Meshfree Approximation with MATLAB

Lecture II: RBF Interpolation and MLS Approximation

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Dolomites Research Week on Approximation
September 8–11, 2008

Outline

- Overview of MLS approximation

- Derive matrix-free meshfree approximation method for scattered data approximation based on MLS and approximate approximation $\longrightarrow$ approximate MLS

- Link (A)MLS and RBF methods

Multivariate Kernel Interpolation

Use data-dependent linear function space

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^s$$

Here $\Phi : \mathbb{R}^s \times \mathbb{R}^s \to \mathbb{R}$ is strictly positive definite (reproducing) kernel

Multivariate Kernel Interpolation

Use data-dependent linear function space

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^s$$

Here $\Phi : \mathbb{R}^s \times \mathbb{R}^s \to \mathbb{R}$ is strictly positive definite (reproducing) kernel

To find c_j solve interpolation equations

$$\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i), \quad i = 1, \ldots, N$$

Leads to linear system with matrix

$$A_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i, j = 1, \ldots, N$$

Matrix-free Methods

Kernel interpolation leads to linear system $A\boldsymbol{c} = \boldsymbol{f}$ with matrix

$$A_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i, j = 1, \ldots, N$$

Goal: **Avoid solution of linear systems**

Matrix-free Methods

Kernel interpolation leads to linear system $A\boldsymbol{c} = \boldsymbol{f}$ with matrix

$$A_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i, j = 1, \ldots, N$$

Goal: **Avoid solution of linear systems**

Use cardinal functions in $\text{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$

$$u^*(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_{ij}, \quad i, j, \ldots, N$$

Then

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u^*(\boldsymbol{x}, \boldsymbol{x}_j), \quad \boldsymbol{x} \in \mathbb{R}^s$$

Matrix-free Methods

Kernel interpolation leads to linear system $A\boldsymbol{c} = \boldsymbol{f}$ with matrix

$$A_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i, j = 1, \ldots, N$$

Goal: **Avoid solution of linear systems**

Use cardinal functions in $\text{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$

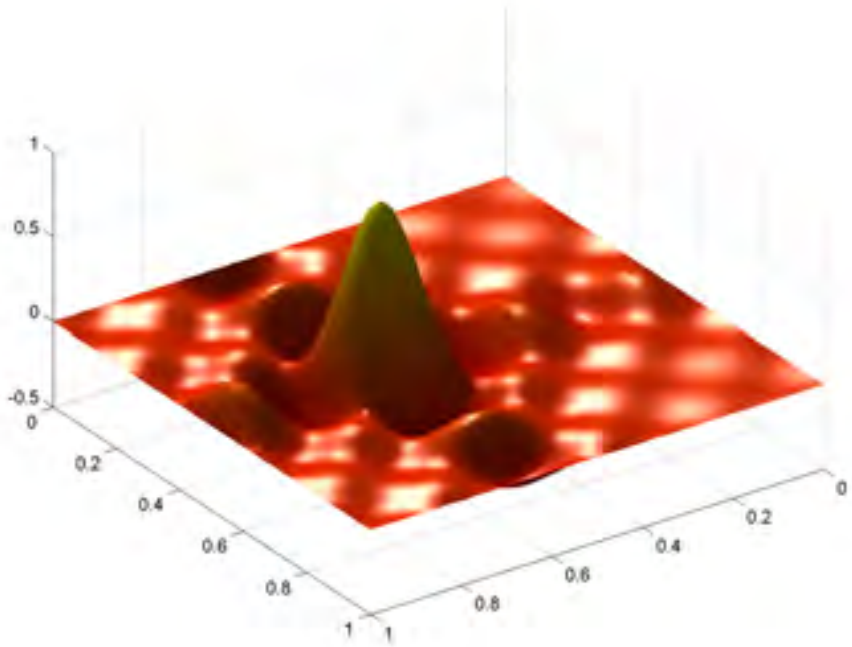
$$u^*(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_{ij}, \quad i, j, \ldots, N$$

Then

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u^*(\boldsymbol{x}, \boldsymbol{x}_j), \quad \boldsymbol{x} \in \mathbb{R}^s$$

Problem: **Cardinal functions difficult/expensive to find**

Cardinal Functions



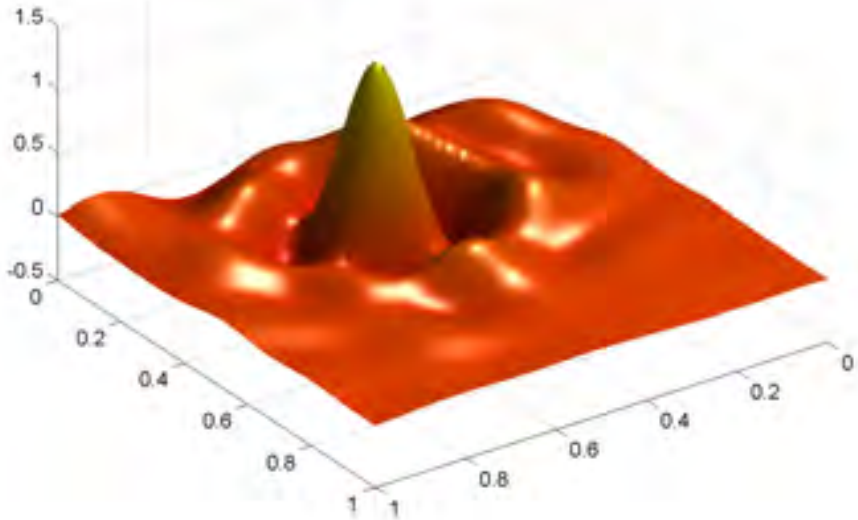


Figure: Cardinal functions centered at an interior point: Gaussian interpolation with $\varepsilon = 5$, 81 uniformly spaced points (left), multiquadric with $\varepsilon = 5$, 81 Halton points (right).

MLS (Backus-Gilbert Formulation)

Assume

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$$

with generating functions $\Psi(\cdot, \boldsymbol{x}_i)$

MLS (Backus-Gilbert Formulation)

Assume

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$$

with generating functions $\Psi(\cdot, \boldsymbol{x}_i)$

Find $\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$ pointwise by solving a linearly constrained quadratic optimization problem.

MLS (Backus-Gilbert Formulation)

Assume

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$$

with generating functions $\Psi(\cdot, \boldsymbol{x}_i)$

Find $\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$ pointwise by solving a linearly constrained quadratic optimization problem.

First discussed in [Bos & Šalkauskas (1989)]

MLS (Backus-Gilbert Formulation)

Assume

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$$

with generating functions $\Psi(\cdot, \boldsymbol{x}_i)$

Find $\Psi(\boldsymbol{x}, \boldsymbol{x}_i)$ pointwise by solving a linearly constrained quadratic optimization problem.

First discussed in [Bos & Šalkauskas (1989)]
Contributions by [Allasia & Giolito (1997), Farwig (1986), Farwig (1987), Farwig (1991), Levin (1998), Wendland (2001)] and many others

Pick positive weight functions $w(\cdot, \boldsymbol{x}_i)$ and minimize

$$\frac{1}{2}\sum_{i=1}^{N} \psi^2(\boldsymbol{x}, \boldsymbol{x}_i)\frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_i)} \quad \Longleftrightarrow \quad \frac{1}{2}\boldsymbol{\Psi}^T(\boldsymbol{x})Q(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}),$$

for fixed evaluation point $\boldsymbol{x}$, where

$$Q(\boldsymbol{x}) = \operatorname{diag}\left(\frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_1)}, \ldots, \frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_N)}\right), \tag{1}$$

and $\boldsymbol{\Psi} = [\psi(\cdot, \boldsymbol{x}_1), \ldots, \psi(\cdot, \boldsymbol{x}_N)]^T$

Pick positive weight functions $w(\cdot, \boldsymbol{x}_i)$ and minimize

$$\frac{1}{2}\sum_{i=1}^{N} \Psi^2(\boldsymbol{x}, \boldsymbol{x}_i)\frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_i)} \quad \Longleftrightarrow \quad \frac{1}{2}\boldsymbol{\Psi}^T(\boldsymbol{x})Q(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}),$$

for fixed evaluation point $\boldsymbol{x}$, where

$$Q(\boldsymbol{x}) = \operatorname{diag}\left(\frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_1)}, \ldots, \frac{1}{w(\boldsymbol{x}, \boldsymbol{x}_N)}\right), \tag{1}$$

and $\boldsymbol{\Psi} = [\Psi(\cdot, \boldsymbol{x}_1), \ldots, \Psi(\cdot, \boldsymbol{x}_N)]^T$
subject to polynomial reproduction (discrete moment conditions)

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i - \boldsymbol{x})\Psi(\boldsymbol{x}, \boldsymbol{x}_i) = p(\boldsymbol{0}), \quad \text{for all } p \in \Pi_d^s \quad \Longleftrightarrow \quad A(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{0})$$

where $A_{ji}(\boldsymbol{x}) = p_j(\boldsymbol{x}_i - \boldsymbol{x})$, $j = 1, \ldots, m = \binom{d+s}{d}$, $i = 1, \ldots, N$

Using Lagrange multipliers $\boldsymbol{\lambda}(\boldsymbol{x}) = [\lambda_1(\boldsymbol{x}), \ldots, \lambda_m(\boldsymbol{x})]^T$ we minimize

$$\frac{1}{2}\boldsymbol{\Psi}^T(\boldsymbol{x})Q(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}) - \boldsymbol{\lambda}^T(\boldsymbol{x})\left[A(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}) - \boldsymbol{p}(\boldsymbol{0})\right]$$

Using Lagrange multipliers $\boldsymbol{\lambda}(\boldsymbol{x}) = [\lambda_1(\boldsymbol{x}), \ldots, \lambda_m(\boldsymbol{x})]^T$ we minimize

$$\frac{1}{2}\boldsymbol{\Psi}^T(\boldsymbol{x})Q(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}) - \boldsymbol{\lambda}^T(\boldsymbol{x})\left[A(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}) - \boldsymbol{p}(\boldsymbol{0})\right]$$

This leads to the system

$$\begin{bmatrix} Q(\boldsymbol{x}) & -A^T(\boldsymbol{x}) \\ A(\boldsymbol{x}) & O \end{bmatrix} \begin{bmatrix} \boldsymbol{\Psi}(\boldsymbol{x}) \\ \boldsymbol{\lambda}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{p}(\boldsymbol{0}) \end{bmatrix}$$

Using Lagrange multipliers $\boldsymbol{\lambda}(\boldsymbol{x}) = [\lambda_1(\boldsymbol{x}), \ldots, \lambda_m(\boldsymbol{x})]^T$ we minimize

$$\frac{1}{2}\boldsymbol{\psi}^T(\boldsymbol{x})Q(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{\lambda}^T(\boldsymbol{x})\left[A(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{p}(\boldsymbol{0})\right]$$

This leads to the system

$$\begin{bmatrix} Q(\boldsymbol{x}) & -A^T(\boldsymbol{x}) \\ A(\boldsymbol{x}) & O \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{x}) \\ \boldsymbol{\lambda}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{p}(\boldsymbol{0}) \end{bmatrix}$$

with solution

$$\begin{aligned} \boldsymbol{\lambda}(\boldsymbol{x}) &= \left(A(\boldsymbol{x})Q^{-1}(\boldsymbol{x})A^T(\boldsymbol{x})\right)^{-1}\boldsymbol{p}(\boldsymbol{0}) \\ \boldsymbol{\psi}(\boldsymbol{x}) &= Q^{-1}(\boldsymbol{x})A^T(\boldsymbol{x})\boldsymbol{\lambda}(\boldsymbol{x}) \end{aligned}$$

If we use a Gram system, the $\lambda_k(\boldsymbol{x})$ are the solution of

$$G(\boldsymbol{x})\boldsymbol{\lambda}(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{0})$$

with Gram matrix

$$G_{j,k}(\boldsymbol{x}) = \sum_{i=1}^{N} p_j(\boldsymbol{x}_i - \boldsymbol{x}) p_k(\boldsymbol{x}_i - \boldsymbol{x}) w(\boldsymbol{x}, \boldsymbol{x}_i)$$

and $\boldsymbol{p} = [p_1, \ldots, p_m]^T$, $\qquad m = \binom{d+s}{d}$

If we use a Gram system, the $\lambda_k(\boldsymbol{x})$ are the solution of

$$G(\boldsymbol{x})\boldsymbol{\lambda}(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{0})$$

with Gram matrix

$$G_{j,k}(\boldsymbol{x}) = \sum_{i=1}^{N} p_j(\boldsymbol{x}_i - \boldsymbol{x}) p_k(\boldsymbol{x}_i - \boldsymbol{x}) w(\boldsymbol{x}, \boldsymbol{x}_i)$$

and $\boldsymbol{p} = [p_1, \ldots, p_m]^T$, $\qquad m = \binom{d+s}{d}$

(Small) linear system for each $\boldsymbol{x}$

If we use a Gram system, the $\lambda_k(\boldsymbol{x})$ are the solution of

$$G(\boldsymbol{x})\lambda(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{0})$$

with Gram matrix

$$G_{j,k}(\boldsymbol{x}) = \sum_{i=1}^{N} p_j(\boldsymbol{x}_i - \boldsymbol{x}) p_k(\boldsymbol{x}_i - \boldsymbol{x}) w(\boldsymbol{x}, \boldsymbol{x}_i)$$

and $\boldsymbol{p} = [p_1, \ldots, p_m]^T$, $\qquad m = \binom{d+s}{d}$

(Small) linear system for each $\boldsymbol{x}$

Following either approach we have componentwise

$$\Psi(\boldsymbol{x}, \boldsymbol{x}_i) = w(\boldsymbol{x}, \boldsymbol{x}_i) \sum_{j=1}^{m} \lambda_j(\boldsymbol{x}) p_j(\boldsymbol{x}_i - \boldsymbol{x}), \quad i = 1, \ldots, N$$

Shepard's Method

Example ($d = 0$)

For any positive weight w

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) \underbrace{\frac{w(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{k=1}^{N} w(\boldsymbol{x}, \boldsymbol{x}_k)}}_{=:\Psi(\boldsymbol{x}, \boldsymbol{x}_j)}$$

partition of unity

Has approximation order $\mathcal{O}(h)$ if $w(\cdot, \boldsymbol{x}_j)$ has support size $\rho_j \propto h$

Does not interpolate — only approximates data

Shepard's Method

Example ($d = 0$)

For any positive weight w

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) \underbrace{\frac{w(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{k=1}^{N} w(\boldsymbol{x}, \boldsymbol{x}_k)}}_{=:\Psi(\boldsymbol{x}, \boldsymbol{x}_j)}$$

partition of unity

Has approximation order $\mathcal{O}(h)$ if $w(\cdot, \boldsymbol{x}_j)$ has support size $\rho_j \propto h$

Does not interpolate — only approximates data

Also known as kernel method or local polynomial regression

Example

Test function

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

Use compactly supported weights

$$w(\boldsymbol{x}_i, \boldsymbol{x}) = (1 - \varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|)_+^4 \, (4\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\| + 1)$$

so that evaluation matrix is sparse
Stationary approximation scheme: $\varepsilon = N^{1/s}$

Program (ShepardCS_sD.m)

```
 1  s = 2;  N = 289; M = 500;
 2  global rbf;  rbf_definition;  ep = nthroot(N,s);
 3  [dsites, N] = CreatePoints(N,s,'h');
 4  ctrs = dsites;
 5  epoints = CreatePoints(M,s,'r');
 6  f = testfunctionsD(dsites);
 7  DM_eval = DistanceMatrixCSRBF(epoints,ctrs,ep);
 8  EM = rbf(ep,DM_eval);
 9  EM = spdiags(1./(EM*ones(N,1)),0,M,M)*EM;
10  Pf = EM*f;
11  exact = testfunctionsD(epoints);
12  maxerr = norm(Pf-exact,inf)
13  rms_err = norm(Pf-exact)/sqrt(M)
```

Remark

- `DistanceMatrixCSRBF` *returns a sparse matrix*
- $\Longrightarrow$ `rbf` *defined differently*

Compactly supported RBFs/weights

To get a sparse matrix from `DistanceMatrixRBF` we express compactly supported functions in a shifted form $\widetilde{\varphi} = \varphi(1 - \cdot)$ so that $\widetilde{\varphi}(1 - \varepsilon r) = \varphi(\varepsilon r)$

k	$\varphi_{3,k}(r)$	$\widetilde{\varphi}_{3,k}(r)$	smoothness
0	$(1-r)_+^2$	r_+^2	C^0
1	$(1-r)_+^4\,(4r+1)$	$r_+^4\,(5-4r)$	C^2
2	$(1-r)_+^6\,(35r^2+18r+3)$	$r_+^6\,(56-88r+35r^2)$	C^4

Table: Wendland functions $\varphi_{s,k}$ and $\widetilde{\varphi}_{s,k} = \varphi_{s,k}(1 - \cdot)$

C^2 Wendland function $\varphi_{3,1}$ in MATLAB

Instead of (full matrix version)

```
rbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1);
```

we now write

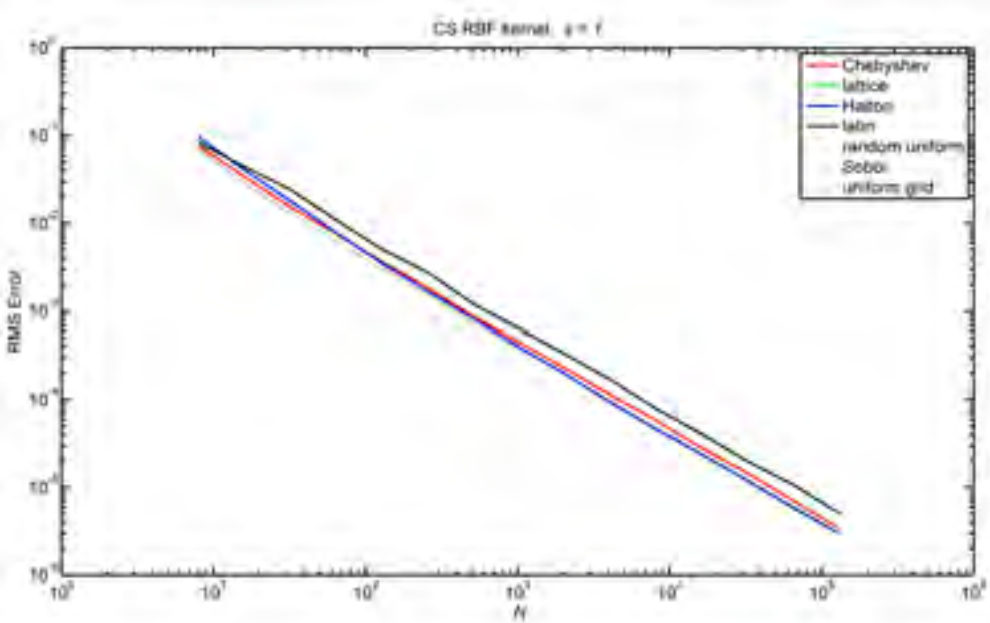
```
rbf = @(e,r) r.^4.*(5*spones(r)-4*r);
```

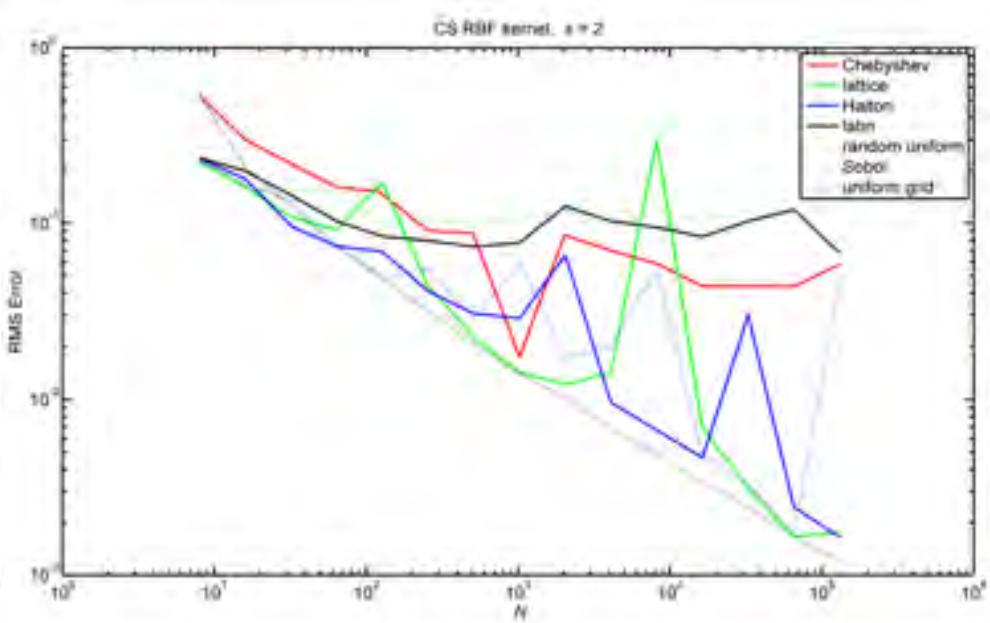
Remark

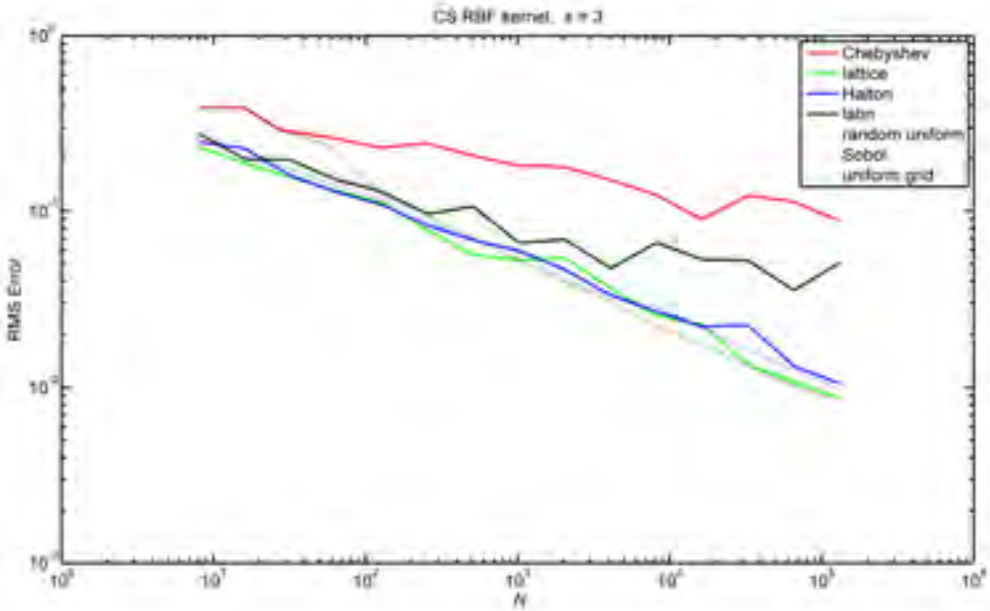
- *We use* `spones` *since* `5-4*r` *would have generated a full matrix (with many additional — and unwanted — ones).*

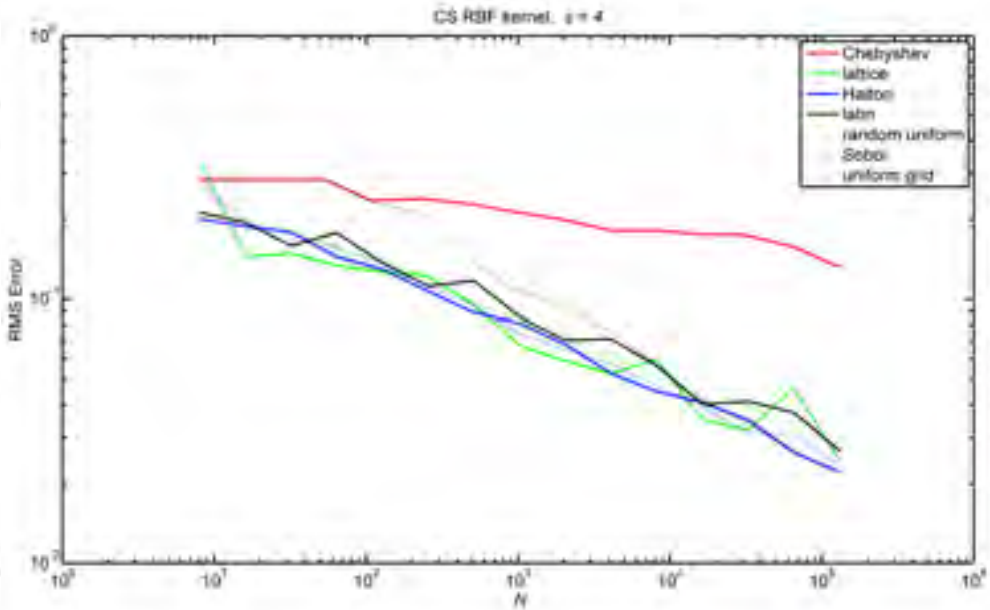
Program (DistanceMatrixCSRBF.m)

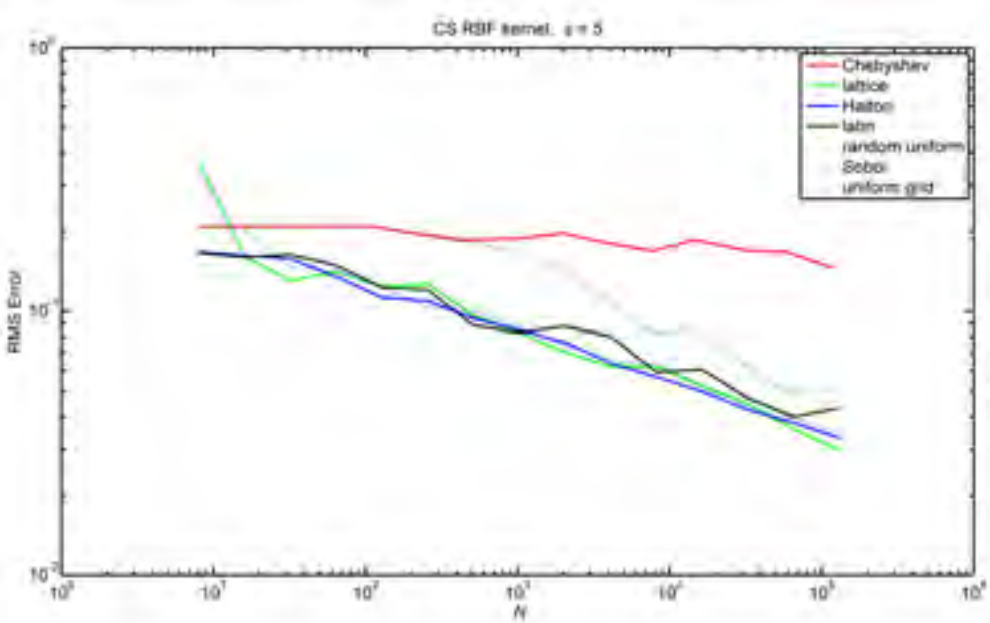
```
 1  function DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
 2  N = size(dsites,1);  M = size(ctrs,1);
    % Build k-D tree for data sites
    % For each ctr/dsite, find the dsites/ctrs
    % in its support along with u-distance u=1-ep*r
 3  supp = 1/ep; nzmax = 25*N; DM = spalloc(N,M,nzmax);
 4  if M > N  % faster if more centers than data sites
 5     [tmp,tmp,T] = kdtree(ctrs,[]);
 6     for i = 1:N
 7        [pts,dist,idx]=kdrangequery(T,dsites(i,:),supp);
 8        DM(i,idx) = 1-ep*dist;
 9     end
10  else
11     [tmp,tmp,T] = kdtree(dsites,[]);
12     for j = 1:M
13        [pts,dist,idx]=kdrangequery(T,ctrs(j,:),supp);
14        DM(idx,j) = 1-ep*dist;
15     end
16  end
17  kdtree([],[],T);
```

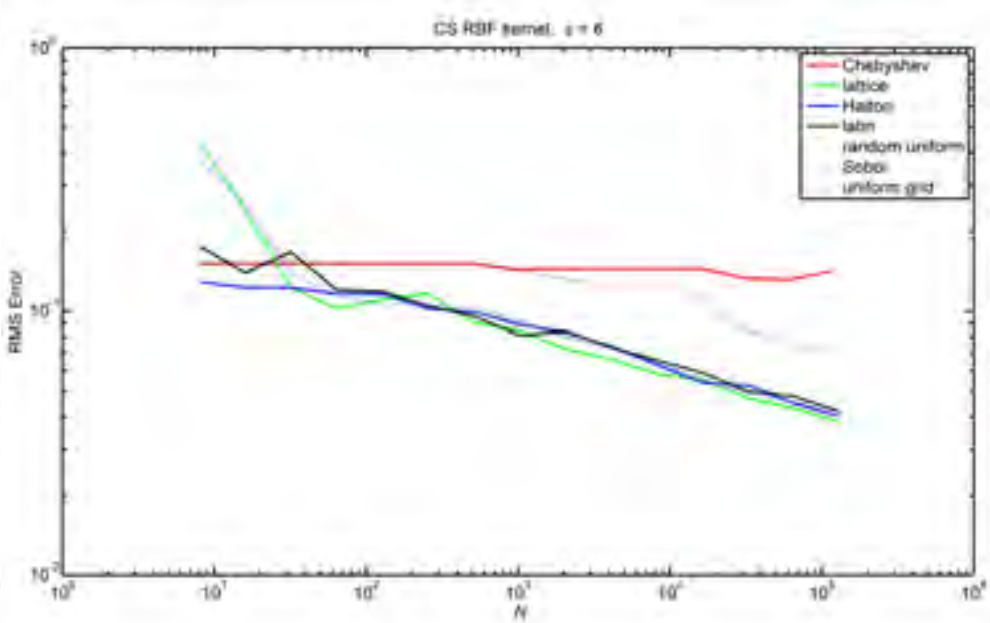

CS RBF kernel, s = 1
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

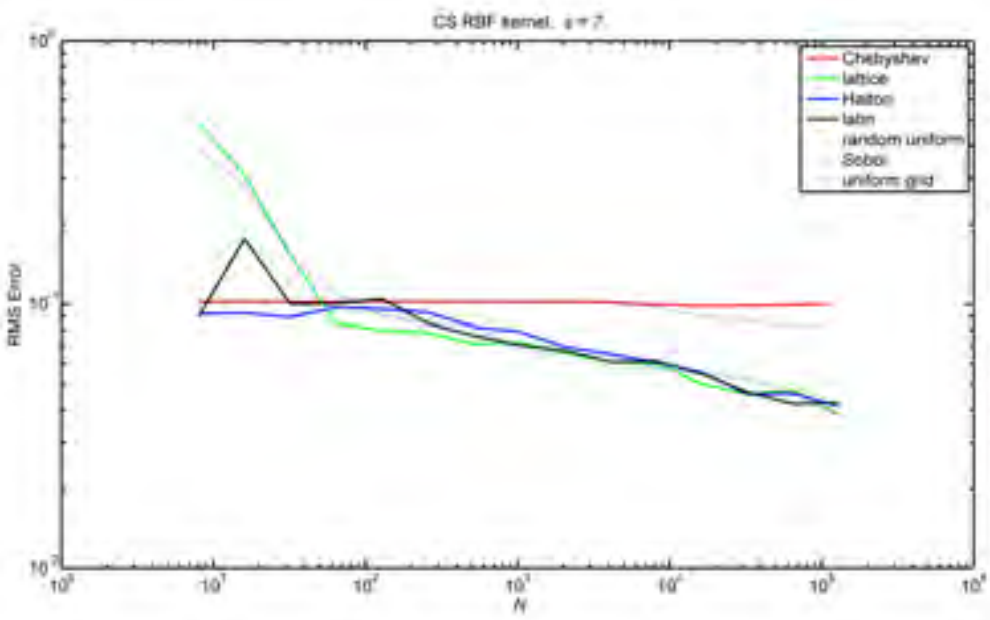

CS RBF kernel, s = 2
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

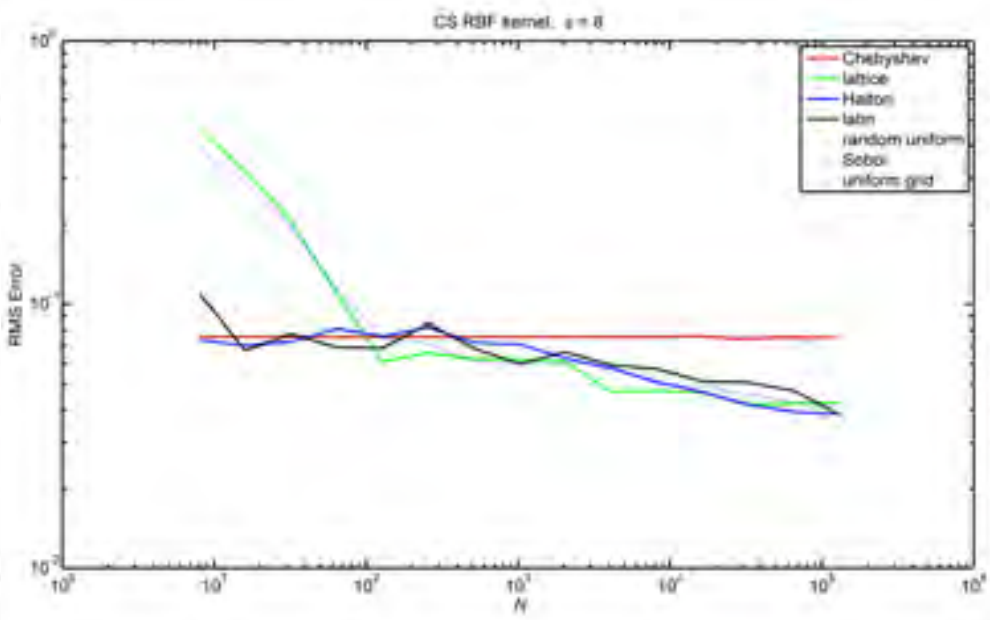

CS RBF kernel, s = 3
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

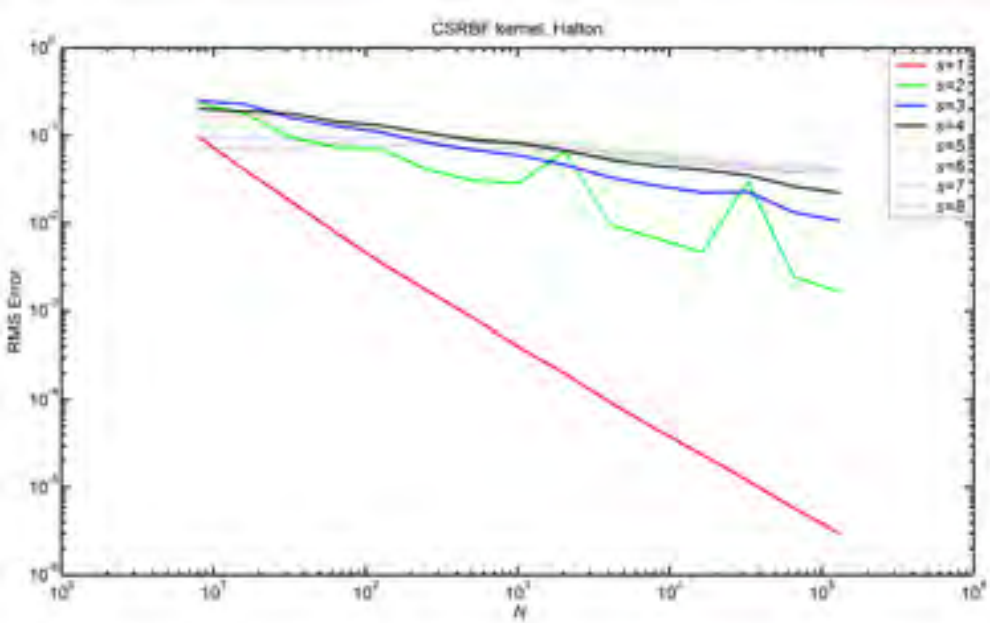

CS RBF kernel, s = 4
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

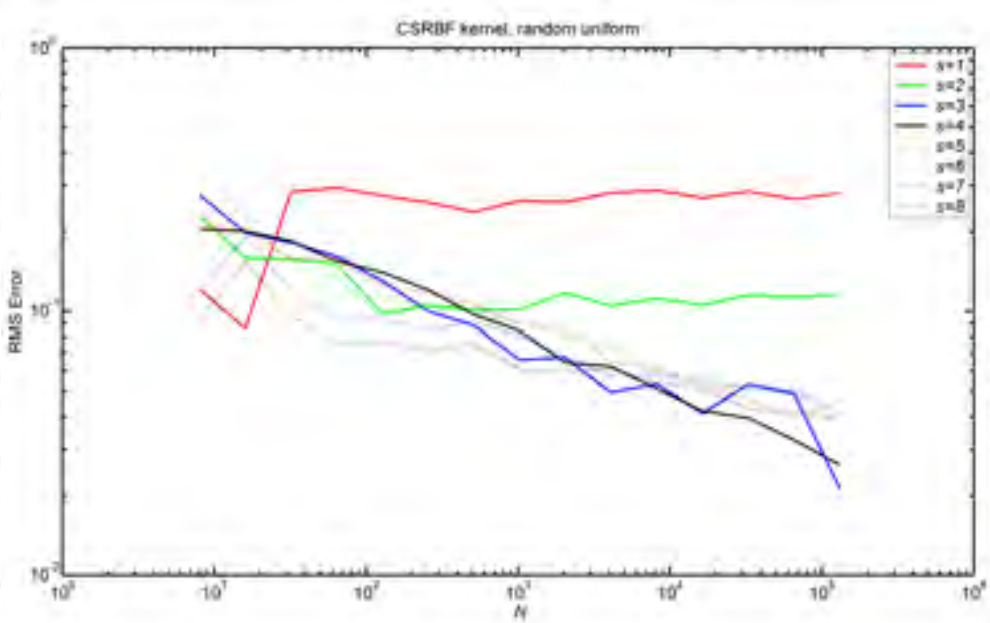

CS RBF kernel, ε = 5
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

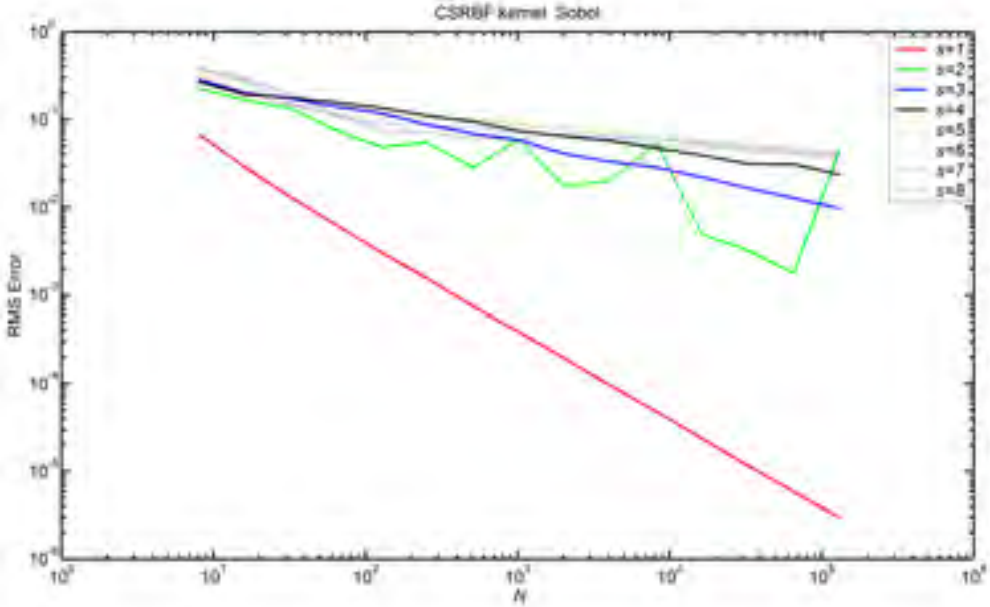

CS RBF kernel, ε = 6
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

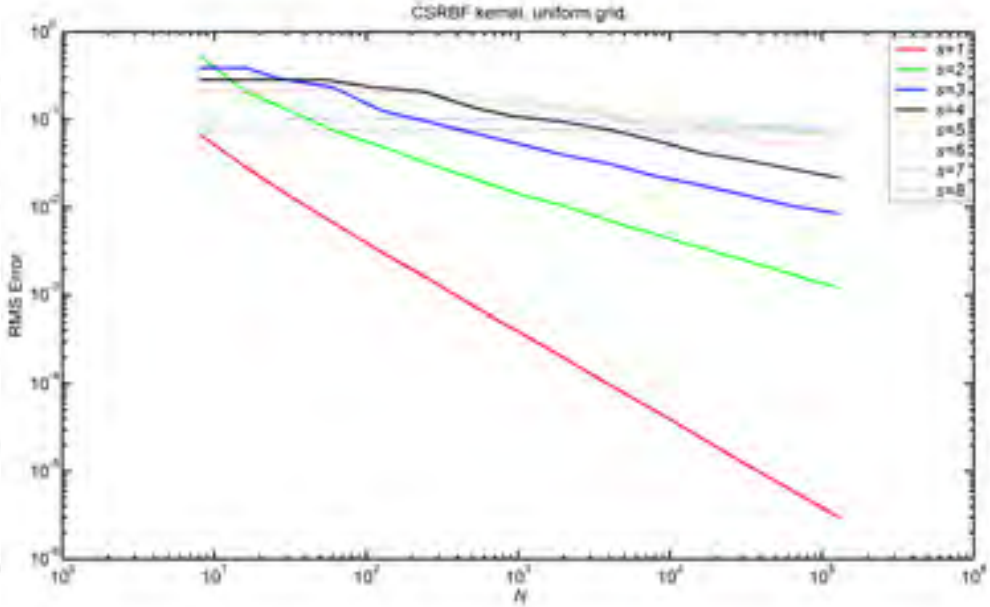

CS RBF kernel, ε = 7
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N


CS RBF kernel, ε = 8
Chebyshev
lattice
Halton
latin
random uniform
Sobol
uniform grid
RMS Error
N

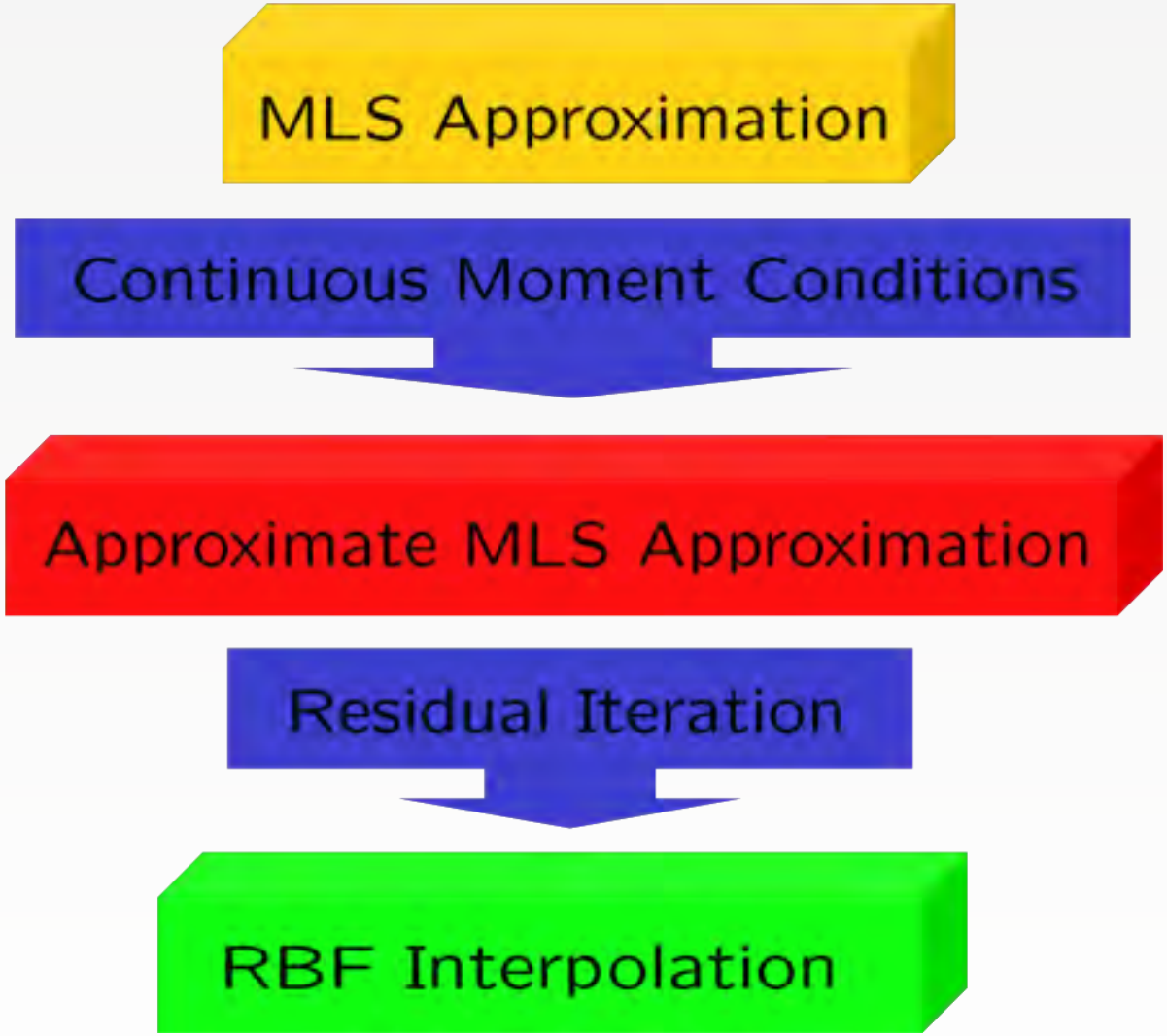

CSRBF kernel, Halton
RMS Error
N
s=1
s=2
s=3
s=4
s=5
s=6
s=7
s=8


CSRBF kernel, random uniform
RMS Error
N
s=1
s=2
s=3
s=4
s=5
s=6
s=7
s=8


CSRBF kernel, Sobol
RMS Error
N
s=1
s=2
s=3
s=4
s=5
s=6
s=7
s=8


CSRBF kernel, uniform grid
RMS Error
N
s=1
s=2
s=3
s=4
s=5
s=6
s=7
s=8

RBF Interpolation via MLS Approximation [Zhang (2007)]



In MLS approximation the generating functions satisfy discrete moment conditions

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i - \boldsymbol{x})\Psi(\boldsymbol{x}, \boldsymbol{x}_i) = p(\mathbf{0}), \quad \text{for all } p \in \Pi_d^s$$

In MLS approximation the generating functions satisfy discrete moment conditions

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i - \boldsymbol{x})\Psi(\boldsymbol{x}, \boldsymbol{x}_i) = p(\mathbf{0}), \quad \text{for all } p \in \Pi_d^s$$

Now we impose continuous moment conditions. If φ is radial we want

$$\int_{\mathbb{R}^s} \|\boldsymbol{x}\|^{2k} \varphi(\|\boldsymbol{x}\|) d\boldsymbol{x} = \delta_{k,0} \quad \text{for } 0 \le k \le d$$

In MLS approximation the generating functions satisfy discrete moment conditions

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i - \boldsymbol{x})\Psi(\boldsymbol{x}, \boldsymbol{x}_i) = p(\boldsymbol{0}), \quad \text{for all } p \in \Pi_d^s$$

Now we impose continuous moment conditions. If φ is radial we want

$$\int_{\mathbb{R}^s} \|\boldsymbol{x}\|^{2k} \varphi(\|\boldsymbol{x}\|) d\boldsymbol{x} = \delta_{k,0} \quad \text{for } 0 \le k \le d$$

Remark

- *The concept of approximate approximations was first suggested by Maz'ya in the early 1990s.*
- *See the recent book [Maz'ya and Schmidt (2007)].*

If φ satisfies the continuous moment conditions, then approximate approximation guarantees that

$$\mathcal{Q}_f(\boldsymbol{x}) = \frac{1}{\mathcal{D}^{s/2}} \sum_{j=1}^{N} f(\boldsymbol{x}_j) \varphi\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_j}{\sqrt{\mathcal{D}} h} \right\| \right)$$

approximates the data with

$$\|f - \mathcal{Q}_f\|_\infty = \mathcal{O}(h^{2d+2}) + \epsilon(\varphi, \mathcal{D})$$

provided $\boldsymbol{x}_j \in \mathbb{R}^s$ are uniformly spaced and $\mathcal{D} \geq 1$

If φ satisfies the continuous moment conditions, then approximate approximation guarantees that

$$\mathcal{Q}_f(\boldsymbol{x}) = \frac{1}{\mathcal{D}^{s/2}} \sum_{j=1}^{N} f(\boldsymbol{x}_j) \varphi\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_j}{\sqrt{\mathcal{D}} h} \right\| \right)$$

approximates the data with

$$\|f - \mathcal{Q}_f\|_\infty = \mathcal{O}(h^{2d+2}) + \epsilon(\varphi, \mathcal{D})$$

provided $\boldsymbol{x}_j \in \mathbb{R}^s$ are uniformly spaced and $\mathcal{D} \geq 1$

Remark

- $\epsilon(\varphi, \mathcal{D})$ *is called saturation error*
- *It depends only on* φ *and the initial scale factor* $\mathcal{D}$

If φ satisfies the continuous moment conditions, then approximate approximation guarantees that

$$\mathcal{Q}_f(\boldsymbol{x}) = \frac{1}{\mathcal{D}^{s/2}} \sum_{j=1}^{N} f(\boldsymbol{x}_j) \varphi\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_j}{\sqrt{\mathcal{D}} h} \right\| \right)$$

approximates the data with

$$\|f - \mathcal{Q}_f\|_\infty = \mathcal{O}(h^{2d+2}) + \epsilon(\varphi, \mathcal{D})$$

provided $\boldsymbol{x}_j \in \mathbb{R}^s$ are uniformly spaced and $\mathcal{D} \geq 1$

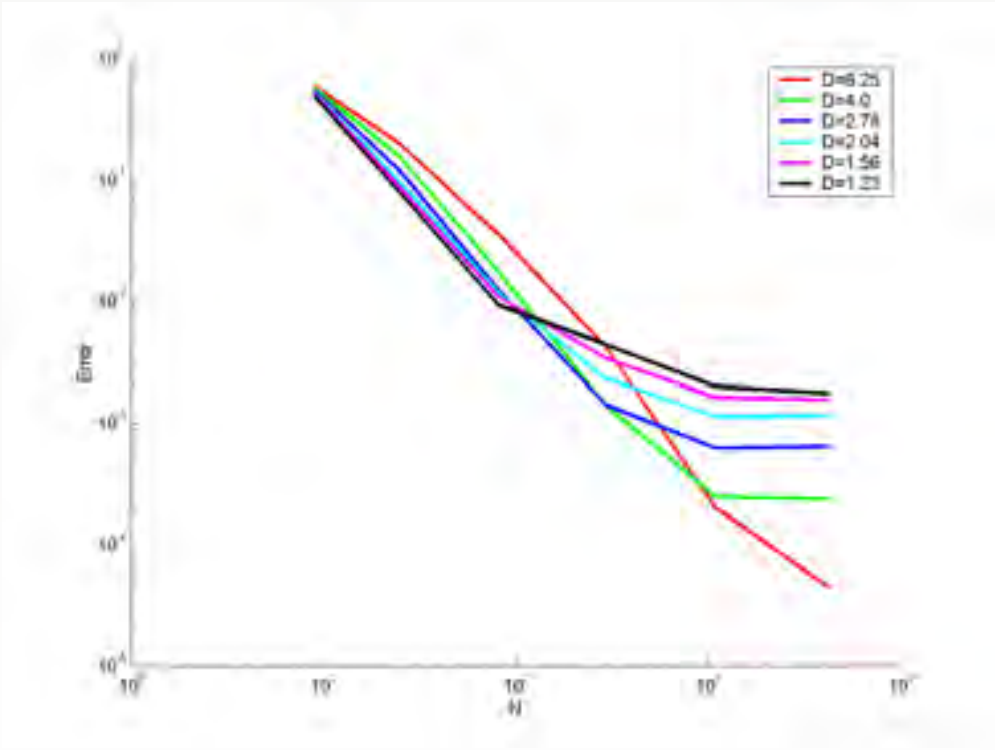
Remark

- $\epsilon(\varphi, \mathcal{D})$ *is called saturation error*
- *It depends only on* φ *and the initial scale factor* $\mathcal{D}$
- *By choosing an appropriate* $\mathcal{D}$*, the saturation error may be pushed down to the level of roundoff error.*

Saturated Gaussian Interpolation

Interpolate with

$$\varphi(r) = e^{-\frac{r^2}{\mathcal{D}h^2}}$$



Summary so far

Data: $\{\boldsymbol{x}_j, f_j\}$, $j = 1, \ldots, N$	
RBF interpolation	**Approximate MLS approximation**
$\mathcal{P}_f(\boldsymbol{x}) = \sum c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$	$\mathcal{Q}_f(\boldsymbol{x}) = \sum f_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$
$\mathcal{P}_f(\boldsymbol{x}_i) = f_i$ (interpolation)	$\mathcal{Q}(\boldsymbol{x}_i) \approx f_i$ (approximation)
c_j unknown	$\Phi(\boldsymbol{x}, \boldsymbol{x}_j)$ unknown
Φ strictly positive definite	Φ meets continuous moment conditions
solve (large) linear system	no linear system to solve

Summary so far

Data: $\{\boldsymbol{x}_j, f_j\}$, $j = 1, \ldots, N$	
RBF interpolation	**Approximate MLS approximation**
$\mathcal{P}_f(\boldsymbol{x}) = \sum c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$	$\mathcal{Q}_f(\boldsymbol{x}) = \sum f_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$
$\mathcal{P}_f(\boldsymbol{x}_i) = f_i$ (interpolation)	$\mathcal{Q}(\boldsymbol{x}_i) \approx f_i$ (approximation)
c_j unknown	$\Phi(\boldsymbol{x}, \boldsymbol{x}_j)$ unknown
Φ strictly positive definite	Φ meets continuous moment conditions
solve (large) linear system	no linear system to solve

Remark

We want to find basic (generating) functions that are both positive definite and satisfy moment conditions.

Some not uncommon misconceptions

- Everyone knows: interpolation matrix is non-singular if Φ is strictly positive definite

Some not uncommon misconceptions

- Everyone knows: interpolation matrix is non-singular if Φ is strictly positive definite
- The literature tells us

Theorem

$\varphi(\|\cdot\|^2)$ *is strictly positive definite and radial on* $\mathbb{R}^s$ *for all* s

$$\Longleftrightarrow$$

φ *is completely monotone and not constant.*

Some not uncommon misconceptions

- Everyone knows: interpolation matrix is non-singular if Φ is strictly positive definite
- The literature tells us

Theorem

$\varphi(\|\cdot\|^2)$ *is strictly positive definite and radial on* $\mathbb{R}^s$ *for all* s

$$\Longleftrightarrow$$

φ *is completely monotone and not constant.*

Definition

φ is completely monotone if

$$(-1)^\ell \varphi^{(\ell)}(r) \geq 0, \qquad r > 0,\ \ell = 0, 1, 2, \ldots$$

Some not uncommon misconceptions

- Everyone knows: interpolation matrix is non-singular if Φ is strictly positive definite
- The literature tells us

Theorem

$\varphi(\|\cdot\|^2)$ *is strictly positive definite and radial on* $\mathbb{R}^s$ *for all* s

$$\Longleftrightarrow$$

φ *is completely monotone and not constant.*

Definition

φ is completely monotone if

$$(-1)^\ell \varphi^{(\ell)}(r) \geq 0, \qquad r > 0,\ \ell = 0, 1, 2, \ldots$$

- Consequence of this definition: φ is non-negative

All we really need is

Theorem

$\varphi(\|\cdot\|^2)$ is strictly positive definite and radial on $\mathbb{R}^s$ for some s

$$\Longleftrightarrow$$

its (radial) Fourier transform is non-negative and not identically equal to zero.

All we really need is

Theorem

$\varphi(\|\cdot\|^2)$ is strictly positive definite and radial on $\mathbb{R}^s$ for some s

$$\Longleftrightarrow$$

its (radial) Fourier transform is non-negative and not identically equal to zero.

Example

- Those well-known non-negative functions (such as Gaussians, inverse MQs)
- Compactly supported RBFs of Wendland, Wu and Buhmann

All we really need is

Theorem

$\varphi(\|\cdot\|^2)$ is strictly positive definite and radial on $\mathbb{R}^s$ for some s

$$\Longleftrightarrow$$

its (radial) Fourier transform is non-negative and not identically equal to zero.

Example

- Those well-known non-negative functions (such as Gaussians, inverse MQs)
- Compactly supported RBFs of Wendland, Wu and Buhmann
- But also
  - oscillatory RBFs of [Fornberg *et al.* (2004)] (Poisson, Schoenberg)
  - Laguerre-Gaussians and generalized IMQs (below)

Definition (Laguerre-Gaussians)

$$\phi(t) = \frac{1}{\sqrt{\pi^s}} e^{-t} L_d^{s/2}(t)$$

Definition (Laguerre-Gaussians)

$$\phi(t) = \frac{1}{\sqrt{\pi^s}} e^{-t} L_d^{s/2}(t)$$

Theorem ([Zhang (2007)])

$\Phi(\boldsymbol{x}) = \phi\left(\|\boldsymbol{x}\|^2\right)$ *is SPD and satisfies* $\int_{\mathbb{R}^s} \boldsymbol{x}^{\boldsymbol{\alpha}} \Phi(\boldsymbol{x}) d\boldsymbol{x} = \delta_{\boldsymbol{\alpha},\mathbf{0}}$, $0 \leq |\boldsymbol{\alpha}| \leq 2d + 1$.

Definition (Laguerre-Gaussians)

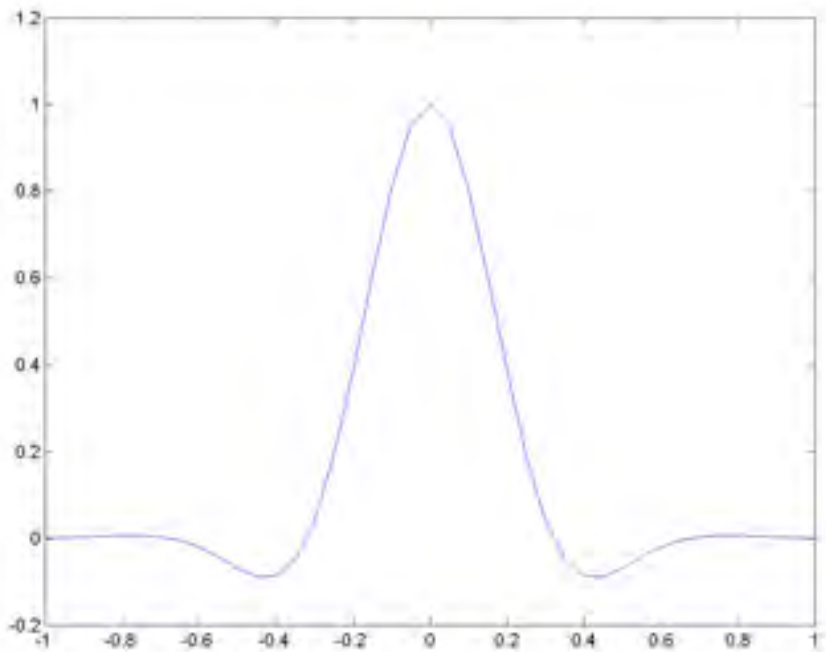
$$\phi(t) = \frac{1}{\sqrt{\pi^s}} e^{-t} L_d^{s/2}(t)$$

Theorem ([Zhang (2007)])

$\Phi(\boldsymbol{x}) = \phi\left(\|\boldsymbol{x}\|^2\right)$ *is SPD and satisfies* $\int_{\mathbb{R}^s} \boldsymbol{x}^{\boldsymbol{\alpha}} \Phi(\boldsymbol{x}) d\boldsymbol{x} = \delta_{\boldsymbol{\alpha},\mathbf{0}}$, $0 \leq |\boldsymbol{\alpha}| \leq 2d+1$.

Examples: $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2} \times$ table entry

$s \backslash d$	0	1	2
1	$\frac{1}{\sqrt{\pi}}$	$\frac{1}{\sqrt{\pi}}\left(\frac{3}{2} - \|\boldsymbol{x}\|^2\right)$	$\frac{1}{\sqrt{\pi}}\left(\frac{15}{8} - \frac{5}{2}\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{x}\|^4\right)$
2	$\frac{1}{\pi}$	$\frac{1}{\pi}\left(2 - \|\boldsymbol{x}\|^2\right)$	$\frac{1}{\pi}\left(3 - 3\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{x}\|^4\right)$
3	$\frac{1}{\pi^{3/2}}$	$\frac{1}{\pi^{3/2}}\left(\frac{5}{2} - \|\boldsymbol{x}\|^2\right)$	$\frac{1}{\pi^{3/2}}\left(\frac{35}{8} - \frac{7}{2}\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{x}\|^4\right)$



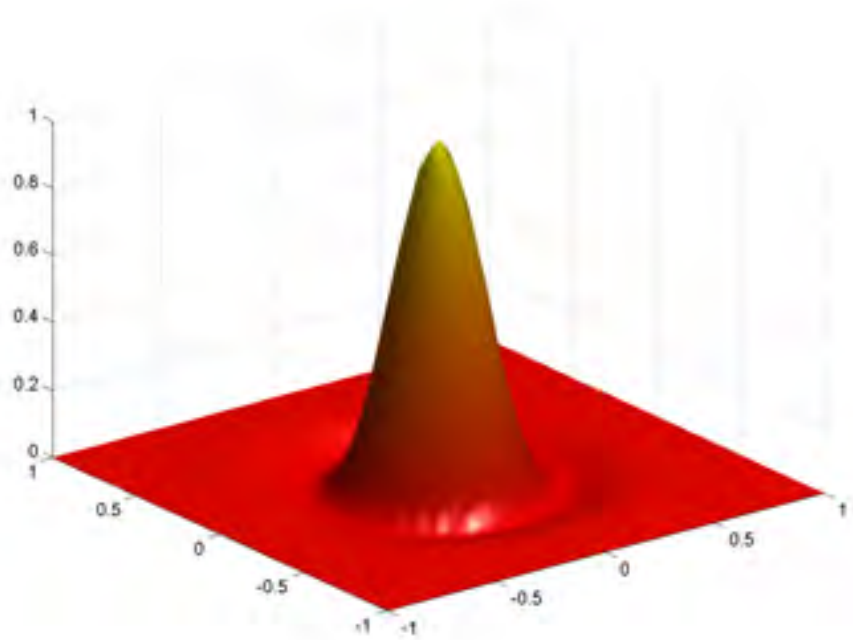


Figure: Laguerre-Gaussians with $s = 1, d = 2$ (left) and $s = 2, d = 2$ (right) centered at the origin.

Definition (Generalized Inverse Multiquadrics)

$$\phi(t) = \frac{1}{\pi^{s/2}} \frac{1}{(1+t)^{2d+s}} \sum_{j=0}^{d} \frac{(-1)^j (2d+s-j-1)!(1+t)^j}{(d-j)!j!\Gamma(d+s/2-j)}$$

Definition (Generalized Inverse Multiquadrics)

$$\phi(t) = \frac{1}{\pi^{s/2}} \frac{1}{(1+t)^{2d+s}} \sum_{j=0}^{d} \frac{(-1)^j (2d+s-j-1)!(1+t)^j}{(d-j)!j!\Gamma(d+s/2-j)}$$

Theorem ([Zhang (2007)])

$\Phi(\boldsymbol{x}) = \phi\left(\|\boldsymbol{x}\|^2\right)$ is SPD and satisfies $\int_{\mathbb{R}^d} \boldsymbol{x}^{\boldsymbol{\alpha}} \Phi(\boldsymbol{x}) d\boldsymbol{x} = \delta_{\boldsymbol{\alpha},\mathbf{0}}$,
$0 \leq |\boldsymbol{\alpha}| \leq 2d+1$.

Definition (Generalized Inverse Multiquadrics)

$$\phi(t) = \frac{1}{\pi^{s/2}} \frac{1}{(1+t)^{2d+s}} \sum_{j=0}^{d} \frac{(-1)^j (2d+s-j-1)!(1+t)^j}{(d-j)!j!\Gamma(d+s/2-j)}$$

Theorem ([Zhang (2007)])

$\Phi(\boldsymbol{x}) = \phi\left(\|\boldsymbol{x}\|^2\right)$ *is SPD and satisfies* $\int_{\mathbb{R}^d} \boldsymbol{x}^{\boldsymbol{\alpha}} \Phi(\boldsymbol{x}) d\boldsymbol{x} = \delta_{\boldsymbol{\alpha},\mathbf{0}}$, $0 \leq |\boldsymbol{\alpha}| \leq 2d+1$.

Examples: $\Phi(\boldsymbol{x})$

$s \backslash d$	0	1	2
1	$\frac{1}{\pi} \frac{1}{1+\|\boldsymbol{x}\|^2}$	$\frac{1}{\pi} \frac{(3-\|\boldsymbol{x}\|^2)}{(1+\|\boldsymbol{x}\|^2)^3}$	$\frac{1}{\pi} \frac{(5-10\|\boldsymbol{x}\|^2+\|\boldsymbol{x}\|^4)}{(1+\|\boldsymbol{x}\|^2)^5}$
2	$\frac{1}{\pi} \frac{1}{(1+\|\boldsymbol{x}\|^2)^2}$	$\frac{2}{\pi} \frac{(2-\|\boldsymbol{x}\|^2)}{(1+\|\boldsymbol{x}\|^2)^4}$	$\frac{3}{\pi} \frac{(3-6\|\boldsymbol{x}\|^2+\|\boldsymbol{x}\|^4)}{(1+\|\boldsymbol{x}\|^2)^6}$
3	$\frac{4}{\pi^2} \frac{1}{(1+\|\boldsymbol{x}\|^2)^3}$	$\frac{4}{\pi^2} \frac{(5-3\|\boldsymbol{x}\|^2)}{(1+\|\boldsymbol{x}\|^2)^5}$	$\frac{8}{\pi^2} \frac{(7-14\|\boldsymbol{x}\|^2+3\|\boldsymbol{x}\|^4)}{(1+\|\boldsymbol{x}\|^2)^7}$

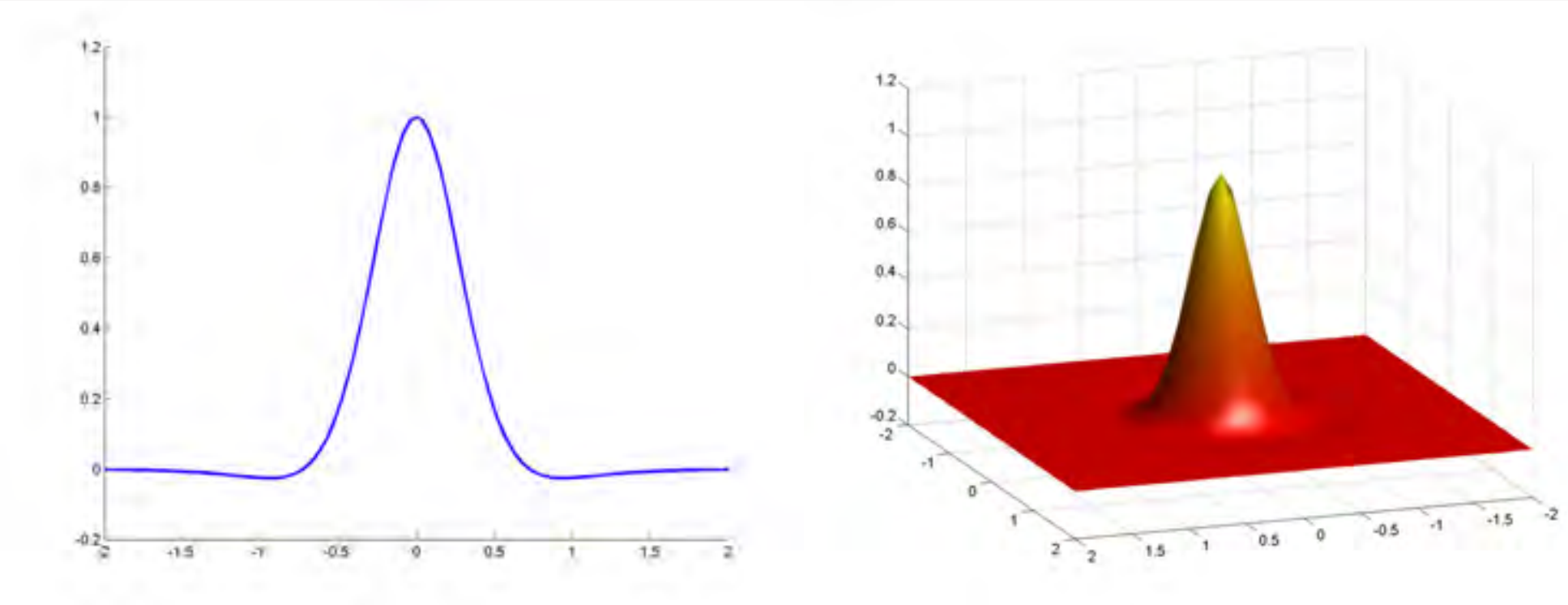


Figure: Generalized inverse MQ with $s = 1, d = 2$ (left) and $s = 2, d = 2$ (right) centered at the origin.

Data: $\{\boldsymbol{x}_j, f_j\}$, $j = 1, \ldots, N$	
RBF interpolation	**Approximate MLS approximation**
$\mathcal{P}_f(\boldsymbol{x}) = \sum c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$	$\mathcal{Q}_f(\boldsymbol{x}) = \sum f_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$
$\mathcal{P}_f(\boldsymbol{x}_i) = f_i$ (interpolation)	$\mathcal{Q}(\boldsymbol{x}_i) \approx f_i$ (approximation)
c_j unknown	$\Phi(\boldsymbol{x}, \boldsymbol{x}_j)$ unknown
Φ strictly positive definite	Φ meets continuous moment conditions
solve (large) linear system	no linear system to solve

Data: $\{\boldsymbol{x}_j, f_j\}$, $j = 1, \ldots, N$	
RBF interpolation	**Approximate MLS approximation**
$\mathcal{P}_f(\boldsymbol{x}) = \sum c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$	$\mathcal{Q}_f(\boldsymbol{x}) = \sum f_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$
$\mathcal{P}_f(\boldsymbol{x}_i) = f_i$ (interpolation)	$\mathcal{Q}(\boldsymbol{x}_i) \approx f_i$ (approximation)
c_j unknown	$\Phi(\boldsymbol{x}, \boldsymbol{x}_j)$ unknown
Φ strictly positive definite	Φ meets continuous moment conditions
solve (large) linear system	no linear system to solve

Iterated approximate MLS approximation

Φ strictly positive definite and meets continuous moment conditions

$$\mathcal{Q}_f^{(0)}(\boldsymbol{x}) = \sum f_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) \quad \text{(approximate MLS approximation)}$$

$$\mathcal{Q}_f^{(1)}(\boldsymbol{x}) = \mathcal{Q}_f^{(0)}(\boldsymbol{x}) + \sum \left[f_j - \mathcal{Q}_f^{(0)}(\boldsymbol{x}_j) \right] \Phi(\boldsymbol{x}, \boldsymbol{x}_j) \quad \text{(residual update)}$$

$$\vdots$$

$$\mathcal{Q}_f^{(\infty)}(\boldsymbol{x}) = \sum c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) \quad \text{(RBF interpolation)}$$

Properties of RBF and MLS methods

- RBFs can be applied without any restriction on the location of the data sites
- approximate MLS (AMLS) mainly applicable to uniformly spaced data

Remark

Approximate approximation for scattered data is significantly more complicated than in the case of uniform data (see, e.g., [Maz'ya and Schmidt (2007)]).

Other properties of RBF and AMLS approximation

- RBFs are known to yield the best approximation to given (scattered) data with respect to the native space norm of the basic function used.
- With RBFs one needs to solve a (generally) large system of linear equations which can also be ill-conditioned.
- Using the AMLS method the solution is obtained via a simple sum based directly on the given data. Thus, the AMLS method is a quasi-interpolation approach.
- The drawback associated with the simplicity of the AMLS method is its lesser degree of accuracy.

Iterative Refinement

For solution of $A\boldsymbol{x} = \boldsymbol{b}$ in numerical linear algebra

1. Compute an approximate solution $\boldsymbol{x}_0$ of $A\boldsymbol{x} = \boldsymbol{b}$
2. For $n = 1, 2, \ldots$ do
   1. Compute the residual $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_{n-1}$
   2. Solve $A\boldsymbol{e}_n = \boldsymbol{r}_n$
   3. Update $\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \boldsymbol{e}_n$

Iterative Refinement for AMLS

1. Initialize $\boldsymbol{r}^{(0)} = \boldsymbol{f}$

$$\mathcal{Q}_f^{(0)}(\boldsymbol{x}) = \sum_{j=1}^{N} r_j^{(0)} \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$$

2. For $n = 1, 2, \ldots$ do
   1. Find the new residuals at the data points

$$r_i^{(n)} = r_i^{(n-1)} - \sum_{j=1}^{N} r_j^{(n-1)} \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i = 1, \ldots, N$$

   2. Update the approximation

$$\mathcal{Q}_f^{(n)}(\boldsymbol{x}) = \mathcal{Q}_f^{(n-1)}(\boldsymbol{x}) + \sum_{j=1}^{N} r_j^{(n)} \Phi(\boldsymbol{x}, \boldsymbol{x}_j)$$

Theorem

Part I (without acceleration)

$$\mathcal{Q}_f^{(n)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \, \boldsymbol{f} =: \mathbf{\Phi}^{(n)T} \boldsymbol{f},$$

i.e., $\{\Phi^{(n)}(\cdot, \boldsymbol{x}_1), \ldots, \Phi^{(n)}(\cdot, \boldsymbol{x}_N)\}$ *provides new — approximately cardinal — basis for* $\operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$.

Theorem

Part I (without acceleration)

$$\mathcal{Q}_f^{(n)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \, \boldsymbol{f} =: \mathbf{\Phi}^{(n)T} \boldsymbol{f},$$

i.e., $\{\Phi^{(n)}(\cdot, \boldsymbol{x}_1), \ldots, \Phi^{(n)}(\cdot, \boldsymbol{x}_N)\}$ *provides new — approximately cardinal — basis for* $\operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$.

Part II (with acceleration)

$$\widetilde{\mathcal{Q}}_f^{(n)} = \mathbf{\Phi}^T \left[\sum_{k=0}^{2^n - 1} (I - A)^k \right] \boldsymbol{f}.$$

Theorem

Part I (without acceleration)

$$\mathcal{Q}_f^{(n)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \, \boldsymbol{f} =: \mathbf{\Phi}^{(n)T} \boldsymbol{f},$$

i.e., $\{\Phi^{(n)}(\cdot, \boldsymbol{x}_1), \ldots, \Phi^{(n)}(\cdot, \boldsymbol{x}_N)\}$ *provides new — approximately cardinal — basis for* $\operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$.

Part II (with acceleration)

$$\widetilde{\mathcal{Q}}_f^{(n)} = \mathbf{\Phi}^T \left[\sum_{k=0}^{2^n - 1} (I - A)^k \right] \boldsymbol{f}.$$

Remark

Theorem can be formulated for any quasi-interpolation scheme provided iteration converges ($\|I - A\| < 1$) and limiting interpolant exists (A non-singular).

Proof of Part I.

By induction

$$\mathcal{Q}_f^{(n+1)} \quad \stackrel{def}{=} \quad \mathcal{Q}_f^{(n)} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{Q}_f^{(n)}(\boldsymbol{x}_j) \right] \Phi(\cdot, \boldsymbol{x}_j)$$

Proof of Part I.

By induction

$$\mathcal{Q}_f^{(n+1)} \overset{def}{=} \mathcal{Q}_f^{(n)} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{Q}_f^{(n)}(\boldsymbol{x}_j) \right] \Phi(\cdot, \boldsymbol{x}_j)$$

$$\overset{IH}{=} \boldsymbol{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \boldsymbol{\Phi}^T(\boldsymbol{x}_j) \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} \right] \Phi(\cdot, \boldsymbol{x}_j)$$

Proof of Part I.

By induction

$$\begin{aligned}
\mathcal{Q}_f^{(n+1)} &\overset{def}{=} \mathcal{Q}_f^{(n)} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{Q}_f^{(n)}(\boldsymbol{x}_j) \right] \Phi(\cdot, \boldsymbol{x}_j) \\
&\overset{IH}{=} \boldsymbol{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \boldsymbol{\Phi}^T(\boldsymbol{x}_j) \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} \right] \Phi(\cdot, \boldsymbol{x}_j) \\
&= \boldsymbol{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \boldsymbol{\Phi}^T \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}
\end{aligned}$$

Proof of Part I.

By induction

$$\begin{aligned}
\mathcal{Q}_f^{(n+1)} &\overset{def}{=} \mathcal{Q}_f^{(n)} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{Q}_f^{(n)}(\boldsymbol{x}_j) \right] \Phi(\cdot, \boldsymbol{x}_j) \\
&\overset{IH}{=} \boldsymbol{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \boldsymbol{\Phi}^T(\boldsymbol{x}_j) \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} \right] \Phi(\cdot, \boldsymbol{x}_j) \\
&= \boldsymbol{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \boldsymbol{\Phi}^T \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}
\end{aligned}$$

Simplify further

$$\begin{aligned}
\mathcal{Q}_f^{(n+1)} &= \boldsymbol{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k+1} \right] \boldsymbol{f} \\
&= \boldsymbol{\Phi}^T \left[\sum_{k=0}^{n+1} (I - A)^k \right] \boldsymbol{f} = \boldsymbol{\Phi}^{(n+1)T} \boldsymbol{f}
\end{aligned}$$

Proof of Part II.

As in Part I: $$\mathcal{Q}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \mathbf{\Phi}^T \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

Proof of Part II.

As in Part I: $$\mathcal{Q}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \mathbf{\Phi}^T \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

Replace $\mathbf{\Phi}^T$ by $\mathbf{\Phi}^{(n)T}$:

$$\widetilde{\mathcal{Q}}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \mathbf{\Phi}^{(n)T} \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

Proof of Part II.

As in Part I: $\mathcal{Q}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \mathbf{\Phi}^T \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$

Replace $\mathbf{\Phi}^T$ by $\mathbf{\Phi}^{(n)T}$:

$$\begin{aligned}
\widetilde{\mathcal{Q}}_f^{(n+1)} &= \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \boldsymbol{f} + \mathbf{\Phi}^{(n)T} \left[I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^{(n)T} \left[2I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \left[2I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}
\end{aligned}$$

Proof of Part II.

As in Part I: $\mathcal{Q}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \boldsymbol{f} + \mathbf{\Phi}^T \left[I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f}$

Replace $\mathbf{\Phi}^T$ by $\mathbf{\Phi}^{(n)T}$:

$$\begin{aligned}
\widetilde{\mathcal{Q}}_f^{(n+1)} &= \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \boldsymbol{f} + \mathbf{\Phi}^{(n)T} \left[I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^{(n)T} \left[2I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \left[2I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^T \left[\sum_{k=0}^{2n+1} (I-A)^k \right] \boldsymbol{f} = \mathbf{\Phi}^{(2n+1)T} \boldsymbol{f} = \mathcal{Q}_f^{(2n+1)}
\end{aligned}$$

Proof of Part II.

As in Part I: $\mathcal{Q}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \boldsymbol{f} + \mathbf{\Phi}^T \left[I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f}$

Replace $\mathbf{\Phi}^T$ by $\mathbf{\Phi}^{(n)T}$:

$$\begin{aligned}
\widetilde{\mathcal{Q}}_f^{(n+1)} &= \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \boldsymbol{f} + \mathbf{\Phi}^{(n)T} \left[I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^{(n)T} \left[2I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^T \sum_{k=0}^{n} (I-A)^k \left[2I - A \sum_{k=0}^{n} (I-A)^k \right] \boldsymbol{f} \\
&= \mathbf{\Phi}^T \left[\sum_{k=0}^{2n+1} (I-A)^k \right] \boldsymbol{f} = \mathbf{\Phi}^{(2n+1)T} \boldsymbol{f} = \mathcal{Q}_f^{(2n+1)}
\end{aligned}$$

We are done by observing that the upper limit of summation satisfies $\tilde{a}_{n+1} = 2\tilde{a}_n + 1$, $\tilde{a}_0 = 0$, i.e., $\tilde{a}_n = 2^n - 1$. □

What about convergence?

- Necessary and sufficient condition for convergence: $\|I - A\|_2 < 1$

What about convergence?

- Necessary and sufficient condition for convergence: $\|I - A\|_2 < 1$
- Sufficient condition:

$$\max_{i=1,2,\ldots,N} \left\{ \sum_{j=1}^{N} |A_{i,j}| \right\} < 2,$$

Here A is specially scaled. For example, scaled s-dimensional Gaussian,

$$\varphi(r) = \frac{\varepsilon^s}{\sqrt{\pi^s}} e^{-\varepsilon^2 r^2 / h^2}$$

For proofs of both see [F. & Zhang (2007)].

What about convergence?

- Necessary and sufficient condition for convergence: $\|I - A\|_2 < 1$
- Sufficient condition:

$$\max_{i=1,2,\ldots,N} \left\{ \sum_{j=1}^{N} |A_{i,j}| \right\} < 2,$$

Here A is specially scaled. For example, scaled s-dimensional Gaussian,

$$\varphi(r) = \frac{\varepsilon^s}{\sqrt{\pi^s}} e^{-\varepsilon^2 r^2/h^2}$$

For proofs of both see [F. & Zhang (2007)].

Remark

- *For convergence ε must be chosen quite small.*
- *For such a choice the iteration will converge very slowly.*
- *BUT, allows stable computation for small ε*

Program (IAMLS_sD.m)

```
 1  s = 2;  N = 289;  M = 500;  maxn = 50;
 2  global rbf;  rbf_definition;  D = 2*s;
 3  [dsites, N] = CreatePoints(N,s,'h');
 4  ctrs = dsites;
 5  epoints = CreatePoints(M,s,'r');
 6  rhs = testfunctionsD(dsites);
 7  h = 1/(nthroot(N,s)-1); ep = 1/(sqrt(D)*h);
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9  IM = rbf(ep,DM_data)/(sqrt(pi*D)^s);
10  DM_eval = DistanceMatrix(epoints,ctrs);
11  EM = rbf(ep,DM_eval)/(sqrt(pi*D)^s);
12  Pf = EM*rhs;
13  maxerr(1) = max(abs(Pf - exact));
14  rms_err(1) = norm(Pf-exact)/sqrt(M);
15  for n=2:maxn
16      rhs = rhs - IM*rhs;
17      Pf = Pf + EM*rhs;
18      maxerr(n) = max(abs(Pf - exact));
19      rms_err(n) = norm(Pf-exact)/sqrt(M);
20  end
```

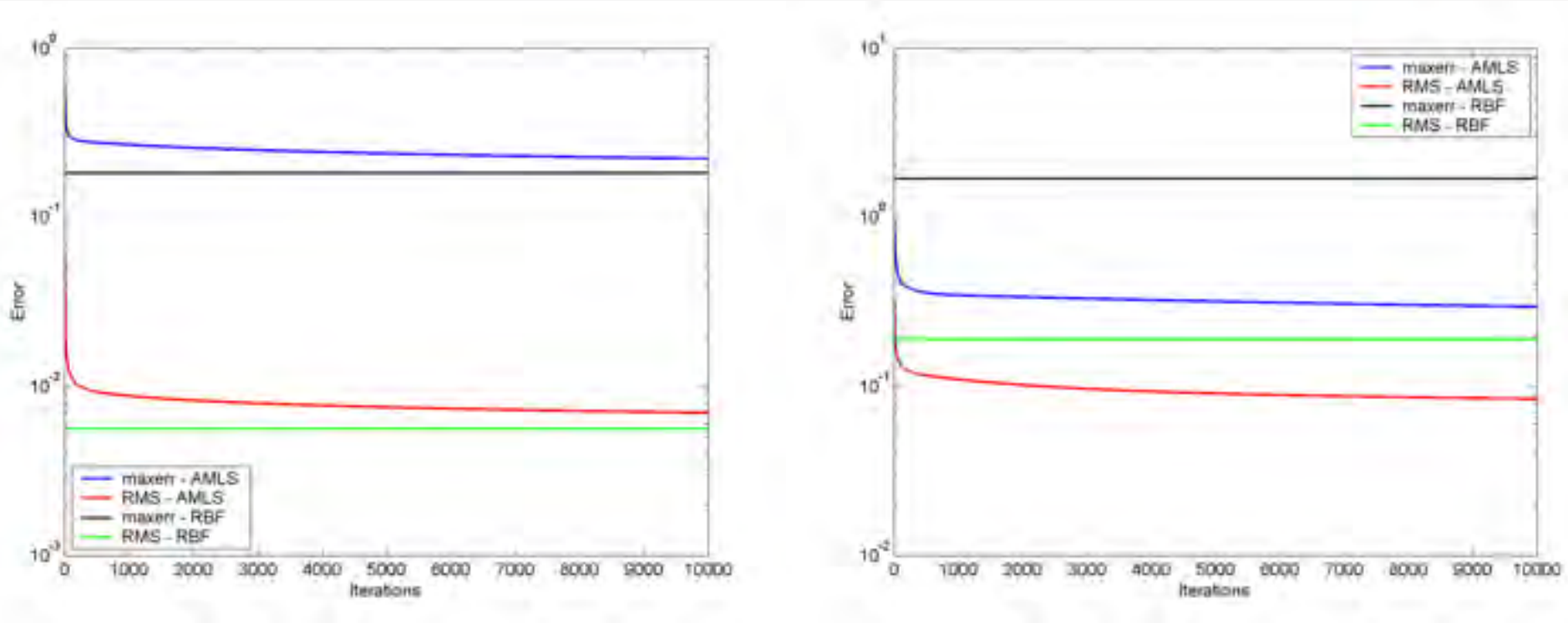


Figure: Convergence of iterated AMLS approximant for 1089 Halton points ($\varepsilon = 16$, left) and 289 Halton points ($\varepsilon = 1$,right).

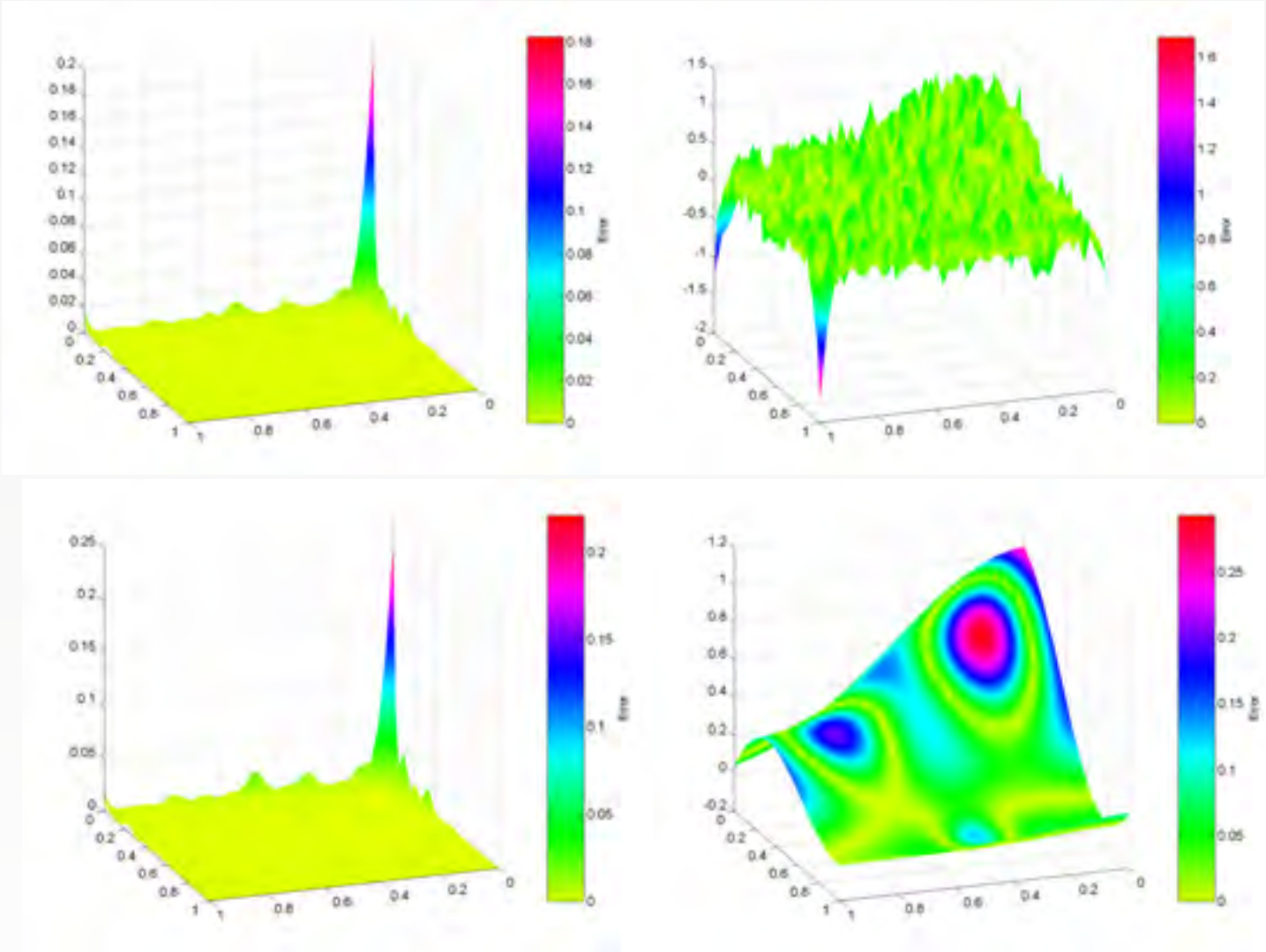


Figure: Comparison for RBF interpolation (top) and IAMLS approximation (bottom) for 1089 Halton points ($\varepsilon = 16$, left, errors) and 289 Halton points ($\varepsilon = 1$, right, fits).

Franke-like test function

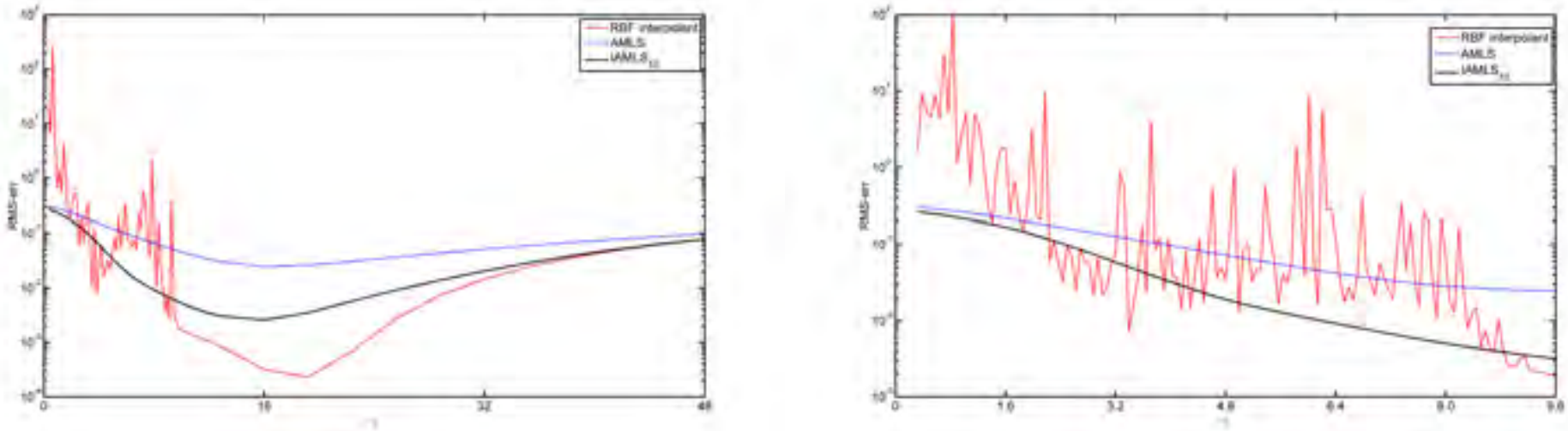


Figure: Accuracy and stability of RBF interpolant, AMLS approximant, and iterated AMLS approximant for 1089 Halton data points in 2D.

- ε "large" if $\varepsilon > 38$ (spiky surfaces for both RBF and AMLS)
- ε too large for convergence (maximum row sum > 2) if $\varepsilon > 48$
- Rapid convergence for $38 < \varepsilon < 58$ (spiky surface, but IAMLS usually smoother)
- "Good" interpolant (slow convergence of IAMLS) for $12 < \varepsilon < 38$, often contains "optimal" ε
- Small ε ($\varepsilon < 12$ here), then IAMLS more stable and may overcome ill-conditioning

From the proof of Part I:

$$\begin{aligned} \mathcal{Q}_f^{(n+1)} &= \boldsymbol{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k+1} \right] \boldsymbol{f} \\ &= \boldsymbol{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k} (I - A) \right] \boldsymbol{f} \end{aligned}$$

From the proof of Part I:

$$\begin{aligned} \mathcal{Q}_f^{(n+1)} &= \boldsymbol{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k+1} \right] \boldsymbol{f} \\ &= \boldsymbol{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k} (I - A) \right] \boldsymbol{f} \end{aligned}$$

Therefore, with $P^{(n)} = \sum_{k=0}^{n} (I - A)^k$, evaluation on the data sites yields

$$\mathcal{Q}_f^{(n+1)} = A \left[I + P^{(n)} (I - A) \right] \boldsymbol{f}$$

From the proof of Part I:

$$\begin{aligned} \mathcal{Q}_f^{(n+1)} &= \mathbf{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k+1} \right] \boldsymbol{f} \\ &= \mathbf{\Phi}^T \left[I + \sum_{k=0}^{n} (I - A)^{k} (I - A) \right] \boldsymbol{f} \end{aligned}$$

Therefore, with $P^{(n)} = \sum_{k=0}^{n} (I - A)^k$, evaluation on the data sites yields

$$\mathcal{Q}_f^{(n+1)} = A \left[I + P^{(n)} (I - A) \right] \boldsymbol{f}$$

or

$$P^{(n+1)} = I + P^{(n)} (I - A)$$

Program (IAMLSVectorized_sD.m)

```
 1  s = 2;  N = 289;  M = 500;  maxn = 50;
 2  global rbf;  rbf_definition;  D = 2*s;
 3  [dsites, N] = CreatePoints(N,s,'h');
 4  ctrs = dsites;  respts = dsites;
 5  epoints = CreatePoints(M,s,'r');
 6  rhs = testfunctionsD(dsites);
 7  h = 1/(nthroot(N,s)-1); ep = 1/(sqrt(D)*h);
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9  IM = rbf(ep,DM_data)/(sqrt(pi*D)^s);
10  DM_eval = DistanceMatrix(epoints,ctrs);
11  EM = rbf(ep,DM_eval)/(sqrt(pi*D)^s);
12  P = eye(N);
13  for n=1:maxn
14     P = eye(N) + P*(eye(N)-IM);
15     Pf = EM*P*rhs;
16     maxerr(n) = norm(Pf-exact,inf);
17     rms_err(n) = norm(Pf-exact)/sqrt(M);
18  end
```

From the proof of Part II:

$$\widetilde{\mathcal{Q}}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \left[2I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

From the proof of Part II:

$$\widetilde{\mathcal{Q}}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \left[2I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

Therefore, with $P^{(n)} = \sum_{k=0}^{n} (I - A)^k$, evaluation on the data sites yields

$$\widetilde{\boldsymbol{\mathcal{Q}}}_f^{(n+1)} = AP^{(n)} \left[2I - AP^{(n)} \right] \boldsymbol{f}$$

From the proof of Part II:

$$\widetilde{\mathcal{Q}}_f^{(n+1)} = \mathbf{\Phi}^T \sum_{k=0}^{n} (I - A)^k \left[2I - A \sum_{k=0}^{n} (I - A)^k \right] \boldsymbol{f}$$

Therefore, with $P^{(n)} = \sum_{k=0}^{n} (I - A)^k$, evaluation on the data sites yields

$$\widetilde{\boldsymbol{\mathcal{Q}}}_f^{(n+1)} = AP^{(n)} \left[2I - AP^{(n)} \right] \boldsymbol{f}$$

or

$$P^{(n+1)} = P^{(n)} \left[2I - AP^{(n)} \right]$$

Program (IAMLSAccel_sD.m)

```
 1  s = 2;  N = 289;  M = 500;  maxn = 50;
 2  global rbf;  rbf_definition;  D = 2*s;
 3  [dsites, N] = CreatePoints(N,s,'h');
 4  ctrs = dsites;
 5  epoints = CreatePoints(M,s,'r');
 6  rhs = testfunctionsD(dsites);
 7  h = 1/(nthroot(N,s)-1); ep = 1/(sqrt(D)*h);
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9  IM = rbf(ep,DM_data)/(sqrt(pi*D)^s);
10  DM_eval = DistanceMatrix(epoints,ctrs);
11  EM = rbf(ep,DM_eval)/(sqrt(pi*D)^s);
12  P = eye(N);  AP = IM*P;
13  for n=1:maxn
14     P = P*(2*eye(N)-AP);
15     AP = IM*P;
16     Pf = EM*P*rhs;
17     maxerr(n) = norm(Pf-exact,inf);
18     rms_err(n) = norm(Pf-exact)/sqrt(M);
19  end
```

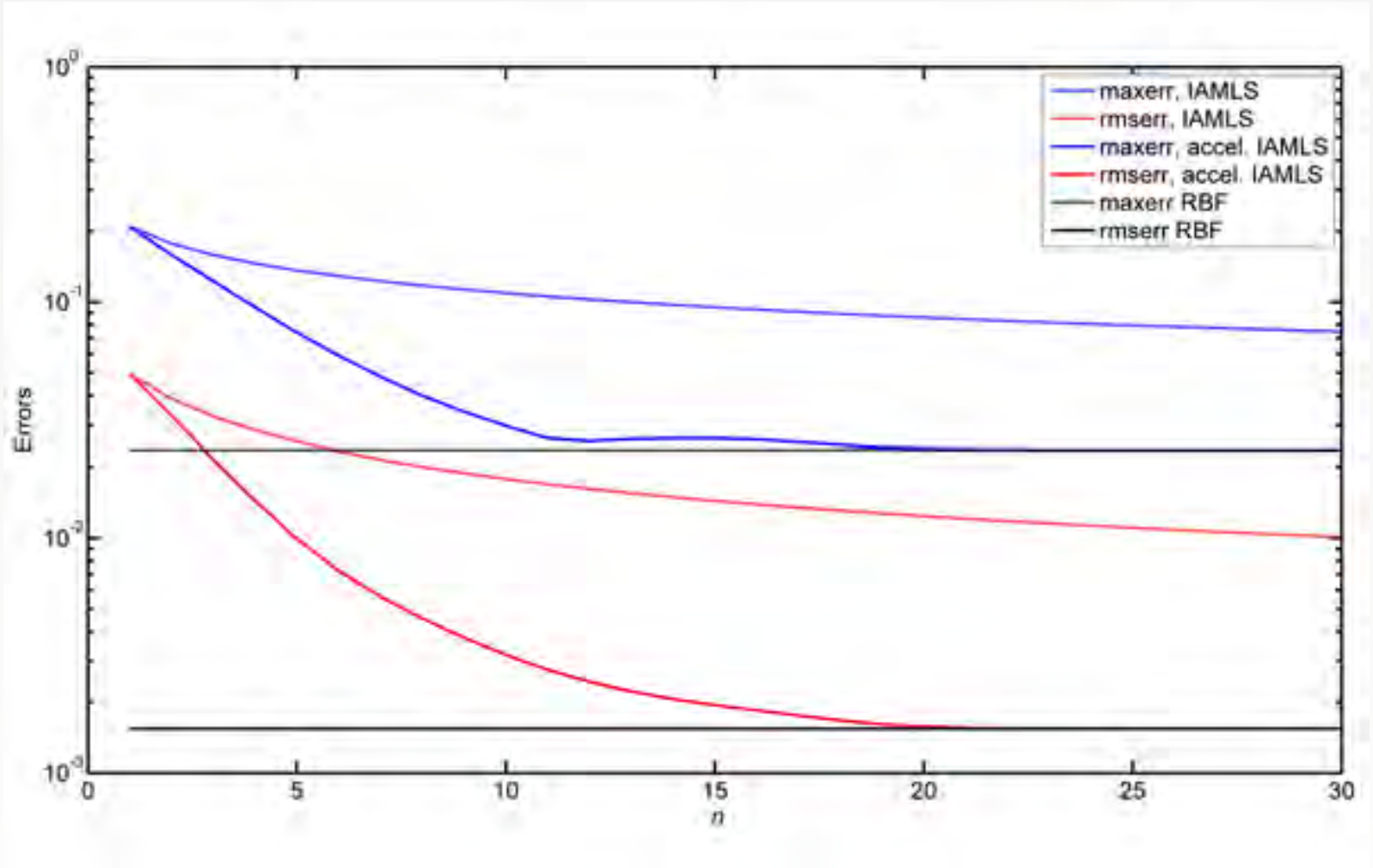


Figure: Errors after n iterations for 1089 Halton points (Gaussians with $\varepsilon = 16$). n accelerated iterations correspond to $2^n - 1$ iterations without acceleration.

- A few iterations of accelerated iterated AMLS can be considered as an efficient and numerically stable alternative to the RBF interpolation approach.
- While the initial iterate of the algorithm is an AMLS approximation designed for uniformly spaced data, we can see how the algorithm generates an equivalently nice solution even when the data sites are irregularly distributed.
- Convergence results for approximate approximation can be transferred to the limiting RBF interpolation. This explains saturation of stationary RBF interpolation.
- Applications of iterated AMLS to
  - preconditioning (next lecture)
  - smoothing of noisy data (lecture 5)

References I


Buhmann, M. D. (2003).
Radial Basis Functions: Theory and Implementations.
Cambridge University Press.


Fasshauer, G. E. (2007).
Meshfree Approximation Methods with MATLAB.
World Scientific Publishers.


Higham, D. J. and Higham, N. J. (2005).
MATLAB *Guide*.
SIAM (2nd ed.), Philadelphia.


Maz'ya, V. and Schmidt, G. (2007).
Approximate Approximations.
Mathematical Surveys and Monographs, vol. 141, Americal Mathematical Society (Providence, RI).


Wendland, H. (2005).
Scattered Data Approximation.
Cambridge University Press.

References II


Allasia, G. and Giolito, P. (1997).
Fast evaluation of cardinal radial basis interpolants.
in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 1–8.


Bos, L. P. and Šalkauskas, K. (1989).
Moving least-squares are Backus-Gilbert optimal.
J. Approx. Theory **59**, pp. 267–275.


Farwig, R. (1986).
Multivariate interpolation of arbitrarily spaced data by moving least squares methods.
J. Comput. Appl. Math. **16**, pp. 79–93.


Farwig, R. (1987).
Multivariate interpolation of scattered data by moving least squares methods.
in *Algorithms for Approximation*, Oxford Univ. Press (New York), pp. 193–211, 1987.

References III


Farwig, R. (1991).
Rate of convergence of moving least squares interpolation methods: the univariate case.
in *Progress in Approximation Theory*, Academic Press (Boston, MA), pp. 313–327.


Fasshauer, G. E. and Zhang, J. G. (2007).
Iterated approximate moving least squares approximation.
in *Advances in Meshfree Techniques*, V. M. A. Leitao, C. Alves and C. A. Duarte (eds.), Springer, pp. 221–240.


Fornberg, B., Larsson, E. and Wright, G. (2004).
A new class of oscillatory radial basis functions.
Comput. Math. Appl. **51** 8, pp. 1209–1222.


Levin, D. (1998).
The approximation power of moving least-squares.
Math. Comp. **67**, pp. 1517–1531.

References IV

Wendland, H. (2001).
Local polynomial reproduction and moving least squares approximation.
IMA J. Numer. Anal. **21** 1, pp. 285–300.

Zhang, J. G. (2007).
Iterated Approximate Moving Least-Squares: Theory and Applications.
Ph.D. Dissertation, Illinois Institute of Technology.

MATLAB Central File Exchange.
available online at
http://www.mathworks.com/matlabcentral/fileexchange/.