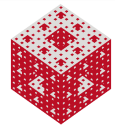
MATH 590: Meshfree Methods

Chapter 1 — Part 3: Radial Basis Function Interpolation in MATLAB

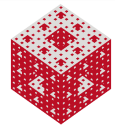
Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

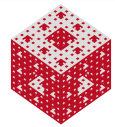
Fall 2014



Outline



Outline

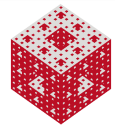


Next goals

- Get a feel for more general kernels (in particular RBFs) before we study them in detail

- Want to overcome limitations of distance matrix interpolation, but keep overall structure:

  combine distance matrix with “good” basic functions



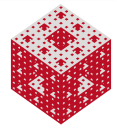
Example

The Gaussian

$$\kappa(r) = e^{-(\varepsilon r)^2}, \qquad r \in \mathbb{R},$$

has a shape parameter ε related to the variance σ^2 of the normal distribution:

$$\varepsilon^2 = 1/(2\sigma^2).$$



Example

The Gaussian

$$\kappa(r) = e^{-(\varepsilon r)^2}, \qquad r \in \mathbb{R},$$

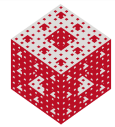
has a shape parameter ε related to the variance σ^2 of the normal distribution:

$$\varepsilon^2 = 1/(2\sigma^2).$$

Compose the Gaussian with Euclidean distance function $\|\cdot\|_2$ and get

$$K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\varepsilon^2 \|\boldsymbol{x}-\boldsymbol{z}\|_2^2}, \qquad \boldsymbol{x} \in \mathbb{R}^d,$$

where $\boldsymbol{z} \in \mathbb{R}^d$ is some fixed center or knot.



Example

The Gaussian

$$\kappa(r) = \mathrm{e}^{-(\varepsilon r)^2}, \qquad r \in \mathbb{R},$$

has a shape parameter ε related to the variance σ^2 of the normal distribution:

$$\varepsilon^2 = 1/(2\sigma^2).$$

Compose the Gaussian with Euclidean distance function $\|\cdot\|_2$ and get

$$K(\boldsymbol{x}, \boldsymbol{z}) = \mathrm{e}^{-\varepsilon^2 \|\boldsymbol{x}-\boldsymbol{z}\|_2^2}, \qquad \boldsymbol{x} \in \mathbb{R}^d,$$

where $\boldsymbol{z} \in \mathbb{R}^d$ is some fixed center or knot.

Remark

Note that κ is univariate, but K is a multivariate kernel function such that

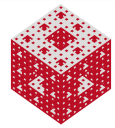
$$B_j(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_j) = \kappa(\|\boldsymbol{x} - \boldsymbol{x}_j\|_2) \qquad \textit{radial basis function (RBF).}$$

Definition

A multivariate function $\widetilde{K} : \mathbb{R}^d \to \mathbb{R}$ is called radial provided there exists a *univariate* function $\kappa : [0, \infty) \to \mathbb{R}$ such that

$$\widetilde{K}(\boldsymbol{x}) = \kappa(r), \quad \text{where} \quad r = \|\boldsymbol{x}\|,$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^d$ — usually the Euclidean norm.



Definition

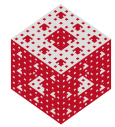
A multivariate function $\widetilde{K} : \mathbb{R}^d \to \mathbb{R}$ is called radial provided there exists a *univariate* function $\kappa : [0, \infty) \to \mathbb{R}$ such that

$$\widetilde{K}(\boldsymbol{x}) = \kappa(r), \quad \text{where} \quad r = \|\boldsymbol{x}\|,$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^d$ — usually the Euclidean norm.

Thus, for a radial function

$$\|\boldsymbol{x}_1\| = \|\boldsymbol{x}_2\| \quad \Longrightarrow \quad \widetilde{K}(\boldsymbol{x}_1) = \widetilde{K}(\boldsymbol{x}_2), \qquad \boldsymbol{x}_1,\ \boldsymbol{x}_2 \in \mathbb{R}^d.$$



Definition

A multivariate function $\widetilde{K} : \mathbb{R}^d \to \mathbb{R}$ is called radial provided there exists a *univariate* function $\kappa : [0, \infty) \to \mathbb{R}$ such that

$$\widetilde{K}(\boldsymbol{x}) = \kappa(r), \quad \text{where} \quad r = \|\boldsymbol{x}\|,$$

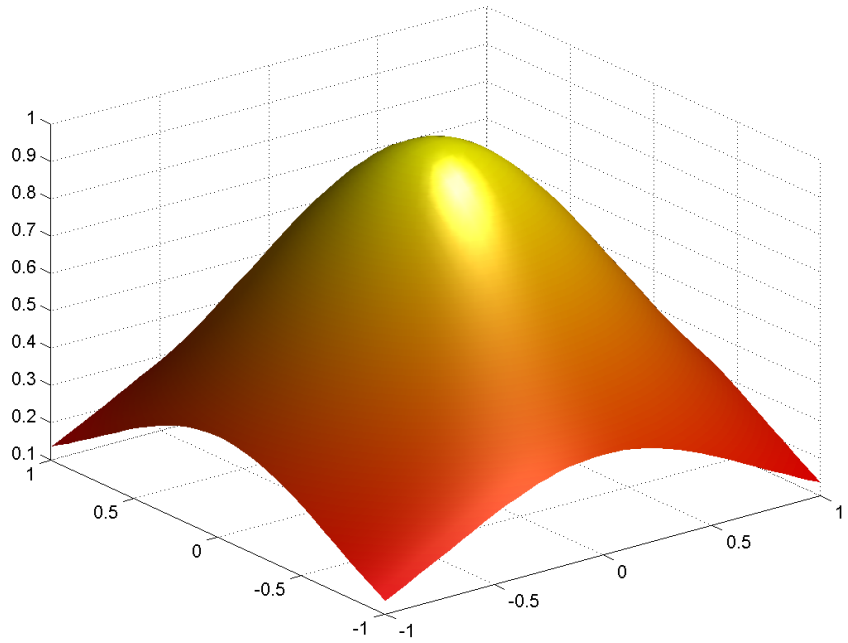
and $\|\cdot\|$ is some norm on $\mathbb{R}^d$ — usually the Euclidean norm.

Thus, for a radial function

$$\|\boldsymbol{x}_1\| = \|\boldsymbol{x}_2\| \quad \Longrightarrow \quad \widetilde{K}(\boldsymbol{x}_1) = \widetilde{K}(\boldsymbol{x}_2), \qquad \boldsymbol{x}_1,\ \boldsymbol{x}_2 \in \mathbb{R}^d.$$

Example

The Euclidean distance function $\widetilde{K}(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$ (or $\kappa(r) = r$) is a special case of a radial (basic) function.



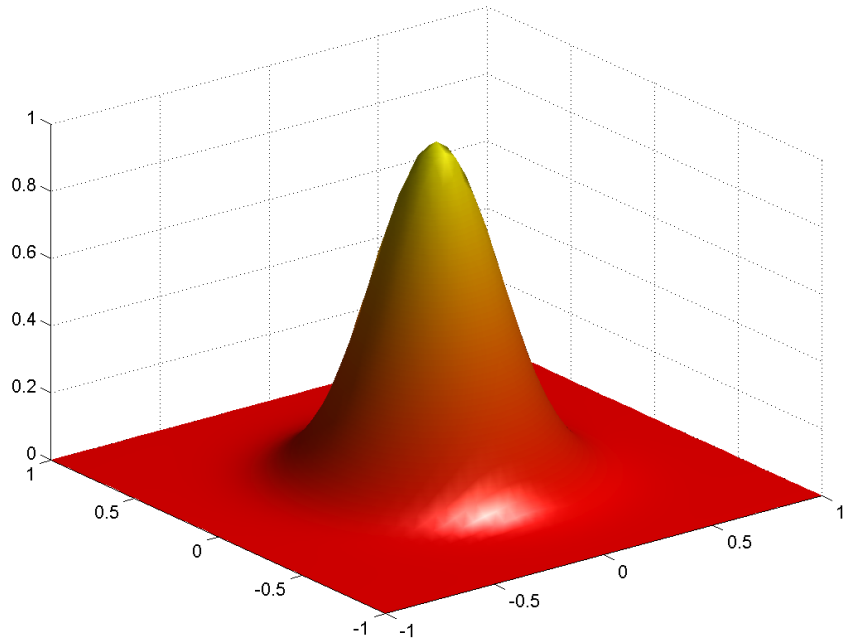
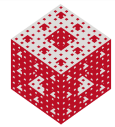


Figure: Gaussian with $\varepsilon = 1$ (left) and $\varepsilon = 3$ (right) centered at the origin.

- A smaller value of ε (i.e., larger variance) causes the function to become "flatter", so it's like an inverse length scale.
- Increasing ε leads to a more peaked RBF.



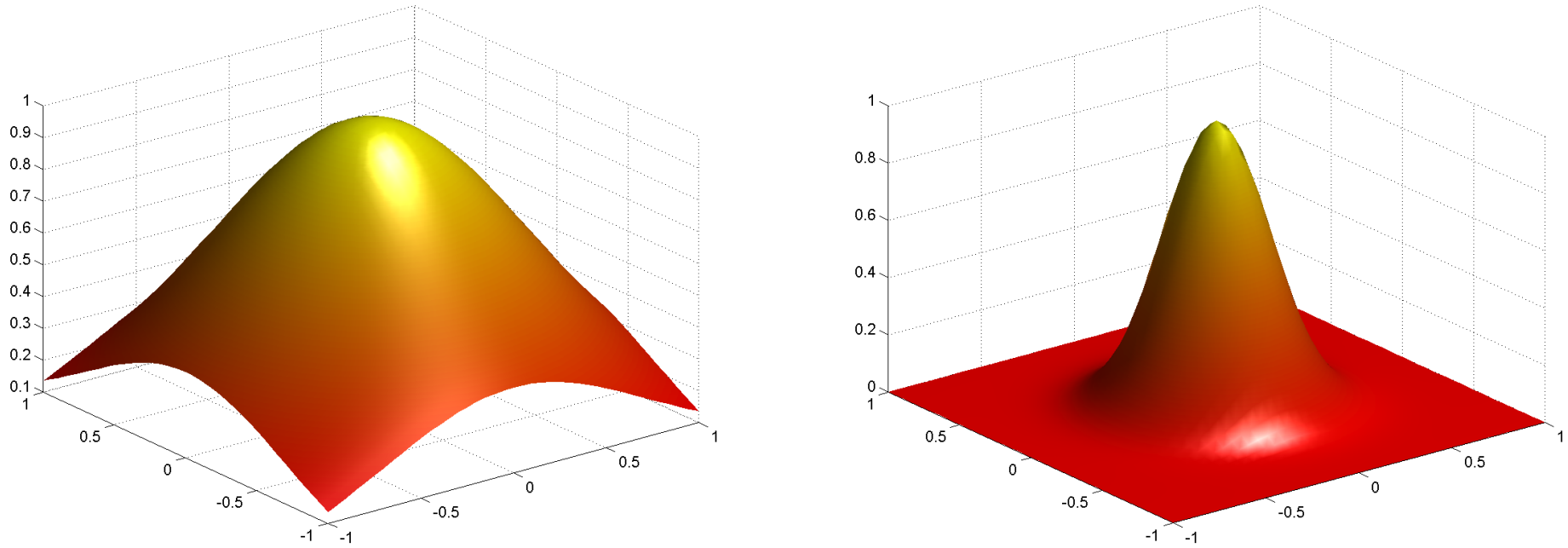
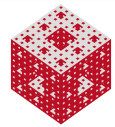


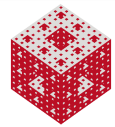
Figure: Gaussian with $\varepsilon = 1$ (left) and $\varepsilon = 3$ (right) centered at the origin.

- A smaller value of ε (i.e., larger variance) causes the function to become "flatter", so it's like an inverse length scale.
- Increasing ε leads to a more peaked RBF.
- The choice of ε influences both accuracy and numerical stability of the solution to our interpolation problem.



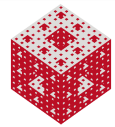
▸ Flat Gaussian CDF

Outline



Instead of distance matrices we now use a radial basis function expansion to solve the scattered data interpolation problem by assuming

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \kappa\left(\|\boldsymbol{x} - \boldsymbol{x}_j\|_2\right), \qquad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

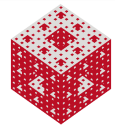


Instead of distance matrices we now use a radial basis function expansion to solve the scattered data interpolation problem by assuming

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \kappa\left(\|\boldsymbol{x} - \boldsymbol{x}_j\|_2\right), \qquad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

Using the interpolation conditions $s(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, we get c_j from

$$\begin{bmatrix} \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_N\|_2\right) \\ \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_N\|_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_N\|_2\right) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}.$$



Instead of distance matrices we now use a radial basis function expansion to solve the scattered data interpolation problem by assuming

$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \kappa\left(\|\boldsymbol{x} - \boldsymbol{x}_j\|_2\right), \qquad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

Using the interpolation conditions $s(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, we get c_j from

$$\begin{bmatrix} \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_N\|_2\right) \\ \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_2 - \boldsymbol{x}_N\|_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2\right) & \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_2\|_2\right) & \ldots & \kappa\left(\|\boldsymbol{x}_N - \boldsymbol{x}_N\|_2\right) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}.$$

Question:

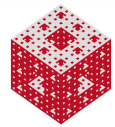
For what type of basic functions κ is the system matrix non-singular?

Setup for Example

- Use $d = 2$
- Basic function: Gaussians or linear function $\kappa(r) = r$
- Code in `RBFInterpolation2D.m` written for 2D (can easily be generalized to dD), and uses `DistanceMatrix.m`.
- Use Franke's function

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

- Use Halton data sites (get others easily from `CreatePoints`)
- Compute errors on 40×40 uniform grid



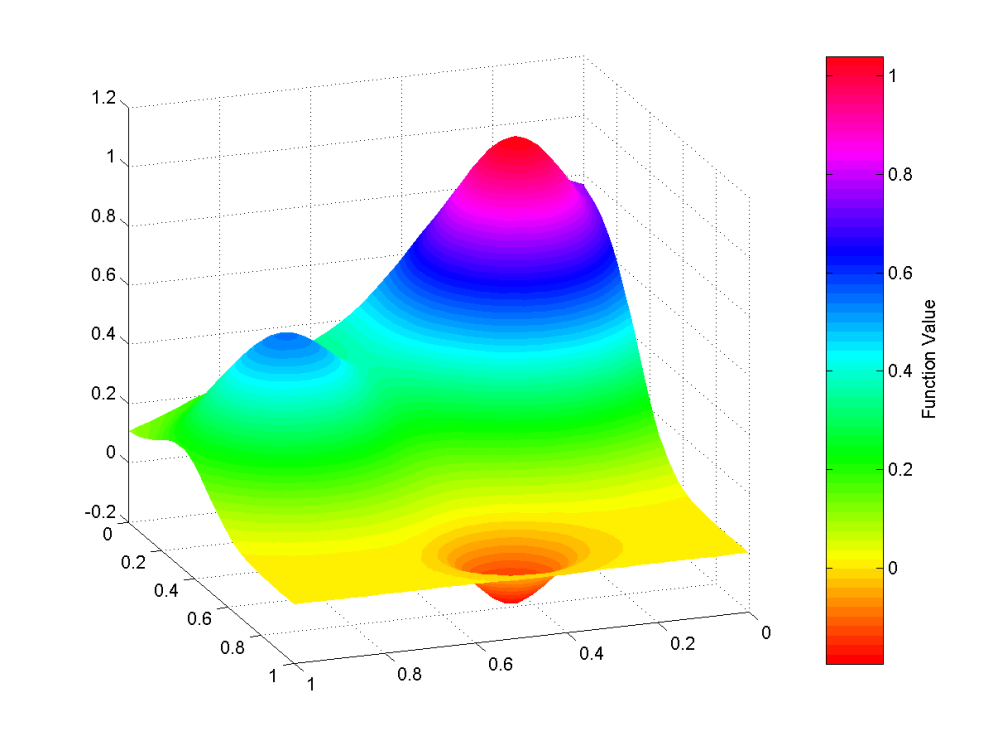
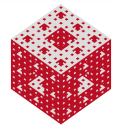


Figure: Franke's test function.



Program (RBFInterpolation2D.m)

```
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 21.1;
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 1089; gridtype = 'h';
 8  dsites = CreatePoints(N,2,gridtype);
 9  ctrs = dsites;
10  neval = 40; M = neval^2;
11  epoints = CreatePoints(M,2,'u');
12  rhs = testfunction(dsites(:,1),dsites(:,2));
13  DM_data = DistanceMatrix(dsites,ctrs);
14  IM = rbf(ep,DM_data);
15  DM_eval = DistanceMatrix(epoints,ctrs);
16  EM = rbf(ep,DM_eval);
17  s = EM * (IM\rhs);
18  exact = testfunction(epoints(:,1),epoints(:,2));
19  maxerr = norm(s-exact,inf)
20  rms_err = norm(s-exact)/neval
```

Stationary and Non-Stationary Approximation

Remark

Note the conflicting use of the term stationary:

here vs. statistics = translation-invariant

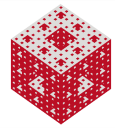
Stationary and Non-Stationary Approximation

Remark

Note the conflicting use of the term stationary:

here vs. statistics = translation-invariant

We usually verify convergence of a method numerically by looking at a sequence of experiments with increasingly larger sets of data.



Stationary and Non-Stationary Approximation

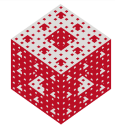
Remark

Note the conflicting use of the term stationary:

here vs. statistics = translation-invariant

We usually verify convergence of a method numerically by looking at a sequence of experiments with increasingly larger sets of data.

Non-stationary approximation: Use one fixed value of ε for all of the experiments.



Stationary and Non-Stationary Approximation

Remark

Note the conflicting use of the term stationary:

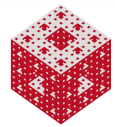
here vs. statistics = translation-invariant

We usually verify convergence of a method numerically by looking at a sequence of experiments with increasingly larger sets of data.

Non-stationary approximation: Use one fixed value of ε for all of the experiments.

Stationary approximation: Scale the shape parameter ε according to the fill distance (or meshsize) h so that we end up using "peaked" basis functions for densely spaced data and "flat" basis functions for coarsely spaced data.

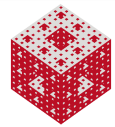
More details later



Fill distance

The fill distance (or covering radius) is usually defined as

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2. \tag{2}$$

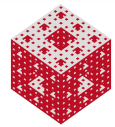


Fill distance

The fill distance (or covering radius) is usually defined as

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2. \tag{2}$$

It indicates how well the data in the set $\mathcal{X}$ fill out the domain Ω.



Fill distance

The fill distance (or covering radius) is usually defined as

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2. \tag{2}$$

It indicates how well the data in the set $\mathcal{X}$ fill out the domain Ω.

In MATLAB:

```
hX = max(min(DM_eval'))
```
(3)

Fill distance

The fill distance (or covering radius) is usually defined as

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2. \tag{2}$$

It indicates how well the data in the set $\mathcal{X}$ fill out the domain Ω.

In MATLAB:

```
hX = max(min(DM_eval'))
```
(3)

Remark

Since `min` *works along columns of a matrix, transposition of the non-symmetric evaluation matrix corresponds to finding — for each evaluation point — the distance to the corresponding closest data site, and then setting* $h_{\mathcal{X},\Omega}$ *as the worst of those distances.*

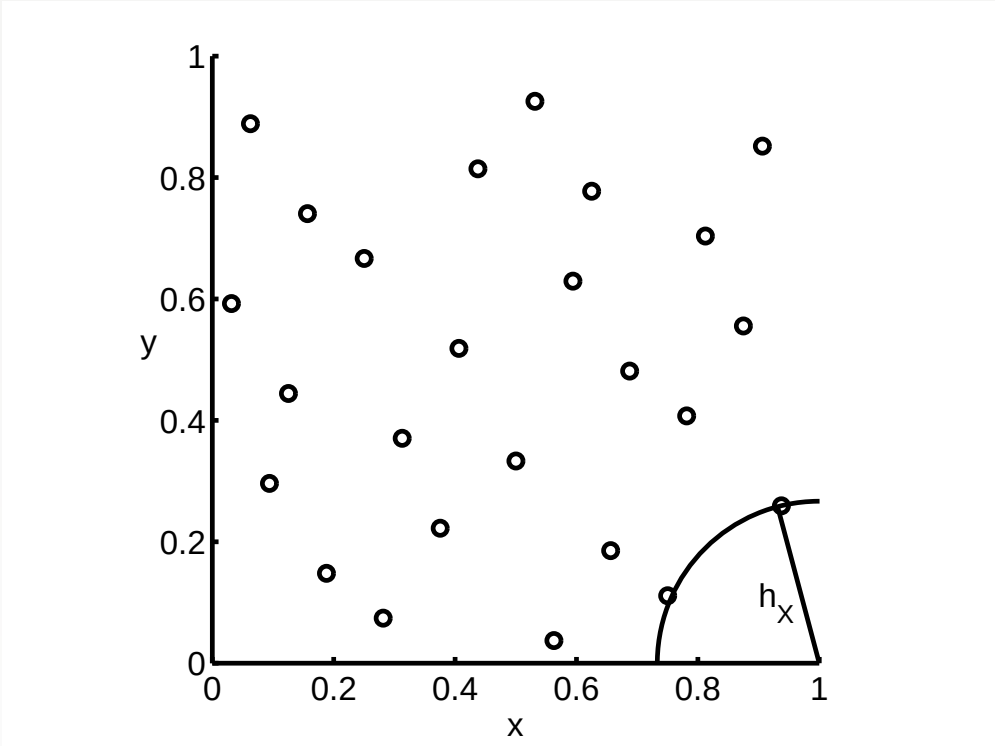
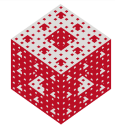


Figure: The fill distance for $N = 25$ Halton points ($h_{\mathcal{X},\Omega} \approx 0.2667$).

It's the radius of the largest possible empty ball that can be placed among the data locations inside Ω.



Test of non-stationary interpolation

- Fix large ε to prevent severe ill-conditioning with large N (if $\varepsilon = 1$, then $N = 25$ already very ill-conditioned)
- Consequence: very localized basis functions that don't capture enough information on small point sets (see left plot)

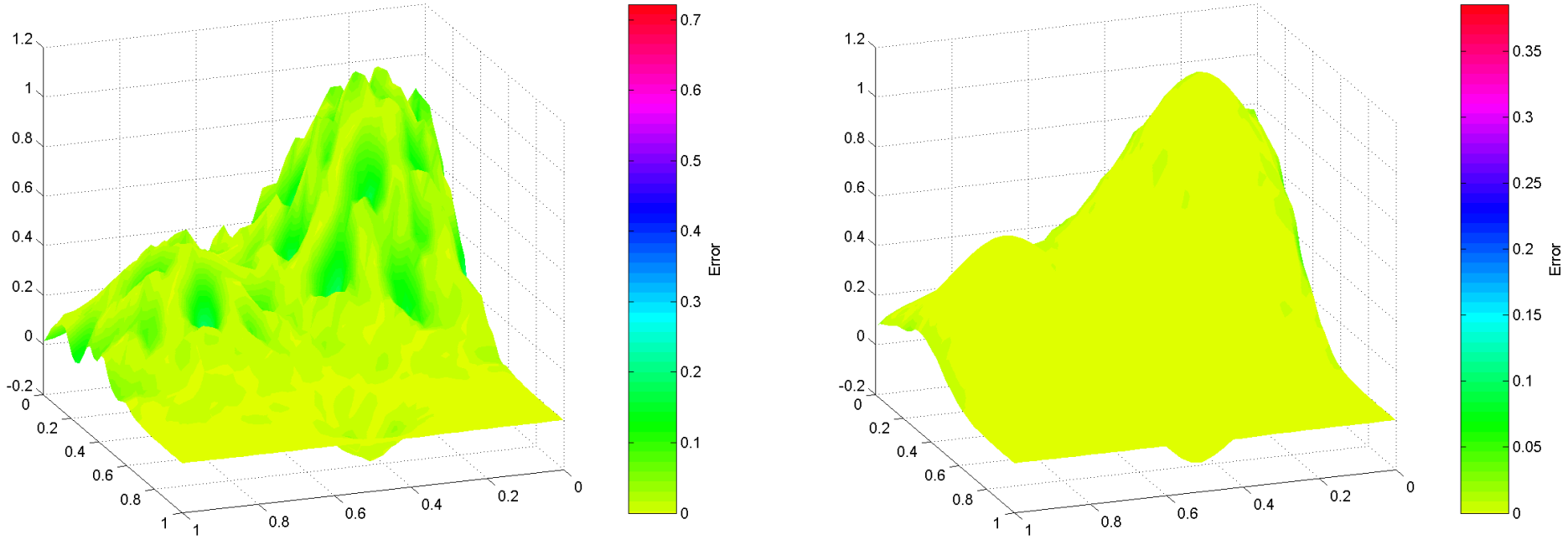
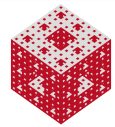


Figure: Gaussian RBF interpolant with $\varepsilon = 21.1$ at $N = 289$ (left) and at $N = 1089$ Halton points (right).

		Gaussian		Distance matrix	
k	N	RMS-error	max-error	RMS-error	max-error
1	9	3.647169e-001	1.039682e+000	1.323106e-001	4.578028e-001
2	25	3.203404e-001	9.670980e-001	6.400558e-002	2.767871e-001
3	81	2.152222e-001	8.455161e-001	1.343780e-002	6.733130e-002
4	289	7.431729e-002	7.219253e-001	3.707360e-003	3.057540e-002
5	1089	1.398297e-002	3.857234e-001	1.143589e-003	1.451950e-002
6	4225	4.890709e-004	1.940675e-002	4.002749e-004	8.022336e-003

Table: Non-stationary RBF interpolation to Franke's function using Gaussians ($\varepsilon = 21.1$) and Euclidean distance matrices.



		Gaussian		Distance matrix	
k	N	RMS-error	max-error	RMS-error	max-error
1	9	3.647169e-001	1.039682e+000	1.323106e-001	4.578028e-001
2	25	3.203404e-001	9.670980e-001	6.400558e-002	2.767871e-001
3	81	2.152222e-001	8.455161e-001	1.343780e-002	6.733130e-002
4	289	7.431729e-002	7.219253e-001	3.707360e-003	3.057540e-002
5	1089	1.398297e-002	3.857234e-001	1.143589e-003	1.451950e-002
6	4225	4.890709e-004	1.940675e-002	4.002749e-004	8.022336e-003

Table: Non-stationary RBF interpolation to Franke's function using Gaussians ($\varepsilon = 21.1$) and Euclidean distance matrices.

Remark

- *Surprising observation in above example: distance matrix fit more accurate than Gaussian.*

		Gaussian		Distance matrix	
k	N	RMS-error	max-error	RMS-error	max-error
1	9	3.647169e-001	1.039682e+000	1.323106e-001	4.578028e-001
2	25	3.203404e-001	9.670980e-001	6.400558e-002	2.767871e-001
3	81	2.152222e-001	8.455161e-001	1.343780e-002	6.733130e-002
4	289	7.431729e-002	7.219253e-001	3.707360e-003	3.057540e-002
5	1089	1.398297e-002	3.857234e-001	1.143589e-003	1.451950e-002
6	4225	4.890709e-004	1.940675e-002	4.002749e-004	8.022336e-003

Table: Non-stationary RBF interpolation to Franke's function using Gaussians ($\varepsilon = 21.1$) and Euclidean distance matrices.

Remark

- *Surprising observation in above example: distance matrix fit more accurate than Gaussian.*
- *Would expect Gaussian to be more accurate $\rightarrow$ find a better ε*

Same test function and Gaussians as before.

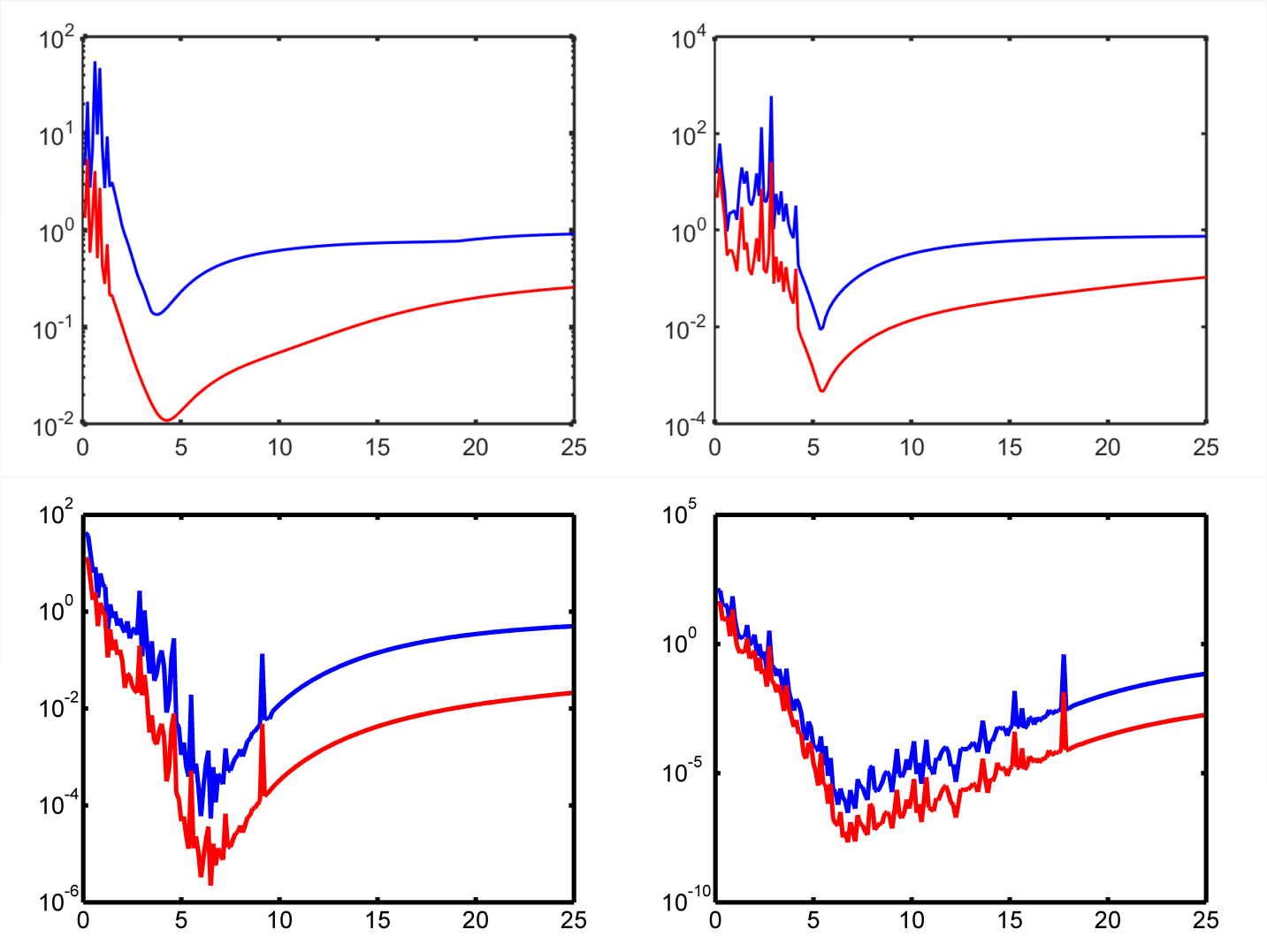
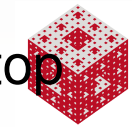
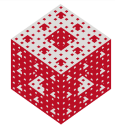


Figure: Maximum (blue) and RMS (red) errors vs. ε for 81 (top left), 289 (top right), 1089 (bottom left), and 4225 Halton points (bottom right).



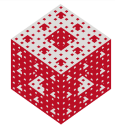
What do we learn from the ε error curves?

- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).



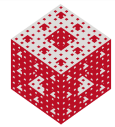
What do we learn from the ε error curves?

- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).
- Error curves are not monotonic.



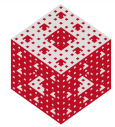
What do we learn from the ε error curves?

- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).
- Error curves are not monotonic.
- Optimal value of ε seems to exist which minimizes errors.



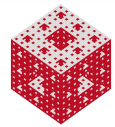
What do we learn from the ε error curves?

- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).
- Error curves are not monotonic.
- Optimal value of ε seems to exist which minimizes errors.
- For small enough values of ε the computational results become unpredictable, and the error curves become erratic (ill-conditioning)



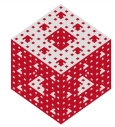
What do we learn from the ε error curves?

- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).
- Error curves are not monotonic.
- Optimal value of ε seems to exist which minimizes errors.
- For small enough values of ε the computational results become unpredictable, and the error curves become erratic (ill-conditioning)
- We'll consider ε to be "safe" if we don't get a MATLAB warning about ill-conditioning.
  - Note that for small N optimal ε is "safe", but for larger N not.
  - We obtain highly accurate solutions from severely ill-conditioned linear systems!
  - This is known as the uncertainty or trade-off principle.



What do we learn from the ε error curves?

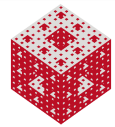
- Errors decrease with decreasing ε (not the same as convergence for $h \to 0$).
- Error curves are not monotonic.
- Optimal value of ε seems to exist which minimizes errors.
- For small enough values of ε the computational results become unpredictable, and the error curves become erratic (ill-conditioning)
- We'll consider ε to be "safe" if we don't get a MATLAB warning about ill-conditioning.
  - Note that for small N optimal ε is "safe", but for larger N not.
  - We obtain highly accurate solutions from severely ill-conditioned linear systems!
  - This is known as the uncertainty or trade-off principle.
- Interesting problem: how to compute optimal solution in a stable way. Later we will see how this can be done with the Hilbert–Schmidt SVD.



Best possible errors for Gaussian interpolation

		smallest "safe" ε			smallest RMS-error		
k	N	ε	RMS-error	max-error	ε	RMS-error	max-error
1	9	0.02	3.658421e-001	1.580259e+000	2.23	1.118026e-001	3.450275e-001
2	25	0.32	3.629342e-001	2.845554e+000	3.64	4.032550e-002	2.996488e-001
3	81	1.64	1.743059e-001	2.398284e+000	4.28	1.090601e-002	1.579465e-001
4	289	4.73	2.785388e-003	5.472502e-002	5.46	4.610079e-004	7.978283e-003
5	1089	10.5	4.945428e-004	1.812246e-002	6.2	2.498848e-006	8.779119e-005
6	4225	21.1	4.890709e-004	1.940675e-002	6.3	4.269292e-008	8.889552e-007

Table: "Optimal" RBF interpolation to Franke's function using Gaussians.



Best possible errors for Gaussian interpolation

		smallest "safe" ε			smallest RMS-error		
k	N	ε	RMS-error	max-error	ε	RMS-error	max-error
1	9	0.02	3.658421e-001	1.580259e+000	2.23	1.118026e-001	3.450275e-001
2	25	0.32	3.629342e-001	2.845554e+000	3.64	4.032550e-002	2.996488e-001
3	81	1.64	1.743059e-001	2.398284e+000	4.28	1.090601e-002	1.579465e-001
4	289	4.73	2.785388e-003	5.472502e-002	5.46	4.610079e-004	7.978283e-003
5	1089	10.5	4.945428e-004	1.812246e-002	6.2	2.498848e-006	8.779119e-005
6	4225	21.1	4.890709e-004	1.940675e-002	6.3	4.269292e-008	8.889552e-007

Table: "Optimal" RBF interpolation to Franke's function using Gaussians.

Remark

- *Errors for "safe" ε now comparable (or smaller) than those for distance matrix interpolation.*

Best possible errors for Gaussian interpolation

		smallest "safe" ε			smallest RMS-error		
k	N	ε	RMS-error	max-error	ε	RMS-error	max-error
1	9	0.02	3.658421e-001	1.580259e+000	2.23	1.118026e-001	3.450275e-001
2	25	0.32	3.629342e-001	2.845554e+000	3.64	4.032550e-002	2.996488e-001
3	81	1.64	1.743059e-001	2.398284e+000	4.28	1.090601e-002	1.579465e-001
4	289	4.73	2.785388e-003	5.472502e-002	5.46	4.610079e-004	7.978283e-003
5	1089	10.5	4.945428e-004	1.812246e-002	6.2	2.498848e-006	8.779119e-005
6	4225	21.1	4.890709e-004	1.940675e-002	6.3	4.269292e-008	8.889552e-007

Table: "Optimal" RBF interpolation to Franke's function using Gaussians.

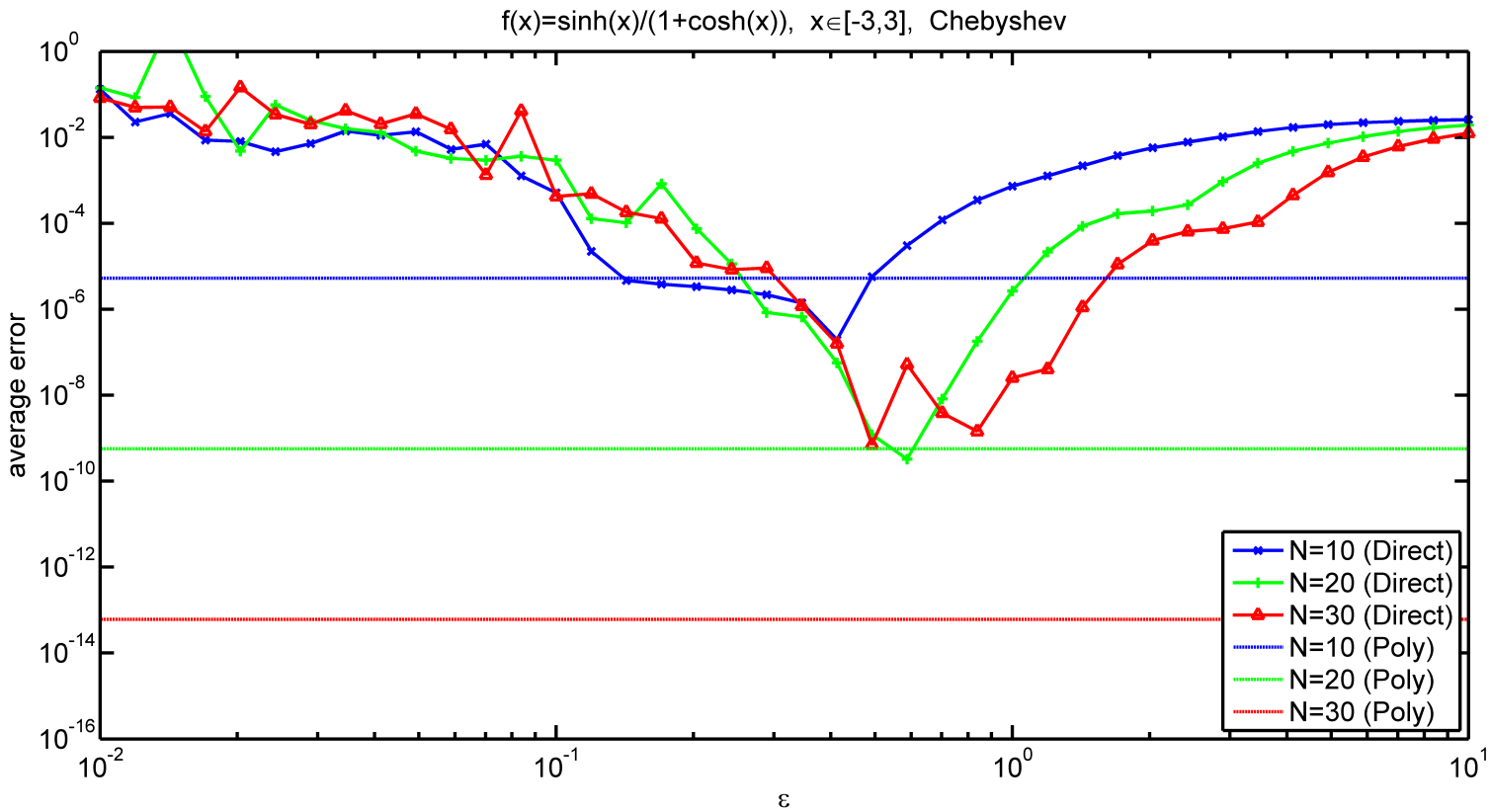
Remark

- *Errors for "safe" ε now comparable (or smaller) than those for distance matrix interpolation.*
- *Much better for "optimal" ε. However, $\varepsilon = 6.2$ with $N = 1089$ Halton points yields* `RCOND` $= 2.683527\mathrm{e}{-020}$.

Avoiding the Uncertainty Principle With the Hilbert–Schmidt SVD

"One can't have high accuracy and stability at the same time."

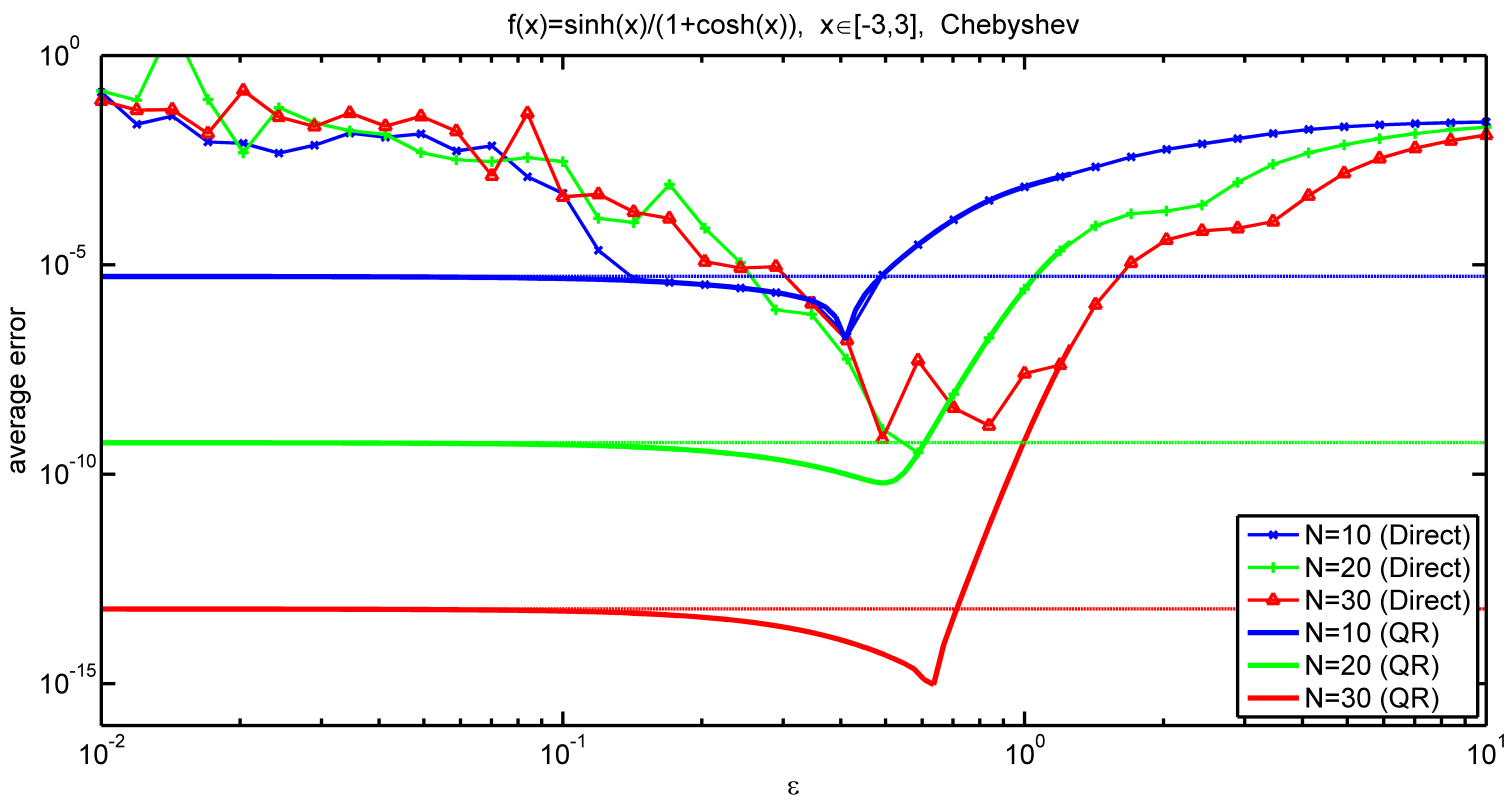
[Sch95a, Sch95b]



Stable evaluation [FM12] inspired by [FP08, FLF11]

Avoiding the Uncertainty Principle With the Hilbert–Schmidt SVD

"Using the standard basis, one can't have high accuracy and stability at the same time." [Sch95a, Sch95b]

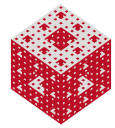


Stable evaluation [FM12] inspired by [FP08, FLF11]

Summary

Remark

If the data are not sampled from a known test function, then we will not be able to choose an "optimal" shape parameter by monitoring the RMS error.

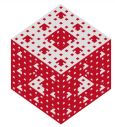


Summary

Remark

If the data are not sampled from a known test function, then we will not be able to choose an "optimal" shape parameter by monitoring the RMS error.

New challenge: how to find "optimal" ε based only on the data?



Summary

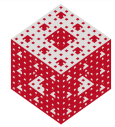
Remark

If the data are not sampled from a known test function, then we will not be able to choose an "optimal" shape parameter by monitoring the RMS error.
New challenge: how to find "optimal" ε based only on the data?

We will study

- ill-conditioning and preconditioning,
- optimal shape parameter selection, and
- alternate stable evaluation methods via Hilbert–Schmidt SVD and eigenfunction expansions

later.



References I

[FLF11] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer, *Stable computations with Gaussian radial basis functions*, SIAM Journal on Scientific Computing **33** (2011), no. 2, 869–892.

[FM12] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM J. Sci. Comput. **34** (2012), no. 2, A737—A762.

[FP08] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput. **30** (2008), no. 1, 60–80.

[Sch95a] R. Schaback, *Error estimates and condition numbers for radial basis function interpolation*, Advances in Computational Mathematics **3** (1995), no. 3, 251–264.

[Sch95b] _____, *Multivariate interpolation and approximation by translates of a basis function*, Approximation Theory VIII, Vol. 1: Approximation and Interpolation (C. K. Chui and L. L. Schumaker, eds.), World Scientific Publishing (Singapore), 1995, pp. 491–514.