Adaptive Radial Basis Function Methods with Residual Subsampling Technique for Interpolation and Collocation Problems

Alfa Heryudono    Tobin Driscoll

Department of Mathematical Sciences
University of Delaware

SIAM Annual Meeting July 13 2006
Boston, Massachusetts
Given data at nodes $x_1, ..., x_N$ in $d$ dimensions, the basic form for an RBF approximation is

$$F(x) = \sum_{j=1}^{N} \lambda_j \phi(\epsilon_j \| x - x_j \|),$$

where $\| \cdot \|$ denotes the Euclidean distance between two points and $\phi(r) = \sqrt{1 + r^2}$ is defined for $r \geq 0$. 
$f_i = f(x_i)$

\[ A \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \]

where $a_{ij} = \phi(\epsilon_j \|x_i - x_j\|)$. Nonsingularity of $A$ is guaranteed for many choices of $\phi$ with mild restrictions and constant shape parameters $\epsilon_j$. 
Advantages of RBF methods

- No need for a mesh / triangulation.
- Simple implementation and dimension independence.
- No staircasing / polygonization for boundaries.
- Depending on chosen RBFs, high-order/spectral convergence can be achieved.
- Easy to implement derivatives and boundary conditions.
Challenges using RBF methods

- As the number of centers grows, the method needs to solve a relatively large algebraic system.
- The matrix is full (except for compactly supported RBF).
- Choosing nodes and shape parameters.
- Ill-conditioning usually makes spectral convergence difficult to achieve.
Problems involve
- geometry
- steep gradients
- corners
- topological changes resulting from nonlinearity
- high degrees of localization in space and/or time

Goal
Obtain an accurate solution using a minimal number of automatically chosen nodes.

Residual Subsampling Scheme

1. Compute Interpolant
   - based on known centers and its values

2. Refine/Coarsen nodes
   - based on residuals of interpolation/PDEs evaluated at a finer point set

3. Shape parameter adjustment
   - based on the node spacings to prevent the growth of condition numbers
Runge Function

\[ N = 13, \text{ Max error } = 1.25e-02. \]

MATLAB CODE

```matlab
% MATLAB CODE

N = 13; Max error = 1.25e−02.

\[ f(x) = \frac{1}{1 + 25 \cdot x^2}; \]
\[ \text{phi}(r, \epsilon) = \sqrt{(\epsilon r)^2 + 1}; \]
\[ x = \text{linspace}(-1,1,N); \]
\[ \text{ref} = \text{true}; \]
\[ \text{while any(ref)} \]
\[ \text{N} = \text{length}(x); \text{dx} = \text{diff}(x); \]
\[ \epsilon = 0.75 \cdot \text{min}(\text{Inf}, \text{1/dx}); \]
\[ y = x(1:N-1) + 0.5 \cdot \text{dx}; \]
\[ A = \text{zeros}(N); B = \text{zeros}(N-1,N); \]
\[ \text{for } j=1:N \]
\[ A(:,j) = \text{phi}(x-x(j), \epsilon(j)); \]
\[ B(:,j) = \text{phi}(y-x(j), \epsilon(j)); \]
\[ \text{end} \]
\[ \lambda = A \backslash f(x); \text{resid} = \text{abs}(B \ast \lambda - f(y)); \]
\[ \text{ref} = \text{resid} > \text{thetar}; x = \text{sort}([x;y(ref)]); \]
\[ \text{coarsen} = \text{resid}(1:N-2) < \text{thetac} \& \ldots \]
\[ \text{resid}(2:N-1) < \text{thetac}; \]
\[ \text{coarsen} = 1 + \text{find}(\text{coarsen}); x(\text{coarsen}) = []; \]
```

Heryudono, Driscoll (UD)
Adaptive RBF Methods
SIAM AN06 7 / 14
Runge Function

N = 25, Max error = 4.95e-04.

% MATLAB CODE
\[
\text{thetar} = 2e^{-5}; \text{thetac} = 1e^{-8}; N = 13;
\]
\[
f = @(x) 1/(1+25*x.^2);
\]
\[
\text{phi} = @(r, \text{epsilon}) \sqrt{(\text{epsilon} \cdot r)^2 + 1};
\]
\[
x = \text{linspace}(-1,1,N)';
\]
\[
\text{ref} = \text{true};
\]
\[
\text{while any( ref )}
\]
\[
N = \text{length}( x ); dx = \text{diff}( x );
\]
\[
\text{epsilon} = 0.75 * \min([\text{Inf}; 1/dx],[1/dx; \text{Inf}]);
\]
\[
y = x(1:N-1) + 0.5*dx;
\]
\[
A = \text{zeros} (N); B = \text{zeros} (N-1,N);
\]
\[
\text{for } j=1:N
\]
\[
A(:,j) = \text{phi}(x-x(j), \text{epsilon}(j));
\]
\[
B(:,j) = \text{phi}(y-x(j), \text{epsilon}(j));
\]
\[
\end
\]
\[
\text{lambda} = A \backslash f(x); \text{resid} = \text{abs}(B \ast \text{lambda} - f(y));
\]
\[
\text{ref} = \text{resid} > \text{thetar}; x = \text{sort}([x;y( \text{ref} )]);
\]
\[
\text{coarsen} = \text{resid}(1:N-2) < \text{thetac} \& \ldots
\]
\[
\text{resid}(2:N-1) < \text{thetac};
\]
\[
\text{coarsen} = 1+\text{find}(\text{coarsen}); x(\text{coarsen}) = [];\]
\[
\end\]
N = 41, Max error = 1.03e−04.

% MATLAB CODE

```matlab
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1 ./ (1 + 25 * x.^2);
phi = @(r, epsilon) sqrt((epsilon * r).^2 + 1);
x = linspace(1, 1, N);
ref = true;
while any(ref)
    N = length(x); dx = diff(x);
    epsilon = 0.75 * min([Inf; 1./dx], [1./dx; Inf]);
    y = x(1:N-1) + 0.5*dx;
    A = zeros(N); B = zeros(N-1,N);
    for j = 1:N
        A(:,j) = phi(x-x(j), epsilon(j));
        B(:,j) = phi(y-x(j), epsilon(j));
    end
    lambda = A \ f(x); resid = abs(B*lambda - f(y));
    ref = resid > thetar; x = sort([x; y(ref)]);
    resid(2:N-1) < thetac;
    coarsen = 1 + find(coarsen); x(coarsen) = [];
end
```

Heryudono, Driscoll (UD)

Adaptive RBF Methods
Runge Function

N = 47, Max error = 5.31e-05.

% MATLAB CODE

```matlab
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r, epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(1,1,N)';
ref = true;

while any(ref)
    N = length(x); dx = diff(x);
    epsilon = 0.75*min([Inf; 1./dx],[1./dx; Inf]);
    y = x(1:N-1) + 0.5*dx;
    A = zeros(N); B = zeros(N-1,N);
    for j = 1:N
        A(:,j) = phi(x-x(j), epsilon(j));
        B(:,j) = phi(y-x(j), epsilon(j));
    end
    lambda = A\f(x); resid = abs(B*lambda-f(y));
    ref = resid > thetar; x = sort([x;y(ref)]);
    resid(1:N-2) < thetac & 
    resid(2:N-1) < thetac;
    coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

Heryudono, Driscoll (UD)
Adaptive RBF Methods
SIAM AN06 7 / 14
Runge Function

N = 49, Max error = 2.63e−05.

**MATLAB CODE**

```matlab
theta_r = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r, epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N);
ref = true;
while any(ref)
    N = length(x); dx = diff(x);
    epsilon = 0.75*min([Inf; 1./dx],[1./dx;Inf]);
y = x(1:N-1) + 0.5*dx;
A = zeros(N); B = zeros(N-1,N);
for j = 1:N
    A(:,j) = phi(x-x(j), epsilon(j));
    B(:,j) = phi(y-x(j), epsilon(j));
end
lambda = A\f(x); resid = abs(B*lambda-f(y));
ref = resid > theta_r; x = sort([x;y(ref)]);
coarsen = resid(1:N-2) < thetac & ... 
        resid(2:N-1) < thetac;
coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

Heryudono, Driscoll (UD)  Adaptive RBF Methods  SIAM AN06  7 / 14
Runge Function

\[ N = 51, \text{Max error} = 2.05 \times 10^{-5}. \]

\[
\begin{align*}
\text{MATLAB CODE} \\
\text{theta}_r &= 2 \times 10^{-5}; \\text{theta}_c = 1 \times 10^{-8}; N = 13; \\
f &= @(x) 1/(1+25x^2); \\
\phi &= @(r,\text{epsilon}) \sqrt{(\text{epsilon}r)^2 + 1};
\end{align*}
\]

\[
\begin{align*}
x &= \text{linspace}(-1,1,N)'; \\
\text{ref} &= \text{true}; \\
\text{while} \ \text{any(ref)} \\
N &= \text{length}(x); \text{dx} = \text{diff}(x); \\
\epsilon &= 0.75 \times \text{min}([\text{Inf};1./\text{dx}],[1./\text{dx};\text{Inf}]); \\
y &= x(1:N-1) + 0.5 \times \text{dx}; \\
A &= \text{zeros}(N); B &= \text{zeros}(N-1,N); \\
\text{for} \ j=1:N \\
A(:,j) &= \phi(x-x(j),\epsilon(j)); \\
B(:,j) &= \phi(y-x(j),\epsilon(j)); \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\lambda &= A\backslash f(x); \text{resid} = \text{abs}(B\lambda-f(y)); \\
\text{ref} &= \text{resid} > \text{theta}_r; x &= \text{sort}([x;y(\text{ref})]); \\
\text{coarsen} &= \text{resid}(1:N-2) < \text{theta}_c \& \cdots \\
&\text{resid}(2:N-1) < \text{theta}_c; \\
\text{coarsen} &= 1+\text{find}(\text{coarsen}); x(\text{coarsen}) = []; \\
\text{end}
\end{align*}
\]
Runge Function

N = 53, Max error = 1.34e−05.

MATLAB CODE

```matlab
% MATLAB CODE
theta_r = 2e−5; theta_c = 1e−8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r, epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(−1,1,N);
ref = true;
while any(ref)
    N = length(x); dx = diff(x);
    epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
y = x(1:N−1) + 0.5*dx;
A = zeros(N); B = zeros(N−1,N);
    for j=1:N
        A(:,j) = phi(x−x(j), epsilon(j));
        B(:,j) = phi(y−x(j), epsilon(j));
    end
lambda = A\f(x); resid = abs(B*lambda−f(y));
ref = resid > theta_r; x = sort([x;y(ref)]);
    resid(2:N−1) < theta_c;
coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

Heryudono, Driscoll (UD)  Adaptive RBF Methods
\[
tanh(60x - .01)
\]

\[
\begin{array}{cccc}
1 & 11 & 0 & 18 \\
2 & 29 & 0 & 34 \\
3 & 63 & 0 & 31 \\
4 & 94 & 3 & 30 \\
5 & 121 & 4 & 12 \\
6 & 129 & 2 & 2 \\
7 & 129 & 0 & 0 \\
\end{array}
\]

\[\kappa(A) \text{ and } \| \cdot \|_{\infty}\]

\(N_r, N_c = \text{Number of centers to be added/removed respectively.}\)
2-D Case

Scheme

1. Initial coarse collection of nonoverlapping regular boxes in $\mathbb{R}^d$ that cover the domain $\Omega$ of interest.
2. Geometric adaptation.
3. Refining/Coarsening steps.
Poisson Equation with Dirichlet condition

\[ \nabla^2 u(x,y) = 0 \]

N=678

N=2100
Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \hat{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]

T = 0.000, N = 378.
Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \hat{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]

T = 0.010, N = 764.
Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \tilde{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]
Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]
Step and Adapt / Method of Lines

Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]

\[ T = 0.810, \ N = 470. \]
Burgers’ Equation

\[ \nu \Delta u - \nabla f(u) \cdot \hat{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{1}{2} u^2 \]

\[ \nu = 2 \times 10^{-3} \]

\[ T = 1.190, \ N = 356. \]
Buckley-Leverett

\[
\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}
\]

\[u = 0 \text{ on } \partial \Omega\]

\[f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2}\]

\[\nu = 10^{-3} \mu = \frac{1}{2}\]
Buckley-Leverett

\[ \nu \Delta u - \nabla f(u) \cdot \bar{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2} \]

\[ \nu = 10^{-3} \quad \mu = \frac{1}{2} \]

\[ u(x,t) \]

\[ T = 0.010, \quad N = 538. \]
Buckley-Leverett

\[ \nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t} \]
\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2} \]
\[ \nu = 10^{-3} \mu = \frac{1}{2} \]

\[ T = 0.050, \ N = 642. \]
Buckley-Leverett

\[ \nu \Delta u - \nabla f(u) \cdot \mathbf{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2} \]

\[ \nu = 10^{-3}, \mu = \frac{1}{2} \]

T = 0.100, N = 942.
Buckley-Leverett

\[ \nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2} \]

\[ \nu = 10^{-3}, \mu = \frac{1}{2} \]

\( T = 0.150, N = 1070. \)
Buckley-Leverett

\[ \nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t} \]

\[ u = 0 \text{ on } \partial \Omega \]

\[ f(u) = \frac{u^2}{u^2 + \mu (1 - u)^2} \]

\[ \nu = 10^{-3}, \mu = \frac{1}{2} \]

\[ T = 0.200, N = 1177. \]
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]
\[ u(0, t) = u(1, t) = 0 \]
\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \]

where, \( \nu = 10^{-3} \)
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]

\[ u(0, t) = u(1, t) = 0 \]

\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \]

where, \( \nu = 10^{-3} \)
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]

\[ u(0, t) = u(1, t) = 0 \]

\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \]

where, \( \nu = 10^{-3} \)
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]

\[ u(0, t) = u(1, t) = 0 \]

\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \]

where, \( \nu = 10^{-3} \)
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]

\[ u(0, t) = u(1, t) = 0 \]

\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \]

where, \( \nu = 10^{-3} \)
1-D Burgers’ Equation

\[ \nu u_{xx} - uu_x = u_t, \quad 0 < x < 1 \]

\[ u(0, t) = u(1, t) = 0 \]

\[ u(x, 0) = \sin(2\pi x) + \frac{1}{2} \sin(\pi x). \]

where, \( \nu = 10^{-3} \)
Things to be done

- Theory and model problems.
- Stability and Accuracy.
- Finding the best way to choose shape parameters.
- Applications (e.g. Lubrication theory in human eye).