RBF-FD Approximation to Solve Poisson Equation in 3D

Jagadeeswaran.R

March 14, 2014
Problem Setup

- Generalized finite difference method.
- Uses numerical differentiations generated by Gaussian RBF interpolation on irregular centers.
- Dirichlet problem for the Poisson Equation in 2D and 3D with smooth solution.
- RBF-QR method used when the Interpolation matrix is ill conditioned due to small "Shape parameters"
Problem Definition

- Dirichlet Problem for the Poisson equation:

\[ \Delta u = f \text{ on } \Omega \]

\[ u|_{\delta\Omega} = g \]

- We want to find a Linear numerical differentiation formula for the Differential operator:

\[ Du(x) \approx \sum_{i=1}^{n} w_i(x) u(x_i) \]

where \( w_i(x) \) called weights, The vector \( w = [w_1, ..., w_n]^T \) called stencil.
Problem Definition ...

- Let Discretization centers: $\Xi \subset \tilde{\Omega}$, $\delta \Xi := \Xi \cap \delta \Omega$, $\Xi_{int} := \Xi \setminus \delta \Xi$
- Assume for each $\zeta \in \Xi_{int}$ a set $\Xi_{\zeta} \subset \Xi$ is chosen such that $\zeta \in \Xi_{\zeta}$ and
  \[\Xi = \bigcup_{\zeta \in \Xi_{int}} \Xi_{\zeta}\]
- For each $\zeta \in \Xi_{int}$, choose a linear numerical differentiation formula for Laplace operator:
  \[\Delta u(\zeta) \approx \sum_{\Xi \in \Xi_{\zeta}} w_{\Xi,\zeta} u(\Xi)\]
- So we get the approximation:
  \[\sum_{\Xi \in \Xi_{\zeta}} w_{\Xi,\zeta} \hat{u}(\Xi) = f(\zeta), \ \zeta \in \Xi_{int}; \ \hat{u}(\Xi) = g(\Xi), \ \Xi \in \delta \Xi \quad (1)\]
My Goals

- Add Stable Computation Algorithm from 'G.E. Fasshauer, M.J. McCourt,'
- Add Support for 3 Dimensional Problems
- Generalize Stencil Selection for n Dimensional Domains
- Experiment Stable computation for n Dimensional Problems
- CVT Points for n Dimensions
Interpolation Problem

Let a Positive(or Conditionally) definite function $\kappa : \mathbb{R}_+ \to \mathbb{R}$ and a continuous function $u : \mathbb{R}^d \to \mathbb{R}$.

Given $\Xi_\zeta = \{x_0, \ldots, x_n\} \subset \mathbb{R}^d$, $\zeta = x_0$, $K_j(x) = K(x - x_j)$, $K(x) = \kappa(||x||)$ the RBF interpolant with a constant term is sought in the form

$$s(x) = \sum_{j=0}^{n} a_j K_j(x) + c, \quad s(x_i) = u(x_i), \quad i = 0, \ldots, n, \quad \sum_{j=0}^{n} a_j = 0 \quad (2)$$

Then the coefficients $a_j$ and $c$ are uniquely determined by the following conditions

$$\sum_{j=0}^{n} a_j K(x_i - x_j) + c = u(x_i), \quad i = 0, \ldots, n, \quad \sum_{j=0}^{n} a_j = 0$$

written in matrix form as

$$\begin{bmatrix} K_X & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} u_{|x} \\ 0 \end{bmatrix}, \quad K_X := [e^{-\epsilon^2 ||x_i - x_j||^2}]_{i,j=0}^n, \quad 1 := [1, \ldots, 1]^T \quad (3)$$
Consider the approximation of a differential operator $'D'$ by using RBF

$$\sum_{j=0}^{n} w_j K(x - x_j) + v = DK(x - x_i), \ i = 0, \ldots, n,$$

Note: This is indept of $u$

These weights can be found by solving the symmetric positive definite linear system

$$\begin{bmatrix} K_X & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} [Dk_i(x_0)]_{i=0}^n \\ 0 \end{bmatrix}$$

(4)

where, $\kappa_j(x) = K(x - x_j), \ K(x) := \kappa(||x||), \ K_X := [e^{-\epsilon^2 ||x_i - x_j||^2}]_{i,j=0}^n$.

For Gaussian RBF $\kappa(r) = e^{-\epsilon r^2}$. Its Laplacian is:

$$\Delta k(x) = 2\epsilon^2 e^{-\epsilon^2 ||x_i - x_j||^2} (2\epsilon^2 ||x_i - x_j||^2 - d), \ d := \text{num of dimensions}$$
Differentiation Problem

- The derivatives of $s$ are good approximations of the derivatives of $u$ if $k$ is sufficiently smooth.
- An approximation of $Du(x)$ at $x_0$, where $D$ is a linear differential operator annihilating constants, may be considered in the form. Apply $D$ to (2)

$$Du(x_0) \approx Ds(x_0) = \sum_{j=0}^{n} a_j DK(x_0 - x_j) = \sum_{i=0}^{n} w_i u(x_i)$$

where the weights $w_i$ (depending on $x$) exist because the coefficients $a_j$ of the interpolation function $s$ defined as before depend linearly on the data $u(x_i)$. This can be verified as below using (4)

$$\sum_{j=0}^{n} a_j DK(x - x_j) = \begin{bmatrix} a \\ c \end{bmatrix}^T \begin{bmatrix} [K(x - x_i)]_{i=0}^n \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix}^T \begin{bmatrix} Kx \\ 1^T \\ 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix}$$

$$= \begin{bmatrix} u_{x} \\ 0 \end{bmatrix}^T \begin{bmatrix} w \\ v \end{bmatrix} = \sum_{i=0}^{n} w_i u(x_i)$$
Example Stencil

Figure: typical Stencil Points in 2D

- Stencil used to compute Laplacian, in case of uniform points with 'h' fill distance

\[
\Delta u(\zeta) = \frac{1}{h^2} \left( u(\zeta + (h, 0)) + u(\zeta - (h, 0)) + u(\zeta + (0, h)) + u(\zeta - (0, h)) - 4u(\zeta) \right)
\]
Stencil Selection algorithm in $\mathbb{R}^2$ from ’Davydov’

- Given $\zeta \in \Xi \setminus \delta \Xi$, we will select a set $\Xi_\zeta$ containing $\zeta$.
- Let $\Xi_\zeta = \{x_0, x_1, ..., x_k\}$, $\zeta := x_0$, where the points $x_0, x_1, ..., x_k$ are ordered counterclockwise with respect to $\zeta$.
- Consider the following cost function: $\mu(x_1, ..., x_k) := \sum_{i=1}^{k} \alpha_i^2$ where $\alpha_i$ denotes the angle between the rays $\zeta x_i, \zeta x_{i+1}$ in the counterclockwise direction.
- We will also need the minimum and the maximum angle: $\alpha_{\min} = \min\{\alpha_1, ..., \alpha_k\}$ $\alpha_{\max} = \max\{\alpha_1, ..., \alpha_k\}$
- Since $\sum_{i=1}^{k} \alpha_i = 2\pi$, the expression $\sum_{i=1}^{k} \alpha_i^2$ achieves its unique minimum for $\alpha_1 = ... = \alpha_k = 2\pi/k$, that is for the uniformly spaced directions $\zeta x_i$ if $x_1, ..., x_k$ were chosen freely in $\mathbb{R}^2$.
- So the goal of the algorithm below is to choose $\{x_1, ..., x_k\} \in \Xi$ such that $\mu(x_1, ..., x_k)$ is minimised while keeping the distances $||x_i - \zeta||$ as small as possible.
- To achieve a balance between the goals of a small $\mu$ and small distances, we introduce the restriction that $x_i$ must be among $'m'$ closest points to $\zeta$, and terminate the algorithm if the set $\{\zeta, x_1, ..., x_k\}$ satisfies $\alpha_{\max} \leq u \alpha_{\min}$, where $m > k$ and $u > 1.0$ are parameters to be determined empirically.
Stencil Selection algorithm in $\mathbb{R}^2$ from 'Davydov'...

Input: $\Xi, \zeta$. Output: $\Xi_\zeta$. Parameters: $k$ (the target number of points $x_i$), $m > k$ (the number of points in the local cloud) and $u > 1$ (the angle uniformity tolerance). Parameter values used in our numerical experiments: $k = 6$, $m = 30$, $u = 3.0$.

I Find $m$ nearest points $\{x_1, \ldots, x_m\}$ closer to $\zeta$ in $\Xi \setminus \zeta$, sorted by increasing distance to $\zeta$, and initialise $\Xi_\zeta := \{\zeta, x_1, \ldots, x_k\}$. If $\alpha_{\max} \leq u \alpha_{\min}$, then STOP: return $\Xi_\zeta$.

II For $i = n + 1, \ldots, m$:
1. Compute the angles $\alpha_1', \ldots, \alpha_{k+1}'$ formed by the extended set $\{x_1', \ldots, x_{k+1}'\} = \{x_1, \ldots, x_k, x_i\}$.
2. If both angles between $\zeta x_i$ and its two neighbouring rays are greater than the minimum angle $\alpha_{\min}' := \alpha_{\min}(x_1, \ldots, x_k, x_i)$:
   i. Find $j$ such that $\alpha_j' = \alpha$. Choose $p = j$ or $p = j + 1$ depending on whether $\alpha_{j-1}' < \alpha_{j+1}'$ or $\alpha_{j-1}' \geq \alpha_{j+1}'$.
   ii. If $\mu(x_1', \ldots, x_{k+1}', x_p') < \mu(x_1, \ldots, x_k)$:
      a. Update $\{x_1, \ldots, x_k\} = \{x_1', \ldots, x_{k+1}'\} \setminus \{x_p'\}$.
      b. If $\alpha_{\max} \leq u \alpha_{\min}$ STOP: return the current set $\Xi_\zeta = \{\zeta, x_1, \ldots, x_k\}$.

III Observe that $\alpha_{\max} > u \alpha_{\min}$ must hold for the current set $\Xi_\zeta = \{\zeta, x_1, \ldots, x_k\}$ if the algorithm has not been terminated earlier. Find $j$ such that $\alpha_j = \alpha_{\min}(x_1, \ldots, x_k)$. Choose $p = j$ or $p = j + 1$ depending on whether $\alpha_{j-1} < \alpha_{j+1}$ or $\alpha_{j-1} \geq \alpha_{j+1}$. STOP: return $\Xi_\zeta = \{\zeta, x_1, \ldots, x_k\} \setminus \{x_p\}$. 

11 / 28
Stencil Selection algorithm in $\mathbb{R}^3$

- Input: $\Xi, \zeta$. Output: $\Xi_\zeta$.
- Parameter values used in our numerical experiments: $k = 12, m = 30, u = 2.25$.
- Cost function: Standard Deviation in Sum of Squared Distance between each other points in the Stencil (Energy). $\mu(x_1, ..., x_k) := \text{std}(\{E_i : i = 1, ..., k\})$
  where $E_i := E(x_i) = \sum_{j=1, i \neq j} \|x_i - x_j\|^2$
- Another option: Projected Distance: shift the current set of stencil point to make $\zeta$ as the center and normalize, i.e. $x'_i = \frac{(x_i - \zeta)}{\|x_i - \zeta\|} : i = 1, ..., k$, then compute the Squared Distance between each other points.

I Find $m$ nearest points $\{x_1, ..., x_m\}$ closer to $\zeta$ in $\Xi \setminus \zeta$, sorted by increasing distance to $\zeta$, and initialise $\Xi_\zeta := \{\zeta, x_1, ..., x_k\}$. If $E_{\text{max}} \leq uE_{\text{min}}$, then STOP: return $\Xi_\zeta$.

II For $i = n + 1, ..., m$:
1. Compute the angles $E'_1, ..., E'_{k+1}$ formed by the extended set $\{x'_1, ..., x'_{k+1}\} = \{x_1, ..., x_k, x_i\}$.
2. Find $p = j$: such that $E_j = \text{min}(\{E_i\})$
3. If $\mu(x'_1, ..., x'_{k+1} x'_p) < \mu(x_1, ..., x_k)$:
   a. Update $\{x_1, ..., x_k\} = \{x'_1, ..., x'_{k+1}\} \setminus \{x'_p\}$.
   b. If $E_{\text{max}} \leq uE_{\text{min}}$
      STOP: return the current set $\Xi_\zeta = \{\zeta, x_1, ..., x_k\}$.

III STOP: return $\Xi_\zeta = \{\zeta, x_1, ..., x_k\}$. 


CVT: Centroidal Voronoi Points

- Algorithm based iterative voronoi tessellation
- There is an initial assignment for the generators [Halton, random, uniform ...]
- In each iteration, estimates is made of the volume and location of the Voronoi cells
- Sampling done by Monte Carlo sampling
- The accuracy of the resulting CVT depends in part on the number of sampling points and the number of iterations taken.
- Lloyd’s algorithm
- Can be customized to generate points for any dimensions, regions
- Boundary points placements Algorithm called: 'Constrained CVT'
- CCVTBOX software
CVT points generation

Initial generators

Final generators
Since the matrix $K_x$ is extremely ill-conditioned for small $\epsilon$, we need to use Stable computation in this case.

Hilbert-Schmidt series:

$$K(x, z) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(x) \varphi_n(z)$$

When we have such an Expansion, we truncate the series to $M$ such that $K(x_i, x_j) = \sum_{n=1}^{M} \lambda_n \varphi_n(x_i) \varphi_n(x_j)$ and write as a finite matrix: So we get $K = \Phi \Lambda \Phi^T$
New Basis

- Since $\lambda$ decays in power $n$, it leads to ill conditioned interpolation matrix when $\epsilon$ is small.

Note:

$$\lambda_n = \sqrt{\frac{\alpha^2}{\alpha^2 + \delta^2 + \epsilon^2}} \left(\frac{\epsilon^2}{\alpha^2 + \delta^2 + \epsilon^2}\right)^{n-1}$$

- We want to find a new Basis which improves the conditioning of the problems.
- Compute $\Phi = QR = Q (R_1 | R_2)$
- $K = (QR) \Lambda \Phi^T = Q (R_1 | R_2) \Lambda \Phi^T$
- With more Simplifications we get a new basis: $\Psi$ as shown next.
Finding New Basis $\Psi$

\[
K = \Phi \Lambda \Phi^T
\]
\[
= Q (R_1 R_2) \begin{pmatrix} \Lambda_1 & \Lambda_2 \end{pmatrix} \Phi^T
\]
\[
= Q \begin{pmatrix} R_1 \Lambda_1 & R_2 \Lambda_2 \end{pmatrix} \Phi^T
\]
\[
= QR_1 \Lambda_1 \begin{pmatrix} I_N & \Lambda_1^{-1} R_1^{-1} R_2 \Lambda_2 \end{pmatrix} \Phi^T
\]
\[
= QR_1 \Lambda_1 \begin{pmatrix} I_N & \Lambda_1^{-1} R_1^{-1} R_2 \Lambda_2 \end{pmatrix} \begin{pmatrix} \Phi_1^T \\ \Phi_2^T \end{pmatrix}
\]
\[
= QR_1 \Lambda_1 \begin{pmatrix} \Phi_1^T + \Lambda_1^{-1} R_1^{-1} R_2 \Lambda_2 \Phi_2^T \end{pmatrix}
\]
\[
= X \begin{pmatrix} \Phi_1^T + \Lambda_1^{-1} R_1^{-1} R_2 \Lambda_2 \Phi_2^T \end{pmatrix} = \Psi
\]

So Now we have a new basis, more stable due to $\Lambda_1^{-1} \ast \Lambda_2$:

\[
\Psi = X^{-1} K
\]
\[
= \Phi_1^T + \Lambda_1^{-1} R_1^{-1} R_2 \Lambda_2 \Phi_2^T
\]
Computation of RBF-FD Stencils weights by QR Method

Better Conditioned basis

With the new basis, modify the weight computation matrix. By left-multiplying both sides of (4) by:

\[
\begin{pmatrix}
(QR_1 \Lambda_1)^{-1} & 0 \\
0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
(K_X & 1^T \\
1 & 0
\end{pmatrix}
\begin{bmatrix}
w \\ v
\end{bmatrix} = \begin{pmatrix}
(QR_1 \Lambda_1)^{-1} & 0 \\
0 & 1
\end{pmatrix}
\begin{bmatrix}
[\Delta k_i(x_0)]^{n}_{i=0} \\
0
\end{bmatrix}
\]

\[
\begin{pmatrix}
(\psi_X & h \\
1^T & 0
\end{pmatrix}
\begin{bmatrix}
w \\ v
\end{bmatrix} = \begin{pmatrix}
[\Delta \psi_i(x_0)]^{n}_{i=0} \\
0
\end{bmatrix}
\]

(5)

where, \( h = (QR_1 \Lambda_1)^{-1} 1 \)

Are We done yet?

Computing \( h \) is still problematic, since it involves \( \Lambda_1^{-1} \)
Numerical Computation of RBF-FD Stencils weights by QR Method

For numerical implementation $\Psi$ is truncated:

$$
\begin{bmatrix}
\tilde{\psi}_X & h \\
1^T & 0
\end{bmatrix}
\begin{bmatrix}
w \\
v
\end{bmatrix}
= 
\begin{bmatrix}
[\Delta \tilde{\psi}_i(x_0)]_{i=0}^n \\
0
\end{bmatrix}
$$

(6)

where,

$$
\tilde{\psi}_X :=
\begin{bmatrix}
\tilde{\psi}_0(x_0) & \ldots & \tilde{\psi}_0(x_n) \\
\vdots & \ddots & \vdots \\
\tilde{\psi}_n(x_0) & \ldots & \tilde{\psi}_n(x_n)
\end{bmatrix}
$$

To improve the stability, following are done: replace $h$ by $\tilde{h} = (QR_1\tilde{\Lambda}_1)^{-1}1$ where $\tilde{\Lambda}_1$ is obtained by zero replacing the elements in $\Lambda_1$ that exceed the reciprocal of machine epsilon. The above equation is replaced by

$$
\sum_{j=1}^n (\tilde{\psi}_i(x_j) - \tilde{\psi}_i(x_0))w_j + \tilde{h}_i v = \Delta \tilde{\psi}_i(x_0), \ i = 0, \ldots, n. \ w_0 = -\sum_{j=1}^n w_j
$$
Numerical Computation of RBF-FD Stencils weights by QR Method

\[
\begin{bmatrix}
\tilde{\psi}_0(x_1) & \tilde{\psi}_0(x_2) & \cdots & \tilde{\psi}_0(x_n) & \tilde{h}(0) \\
\tilde{\psi}_1(x_1) & \tilde{\psi}_1(x_2) & \cdots & \tilde{\psi}_1(x_n) & \tilde{h}(1) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\tilde{\psi}_n(x_1) & \tilde{\psi}_n(x_2) & \cdots & \tilde{\psi}_n(x_n) & \tilde{h}(n)
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_n
\end{bmatrix}
= 
\begin{bmatrix}
\Delta \tilde{\psi}_0(x_0) \\
\Delta \tilde{\psi}_1(x_0) \\
\vdots \\
\Delta \tilde{\psi}_n(x_0)
\end{bmatrix}
\]

(7)

Also the Stencil points \(\{x_0, x_1, x_2, \ldots, x_n\}\) are centered to \(x_0\): \(\{x_0 - x_0, x_1 - x_0, x_2 - x_0, \ldots, x_n - x_0\}\). So that we have \(\{0, x'_1, x'_2, \ldots, x'_n\}\). This simplifies the computation of \(\Delta \tilde{\psi}_i(x_0)\) as \(x_0 = 0\)
Open Questions

- Points Selection?
- it does not work for Quasi random points (Sobol, Halton) !!
- Adaptive points placements, more points closer boundary, corners
- Boundary points placements?
- How many points required for the Stencil?
- Stencil selection criterion?
What it does?

- Computes Mesh Triangulations upto 5 refinements, either in Square or Circle Domain
- Computes Stencil for each Interior point in the domain
- For each $\epsilon$ compute the Stencil weights using Direct Method if the condition number of the interpolation is $< 10^{12}$
- Else use the Stable Method to compute the weights
- For the given "Problem" computes the Finite Difference Matrix
- solves the Problem using the above matrix
- Computes RMS Error between the actual function and approximation
- Plots the Error values the defined range of $\epsilon$
RMS Error for the 2D test function

**Figure:** RMS Error in Function approx, RMS Error in Differentiation
RMS Error for the 3D test function

Figure: RMS Error in Function approx, RMS Error in Differentiation
RMS Error for the 4D test function

Figure: RMS Error in Function approx, RMS Error in Differentiation
Conclusion

- Implemented Davydov’s 2D Poisson solver method
- Integrated with Stable computation of Fasshauer
- Extended to 3D support
- Next planning to work on Survey of similar work done so far
- Work on answering some theoretical questions??
- Adaptive meshless refinement
- Hermite interpolation
- Solve some real and difficult PDEs
Reference


- [http://math.iit.edu/~mccomic/gaussqr/](http://math.iit.edu/~mccomic/gaussqr/)

- CVT Points: [http://people.sc.fsu.edu/~jburkardt/m_src/cvt/cvt.html](http://people.sc.fsu.edu/~jburkardt/m_src/cvt/cvt.html)