suggested by Haar–Mairhuber–Curtis

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

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

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

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

$B(x) = |x|$ 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 $x_j$

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

$B(x) = |x|$ 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 $x_j$

$ightarrow$ 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

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

(3)
Now the coefficients $c_j$ in the scattered data interpolation problem are found by solving the linear system

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

(3)

The matrix in (3) is a distance matrix.
Now the coefficients $c_j$ in the scattered data interpolation problem are found by solving the linear system

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

(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]).
Now the coefficients $c_j$ in the scattered data interpolation problem are found by solving the linear system

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

(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).
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| & \ldots & |x_1 - x_N| \\
|x_2 - x_1| & |x_2 - x_2| & \ldots & |x_2 - x_N| \\
\vdots & \vdots & \ddots & \vdots \\
|x_N - x_1| & |x_N - x_2| & \ldots & |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}

(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:

$$s(x) = \sum_{j=1}^{N} c_j \|x - x_j\|_2,$$

$x \in [0, 1]^d$, (4)

and find the $c_j$ by solving

$$\begin{bmatrix}
\|x_1 - x_1\|_2 & \|x_1 - x_2\|_2 & \cdots & \|x_1 - x_N\|_2 \\
\|x_2 - x_1\|_2 & \|x_2 - x_2\|_2 & \cdots & \|x_2 - x_N\|_2 \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - x_1\|_2 & \|x_N - x_2\|_2 & \cdots & \|x_N - x_N\|_2
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_N
\end{bmatrix}
= \begin{bmatrix}
f_d(x_1) \\
f_d(x_2) \\
\vdots \\
f_d(x_N)
\end{bmatrix}.$$

Note that the basis is again data dependent

For $d > 1$ span $\{\|\cdot - x_j\|_2, j = 1, \ldots, N\}$ is not piecewise linear

Piecewise linear splines in higher space dimensions are usually constructed differently (via a cardinal basis on an underlying computational mesh)
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

$$s(x) = \sum_{j=1}^{N} c_j \|x - x_j\|_2, \quad x \in [0, 1]^d,$$

and find the $c_j$ by solving

$$
\begin{bmatrix}
\|x_1 - x_1\|_2 & \|x_1 - x_2\|_2 & \cdots & \|x_1 - x_N\|_2 \\
\|x_2 - x_1\|_2 & \|x_2 - x_2\|_2 & \cdots & \|x_2 - x_N\|_2 \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - x_1\|_2 & \|x_N - x_2\|_2 & \cdots & \|x_N - x_N\|_2
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_N
\end{bmatrix}
= 
\begin{bmatrix}
f_d(x_1) \\
f_d(x_2) \\
\vdots \\
f_d(x_N)
\end{bmatrix}.
$$
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

$$s(x) = \sum_{j=1}^{N} c_j \|x - x_j\|_2, \quad x \in [0, 1]^d,$$

and find the $c_j$ by solving

$$
\begin{bmatrix}
\|x_1 - x_1\|_2 & \|x_1 - x_2\|_2 & \ldots & \|x_1 - x_N\|_2 \\
\|x_2 - x_1\|_2 & \|x_2 - x_2\|_2 & \ldots & \|x_2 - x_N\|_2 \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - x_1\|_2 & \|x_N - x_2\|_2 & \ldots & \|x_N - x_N\|_2 \\
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_N \\
\end{bmatrix}
= 
\begin{bmatrix}
f_d(x_1) \\
f_d(x_2) \\
\vdots \\
f_d(x_N) \\
\end{bmatrix}.
$$

Note that the basis is again data dependent.
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

\[
s(x) = \sum_{j=1}^{N} c_j \|x - x_j\|_2, \quad x \in [0, 1]^d,
\]

and find the \(c_j\) by solving

\[
\begin{bmatrix}
\|x_1 - x_1\|_2 & \|x_1 - x_2\|_2 & \cdots & \|x_1 - x_N\|_2 \\
\|x_2 - x_1\|_2 & \|x_2 - x_2\|_2 & \cdots & \|x_2 - x_N\|_2 \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - x_1\|_2 & \|x_N - x_2\|_2 & \cdots & \|x_N - x_N\|_2 \\
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_N \\
\end{bmatrix}
= \begin{bmatrix}
f_d(x_1) \\
f_d(x_2) \\
\vdots \\
f_d(x_N) \\
\end{bmatrix}.
\]

- Note that the basis is again data dependent
- For \(d > 1\) \(\text{span}\{\|\cdot - x_j\|_2, \ j = 1, \ldots, N\}\) is not piecewise linear
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

$$s(x) = \sum_{j=1}^{N} c_j \|x - x_j\|_2, \quad x \in [0, 1]^d,$$

and find the $c_j$ by solving

$$
\begin{bmatrix}
\|x_1 - x_1\|_2 & \|x_1 - x_2\|_2 & \cdots & \|x_1 - x_N\|_2 \\
\|x_2 - x_1\|_2 & \|x_2 - x_2\|_2 & \cdots & \|x_2 - x_N\|_2 \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - x_1\|_2 & \|x_N - x_2\|_2 & \cdots & \|x_N - x_N\|_2 \\
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_N \\
\end{bmatrix}
= 
\begin{bmatrix}
f_d(x_1) \\
f_d(x_2) \\
\vdots \\
f_d(x_N) \\
\end{bmatrix}.
$$

- Note that the basis is again data dependent
- For $d > 1$ span$\{\|\cdot - x_j\|_2, j = 1, \ldots, N\}$ is not piecewise linear
- Piecewise linear splines in higher space dimensions are usually constructed differently (via a cardinal basis on an underlying computational mesh)
A typical basis function for the Euclidean distance matrix fit, $B_j(x) = \|x - x_j\|_2$ with $x_j = 0$ 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).

```matlab
function DM = DistanceMatrixBookBook(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);
```

One of our main MATLAB subroutines

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

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

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

```
<table>
<thead>
<tr>
<th>dr</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>cc</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>dr</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>cc</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
```
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).

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

Works for any space dimension!
Alternate forms of DistanceMatrix.m

Program (DistanceMatrixRepmat.m)

1 function DM = DistanceMatrixRepmat(dsites,ctrs)
2 [M,d] = size(dsites); [N,d] = size(ctrs);
3 DM = zeros(M,N);
4 for l=1:d
5a DM = DM + (repmat(dsites(:,l),1,N) - ...
5b repmat(ctrs(:,l)',M,1)).^2;
6 end
7 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`)

```matlab
1 function DM = DistanceMatrixA(dsites,ctrs)
2 M = size(dsites,1); N = size(ctrs,1);
3 T1 = sum(dsites.*dsites,2);
3 T2 = -2*dsites*ctrs';
5 T3 = (sum(ctrs.*ctrs,2))';
6 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)*

```matlab
1 function DM = DistanceMatrix(dsites,ctrs)
2     M = size(dsites,1); N = size(ctrs,1);
3     DM = repmat(sum(dsites.*dsites,2),1,N) - ...
4         2*dsites*ctrs' + ...
5     repmat((sum(ctrs.*ctrs,2))',M,1);
6     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.**
Outline

1. Motivation: Scattered Data Interpolation in $\mathbb{R}^d$
2. Mathematical Description of Scattered Data Fitting
3. Example: Interpolation with Distance Matrices
4. Using Different Designs
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.)
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:
The difference between the standard (tensor product) designs and the quasi-random designs shows especially in higher space dimensions:
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:

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

where the \( \xi_i, j = 1, \ldots, M \) are the \textit{evaluation points}. 
Root-mean-square error:

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

where the \( \xi_i, j = 1, \ldots, M \) 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.
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 \texttt{DistanceMatrixFit}.

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

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 \texttt{DistanceMatrixFit}.
Using Different Designs

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$N$</th>
<th>RMS-error</th>
<th>$N$</th>
<th>RMS-error</th>
<th>$N$</th>
<th>RMS-error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>81</td>
<td>1.339581e-001</td>
<td>243</td>
<td>9.558350e-002</td>
<td>729</td>
<td>5.097600e-002</td>
</tr>
<tr>
<td>2</td>
<td>625</td>
<td>6.817424e-002</td>
<td>3125</td>
<td>3.118905e-002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $B_j = \| \cdot - x_j \|_2$ 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 $O(h)$, 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


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


References II


References III


References IV