Welcome from the Department of Mathematics at the University of Southern Mississippi
Local Radial Basis Function Methods for Solving PDEs

Guangming Yao

The University of Southern Mississippi

June 22, 2010
Abstract

The purpose of this dissertation is to develop two new localized meshless methods, the localized method of approximate particular solutions (LMAPS) and the explicit localized method of approximate particular solutions (ELMAPS), for solving a variety partial differential equations (PDEs).

In constructing the approximating solution to PDEs using local methods, the only additional information that is needed consists of the local nodes that fall within the domain of influence of the data, thus the proposed methods are suitable for large-scale problems. Since the methods are totally mesh free, they can be used for irregularly shaped domains.
Problem Introduction.

Theoretical Background:

(i) LMAPS: The Localized Method of Approximate Particular Solutions;

(ii) ELMAPS: The Explicit Localized Method of Approximate Particular Solutions.

Note LMAPS is an implicit scheme.

Examination of the Performance of the Methods.

Conclusions.
In this dissertation, we consider the following PDEs:

- **Elliptic** PDEs, e.g., the Poisson, Helmholtz equations, ...
- **Parabolic** PDE, e.g., the diffusion equation, ...
- **PDEs with variable coefficients**, e.g., Poisson & Helmholtz type equations, ...

with

- Dirichlet or Neumann boundary conditions;
- Irregular domains such as extremely long rectangular;
- High boundary gradients.
Mesh and Meshless

Advantages of meshless methods:

1. Ease of learning, ease of coding;
2. Complex domain geometries;
3. High dimensional problems;
4. No meshing or integrations needed.
Radial Basis Functions (RBFs)

- A **radial basis function** is a real-valued function which satisfies

\[
\phi(x, c) = \phi(\|x - c\|).
\]

Notation \(\phi(r)\) can be used to represent RBF, where \(r = \|x - c\|\).

- RBF types:
  - Gaussian \(\phi(r) = \exp(-cr^2), \quad c > 0\).
  - Polyharmonic spline \(\phi(r) = r^k, \quad k = 1, 3, 5, \ldots \phi(r) = r^k \ln r, \quad k = 2, 4, 6, \ldots\)
  - Thin plate spline (TPS) \(\phi(r) = r^2 \ln r\).
  - Multiquadric (MQ) \(\phi(r) = \sqrt{r^2 + c^2}, \quad c > 0, \quad c \) is called the shape parameter
  - Inverse multiquadric (IMQ) \(\phi(r) = 1/\sqrt{r^2 + c^2}, \quad c > 0\).
  - Compactly supported RBFs such as \(\phi(r) = (1 - r)^2\) if \(r < 1\), otherwise, \(\phi(r) = 0\).
Radial Basis Functions Considered: MQ, IMQ, TPS

Figure: The MQ, IMQ and CS-RBF in $\mathbb{R}^2$.

The CS-RBF is not the type of RBFs we localized in this dissertation. Note it is strongly peaked and much more rapidly varying.
Radial Basis Functions Considered: MQ, IMQ, TPS

Figure: The MQ, IMQ and CS-RBF in $\mathbb{R}^2$.

The CS-RBF is not the type of RBFs we localized in this dissertation. Note it is strongly peaked and much more rapidly varying.
Surface reconstruction scheme:

- Assume \( f(x) \) is a known function. To approximate \( f(x) \) by \( \hat{f}(x) \), we fit the given data set \( \{x_i, y_i\}_{i=1}^{N} \) of pairwise distinct centers with the imposed condition:

\[
y_i = f(x_i) = \hat{f}(x_i), \quad i = 1, 2, \ldots, N,
\]

i.e., the reconstruction of \( f \) by \( \hat{f} \) is interpolating.

- We construct \( \hat{f}(x) \) by RBFs:

\[
\hat{f}(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j).
\]

- Then the interpolation leads following linear system:

\[
\hat{f}(x_i) = \sum_{j=1}^{N} \alpha_j \phi(x_i, x_j), \quad i = 1, 2, \ldots, N. \tag{1}
\]

- If the interpolation matrix

\[
P_{NN} = [\phi(x_i, x_j)]_{1 \leq i, j \leq N}
\]

is non-singular, then (1) is a well-posed problem.
Surface reconstruction scheme:

- Assume \( f(x) \) is a known function. To approximate \( f(x) \) by \( \hat{f}(x) \), we fit the given data set \( \{x_i, y_i\}_{i=1}^N \) of pairwise distinct centers with the imposed condition:

\[
y_i = f(x_i) = \hat{f}(x_i), \quad i = 1, 2, \ldots, N,
\]

i.e., the reconstruction of \( f \) by \( \hat{f} \) is interpolating.

- We construct \( \hat{f}(x) \) by RBFs: \( \hat{f}(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j) \).

- Then the interpolation leads following linear system:

\[
\hat{f}(x_i) = \sum_{j=1}^{N} \alpha_j \phi(x_i, x_j), \quad i = 1, 2, \ldots, N. \tag{1}
\]

- If the interpolation matrix

\[
P_{NN} = [\phi(x_i, x_j)]_{1 \leq i, j \leq N}
\]

is non-singular, then (1) is a well-posed problem.
Surface reconstruction scheme:

- Assume \( f(\mathbf{x}) \) is a known function. To approximate \( f(\mathbf{x}) \) by \( \hat{f}(\mathbf{x}) \), we fit the given data set \( \{ \mathbf{x}_i, y_i \}_{i=1}^{N} \) of pairwise distinct centers with the imposed condition:

\[
y_i = f(\mathbf{x}_i) = \hat{f}(\mathbf{x}_i), \quad i = 1, 2, \ldots, N,
\]

i.e., the reconstruction of \( f \) by \( \hat{f} \) is interpolating.

- We construct \( \hat{f}(\mathbf{x}) \) by RBFs:

\[
\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi(\mathbf{x}, \mathbf{x}_j).
\]

- Then the interpolation leads to the following linear system:

\[
\hat{f}(\mathbf{x}_i) = \sum_{j=1}^{N} \alpha_j \phi(\mathbf{x}_i, \mathbf{x}_j), \quad i = 1, 2, \ldots, N. \tag{1}
\]

- If the interpolation matrix

\[
P_{NN} = [\phi(\mathbf{x}_i, \mathbf{x}_j)]_{1 \leq i, j \leq N}
\]

is non-singular, then (1) is a well-posed problem.
Surface reconstruction scheme:

- Assume $f(x)$ is a known function. To approximate $f(x)$ by $\hat{f}(x)$, we fit the given data set $\{x_i, y_i\}_{i=1}^{N}$ of pairwise distinct centers with the imposed condition:

\[
y_i = f(x_i) = \hat{f}(x_i), \quad i = 1, 2, \ldots, N,
\]

i.e., the reconstruction of $f$ by $\hat{f}$ is interpolating.

- We construct $\hat{f}(x)$ by RBFs: $\hat{f}(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j)$.

- Then the interpolation leads following linear system:

\[
\hat{f}(x_i) = \sum_{j=1}^{N} \alpha_j \phi(x_i, x_j), \quad i = 1, 2, \ldots, N. \tag{1}
\]

- If the interpolation matrix

\[
P_{NN} = [\phi(x_i, x_j)]_{1 \leq i, j \leq N}
\]

is non-singular, then (1) is a well-posed problem.
Surface reconstruction scheme:

- Assume \( f(x) \) is a known function. To approximate \( f(x) \) by \( \hat{f}(x) \), we fit the given data set \( \{x_i, y_i\}_{i=1}^N \) of pairwise distinct centers with the imposed condition:

\[
y_i = f(x_i) = \hat{f}(x_i), \quad i = 1, 2, \ldots, N,
\]

i.e., the reconstruction of \( f \) by \( \hat{f} \) is interpolating.

- We construct \( \hat{f}(x) \) by RBFs: \( \hat{f}(x) = \sum_{j=1}^N \alpha_j \phi(x, x_j) \).

- Then the interpolation leads following linear system:

\[
\hat{f}(x_i) = \sum_{j=1}^N \alpha_j \phi(x_i, x_j), \quad i = 1, 2, \ldots, N. \tag{1}
\]

- If the interpolation matrix

\[
P_{NN} = [\phi(x_i, x_j)]_{1 \leq i, j \leq N}
\]

is non-singular, then (1) is a well-posed problem.
Example. RBF interpolation of

\[ f(x, y) = -\frac{751\pi^2}{144} \sin(\frac{\pi x}{6}) \sin(\frac{7\pi x}{4}) \sin(\frac{3\pi y}{4}) \sin(\frac{5\pi y}{4}) \\
+ \frac{7\pi^2}{12} \cos(\frac{\pi x}{6}) \cos(\frac{7\pi x}{4}) \sin(\frac{3\pi y}{4}) \sin(\frac{5\pi x}{4}) \\
+ \frac{15\pi^2}{8} \sin(\frac{\pi x}{6}) \sin(\frac{7\pi x}{4}) \cos(\frac{3\pi y}{4}) \cos(\frac{5\pi y}{4}). \]

Figure: Given date set and its interpolant surface.

Note interpolant surface is nearly indistinguishable from \( \hat{f} \).
Example. RBF interpolation of

\[ f(x, y) = -\frac{751\pi^2}{144} \sin\left(\frac{\pi x}{6}\right) \sin\left(\frac{7\pi x}{4}\right) \sin\left(\frac{3\pi y}{4}\right) \sin\left(\frac{5\pi y}{4}\right) + \frac{7\pi^2}{12} \cos\left(\frac{\pi x}{6}\right) \cos\left(\frac{7\pi x}{4}\right) \sin\left(\frac{3\pi y}{4}\right) \sin\left(\frac{5\pi y}{4}\right) + \frac{15\pi^2}{8} \sin\left(\frac{\pi x}{6}\right) \sin\left(\frac{7\pi x}{4}\right) \cos\left(\frac{3\pi y}{4}\right) \cos\left(\frac{5\pi y}{4}\right). \]

Note interpolant surface is nearly indistinguishable from \( \hat{f} \).
The Method of Approximate Particular Solutions (MAPS)

- General PDE: \( \mathcal{L}, \mathcal{B} \) are differential operators

\[
\mathcal{L}u(x) = f(x), \quad x \in \Omega, \quad (2)
\]

\[
\mathcal{B}u(x) = g(x), \quad x \in \Gamma. \quad (3)
\]

- Interpolate right-hand side of (2) using RBFs:

\[
f(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j). \quad (4)
\]

- Approximate solutions to (2)–(3) by:

\[
\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j), \quad (5)
\]

where \( \Phi(r) \) is a solution to \( \mathcal{L} \Phi(r) = \phi(r) \).
The Method of Approximate Particular Solutions (MAPS)

- General PDE: \( L, B \) are differential operators

\[
Lu(x) = f(x), \quad x \in \Omega, \\
Bu(x) = g(x), \quad x \in \Gamma.
\]

- Interpolate right-hand side of (2) using RBFs:

\[
f(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j). \tag{4}
\]

- Approximate solutions to (2)–(3) by:

\[
\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j), \tag{5}
\]

where \( \Phi(r) \) is a solution to \( L \Phi(r) = \phi(r) \).
The Method of Approximate Particular Solutions (MAPS)

- General PDE: $L, B$ are differential operators

$$Lu(x) = f(x), \quad x \in \Omega, \tag{2}$$
$$Bu(x) = g(x), \quad x \in \Gamma. \tag{3}$$

- Interpolate right-hand side of (2) using RBFs:

$$f(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j). \tag{4}$$

- Approximate solutions to (2)–(3) by:

$$\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j), \tag{5}$$

where $\Phi(r)$ is a solution to $L\Phi(r) = \phi(r)$. 
The Method of Approximate Particular Solutions (MAPS)

- General PDE: $\mathcal{L}, \mathcal{B}$ are differential operators

\[
\mathcal{L}u(x) = f(x), \quad x \in \Omega, \\
\mathcal{B}u(x) = g(x), \quad x \in \Gamma.
\]  

- Interpolate right-hand side of (2) using RBFs:

\[
f(x) = \sum_{j=1}^{N} \alpha_j \phi(x, x_j).
\]  

- Approximate solutions to (2)–(3) by:

\[
\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j),
\]

where $\Phi(r)$ is a solution to $\mathcal{L}\Phi(r) = \phi(r)$. 
The solution $\Phi(r)$ to the differential operator equation

$$\mathcal{L}\Phi(r) = \phi(r)$$  \hspace{1cm} (6)

must be known in advance, i.e., we require $\Phi(r) = \mathcal{L}^{-1}\phi(r)$.

Substituting (5) into (2)–(3), we have

$$\sum_{j=1}^{N} \alpha_j \phi(x_i, x_j) = f(x_i), \quad x_i \in \Omega.$$  

$$\sum_{j=1}^{N} \alpha_j \mathcal{B}\Phi(x_i, x_j) = g(x_i), \quad x_i \in \Gamma.$$
Unknown $\{\alpha_j\}_{j=1}^N$ can be obtained by solving $N \times N$ linear system

$$
\begin{bmatrix}
\phi \\
B \Phi
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\end{bmatrix}
=
\begin{bmatrix}
f \\
g
\end{bmatrix}.
$$

(7)

Thus, an approximate solution to (2)–(3) is given by

$$
\hat{u}(x) = \sum_{j=1}^N \alpha_j \Phi(x, x_j).
$$

(8)

Conclusions

(i) Easy numerical implementation;

(ii) Not suitable for large scale problem.
Unknown \( \{ \alpha_j \}_{j=1}^{N} \) can be obtained by solving \( N \times N \) linear system

\[
\begin{bmatrix}
\Phi \\
B \Phi
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\end{bmatrix} =
\begin{bmatrix}
f \\
g
\end{bmatrix}.
\]

Thus, an approximate solution to (2)–(3) is given by

\[
\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j).
\]

Conclusions
(i) Easy numerical implementation;
(ii) Not suitable for large scale problem.
Unknown \( \{ \alpha_j \}_{j=1}^N \) can be obtained by solving \( N \times N \) linear system
\[
\begin{bmatrix}
\phi \\
B \Phi
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\end{bmatrix}
=
\begin{bmatrix}
f \\
g
\end{bmatrix}.
\] (7)

Thus, an approximate solution to (2)–(3) is given by
\[
\hat{u}(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, x_j).
\] (8)

Conclusions
(i) Easy numerical implementation;
(ii) Not suitable for large scale problem.
In implementation of the local methods, the source function $f$’s is represented in terms of approximate solutions, $\hat{u}$:

- From (5), we have following system

$$
\begin{bmatrix}
\Phi_{11} & \cdots & \Phi_{1N} \\
\vdots & \ddots & \vdots \\
\Phi_{N1} & \cdots & \Phi_{NN}
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_N
\end{bmatrix}
= 
\begin{bmatrix}
\hat{u}(x_1) \\
\vdots \\
\hat{u}(x_N)
\end{bmatrix}.
$$

- By (4), we have

$$f(x_i) = \sum_{j=1}^{N} \alpha_j \phi(x_i, x_j)$$

$$= [\phi_{i1}, \phi_{i2}, \ldots, \phi_{iN}] \begin{bmatrix}
\Phi_{11} & \cdots & \Phi_{1N} \\
\vdots & \ddots & \vdots \\
\Phi_{N1} & \cdots & \Phi_{NN}
\end{bmatrix}^{-1}
\begin{bmatrix}
\hat{u}(x_1) \\
\vdots \\
\hat{u}(x_N)
\end{bmatrix}. \quad (9)$$
(i) Derivation of solutions to $\mathcal{L}\Phi(r) = \phi(r)$ in RBF interpolant.

(ii) For some typical RBFs and elliptic operators, the function $\Phi(r)$ is derived in detail in Chapter 2:

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^2$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$r^{2n} \ln r$</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$x^k y^l$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^3$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$r^{2n-1}$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CS-RBF</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Theoretical Background

Local RBF Methods for Solving PDEs

(i) Derivation of solutions to $L\Phi(r) = \phi(r)$ in RBF interpolant.

(ii) For some typical RBFs and elliptic operators, the function $\Phi(r)$ is derived in detail in Chapter 2:

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^2$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$r^{2n}\ln r$</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$x^k y^l$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^3$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$r^{2n-1}$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CS-RBF</td>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
(i) Derivation of solutions to $L\Phi(r) = \phi(r)$ in RBF interpolant.

(ii) For some typical RBFs and elliptic operators, the function $\Phi(r)$ is derived in detail in Chapter 2:

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^2$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$r^{2n} \ln r$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$x^k y^l$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi(r)$ in $\mathbb{R}^3$</th>
<th>$\Delta$</th>
<th>$\Delta^2$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta + \lambda^2$</th>
<th>$\Delta^2 - \lambda^4$</th>
<th>$\Delta^2 + \lambda^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{r^2 + c^2}$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$1/\sqrt{r^2 + c^2}$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$r^{2n-1}$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>CS-RBF</td>
<td>$-$</td>
<td>$\checkmark$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Theorem. Let \( n \) be a positive integer, \( \lambda > 0 \). In \( \mathbb{R}^2 \), the following equation

\[
(\Delta^2 - \lambda^4) \Phi(r) = r^4 \ln r,
\]

has a solution

\[
\Phi(r) = \begin{cases} 
- \frac{r^4 \ln r}{\lambda^4} - \frac{64 \ln r + 96}{\lambda^8} - \frac{16 (2K_0(\lambda r) - \pi Y_0(\lambda r))}{\lambda^8}, & r \neq 0 \\
\frac{64}{\lambda^8} \left( \gamma + \ln \left( \frac{\lambda}{2} \right) \right) - \frac{96}{\lambda^8}, & r = 0
\end{cases}
\]

where \( K_0 \) and \( Y_0 \) are first and second kind of Bessel functions of order zero, respectively.
Proof. Main steps of construction of solution

1. Identity $\Delta^4 r^4 \ln r = 0$.

2. It follows that

$$
\left(1 - \frac{\Delta^2}{\lambda^4}\right) \left(1 + \frac{\Delta^2}{\lambda^4}\right) r^4 \ln r = r^4 \ln r.
$$

3. Rearranging (10), we have that

$$
(\Delta^2 - \lambda^4) \left(-\frac{1}{\lambda^4}\right) \left(1 + \frac{\Delta^2}{\lambda^4}\right) r^4 \ln r = r^4 \ln r.
$$

4. A solution can be written as

$$
\Phi(r) = -\frac{r^4 \ln r}{\lambda^4} - \frac{64 \ln r + 96}{\lambda^8}.
$$
5. To remove the singularities in $\Phi$ and $\Delta \Phi$, the following identities are needed:

$$\left( \Delta^2 - \lambda^4 \right) K_0(\lambda r) = 0,$$
$$\left( \Delta^2 - \lambda^4 \right) Y_0(\lambda r) = 0.$$ 

6. Assume

$$\Phi(r) = -\frac{r^4 \ln r}{\lambda^4} - \frac{64 \ln r + 96}{\lambda^8} + aK_0(\lambda r) + bY_0(\lambda r). \quad (13)$$

7. Find $a$ and $b$ such that $\lim_{r \to 0} \Phi(r)$ exists, and there is no singularity in $\Phi$, $\partial \Phi / \partial n$ and $\Delta \Phi$. ■
The Localized Method of Approximate Particular Solutions (LMAPS)

- Domain Localization

For all \( x_i \in \Omega \), we create a subdomain \( \Omega_i \) which satisfies

(i) \( \overline{\Omega} \subset \bigcup_{i=1}^{N} \Omega_i \),

(ii) \( \Omega_i \cap \Omega_j \neq \emptyset \) for some \( i \neq j \).

Then we denote \( \Omega_s \) as the local domain of \( x_s \) and denote by \( x_{js} \), \( j = 1, 2, \ldots, n \), the local index of the points associated with the node \( x_s \) in \( \Omega_s \).

Thus, each local domain \( \Omega_s \) is uniquely associated with each data point \( x_s \) in the domain.
The Localized Method of Approximate Particular Solutions (LMAPS)

- **Domain Localization**

For all \( x_i \in \Omega \), we create a subdomain \( \Omega_i \) which satisfies

1. \( \Omega \subseteq \bigcup_{i=1}^{N} \Omega_i \),
2. \( \Omega_i \cap \Omega_j \neq \emptyset \) for some \( i \neq j \).

Then we denote \( \Omega_s \) as the local domain of \( x_s \) and denote by \( x_{[s]}^j, j = 1, 2, \ldots, n \), the local index of the points associated with the node \( x_s \) in \( \Omega_s \).

Thus, each local domain \( \Omega_s \) is uniquely associated with each data point \( x_s \) in the domain.
Example. Nine-noded domain localization.

The nodes are distributed to assure a reasonable dispersion of the points to avoid topological points sets in which large regions in the domain are empty.

Note that the distribution of nodes is problem dependent, as for all PDE solution methods.

Size of subdomains is problem dependent.

The local domains, $\Omega_s$, $s = 1, 2, \ldots, N$ are associated with global point, $x_s$, which includes 9 nearest neighbors $x_{j[s]}$, $j = 1, 2, \ldots, 9$ in this example.
Example. Nine-noded domain localization.

The nodes are distributed to assure a reasonable dispersion of the points to avoid topological points sets in which large regions in the domain are empty.

Note that the distribution of nodes is problem dependent, as for all PDE solution methods.

Size of subdomains is problem dependent.

The local domains, $\Omega_s$, $s = 1, 2, \ldots, N$ are associated with global point, $x_s$, which includes 9 nearest neighbors $x^{[s]}_j, j = 1, 2, \ldots 9$ in this example.
Example. Construct LMAPS using nearest 3 neighbors on a 5 point data set in \( \mathbb{R} \).

\[
\begin{align*}
\{x_1, x_2, x_3, x_4, x_5\}
\end{align*}
\]

Solution. For global point \( x_2 \), we construct local domain

\[
\Omega_2 = \{x_1, x_2, x_3\}.
\]

Since \( x_2 \in \Omega \),

\[
\begin{align*}
f(x_2) &= \left[ \Phi_{11}, \Phi_{12}, \Phi_{13} \right]
\begin{bmatrix}
\Phi_{11} & \Phi_{12} & \Phi_{13} \\
\Phi_{21} & \Phi_{22} & \Phi_{23} \\
\Phi_{31} & \Phi_{32} & \Phi_{33}
\end{bmatrix}^{-1}
\begin{bmatrix}
\hat{u}(x_1) \\
\hat{u}(x_2) \\
\hat{u}(x_3)
\end{bmatrix} \\
&= [⋆, ⋆, ⋆] \begin{bmatrix}
\hat{u}(x_1), \\
\hat{u}(x_2), \\
\hat{u}(x_3)
\end{bmatrix}^T \\
&= [⋆, ⋆, ⋆, 0, 0] \begin{bmatrix}
\hat{u}(x_1), \\
\hat{u}(x_2), \\
\hat{u}(x_3), \\
\hat{u}(x_4), \\
\hat{u}(x_5)
\end{bmatrix}^T \\
&= [⋆, ⋆, ⋆, 0, 0] \hat{u},
\end{align*}
\]

Example. Construct LMAPS using nearest 3 neighbors on a 5 point data set in $\mathbb{R}$.

Solution. For global point $x_2$, we construct local domain

$$\Omega_2 = \{x_1, x_2, x_3\}.$$ 

Since $x_2 \in \Omega$,

$$f(x_2) = [\phi_{11}, \phi_{12}, \phi_{13}] \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} \\ \Phi_{21} & \Phi_{22} & \Phi_{23} \\ \Phi_{31} & \Phi_{32} & \Phi_{33} \end{bmatrix}^{-1} \begin{bmatrix} \hat{u}(x_1) \\ \hat{u}(x_2) \\ \hat{u}(x_3) \end{bmatrix}$$

$$= [\ast, \ast, \ast] [\hat{u}(x_1), \hat{u}(x_2), \hat{u}(x_3)]^T$$

$$= [\ast, \ast, \ast, 0, 0] [\hat{u}(x_1), \hat{u}(x_2), \hat{u}(x_3), \hat{u}(x_4), \hat{u}(x_5)]^T$$

$$= [\ast, \ast, \ast, 0, 0] \hat{u},$$
Similarly, we have

\[
f(x_3) = [0, *, *, *, 0, 0]\hat{u},
f(x_4) = [0, 0, *, *, *]\hat{u}.
\]

Thus,

\[
\begin{bmatrix}
  f(x_2) \\
  f(x_3) \\
  f(x_4) \\
  g(x_1) \\
  g(x_5)
\end{bmatrix}
= \begin{bmatrix}
  * & * & * & 0 & 0 \\
  0 & * & * & * & 0 \\
  0 & 0 & * & * & * \\
  * & * & * & 0 & 0 \\
  0 & 0 & * & * & *
\end{bmatrix}
\begin{bmatrix}
  \hat{u}(x_1) \\
  \hat{u}(x_2) \\
  \hat{u}(x_3) \\
  \hat{u}(x_4) \\
  \hat{u}(x_5)
\end{bmatrix}.
\]
Consider solving problem (2)–(3) using LMAPS.

Applying MAPS on local domain $\Omega_s$, we have

$$\hat{u}(x_j) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_s, x_k^{[s]}),$$

where $j = 1, 2, \ldots, n$, i.e.,

$$\hat{u}^{[s]} = P_{nn} \alpha^{[s]}.$$

Both $\hat{u}^{[s]}$ and $\alpha^{[s]}$ are unknown:

$$\alpha^{[s]} = P_{nn}^{-1} \hat{u}^{[s]}.$$
Continuing the LMAPS solution

▶ By MAPS,

\[
\begin{align*}
  f(x_s) &= \sum_{k=1}^{n} \alpha^{[s]}_k \phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Omega, \\
g(x_s) &= \sum_{s=1}^{n} \alpha^{[s]}_k B \Phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Gamma.
\end{align*}
\]

Note that for \( k = 1, 2, \ldots, n \), we have \( \alpha^{[s]}_k = \sum_{j=1}^{n} P_{kj}^{-1} u(x_j^{[s]}). \)

▶ Substitute \( \alpha^{[s]}_n \) into (14)–(15),

\[
\begin{align*}
\sum_{k=1}^{n} \sum_{s=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) &= f(x_s), \quad x_s \in \Omega, \\
\sum_{k=1}^{n} \sum_{s=1}^{n} P_{kj}^{-1} B \Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) &= g(x_s), \quad x_s \in \Gamma.
\end{align*}
\]

We need to take a closer look at (16)–(17).
Continuing the LMAPS solution

- By MAPS,

\[
  f(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Omega, \quad (14)
\]

\[
  g(x_s) = \sum_{s=1}^{n} \alpha_k^{[s]} \mathcal{B}\Phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Gamma. \quad (15)
\]

- Note that for \( k = 1, 2, \ldots, n \), we have \( \alpha_k^{[s]} = \sum_{j=1}^{n} P_{kj}^{-1} u(x_j^{[s]}) \).

- Substitute \( \alpha_n^{[s]} \) into (14)–(15),

\[
  \sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega, \quad (16)
\]

\[
  \sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \mathcal{B}\Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma. \quad (17)
\]

We need to take a closer look at (16)–(17).
Continuing the LMAPS solution

- By MAPS,

\[ f(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Omega, \]  

(14)

\[ g(x_s) = \sum_{s=1}^{n} \alpha_k^{[s]} B\Phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Gamma. \]  

(15)

- Note that for \( k = 1, 2, \ldots, n \), we have \( \alpha_k^{[s]} = \sum_{j=1}^{n} P_{kj}^{-1} u(x_j^{[s]}). \)

- Substitute \( \alpha_k^{[s]} \) into (14)–(15),

\[ \sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega, \]  

(16)

\[ \sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} B\Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma. \]  

(17)

We need to take a closer look at (16)–(17).
Continuing the LMAPS solution

- By MAPS,

\[
f(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Omega, \tag{14}
\]

\[
g(x_s) = \sum_{s=1}^{n} \alpha_k^{[s]} B \Phi(x_s, x_k^{[s]}), \quad \text{if } x_s \in \Gamma. \tag{15}
\]

- Note that for \(k = 1, 2, \ldots, n\), we have \(\alpha_k^{[s]} = \sum_{j=1}^{n} P_{kj}^{-1} u(x_j^{[s]}). \)

- Substitute \(\alpha_n^{[s]}\) into (14)–(15),

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega, \tag{16}
\]

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma. \tag{17}
\]

We need to take a closer look at (16)–(17).
Thus, we have

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega, \quad (16)
\]

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \mathcal{B}\phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma. \quad (17)
\]

\[
\begin{bmatrix}
P_{nn}^{-1} \phi \\
P_{nn}^{-1} \mathcal{B}\phi
\end{bmatrix}
\begin{bmatrix}
\hat{u}
\end{bmatrix}
= 
\begin{bmatrix}
f \\
g
\end{bmatrix}. \quad (18)
\]

(i) Note there are only \(n\) terms in each equation, thus (16)–(17) is an \(N \times N\) sparse system.

(ii) Therefore, an approximate solution \(\hat{u}\) can be obtained by using a sparse system solver.
Thus, we have

$$\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega,$$

(16)

$$\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma.$$  

(17)

$$\Downarrow$$

$$\begin{bmatrix} P_{nn}^{-1} \phi \\ P_{nn}^{-1} \Phi \end{bmatrix} \begin{bmatrix} \hat{u} \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.$$  

(18)

(i) Note there are only $n$ terms in each equation, thus (16)–(17) is an $N \times N$ sparse system.

(ii) Therefore, an approximate solution $\hat{u}$ can be obtained by using a sparse system solver.
Thus, we have

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} \phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = f(x_s), \quad x_s \in \Omega,
\]

(16)

\[
\sum_{k=1}^{n} \sum_{j=1}^{n} P_{kj}^{-1} B \Phi(x_s, x_k^{[s]}) \hat{u}(x_j^{[s]}) = g(x_s), \quad x_s \in \Gamma.
\]

(17)

\[
\Downarrow
\]

\[
\begin{bmatrix}
P_{nn}^{-1} \phi \\
B \Phi
\end{bmatrix}
\begin{bmatrix}
\hat{u}
\end{bmatrix}
= \begin{bmatrix}
f \\
g
\end{bmatrix}.
\]

(18)

(i) Note there are only \( n \) terms in each equation, thus (16)–(17) is an \( N \times N \) sparse system.

(ii) Therefore, an approximate solution \( \hat{u} \) can be obtained by using a sparse system solver.
Example. Consider general PDE as shown in (2)–(3). Construct ELMAPS using nearest 3 neighbors on a 5 point data set in $\mathbb{R}$.

\begin{center}
\begin{tikzpicture}
\draw[blue,thick] (0,0) -- (4,0);
\filldraw[blue] (0,0) circle (4pt) node[below] {$x_1$};
\filldraw[blue] (1,0) circle (4pt) node[below] {$x_2$};
\filldraw[blue] (2,0) circle (4pt) node[below] {$x_3$};
\filldraw[blue] (3,0) circle (4pt) node[below] {$x_4$};
\filldraw[blue] (4,0) circle (4pt) node[below] {$x_5$};
\end{tikzpicture}
\end{center}

Solution. For $x_2$, we have $\Omega_2 = \{x_1, x_2, x_3\}$. Denote $\Omega_2$ by $\{x_k^{[2]}\}_{k=1}^3$. Then,

$$x_1^{[2]} = x_1, \quad x_2^{[2]} = x_2, \quad x_3^{[2]} = x_3.$$ 

To approximate $\hat{u}(x_2)$, we focus only on $\Omega_2$. 
Example. Consider general PDE as shown in (2)–(3). Construct ELMAPS using nearest 3 neighbors on a 5 point data set in $\mathbb{R}$.

Solution. For $x_2$, we have $\Omega_2 = \{x_1, x_2, x_3\}$. Denote $\Omega_2$ by $\{x_k^{[2]}\}_{k=1}^3$. Then,

$$x_1^{[2]} = x_1, \quad x_2^{[2]} = x_2, \quad x_3^{[2]} = x_3.$$

To approximate $\hat{u}(x_2)$, we focus only on $\Omega_2$. 
We construct a solution to the local problem on $\Omega_2$. The following linear system can be obtained from (2)–(3),

$$
\begin{bmatrix}
  f(x_2) \\
  f(x_3) \\
  g(x_1)
\end{bmatrix} =
\begin{bmatrix}
  \phi(x_2, x_1) & \phi(x_2, x_2) & \phi(x_2, x_3) \\
  \phi(x_3, x_1) & \phi(x_3, x_2) & \phi(x_3, x_3) \\
  B\Phi(x_1, x_1) & B\Phi(x_1, x_2) & B\Phi(x_1, x_3)
\end{bmatrix}
\begin{bmatrix}
  \alpha_1^{[2]} \\
  \alpha_2^{[2]} \\
  \alpha_3^{[2]}
\end{bmatrix}.
$$

To determine $\alpha_k^{[2]}$’s, we need to invert the small local matrix on the left-hand side of (19). Thus,

$$
\hat{u}(x_2) = \sum_{k=1}^{3} \alpha_k^{[2]} \Phi(x_2, x_k^{[2]}) = \sum_{k=1}^{3} \alpha_k^{[2]} \Phi(x_2, x_k).
$$
We construct a solution to the local problem on $\Omega_2$. The following linear system can be obtained from (2)–(3),

$$
\begin{bmatrix}
  f(x_2) \\
  f(x_3) \\
  g(x_1)
\end{bmatrix}
= 
\begin{bmatrix}
  \phi(x_2, x_1) & \phi(x_2, x_2) & \phi(x_2, x_3) \\
  \phi(x_3, x_1) & \phi(x_3, x_2) & \phi(x_3, x_3) \\
  B\Phi(x_1, x_1) & B\Phi(x_1, x_2) & B\Phi(x_1, x_3)
\end{bmatrix}
\begin{bmatrix}
  \alpha_{1}^{[2]} \\
  \alpha_{2}^{[2]} \\
  \alpha_{3}^{[2]}
\end{bmatrix}.
$$

(19)

To determine $\alpha_k^{[2]}$'s, we need to invert the small local matrix on the left-hand side of (19). Thus,

$$
\hat{u}(x_2) = \sum_{k=1}^{3} \alpha_k^{[2]} \Phi(x_2, x_k^{[2]}) = \sum_{k=1}^{3} \alpha_k^{[2]} \Phi(x_2, x_k).
$$

(20)
General case for ELMAPS:

Again consider the problem (2)–(3):

Let

\[ \hat{u}(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_s, x_k^{[s]}). \]

Applying MAPS on local domain \( \Omega_s \), we have

\[ u(x_j^{[s]}) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_j^{[s]}, x_k^{[s]}), \]

where \( j = 1, 2, \ldots, n \). Then

\[ \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_j^{[s]}, x_k^{[s]}) = f(x_j^{[s]}), \quad x_j^{[s]} \in \Omega, \]

\[ \sum_{k=1}^{n} \alpha_k^{[s]} \mathcal{B} \Phi(x_j^{[s]}, x_k^{[s]}) = g(x_j^{[s]}), \quad x_j^{[s]} \in \Gamma. \]
General case for ELMAPS:
- Again consider the problem (2)–(3):
  
  Let

  \[ \hat{u}(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_s, x_k^{[s]}). \]

  Applying MAPS on local domain \( \Omega_s \), we have

  \[ u(x_j^{[s]}) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_j^{[s]}, x_k^{[s]}), \]

  where \( j = 1, 2, \ldots, n \). Then

  \[ \sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_j^{[s]}, x_k^{[s]}) = f(x_j^{[s]}), \quad x_j^{[s]} \in \Omega, \]

  \[ \sum_{k=1}^{n} \alpha_k^{[s]} B \Phi(x_j^{[s]}, x_k^{[s]}) = g(x_j^{[s]}), \quad x_j^{[s]} \in \Gamma. \]
General case for ELMAPS:

- Again consider the problem (2)–(3):

Let

\[ \hat{u}(x_s) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_s, x_k^{[s]}). \]

Applying MAPS on local domain \( \Omega_s \), we have

\[ u(x_j^{[s]}) = \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_j^{[s]}, x_k^{[s]}), \]

where \( j = 1, 2, \ldots, n \). Then

\[ \sum_{k=1}^{n} \alpha_k^{[s]} \Phi(x_j^{[s]}, x_k^{[s]}) = f(x_j^{[s]}), \quad x_j^{[s]} \in \Omega, \]

\[ \sum_{k=1}^{n} \alpha_k^{[s]} \mathcal{B}\Phi(x_j^{[s]}, x_k^{[s]}) = g(x_j^{[s]}), \quad x_j^{[s]} \in \Gamma. \]
ELMAPS constructing the solution:

- To determine $\alpha^{[s]}_k$'s, we need to solve a small system of equations

\[
\sum_{k=1}^{n} \alpha^{[s]}_k \phi(x^{[s]}_j, x^{[s]}_k) = f(x^{[s]}_j), \quad x^{[s]}_j \in \Omega,
\]

\[
\sum_{k=1}^{n} \alpha^{[s]}_k B\Phi(x^{[s]}_j, x^{[s]}_k) = g(x^{[s]}_j), \quad x^{[s]}_j \in \Gamma.
\]

\[
\begin{bmatrix}
\phi \\
B\Phi
\end{bmatrix}
\begin{bmatrix}
\alpha^{[s]} \\
\end{bmatrix}
= 
\begin{bmatrix}
f \\
g
\end{bmatrix}.
\quad (21)
\]

- The approximation $\hat{u}$ is obtained by inverting the matrix in (21).

Note the matrix is a local matrix with size of the number of points in the local domain.
ELMAPS constructing the solution:

- To determine $\alpha_k^{[s]}$'s, we need to solve a small system of equations

$$
\sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_j^{[s]}, x_k^{[s]}) = f(x_j^{[s]}), \quad x_j^{[s]} \in \Omega,
$$

$$
\sum_{k=1}^{n} \alpha_k^{[s]} B\Phi(x_j^{[s]}, x_k^{[s]}) = g(x_j^{[s]}), \quad x_j^{[s]} \in \Gamma.
$$

\[\downarrow\]

$$
\begin{bmatrix}
\phi \\
B\Phi
\end{bmatrix}
\begin{bmatrix}
\alpha^{[s]}
\end{bmatrix}
= 
\begin{bmatrix}
f \\
g
\end{bmatrix}.
\] (21)

- The approximation $\hat{u}$ is obtained by inverting the matrix in (21).

Note the matrix is a local matrix with size of the number of points in the local domain.
ELMAPS constructing the solution:

- To determine $\alpha_k^{[s]}$'s, we need to solve a small system of equations:

$$
\sum_{k=1}^{n} \alpha_k^{[s]} \phi(x_j^{[s]}, x_k^{[s]}) = f(x_j^{[s]}), \quad x_j^{[s]} \in \Omega,
$$

$$
\sum_{k=1}^{n} \alpha_k^{[s]} B\Phi(x_j^{[s]}, x_k^{[s]}) = g(x_j^{[s]}), \quad x_j^{[s]} \in \Gamma.
$$

$$
\begin{bmatrix}
\phi \\
B\Phi
\end{bmatrix}
\begin{bmatrix}
\alpha^{[s]}
\end{bmatrix}
=
\begin{bmatrix}
f \\
g
\end{bmatrix}.
$$

(21)

- The approximation $\hat{u}$ is obtained by inverting the matrix in (21).

Note the matrix is a local matrix with size of the number of points in the local domain.
Thus far we have only worked with constant coefficient differential equations. We need to develop a generalization of the LMAPS and ELMAPS in order to have a workable numerical method.

(i) Consider (2)–(3). The solution, $\Phi(r)$, of $\mathcal{L}\Phi(r) = \phi(r)$ is unknown.

(ii) We will solve an equivalent PDE

$$\mathcal{L}'u(x) = f(x) + (\mathcal{L}' - \mathcal{L})u(x).$$  \hfill (22)

(iii) The solution to $\mathcal{L}'\Phi(r) = \phi(r)$ is known.

(iv) Using LMAPS, an approximate solution of (22), $\hat{u}(x)$, can be obtained.

(v) Then $\hat{u}(x)$ is also a solution to (2)–(3).
Thus far we have only worked with constant coefficient differential equations. We need to develop a generalization of the LMAPS and ELMAPS in order to have a workable numerical method.

(i) Consider (2)–(3). The solution, $\Phi(r)$, of $L\Phi(r) = \phi(r)$ is unknown.

(ii) We will solve an equivalent PDE

$$L'u(x) = f(x) + (L' - L)u(x).$$  \hspace{1cm} (22)

(iii) The solution to $L'\Phi(r) = \phi(r)$ is known.

(iv) Using LMAPS, an approximate solution of (22), $\hat{u}(x)$, can be obtained.

(v) Then $\hat{u}(x)$ is also a solution to (2)–(3).
Thus far we have only worked with constant coefficient differential equations. We need to develop a generalization of the LMAPS and ELMAPS in order to have a workable numerical method.

(i) Consider (2)–(3). The solution, $\Phi(r)$, of $L\Phi(r) = \phi(r)$ is unknown.

(ii) We will solve an equivalent PDE

$$L'u(x) = f(x) + (L' - L)u(x).$$  \hspace{1cm} (22)

(iii) The solution to $L'\Phi(r) = \phi(r)$ is known.

(iv) Using LMAPS, an approximate solution of (22), $\hat{u}(x)$, can be obtained.

(v) Then $\hat{u}(x)$ is also a solution to (2)–(3).
Thus far we have only worked with constant coefficient differential equations. We need to develop a generalization of the LMAPS and ELMAPS in order to have a workable numerical method.

(i) Consider (2)–(3). The solution, \( \Phi(r) \), of \( L\Phi(r) = \phi(r) \) is unknown.

(ii) We will solve an equivalent PDE

\[
L' u(x) = f(x) + (L' - L) u(x). \tag{22}
\]

(iii) The solution to \( L' \Phi(r) = \phi(r) \) is known.

(iv) Using LMAPS, an approximate solution of (22), \( \hat{u}(x) \), can be obtained.

(v) Then \( \hat{u}(x) \) is also a solution to (2)–(3).
Thus far we have only worked with constant coefficient differential equations. We need to develop a generalization of the LMAPS and ELMAPS in order to have a workable numerical method.

(i) Consider (2)–(3). The solution, \( \Phi(r) \), of \( \mathcal{L}\Phi(r) = \phi(r) \) is unknown.

(ii) We will solve an equivalent PDE

\[
\mathcal{L}'u(x) = f(x) + (\mathcal{L}' - \mathcal{L})u(x).
\]

(iii) The solution to \( \mathcal{L}'\Phi(r) = \phi(r) \) is known.

(iv) Using LMAPS, an approximate solution of (22), \( \hat{u}(x) \), can be obtained.

(v) Then \( \hat{u}(x) \) is also a solution to (2)–(3).
Consider the diffusion equation

$$\frac{\partial u(x, t)}{\partial t} = \Delta u(x, t).$$ \hspace{1cm} (23)

- **Time-stepping methods**

  $$\frac{\partial u(x, t)}{\partial t} \approx \frac{u(x, t_m) - u(x, t_{m-1})}{\delta_t},$$

  where $\delta_t > 0$ is the time step, and $t_m = m\delta_t$.

- To approximate $u(x, t)$ on the interval $(t_{m-1}, t_m]$, let $\Delta u(x, t)$ be determined by the value at $t_m$ or $t_{m-1}$, i.e.,

  $$\Delta u(x, t) \approx \begin{cases} 
  \Delta u(x, t_m), & \text{implicit method} \\
  \Delta u(x, t_{m-1}), & \text{explicit method}
  \end{cases} \quad (24)$$
Consider the diffusion equation

\[
\frac{\partial u(x, t)}{\partial t} = \Delta u(x, t). \tag{23}
\]

- **Time-stepping methods**

\[
\frac{\partial u(x, t)}{\partial t} \approx \frac{u(x, t_m) - u(x, t_{m-1})}{\delta_t},
\]

where \(\delta_t > 0\) is the time step, and \(t_m = m\delta_t\).

- To approximate \(u(x, t)\) on the interval \((t_{m-1}, t_m]\), let \(\Delta u(x, t)\) be determined by the value at \(t_m\) or \(t_{m-1}\), i.e.,

\[
\Delta u(x, t) \approx \begin{cases} \\
\Delta u(x, t_m), & \text{implicit method} \\
\Delta u(x, t_{m-1}), & \text{explicit method}
\end{cases}
\]
Consider the diffusion equation

\[
\frac{\partial u(x,t)}{\partial t} = \Delta u(x,t).
\]

(23)

- **Time-stepping methods**

\[
\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x,t_m) - u(x,t_{m-1})}{\delta_t},
\]

where \(\delta_t > 0\) is the time step, and \(t_m = m\delta_t\).

- To approximate \(u(x,t)\) on the interval \((t_{m-1}, t_m]\), let \(\Delta u(x,t)\) be determined by the value at \(t_m\) or \(t_{m-1}\), i.e.,

\[
\Delta u(x,t) \approx \begin{cases} 
\Delta u(x,t_m), & \text{implicit method} \\
\Delta u(x,t_{m-1}), & \text{explicit method}
\end{cases}
\]

(24)
Implicit (1-step): $\Delta u(x, t) = \Delta u(x, t_m)$ We have

$$\frac{u(x, t_m) - u(t_{m-1})}{\delta_t} = \Delta u(x, t_m),$$

$$\Rightarrow \left(\Delta - \frac{1}{\delta_t}\right)u(x, t_m) = -\frac{1}{\delta_t}u(x, t_{m-1}), \quad x \in \Omega; \quad (25)$$

Explicit (1-step): $\Delta u(x, t) = \Delta u(x, t_{m-1})$ We have

$$\frac{u(x, t_m) - u(t_{m-1})}{\delta_t} = \Delta u(x, t_{m-1}),$$

$$\Rightarrow u(x, t_m) = u(x, t_{m-1}) + \delta_t \Delta u(x, t_{m-1}), \quad x \in \Omega. \quad (26)$$
Implicit (1-step): $\Delta u(x, t) = \Delta u(x, t_m)$ We have

$$\frac{u(x, t_m) - u(t_{m-1})}{\delta_t} = \Delta u(x, t_m),$$

$$\implies \left(\Delta - \frac{1}{\delta_t}\right) u(x, t_m) = -\frac{1}{\delta_t} u(x, t_{m-1}), \quad x \in \Omega; \quad (25)$$

Explicit (1-step): $\Delta u(x, t) = \Delta u(x, t_{m-1})$ We have

$$\frac{u(x, t_m) - u(t_{m-1})}{\delta_t} = \Delta u(x, t_{m-1}),$$

$$\implies u(x, t_m) = u(x, t_{m-1}) + \delta_t \Delta u(x, t_{m-1}), \quad x \in \Omega. \quad (26)$$
## Comparison of the LMAPS and ELMAPS:

<table>
<thead>
<tr>
<th></th>
<th>LMAPS</th>
<th>ELMAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of $P_{nn}$</td>
<td>$n \times n$</td>
<td>$n \times n$</td>
</tr>
<tr>
<td># of $P_{nn}$</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Invertibility of $P_{nn}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Global matrix</td>
<td>$N \times N$ sparse</td>
<td>No</td>
</tr>
<tr>
<td>Neighbors searching</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sparse solver</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Time step size $\delta_t$</td>
<td>Can be relative large</td>
<td>Small*</td>
</tr>
<tr>
<td>Random nodes</td>
<td>works for quasi-random nodes</td>
<td>not very well</td>
</tr>
</tbody>
</table>

Table: The total number of interpolation points is denoted as $N$, and the number of points in the local domain is $n$. *The time step $\delta_t$ is limited by a CFL limit much the same as for explicit FEM.

- LMAPS generates a large sparse matrix;
- ELMAPS gives small dense matrices only;
- As always, the solutions reduce to challenges in computational matrix algebra.
Comparison of the LMAPS and ELMAPS:

<table>
<thead>
<tr>
<th></th>
<th>LMAPS</th>
<th>ELMAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of $P_{nn}$</td>
<td>$n \times n$</td>
<td>$n \times n$</td>
</tr>
<tr>
<td># of $P_{nn}$</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Invertibility of $P_{nn}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Global matrix</td>
<td>$N \times N$ sparse</td>
<td>No</td>
</tr>
<tr>
<td>Neighbors searching</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sparse solver</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Time step size $\delta_t$</td>
<td>Can be relative large</td>
<td>Small*</td>
</tr>
<tr>
<td>Random nodes</td>
<td>works for quasi-random nodes</td>
<td>not very well</td>
</tr>
</tbody>
</table>

Table: The total number of interpolation points is denoted as $N$, and the number of points in the local domain is $n$. *The time step $\delta_t$ is limited by a CFL limit much the same as for explicit FEM.

(i) LMAPS generates a large sparse matrix;

(ii) ELMAPS gives small dense matrices only;

(iii) As always, the solutions reduce to challenges in computational matrix algebra.
Comparison of the LMAPS and ELMAPS:

<table>
<thead>
<tr>
<th></th>
<th>LMAPS</th>
<th>ELMAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of $P_{nn}$</td>
<td>$n \times n$</td>
<td>$n \times n$</td>
</tr>
<tr>
<td># of $P_{nn}$</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Invertibility of $P_{nn}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Global matrix</td>
<td>$N \times N$ sparse</td>
<td>No</td>
</tr>
<tr>
<td>Neighbors searching</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sparse solver</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Time step size $\delta_t$</td>
<td>Can be relative large</td>
<td>Small*</td>
</tr>
<tr>
<td>Random nodes</td>
<td>works for quasi-random nodes</td>
<td>not very well</td>
</tr>
</tbody>
</table>

Table: The total number of interpolation points is denoted as $N$, and the number of points in the local domain is $n$. *The time step $\delta_t$ is limited by a CFL limit much the same as for explicit FEM.

(i) LMAPS generates a large sparse matrix;

(ii) ELMAPS gives small dense matrices only;

(iii) As always, the solutions reduce to challenges in computational matrix algebra.
### Comparison of the LMAPS and ELMAPS:

<table>
<thead>
<tr>
<th>Comparison</th>
<th>LMAPS</th>
<th>ELMAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of $P_{nn}$</td>
<td>$n \times n$</td>
<td>$n \times n$</td>
</tr>
<tr>
<td># of $P_{nn}$</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Invertibility of $P_{nn}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Global matrix</td>
<td>$N \times N$ sparse</td>
<td>No</td>
</tr>
<tr>
<td>Neighbors searching</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sparse solver</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Time step size $\delta_t$</td>
<td>Can be relatively large</td>
<td>Small*</td>
</tr>
<tr>
<td>Random nodes</td>
<td>works for quasi-random nodes</td>
<td>not very well</td>
</tr>
</tbody>
</table>

Table: The total number of interpolation points is denoted as $N$, and the number of points in the local domain is $n$. *The time step $\delta_t$ is limited by a CFL limit much the same as for explicit FEM.

(i) LMAPS generates a large sparse matrix;  
(ii) ELMAPS gives small dense matrices only;  
(iii) As always, the solutions reduce to challenges in computational matrix algebra.
Example. We consider second-order PDE with variable coefficients

\[ Lu(x,y) = \Delta u + y \cos(y)u_x + \sinh(x)u_y + 10xyu = f(x,y), \text{ in } \Omega, \]
\[ Bu(x,y) = \frac{\partial u}{\partial n} = g(x,y), \text{ in } \Gamma, \]

where \( n \) is the outward normal derivative. The source \( f \) and \( g \) are given by applying \( L \) and \( B \) to the analytical solution of

\[ u(x,y) = \sin \left( \frac{\pi x}{6} \right) \sin \left( \frac{7\pi x}{4} \right) \sin \left( \frac{3\pi y}{4} \right) \sin \left( \frac{5\pi y}{4} \right). \]  

(27)
- Bottom: analytical solution.
- Right: computational domain showing node distribution.

Note, the interior nodes are shown in red, and boundary nodes are in blue. The analytical solution is shown extended to the entire box $[-2, 2] \times [-2, 2]$ containing the star shaped computational domain.
The errors v. $n$.

The error v. $c$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\varepsilon_r$</th>
<th>$\varepsilon_{xr}$</th>
<th>$c_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$1.41 \times 10^{-4}$</td>
<td>$4.32 \times 10^{-3}$</td>
<td>0.4</td>
</tr>
<tr>
<td>9</td>
<td>$1.90 \times 10^{-4}$</td>
<td>$5.31 \times 10^{-3}$</td>
<td>1.4</td>
</tr>
<tr>
<td>11</td>
<td>$6.29 \times 10^{-5}$</td>
<td>$8.46 \times 10^{-4}$</td>
<td>1.3</td>
</tr>
<tr>
<td>13</td>
<td>$1.56 \times 10^{-5}$</td>
<td>$5.91 \times 10^{-4}$</td>
<td>0.5</td>
</tr>
<tr>
<td>15</td>
<td>$4.32 \times 10^{-6}$</td>
<td>$3.31 \times 10^{-4}$</td>
<td>0.6</td>
</tr>
</tbody>
</table>

- **Parameters:**
  - $\varepsilon_r, \varepsilon_{xr}$ are root mean square error of $\hat{u}$ and $\partial \hat{u}/\partial x$;
  - $c_o$: optimal shape parameter in MQ;

  - The accuracy improves, when larger $n$ is used;
  - Small $n$ can reach accurate solution;
  - The LMAPS is more stable and accurate than LMQ in $c$. 

  ![Graph showing the errors versus $n$ and $c$](image)
The errors $v. n.$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\varepsilon_r$</th>
<th>$\varepsilon_{xr}$</th>
<th>$c_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$1.41 \times 10^{-4}$</td>
<td>$4.32 \times 10^{-3}$</td>
<td>0.4</td>
</tr>
<tr>
<td>9</td>
<td>$1.90 \times 10^{-4}$</td>
<td>$5.31 \times 10^{-3}$</td>
<td>1.4</td>
</tr>
<tr>
<td>11</td>
<td>$6.29 \times 10^{-5}$</td>
<td>$8.46 \times 10^{-4}$</td>
<td>1.3</td>
</tr>
<tr>
<td>13</td>
<td>$1.56 \times 10^{-5}$</td>
<td>$5.91 \times 10^{-4}$</td>
<td>0.5</td>
</tr>
<tr>
<td>15</td>
<td>$4.32 \times 10^{-6}$</td>
<td>$3.31 \times 10^{-4}$</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The accuracy improves, when larger $n$ is used;

- Small $n$ can reach accurate solution;
- The LMAPS is more stable and accurate than LMQ in $c$.

Parameters:
- $\varepsilon_r, \varepsilon_{xr}$ are root mean square error of $\hat{u}$ and $\partial \hat{u}/\partial x$;
- $c_o$: optimal shape parameter in MQ;
The errors $v. n$. 

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\varepsilon_r$</th>
<th>$\varepsilon_{xr}$</th>
<th>$c_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$1.41 \times 10^{-4}$</td>
<td>$4.32 \times 10^{-3}$</td>
<td>0.4</td>
</tr>
<tr>
<td>9</td>
<td>$1.90 \times 10^{-4}$</td>
<td>$5.31 \times 10^{-3}$</td>
<td>1.4</td>
</tr>
<tr>
<td>11</td>
<td>$6.29 \times 10^{-5}$</td>
<td>$8.46 \times 10^{-4}$</td>
<td>1.3</td>
</tr>
<tr>
<td>13</td>
<td>$1.56 \times 10^{-5}$</td>
<td>$5.91 \times 10^{-4}$</td>
<td>0.5</td>
</tr>
<tr>
<td>15</td>
<td>$4.32 \times 10^{-6}$</td>
<td>$3.31 \times 10^{-4}$</td>
<td>0.6</td>
</tr>
</tbody>
</table>

- **Parameters:**
  - $\varepsilon_r, \varepsilon_{xr}$ are root mean square error of $\hat{u}$ and $\partial \hat{u} / \partial x$;
  - $c_o$: optimal shape parameter in MQ;

- The accuracy improves, when larger $n$ is used;

- Small $n$ can reach accurate solution;

- The LMAPS is more stable and accurate than LMQ in $c$. 

The error $v. c$
### Parameters:
- \( \varepsilon_r, \varepsilon_{xr} \) are root mean square error of \( \hat{u} \) and \( \partial \hat{u}/\partial x \);
- \( c_o \): optimal shape parameter in MQ;

### The accuracy improves, when larger \( n \) is used;

### Small \( n \) can reach accurate solution;

### The LMAPS is more stable and accurate than LMQ in \( c \).
Example. Consider the following Poisson equation in $\mathbb{R}^2$:

$$\Delta u(x, y) = f(x, y), \quad (x, y) \in \Omega,$$

$$u(x, y) = g(x, y), \quad (x, y) \in \Gamma,$$

where $\Omega \cup \Gamma = [0, 1] \times [0, L]$. The functions $f$ and $g$ are chosen according to the following analytical solution

$$u(x, y) = \frac{1.25 + \cos(5.4y + 2.7)}{6[1 + (3x + 0.5)^2]}.$$
The $\varepsilon_r$ and $\varepsilon_{xr}$ versus $L$ with the number of nodes in local domain $n = 9$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\varepsilon_r$</th>
<th>$\varepsilon_{xr}$</th>
<th>$c_o$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>99,232</td>
<td>$1.10 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>100</td>
</tr>
<tr>
<td>198,432</td>
<td>$1.08 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>200</td>
</tr>
<tr>
<td>376,992</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.53 \times 10^{-3}$</td>
<td>1.6</td>
<td>380</td>
</tr>
<tr>
<td>496,032</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.54 \times 10^{-3}$</td>
<td>1.6</td>
<td>500</td>
</tr>
<tr>
<td>595,232</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>600</td>
</tr>
<tr>
<td>694,432</td>
<td>$1.10 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>700</td>
</tr>
<tr>
<td>803,552</td>
<td>$1.09 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>810</td>
</tr>
</tbody>
</table>

As $L$ increases, none of the global methods could handle this problem due to ill-conditioning, however, the LMAPS achieved a quality solution over an extremely irregular domains.

$c_o$: optimal shape parameter;
$\varepsilon_r$: root mean square error of $\hat{u}$;
$\varepsilon_{xr}$: root mean square error of $\partial \hat{u}/\partial x$.

LMAPS works well for rectangular domain $[0,1] \times [0,810]$. 
The $\varepsilon_r$ and $\varepsilon_{xr}$ versus $L$ with the number of nodes in local domain $n = 9$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\varepsilon_r$</th>
<th>$\varepsilon_{xr}$</th>
<th>$c_o$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>99,232</td>
<td>$1.10 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>100</td>
</tr>
<tr>
<td>198,432</td>
<td>$1.08 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>200</td>
</tr>
<tr>
<td>376,992</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.53 \times 10^{-3}$</td>
<td>1.6</td>
<td>380</td>
</tr>
<tr>
<td>496,032</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.54 \times 10^{-3}$</td>
<td>1.6</td>
<td>500</td>
</tr>
<tr>
<td>595,232</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>600</td>
</tr>
<tr>
<td>694,432</td>
<td>$1.10 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>700</td>
</tr>
<tr>
<td>803,552</td>
<td>$1.09 \times 10^{-4}$</td>
<td>$1.52 \times 10^{-3}$</td>
<td>1.6</td>
<td>810</td>
</tr>
</tbody>
</table>

As $L$ increases, none of the global methods could handle this problem due to ill-conditioning, however, the LMAPS achieved a quality solution over an extremely irregular domains.

$c_o$: optimal shape parameter; $\varepsilon_r$: root mean square error of $\hat{u}$; $\varepsilon_{xr}$: root mean square error of $\partial \hat{u}/\partial x$.

LMAPS works well for rectangular domain $[0, 1] \times [0, 810]$. 
Left: relative error with \( L = 1, n = 5, N = 10^4, c = 8.9 \).

Right: the rate of convergence on \([0, 1] \times [0, 1]\), with \( n = 5 \).

Nearly second-order convergence in unit square domain.
- Left: relative error with $L = 1, n = 5, N = 10^4, c = 8.9$.

- Right: the rate of convergence on $[0, 1] \times [0, 1]$, with $n = 5$.

Nearly second-order convergence in unit square domain.
We examine the quality of the solution based on varying number of points, $N$, on the unit square with $n = 5$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$c_o$</th>
<th>$\varepsilon_m$</th>
<th>$\varepsilon_{xm}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$</td>
<td>7.0</td>
<td>$5.37 \times 10^{-3}$</td>
<td>$2.31 \times 10^{-2}$</td>
</tr>
<tr>
<td>$7^2$</td>
<td>7.5</td>
<td>$2.31 \times 10^{-3}$</td>
<td>$1.04 \times 10^{-2}$</td>
</tr>
<tr>
<td>$12^2$</td>
<td>5.9</td>
<td>$7.16 \times 10^{-4}$</td>
<td>$2.41 \times 10^{-4}$</td>
</tr>
<tr>
<td>$22^2$</td>
<td>8.4</td>
<td>$2.08 \times 10^{-4}$</td>
<td>$8.32 \times 10^{-5}$</td>
</tr>
<tr>
<td>$42^2$</td>
<td>9.8</td>
<td>$5.45 \times 10^{-5}$</td>
<td>$3.23 \times 10^{-5}$</td>
</tr>
<tr>
<td>$82^2$</td>
<td>8.9</td>
<td>$1.10 \times 10^{-5}$</td>
<td>$5.65 \times 10^{-5}$</td>
</tr>
<tr>
<td>$122^2$</td>
<td>1.0</td>
<td>$7.06 \times 10^{-6}$</td>
<td>$2.94 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

The maximum absolute errors of $\hat{u}$ and $\hat{u}_x$ versus number of points $N$. As $N$ increases, accuracy improves.

$\varepsilon_m$: maximum absolute error of $\hat{u}$; $\varepsilon_{xm}$: maximum absolute error of $\partial \hat{u} / \partial x$.

Maximum errors of $\hat{u}$ and $\partial \hat{u} / \partial x$ are close to each other. The errors in the derivatives are even better for the local method than the global method.
We examine the quality of the solution based on varying number of points, $N$, on the unit square with $n = 5$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$c_o$</th>
<th>$\varepsilon_m$</th>
<th>$\varepsilon_{xm}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$</td>
<td>7.0</td>
<td>$5.37 \times 10^{-3}$</td>
<td>$2.31 \times 10^{-2}$</td>
</tr>
<tr>
<td>$7^2$</td>
<td>7.5</td>
<td>$2.31 \times 10^{-3}$</td>
<td>$1.04 \times 10^{-2}$</td>
</tr>
<tr>
<td>$12^2$</td>
<td>5.9</td>
<td>$7.16 \times 10^{-4}$</td>
<td>$2.41 \times 10^{-4}$</td>
</tr>
<tr>
<td>$22^2$</td>
<td>8.4</td>
<td>$2.08 \times 10^{-4}$</td>
<td>$8.32 \times 10^{-5}$</td>
</tr>
<tr>
<td>$42^2$</td>
<td>9.8</td>
<td>$5.45 \times 10^{-5}$</td>
<td>$3.23 \times 10^{-5}$</td>
</tr>
<tr>
<td>$82^2$</td>
<td>8.9</td>
<td>$1.10 \times 10^{-5}$</td>
<td>$5.65 \times 10^{-5}$</td>
</tr>
<tr>
<td>$122^2$</td>
<td>1.0</td>
<td>$7.06 \times 10^{-6}$</td>
<td>$2.94 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

The maximum absolute errors of $\hat{u}$ and $\hat{u}_x$ versus number of points $N$. As $N$ increases, accuracy improves.

- $\varepsilon_m$: maximum absolute error of $\hat{u}$;
- $\varepsilon_{xm}$: maximum absolute error of $\partial \hat{u} / \partial x$.

Maximum errors of $\hat{u}$ and $\partial \hat{u} / \partial x$ are close to each other. The errors in the derivatives are even better for the local method than the global method.
Example. We consider a simple Dirichlet jump boundary problem

\[
\frac{\partial u}{\partial t} (x, y, t) = \Delta u (x, y, t), \quad (x, y) \in \Omega, t > 0
\]

\[
u(x, y, t) = 0, \quad (x, y), t > 0 \in \Gamma,
\]

\[
u(x, y, 0) = 1, \quad (x, y) \in \Omega,
\]

where \( \Omega \) is square \([-0.5, 0.5] \times [-0.5, 0.5] \) with Dirichlet boundary conditions, the analytical solution is given by

\[
u(x, y, t) = T(x, t)T(y, t) \quad (28)
\]

where \( T \) is given by

\[
T(\xi, t) = \frac{4}{\pi} \sum_{i=0}^{\infty} \frac{(-1)^i}{2i + 1} \exp \left[-(2i + 1)^2 \pi^2 t\right] \cos \left[(2i + 1)\pi \xi\right], \ \xi = x, y.
\]
Analytical solutions at $t = 0.001\,\text{s}$:

The temperature field has a large gradient near the boundary of the domain.

Local method represents well for the solution of the problem with large gradient.
Numerical Examination

Local RBF Methods for Solving PDEs

Analytical solutions at $t = 0.001\text{s}$:

The temperature field has a large gradient near the boundary of the domain.

Local method represents well for the solution of the problem with large gradient.
Analytical solutions at $t = 0.001\text{s}$:

The temperature field has a large gradient near the boundary of the domain.

Local method represents well for the solution of the problem with large gradient.
We compare solutions using different methods: LMAPS & ELMAPS.

I. The comparison of LMAPS with various types of RBFs.

<table>
<thead>
<tr>
<th>( L )</th>
<th>( \Delta - \lambda^2 )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>TS</td>
<td>MQ</td>
</tr>
<tr>
<td>( 11^2 )</td>
<td>4.46\times10^{-2}</td>
<td>1.42\times10^{-3}</td>
</tr>
<tr>
<td>( 21^2 )</td>
<td>2.52\times10^{-2}</td>
<td>8.41\times10^{-4}</td>
</tr>
<tr>
<td>( 31^2 )</td>
<td>1.72\times10^{-2}</td>
<td>1.73\times10^{-3}</td>
</tr>
<tr>
<td>( 41^2 )</td>
<td>1.30\times10^{-2}</td>
<td>1.50\times10^{-3}</td>
</tr>
<tr>
<td>( 51^2 )</td>
<td>1.05\times10^{-2}</td>
<td>1.65\times10^{-3}</td>
</tr>
</tbody>
</table>

Error versus \( N \), where \( n = 9, \ c = 10, \ \delta_t = 10^{-3} \) at \( t = 0.1 \).

LMAPS with MQ and TPS using \( L = \Delta \) give similar errors and slightly better results than LMAPS using TPS and \( L = \Delta - \lambda^2 \).

LMAPS works for larger time steps than ELMAPS.
We compare solutions using different methods: LMAPS & ELMAPS.  

1. The comparison of LMAPS with various types of RBFs.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>TS</td>
<td>MQ</td>
</tr>
<tr>
<td>$11^2$</td>
<td>$4.46 \times 10^{-2}$</td>
<td>$1.42 \times 10^{-3}$</td>
</tr>
<tr>
<td>$21^2$</td>
<td>$2.52 \times 10^{-2}$</td>
<td>$8.41 \times 10^{-4}$</td>
</tr>
<tr>
<td>$31^2$</td>
<td>$1.72 \times 10^{-2}$</td>
<td>$1.73 \times 10^{-3}$</td>
</tr>
<tr>
<td>$41^2$</td>
<td>$1.30 \times 10^{-2}$</td>
<td>$1.50 \times 10^{-3}$</td>
</tr>
<tr>
<td>$51^2$</td>
<td>$1.05 \times 10^{-2}$</td>
<td>$1.65 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

LMAPS works for larger time steps than ELMAPS.
We compare solutions using different methods: LMAPS & ELMAPS.

I. The comparison of LMAPS with various types of RBFs.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\Delta - \lambda^2$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>TS</td>
<td>MQ</td>
</tr>
<tr>
<td>$11^2$</td>
<td>4.46×10$^{-2}$</td>
<td>1.42×10$^{-3}$</td>
</tr>
<tr>
<td>$21^2$</td>
<td>2.52×10$^{-2}$</td>
<td>8.41×10$^{-4}$</td>
</tr>
<tr>
<td>$31^2$</td>
<td>1.72×10$^{-2}$</td>
<td>1.73×10$^{-3}$</td>
</tr>
<tr>
<td>$41^2$</td>
<td>1.30×10$^{-2}$</td>
<td>1.50×10$^{-3}$</td>
</tr>
<tr>
<td>$51^2$</td>
<td>1.05×10$^{-2}$</td>
<td>1.65×10$^{-3}$</td>
</tr>
</tbody>
</table>

Error versus $N$, where $n = 9$, $c = 10$, $\delta_t = 10^{-3}$ at $t = 0.1$.

LMAPS with MQ and TPS using $L = \Delta$ give similar errors and slightly better results than LMAPS using TPS and $L = \Delta - \lambda^2$.

LMAPS works for larger time steps than ELMAPS.
We compare solutions using different methods: LMAPS & ELMAPS.

I. The comparison of LMAPS with various types of RBFs.

<table>
<thead>
<tr>
<th>( L )</th>
<th>( \Delta - \lambda^2 )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>TS</td>
<td>MQ</td>
</tr>
<tr>
<td>11^2</td>
<td>4.46 \times 10^{-2}</td>
<td>1.42 \times 10^{-3}</td>
</tr>
<tr>
<td>21^2</td>
<td>2.52 \times 10^{-2}</td>
<td>8.41 \times 10^{-4}</td>
</tr>
<tr>
<td>31^2</td>
<td>1.72 \times 10^{-2}</td>
<td>1.73 \times 10^{-3}</td>
</tr>
<tr>
<td>41^2</td>
<td>1.30 \times 10^{-2}</td>
<td>1.50 \times 10^{-3}</td>
</tr>
<tr>
<td>51^2</td>
<td>1.05 \times 10^{-2}</td>
<td>1.65 \times 10^{-3}</td>
</tr>
</tbody>
</table>

Error versus \( N \), where \( n = 9, \ c = 10, \ \delta_t = 10^{-3} \) at \( t = 0.1 \).

LMAPS with MQ and TPS using \( L = \Delta \) give similar errors and slightly better results than LMAPS using TPS and \( L = \Delta - \lambda^2 \).

LMAPS works for larger time steps than ELMAPS.
II. The comparison of ELMAPS with other local methods.
  ▶ Left: the errors for evenly spaced node at $t = 0.1$ s;
  ▶ Right: the average errors as function of time step lengths.

Numerical results are accurate with small time steps.
II. The comparison of ELMAPS with other local methods.

- Left: the errors for evenly spaced node at $t = 0.1$s;
- Right: the average errors as function of time step lengths.

Numerical results are accurate with small time steps.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.
Advantages of LMAPS and ELMAPS:

- easy to extend to high-dimensional space;
- suitable for solving large-scale problems;
- solvable for problems with irregular domains;
- complete mesh free method;
- insensitive to the shape parameters in MQ and IMQ RBFs;
- efficient to reduce computational cost.


I would like to thank my advisor, Dr. C. S. Chen, and Dr. Joseph Kolibal for their instructive guidance and useful suggestions and help over past few years. I also really appreciate the time that Dr. Jiu Ding, Dr. Haiyan Tian and Dr. Chaoyang Zhang have spent on my dissertation. Thanks for the support from Department of Mathematics during my PhD study at the USM.


Thank You

Department of Mathematics at
The University of Southern Mississippi

guangmingyao@gmail.com