Adaptive Node Selection in
Periodic Radial Basis Function Interpolations

Muhammad Shams
Dept. of Mathematics
UMass Dartmouth
Dartmouth MA 02747
Email: mshams@umassd.edu
December 19, 2011

Abstract

In RBFs, known center points are used to interpolate an approximation. Generally, these centers are equidistant from one another, but RBFs are not restricted to equidistant point meshes. In certain functions, the function’s slope can quickly become sharp. In these cases, the interpolation might suffer if an equidistant spacing is used. For these functions, choosing nodes centralized in the area of sharpness would help reduce error.

RBF Interpolations are not inherently periodic but they can become so by mapping the basis function onto a periodic function. By using a periodic RBF, we can interpolate periodic functions more accurately. Taking this a step further, we can apply the previously mentioned adaptive node selection to the periodic RBF.

1 Introduction

This project was undertaken in order to learn more about the RBF Interpolation process. Under the guidance of Dr. Alfa Heryudono I began looking into an adaptive way of selecting node points in RBFs. Before getting too far we came to the idea, what if we were working with periodic RBFs. The intention of this project is to combine these ideas and show where they may lead. In this paper, the techniques involved will be explained, numerical results will be shown, and some conclusions and speculation will be presented.

2 Radial Basis Function

2.1 RBF Background

Radial Basis Function Interpolation is a technique on the rise in the mathematical community. RBFs have already proven useful in a number of areas. As global approximations, RBFs can be used for scattered data problems in higher dimensions and for solving partial differential equations.

RBFs are a family of functions dependent on the distance from point to point or the distance from center points. The distance, \( r \), is taken as \( (x_i - x_j) \). Due to RBFs being based on distances, they have the property of being meshless. They do not need to be interpolated over equally spaced points. RBFs employ the use of a shape parameter designated as \( \epsilon \) that play a pivotal role in approximations. The shape parameter is a measure of how flat the basis function approximating the actual function is. As \( \epsilon \) approaches zero the basis function flattens out and loosens curvature. In Figure 1 we can see the effect on the Multiquadric basis function of the shape parameter.
Multiquadric

\[ \phi_j = \sqrt{1 + \epsilon^2(x_i - x_j)^2} \]  

Gaussian

\[ \phi_j = e^{-\epsilon(x_i - x_j)^2} \]  

Inverse Multiquadric

\[ \phi_j = \frac{1}{\sqrt{1 + \epsilon^2(x_i - x_j)^2}} \]  

Inverse Quadratic

\[ \phi_j = \frac{1}{1 + \epsilon^2(x_i - x_j)^2} \]

2.2 RBF Interpolation

The approximation of a function using RBF Interpolation is done by, first, building an Interpolation matrix. The Interpolation matrix is a linear combination of basis vectors. The matrix is created using one of the RBF basis and a set number of points. The matrix multiplied by the unknown coefficients, \( \lambda \), equals the real values of the approximated function.

\[
\begin{bmatrix}
\phi_1, x_1 & \phi_2, x_1 & \ldots & \phi_n, x_1 \\
\phi_1, x_2 & \phi_2, x_2 & \ldots & \phi_n, x_2 \\
\vdots & \vdots & \ddots & \vdots \\
\phi_1, x_n & \phi_2, x_n & \ldots & \phi_n, x_n
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n
\end{bmatrix}
= 
\begin{bmatrix}
f_1 \\
f_2 \\
\vdots \\
f_n
\end{bmatrix}
\]
\begin{equation}
\lambda = M^{-1} f
\end{equation}

Next, the inverse of the interpolation matrix is multiplied by the function values in order to obtain the value of the coefficients. After which, the interpolation matrix and the coefficients are multiplied over a new set of points, in order to build the approximated value of the function.

### 3 Periodicity

#### 3.1 Need for Periodicity

RBFs are not periodic and as such, higher error is observed when interpolating periodic functions. A periodic interpolation would reduce error throughout the entire interval and would alleviate Gibbs oscillations at the boundaries. Since it is assumed the interval does not end but rather connects back to the beginning, the last point to be interpolated is left out. The last point would have the same value as the initial point. A drawback of the periodic RBF is that it cannot interpolate a non-periodic function. The assumptions about the starting and ending point of a function can confuse the approximation. We can see from Figure 2 and Figure 3 the difference mapping a basis onto the periodic domain makes.

![Figure 2: Multiquadric Basis](image1)

![Figure 3: Periodic Multiquadric Basis](image2)

#### 3.2 Change of Basis

RBFs can achieve periodicity by being mapped onto a periodic function. In this case we mapped \( x \) to \( \cos(\theta) \). By doing so we come to the following formula for the Multiquadric basis:

\begin{equation}
\phi_j = \sqrt{1 + \epsilon^2 (\cos(\theta_i) - \cos(\theta_j))^2}
\end{equation}

The other basis mentioned can be mapped similarly. We find that for stability purposes a better form of the basis is derived from observing that along a unit circle, the distance from the origin to any point on the circle is one. Creating a triangle connected by two points along the circle and the origin, we can find the distance between the two points by using the Law of Cosines.
\[ c^2 = a^2 + b^2 - 2abc\cos(\theta) \] (8)

Working with a unit circle \( a \) and \( b \) are equal to 1. We interpret \( c^2 \) as the square of the distance between the two points or \( r^2 \). Substituting this new form of \( r^2 \) back into our periodic basis we get our new formula:

\[ \phi_j = \sqrt{1 + \epsilon^2(2 - 2(\cos(\theta) - \cos(\theta)_j))} \] (9)

The new interval for our periodic RBF is \( \theta \) from 0 to \( 2\pi \).

### 3.3 Error

As we increase \( N \), the errors of the RBF approximations decay in both cases of interpolating \( \sin(\theta) \), but the error remains large near the bounds in the non-periodic interpolation. In Figure 4, the pointwise errors (in logscale) of the Multiquadric RBF, with \( \epsilon = .85 \) over the interval \([0, 2\pi]\), are shown for \( N = 16 \) (blue), 32 (red), 48 (green), 64 (magenta), 80 (cyan), and 96 (black). On the right, in Figure 5, we have the pointwise errors (in logscale) of the periodic Multiquadric RBF with the same conditions.

![Figure 4: Error in Multiquadric Interpolation](image1)

![Figure 5: Error in Periodic Multiquadric Interpolation](image2)

We can see that both interpolations seem to converge towards a bound for error with the periodic Multiquadric’s error being lower. We can accurately locate this convergence by observing the \( L_\infty \) errors in Table 1.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Non-Periodic MQ</th>
<th>Periodic MQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.80 (-3)</td>
<td>8.43 (-4)</td>
</tr>
<tr>
<td>32</td>
<td>3.31 (-5)</td>
<td>1.72 (-7)</td>
</tr>
<tr>
<td>48</td>
<td>1.02 (-6)</td>
<td>3.99 (-11)</td>
</tr>
<tr>
<td>64</td>
<td>5.40 (-8)</td>
<td>1.89 (-15)</td>
</tr>
<tr>
<td>80</td>
<td>1.95 (-8)</td>
<td>2.11 (-15)</td>
</tr>
<tr>
<td>96</td>
<td>5.50 (-9)</td>
<td>2.86 (-15)</td>
</tr>
</tbody>
</table>

**Table 1: \( L_\infty \) Errors Multiquadric**

In Table 1 it is easy to see that the errors of the non-periodic Multiquadric Interpolation decay with \( N \) but are bounded near values of the order \( 10^{-11} \). The periodic multiquadric interpolation errors decay rapidly and are bounded near values of the order \( 10^{-15} \). In both cases the error ceases to decay and instead becomes worse at a point. This is due to the ill conditioning of the Interpolation Matrix, as a result of the shape parameter selection.
4 Unequally Spaced Points

4.1 Test Function

To address the problem of functions with very sharp areas of steepness, we formulate a technique for respacing the interval. At first we chose our interval based on each individual function with the aim to eventually come up with a technique of determining the point spacing on an interval adaptively. We start with the test function:

\[ f = \cos(\theta)e^{-4\sin(\theta)} \]  

(10)

By locating the zeroes of the derivative of the function we find that the function is at its sharpest between 4 and 5.5 on the interval 0 to 2\(\pi\). The idea is to increase points within this subinterval to decrease error. From 0 to 4 we placed half of the points, from 4 to 5.5 we placed four tenths of the points, and from 5.5 to 2\(\pi\) we placed the remaining tenth of the points.

4.2 Error

Again we notice, as we increase \(N\), the errors of the RBF approximations decay in both cases of interpolating the test function, but have a lower bound they converge to. In Figure 7, the pointwise errors (in logscale) of the Periodic Multiquadric RBF, with \(\epsilon = .85\) over the interval \([0, 2\pi]\), are shown for \(N = 16\) (blue), 32 (red), 48 (green), 64 (magenta), 80 (cyan), and 96 (black). On the right, in Figure 7, we have the pointwise errors (in logscale) of the Unequally-spaced Periodic Multiquadric RBF with the same conditions.
The difference in error isn’t as easily noticeable in these graphs. To get a better idea of the convergence we look at the $L_{\infty}$ error in Table 2.

<table>
<thead>
<tr>
<th>N</th>
<th>Non-Periodic MQ</th>
<th>Periodic MQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.18 (1)</td>
<td>3.55 (-2)</td>
</tr>
<tr>
<td>32</td>
<td>3.18 (-4)</td>
<td>1.52 (-5)</td>
</tr>
<tr>
<td>48</td>
<td>1.34 (-6)</td>
<td>9.00 (-9)</td>
</tr>
<tr>
<td>64</td>
<td>5.89 (-12)</td>
<td>3.14 (-12)</td>
</tr>
<tr>
<td>80</td>
<td>1.11 (-11)</td>
<td>3.39 (-11)</td>
</tr>
<tr>
<td>96</td>
<td>1.94 (-11)</td>
<td>4.77 (-12)</td>
</tr>
</tbody>
</table>

Table 2: $L_{\infty}$ Errors over Intervals

In Table 2 we see that the errors of both Interpolations decay with $N$ but are bounded near values of the order $10^{-12}$. Again, we notice in both cases the error ceases to decay and becomes worse at a point. Once again, this is due to the ill conditioning of the Interpolation Matrix, as a result of the shape parameter selection. The function being approximated would be better approximated with a changing shape parameter based upon the derivative of the function at points.

We notice that altering the point spacing does help the approximation if not by much in this example. By better defining the interval, better distributing points, and better selecting the shape parameter the error could be further reduced.

5 Conclusions

Working on this project I did not get to develop a true adaptive method for my Periodic RBF. The idea was to apply the adaptive method Dr. Heryudono had worked on to the periodic RBF. The adaptive method involved starting with a set number of equally spaced points and checking the error at points between them. If the error at these points was above some set threshold, points would be added there. This would go through iterations until the error fell below the threshold at which point it would end. In this method, the shape parameter varied based on the distance between each set of points. Creating an adaptive method would be the logical next step in the project.

Once a complete Adaptive Periodic Interpolation method was created, we were going to use it to solve PDEs and Interpolation problems.
We had the idea to try mapping Chebyshev points onto a periodic basis but again that was a path that I did not end up having the time to go down.

For future work with RBFs I would like to better understand how shape parameter selection can be optimised to avoid ill conditioned matrices. When I have the time, I hope to revisit this project.

6 Appendix

6.1 Matlab Code

clear all

N=100;
%theta=linspace(0,2*pi,N);
then1=linspace(0,4,5*N/10);
then1(end)=[];
then2=linspace(4,5.5,4*N/10);
then2(end)=[];
then3=linspace(5.5,2*pi,N/10);
then3(end)=[];
then=[then1,then2,then3];

thenac=then;
%f=sin(theta);
f=cos(theta).*exp(-4*sin(theta));
f=f(:);
ep=.85;

for i=1:length(then)
    for j=1:length(thenac)
        phi(i,j)=sqrt(1+(ep)^2*(2-2*cos(then(i)-thenac(j))));
%fphi(i,j)=exp(-1*(ep)^2*(2-2*cos(then(i)-thenac(j))));
%fphi(i,j)=1/(sqrt(1+(ep)^2*(2-2*cos(then(i)-thenac(j)))));
%fphi(i,j)=1/(1+(ep)^2*(2-2*cos(then(i)-thenac(j))));
    end
end
lambda=phi;
lambda=lambda(:);

%thenan=linspace(0,2*pi,1.5*N);
thenan1=linspace(0,4,7.5*N/10);
thenan1(end)=[];
thenan2=linspace(4,5.5,6*N/10);
thenan2(end)=[];
thenan3=linspace(5.5,2*pi,1.5*N/10);
thenan3(end)=[];
thenan=[thenan1,thenan2,thenan3];

for i=1:length(thenan)
    for j=1:length(thenac)
\begin{verbatim}
phi(i,j)=sqrt(1+(ep)^2*(2-2*cos(thetan(i)-thetac(j))));
%phi(i,j)=exp(-1*(ep)^2*(2-2*cos(thetan(i)-thetac(j))));
%phi(i,j)=1/(sqrt(1+(ep)^2*(2-2*cos(thetan(i)-thetac(j)))));
%phi(i,j)=1/(1+(ep)^2*(2-2*cos(thetan(i)-thetac(j))));
end
end

fn=phi*lambda;
%ff=sin(thetan);
ff=cos(thetan).*exp(-4*sin(thetan));
ff=ff(:);

figure(1)
plot(theta,f,'b','LineWidth',2),hold on
plot(thetan,fn,'r','LineWidth',2)
%plot(thetan,N,'o')

figure(2)
error = abs(ff-fn);
plot(thetan,log10(error),'b','LineWidth',2),hold on

6.2 References


\end{verbatim}