Meshfree Approximation with MATLAB

Lecture IV: RBF Collocation and Polynomial Pseudospectral Methods

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Dolomites Research Week on Approximation
September 8–11, 2008

Outline

Linear elliptic PDE with boundary conditions

$$
\begin{aligned}
\mathcal{L}u &= f \quad \text{in } \Omega \\
u &= g \quad \text{on } \Gamma = \partial\Omega
\end{aligned}
$$

Linear elliptic PDE with boundary conditions

$$\begin{aligned} \mathcal{L}u &= f \quad \text{in } \Omega \\ u &= g \quad \text{on } \Gamma = \partial\Omega \end{aligned}$$

Time-dependent PDE with initial and boundary conditions

$$\begin{aligned} u_t(\boldsymbol{x},t) + \mathcal{L}u(\boldsymbol{x},t) &= f(\boldsymbol{x},t), \quad \boldsymbol{x} \in \Omega \cup \Gamma, \; t \geq 0 \\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \\ u(\boldsymbol{x},t) &= g(t), \quad \boldsymbol{x} \in \Gamma \end{aligned}$$

According to [Kansa (1986)] we consider an elliptic PDE and start with

$$u(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}$$

According to [Kansa (1986)] we consider an elliptic PDE and start with

$$u(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}$$

Coefficients $\boldsymbol{\lambda}$ determined by solving

$$\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} \boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix},$$

with (rectangular) *collocation* matrices

$$\begin{aligned} \widetilde{A}_{\mathcal{L},ij} &= \mathcal{L}\Phi_j(\boldsymbol{x}_i) = \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_i}, \\ & \quad i = 1, \ldots, N - N_B,\ j = 1, \ldots, N, \\ \widetilde{A}_{ij} &= \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \\ & \quad i = N - N_B + 1, \ldots, N,\ j = 1, \ldots, N. \end{aligned}$$

If $\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}$ invertible (open problem since 1986), then approximate solution (at any point x) found by inserting $\boldsymbol{\lambda}$ in basis expansion, i.e.,

$$u(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}$$

If $\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}$ invertible (open problem since 1986), then approximate solution (at any point x) found by inserting $\boldsymbol{\lambda}$ in basis expansion, i.e.,

$$u(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}$$

Solution at collocation points

$$\boldsymbol{u} = A \underbrace{\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}}_{=\boldsymbol{\lambda}}, \qquad A_{ij} = \Phi_j(\boldsymbol{x}_i)$$

Note

$$\boldsymbol{u} = A \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix} \iff \underbrace{\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}}_{=L_{\Gamma}} \boldsymbol{u} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

Note

$$\boldsymbol{u} = A \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix} \iff \underbrace{\begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}}_{=L_\Gamma} \boldsymbol{u} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

with

$$L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} A_I^{-1} & \widetilde{A}_{\mathcal{L}} A_B^{-1} \\ \widetilde{A} A_I^{-1} & \widetilde{A} A_B^{-1} \end{bmatrix} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix},$$

$\longrightarrow$ RBF-PS method (see later).

Example (2D Laplace Equation)

$$u_{xx} + u_{yy} = 0, \quad x, y \in (-1, 1)^2$$

Boundary conditions:

$$u(x, y) = \begin{cases} \sin^4(\pi x), & y = 1 \text{ and } -1 < x < 0, \\ \frac{1}{5}\sin(3\pi y), & x = 1, \\ 0, & \text{otherwise.} \end{cases}$$

See [Trefethen (2000)]

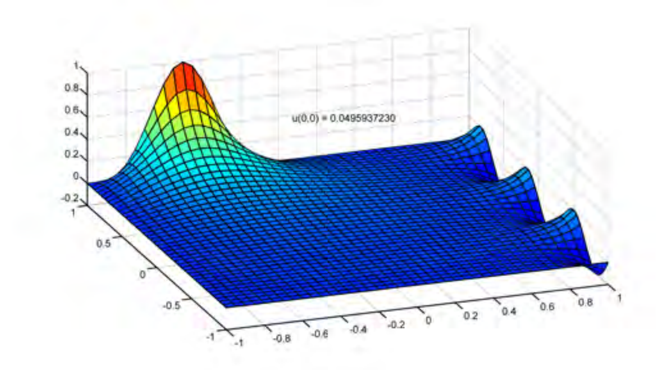

Figure: Gaussian-RBF ($\varepsilon = 2.75$), $N = 24 \times 24$

Program (KansaLaplaceMixedBCTref_2D.m)

```
global rbf Lrbf;  rbf_definition;  epsilon = 3;
N = 289;  M = 1681;
Lu = @(x,y) zeros(size(x));
[collpts, N] = CreatePoints(N, 2, 'u');  collpts = 2*collpts-1;
indx = find(abs(collpts(:,1))==1 | abs(collpts(:,2))==1);
bdypts = collpts(indx,:);
intpts = collpts(setdiff([1:N],indx),:);
ctrs = [intpts; bdypts];
evalpts = CreatePoints(M,2,'u');  evalpts = 2*evalpts-1;
DM_eval = DistanceMatrix(evalpts,ctrs);
EM = rbf(epsilon,DM_eval);
DM_int = DistanceMatrix(intpts,ctrs);
DM_bdy = DistanceMatrix(bdypts,ctrs);
LCM = Lrbf(epsilon,DM_int);
BCM = rbf(epsilon,DM_bdy);
CM = [LCM; BCM];
rhs = zeros(N,1);  NI = size(intpts,1);
indx = find(bdypts(:,1)==1 | (bdypts(:,1)<0) & (bdypts(:,2)==1));
rhs(NI+indx) = (bdypts(indx,1)==1)*0.2.*sin(3*pi*bdypts(indx,2)) + ...
    (bdypts(indx,1)<0).*(bdypts(indx,2)==1).*sin(pi*bdypts(indx,1)).^4;
Pf = EM * (CM\rhs);
disp(sprintf('u(0,0) = %16.12f',Pf(841)))
```

We just showed that we always have (even if the Kansa matrix is not invertible)

$$L_{\Gamma} = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

We just showed that we always have (even if the Kansa matrix is not invertible)

$$L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

It is known that

- L_Γ is invertible for polynomial basis (1D)
- In a certain limiting case RBF interpolant yields polynomial interpolant

We just showed that we always have (even if the Kansa matrix is not invertible)

$$L_\Gamma = \begin{bmatrix} \widetilde{A}_\mathcal{L} \\ \widetilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

It is known that

- L_Γ is invertible for polynomial basis (1D)
- In a certain limiting case RBF interpolant yields polynomial interpolant

$\Longrightarrow$ Kansa's collocation matrix is invertible in the limiting case

We just showed that we always have (even if the Kansa matrix is not invertible)

$$L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

It is known that

- L_Γ is invertible for polynomial basis (1D)
- In a certain limiting case RBF interpolant yields polynomial interpolant

$\Longrightarrow$ Kansa's collocation matrix is invertible in the limiting case

Other recent work on a well-defined approach to Kansa's method, e.g., in [Schaback (2007)]

Consider time-dependent PDE

$$u_t + \mathcal{L}u = f$$

and discretize the time, e.g.,

$$u_t(\boldsymbol{x}, t_k) \approx \frac{u(\boldsymbol{x}, t_k) - u(\boldsymbol{x}, t_{k-1})}{\Delta t} = \frac{u_k(\boldsymbol{x}) - u_{k-1}(\boldsymbol{x})}{\Delta t}$$

Consider time-dependent PDE

$$u_t + \mathcal{L}u = f$$

and discretize the time, e.g.,

$$u_t(\boldsymbol{x}, t_k) \approx \frac{u(\boldsymbol{x}, t_k) - u(\boldsymbol{x}, t_{k-1})}{\Delta t} = \frac{u_k(\boldsymbol{x}) - u_{k-1}(\boldsymbol{x})}{\Delta t}$$

Then

$$u_k(\boldsymbol{x}) \approx u_{k-1}(\boldsymbol{x}) + \Delta t \left[f(\boldsymbol{x}) - \mathcal{L}u_{k-1}(\boldsymbol{x}) \right]$$

Start with

$$u(\boldsymbol{x}, t) = \sum_{j=1}^{N} \lambda_j(t) \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x}) \boldsymbol{\lambda}(t)$$

Start with

$$u(\boldsymbol{x}, t) = \sum_{j=1}^{N} \lambda_j(t)\Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}(t)$$

Then after time discretization

$$\sum_{j=1}^{N} \lambda_j^{(k)}\Phi_j(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j^{(k-1)}\Phi_j(\boldsymbol{x}) + \Delta t \left[f(\boldsymbol{x}) - \sum_{j=1}^{N} \lambda_j^{(k-1)}\mathcal{L}\Phi_j(\boldsymbol{x}) \right]$$

Start with

$$u(\boldsymbol{x},t) = \sum_{j=1}^{N} \lambda_j(t)\Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}(t)$$

Then after time discretization

$$\sum_{j=1}^{N} \lambda_j^{(k)}\Phi_j(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j^{(k-1)}\Phi_j(\boldsymbol{x}) + \Delta t \left[f(\boldsymbol{x}) - \sum_{j=1}^{N} \lambda_j^{(k-1)} \mathcal{L}\Phi_j(\boldsymbol{x}) \right]$$

or

$$\boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}^{(k)} = \left[\boldsymbol{\Phi}^T(\boldsymbol{x}) - \Delta t \mathcal{L}\boldsymbol{\Phi}^T(\boldsymbol{x})\right]\boldsymbol{\lambda}^{(k-1)} + \Delta t f(\boldsymbol{x})$$

Collocation on $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ yields

$$\boxed{A\boldsymbol{\lambda}^{(k)} = [A - \Delta t A_{\mathcal{L}}]\,\boldsymbol{\lambda}^{(k-1)} + \Delta t \boldsymbol{f}}$$

Collocation on $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ yields

$$\boxed{A\boldsymbol{\lambda}^{(k)} = \left[A - \Delta t A_{\mathcal{L}}\right] \boldsymbol{\lambda}^{(k-1)} + \Delta t \boldsymbol{f}}$$

Solve for $\boldsymbol{\lambda}^{(k)}$, then (for any $\boldsymbol{x}$)

$$u^{(k)}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j^{(k)} \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}^{(k)}$$

Collocation on $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ yields

$$\boxed{A\boldsymbol{\lambda}^{(k)} = [A - \Delta t A_{\mathcal{L}}]\, \boldsymbol{\lambda}^{(k-1)} + \Delta t \boldsymbol{f}}$$

Solve for $\boldsymbol{\lambda}^{(k)}$, then (for any $\boldsymbol{x}$)

$$u^{(k)}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j^{(k)} \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}^{(k)}$$

Solution at collocation points

$$\boldsymbol{u}^{(k)} = A\boldsymbol{\lambda}^{(k)}, \qquad A_{ij} = \Phi_j(\boldsymbol{x}_i)$$

Collocation on $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ yields

$$\boxed{A\boldsymbol{\lambda}^{(k)} = \left[A - \Delta t A_{\mathcal{L}}\right] \boldsymbol{\lambda}^{(k-1)} + \Delta t \boldsymbol{f}}$$

Solve for $\boldsymbol{\lambda}^{(k)}$, then (for any $\boldsymbol{x}$)

$$u^{(k)}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j^{(k)} \Phi_j(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\lambda}^{(k)}$$

Solution at collocation points

$$\boldsymbol{u}^{(k)} = A\boldsymbol{\lambda}^{(k)}, \qquad A_{ij} = \Phi_j(\boldsymbol{x}_i)$$

Much too complicated!

PS methods are known as highly accurate solvers for PDEs

Basic idea (in 1D)

$$u(x) = \sum_{j=1}^{N} \lambda_j \Phi_j(x), \quad x \in [a, b]$$

with (smooth and global) basis functions Φ_j, $j = 1, \ldots, N$

Here u is the unknown (approximate) solution of the PDE

For PDEs we need to be able to represent values of derivatives of u.

For PS methods values at grid points suffice.

Key idea: find differentiation matrix D such that

$$\boldsymbol{u}' = D\boldsymbol{u}$$

where $\boldsymbol{u} = [u(x_1), \ldots, u(x_N)]^T$

For PDEs we need to be able to represent values of derivatives of u.

For PS methods values at grid points suffice.

Key idea: find differentiation matrix D such that

$$\boldsymbol{u}' = D\boldsymbol{u}$$

where $\boldsymbol{u} = [u(x_1), \ldots, u(x_N)]^T$

Example

Chebyshev polynomials on Chebyshev points. In this case explicit formulas for entries of D are known.

$$u(x) = \sum_{j=1}^{N} \lambda_j \Phi_j(x) \implies u'(x) = \sum_{j=1}^{N} \lambda_j \frac{d}{dx} \Phi_j(x).$$

$$u(x) = \sum_{j=1}^{N} \lambda_j \Phi_j(x) \implies u'(x) = \sum_{j=1}^{N} \lambda_j \frac{d}{dx} \Phi_j(x).$$

Evaluate at grid points:

$$\boldsymbol{u} = A\boldsymbol{\lambda} \quad \text{with } A_{ij} = \Phi_j(x_i)$$

and

$$\boldsymbol{u}' = A_x\boldsymbol{\lambda} \quad \text{with } A_{x,ij} = \frac{d}{dx}\Phi_j(x_i).$$

$$u(x) = \sum_{j=1}^{N} \lambda_j \Phi_j(x) \implies u'(x) = \sum_{j=1}^{N} \lambda_j \frac{d}{dx} \Phi_j(x).$$

Evaluate at grid points:

$$\boldsymbol{u} = A\boldsymbol{\lambda} \quad \text{with } A_{ij} = \Phi_j(x_i)$$

and

$$\boldsymbol{u}' = A_x\boldsymbol{\lambda} \quad \text{with } A_{x,ij} = \frac{d}{dx}\Phi_j(x_i).$$

Therefore

$$\boldsymbol{u}' = A_x A^{-1}\boldsymbol{u} \quad \implies \quad D = A_x A^{-1}.$$

Want to use radial basis functions

$$\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|), \quad \boldsymbol{x} \in \mathbb{R}^s$$

and a general linear differential operator $\mathcal{L}$ with constant coefficients.

Want to use radial basis functions

$$\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|), \quad \boldsymbol{x} \in \mathbb{R}^s$$

and a general linear differential operator $\mathcal{L}$ with constant coefficients.

Discretized differential operator (differentiation matrix):

$$L = A_{\mathcal{L}} A^{-1}$$

with $A_{ij} = \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$
and $A_{\mathcal{L},ij} = \mathcal{L}\Phi_j(\boldsymbol{x}_i) = \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_i}$.

Want to use radial basis functions

$$\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|), \quad \boldsymbol{x} \in \mathbb{R}^s$$

and a general linear differential operator $\mathcal{L}$ with constant coefficients.

Discretized differential operator (differentiation matrix):

$$L = A_{\mathcal{L}} A^{-1}$$

with $A_{ij} = \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$
and $A_{\mathcal{L},ij} = \mathcal{L}\Phi_j(\boldsymbol{x}_i) = \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_i}$.

A is (non-singular) RBF interpolation matrix

How to proceed for a time-independent PDE

$$\mathcal{L}u = f$$

- Discretize space

$$\mathcal{L}\boldsymbol{u} \approx L\boldsymbol{u}$$

- Then (at grid points)

$$\boldsymbol{u} = L^{-1}\boldsymbol{f}$$

How to proceed for a time-independent PDE

$$\mathcal{L}u = f$$

- Discretize space

$$\mathcal{L}\boldsymbol{u} \approx L\boldsymbol{u}$$

- Then (at grid points)

$$\boldsymbol{u} = L^{-1}\boldsymbol{f}$$

Challenge

Need to ensure invertibility of L

Want to solve $\mathcal{L}u = f$ without BCs.

Discretize:

$$L\boldsymbol{u} = \boldsymbol{f} \quad \Longrightarrow \quad \boldsymbol{u} = L^{-1}\boldsymbol{f} = A(A_{\mathcal{L}})^{-1}\boldsymbol{f}$$

Invertibility of L (and therefore $A_{\mathcal{L}}$) required.

Want to solve $\mathcal{L}u = f$ without BCs.

Discretize:

$$L\boldsymbol{u} = \boldsymbol{f} \quad \Longrightarrow \quad \boldsymbol{u} = L^{-1}\boldsymbol{f} = A\left(A_{\mathcal{L}}\right)^{-1}\boldsymbol{f}$$

Invertibility of L (and therefore $A_{\mathcal{L}}$) required.

- Chebyshev PS: L is singular

Want to solve $\mathcal{L}u = f$ without BCs.

Discretize:

$$L\boldsymbol{u} = \boldsymbol{f} \quad \Longrightarrow \quad \boldsymbol{u} = L^{-1}\boldsymbol{f} = A\left(A_{\mathcal{L}}\right)^{-1}\boldsymbol{f}$$

Invertibility of L (and therefore $A_{\mathcal{L}}$) required.

- Chebyshev PS: L is singular
- RBF-PS: L is non-singular since $A_{\mathcal{L}}$ invertible for positive definite RBFs and elliptic $\mathcal{L}$.

Want to solve $\mathcal{L}u = f$ without BCs.

Discretize:

$$L\boldsymbol{u} = \boldsymbol{f} \quad \Longrightarrow \quad \boldsymbol{u} = L^{-1}\boldsymbol{f} = A(A_{\mathcal{L}})^{-1}\boldsymbol{f}$$

Invertibility of L (and therefore $A_{\mathcal{L}}$) required.

- Chebyshev PS: L is singular
- RBF-PS: L is non-singular since $A_{\mathcal{L}}$ invertible for positive definite RBFs and elliptic $\mathcal{L}$.

Remark

RBFs "too good to be true". Built-in regularization due to variational framework.

Consider linear elliptic PDE

$$\mathcal{L}u = f \quad \text{in } \Omega$$

with boundary condition

$$u = g \quad \text{on } \Gamma = \partial\Omega$$

and assume basis functions do not satisfy BCs.

Consider linear elliptic PDE

$$\mathcal{L}u = f \quad \text{in } \Omega$$

with boundary condition

$$u = g \quad \text{on } \Gamma = \partial\Omega$$

and assume basis functions do not satisfy BCs.

- Construct differentiation matrix L based on all grid points $\boldsymbol{x}_i$.
- Then replace diagonal entries corresponding to boundary points with ones and the remainder of those rows with zeros.

Reorder rows and columns to obtain

$$L_\Gamma = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix},$$

Here

- M is $N_I \times N_I$ (interior points)
- I is $N_B \times N_B$ (boundary points)
- N_B number of grid points on boundary Γ
- $N_I = N - N_B$ number of grid points in the interior Ω

Reorder rows and columns to obtain

$$L_\Gamma = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix},$$

Here

- M is $N_I \times N_I$ (interior points)
- I is $N_B \times N_B$ (boundary points)
- N_B number of grid points on boundary Γ
- $N_I = N - N_B$ number of grid points in the interior Ω

Then

$$\boldsymbol{u} = L_\Gamma^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

Using $\boldsymbol{u} = [\boldsymbol{u}_\Omega, \boldsymbol{u}_\Gamma]^T$ and substituting $\boldsymbol{u}_\Gamma = \boldsymbol{g}$ back in we get

$$\boldsymbol{u}_\Omega = M^{-1}(\boldsymbol{f} - P\boldsymbol{g}),$$

or, for homogeneous boundary conditions,

$$\boldsymbol{u}_\Omega = M^{-1}\boldsymbol{f}.$$

Using $\boldsymbol{u} = [\boldsymbol{u}_\Omega, \boldsymbol{u}_\Gamma]^T$ and substituting $\boldsymbol{u}_\Gamma = \boldsymbol{g}$ back in we get

$$\boldsymbol{u}_\Omega = M^{-1}(\boldsymbol{f} - P\boldsymbol{g}),$$

or, for homogeneous boundary conditions,

$$\boldsymbol{u}_\Omega = M^{-1}\boldsymbol{f}.$$

Remark

For standard PS methods the block M is invertible. Its spectrum is well studied for many different $\mathcal{L}$ and BCs.

As in [F. (1997)] we start with

$$u(\boldsymbol{x}) = \sum_{j=1}^{N_I} \lambda_j \mathcal{L}_j^* \Phi(\boldsymbol{x}) + \sum_{j=N_I+1}^{N} \lambda_j \Phi_j(\boldsymbol{x}).$$

Since $\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$, $\mathcal{L}_j^*$ denotes application of $\mathcal{L}$ to φ viewed as a function of the second variable followed by evaluation at $\boldsymbol{x}_j$.

As in [F. (1997)] we start with

$$u(\boldsymbol{x}) = \sum_{j=1}^{N_I} \lambda_j \mathcal{L}_j^* \Phi(\boldsymbol{x}) + \sum_{j=N_I+1}^{N} \lambda_j \Phi_j(\boldsymbol{x}).$$

Since $\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$, $\mathcal{L}_j^*$ denotes application of $\mathcal{L}$ to φ viewed as a function of the second variable followed by evaluation at $\boldsymbol{x}_j$.

Then $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_\Omega, \boldsymbol{\lambda}_\Gamma]^T$ obtained from

$$\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_\Omega \\ \boldsymbol{\lambda}_\Gamma \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}.$$

The collocation matrix

$$\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix}$$

consists of

- square blocks

$$\begin{aligned} \hat{A}_{\mathcal{L}\mathcal{L}^*,ij} &= \left[\mathcal{L}\left[\mathcal{L}^*\varphi(\|\boldsymbol{x}-\boldsymbol{\xi}\|)\right]_{\boldsymbol{\xi}=\boldsymbol{x}_j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i,j=1,\ldots,N_I \\ \hat{A}_{ij} &= \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|), \quad i,j=N_I+1,\ldots,N \end{aligned}$$

- rectangular blocks

$$\begin{aligned} \hat{A}_{\mathcal{L},ij} &= \left[\mathcal{L}\varphi(\|\boldsymbol{x}-\boldsymbol{x}_j\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i=1,\ldots,N_I,\ j=N_I+1,\ldots,N \\ \hat{A}_{\mathcal{L}^*,ij} &= \left[\mathcal{L}^*\varphi(\|\boldsymbol{x}_i-\boldsymbol{x}\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_j}, \quad i=N_I+1,\ldots,N,\ j=1,\ldots,N_I \end{aligned}$$

2D Laplace Equation from [Trefethen (2000)]

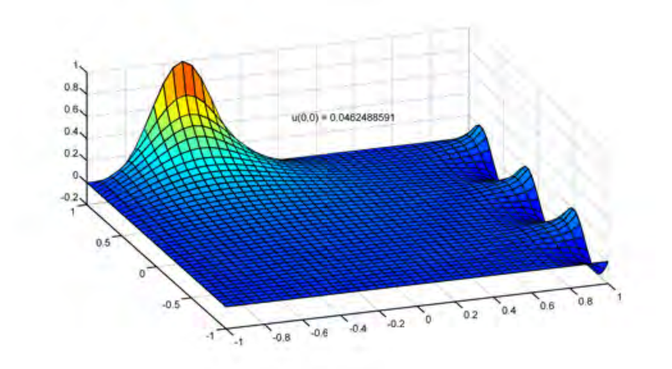

Figure: Gaussian-RBF ($\varepsilon = 6$), $N = 17 \times 17$ (+ 64 boundary points)

Program (HermiteLaplaceMixedBCTref_2D.m)

```
global rbf Lrbf L2rbf;  rbf_definition;  epsilon = 3;
N = 289;  M = 1681;
Lu = @(x,y) zeros(size(x));
[datasites, N] = CreatePoints(N, 2, 'u');  intdata = 2*datasites-1;
sg = sqrt(N); bdylin = linspace(-1,1,sg)'; bdy1 = ones(sg-1,1);
bdydata = [bdylin(1:end-1) -bdy1; bdy1 bdylin(1:end-1);...
           flipud(bdylin(2:end)) bdy1; -bdy1 flipud(bdylin(2:end))];
h = 2/(sg-1); bdylin = (-1+h:h:1-h)'; bdy0 = repmat(-1-h,sg-2,1); bdy1 = repmat(1+h,sg-2,1);
bdycenters = [-1-h -1-h; bdylin bdy0; 1+h -1-h; bdy1 bdylin;...
              1+h 1+h; flipud(bdylin) bdy1; -1-h 1+h; bdy0 flipud(bdylin)];
centers = [intdata; bdycenters];
evalpoints = CreatePoints(M, 2, 'u');  evalpoints = 2*evalpoints-1;
DM_inteval = DistanceMatrix(evalpoints,intdata);
DM_bdyeval = DistanceMatrix(evalpoints,bdycenters);
LEM = Lrbf(epsilon,DM_inteval);
BEM = rbf(epsilon,DM_bdyeval);
EM = [LEM BEM];
DM_IIdata = DistanceMatrix(intdata,intdata);
DM_IBdata = DistanceMatrix(intdata,bdycenters);
DM_BIdata = DistanceMatrix(bdydata,intdata);
DM_BBdata = DistanceMatrix(bdydata,bdycenters);
LLCM = L2rbf(epsilon,DM_IIdata);
LBCM = Lrbf(epsilon,DM_IBdata);
BLCM = Lrbf(epsilon,DM_BIdata);
BBCM = rbf(epsilon,DM_BBdata);
CM = [LLCM LBCM; BLCM BBCM];
rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sg-1,1); 0.2*sin(3*pi*bdydata(sg:2*sg-2,2)); ...
        zeros((sg-1)/2,1); sin(pi*bdydata((5*sg-3)/2:3*sg-3,1)).^4; zeros(sg-1,1)];
Pf = EM * (CM\rhs);
disp(sprintf('u(0,0) = %16.12f',Pf(841)))
```

It is well known that the symmetric collocation matrix

$$\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix}$$

is invertible.

It is well known that the symmetric collocation matrix

$$\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix}$$

is invertible.

Therefore, the solution at any point is obtained by inserting $\boldsymbol{\lambda}$ into basis expansion, or at grid points

$$\boldsymbol{u} = \underbrace{\begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix}}_{=K^T} \underbrace{\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}}_{=\boldsymbol{\lambda}}$$

with evaluation matrix K^T.

Entries of K^T given by

$$
\begin{aligned}
A_{\mathcal{L}^*,ij} &= \left[\mathcal{L}^*\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_j}, \quad i = 1,\ldots,N,\ j = 1,\ldots,N_I \\
\widetilde{A}_{ij}^T &= \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \quad i = 1,\ldots,N,\ j = N_I + 1,\ldots,N
\end{aligned}
$$

Entries of K^T given by

$$\begin{aligned} A_{\mathcal{L}^*,ij} &= [\mathcal{L}^*\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}\|)]_{\boldsymbol{x}=\boldsymbol{x}_j}, \quad i = 1,\ldots,N, \ j = 1,\ldots,N_I \\ \widetilde{A}^T_{ij} &= \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \quad i = 1,\ldots,N, \ j = N_I + 1,\ldots,N \end{aligned}$$

Symmetry of RBFs implies $A_{\mathcal{L}^*} = \widetilde{A}^T_{\mathcal{L}}$, and therefore

$$K^T = \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix} = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix}^T$$

$\longrightarrow$ transpose of Kansa's matrix

Start with “symmetric basis” expansion

$$u(\boldsymbol{x}) = \sum_{j=1}^{N_I} \lambda_j \mathcal{L}_j^* \Phi(\boldsymbol{x}) + \sum_{j=N_I+1}^{N} \lambda_j \Phi_j(\boldsymbol{x}).$$

At the grid points in matrix notation we have

$$\boldsymbol{u} = \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_\Omega \\ \boldsymbol{\lambda}_\Gamma \end{bmatrix}$$

or

$$\boldsymbol{\lambda} = \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix}^{-1} \boldsymbol{u}.$$

Apply $\mathcal{L}$ to basis expansion and restrict to grid

$$\mathcal{L}\boldsymbol{u} = \begin{bmatrix} A_{\mathcal{L}\mathcal{L}^*} & A_{\mathcal{L}} \end{bmatrix} \boldsymbol{\lambda}$$

with

$$\begin{aligned} A_{\mathcal{L}\mathcal{L}^*,ij} &= \left[\mathcal{L}\left[\mathcal{L}^*\varphi(\|\boldsymbol{x}-\boldsymbol{\xi}\|)\right]_{\boldsymbol{\xi}=\boldsymbol{x}_j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i=1,\ldots,N,\ j=1,\ldots,N_I \\ A_{\mathcal{L},ij} &= \left[\mathcal{L}\varphi(\|\boldsymbol{x}-\boldsymbol{x}_j\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i=1,\ldots,N,\ j=N_I+1,\ldots,N \end{aligned}$$

Apply $\mathcal{L}$ to basis expansion and restrict to grid

$$\mathcal{L}\boldsymbol{u} = \begin{bmatrix} A_{\mathcal{L}\mathcal{L}^*} & A_{\mathcal{L}} \end{bmatrix} \boldsymbol{\lambda}$$

with

$$\begin{aligned} A_{\mathcal{L}\mathcal{L}^*,ij} &= \left[\mathcal{L}\left[\mathcal{L}^*\varphi(\|\boldsymbol{x}-\boldsymbol{\xi}\|)\right]_{\boldsymbol{\xi}=\boldsymbol{x}_j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i=1,\ldots,N,\ j=1,\ldots,N_I \\ A_{\mathcal{L},ij} &= \left[\mathcal{L}\varphi(\|\boldsymbol{x}-\boldsymbol{x}_j\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i=1,\ldots,N,\ j=N_I+1,\ldots,N \end{aligned}$$

Replace $\boldsymbol{\lambda}$ and get

$$\hat{L}\boldsymbol{u} = \begin{bmatrix} A_{\mathcal{L}\mathcal{L}^*} & A_{\mathcal{L}} \end{bmatrix} \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix}^{-1} \boldsymbol{u}.$$

Remark

Note that $\hat{L}$ differs from

$$\hat{L}_\Gamma = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix} \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix}^{-1}$$

since the BCs are not yet enforced.

Non-symmetric (Kansa):

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}$

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ
- $\Longrightarrow$ OK for time-dependent PDEs

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ
- $\Longrightarrow$ OK for time-dependent PDEs

Symmetric:

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_{\mathcal{L}} \\ \widetilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ
- $\Longrightarrow$ OK for time-dependent PDEs

Symmetric:

- Can ensure general solution of $\mathcal{L}u = f$

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \widetilde{A}_\mathcal{L} \\ \widetilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ
- $\Longrightarrow$ OK for time-dependent PDEs

Symmetric:

- Can ensure general solution of $\mathcal{L}u = f$
- Cannot in general formulate discrete differential operator

$$\hat{L}_\Gamma = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_\mathcal{L} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix} \begin{bmatrix} A_{\mathcal{L}^*} & \widetilde{A}^T \end{bmatrix}^{-1}$$

Non-symmetric (Kansa):

- Can formulate discrete differential operator $L_\Gamma = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1}$
- Cannot ensure general invertibility of L_Γ
- $\Longrightarrow$ OK for time-dependent PDEs

Symmetric:

- Can ensure general solution of $\mathcal{L}u = f$
- Cannot in general formulate discrete differential operator $\hat{L}_\Gamma = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^*} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^*} & \hat{A} \end{bmatrix} \begin{bmatrix} A_{\mathcal{L}^*} & \tilde{A}^T \end{bmatrix}^{-1}$
- $\Longrightarrow$ OK for time-independent PDEs

First-order Derivatives

The chain rule says

$$\frac{\partial}{\partial x_j}\varphi(\|\boldsymbol{x}\|) = \frac{x_j}{r}\frac{d}{dr}\varphi(r),$$

where x_j is a component of $\boldsymbol{x}$, and $r = \|\boldsymbol{x}\| = \sqrt{x_1^2 + \ldots + x_s^2}$.

First-order Derivatives

The chain rule says

$$\frac{\partial}{\partial x_j}\varphi(\|\boldsymbol{x}\|) = \frac{x_j}{r}\frac{d}{dr}\varphi(r),$$

where x_j is a component of $\boldsymbol{x}$, and $r = \|\boldsymbol{x}\| = \sqrt{x_1^2 + \ldots + x_s^2}$.

This implies we need to

- provide code for derivatives of φ, e.g., for the Gaussian

```
dxrbf = @(ep,r,dx) -2*dx*ep^2.*exp(-(ep*r).^2);
```

First-order Derivatives

The chain rule says

$$\frac{\partial}{\partial x_j}\varphi(\|\boldsymbol{x}\|) = \frac{x_j}{r}\frac{d}{dr}\varphi(r),$$

where x_j is a component of $\boldsymbol{x}$, and $r = \|\boldsymbol{x}\| = \sqrt{x_1^2 + \ldots + x_s^2}$.

This implies we need to

- provide code for derivatives of φ, e.g., for the Gaussian
```
dxrbf = @(ep,r,dx) -2*dx*ep^2.*exp(-(ep*r).^2);
```
- compute distances r (with `DistanceMatrix.m`),

First-order Derivatives

The chain rule says

$$\frac{\partial}{\partial x_j}\varphi(\|\boldsymbol{x}\|) = \frac{x_j}{r}\frac{d}{dr}\varphi(r),$$

where x_j is a component of $\boldsymbol{x}$, and $r = \|\boldsymbol{x}\| = \sqrt{x_1^2 + \ldots + x_s^2}$.
This implies we need to

- provide code for derivatives of φ, e.g., for the Gaussian
  `dxrbf = @(ep,r,dx) -2*dx*ep^2.*exp(-(ep*r).^2);`
- compute distances r (with `DistanceMatrix.m`),
- and compute differences in x.

Program (`DifferenceMatrix.m`)

```
function DM = DifferenceMatrix(dcoord,ccoord)
[dr,cc] = ndgrid(dcoord(:),ccoord(:));
DM = dr-cc;
```

Program (DRBF.m)

```
function [D,x] = DRBF(N,rbf,dxrbf,ep)
if N==0, D=0; x=1; return, end
x = cos(pi*(0:N)/N)';    % Chebyshev points
r = DistanceMatrix(x,x);
dx = DifferenceMatrix(x,x);
A = rbf(ep,r);
Ax = dxrbf(ep,r,dx);
D = Ax/A;
```

Remark

- *The differentiation matrix is given by* $D = A_x A^{-1}$*. In* MATLAB *we implement this as solution of* $DA = A_x$ *using* `mrdivide` *(/).*
- *Could add a version of LOOCV to find "optimal"* ε*.*

Example (1D Transport Equation)

Consider

$$\begin{aligned} u_t(x,t) + c u_x(x,t) &= 0, \quad x > -1, \; t > 0 \\ u(-1,t) &= 0 \\ u(x,0) &= f(x) \end{aligned}$$

with solution

$$u(x,t) = f(x - ct)$$

Use implicit Euler for time discretization

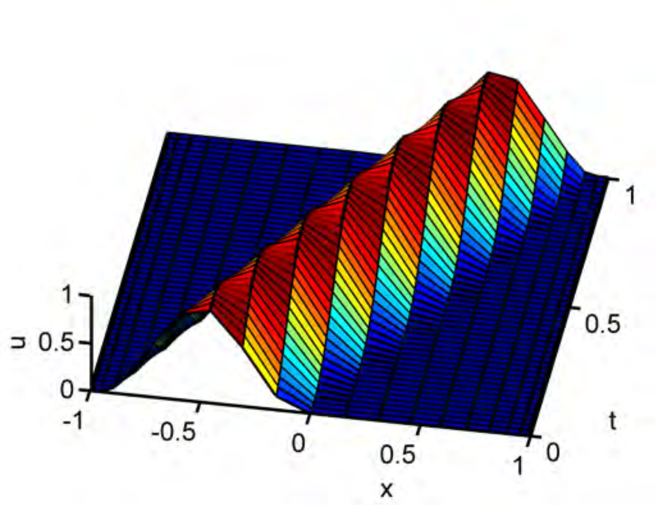

Figure: Gaussian RBFs with “optimal” $\varepsilon = 1.874528$, $\Delta t = 0.01$, collocation on 21 Chebyshev points

Example (Allen-Cahn)

$$u_t = \mu u_{xx} + u - u^3, \quad x \in (-1,1),\ t \geq 0,$$

μ coupling parameter (governs transition between stable solutions), here $\mu = 0.01$
Initial condition:

$$u(x,0) = 0.53x + 0.47 \sin\left(-\frac{3}{2}\pi x\right), \quad x \in [-1,1],$$

Boundary conditions:

$$u(-1,t) = -1 \text{ and } u(1,t) = \sin^2(t/5)$$

See [Trefethen (2000)]

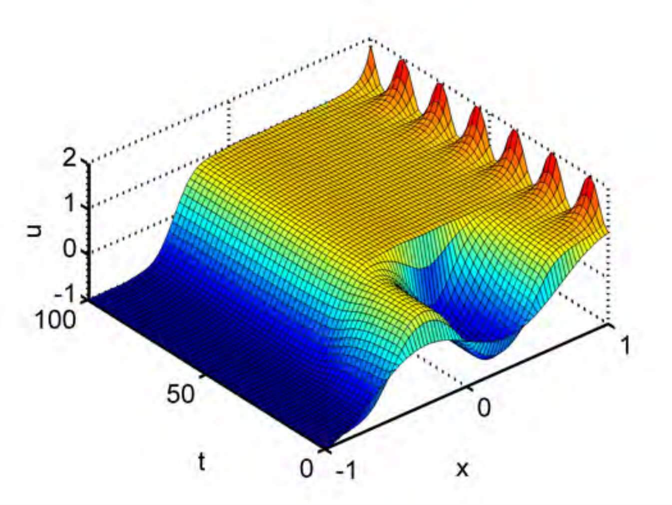

Figure: Matérn-RBF with “optimal” $\varepsilon = 0.351011$, collocated on 21 Chebyshev points

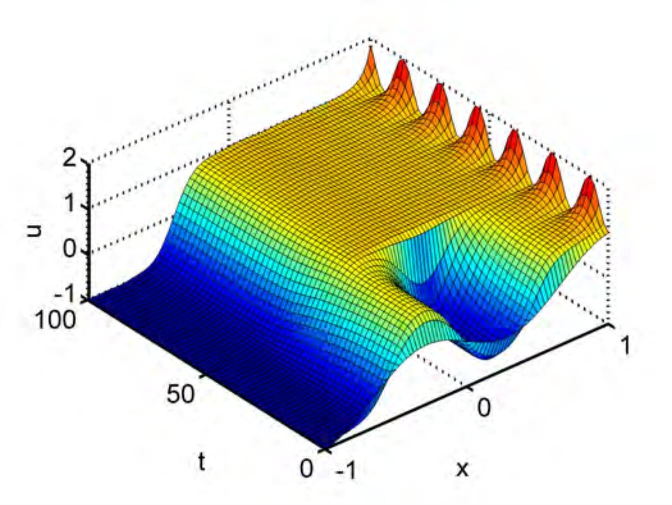

Figure: Chebyshev pseudospectral with 21 points

Example (2D Laplace Equation)

$$u_{xx} + u_{yy} = 0, \quad x, y \in (-1,1)^2$$

Boundary conditions:

$$u(x,y) = \begin{cases} \sin^4(\pi x), & y = 1 \text{ and } -1 < x < 0, \\ \frac{1}{5}\sin(3\pi y), & x = 1, \\ 0, & \text{otherwise.} \end{cases}$$

See [Trefethen (2000)]

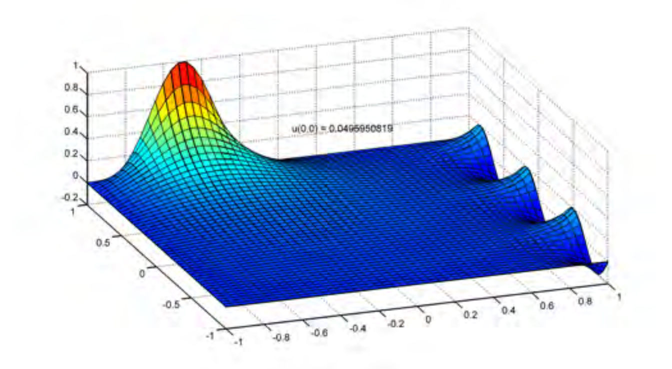

Figure: Matérn-RBF ($\varepsilon = 2.4$), $N = 24 \times 24$

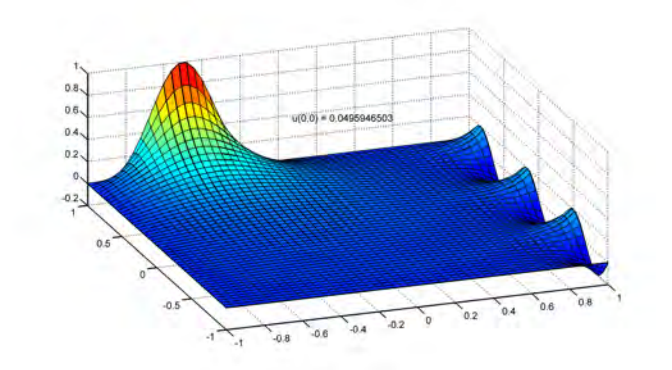

Figure: Chebyshev pseudospectral, $N = 24 \times 24$

Program (p36_2D.m)

```
rbf=@(e,r) exp(-e*r).*(15+15*e*r+6*(e*r).^2+(e*r).^3);
Lrbf=@(e,r)e^2*exp(-e*r).*((e*r).^3-(e*r).^2-6*e*r-6);
N = 24;   ep = 2.4;
[L,x,y] = LRBF(N,rbf,Lrbf,ep);
[xx,yy] = meshgrid(x,y);   xx = xx(:); yy = yy(:);
b = find(abs(xx)==1 | abs(yy)==1);      % boundary pts
L(b,:) = zeros(4*N,(N+1)^2); L(b,b) = eye(4*N);
rhs = zeros((N+1)^2,1);
rhs(b) = (yy(b)==1).*(xx(b)<0).*sin(pi*xx(b)).^4 + ...
         .2*(xx(b)==1).*sin(3*pi*yy(b));
u = L\rhs;
uu = reshape(u,N+1,N+1);   [xx,yy] = meshgrid(x,y);
[xxx,yyy] = meshgrid(-1:.04:1,-1:.04:1);
uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
surf(xxx,yyy,uuu), colormap('default');
axis([-1 1 -1 1 -.2 1]), view(-20,45)
text(0,.8,.4,sprintf('u(0,0) = %12.10f',...
                     uu(N/2+1,N/2+1)))
```

Remark

Note that for this type of elliptic problem we require inversion of the differentiation matrix.
As pointed out above, we use the non-symmetric RBF-PS method even though this may not be warranted theoretically.

Program (LRBF.m)

```
function [L,x,y] = LRBF(N,rbf,Lrbf,ep)
if N==0, L=0; x=1; return, end
x = cos(pi*(0:N)/N)';    % Chebyshev points
y = x; [xx,yy] = meshgrid(x,y);
points = [xx(:) yy(:)];
r = DistanceMatrix(points,points);
A = rbf(ep,r);
AL = Lrbf(ep,r);
L = AL/A;
```

Summary

- Coupling RBF collocation and PS methods yields additional insights about RBF methods
- Provides “standard" procedure for solving time-dependent PDEs with RBFs
- Can apply many standard PS procedures to RBF solvers, but now can take advantage of scattered (multivariate) grids
- RBF-PS method for $\varepsilon = 0$ identical to Chebyshev-PS method and more accurate for small ε
- RBF-PS method has been applied successfully to a number of engineering problems (see, e.g., [Ferreira & F. (2006), Ferreira & F. (2007)])

Future work:

- Need to think about stable way to compute larger problems with RBFs (preconditioning) – especially for eigenvalue problems
- Need efficient computation of differentiation matrix analogous to FFT
- Can think about adaptive PS methods with moving grids

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.

Fornberg, B. (1998).
A Practical Guide to Pseudospectral Methods.
Cambridge Univ. Press.

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

Trefethen, L. N. (2000).
Spectral Methods in MATLAB.
SIAM (Philadelphia, PA).

References II

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

Fasshauer, G. E. (1997).
Solving partial differential equations by collocation with radial basis functions.
in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 131–138.

Ferreira, A. J. M. and Fasshauer, G. E. (2006).
Computation of natural frequencies of shear deformable beams and plates by an RBF-pseudospectral method.
Comput. Methods Appl. Mech. Engrg. **196**, pp. 134–146.

Ferreira, A. J. M. and Fasshauer, G. E. (2007).
Analysis of natural frequencies of composite plates by an RBF-pseudospectral method.
Composite Structures, **79**, pp. 202–210.

References III

Kansa, E. J. (1986).
Application of Hardy's multiquadric interpolation to hydrodynamics.
Proc. 1986 Simul. Conf. **4**, pp. 111–117.

Schaback, R. (2007).
Convergence of unsymmetric kernel-based meshless collocation methods,
SIAM J. Numer. Anal. **45**(1), pp. 333–351.