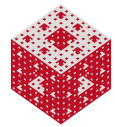
MATH 590: Meshfree Methods

Non-Symmetric Kernel Collocation for the Solution of Elliptic Partial Differential Equations

Greg Fasshauer

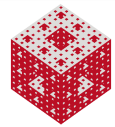
Department of Applied Mathematics
Illinois Institute of Technology

Fall 2014

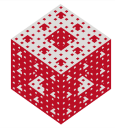


Outline

1. Kansa's Approach
2. MATLAB Implementation of Kansa's Method
3. Error Bounds for Non-Symmetric Collocation
4. Non-Symmetric Collocation with the Hilbert–Schmidt SVD
5. Toward Solving the Navier-Stokes Equations

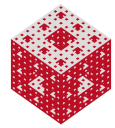


In this chapter we discuss how the techniques we studied for interpolation can be applied to the numerical solution of elliptic PDEs.



In this chapter we discuss how the techniques we studied for interpolation can be applied to the numerical solution of elliptic PDEs.

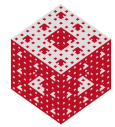
The resulting numerical method will be a collocation approach based on positive definite kernels.



In this chapter we discuss how the techniques we studied for interpolation can be applied to the numerical solution of elliptic PDEs.

The resulting numerical method will be a collocation approach based on positive definite kernels.

In the PDE literature this is also often referred to as a strong form solution.

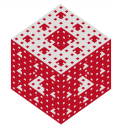


In this chapter we discuss how the techniques we studied for interpolation can be applied to the numerical solution of elliptic PDEs.

The resulting numerical method will be a collocation approach based on positive definite kernels.

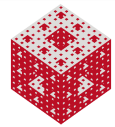
In the PDE literature this is also often referred to as a strong form solution.

To make the discussion transparent we will focus on the case of a time independent linear elliptic PDE in $\mathbb{R}^2$.

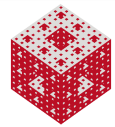


Outline

1. Kansa's Approach
2. MATLAB Implementation of Kansa's Method
3. Error Bounds for Non-Symmetric Collocation
4. Non-Symmetric Collocation with the Hilbert–Schmidt SVD
5. Toward Solving the Navier-Stokes Equations



A now very popular non-symmetric method for the solution of elliptic PDEs with RBFs was suggested by Ed Kansa in [Kan90].



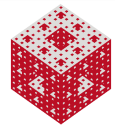
A now very popular non-symmetric method for the solution of elliptic PDEs with RBFs was suggested by Ed Kansa in [Kan90].

An alternative to

- Kansa's method
- is a symmetric approach proposed in [Fas97].

However, we will not have time to cover this alternative approach.

To begin, it will help to recall some of the basics of scattered data interpolation with kernels in $\mathbb{R}^d$.



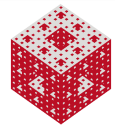
For scattered data interpolation we are given data $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, where we think of the y_i being samples of a function $f : \mathbb{R}^d \to \mathbb{R}$.

For scattered data interpolation we are given data $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, where we think of the y_i being samples of a function $f : \mathbb{R}^d \to \mathbb{R}$.
The goal is to find an interpolant of the form

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

such that

$$s(\boldsymbol{x}_i) = y_i, \qquad i = 1, \ldots, N.$$



For scattered data interpolation we are given data $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, where we think of the y_i being samples of a function $f : \mathbb{R}^d \to \mathbb{R}$.
The goal is to find an interpolant of the form

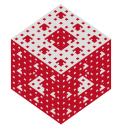
$$s(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^d, \tag{1}$$

such that

$$s(\boldsymbol{x}_i) = y_i, \qquad i = 1, \ldots, N.$$

The solution leads to a linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ with

$$\mathrm{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j), \qquad i, j = 1, \ldots, N. \tag{2}$$



For scattered data interpolation we are given data $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, where we think of the y_i being samples of a function $f : \mathbb{R}^d \to \mathbb{R}$.
The goal is to find an interpolant of the form

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

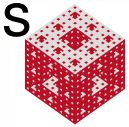
such that

$$s(\boldsymbol{x}_i) = y_i, \qquad i = 1, \ldots, N.$$

The solution leads to a linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$ with

$$\mathrm{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j), \qquad i, j = 1, \ldots, N. \tag{2}$$

The matrix K is non-singular for a large class of kernels including (inverse) multiquadrics, Gaussians, Matérn kernels and the CSRBFs of Wendland.



We now switch to the collocation solution of PDEs.

We now switch to the collocation solution of PDEs.
Assume we are given

- a domain $\Omega \subset \mathbb{R}^d$,

We now switch to the collocation solution of PDEs.
Assume we are given

- a domain $\Omega \subset \mathbb{R}^d$,
- and a linear elliptic PDE of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ in } \Omega, \tag{3}$$

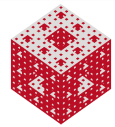
We now switch to the collocation solution of PDEs.
Assume we are given

- a domain $\Omega \subset \mathbb{R}^d$,
- and a linear elliptic PDE of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ in } \Omega, \tag{3}$$

- with (for simplicity of description) Dirichlet boundary conditions

$$\mathcal{B}u(\boldsymbol{x}) = u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ on } \partial\Omega. \tag{4}$$



We now switch to the collocation solution of PDEs.
Assume we are given

- a domain $\Omega \subset \mathbb{R}^d$,
- and a linear elliptic PDE of the form

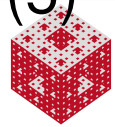
$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ in } \Omega, \tag{3}$$

- with (for simplicity of description) Dirichlet boundary conditions

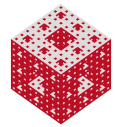
$$\mathcal{B}u(\boldsymbol{x}) = u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ on } \partial\Omega. \tag{4}$$

For Kansa's collocation method we then choose to represent the approximate solution $\hat{u}$ by a kernel expansion analogous to that used for scattered data interpolation, i.e.,

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{z}_j). \tag{5}$$

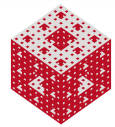


- To be able to easier distinguish between points that act as centers and those that act as "data sites" we now introduce the following notation for
  - centers $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and
  - collocation points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$.

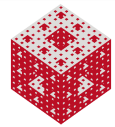


- To be able to easier distinguish between points that act as centers and those that act as "data sites" we now introduce the following notation for
  - centers $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and
  - collocation points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$.

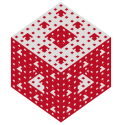
- While formally different, these points will often physically coincide.



- To be able to easier distinguish between points that act as centers and those that act as "data sites" we now introduce the following notation for
  - centers $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and
  - collocation points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$.
- While formally different, these points will often physically coincide.
- A scenario with $\mathcal{Z} \neq \mathcal{X}$ will be explored later.



- To be able to easier distinguish between points that act as centers and those that act as "data sites" we now introduce the following notation for
  - centers $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and
  - collocation points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$.
- While formally different, these points will often physically coincide.
- A scenario with $\mathcal{Z} \neq \mathcal{X}$ will be explored later.
- For the following discussion we assume the simplest possible setting, i.e., $\mathcal{Z} = \mathcal{X}$ and no polynomial terms are added to the expansion (5).



The collocation matrix that arises when matching the differential equation (3) and the boundary conditions (4) at the collocation points $\mathcal{X}$ will be of the form

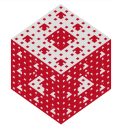
$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_{\mathcal{L}} \\ \mathrm{K}_{\mathcal{B}} \end{bmatrix}, \tag{6}$$

The collocation matrix that arises when matching the differential equation (3) and the boundary conditions (4) at the collocation points $\mathcal{X}$ will be of the form

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_{\mathcal{L}} \\ \mathrm{K}_{\mathcal{B}} \end{bmatrix}, \tag{6}$$

where the two blocks are generated as follows:

$$(\mathrm{K}_{\mathcal{L}})_{ij} = \mathcal{L}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \qquad \boldsymbol{x}_i \in \mathcal{X}_{\text{int}},\ \boldsymbol{z}_j \in \mathcal{Z},$$
$$(\mathrm{K}_{\mathcal{B}})_{ij} = \mathcal{B}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i} = K(\boldsymbol{x}_i, \boldsymbol{z}_j), \qquad \boldsymbol{x}_i \in \mathcal{X}_{\text{bdy}},\ \boldsymbol{z}_j \in \mathcal{Z}.$$



The collocation matrix that arises when matching the differential equation (3) and the boundary conditions (4) at the collocation points $\mathcal{X}$ will be of the form

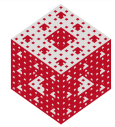
$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_{\mathcal{L}} \\ \mathrm{K}_{\mathcal{B}} \end{bmatrix}, \tag{6}$$

where the two blocks are generated as follows:

$$\begin{aligned} (\mathrm{K}_{\mathcal{L}})_{ij} &= \mathcal{L}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i}, & \boldsymbol{x}_i \in \mathcal{X}_{\text{int}},\ \boldsymbol{z}_j \in \mathcal{Z}, \\ (\mathrm{K}_{\mathcal{B}})_{ij} &= \mathcal{B}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i} = K(\boldsymbol{x}_i, \boldsymbol{z}_j), & \boldsymbol{x}_i \in \mathcal{X}_{\text{bdy}},\ \boldsymbol{z}_j \in \mathcal{Z}. \end{aligned}$$

Here the set $\mathcal{X}$ of collocation points is split into

- a set $\mathcal{X}_{\text{int}}$ of interior points,
- and a set $\mathcal{X}_{\text{bdy}}$ of boundary points.



The collocation matrix that arises when matching the differential equation (3) and the boundary conditions (4) at the collocation points $\mathcal{X}$ will be of the form

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_{\mathcal{L}} \\ \mathrm{K}_{\mathcal{B}} \end{bmatrix}, \tag{6}$$

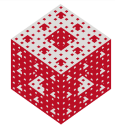
where the two blocks are generated as follows:

$$\begin{aligned} (\mathrm{K}_{\mathcal{L}})_{ij} &= \mathcal{L}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i}, & \boldsymbol{x}_i \in \mathcal{X}_{\text{int}},\ \boldsymbol{z}_j \in \mathcal{Z}, \\ (\mathrm{K}_{\mathcal{B}})_{ij} &= \mathcal{B}K(\boldsymbol{x}, \boldsymbol{z}_j)|_{\boldsymbol{x}=\boldsymbol{x}_i} = K(\boldsymbol{x}_i, \boldsymbol{z}_j), & \boldsymbol{x}_i \in \mathcal{X}_{\text{bdy}},\ \boldsymbol{z}_j \in \mathcal{Z}. \end{aligned}$$

Here the set $\mathcal{X}$ of collocation points is split into

- a set $\mathcal{X}_{\text{int}}$ of interior points,
- and a set $\mathcal{X}_{\text{bdy}}$ of boundary points.

The problem is well-posed if the $N \times N$ linear system $\mathrm{K}\boldsymbol{c} = \boldsymbol{y}$, with $\boldsymbol{y}$ a vector consisting of entries $f(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{X}_{\text{int}}$, followed by $g(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{X}_{\text{bdy}}$, has a unique solution.



Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix* K *in (6)*
  - *as well as on the right-hand side* $\mathbf{y}$.

Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix K in (6)*
  - *as well as on the right-hand side* $\mathbf{y}$*.*
- *The description here is rather general with no particular kernel in mind. However, Kansa specifically proposed to use multiquadric RBFs in (5), and consequently the method is sometimes also called the multiquadric method.*

Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix* K *in (6)*
  - *as well as on the right-hand side* $\boldsymbol{y}$.
- *The description here is rather general with no particular kernel in mind. However, Kansa specifically proposed to use multiquadric RBFs in (5), and consequently the method is sometimes also called the multiquadric method.*
- *Kansa also suggests the use of varying shape parameters* ε_j, $j = 1, \ldots, N$.

Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix* K *in (6)*
  - *as well as on the right-hand side* $\boldsymbol{y}$.
- *The description here is rather general with no particular kernel in mind. However, Kansa specifically proposed to use multiquadric RBFs in (5), and consequently the method is sometimes also called the multiquadric method.*
- *Kansa also suggests the use of varying shape parameters* ε_j, $j = 1, \ldots, N$.
  - *While the theoretical analysis of the resulting method is nearly intractable, Kansa showed that this technique improves the accuracy and stability of the method.*

Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix K in (6)*
  - *as well as on the right-hand side* $\boldsymbol{y}$.
- *The description here is rather general with no particular kernel in mind. However, Kansa specifically proposed to use multiquadric RBFs in (5), and consequently the method is sometimes also called the multiquadric method.*
- *Kansa also suggests the use of varying shape parameters* ε_j, $j = 1, \ldots, N$.
  - *While the theoretical analysis of the resulting method is nearly intractable, Kansa showed that this technique improves the accuracy and stability of the method.*
  - *In an RBF-FD setting, the recent paper [BMK12] studies the use of (optimal) variable shape parameters.*

Remark

- *A change in the boundary conditions (4) is as simple as*
  - *making changes to a few rows of the matrix K in (6)*
  - *as well as on the right-hand side* $\boldsymbol{y}$.
- *The description here is rather general with no particular kernel in mind. However, Kansa specifically proposed to use multiquadric RBFs in (5), and consequently the method is sometimes also called the multiquadric method.*
- *Kansa also suggests the use of varying shape parameters* ε_j, $j = 1, \ldots, N$.
  - *While the theoretical analysis of the resulting method is nearly intractable, Kansa showed that this technique improves the accuracy and stability of the method.*
  - *In an RBF-FD setting, the recent paper [BMK12] studies the use of (optimal) variable shape parameters.*
  - *Some promising insights into the theoretical aspects of varying shape parameters (in the interpolation setting) were recently provided in [BLRS14].*

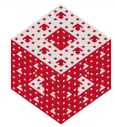
Problem with Kansa's method:

Problem with Kansa's method:

- For a constant shape parameter ε the matrix K may be singular for certain configurations of the centers $\boldsymbol{z}_j$.

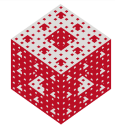
Problem with Kansa's method:

- For a constant shape parameter ε the matrix K may be singular for certain configurations of the centers $\boldsymbol{z}_j$.
- Originally, Kansa assumed that the non-singularity results established by Micchelli for interpolation matrices would carry over to the PDE case.



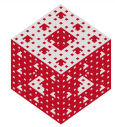
Problem with Kansa's method:

- For a constant shape parameter ε the matrix K may be singular for certain configurations of the centers $\boldsymbol{z}_j$.
- Originally, Kansa assumed that the non-singularity results established by Micchelli for interpolation matrices would carry over to the PDE case.
- As the numerical experiments of [HS01] show, this is not so.



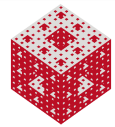
Problem with Kansa's method:

- For a constant shape parameter ε the matrix K may be singular for certain configurations of the centers $\boldsymbol{z}_j$.
- Originally, Kansa assumed that the non-singularity results established by Micchelli for interpolation matrices would carry over to the PDE case.
- As the numerical experiments of [HS01] show, this is not so.
  - This fact is not really surprising since the matrix for the collocation problem is composed of rows that are built from different functions, which — depending on the differential operator $\mathcal{L}$ — might not even correspond to a positive definite kernel.

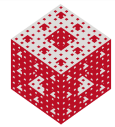


Problem with Kansa's method:

- For a constant shape parameter ε the matrix K may be singular for certain configurations of the centers $\boldsymbol{z}_j$.
- Originally, Kansa assumed that the non-singularity results established by Micchelli for interpolation matrices would carry over to the PDE case.
- As the numerical experiments of [HS01] show, this is not so.
  - This fact is not really surprising since the matrix for the collocation problem is composed of rows that are built from different functions, which — depending on the differential operator $\mathcal{L}$ — might not even correspond to a positive definite kernel.
  - The results for the non-singularity of interpolation matrices, however, are based on the fact that K is generated by a single (conditionally) positive definite kernel K.

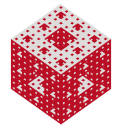


Nevertheless, an indication of the success of Kansa's method are the early papers [Dub92, Dub94, GCK96, Kan92, MK94] and many, many more since.



Nevertheless, an indication of the success of Kansa's method are the early papers [Dub92, Dub94, GCK96, Kan92, MK94] and many, many more since.

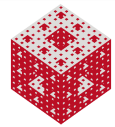
Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix.



Nevertheless, an indication of the success of Kansa's method are the early papers [Dub92, Dub94, GCK96, Kan92, MK94] and many, many more since.

Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix.

One possible approach — built on the basic ideas of greedy algorithms (see, e.g., [Fas07, Chapter 33]) — is to adaptively select "good" centers from a large set of possible candidates.

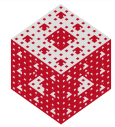


Nevertheless, an indication of the success of Kansa's method are the early papers [Dub92, Dub94, GCK96, Kan92, MK94] and many, many more since.

Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix.

One possible approach — built on the basic ideas of greedy algorithms (see, e.g., [Fas07, Chapter 33]) — is to adaptively select "good" centers from a large set of possible candidates.

Following this strategy it is possible to ensure invertibility of the collocation matrix throughout the iterative algorithm.



Nevertheless, an indication of the success of Kansa's method are the early papers [Dub92, Dub94, GCK96, Kan92, MK94] and many, many more since.

Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix.

One possible approach — built on the basic ideas of greedy algorithms (see, e.g., [Fas07, Chapter 33]) — is to adaptively select "good" centers from a large set of possible candidates.

Following this strategy it is possible to ensure invertibility of the collocation matrix throughout the iterative algorithm.

This approach is described in the paper [LOS06] (see discussion and example later).

Outline

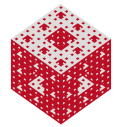
1 Kansa's Approach

2 MATLAB Implementation of Kansa's Method

3 Error Bounds for Non-Symmetric Collocation

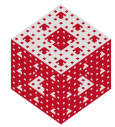
4 Non-Symmetric Collocation with the Hilbert–Schmidt SVD

5 Toward Solving the Navier-Stokes Equations

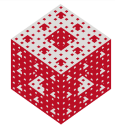


- We present a number of MATLAB implementations for
  - standard Laplace/Possion problems,
  - problems with variable coefficients,
  - and problems with mixed or piecewise defined boundary conditions.

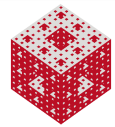
- We present a number of MATLAB implementations for
  - standard Laplace/Possion problems,
  - problems with variable coefficients,
  - and problems with mixed or piecewise defined boundary conditions.
- We provide a fairly detailed presentation since the MATLAB code changes rather significantly from one problem to another.



- We present a number of MATLAB implementations for
  - standard Laplace/Possion problems,
  - problems with variable coefficients,
  - and problems with mixed or piecewise defined boundary conditions.
- We provide a fairly detailed presentation since the MATLAB code changes rather significantly from one problem to another.
- Most of the test examples are similar to those in [LCC03].



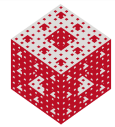
- We present a number of MATLAB implementations for
  - standard Laplace/Possion problems,
  - problems with variable coefficients,
  - and problems with mixed or piecewise defined boundary conditions.
- We provide a fairly detailed presentation since the MATLAB code changes rather significantly from one problem to another.
- Most of the test examples are similar to those in [LCC03].
- All problems are 2D elliptic with known analytic solution.



Consider the following Poisson problem with Dirichlet boundary conditions:

$$\begin{aligned} \nabla^2 u(x,y) &= -\frac{5}{4}\pi^2 \sin(\pi x)\cos\left(\frac{\pi y}{2}\right), \quad (x,y) \in \Omega = [0,1]^2, \quad (7) \\ u(x,y) &= \sin(\pi x), \quad (x,y) \in \Gamma_1, \\ u(x,y) &= 0, \quad (x,y) \in \Gamma_2, \end{aligned}$$

where $\Gamma_1 = \{(x,y) : \ 0 \le x \le 1, \ y = 0\}$ and $\Gamma_2 = \partial\Omega \setminus \Gamma_1$.



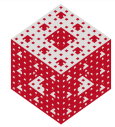
Consider the following Poisson problem with Dirichlet boundary conditions:

$$\begin{aligned} \nabla^2 u(x,y) &= -\frac{5}{4}\pi^2 \sin(\pi x)\cos\left(\frac{\pi y}{2}\right), \quad (x,y) \in \Omega = [0,1]^2, \quad (7) \\ u(x,y) &= \sin(\pi x), \quad (x,y) \in \Gamma_1, \\ u(x,y) &= 0, \quad (x,y) \in \Gamma_2, \end{aligned}$$

where $\Gamma_1 = \{(x,y) : \ 0 \le x \le 1, \ y = 0\}$ and $\Gamma_2 = \partial\Omega \setminus \Gamma_1$.

The exact solution is given by

$$u(x,y) = \sin(\pi x)\cos\left(\frac{\pi y}{2}\right).$$

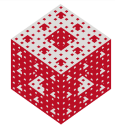


Program (KansaLaplace_2D.m)

```
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
 3  u = @(x,y) sin(pi*x).*cos(pi*y/2);
 4  Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
 5  N = 289;  [collpts, N] = CreatePoints(N, 2, 'u');
 6a indx = find(collpts(:,1)==0 | collpts(:,2)==0 | ...
 6b             collpts(:,1)==1 | collpts(:,2)==1);
 7  bdypts = collpts(indx,:);  % find boundary points
 8  intpts = collpts(setdiff([1:N],indx),:);  % interior points
 9  ctrs = [intpts; bdypts];
10  M = 1600;  epoints = CreatePoints(M,2,'u');
11  DM_eval = DistanceMatrix(epoints,ctrs);
12  EM = rbf(ep,DM_eval);
13  exact = u(epoints(:,1),epoints(:,2));
14  DM_int = DistanceMatrix(intpts,ctrs); LCM = Lrbf(ep,DM_int);
15  DM_bdy = DistanceMatrix(bdypts,ctrs); BCM = rbf(ep,DM_bdy);
16  CM = [LCM; BCM];
17a rhs = [Lu(intpts(:,1),intpts(:,2)); ...
17b       u(bdypts(:,1),bdypts(:,2))];
18  s = EM * (CM\rhs);
19  rms_err = norm(s-exact)/sqrt(M);
```

Remark

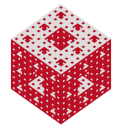
We "cheat" when we define the right-hand side of the problem (on line 17) since we simply evaluate the known solution on the boundary.



Remark

We "cheat" when we define the right-hand side of the problem (on line 17) since we simply evaluate the known solution on the boundary.

Of course, in general the solution will not be known, and this will not be possible.



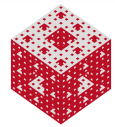
Remark

We "cheat" when we define the right-hand side of the problem (on line 17) since we simply evaluate the known solution on the boundary.

Of course, in general the solution will not be known, and this will not be possible.

In that case one would have to replace line 17 by something like (see `KansaLaplace_2D.m` *on the website)*

```
rhs = zeros(N,1);  NI = size(intpts,1);
rhs(1:NI) = Lu(intpts(:,1),intpts(:,2));
indx = find(bdypts(:,2)==0);
rhs(NI+indx) = sin(pi*bdypts(indx,1));
```



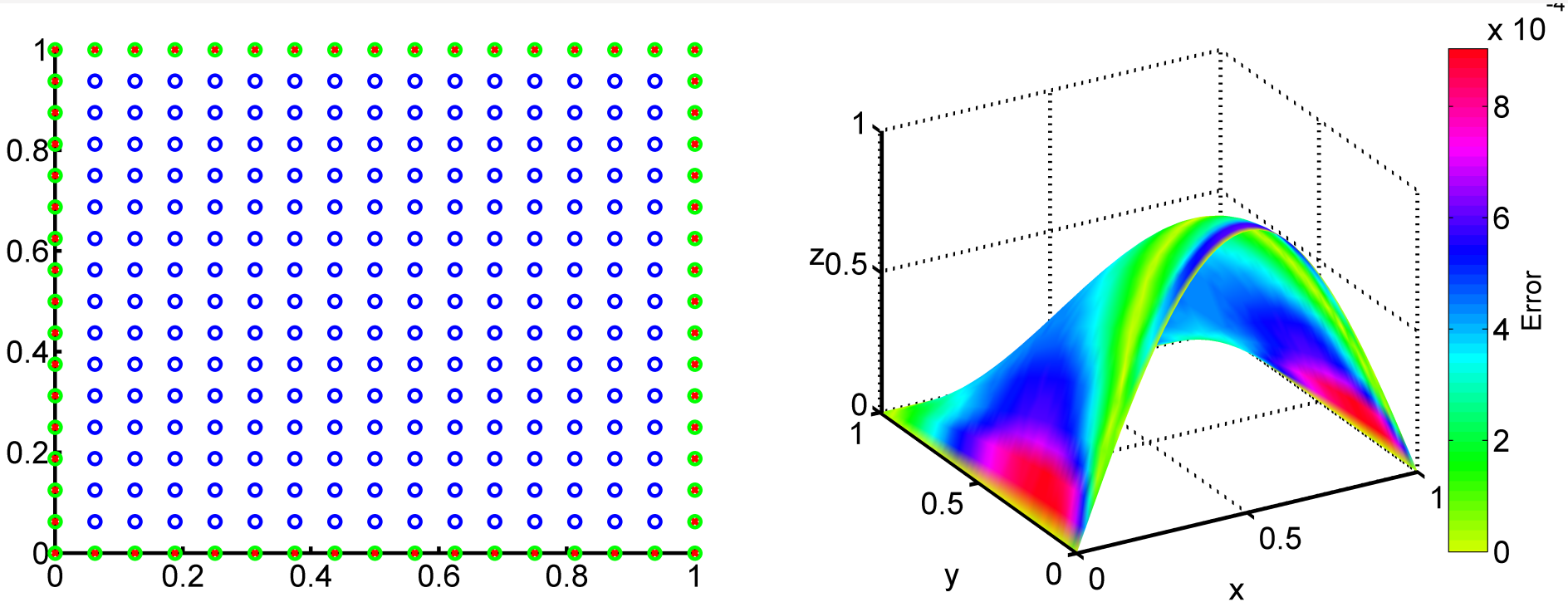
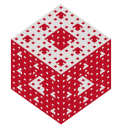


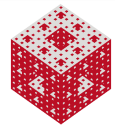
Figure: Collocation points (interior: blue circles, boundary: red crosses) and centers (interior: blue circles, boundary: green circles) (left) and non-symmetric RBF collocation solution (right) using IMQs with $\varepsilon = 3$ and $N = 289$ points.



There are several ways to handle collocation of the boundary conditions:

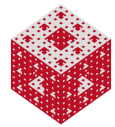
There are several ways to handle collocation of the boundary conditions:

- The most natural approach is to use those collocation points that lie on the boundary as boundary collocation points (and centers). This matches the theory discussed earlier and is what is implemented in the code above.



There are several ways to handle collocation of the boundary conditions:

- The most natural approach is to use those collocation points that lie on the boundary as boundary collocation points (and centers). This matches the theory discussed earlier and is what is implemented in the code above.
- We can create additional collocation points for the boundary conditions. These points can lie anywhere on the boundary. This is what is implemented in the code in [Fas07] (see also `KansaLaplace_2DBook.m` on the website).



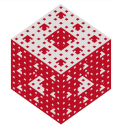
Now we have several choices for the boundary centers:

Now we have several choices for the boundary centers:

- We can let the boundary centers coincide with the boundary collocation points.
  - This approach will lead to a singular collocation matrix for uniform interior points (since that set already contains points on the boundary, and therefore duplicate columns are created).

Now we have several choices for the boundary centers:

- We can let the boundary centers coincide with the boundary collocation points.
  - This approach will lead to a singular collocation matrix for uniform interior points (since that set already contains points on the boundary, and therefore duplicate columns are created).
  - This approach does work if we take the interior collocation points to be Halton points (or some other set of points that do not lie on the boundary).

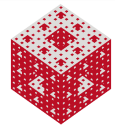


Now we have several choices for the boundary centers:

- We can let the boundary centers coincide with the boundary collocation points.
  - This approach will lead to a singular collocation matrix for uniform interior points (since that set already contains points on the boundary, and therefore duplicate columns are created).
  - This approach does work if we take the interior collocation points to be Halton points (or some other set of points that do not lie on the boundary).
  - This approach can be realized by uncommenting the line

    ```
    bdyctrs = bdydata;
    ```

    in the MATLAB routine `KansaLaplace_2DBook.m`.

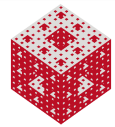


Now we have several choices for the boundary centers:

- We can let the boundary centers coincide with the boundary collocation points.
  - This approach will lead to a singular collocation matrix for uniform interior points (since that set already contains points on the boundary, and therefore duplicate columns are created).
  - This approach does work if we take the interior collocation points to be Halton points (or some other set of points that do not lie on the boundary).
  - This approach can be realized by uncommenting the line

    ```
    bdyctrs = bdydata;
    ```

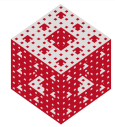
    in the MATLAB routine `KansaLaplace_2DBook.m`.
  - However, care must be taken that the Halton points do not include the origin, i.e., the MATLAB `haltonset` command will not work without modification.



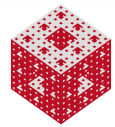
- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).

- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).
  - We follow this approach in most experiments since it seems to be slightly more accurate.

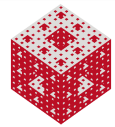
- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).
  - We follow this approach in most experiments since it seems to be slightly more accurate.
  - Placing boundary centers away from the boundary has been recommended by a number of authors.



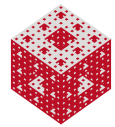
- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).
  - We follow this approach in most experiments since it seems to be slightly more accurate.
  - Placing boundary centers away from the boundary has been recommended by a number of authors.
  - This approach takes us into the realm of kernel methods for which the centers differ from the data sites (or collocation points), and we stated earlier that not much is known theoretically about this setting (i.e., invertibility of system matrices or error bounds).



- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).
  - We follow this approach in most experiments since it seems to be slightly more accurate.
  - Placing boundary centers away from the boundary has been recommended by a number of authors.
  - This approach takes us into the realm of kernel methods for which the centers differ from the data sites (or collocation points), and we stated earlier that not much is known theoretically about this setting (i.e., invertibility of system matrices or error bounds).
  - It is an open problem how to find the best location for the boundary centers.



- We can create additional boundary centers outside the domain (see `KansaLaplace_2DBook.m` or the discussion in [Fas07])).
  - We follow this approach in most experiments since it seems to be slightly more accurate.
  - Placing boundary centers away from the boundary has been recommended by a number of authors.
  - This approach takes us into the realm of kernel methods for which the centers differ from the data sites (or collocation points), and we stated earlier that not much is known theoretically about this setting (i.e., invertibility of system matrices or error bounds).
  - It is an open problem how to find the best location for the boundary centers.
  - We take them a small distance perpendicularly from the boundary collocation points (see the following figure).



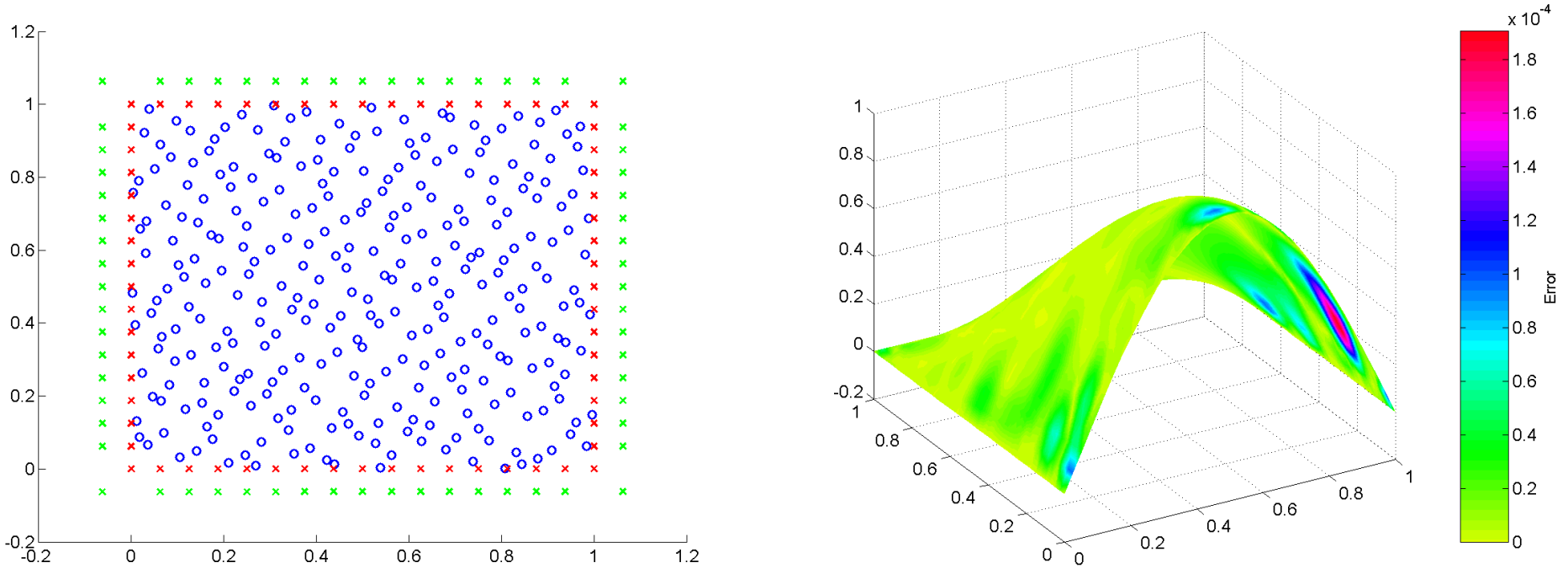
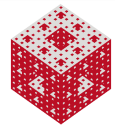
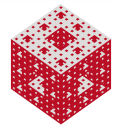


Figure: Collocation points (interior: blue circles, boundary: red crosses) and centers (interior: blue circles, boundary: green crosses) (left) and non-symmetric RBF collocation solution (right) using IMQs with $\varepsilon = 3$ and $N = 289$ interior points.



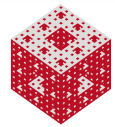
N_{int}	centers on boundary		centers outside	
	RMS-error	cond(A)	RMS-error	cond(A)
9	5.642192e-02	5.276474e+02	6.029293e-02	4.399608e+02
25	1.039322e-02	3.418858e+03	4.187975e-03	2.259698e+03
81	2.386062e-03	1.726995e+06	4.895870e-04	3.650369e+05
289	4.904715e-05	1.706884e+10	2.668524e-05	5.328110e+09
1089	3.676576e-08	1.446865e+18	1.946954e-08	5.015917e+17

Table: Non-symmetric collocation solution with IMQs, $\varepsilon = 3$ and interior Halton points.



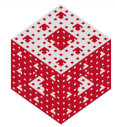
N_{int}	Gaussian		IMQ	
	RMS-error	cond(A)	RMS-error	cond(A)
3×3	1.981675e-01	1.258837e+03	1.526456e-01	2.794516e+02
5×5	7.199931e-03	4.136193e+03	6.096534e-03	2.409431e+03
9×9	1.947108e-04	2.529708e+10	8.071271e-04	8.771630e+05
17×17	4.174290e-08	5.335000e+19	3.219110e-05	5.981238e+10
33×33	1.408750e-05	7.106505e+20	1.552047e-07	1.706638e+20

Table: Non-symmetric collocation solution with Gaussians and IMQs, $\varepsilon = 3$ and uniform interior points and boundary centers outside the domain.



Remark

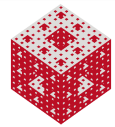
Several observations can be made by looking at the tables.



Remark

Several observations can be made by looking at the tables.

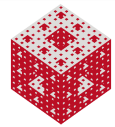
- *The use of Halton points instead of uniform points seems to be beneficial since both the errors and the condition numbers are smaller (cf. the right parts of the tables).*



Remark

Several observations can be made by looking at the tables.

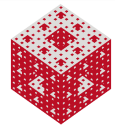
- *The use of Halton points instead of uniform points seems to be beneficial since both the errors and the condition numbers are smaller (cf. the right parts of the tables).*
- *Placement of the boundary centers outside the domain seems to be advantageous since again both the errors and the condition numbers decrease.*



Remark

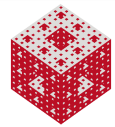
Several observations can be made by looking at the tables.

- *The use of Halton points instead of uniform points seems to be beneficial since both the errors and the condition numbers are smaller (cf. the right parts of the tables).*
- *Placement of the boundary centers outside the domain seems to be advantageous since again both the errors and the condition numbers decrease.*
- *Gaussians are more prone to ill-conditioning than inverse multiquadrics.*



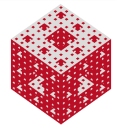
Remark

- *Of course, these are rather superficial observations based on only a few numerical experiments.*



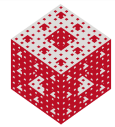
Remark

- *Of course, these are rather superficial observations based on only a few numerical experiments.*
- *For many of these claims there is no theoretical foundation, and many more experiments would be needed to make a more conclusive statement (for example, no attempt was made here to find the best approximations, i.e., optimize the value of the shape parameter).*



Remark

- *Of course, these are rather superficial observations based on only a few numerical experiments.*
- *For many of these claims there is no theoretical foundation, and many more experiments would be needed to make a more conclusive statement (for example, no attempt was made here to find the best approximations, i.e., optimize the value of the shape parameter).*
- *Also, one could experiment with different values of the shape parameter on the boundary and in the interior (as suggested, e.g., in [KC92, WKL06]).*



Consider the following elliptic equation with variable coefficients and homogeneous Dirichlet boundary conditions on $\Omega = [0,1]^2$:

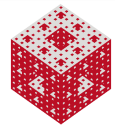
$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}u(x,y)\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}u(x,y)\right) = f(x,y), \quad (x,y) \in \Omega,$$
$$u(x,y) = 0, \quad (x,y) \in \Gamma = \partial\Omega,$$

where

$$f(x,y) = -16x(1-x)(3-2y)e^{x-y} + 32y(1-y)(3x^2+y^2-x-2),$$

and the coefficients are given by

$$a(x,y) = 2 - x^2 - y^2, \qquad b(x,y) = e^{x-y}.$$



Consider the following elliptic equation with variable coefficients and homogeneous Dirichlet boundary conditions on $\Omega = [0,1]^2$:

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}u(x,y)\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}u(x,y)\right) = f(x,y), \quad (x,y) \in \Omega,$$
$$u(x,y) = 0, \quad (x,y) \in \Gamma = \partial\Omega,$$

where

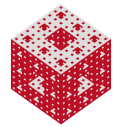
$$f(x,y) = -16x(1-x)(3-2y)e^{x-y} + 32y(1-y)(3x^2+y^2-x-2),$$

and the coefficients are given by

$$a(x,y) = 2 - x^2 - y^2, \qquad b(x,y) = e^{x-y}.$$

The exact solution for this problem is given by

$$u(x,y) = 16x(1-x)y(1-y).$$



Program (KansaEllipticVC_2D.m)

```
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  dxrbf = @(e,r,dx) -dx*e^2./(1+(e*r).^2).^(3/2);
 3  dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 4a dxxrbf = @(e,r,dx) e^2*(3*(e*dx).^2-1-(e*r).^2)./...
 4b                          (1+(e*r).^2).^(5/2);
 5a dyyrbf = @(e,r,dy) e^2*(3*(e*dy).^2-1-(e*r).^2)./...
 5b                          (1+(e*r).^2).^(5/2);
 6  u = @(x,y) 16*x.*(1-x).*y.*(1-y);
 7a Lu = @(x,y) -16*x.*exp(x-y).*(1-x).*(3-2*y)+...
 7b            32*y.*(1-y).*(3*x.^2+y.^2-x-2);
 8  a = @(x,y) 2-x.^2-y.^2; ax = @(x,y) -2*x;
 9  b = @(x,y) exp(x-y); by = @(x,y)-exp(x-y);
10  N = 289;  [collpts, N] = CreatePoints(N, 2, 'u');
11a indx = find(collpts(:,1)==0 | collpts(:,2)==0 | ...
11b             collpts(:,1)==1 | collpts(:,2)==1);
12  bdypts = collpts(indx,:);  % uniform boundary pts
13  [intpts, N] = CreatePoints(N, 2, 'h'); %Halton inside
14  sn = sqrt(N);  h = 1/(sn-1);
15  bdyctrs = bdypts;  bdyctrs = (1+2*h)*bdyctrs-h;
16  ctrs = [intpts; bdyctrs];
```

Program (KansaEllipticVC_2D.m (cont.))

```
17  M = 1600;  epoints = CreatePoints(M,2,'u');
18  DM_eval = DistanceMatrix(epoints,ctrs);
19  EM = rbf(ep,DM_eval);
20  exact = u(epoints(:,1),epoints(:,2));
21  DM_int = DistanceMatrix(intpts,ctrs);
22  DM_bdy = DistanceMatrix(bdypts,ctrs);
23  dx_int = Differencematrix(intpts(:,1),ctrs(:,1));
24  dy_int = Differencematrix(intpts(:,2),ctrs(:,2));
25a LCM = diag(ax(intpts(:,1)))*dxrbf(ep,DM_int,dx_int)+...
25b             diag(a(intpts(:,1),intpts(:,2))) * ...
25c             dxxrbf(ep,DM_int,dx_int) + ...
25d             diag(by(intpts(:,1),intpts(:,2))) * ...
25e             dyrbf(ep,DM_int,dy_int) + ...
25f             diag(b(intpts(:,1),intpts(:,2))) * ...
25g             dyyrbf(ep,DM_int,dy_int);
26  BCM = rbf(ep,DM_bdy);
27  CM = [LCM; BCM];
28  rhs = [Lu(intpts(:,1),intpts(:,2)); zeros(4*(sn-1),1)];
29  s = EM * (CM\rhs);
30  rms_err = norm(s-exact)/sqrt(M);
```

In the following table we compare the solution obtained with

- Gaussians
- and inverse multiquadrics

based on interior Halton points.

In the following table we compare the solution obtained with

- Gaussians
- and inverse multiquadrics

based on interior Halton points.

The boundary centers are taken to lie outside the domain.

In the following table we compare the solution obtained with

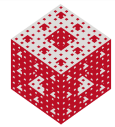
- Gaussians
- and inverse multiquadrics

based on interior Halton points.

The boundary centers are taken to lie outside the domain.

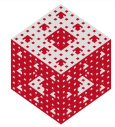
For Gaussians we need to replace lines 1–5 of `KansaEllipticVC_2D.m` by

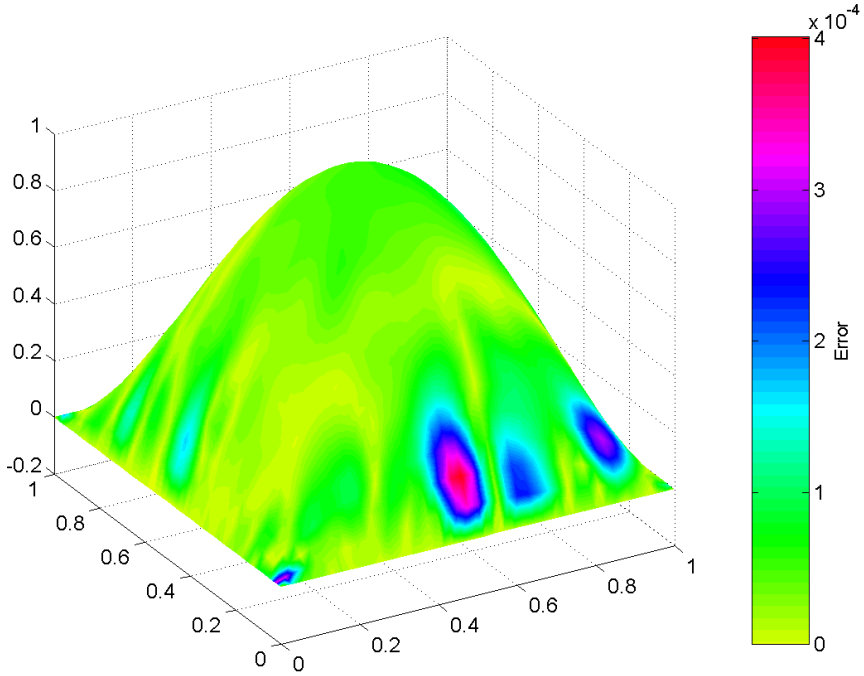
```
1   rbf = @(e,r) exp(-(e*r).^2); ep = 3;
2   dxrbf = @(e,r,dx) -2*dx*e^2.*exp(-(e*r).^2);
3   dyrbf = @(e,r,dy) -2*dy*e^2.*exp(-(e*r).^2);
4a dxxrbf = @(e,r,dx) 2*e^2*(2*(e*dx).^2-1) .* ...
4b                    exp(-(e*r).^2);
5a dyyrbf = @(e,r,dy) 2*e^2*(2*(e*dy).^2-1) .* ...
5b                    exp(-(e*r).^2);
```



N_{int}	Gaussian		IMQ	
	RMS-error	cond(A)	RMS-error	cond(A)
9	6.852103e-02	8.874341e+03	1.123770e-01	6.954910e+02
25	1.091888e-02	4.898291e+03	1.123575e-02	3.302471e+03
81	1.854386e-04	1.286993e+09	1.370992e-03	4.992219e+05
289	8.445637e-07	7.031011e+19	8.105109e-05	7.527456e+09
1089	2.559824e-05	4.553162e+20	7.041415e-08	7.785955e+17

Table: Solution with Gaussians and IMQs, $\varepsilon = 3$ and interior Halton points.





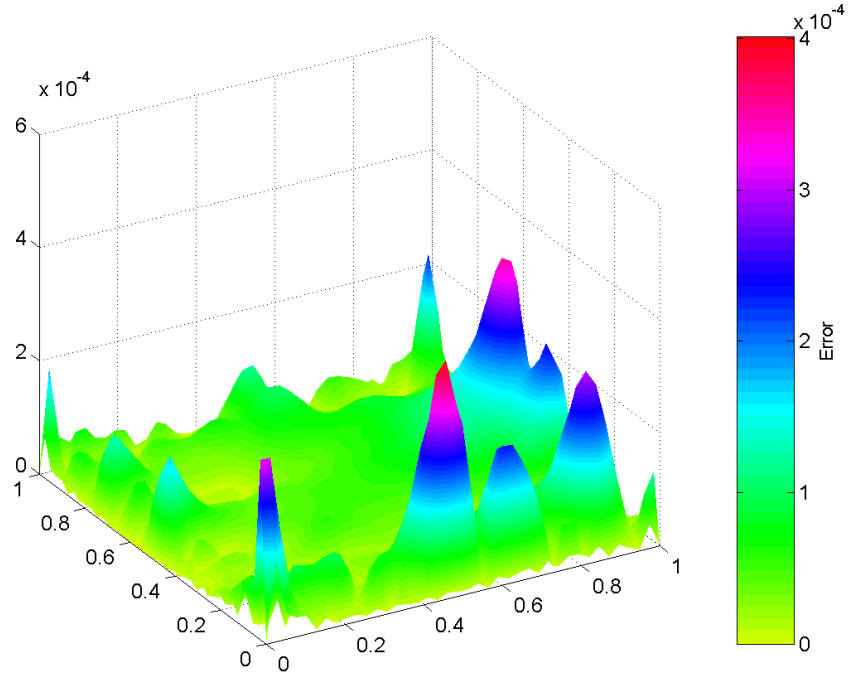
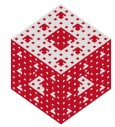


Figure: Non-symmetric collocation solution (left) and error plot (right) using IMQs with $\varepsilon = 3$ and $N = 289$ interior Halton points.

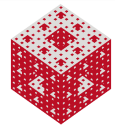


Consider the Poisson problem with mixed boundary conditions

$$\begin{aligned}
\nabla^2 u(x,y) &= -5.4x, \quad (x,y) \in \Omega = [0,1]^2, \\
\frac{\partial}{\partial \boldsymbol{n}} u(x,y) &= 0, \quad (x,y) \in \Gamma_1 \cup \Gamma_3, \\
u(x,y) &= 0.1, \quad (x,y) \in \Gamma_2, \\
u(x,y) &= 1, \quad (x,y) \in \Gamma_4,
\end{aligned}$$

where

$$\begin{aligned}
\Gamma_1 &= \{(x,y) : \ 0 \le x \le 1, \ y = 0\}, \\
\Gamma_2 &= \{(x,y) : \ x = 1, \ 0 \le y \le 1\}, \\
\Gamma_3 &= \{(x,y) : \ 0 \le x \le 1, \ y = 1\}, \\
\Gamma_4 &= \{(x,y) : \ x = 0, \ 0 \le y \le 1\}.
\end{aligned}$$



Consider the Poisson problem with mixed boundary conditions

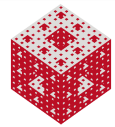
$$\begin{aligned}
\nabla^2 u(x,y) &= -5.4x, \quad (x,y) \in \Omega = [0,1]^2, \\
\frac{\partial}{\partial \boldsymbol{n}} u(x,y) &= 0, \quad (x,y) \in \Gamma_1 \cup \Gamma_3, \\
u(x,y) &= 0.1, \quad (x,y) \in \Gamma_2, \\
u(x,y) &= 1, \quad (x,y) \in \Gamma_4,
\end{aligned}$$

where

$$\begin{aligned}
\Gamma_1 &= \{(x,y) : \ 0 \le x \le 1, \ y = 0\}, \\
\Gamma_2 &= \{(x,y) : \ x = 1, \ 0 \le y \le 1\}, \\
\Gamma_3 &= \{(x,y) : \ 0 \le x \le 1, \ y = 1\}, \\
\Gamma_4 &= \{(x,y) : \ x = 0, \ 0 \le y \le 1\}.
\end{aligned}$$

The exact solution is given by

$$u(x,y) = 1 - 0.9x^3.$$



Remark

- *The normal derivative on the edges is given by:*

  on Γ_1: $\frac{\partial}{\partial y}$

  on Γ_3: $-\frac{\partial}{\partial y}$

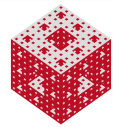
Remark

- *The normal derivative on the edges is given by:*

  on Γ_1: $\frac{\partial}{\partial y}$

  on Γ_3: $-\frac{\partial}{\partial y}$

- *Therefore, the* MATLAB *program also requires the y-partial of the basic function.*



Remark

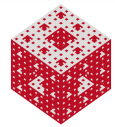
- *The normal derivative on the edges is given by:*

  on Γ_1: $\frac{\partial}{\partial y}$

  on Γ_3: $-\frac{\partial}{\partial y}$

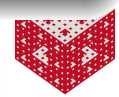
- *Therefore, the* MATLAB *program also requires the y-partial of the basic function.*

- *Note that this time we need to carefully pick out the different boundary collocation points (see lines 7–11).*



Program (KansaLaplaceMixedBC_2D.m)

```
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 3  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
 4  u = @(x,y) 1-0.9*x.^3+0*y;
 5  Lu = @(x,y) -5.4*x+0*y;
 6  N = 289;  [collpts, N] = CreatePoints(N, 2, 'u');
 7  indx1 = find(collpts(:,2)==0 & collpts(:,1)~=1);
 8  indx2 = find(collpts(:,1)==1 & collpts(:,2)~=1);
 9  indx3 = find(collpts(:,2)==1 & collpts(:,1)~=0);
10  indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
11  bdypts = collpts([indx1;indx2;indx3;indx4],:);
12  [intpts, N] = CreatePoints(N, 2, 'h');
13  sn = sqrt(N);  h = 1/(sn-1);
14  bdyctrs = bdypts;  bdyctrs = (1+2*h)*bdyctrs-h;
15  ctrs = [intpts; bdyctrs];
```

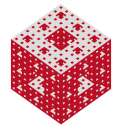


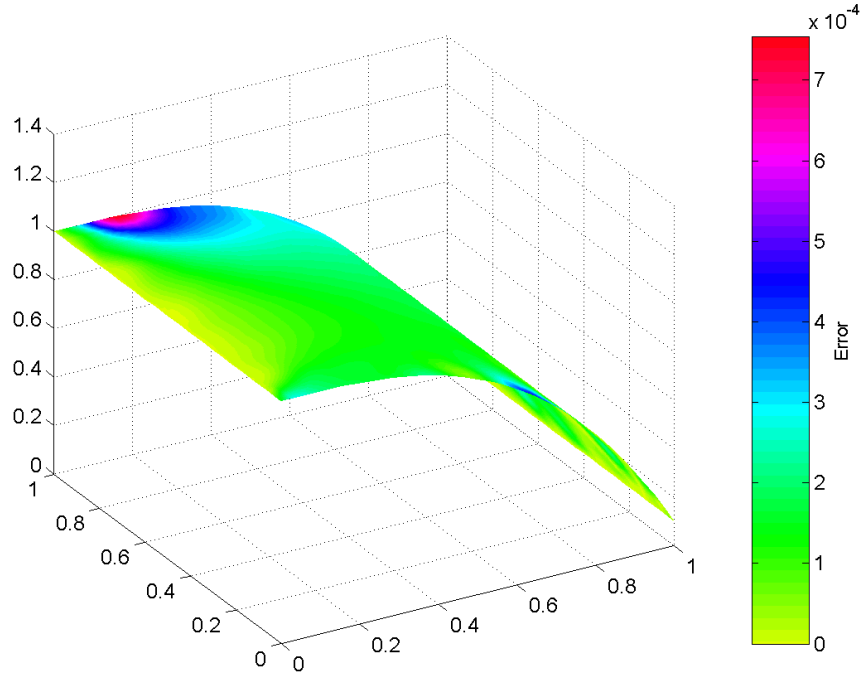
Program (KansaLaplaceMixedBC_2D.m (cont.))

```
16  M = 1600;  epoints = CreatePoints(M,2,'u');
17  DM_eval = DistanceMatrix(epoints,ctrs);
18  EM = rbf(ep,DM_eval);
19  exact = u(epoints(:,1),epoints(:,2));
20  DM_int = DistanceMatrix(intpts,ctrs);
21  DM_bdy = DistanceMatrix(bdypts,ctrs);
22  dy_bdy = Differencematrix(bdypts(:,2),ctrs(:,2));
23  LCM = Lrbf(ep,DM_int);
24  BCM1 = -dyrbf(ep,DM_bdy(1:sn-1,:),dy_bdy(1:sn-1,:));
25  BCM2 = rbf(ep,DM_bdy(sn:2*sn-2,:));
26a BCM3 = dyrbf(ep,DM_bdy(2*sn-1:3*sn-3,:),...
26b                dy_bdy(2*sn-1:3*sn-3,:));
27  BCM4 = rbf(ep,DM_bdy(3*sn-2:end,:));
28  CM = [LCM; BCM1; BCM2; BCM3; BCM4];
29a rhs = [Lu(intpts(:,1),intpts(:,2)); zeros(sn-1,1); ...
29b        0.1*ones(sn-1,1); zeros(sn-1,1); ones(sn-1,1)];
30  s = EM * (CM\rhs);
31  rms_err = norm(s-exact)/sqrt(M);
```

N_{int}	Gaussian		IMQ	
	RMS-error	cond(A)	RMS-error	cond(A)
9	3.423330e-01	5.430073e+03	7.937403e-02	2.782348e+02
25	1.065826e-02	1.605086e+03	5.605445e-03	1.680888e+03
81	5.382387e-04	3.684159e+08	1.487160e-03	2.611650e+05
289	6.181855e-06	1.452124e+19	1.822077e-04	3.775455e+09
1089	2.060470e-06	1.628262e+21	1.822221e-07	3.155751e+17

Table: Non-symmetric collocation solution with Gaussians and IMQs, $\varepsilon = 3$ and interior Halton points.





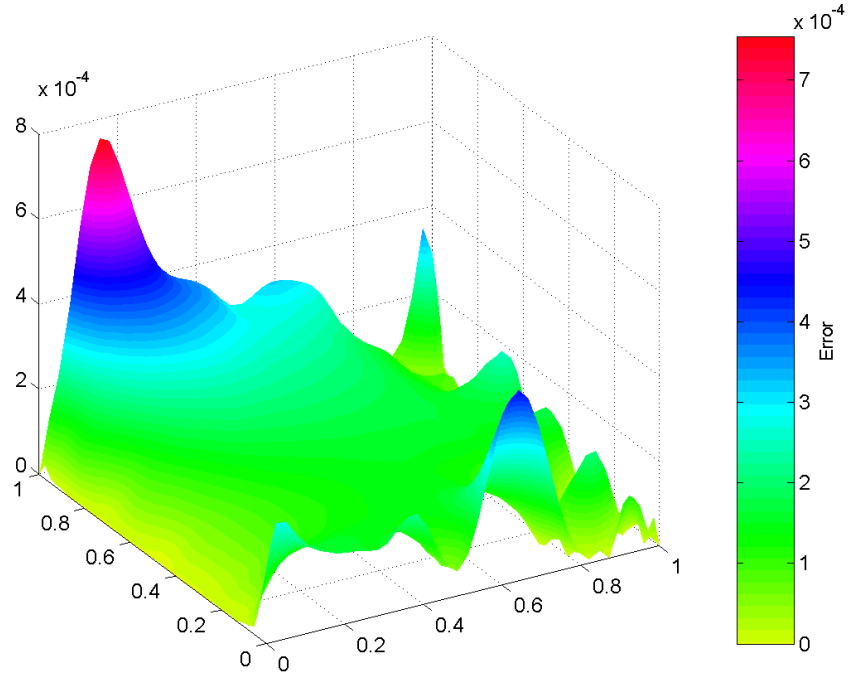
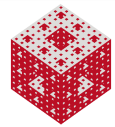
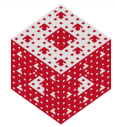


Figure: Approximate solution (left) and error plot (right) using IMQs with $\varepsilon = 3$ and $N = 289$ interior Halton points.



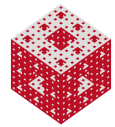
Remark

- *Note that — even though the problem has a symmetric solution — the approximate solution is not quite symmetric (as demonstrated by the error plot). This is due to the use of interior Halton points.*



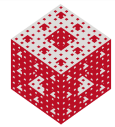
Remark

- *Note that — even though the problem has a symmetric solution — the approximate solution is not quite symmetric (as demonstrated by the error plot). This is due to the use of interior Halton points.*
- *The same happened in the previous example.*



Remark

- *Note that — even though the problem has a symmetric solution — the approximate solution is not quite symmetric (as demonstrated by the error plot). This is due to the use of interior Halton points.*
- *The same happened in the previous example.*
- *In [LCC03] the authors report that the non-symmetric collocation solution for this problem with multiquadric RBFs is several orders of magnitude more accurate than a solution with piecewise linear finite elements using the same number of nodes.*



Theorem ([LOS06])

Let Λ be an infinite set of real-valued linear functionals so that the PDE problem corresponds to $\lambda[u] = f_\lambda$, $\lambda \in \mathbb{R}$, and let $\Lambda_N \subset \Lambda$ be the finite set of test functionals used to discretize the PDE problem. Assume the kernel K to be smooth enough to guarantee that the functions $\lambda^{\boldsymbol{x}} K(\boldsymbol{x}, \cdot)$ for $\lambda \in \Lambda$ are continuous. Then the set of functions $\{\lambda_i^{\boldsymbol{x}} K(\boldsymbol{x}, \cdot),\ \lambda_i \in \Lambda_N\}_{i=1}^N$ is linearly independent, and hence Kansa's collocation matrix K with $\mathrm{K}_{ij} = \lambda_i^{\boldsymbol{x}} K(\boldsymbol{x}, \boldsymbol{z}_j)$ is nonsingular for properly chosen trial centers $\boldsymbol{z}_j$.

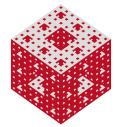
Remark

- *The functionals in Λ capture the action of both differential and boundary operators of the PDE problem.*
- *The functions $\lambda_i^{\boldsymbol{x}} K(\boldsymbol{x}, \cdot)$, $i = 1, \ldots, N$ define the rows of K.*
- *The trial functions $K(\cdot, \boldsymbol{z}_j)$, $j = 1, \ldots, N$ define the columns of K.*

One way to ensure the use of properly chosen trial functions is to

- pick a discretization, i.e., pick a set λ_N, and then
- use a (column-pivoted) QR decomposition of a short and fat matrix $\hat{K}$ based on a set $\mathcal{Z}$ of $M \gg N$ reasonably well distributed trial centers to find a full-rank (square) collocation matrix K.

Note that the set of trial centers may include points outside the domain Ω.



Example ([LOS06])

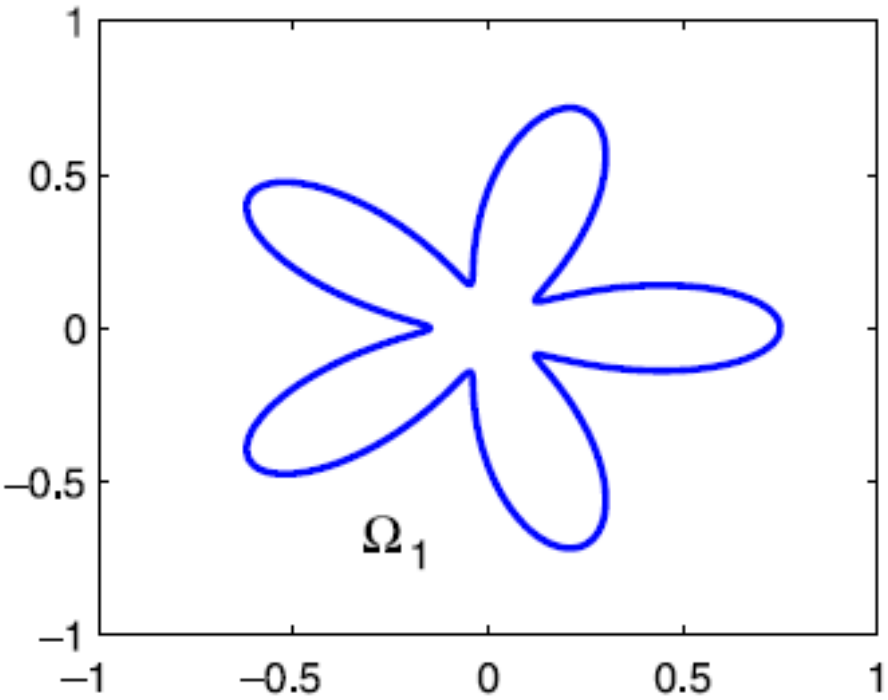
Use multiquadrics in the form

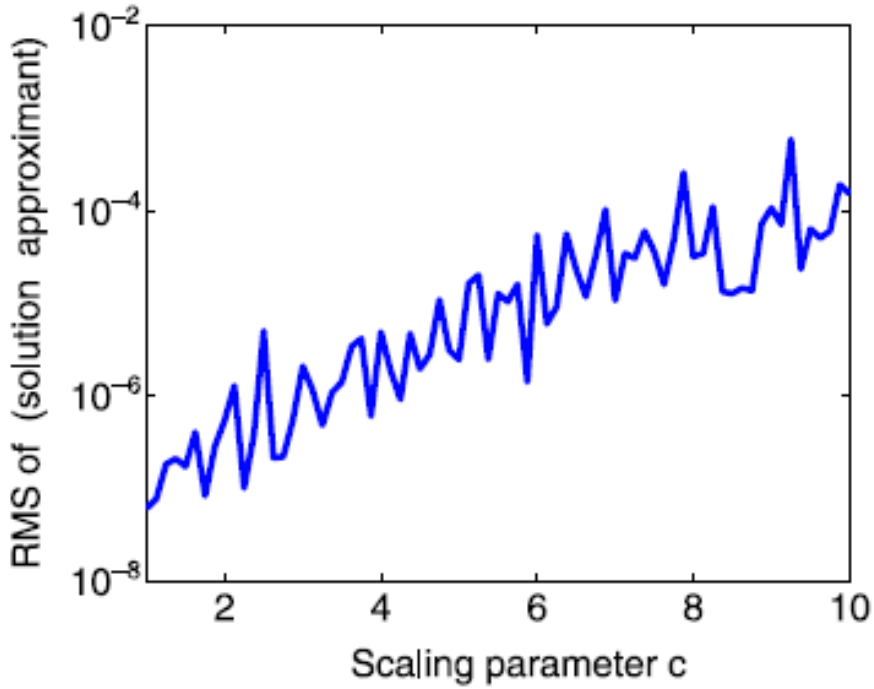
$$K(\boldsymbol{x}, \boldsymbol{z}) = \sqrt{1 + \frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{c^2}}$$

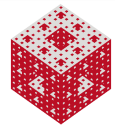
for

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

with $g(x, y) = \frac{1}{2}\log(x^2 - 4x + 8 + y^2 - 4y)$ and $f = \Delta g$.
Collocation at $N = N_{\text{int}} + N_{\text{bdy}}$ points picked from a set of 1116 test functional candidates.
$M = 47229$, so that trial centers $\mathcal{Z}$ come from M equally spaced points in $[-6, 6]$.
A greedy matrix-free algorithm was used to pick the points (see [LOS06] for details).


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1
Ω_1


10^{-2}
10^{-4}
10^{-6}
10^{-8}
RMS of (solution approximant)
2
4
6
8
10
Scaling parameter c



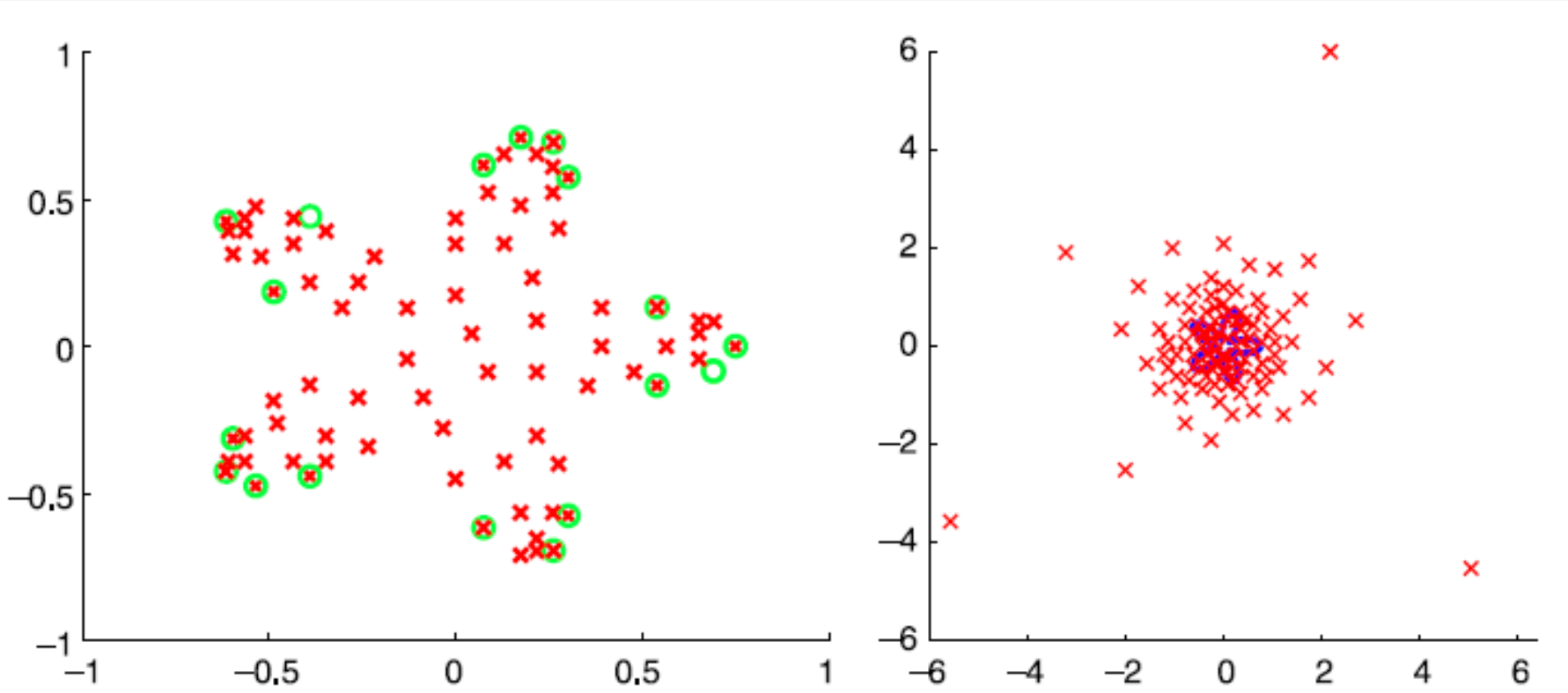


Fig. 3. Loci of points for $c = 1$. Left: chosen test functionals. Right: chosen trial centers.

Observations:

- about 110 points chosen,
- small c (i.e., large ε) results in many trial centers and test functionals and high accuracy,
- trial centers chosen pretty uniformly and close to the domain Ω.

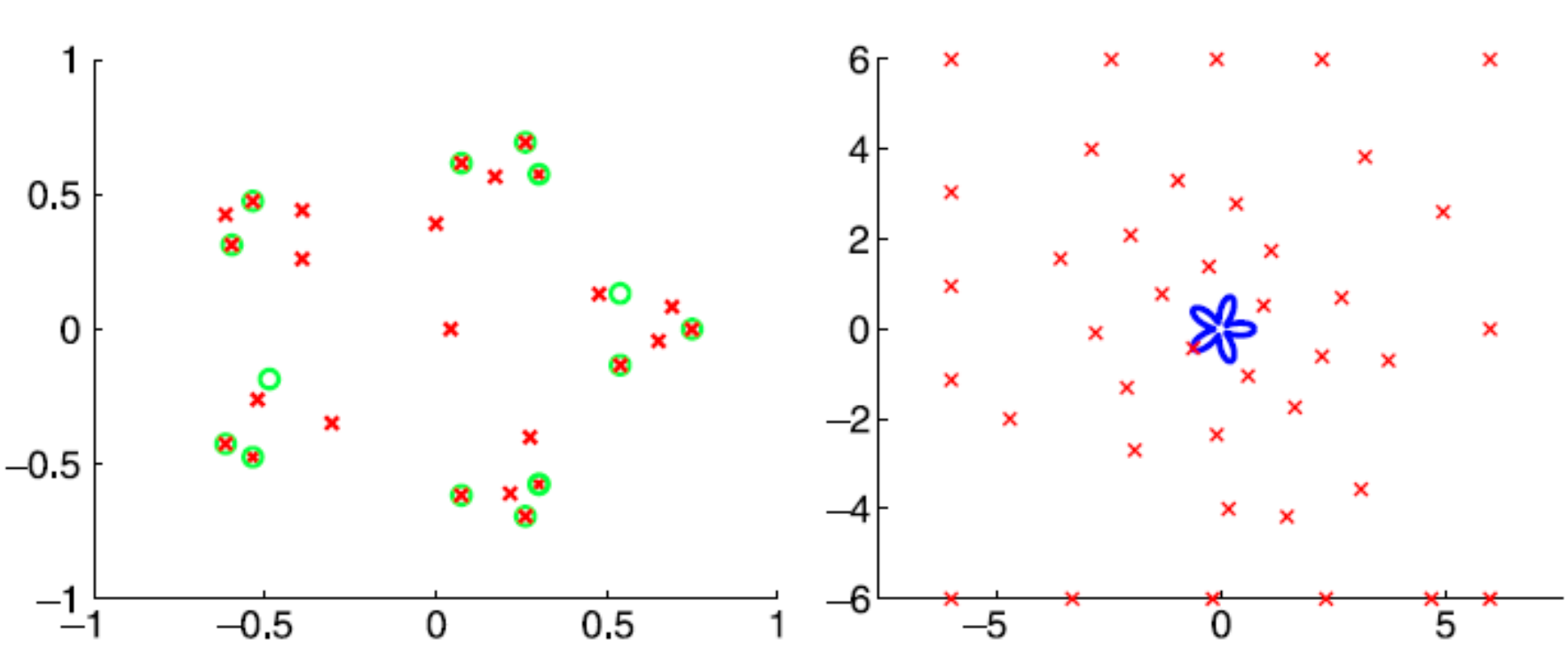


Fig. 4. Loci of points for $c=5$. Left: chosen test functionals. Right: chosen trial centers.

Observations:

- only 40 points chosen,
- large c (i.e., small ε) results in ill-conditioning, so use only a few centers and test functionals, but still get good accuracy,
- all but one trial center chosen outside the domain Ω.

Outline

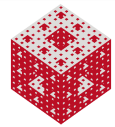
1 Kansa's Approach

2 MATLAB Implementation of Kansa's Method

3 Error Bounds for Non-Symmetric Collocation

4 Non-Symmetric Collocation with the Hilbert–Schmidt SVD

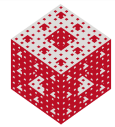
5 Toward Solving the Navier-Stokes Equations



Due to the known counterexamples from [HS01] for the non-symmetric method, a convergence analysis is still lacking for that method.

Due to the known counterexamples from [HS01] for the non-symmetric method, a convergence analysis is still lacking for that method.

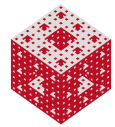
However, for an adaptive version of the non-symmetric method (see Ex. 4 above) Schaback has analyzed the convergence in [Sch05, Sch10] and the more "applied" paper [LS08].



Due to the known counterexamples from [HS01] for the non-symmetric method, a convergence analysis is still lacking for that method.

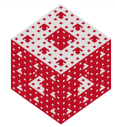
However, for an adaptive version of the non-symmetric method (see Ex. 4 above) Schaback has analyzed the convergence in [Sch05, Sch10] and the more "applied" paper [LS08].

They establish that the nonsymmetric collocation method, if carried out with smooth kernel-based trial functions and sufficiently many test functionals, converges at the same rate as interpolation of the solution. Thus, trial spaces formed using multiquadrics or Gaussians result in exponential convergence if the solution is analytic.



Optimal rates of convergence for general operator equations in strong form or weak form were recently proven in [Sch10] assuming

- the numerical method is based on residual minimization
- and using the sampling inequalities of [NWW05] mentioned in our general error discussion.



Optimal rates of convergence for general operator equations in strong form or weak form were recently proven in [Sch10] assuming

- the numerical method is based on residual minimization
- and using the sampling inequalities of [NWW05] mentioned in our general error discussion.

In [LLS09] the authors use arbitrary-precision arithmetic in Mathematica (100 digits) to show that the nonsymmetric collocation converges even faster than the interpolant to the solution. However, they used shape parameters from [HLC07], which were determined to be “optimal” for the PDE they used.

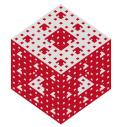
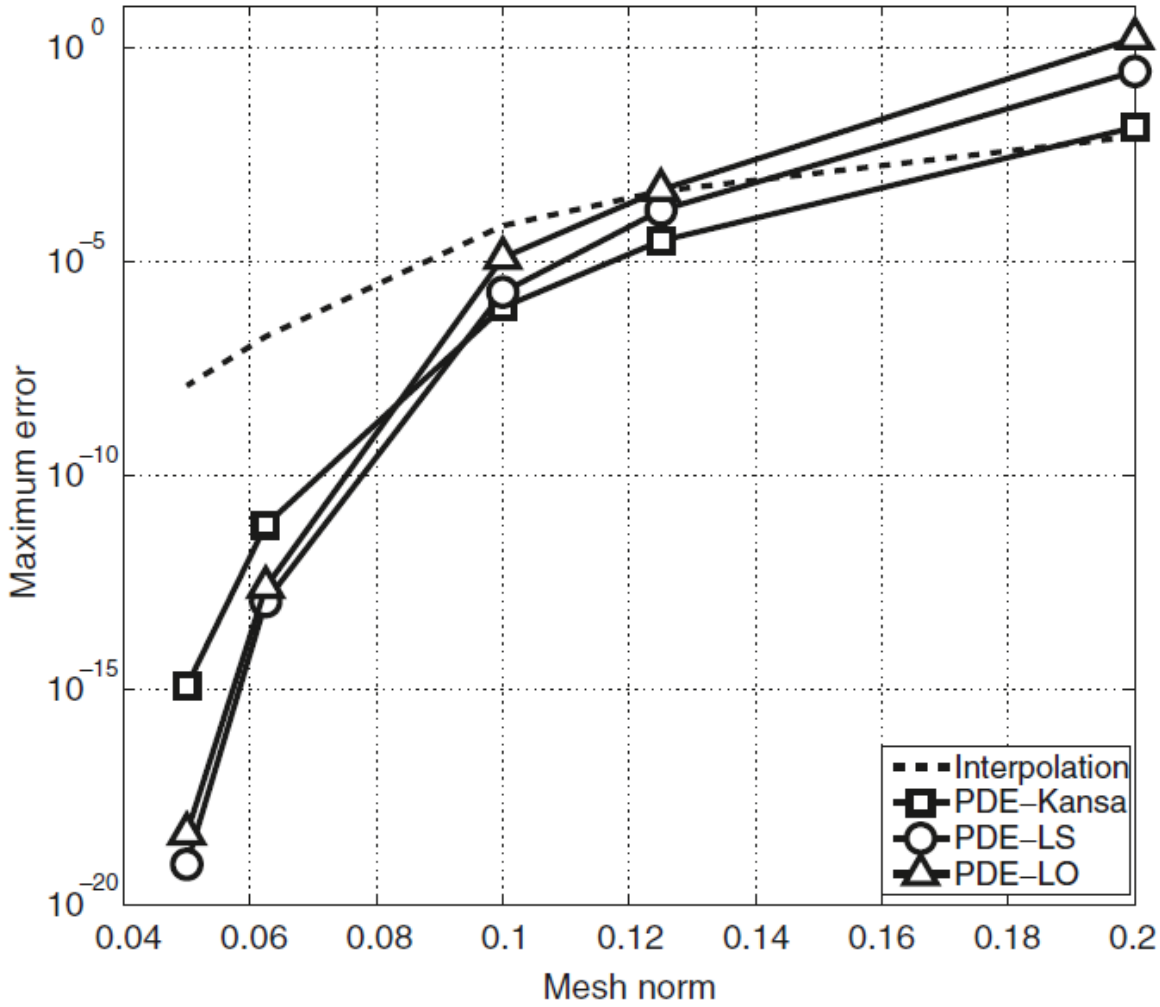


Fig. 4 Arbitrary–precision computations: Error profiles for various unsymmetric meshless formulations with the optimal shape parameter c^*



Outline

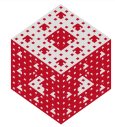
1 Kansa's Approach

2 MATLAB Implementation of Kansa's Method

3 Error Bounds for Non-Symmetric Collocation

4 Non-Symmetric Collocation with the Hilbert–Schmidt SVD

5 Toward Solving the Navier-Stokes Equations



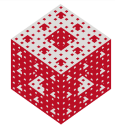
Transitioning to the Stable Basis (based on [McC13])

In order to overcome potential ill-conditioning, we can apply the stable basis. In the standard basis, our approximate solution is of the form

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{z}_j) = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c},$$

where

$$\boldsymbol{k}(\boldsymbol{x})^T = (K(\boldsymbol{x}, \boldsymbol{z}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{z}_N)).$$



Transitioning to the Stable Basis (based on [McC13])

In order to overcome potential ill-conditioning, we can apply the stable basis. In the standard basis, our approximate solution is of the form

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j K(\boldsymbol{x}, \boldsymbol{z}_j) = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c},$$

where

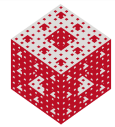
$$\boldsymbol{k}(\boldsymbol{x})^T = (K(\boldsymbol{x}, \boldsymbol{z}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{z}_N)).$$

To shift to the stable basis, our solution should take the form

$$\hat{u}_s(\boldsymbol{x}) = \sum_{j=1}^{N} b_j \psi_j(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})^T \boldsymbol{b},$$

where

$$\boldsymbol{\psi}(\boldsymbol{x})^T = (\psi_1(\boldsymbol{x}), \ldots, \psi_N(\boldsymbol{x})).$$



Transitioning to the Stable Basis

Recall the Hilbert–Schmidt SVD:

$$\begin{aligned} K &= \Psi \Lambda_1 \Phi_1^T \\ &= \Phi \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T. \end{aligned}$$

Transitioning to the Stable Basis

Recall the Hilbert–Schmidt SVD:

$$\begin{aligned} K &= \Psi \Lambda_1 \Phi_1^T \\ &= \Phi \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T. \end{aligned}$$

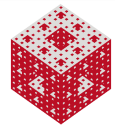
Each row of this matrix can be thought of as

$$\boldsymbol{k}(\boldsymbol{x})^T = \phi(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T$$

with the vector of eigenfunctions truncated to length $M > N$

$$\phi(\boldsymbol{x})^T = (\varphi_1(\boldsymbol{x}), \ldots, \varphi_M(\boldsymbol{x})).$$

Recall: Each row of K corresponds to a collocation point, and each column corresponds to a kernel center.



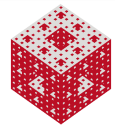
Derivatives in the Stable Basis

Solving BVPs via collocation requires differentiating the basis:

$$\mathcal{D}\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \mathcal{D}K(\boldsymbol{x}, \boldsymbol{z}_j) = \mathcal{D}\boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c}$$

$$= \left(\mathcal{D}\boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \right) \boldsymbol{c}$$

Here $\mathcal{D}$ is either $\mathcal{L}$ or $\mathcal{B}$ based on the collocation point location.
Note that $\mathcal{D}$ affects only the $\boldsymbol{x}$-variable of $K(\boldsymbol{x}, \boldsymbol{z}_j)$.



Derivatives in the Stable Basis

Solving BVPs via collocation requires differentiating the basis:

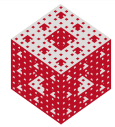
$$\mathcal{D}\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \mathcal{D}K(\boldsymbol{x}, \boldsymbol{z}_j) = \mathcal{D}\boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c}$$

$$= \left(\mathcal{D}\boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \right) \boldsymbol{c}$$

Here $\mathcal{D}$ is either $\mathcal{L}$ or $\mathcal{B}$ based on the collocation point location.
Note that $\mathcal{D}$ affects only the $\boldsymbol{x}$-variable of $K(\boldsymbol{x}, \boldsymbol{z}_j)$.
In order to create the stable basis we identify $\Lambda_1 \Phi_1^T \boldsymbol{c} = \boldsymbol{b}$ and

$$\boldsymbol{\psi}(\boldsymbol{x})^T = \boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix}$$

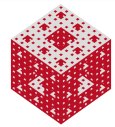
$$\mathcal{D}\hat{u}_s(\boldsymbol{x}) = \left(\mathcal{D}\boldsymbol{\phi}(\boldsymbol{x})^T \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \right) \boldsymbol{b}$$



Stable Basis Collocation

Our original collocation matrix can be converted to

$$\begin{pmatrix} K_{\mathcal{L}} \\ K_{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} \Phi_{\mathcal{L}} \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \\ \Phi_{\mathcal{B}} \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \end{pmatrix} = \begin{pmatrix} \Phi_{\mathcal{L}} \\ \Phi_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} I_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T.$$



Stable Basis Collocation

Our original collocation matrix can be converted to

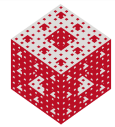
$$\begin{pmatrix} \mathrm{K}_{\mathcal{L}} \\ \mathrm{K}_{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} \Phi_{\mathcal{L}} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \\ \Phi_{\mathcal{B}} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \end{pmatrix} = \begin{pmatrix} \Phi_{\mathcal{L}} \\ \Phi_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T.$$

This changes our system to

$$\begin{pmatrix} \Phi_{\mathcal{L}} \\ \Phi_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \Lambda_1 \Phi_1^T \boldsymbol{c} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix},$$

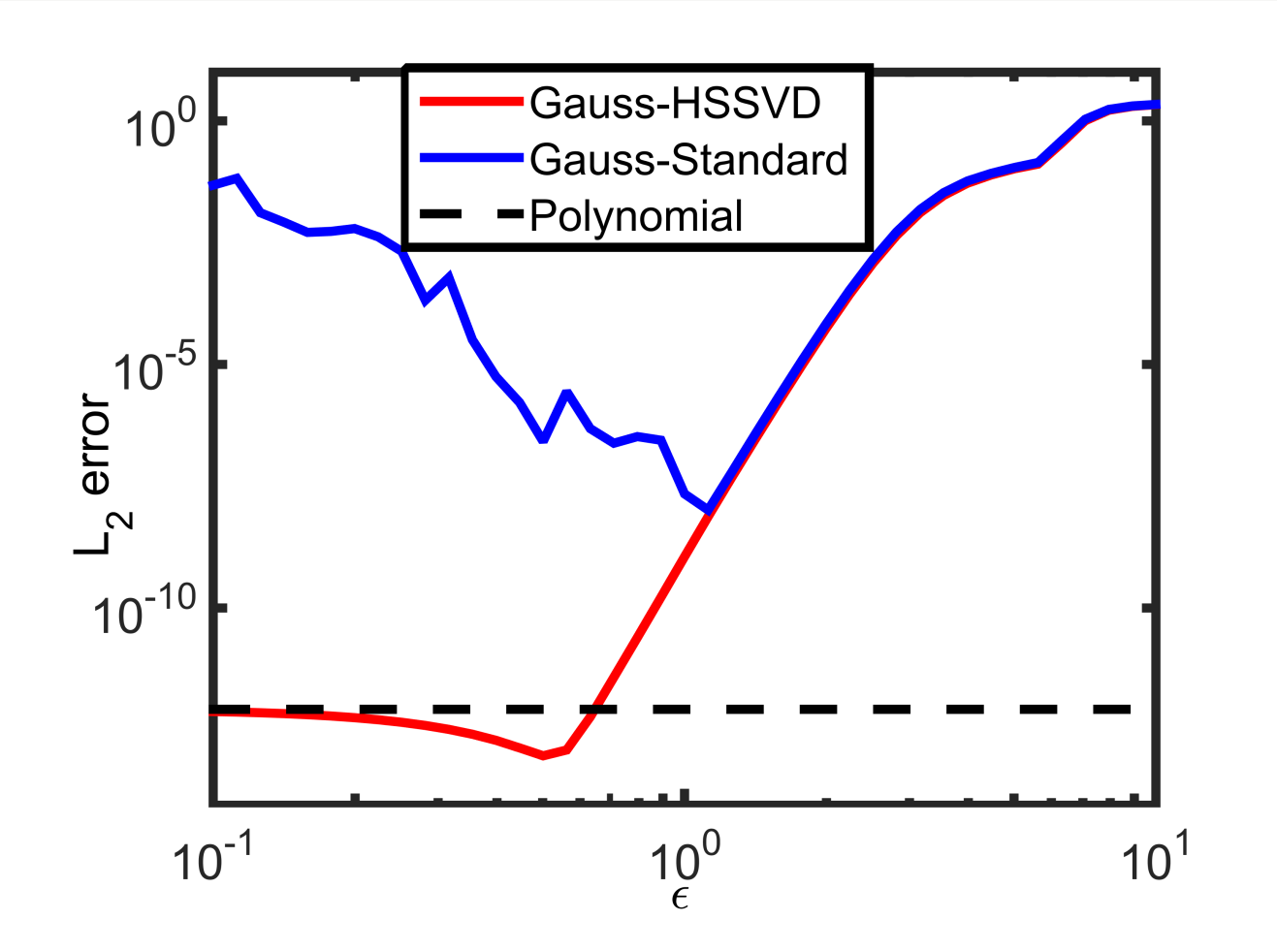
$$\begin{pmatrix} \Phi_{\mathcal{L}} \\ \Phi_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} \mathrm{I}_N \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{pmatrix} \boldsymbol{b} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix},$$

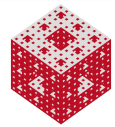
$$\begin{pmatrix} \Psi_{\mathcal{L}} \\ \Psi_{\mathcal{B}} \end{pmatrix} \boldsymbol{b} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix}.$$



We can now perform collocation stably (without inverting Λ_1).

Stable Basis Collocation





There is a clear benefit to using the stable Gaussian basis.

Polynomial Example (based on [Tre00])

```
N = 16;
NN = 200;
xx = pickpoints(-1,1,NN);
f = @(x) -sinh(x)./(1+cosh(x)).^2;
exact = @(x) sinh(x)./(1+cosh(x));

[D,t] = cheb(N);
D2 = D^2;
D2([1,end],:) = [1,zeros(1,N-1);zeros(1,N-1),1];
rhs = [exact(t(1));f(t(2:end-1));exact(t(end))];

u = D2\rhs;
pcoef = polyfit(t,u,N-1);
polyprediction = polyval(pcoef,xx);
err_Trefethen = errcompute(polyprediction,exact(xx))
```

Kernel Example (Direct)

```
N = 16; NN = 200;
x = pickpoints(-1,1,N,'cheb');
xx = pickpoints(-1,1,NN);
f = @(x) -sinh(x)./(1+cosh(x)).^2;
exact = @(x) sinh(x)./(1+cosh(x));

rbf = @(e,r) exp(-(e*r).^2);
d2rbf = @(e,r) 2*e^2*(2*(e*r).^2-1).*exp(-(e*r).^2);
ep = 1;

r = DistanceMatrix(x,x); D2 = d2rbf(ep,r); K = rbf(ep,r);
D2([1,end],:) = K([1,end],:);
rhs = [exact(x(1));f(x(2:end-1));exact(x(end))];

coef = D2\rhs;
reval = DistanceMatrix(xx,x); Keval = rbf(ep,reval);
err_Kansa = errcompute(Keval*coef,exact(xx));
```

Kernel Example (HS-SVD)

```
N = 16; NN = 200;
x = pickpoints(-1,1,N,'cheb');
xx = pickpoints(-1,1,NN);
f = @(x) -sinh(x)./(1+cosh(x)).^2;
exact = @(x) sinh(x)./(1+cosh(x));

ep = 1;

GQR = gqr_solveprep(0,x,ep);
phi_B = gqr_phi(GQR,x([1,end]));
phi_L = gqr_phi(GQR,x(2:end-1),2);
K = [phi_B;phi_L]*[eye(N);GQR.Rbar];
rhs = [exact(x([1,end]));f(x(2:end-1))];

GQR.coef = K\rhs;
err_GQR = errcompute(gqr_eval(GQR,xx),exact(xx));
```

Outline

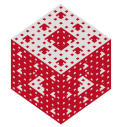
1 Kansa's Approach

2 MATLAB Implementation of Kansa's Method

3 Error Bounds for Non-Symmetric Collocation

4 Non-Symmetric Collocation with the Hilbert–Schmidt SVD

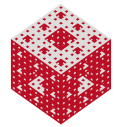
5 Toward Solving the Navier-Stokes Equations



Toward Solving the Navier-Stokes Equations

In order to be able tackle more sophisticated PDEs we need to be able to better handle a number of issues:

- computational efficiency for large problems (e.g., local representations)
- Eulerian and semi-Lagrangian solvers for fluid flow problems
- adaptivity
- special properties of solution (divergence-free, rotation-free, etc.)

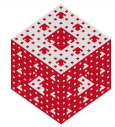


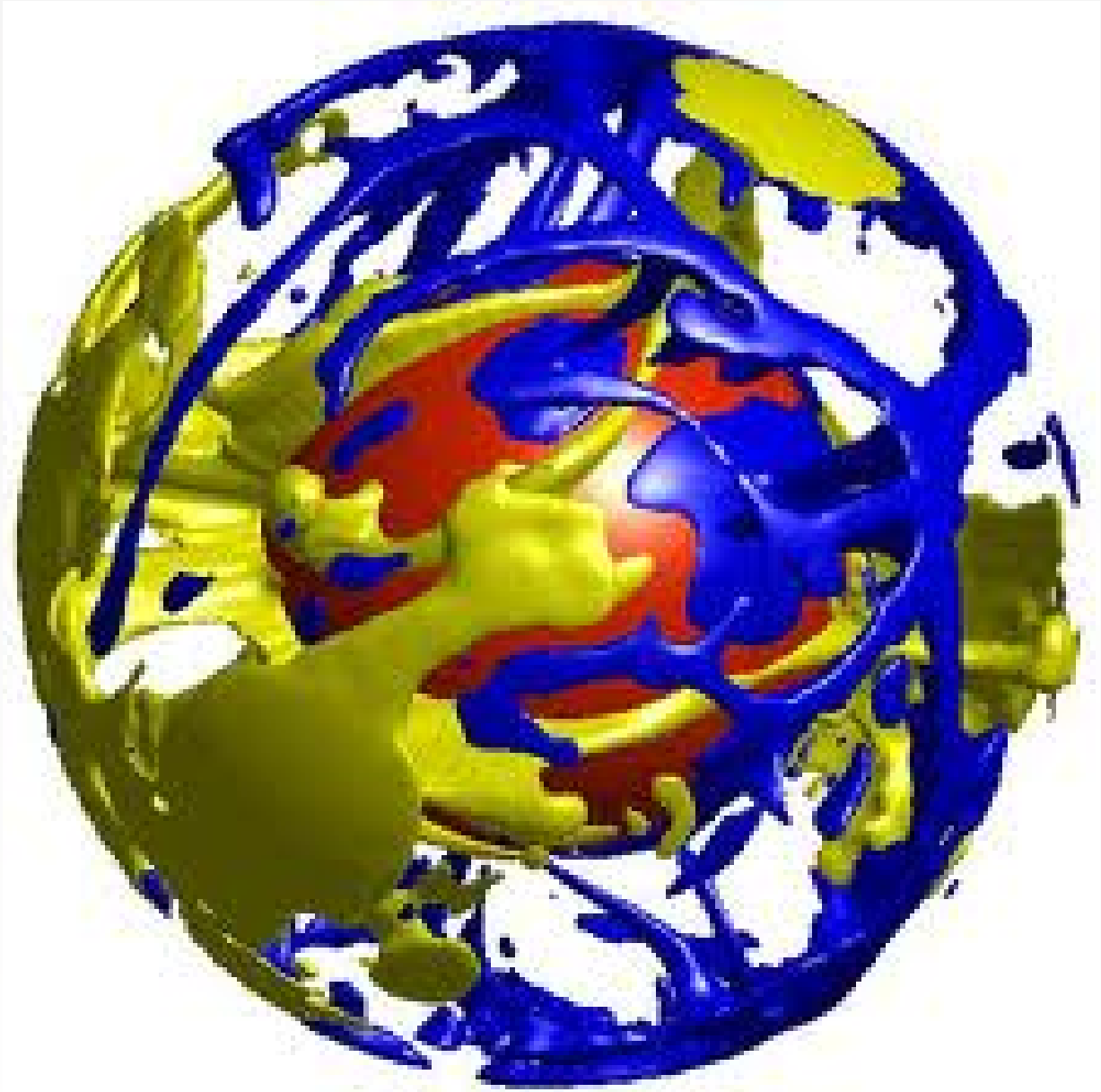
Toward Solving the Navier-Stokes Equations

In order to be able tackle more sophisticated PDEs we need to be able to better handle a number of issues:

- computational efficiency for large problems (e.g., local representations)
- Eulerian and semi-Lagrangian solvers for fluid flow problems
- adaptivity
- special properties of solution (divergence-free, rotation-free, etc.)

In the remaining few slides we present a few initial investigations in this direction performed by a former student.
Other research with significant progress is under way by, e.g., [FW07, FW09a, FP08, Fus08b, FNWW09, Wen09, WFY10a].





[WFY10b] to illustrate [WFY10a]

Ex.1: Lid-driven cavity flow

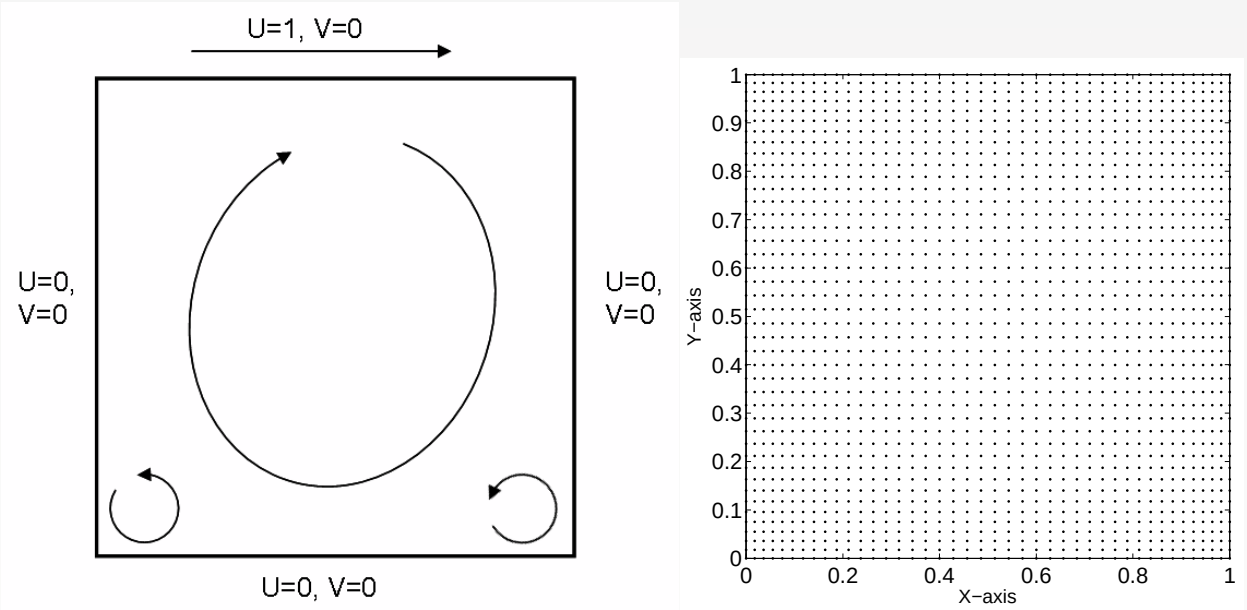
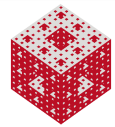


Figure: (a) Schematic of the driven cavity problem and (b) Distribution of the RBF centers.

Features in the flow depend on the Reynolds number

$$Re = UL/\nu, \qquad \text{(for us } L = U = 1\text{)}$$



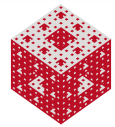
We now describe an

- Eulerian (observe the flow from a fixed outside point)
- and semi-Lagrangian (observe the flow from a moving inside point)

method to solve the Navier-Stokes equations.

We solve the Navier-Stokes equations in the primitive-variables form (directly measurable variables – as opposed to conserved variables such as momentum or mass) for the velocity components and the pressure at each time step.

For this set of experiments spatial derivatives are approximated using a local RBF method.



Since the pressure term does not have an explicit evolution equation, we use a fractional-step method to compute the velocity and pressure fields.

Using a Crank-Nicholson scheme in the Eulerian framework we need to solve at each time step

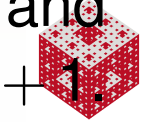
$$u_i^\star = u_i^n - \Delta t \left(u_j^n \frac{\partial u_i^n}{\partial x_j} \right) \tag{8}$$

$$u_i^{\star\star} - \left(\frac{\Delta t}{Re} \right) \nabla^2 u_i^{\star\star} = u_i^\star \tag{9}$$

$$\frac{\partial^2 p}{\partial x_j \partial x_j} = \frac{1}{\Delta t} \frac{\partial u_i^{\star\star}}{\partial x_i} \tag{10}$$

$$u_i^{n+1} = u_i^{\star\star} - \Delta t \frac{\partial p}{\partial x_i} \tag{11}$$

The subscript i in u_i refers to the i^{th} component of the velocity field and the superscript $n+1$ in u^{n+1} refers to the velocity at the time step $n+1$.



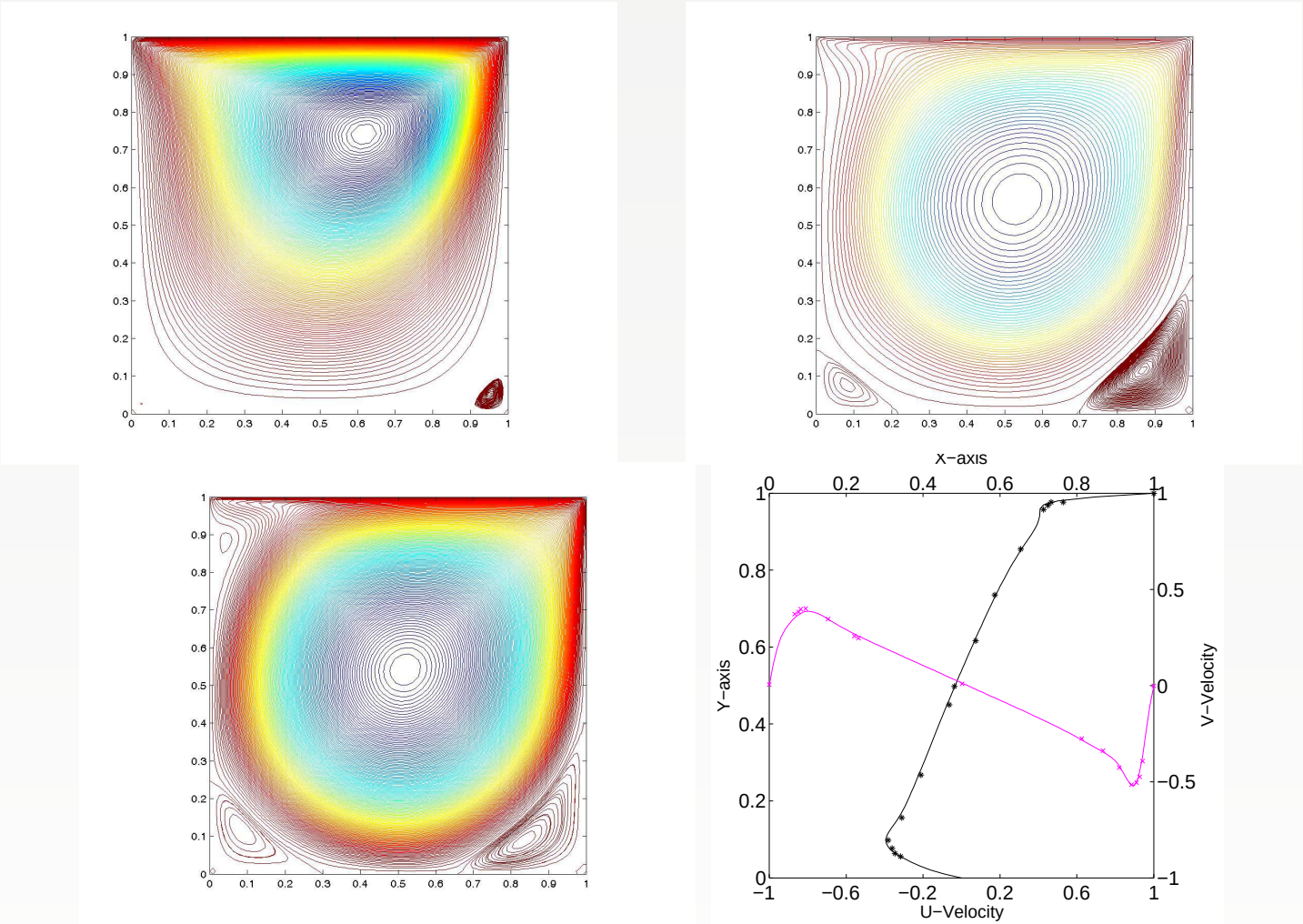


Figure: Plot of streamfunction for (a) Re=100 (1681 nodes) (b) Re=1000 (5041 nodes) (c) Re=3200 (8281 nodes) (d) Comparison of centerline velocities for Re=3200 (solid lines) with [GGS82] (symbols). Simulation performed in the Eulerian framework – time-dependent problem run to steady-state.

For the semi-Lagrangian approach we compute the convective term in (8) by tracking the evolution of particles in the domain and recording the velocities associated with the particles.
Using a second-order Adams-Bashforth method with a Crank-Nicholson scheme we need to solve

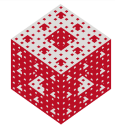
$$\frac{\hat{u}_i - 2u_i(\boldsymbol{x}_d^n) + \frac{1}{2}u_i(\boldsymbol{x}_d^{n-1})}{\Delta t} = 0 \tag{12}$$

$$\frac{\partial^2 p}{\partial x_j \partial x_j} = \frac{1}{\Delta t}\frac{\partial \hat{u}_i}{\partial x_i} \tag{13}$$

$$\hat{\hat{u}}_i = \hat{u}_i - \Delta t \frac{\partial p}{\partial x_i} \tag{14}$$

$$\frac{3}{2}u_i^{n+1} - \left(\frac{\Delta t}{Re}\right)\frac{\partial^2 u_j^{n+1}}{\partial x_j \partial x_j} = \hat{\hat{u}}_i \tag{15}$$

The term $u_i(\boldsymbol{x}_d^n)$ in (12) represents the i^{th} velocity component at the departure locations $\boldsymbol{x}_d$ at time t^n.



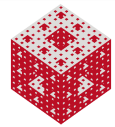
To solve (12) we need to compute the velocities at different spatial locations: requires an accurate interpolation scheme
According to [XK01] any semi-Lagrangian method has order of accuracy

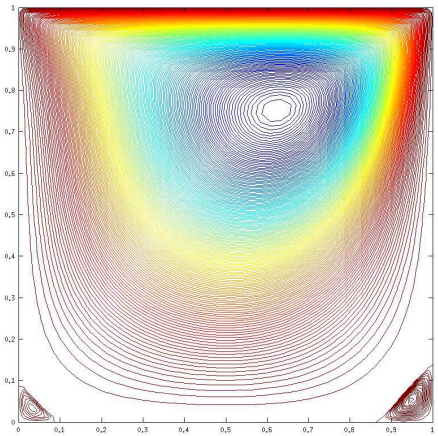
$$\mathcal{O}\left((\Delta t)^{\alpha} + \frac{(\Delta x)^{\beta+1}}{\Delta t}\right),$$

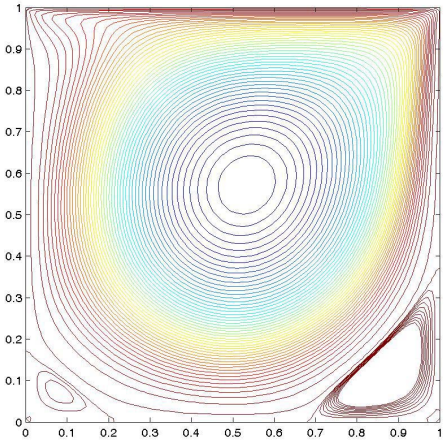
where α is the accuracy of the temporal scheme and β is the order of the interpolation scheme.

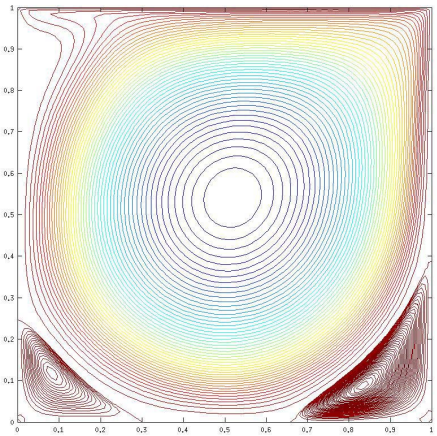
Remark

The equations being solved for the Eulerian and the semi-Lagrangian method are virtually identical. The only difference lies with the convective term.









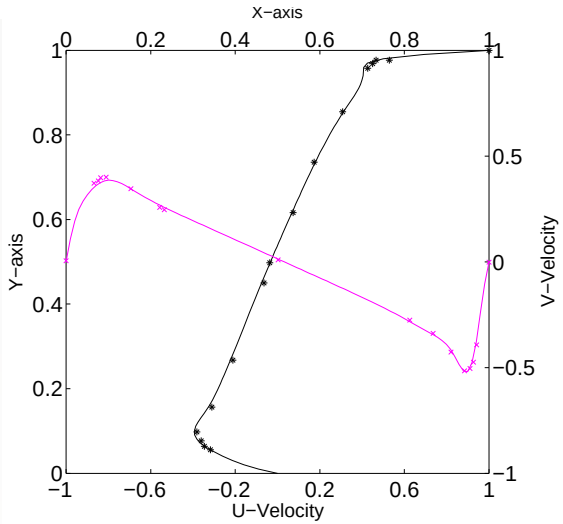
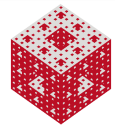


Figure: Plot of streamfunction for (a) Re=100 (1681 nodes) (b) Re=1000 (5041 nodes) (c) Re=3200 (8281 nodes) (d) Comparison of centerline velocities for Re=3200 (solid lines) with [GGS82] (symbols). Simulation performed in the semi-Lagrangian framework – time-dependent problem run to steady-state.

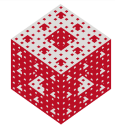


Effect of the interpolation scheme in the semi-Lagrangian method:

- using a small local support (on the order of 10–12 grid points) for the interpolation led to numerical diffusion. The numerical method was stable, but the solution invariably resembled Stokes flow – even for a high Reynolds number
- The solutions in the figure used supports with about 40 points. With more points the computational cost increases.

Comparison of Eulerian and semi-Lagrangian schemes:

- The semi-Lagrangian scheme requires more operations per time step, but we can take larger time steps and so the cost is roughly the same.
- The semi-Lagrangian scheme is highly parallelizable, so we can performing numerical simulations on high-performance computer clusters.



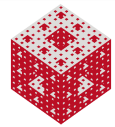
Adaptivity

We know that Gaussian RBF interpolation is spectrally accurate.

We now explore the effect of adaptivity.

Two major advantages of adaptivity:

- adaptivity will let us efficiently resolve regions of high vorticity without having to uniformly increase the number of nodes
- by correctly implementing adaptivity one can produce very accurate results [DH07]



Adaptivity I

Sub-sampling algorithm similar to [DH07]:

- Select initial distribution
- Select test-points such that they lie mid-point between two centers and compute residual at test-points
- If a group of test-centers surrounding an "active" center falls below a certain threshold, eliminate that center.
- If a given test-point is above a certain threshold, promote that test-point to a center.
- Repeat recursively until stabilized.

Remark

The algorithm is extremely fast and the adaptivity converges within 10-15 iterations. In higher-dimensional cases, locations of test-points are determined with the help of d-dimensional Voronoi cells.

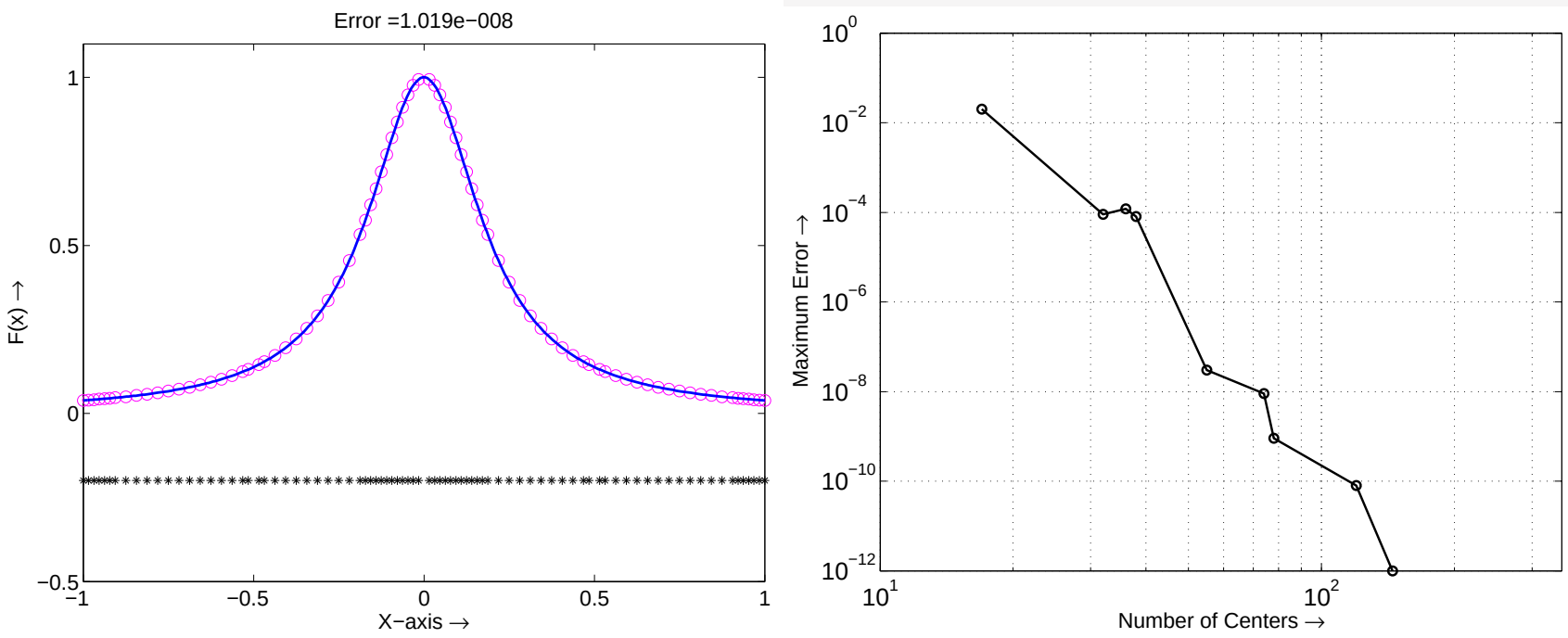
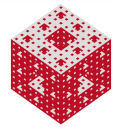


Figure: (a) RBF approximation for Runge function. (b) Spectral convergence of adaptive RBF scheme.

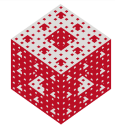


Ex.2: Burgers' equation

We now consider Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < 1,$$
$$u(0,t) = u(1,t) = 0,$$
$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x).$$

- Also used in [DH07].
- Discretize using a method of lines and do time-stepping with an explicit Runge-Kutta method.
- Apply the spatial adaptive algorithm at each time step.



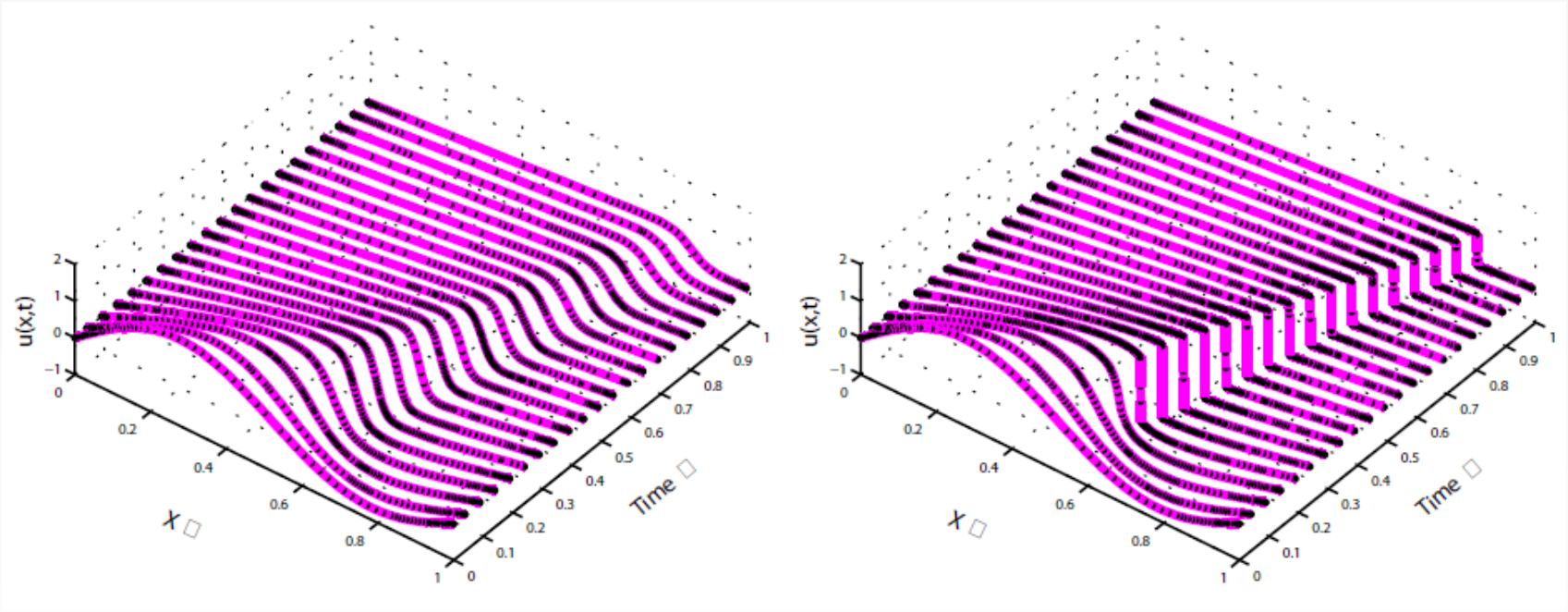
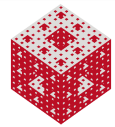


Figure: Burgers' solution using adaptive RBFs for (a) $\nu = 0.01$ (b) $\nu = 0.00001$.



Observations concerning incompressible flows

- Adaptivity will enable us to efficiently resolve regions of high vorticity without having to uniformly increase the number of nodes in our numerical simulations.
- As a consequence, the overall scheme will be able to produce very accurate results.
- Currently, the ALE method is the only method that employs an adaptive grid. We do not directly deal with grids but have to choose where to best place the interpolation nodes to achieve high accuracy.
- We determine these locations by checking at each time step if the residuals are below a certain tolerance (see also [Sar05, DH07, SW00]).
- Interesting to explore whether it is more efficient to develop cost-functions for adaptivity that are dependent on physical phenomena on mathematical properties such as the condition number of the interpolation matrix.

Divergence-free RBFs

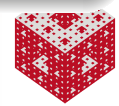
For the semi-Lagrangian scheme we need to perform interpolation to compute the velocities at arbitrary nodal locations.

On the other hand, the conservation of mass equations states that the flow is divergence-free.

Using matrix-valued RBFs, we can directly build this property into our basis.

Remark

Divergence-free interpolation using RBFs has been studied theoretically by [NW94, Low05, Fus08b, Fus08a, FNWW09, FW09b].

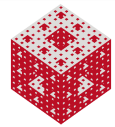


Remark

To our knowledge fluid flow problems have never been considered in the divergence-free framework.
The papers [Wen09, SW11, FW13] on

- *stationary Stokes flow*
- *Darcy flow in porous media*
- *reaction and diffusion of chemicals on biological cells or membranes, and pattern formations in biology*

come closest.



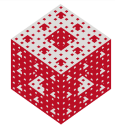
Any divergence-free vector field $\boldsymbol{u}$ can be written as $\boldsymbol{u} = \nabla \times \boldsymbol{w}$, where $\boldsymbol{w}$ is a (divergence-free) vector potential.
If we approximate $\boldsymbol{w}$ in terms of vector-kernels

$$\boldsymbol{w} = \nabla \times \sum_{j=1}^{N} \boldsymbol{c}_j K(\cdot, \boldsymbol{z}_j),$$

where $\boldsymbol{c}_j = [c_{j,x}, c_{j,y}, c_{j,z}]^T$, then

$$\begin{aligned} \boldsymbol{u} &= \nabla \times \nabla \times \sum_{j=1}^{N} \boldsymbol{c}_j K(\cdot, \boldsymbol{z}_j) \\ &= \left(-\Delta \mathrm{I} + \nabla \nabla^T\right) \sum_{j=1}^{N} \boldsymbol{c}_j K(\cdot, \boldsymbol{z}_j), \end{aligned}$$

where Δ is the Laplacian, I is an identity matrix, and ∇ is the gradient operator.

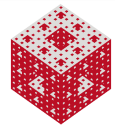


$\longrightarrow$ Divergence-free matrix-valued kernels

The use of divergence-free interpolation essentially reduces the Navier-Stokes equations to the solution of a Laplace problem for pressure.

With solid boundaries everywhere (as for the driven-cavity problem), no pressure correction is needed.

Potential for significant savings in computational cost.



Ex.3: Hill's spherical vortex

We use adaptive divergence-free RBFs to approximate the velocity field.

We use the same adaptive sub-sampling algorithm as above.

Velocities (in 2D) are approximated as

$$\boldsymbol{u} = -\sum_{j=1}^{N} c_j \frac{\partial^2 K(\cdot, \boldsymbol{z}_j)}{\partial y^2} + \sum_{\ell=1}^{N} d_\ell \frac{\partial^2 K(\cdot, \boldsymbol{z}_\ell)}{\partial x \partial y} \tag{16}$$

$$\boldsymbol{v} = \sum_{j=1}^{N} c_j \frac{\partial^2 K(\cdot, \boldsymbol{z}_j)}{\partial x \partial y} - \sum_{\ell=1}^{N} d_\ell \frac{\partial^2 K(\cdot, \boldsymbol{z}_\ell)}{\partial x^2} \tag{17}$$

The result is a system of $2N$ equations rather than N.

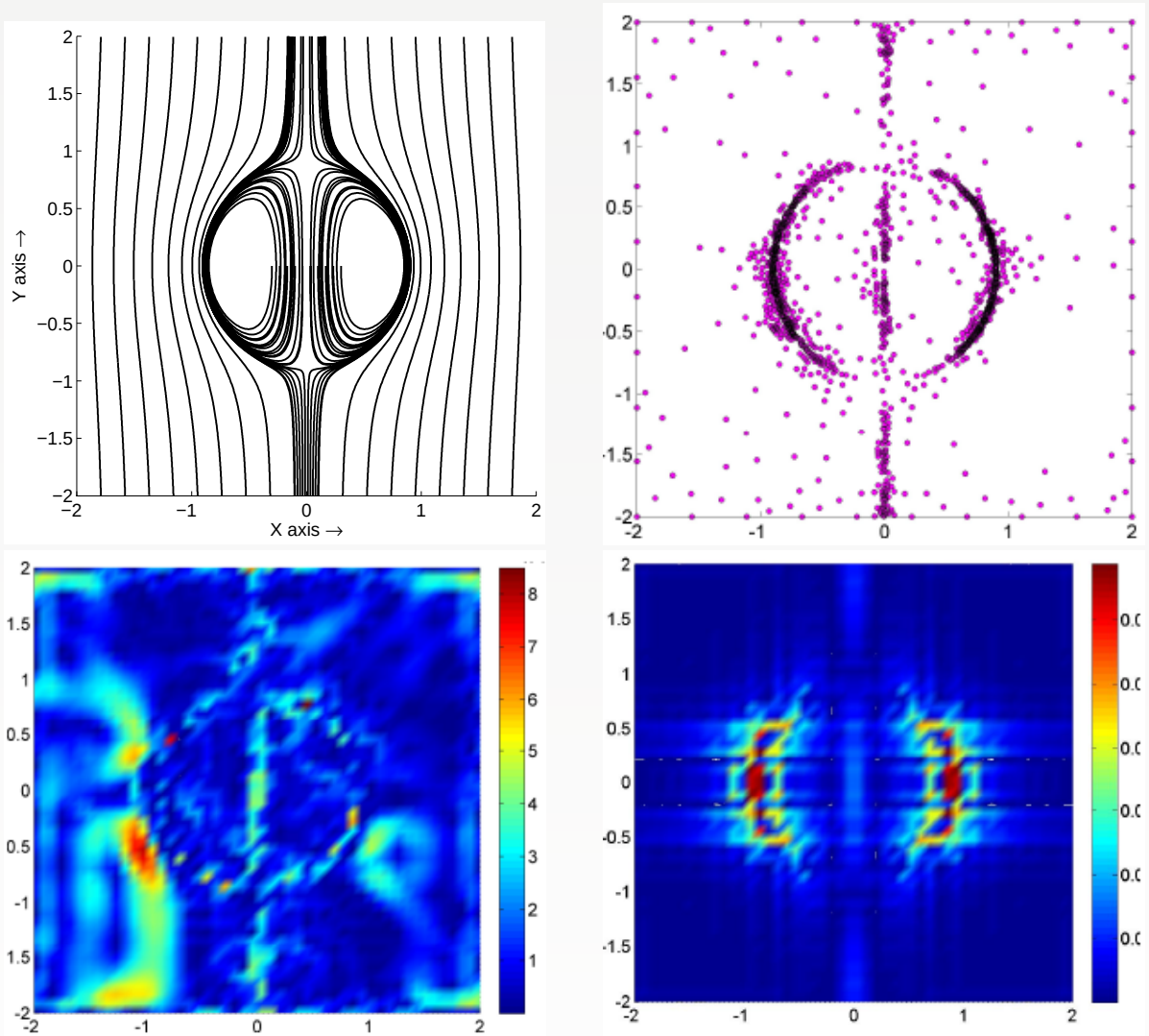
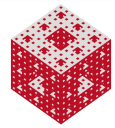


Figure: (a) Streamlines for Hill's vortex. (b) Location of RBF centers ($N = 900$). (c) Absolute error for divergence-free adaptive RBF interpolation, $\mathcal{O}(10^{-4})$. (d) Absolute error for divergence-free RBF interpolation, $\mathcal{O}(10^{-2})$.

So far, most of the discussion has focussed on elliptic PDEs.

So far, most of the discussion has focussed on elliptic PDEs.

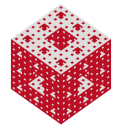
An important part of the theoretical foundation for time-dependent PDEs is the convergence and stability of the kernel-based approach.



So far, most of the discussion has focussed on elliptic PDEs.

An important part of the theoretical foundation for time-dependent PDEs is the convergence and stability of the kernel-based approach.

The manuscript [HS10] provides the equivalent of a CFL condition for the solution of the standard heat equation with a method of lines approach that uses positive definite kernels for the spatial discretization and an Euler method in time.

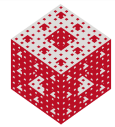


So far, most of the discussion has focussed on elliptic PDEs.

An important part of the theoretical foundation for time-dependent PDEs is the convergence and stability of the kernel-based approach.

The manuscript [HS10] provides the equivalent of a CFL condition for the solution of the standard heat equation with a method of lines approach that uses positive definite kernels for the spatial discretization and an Euler method in time.

The kernel-based CFL condition mirrors that for standard finite-difference methods and requires that the time step Δt satisfies $\Delta t \leq C(\Delta x)^2$, where C is some positive constant and Δx denotes the spacing of equidistant spatial collocation points.



References I

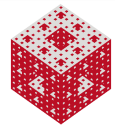
[BLRS14] Mira Bozzini, Licia Lenarduzzi, Milvia Rossini, and Robert Schaback, *Interpolation with variably scaled kernels*, IMA Journal of Numerical Analysis (2014).

[BMK12] Victor Bayona, Miguel Moscoso, and Manuel Kindelan, *Optimal variable shape parameter for multiquadric based RBF-FD method*, Journal of Computational Physics **231** (2012), no. 6, 2466–2481.

[DH07] T. A. Driscoll and A. Heryudono, *Adaptive residual subsampling methods for radial basis function interpolation and collocation problems*, Comput. Math. Appl. **53** (2007), no. 6, 927–939.

[Dub92] M. R. Dubal, *Construction of three-dimensional black-hole initial data via multiquadrics*, Phys. Rev. D, **45** (1992), 1178–1187.

[Dub94] _____, *Domain decomposition and local refinement for multiquadric approximations. I: Second-order equations in one-dimension*, J. Appl. Sc. Comp. **1** (1994), 146–171.



References II

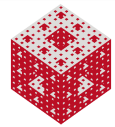
[Fas97] G. E. Fasshauer, *Solving partial differential equations by collocation with radial basis functions*, Surface Fitting and Multiresolution Methods (C. Rabut A. Le Méhauté and L. L. Schumaker, eds.), University Press, 1997, pp. 131–138.

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

[FNWW09] E. Fuselier, F. Narcowich, J. Ward, and G. Wright, *Error and stability estimates for surface-divergence free RBF interpolants on the sphere*, Math. Comp. **78** (2009), no. 268, 2157–2186.

[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.

[Fus08a] E. Fuselier, *Improved stability estimates and a characterization of the native space for matrix-valued RBFs*, Adv. Comput. Math. **29** (2008), no. 3, 311–313.



References III

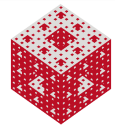
[Fus08b] _____, *Sobolev-type approximation rates for divergence-free and curl-free RBF interpolants*, Math. Comp. **77** (2008), no. 263, 1407–1423.

[FW07] Natasha Flyer and Grady B. Wright, *Transport schemes on a sphere using radial basis functions*, Journal of Computational Physics **226** (2007), no. 1, 1059–1084.

[FW09a] _____, *A radial basis function method for the shallow water equations on a sphere*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science **465** (2009), no. 2106, 1949 –1976.

[FW09b] Edward J. Fuselier and Grady B. Wright, *Stability and error estimates for vector field interpolation and decomposition on the sphere with RBFs*, SIAM Journal on Numerical Analysis **47** (2009), no. 5, 3213–3239.

[FW13] _____, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, Journal of Scientific Computing **56** (2013), no. 3, 535–565.



References IV

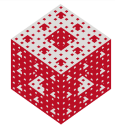
[GCK96] M. A. Golberg, C. S. Chen, and S. R. Karur, *Improved multiquadric approximation for partial differential equations*, Eng. Anal. with Bound. Elem. **18** (1996), 9–17.

[GGS82] U Ghia, K. N Ghia, and C. T Shin, *High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method*, Journal of Computational Physics **48** (1982), no. 3, 387–411.

[HLC07] C.-S. Huang, C.-F. Lee, and A.H.-D. Cheng, *Error estimate, optimal shape factor, and high precision computation of multiquadric collocation method*, Engineering Analysis with Boundary Elements **31** (2007), no. 7, 614–623.

[HS01] Y. C. Hon and R. Schaback, *On unsymmetric collocation by radial basis functions*, Applied Mathematics and Computation **119** (2001), no. 2–3, 177–186.

[HS10] Y. C. Hon and R. Schaback, *Meshless kernel techniques for the heat equation*, 2010.



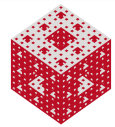
References V

[Kan90] E. J. Kansa, *Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics–II solutions to parabolic, hyperbolic and elliptic partial differential equations*, Computers & Mathematics with Applications **19** (1990), no. 8-9, 147–161.

[Kan92] E. J. Kansa, *A strictly conservative spatial approximation scheme for the governing engineering and physics equations over irregular regions and inhomogeneous scattered nodes*, Comput. Math. Appl. **24** (1992), 169–190.

[KC92] E. J. Kansa and R. E. Carlson, *Improved accuracy of multiquadric interpolation using variable shape parameters*, Comput. Math. Appl. **24** (1992), 99–120.

[LCC03] J. Li, A. H.-D. Cheng, and C-S. Chen, *On the efficiency and exponential convergence of multiquadric collocation method compared to finite element method*, Engineering Analysis with Boundary Elements **27** (2003), no. 3, 251–257.



References VI

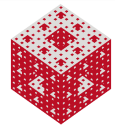
[LLS09] Cheng-Feng Lee, Leevan Ling, and Robert Schaback, *On convergent numerical algorithms for unsymmetric collocation*, Advances in Computational Mathematics **30** (2009), no. 4, 339–354.

[LOS06] Leevan Ling, Roland Opfer, and Robert Schaback, *Results on meshless collocation techniques*, Engineering Analysis with Boundary Elements **30** (2006), no. 4, 247–253.

[Low05] S. Lowitzsch, *Matrix-valued radial basis functions: stability estimates and applications*, Advances in Computational Mathematics **23** (2005), no. 3, 299–315.

[LS08] Leevan Ling and Robert Schaback, *Stable and convergent unsymmetric meshless collocation methods*, SIAM Journal on Numerical Analysis **46** (2008), no. 3, 1097–1115.

[McC13] Michael McCourt, *Using Gaussian eigenfunctions to solve boundary value problems*, Advances in Applied Mathematics and Mechanics **5** (2013), 569–594.



References VII

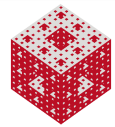
[MK94] G. J. Moridis and E. J. Kansa, *The Laplace transform multiquadric method: A highly accurate scheme for the numerical solution of linear partial differential equations*, J. Appl. Sc. Comp. **1** (1994), 375–407.

[NW94] F. J. Narcowich and J. D. Ward, *Generalized Hermite interpolation via matrix-valued conditionally positive definite functions*, Mathematics of Computation **63** (1994), no. 208, 661–687.

[NWW05] F. J. Narcowich, J. D. Ward, and H. Wendland, *Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting*, Math. Comp. **74** (2005), 743–763.

[Sar05] Scott A. Sarra, *Adaptive radial basis function methods for time dependent partial differential equations*, Applied Numerical Mathematics **54** (2005), no. 1, 79 – 94.

[Sch05] Robert Schaback, *Multivariate interpolation by polynomials and radial basis functions*, Constructive Approximation **21** (2005), 293–317.



References VIII

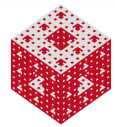
[Sch10] R. Schaback, *Unsymmetric meshless methods for operator equations*, Numerische Mathematik **114** (2010), no. 4, 629–651.

[SW00] R. Schaback and H. Wendland, *Adaptive greedy techniques for approximate solution of large RBF systems*, Numer. Algorithms **24** (2000), 239–254.

[SW11] D. Schräder and H. Wendland, *A high-order, analytically divergence-free discretization method for Darcy's problem*, Math. Comp. **80** (2011), 263–277.

[Tre00] Lloyd N. Trefethen, *Spectral Methods in MATLAB*, Software, Environments, Tools, SIAM: Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[Wen09] Holger Wendland, *Divergence-free kernel methods for approximating the Stokes problem*, SIAM Journal on Numerical Analysis **47** (2009), no. 4, 3158–3179.



References IX

[WFY10a] G. B. Wright, N. Flyer, and D. A. Yuen, *A hybrid radial basis function-pseudospectral method for thermal convection in a 3-D spherical shell*, Geochem. Geophys. Geosyst. **11** (2010), Q07003.

[WFY10b] _____, *Mantle convection simulation with a hybrid radial basis function/chebyshev pseudospectral method*, http://www.youtube.com/watch?v=-kDb0HlDsIM, 2010.

[WKL06] J. Wertz, E. J. Kansa, and L. Ling, *The role of the multiquadric shape parameters in solving elliptic partial differential equations*, Comput. Math. Appl. **51** (2006), no. 8, 1335–1348.

[XK01] Dongbin Xiu and George Em Karniadakis, *A semi-Lagrangian high-order method for Navier–Stokes equations*, Journal of Computational Physics **172** (2001), no. 2, 658–684.