An Examination of Evaluation Algorithms for the RBF Method

Scott A. Sarra
Marshall University
and
Samuel Cogar
Marshall University

October 27, 2016

Abstract

Radial Basis Function (RBF) methods are important tools for scattered data interpolation and for the solution of PDEs in complexly shaped domains. Several approaches for the evaluation of RBF methods are known. To date, the most noteworthy methods are solving a linear system in the standard RBF basis using both double and extended precision floating point arithmetic and two approaches that make a change of basis for the purpose of obtaining a better conditioned linear system. In this work the approaches are compared and contrasted for the purpose of illustrating the strengths and weakness of each method as well as to give insight into the application of each approach.

1 Introduction

Radial Basis Function (RBF) methods are important tools for scattered data interpolation and for the solution of PDEs in complexly shaped domains. The most straight forward approach to evaluate the method uses the "standard basis functions" and involves solving a linear system which is typically poorly conditioned. Two variations of a method, dubbed the RBF-QR approach, use a different basis that spans the same space as the standard basis but results in a better conditioned linear system.

Extended precision floating point arithmetic can be used to accurately evaluate the ill-conditioned problem in the standard basis. This approach has been used and implemented in several different software environments that include: Mathematica [11], the Matlab Symbolic Toolbox [25], C++ [21], and Fortran [7]. The extended precision approach is attractive because it retains one of the great strengths of the RBF method which is simplicity. Whereas the RBF-QR methods, as can be ascertained by browsing the software that implements the methods, are far more complex.

Software packages that implement each of the methods are freely available and include: standard basis with double and extended precision ([23] and [20]), Mercer alternative basis method [14], and a RBF-QR alternative basis method [13].

2 The RBF Method

RBF interpolation uses a set of N distinct points $X = \{x_1^c, \ldots, x_N^c\}$ in $\mathcal{R}^d$ called centers. No restrictions are placed on the shape of problem domains or on the location of the centers. A RBF

$$\phi_k(x) = \phi(\|x - x_k^c\|_2, \varepsilon), \qquad x, x_k^c \in \mathcal{R}^d \tag{1}$$

is an infinitely differentiable (compactly supported and global RBFs without a shape parameter and with less smoothness exist but are not considered here) function of one variable $r = \|x - x_k^c\|_2$ that is centered at x_k^c and that contains a free parameter ε called the shape parameter. The RBF interpolant assumes the form

$$\mathcal{I}_N f(x) = \sum_{k=1}^{N} a_k \phi_k \left(\|x - x_k^c\|_2, \varepsilon_k\right) \tag{2}$$

where a is a vector of expansion coefficients. The Gaussian (GA) RBF

$$\phi(r) = e^{-\varepsilon^2 r^2} \tag{3}$$

is used throughout and is a representative member of the class of global, infinitely differently RBFs containing a shape parameter that interpolate with exponential accuracy. The reason for the restriction to the GA RBF is that both alternative basis algorithms work with this RBF. The expansion coefficients are determined by enforcing the interpolation conditions

$$\mathcal{I}_N f(x_k^c) = f(x_k^c), \qquad k = 1, 2, \ldots, N \tag{4}$$

which result in a $N \times N$ linear system

$$Ba = f. \tag{5}$$

The matrix B with entries

$$b_{jk} = \phi(\left\|x_j^c - x_k^c\right\|_2, \varepsilon_k), \qquad j, k = 1, \ldots, N \tag{6}$$

is called the system matrix. The evaluation of the interpolant (2) at M points x_j is accomplished by multiplying the expansion coefficients by the $M \times N$ evaluation matrix H that has entries entries

$$h_{jk} = \phi_k(\|x_j - x_k^c\|_2, \varepsilon_k), \qquad j = 1, \ldots, M \text{ and } k = 1, \ldots, N. \tag{7}$$

Derivatives are approximated by differentiating the RBF interpolant as

$$\mathcal{D}\left(\mathcal{I}_N f(x)\right) = \sum_{k=1}^{N} a_k \mathcal{D}\phi_k\left(\|x - x_k^c\|_2, \varepsilon_k\right) \tag{8}$$

where $\mathcal{D}$ is a linear differential operator. The operator $\mathcal{D}$ may be a single differential operator or a linear differential operator such as the Laplacian. Evaluating (8) at the centers X can be accomplished by multiplying the expansion coefficients by the evaluation matrix $H_{\mathcal{D}}$ with entries

$$h_{jk} = \mathcal{D}\phi(\left\|x_j^c - x_k^c\right\|_2, \varepsilon_k), \qquad j,k = 1,\ldots,N. \tag{9}$$

That is, $\mathcal{D}f \approx H_{\mathcal{D}}a$. Alternatively, derivatives can be approximated by multiplying the grid function values $\{f(x_k^c)\}_{k=1}^N$ by the differentiation matrix $D = H_{\mathcal{D}}B^{-1}$ since

$$\mathcal{D}f \approx H_{\mathcal{D}}a = H_{\mathcal{D}}(B^{-1}f) = (H_{\mathcal{D}}B^{-1})f. \tag{10}$$

The shape parameter ε_k may take on different values at each center x_k^c. Such an approach is called a variable shape parameter strategy. Several variable shape strategies [12, 26] have been suggested and present some advantages. A drawback of variable shape parameter strategies is that they cause the RBF system matrix to be non-symmetric. A constant shape has been used throughout since the alternative basis algorithms are not applicable with variable shape parameters.

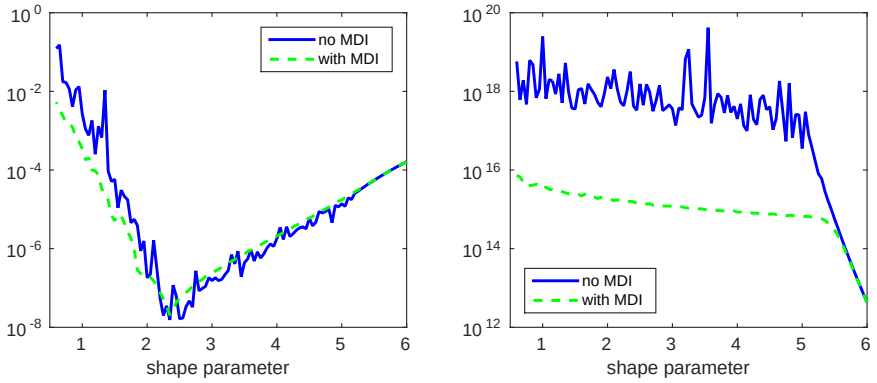

Figure 1: 1d interpolation, method ***D***. Left: error versus the shape parameter with and without MDI regularization. Right: system matrix condition number versus the shape parameter with and without regularization.

Both equations (5) for the expansion coefficients and (10) for the differentiation matrix assume that the system matrix is invertible. The GA system matrix is symmetric positive definite (SPD) and therefore invertible. While it is

invertible, the system matrix is typically very poorly conditioned in the standard basis. The eigenvalues of B satisfy $0 < \lambda_{min} = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N = \lambda_{max}$ and the matrix condition number in the 2-norm is $\kappa(B) = \lambda_{max}/\lambda_{min}$.

For a fixed set of centers and with the standard basis functions, the shape parameter affects both the accuracy of the method and the conditioning of the system matrix. The RBF method is most accurate for smaller values of the shape parameter where the system matrix is ill-conditioned. The attainable error and the condition number of the system matrix cannot both be kept small. This relationship has been dubbed the uncertainty principle [27]. Approaches which attempt to find a better conditioned basis that spans the same space will be examined in sections which follow.

In order to mitigate the inherent ill-conditioning of the standard basis, a regularization technique was shown to be effective in [22]. Instead of solving the system (5) the regularized system

$$(B + \mu I)y = f \tag{11}$$

is solved. The regularization parameter μ is a small positive constant and I is the identity matrix. The technique is called the method of diagonal increments (MDI) and its first use dates back to the 1940's [18]. The matrix $B + \mu I$ is better conditioned than B as

$$\kappa(B + \mu I) = \frac{\lambda_{max} + \mu}{\lambda_{min} + \mu} < \kappa(B) = \frac{\lambda_{max}}{\lambda_{min}}.$$

For small μ, $(B+\mu I)^{-1}$ is close to B^{-1} and MDI simply replaces B with $(B+\mu I)$ in computing the solution of a system. Equation

$$B^{-1} - (B + \mu I)^{-1} = \mu^2 B^{-1}(I + B^{-1}/\mu)B \tag{12}$$

quantifies how close that $(B + \mu I)^{-1}$ and B^{-1} are [9, 10]. For very small μ the difference is negligible.

Recent monographs [3, 5, 25, 29] on RBF methods can be consulted for more information.

3 Evaluation Methods

The names of the evaluation methods can be quite lengthy. For simplicity and brevity the following abbreviations are used: method $\boldsymbol{D}$ - the standard RBF method as described in section 2 implemented in IEEE 64-bit double precision with regularization by the method of diagonal increments [22]; method $\boldsymbol{X}$ - the standard RBF method as described in section 2 implemented in IEEE 128-bit quadruple precision with regularization by the method of diagonal increments; method $\boldsymbol{X}$(p) - the same as method $\boldsymbol{X}$ except that the floating point systems uses numbers with $p > 34$ decimal places of precision; method $\boldsymbol{Q}$ - the alternative basis method dubbed the RBF-QR method and described in reference [7]; $\boldsymbol{M}$ -

the alternative basis method described in reference [4] that is based on a Mercer expansion of the basis functions.

In order to demonstrate features of the methods, the following 1d interpolation problem is used as the methods are described in the next four subsections. The function $f(x) = e^{\sin(\pi x)}$ on the domain $[-1, 1]$ is interpolated using $N = 44$ centers and $M = 175$ evenly spaced evaluation points. Two sets of centers are used, uniformly spaced and the Chebyshev-Gauss-Lobatto (CGL) points $x_k = \cos(k\pi/(N-1)), \quad k = 0, 1, \ldots, N-1$ that cluster around the boundary points.

3.1 Method D

Method ***D*** is the standard, most basic approach, for evaluating the RBF method. The linear system (5) is solved using algorithms implemented in double precision floating point arithmetic as specified by IEEE 754-2008 standard [16]. Double precision is efficiently implemented in hardware in nearly every computer manufactured worldwide. The approach can be efficiently regularized via MDI as described in section 2.

In the right image of figure 1 the condition number of the system matrix is shown to increase with decreasing shape parameter. Once the exact condition number is larger than $\mathcal{O}\left(10^{16}\right)$ it can not be accurately calculated in double precision and the calculated condition numbers oscillate in the $\mathcal{O}\left(10^{18}\right)$ range. Coinciding with the inability to accurately calculate condition numbers the error curve, while still decreasing, begins to oscillate. In this example both begin to occur as the shape parameter decreases to approximately $\varepsilon = 5.25$. MDI regularization keeps the condition number of the system being solved at or below $\mathcal{O}\left(10^{16}\right)$ and prevents the error curve from oscillating as the shape parameter decreases. In this example the regularized solution is about one decimal place more accurate than the un-regularized solution in the small shape parameter range. In many cases the regularized solution is accurate to several more decimal places [22]. The 16 in the exponent of the condition number bound for MDI coincides with approximately 16 decimal places in which a number can be represented in double precision. A regularization parameter of $\mu = 5\varepsilon_M$ is used where machine epsilon in double precision is $\varepsilon_M \approx$ 2.2e-16.

3.2 Method X

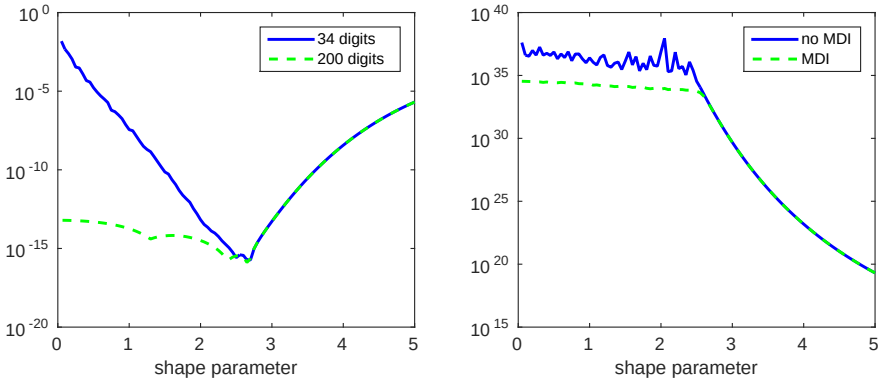

Figure 2: 1d interpolation, method $\boldsymbol{X}$ with CGL centers. Left: method $\boldsymbol{X}$ and $\boldsymbol{X}(200)$ error versus the shape parameter with MDI regularization. Right: system matrix condition number using 34 digits (quadruple precision) versus the shape parameter with and without regularization.

Method $\boldsymbol{X}$ refers to the standard basis RBF method implemented in IEEE 754-2008 standard [16] 128 bit quadruple floating point arithmetic. In quadruple precision, a number can be represented accurately to approximately 34 decimal places. Method $\boldsymbol{X}$(p) represents the method implemented in a floating point number system with p decimal digits of precision. Unlike the double type, which is implemented efficiently in specialized hardware registers, there are currently no general purpose processors that implement quadruple precision so it must be implemented in software and suffer a performance penalty. More accurate, arbitrary precision floating point number systems with p decimal digits of precision can be implemented in software as well, but generally not as efficiently as the quadruple type. The Matlab Radial Basis Function Toolbox (MRBFT) [23] that implements methods $\boldsymbol{X}$ and $\boldsymbol{X}$(p) uses the Multiprecision Computing Toolbox (MCT) for MATLAB [2] to provide extended precision. In benchmarks, the MCT has been shown to be much more efficient in implementing extended precision than other available options. Changing the precision of the floating point number system is simply done via the one line

```
mp.Digits(p);
```

where p is the desired precision and where $p = 34$ corresponds to the quadruple type.

The left image of figure 2 shows the accuracy versus the shape parameter of the MDI regularized $\boldsymbol{X}$ and $\boldsymbol{X}(200)$ methods. The MDI regularization parameter in each method is taken as $\mu = 10\varepsilon_M$ where $\varepsilon_M \approx$ 1.9e-34 for method $\boldsymbol{X}$ and $\varepsilon_M \approx$1.3e-200 for method $\boldsymbol{X}(200)$. Increasing the precision beyond 200 does not

increase the accuracy. The right image of figure 2 shows the condition number of the method $\boldsymbol{X}$ system matrix with and without regularization. The goal of regularization in method $\boldsymbol{X}$ is for the condition number to remain $\mathcal{O}\left(10^{34}\right)$ and for method $\boldsymbol{X}(200)$ to remain $\mathcal{O}\left(10^{200}\right)$. Method $\boldsymbol{X}$ takes approximately 37 times longer to execute than method $\boldsymbol{D}$ and method $\boldsymbol{X}(200)$ takes about 11 times as long as method $\boldsymbol{X}$.

3.3 Alternative basis

The search for an alternative basis is motivated by the fact that the standard basis is not a good choice for small shape parameters. For small values of the shape parameter, it is an ill-conditioned basis for a good approximation space. Finding a better basis that spans the same space depends on being able to expand the standard basis functions as

$$\phi_k(x) = \sum_{\ell=1}^{\infty} c_\ell(x_k)\, d_\ell\, e_\ell(x). \tag{13}$$

The action of a linear operator $\mathcal{L}$ on the basis functions can then be calculated as

$$\mathcal{L}\phi_k(x) = \sum_{\ell=1}^{\infty} c_\ell(x_k)\, d_\ell\, \mathcal{L}e_\ell(x).$$

The infinite expansion is truncated at a finite M which allows the system matrix in the standard basis to be approximately factorized as

$$B \approx CDE \tag{14}$$

where $M \geq N$ and $P = M - N$. The $N \times M$ coefficient matrix C has elements $c_{k\ell} = c_\ell(x_k)$, the $M \times M$ diagonal scaling matrix D has diagonal elements d_ℓ, and the $M \times N$ matrix E contains the expansion functions (in method $\boldsymbol{M}$, $E = C^T$) and has elements $e_{\ell k} = e_\ell(x_k)$.

The value of M is determined by setting it equal to the smallest value of $\ell \geq N$ such that $d_\ell < \epsilon_M$ where ϵ_M is machine epsilon ($\epsilon_M = 2^{-52}$ in IEEE double precision). The elements of d_ℓ will decay rapidly with ℓ with small values of the shape parameter but will decay slower for large values.

The purpose of the factorization is to remove the ill-conditioning from the matrix B and isolate it in the diagonal matrix D which can then be safely inverted by simply inverting the diagonal elements. The coefficient matrix C is factorized via a QR factorization and written in block form as

$$C = QR = Q\,[R_1\ R_2]$$

where Q is a $N \times N$ orthogonal matrix, R is a $N \times M$ upper triangular matrix, R_1 is $N \times N$ and contains the first N rows of R, and R_2 is $N \times P$ and contains the last P rows of R. The matrix R is divided into blocks in order to separate the contributions of the coefficients from the first N basis functions from those of the last P.

The new basis functions are then built with first a contribution from the first N expansion functions

$$E_1 = D_1^{-1} R_1^{-1} Q^T B.$$

where now E has been divided into a $N \times N$ matrix E_1 and a $N \times P$ matrix E_2 and D has been split into a $N \times N$ diagonal matrix D_1 and a $P \times P$ diagonal matrix D_2. The second contribution to the new basis functions comes from the last P basis functions with expansion coefficients that are stored in a $N \times P$ correction matrix

$$X = D_1^{-1} R_1^{-1} R_2 D_2.$$

In the new basis, a system matrix, interpolation evaluation matrix, or derivative evaluation matrix is formed as

$$\Psi = E_1 + E_2 X^T$$

depending on if the expansion functions are used to fill E or if it is filled with the expansion functions that have been acted upon by $\mathcal{L}$. Unlike the GA system matrix in the standard basis which is SPD and can be factorized with a Cholesky factorization, the system matrix in the new basis is no longer positive definite and it is not even symmetric. A LU factorization with twice as many flops must be used.

The basis function expansion (13) is different in each space dimension. The next sections discuss the alternative basis methods in 1d. The details of the alternative basis algorithms in higher dimension are much more involved. The original manuscripts can be consulted for details of the expansions in higher dimensions. Section 4 gives multiple 2d examples.

3.3.1 Method Q

The elements of the diagonal scaling matrix

$$d_\ell = \frac{2\varepsilon^{2\ell}}{\ell!}$$

of method ***Q*** are determined by the sizes of the eigenvalues of the system matrix (6) in the standard basis. The expansion functions

$$e_\ell(x) = e^{-\varepsilon^2 x^2} \cos(\ell \arccos(x))$$

are a product of Gaussian functions and Chebyshev polynomials. The expansion coefficients are

$$c_\ell(x_k) = t_\ell \, e^{-\varepsilon^2 x_k^2} \, x_k^\ell \;\; {}_0F_1(\ell + 1, \varepsilon^4 x_k^2).$$

where $t_0 = 0.5$ and $t_\ell = 1$ for $\ell \geq 1$ and ${}_0F_1$ is a hypergeometric function.

The left image of figure 3 displays the accuracy of method ***Q*** for 1d interpolation over a range of shape parameter using both clustered and uniform centers. With clustered centers, the method can interpolate the function to near machine

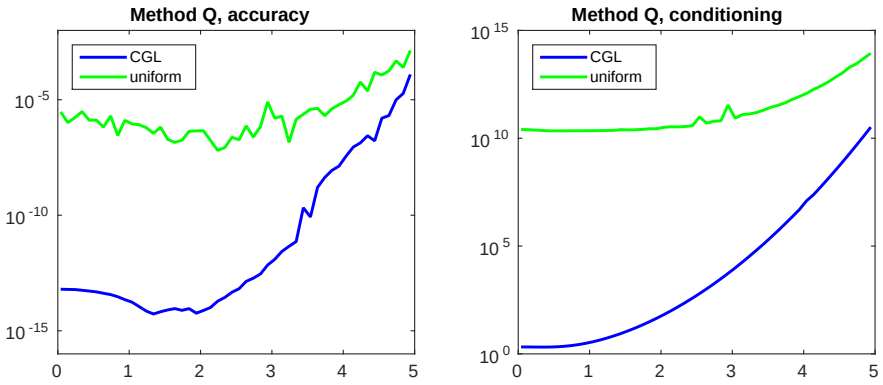

Figure 3: 1d interpolation, method $\boldsymbol{Q}$. Left: error versus the shape parameter. Right: system matrix condition number versus the shape parameter.

precision for $\varepsilon \leq 2$. In the right image of figure 3 the condition number of the system matrix in the new basis is very small with clustered centers for $\varepsilon \leq 2$. A center distribution with a "good" clustering of centers near the boundary is important for the success of method $\boldsymbol{Q}$. In this example, the truncation parameter M ranges from 83 at the largest shape to 47 at the smallest shape for both the CGL and uniform centers. In order to be more robust, it is necessary for the QR factorization to incorporate pivoting [7]. The standard QR factorization in most linear algebra packages such as Matlab does not use pivoting, however the authors of the software have supplied a QR factorization with pivoting with their software [13].

3.3.2 Method M

Method $\boldsymbol{M}$ is based on a Mercer expansion [15] of a positive definite kernel function, the Gaussian RBF (3). Mercer's theorem allows the GA to be expanded in an eigenvalue/eigenfunction expansion that fits into the framework of (13). For the GA, both the eigenvalues and eigenfunctions are known in closed form [19]. Let

$$\beta = \left(1 + \left(\frac{2\varepsilon}{\alpha}\right)^2\right)^{\frac{1}{4}}$$

and

$$\delta^2 = \frac{\alpha^2}{2}\left(\beta^2 - 1\right).$$

Then the eigenvalues

$$d_\ell = \sqrt{\frac{\alpha^2}{\alpha^2 + \delta^2 + \varepsilon^2}}\left(\frac{\varepsilon^2}{\alpha^2 + \delta^2 + \varepsilon^2}\right)^{\ell-1}$$

are the diagonal elements of the scaling matrix and $\{d_\ell\}_{\ell=1}^{\infty}$ is a non-increasing sequence of positive numbers such that $d_\ell \to 0$ as $\ell \to \infty$. The eigenfunctions are

$$e_\ell(x) = e^{-\delta^2 x^2} H_{\ell-1}(\alpha\beta x)\sqrt{\frac{\beta}{2^{\ell-1}\Gamma(\ell)}}$$

where $H_{\ell-1}$ are degree $\ell - 1$ Hermite polynomials. The expansion coefficients are

$$c_\ell(x_k) = e_\ell(x_k).$$

Method $\boldsymbol{M}$ is applicable for basis functions, such as the GA and Matern RBF, for which an eigen expansion is known in closed form. If the expansion is not known in closed form, it is possible to approximate the eigenvalues and eigenfunctions numerically [19]. However, while the larger eigenvalues can be approximated very accurately, it is difficult if not impossible to approximate the smallest eigenvalues using double precision. It is our experience that method $\boldsymbol{M}$ can not be accurately implemented in double precision when the eigenvalues and eigenfunctions are calculated numerically. Of course extended precision could be used, but it would be simpler to work with method $\boldsymbol{X}$ and the standard RBF basis in this case.

A formula does not exist for an optimal value of the parameter α. In selecting a good value of α there are two conflicting goals. Small α leads to eigenfunctions of relatively the same scale and large α maintains numerical orthogonality on a finite domain. The software [14] has a function to automatically specify the parameter α. It returns the smallest value of α for which the eigenfunctions remain orthonormal within a tolerance. In general, the best value of alpha is different for each value of the shape parameter.

In the top left image of figure 4 is the value of alpha that the software package selects, which is based on orthogonality, plotted against the shape parameter. The smallest value of alpha is 0.24 at $\varepsilon = 6$ and the largest value is 1.58 at $\varepsilon = 0.04$. In the right image is the corresponding accuracy versus the shape parameter. In this example, as well as all the examples that follow, the value of alpha selected based on orthogonality did not even come close to producing the most accurate results. In this example using $\alpha = 6$ at every value of the shape parameter produced good results (bottom left image of figure 4). We found no good way to find this value other than trial and error. As expected, method $\boldsymbol{M}$ is both more accurate and better conditioned when the centers are clustered near the boundary. However, later examples will show that method $\boldsymbol{M}$ is less sensitive to how the clustering is done than is method $\boldsymbol{Q}$. The truncation parameter M ranges from 113 for the largest shape to 49 for the smallest shape in the run in the bottom left figure.

In higher dimensions a tensor product formulation of the 1d expansion is used. This approach can be computationally expensive even for small ε and moderate N and M. A low-rank approximation of the full RBF interpolant using $M < N$ eigenfunctions is explored in [4]. In this work, due to limited space, only the full RBF interpolant is considered. More detailed information on method $\boldsymbol{M}$ can be found in the recent monograph [6].

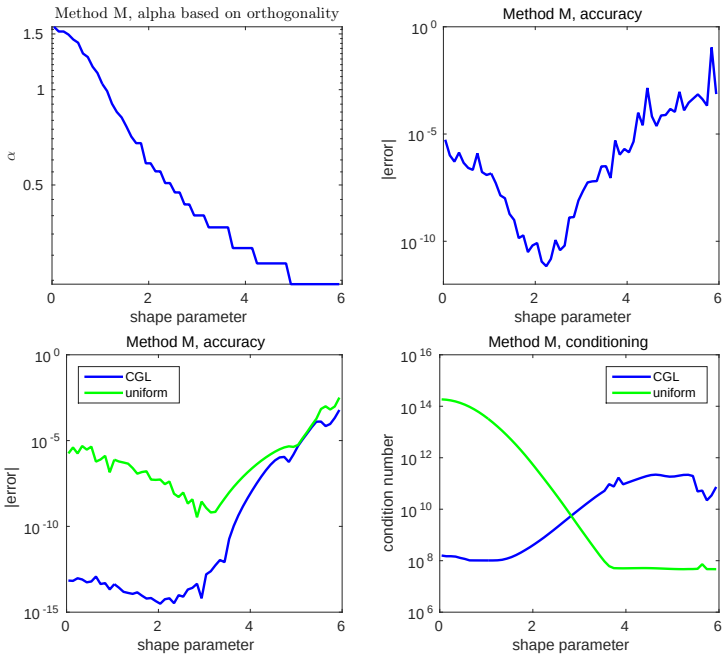

Figure 4: Top: 1d interpolation with CGL centers, method $\boldsymbol{M}$ with the parameter α selected based on othogonality of eigenfunctions. Left: value of α versus the shape. Right: error versus the shape parameter. Bottom: 1d interpolation with CGL centers and $\alpha = 6$. Left: error versus the shape parameter. Right: system matrix condition number versus the shape parameter.

3.3.3 Summary 1d Example

Figure 5 shows the accuracy versus the shape parameter for all four approaches. On a CGL center distribution, methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ produce errors close to machine epsilon with small values of the shape parameter. While method $\boldsymbol{Q}$ performs well for small N, it returns NaN (not a number) for $N \geq 172$ CGL centers. The best accuracy is obtained by method $\boldsymbol{X}$ at a shape parameter that is larger than those in the shape region where methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ are most effective. In the small shape parameter region where methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ are most accurate their accuracy can be matched by method $\boldsymbol{X}$ if the decimal precision is increased from 34 to 128. Execution times (normalized so that the double precision time is one): $\boldsymbol{D}(1)$, $\boldsymbol{M}(195)$, $\boldsymbol{Q}(25)$, $\boldsymbol{X}(25)$, and $\boldsymbol{X(128)}(177)$. The following scripts implement the example: ***interp1d_d.m***, ***interp1d_x.m***, ***interp1d_q.m***, and ***interp1d_m.m***.

Figure 6 shows the accuracy versus the shape parameter for all four approaches in approximating a derivative. The interpolants on the CGL grid in the left image of figure 5 are differentiated. Execution times : $\boldsymbol{D}(1)$, $\boldsymbol{M}(712)$, $\boldsymbol{Q}(43)$, $\boldsymbol{X}(46)$, and $\boldsymbol{X(128)}(195)$. The following scripts implement the example:

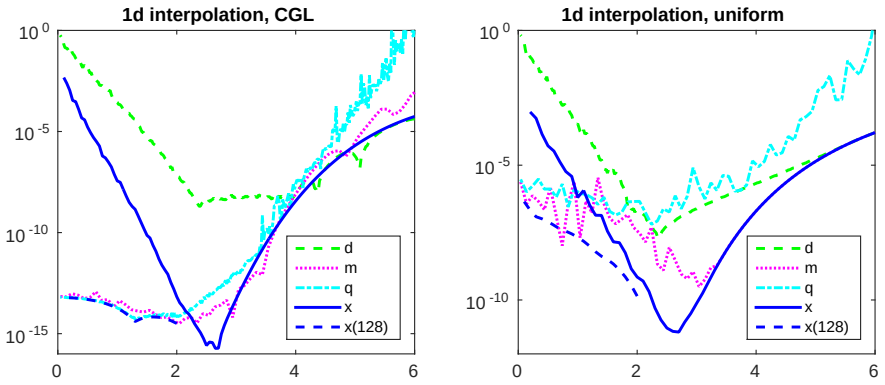

Figure 5: 1d interpolation, error versus the shape parameter. In method $\boldsymbol{M}$ the parameter $\alpha = 6$ is used. Left: CGL centers. Right: uniform centers.

diff1d_d.m, ***diff1d_x.m***, ***diff1d_q.m***, and ***diff1d_m.m***.

4 2d Examples

The smoothly varying function (15)

$$f(x,y) = e^{(x/2+y/5)}\cos(xy) \tag{15}$$

and the more rapidly varying Franke function

$$\begin{aligned} f(x,y) &= \frac{3}{4}e^{\left[\frac{-1}{4}(9x-2)^2-\frac{1}{4}(9y-2)^2\right]} + \frac{3}{4}e^{\left[\frac{-1}{49}(9x+1)^2-\frac{1}{10}(9y+1)^2\right]} \\ &= \frac{1}{2}e^{\left[\frac{-1}{4}(9x-7)^2-\frac{-1}{4}(9y-3)^2\right]} - \frac{1}{5}e^{\left[-(9x-4)^2-(9y-7)^2\right]} \end{aligned} \tag{16}$$

which is a classic test problem for RBF methods [8] are interpolated and differentiated on various sets of scattered centers.

RBF methods have complete freedom on center placement. Both experience and theory dictate that a good center distribution both well-cover a domain (no large holes or clumps) and also mildly cluster centers around boundaries. Randomly located centers tend to clump too much to be an effective choice. Quasi-random sequences [17] scatter centers without a clear pattern and do not clump centers. For this reason quasi-random sequences have become popular tools to locate RBF centers. Hammersley and Halton sequences are two examples [17]. On the unit circle the quasi-random centers x_k^c can be clustered near the boundary via the map $x_k = \sin(\pi x_k/2)$. Example center distributions are shown in figure 7.

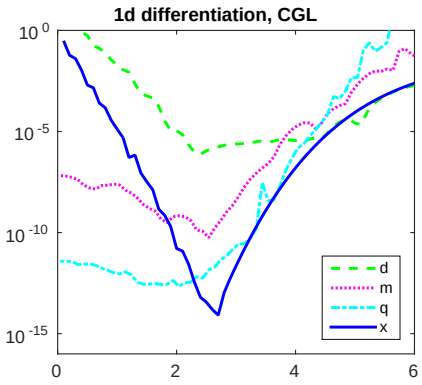

Figure 6: 1d differentiation with CGL center distribution, error versus the shape parameter. In method $\boldsymbol{M}$ the parameter $\alpha = 6$ is used.

4.1 Smoothly varying function

Figure 8 shows the results of interpolating function (15) on both clustered and uniform centers on the unit circle. The function is simple enough so that it can be well-resolved with a small number ($N = 787$) of centers. With the clustered centers methods $\boldsymbol{Q}$ and $\boldsymbol{M}$ have small $\mathcal{O}\left(10^{-12}\right)$ errors. In the same small shape region method $\boldsymbol{X}$ has an error of 9.6e-17 at $\varepsilon = 1.2$. With the uniform centers the method $\boldsymbol{M}$ errors in the small shape region are $\mathcal{O}\left(10^{-10}\right)$. However, method $\boldsymbol{Q}$ is unable to resolve the solution on the uniform centers and has errors larger than the double precision results of the standard basis algorithm method $\boldsymbol{D}$. Execution times: $\boldsymbol{D}(1)$, $\boldsymbol{M}(28)$, $\boldsymbol{Q}(129)$, and $\boldsymbol{X}(112)$. The following scripts implement this example as well as the rapidly varying function interpolation example in the next section: ***interp2d_d.m***, ***interp2d_x.m***, ***interp2d_q.m***, and ***interp2d_m.m***.

4.2 Rapidly varying function

The Franke function (16) is a more difficult test than function (15) as it takes a larger number of centers to accurately resolve the function. Figure 9 shows the accuracy over a range of shape parameter of the four methods with clustered centers as N increases in the sequence 787, 1182, 1572, and 3140. In this example with a larger number of centers, methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ are more accurate in the small shape region $\varepsilon < 1.5$, but they only achieve modest accuracy compared to previous examples with a smaller number of centers. With all four values of N, method $\boldsymbol{X}$ is more accurate at a shape parameter $\varepsilon > 1.5$ than are methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ at any value of the shape parameter. An oddity of the example is that for the three smaller values of N, method $\boldsymbol{D}$ is more accurate than the other methods and the accuracy is achieved at a larger shape parameter $\varepsilon > 6$. With $N = 3140$, the method $\boldsymbol{Q}$ software has a runtime error in the pivoted QR

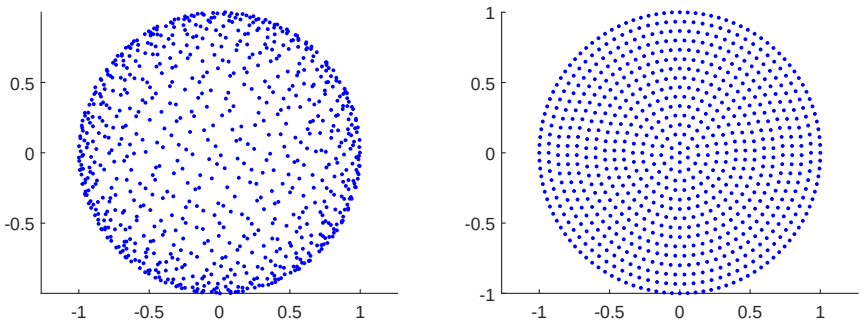

Figure 7: 2d interpolation center distributions. Left: 787 clustered Halton centers. Right: 787 uniformly spaced centers.

routine for shape parameters less than $\varepsilon = 0.41$. Execution times for $N = 3140$: $\boldsymbol{D}(1)$, $\boldsymbol{M}(29)$, $\boldsymbol{Q}(51)$, and $\boldsymbol{X}(301)$.

4.3 Gradient of a smooth function

The gradient $\mathcal{L} = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}$ of function (15) is approximated on the 787 centers that it was accurately interpolated on in the left image of figure 7. The accuracy of the four methods is shown in figure 10. Method $\boldsymbol{Q}$ accurately resolves the gradient throughout the small shape region with about 10 decimal places of accuracy. However, method $\boldsymbol{M}$ is unable to resolve the gradient in this problem. Method $\boldsymbol{M}$ is only able to get moderate accuracy, about 3 decimal places, for very small shape parameter ($\varepsilon < 0.1$). This was typical of all 2d derivative approximations with method $\boldsymbol{M}$. Our results are consistent with the examples that approximate derivatives on scattered centers in two dimensions that accompany the software package for method $\boldsymbol{M}$. For example, the script ***ex15b.m*** from [14] approximates the solution of a Helmholtz problem with a local RBF method with scattered centers. The scripts implements method $\boldsymbol{D}$ for which the error is 1.6e-5 and method $\boldsymbol{M}$ for which the error is 7.2e-2. It is unclear as to whether this is a problem inherent to the method or if it is a problem with its implementation in software. An examination of this issue will be done when a 2d version of method$\boldsymbol{M}$ is implemented in the software package described in [23]. Method $\boldsymbol{X}$ has 13 digits of accuracy at $\varepsilon = 1.2$. Execution times: $\boldsymbol{D}(1)$, $\boldsymbol{M}(83)$, $\boldsymbol{Q}(32)$, and $\boldsymbol{X}(106)$. The following scripts implement the example: ***gradientCircle_d.m***, ***gradientCircle_x.m***, ***gradientCircle_q.m***, and ***gradientCircle_m.m***.

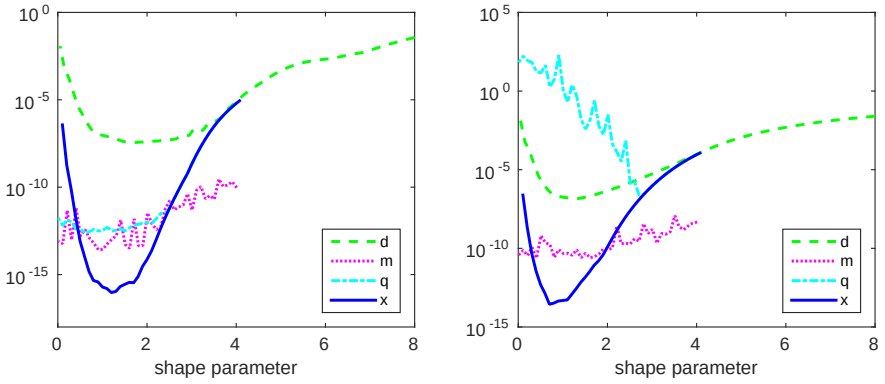

Figure 8: 2d interpolation of function (15). Left: 787 clustered Halton centers. Right: 787 uniformly spaced centers. Method $\boldsymbol{M}$ uses $\alpha = 1.5$.

4.4 Challenging problems

In this section two problems are discussed for which we were unable to successfully apply methods $\boldsymbol{Q}$ and $\boldsymbol{M}$. There were many other problems for which attempts to apply the two methods were unsuccessful as well. A time consuming challenge in writing this manuscript was in finding problems for which all methods could be applied. In the end, that is why circular domains were used for which it was previously documented [7] as how to cluster centers so that method $\boldsymbol{Q}$ could be successfully applied. For a non-circular domain, it is a non-trivial task to find a "good" clustering of centers that allow method $\boldsymbol{Q}$ be accurate. The domains and center distributions in figure 11 represent two cases in which method $\boldsymbol{Q}$ was unable to produce acceptable results. Both problems entail derivative calculation on scattered centers which is difficult for method $\boldsymbol{M}$.

The Poisson problem

$$-u_{xx} - u_{yy} = \pi^2 \sin(\pi x) \sin(\pi y) \tag{17}$$

with Dirchlet boundary conditions prescribed according to the exact solution

$$u(x, y) = 1 - x + xy + \frac{1}{2} \sin(\pi x) \sin(\pi y)$$

is discretized on the complexly shaped domain in the right image of figure 11 with 2509 centers. Method $\boldsymbol{D}$ produced its best accuracy at $\varepsilon = 3.6$ with an error of 8.6e-6 and method $\boldsymbol{X}$ was most accurate at $\varepsilon = 1.85$ with an error of 4.3e-14. Execution times: $\boldsymbol{D}(1)$ and $\boldsymbol{X}(268)$. The following scripts implement the example: ***poisson_d.m***, ***poisson_x.m***, ***poisson_q.m***, and ***poisson_m.m***.

The gradient of function 15 is approximated on the 2839 centers in the complexly shaped domain shown in the left image of figure 11. Method $\boldsymbol{D}$ produced

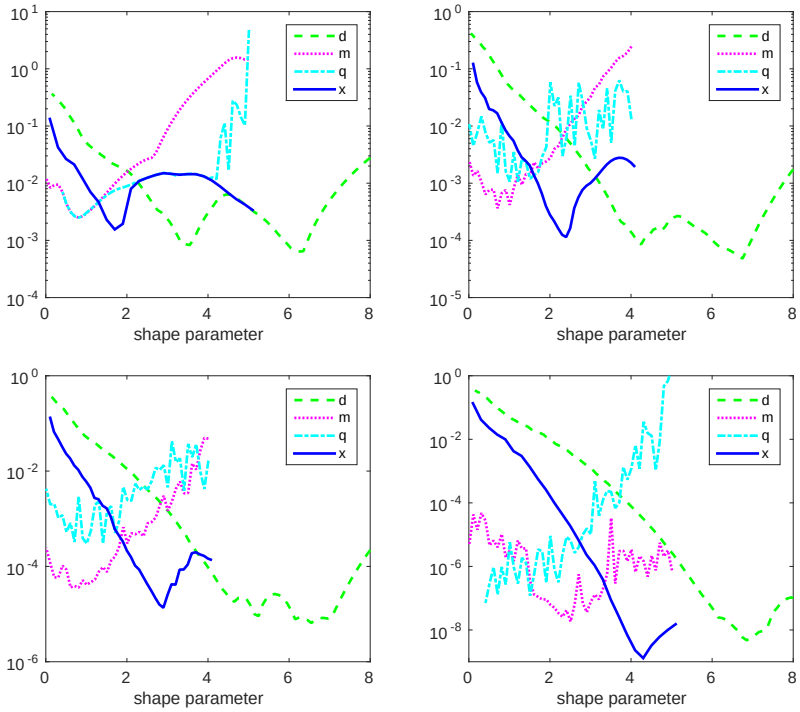

Figure 9: Franke function interpolation with clustered centers. Accuracy versus the shape parameter. The method $\boldsymbol{M}$ parameter is $\alpha = 6$. Top left: $N = 787$. Top right, $N = 1182$. Lower left, $N = 1572$. Lower right, $N = 3140$.

its best accuracy at $\varepsilon = 1.35$ with an error of 6.5e-5 and method $\boldsymbol{X}$ was most accurate at $\varepsilon = 1.5$ with an error of 8.6e-13. Execution times: $\boldsymbol{D}(1)$ and $\boldsymbol{X}(130)$. The following scripts implement the example: ***gradientComplex_d.m***, ***gradientComplex_x.m***, ***gradientComplex_q.m***, and ***gradientComplex_m.m***.

4.5 Local RBF method

The local RBF method for the numerical solution of PDEs sets up a stencil of $n-1$ neighboring centers at each of the N centers in the domain. N size $n \times n$ linear systems are solved to determine the weights on the stencils. In this section an example is constructed that uses the evaluation methods to determine the stencils weights and compares the methods on accuracy and execution times. An example stencil is shown in figure 12. The center distribution consists of $N =$ 5000 clustered Hammersley points on the unit circle. Stencil sizes of 8, 20, 50, and 100 are used and a fixed shape parameter of $\varepsilon = 1.75$ is used. The accuracy is check at the stencil that is centered at (-0.38,0.63). The reference stencil for which the accuracy of the other methods is checked against is calculated with

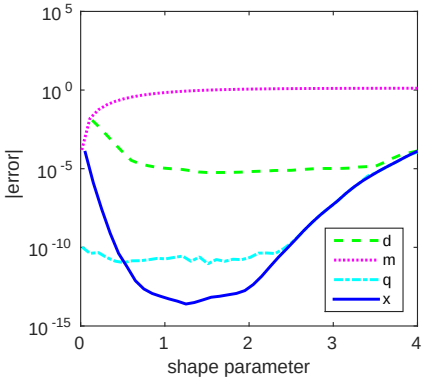

Figure 10: Gradient of function function (15). Differentiated interpolant from the left image of figure 8 using 787 clustered Halton centers. Method ***M*** parameter $\alpha = 1.5$.

method ***X(p)*** with $p = 500$. The condition number of the reference stencil and the accuracy of each approach is summarized in the following table:

n	$\kappa(B)$	D	X	Q
8	3.2e8	3.8e-4	6.8e-23	7.8e-11
20	3.8e13	-	2.4e-18	1.9e-10
50	3.7e22	-	3.9e-9	5.4e-9
100	2.3e30	-	2.0e-1	1.3e-7

The weights in the standard RBF basis (methods ***D***, ***X***, and ***X(p)***) are the solution of a linear system and their accuracy is roughly explained by condition number rule of thumb from elementary numerical analysis [28]. That is, the number of accurate decimal places in the solution of a linear system is approximately $p - k$ where p is the number of digits of accurate decimal places that the system is capable of and the condition number of the matrix is $\mathcal{O}\left(10^k\right)$. Thus, the solution of a linear system using ***X(100)*** should accurately calculate stencils weights to at least 16 decimal places for condition numbers up to $\mathcal{O}\left(10^{84}\right)$. The execution time in seconds of each method for calculating all 5000 stencils weights is summarized in the following table:

	8	20	50	100
D	0.24	0.36	0.51	0.98
X	6.7	9.2	25.9	101.6
X(100)	10.7	25.6	127.9	479.0
Q	41.9	58.1	102.9	168.2

The scripts ***localRbfWeights.m*** and ***localRbfWeightsTiming.m*** carry out the example.

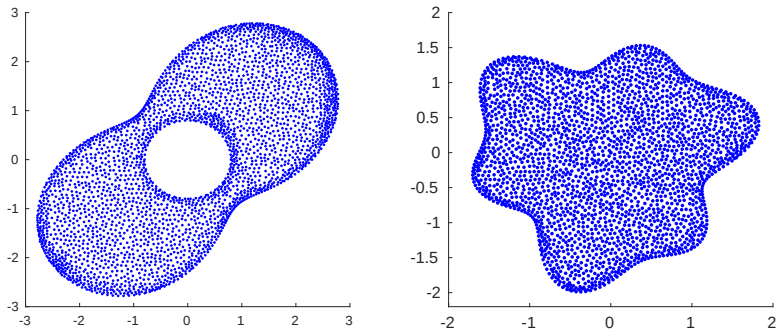

Figure 11: Bad centers for alternative basis algorithms: Left: region with holes, 3061 centers. Right: complexly shaped domain with 2509 centers

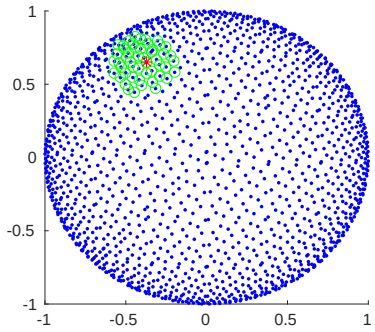

Figure 12: $N = 2000$ clustered Hammersly points on the unit circle. A local stencil centered at (-0.38,0.63) (red asterick) and 49 nieghboring points (green circles) on the stencil.

5 Conclusions

The following methods have been considered for the evaluation of the RBF approximation method: Method $\boldsymbol{D}$ uses double precision floating arithmetic and the standard basis functions for the RBF method. Method $\boldsymbol{X}$ uses extended precision floating arithmetic and the standard basis functions for the RBF method. Method $\boldsymbol{X}$, using 34 digits of decimal precision which corresponds to the IEEE quadruple type, is optimized for efficiency compared to arbitrary length extended types. Both methods $\boldsymbol{D}$ and $\boldsymbol{X}$ can be regularized by the method of diagonal increments in order to alleviate the ill-conditioning problem. Methods $\boldsymbol{Q}$ and $\boldsymbol{M}$ uses alternative basis functions that span the same space as the standard basis. The expansion of method $\boldsymbol{Q}$ is based on observations about the eigenvalues of the system matrix while method $\boldsymbol{M}$ is based on a Mercer

expansion of the Gaussian RBF. In the rest of the conclusion the strengths and weaknesses of each method are summarized.

Methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ in double precision are capable of exceptional accuracy with small shape parameters and a small to moderate number of centers. The success of method $\boldsymbol{Q}$ is strongly dependent on a distribution of centers that are "adequately" clustered around boundaries. Method $\boldsymbol{M}$ is less sensitive to the location of centers. With a larger number of centers ($N > 1000$), both methods may still produce several accurate decimal places with small shape parameters. Extended precision could be used to increase this number, but in this case it would simply be better to use method $\boldsymbol{X}$. On center distributions in which method $\boldsymbol{Q}$ accurately interpolated functions, it also accurately resolved derivatives. Method $\boldsymbol{M}$ was unable to accurately approximate derivatives in 2d, even with center locations in which it could accurately interpolate. It was limited to providing two or three decimal places of accuracy for $\epsilon < 0.1$. Method $\boldsymbol{Q}$ does not contain any free parameters. Method $\boldsymbol{M}$ has a parameter α that must be specified. The method $\boldsymbol{M}$ software will select a value of α based on eigenfunction orthogonality but in all numerical examples the value was far from optimal. In the examples, trial and error was used to find the value of the parameter that worked best. A good choice of α is vital to the success of method $\boldsymbol{M}$. Both method $\boldsymbol{M}$ and $\boldsymbol{Q}$ are complex, and their software implementations even more so, when compared to the standard RBF method (method $\boldsymbol{D}$) and they stray far from the hallmark of the RBF method, which is simplicity.

Method $\boldsymbol{X}$ is undeniably the simplest alternative to method $\boldsymbol{D}$. It uses reliable, well tested, linear algebra routines that accept a custom data type. Converting computer code from method $\boldsymbol{D}$ to method $\boldsymbol{X}$ entails only very minor changes. In the example problems, method $\boldsymbol{X}$ usually produced the overall best accuracy which was at a shape parameter that was located outside of the small shape parameter region in which methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ are most suited. In the small shape parameter region in which methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ are most suited, method $\boldsymbol{X}$ typically is unable to match their accuracy. However, if the number of accurate decimal digits is increased to above 34, method $\boldsymbol{X(p)}$ can match or exceed the accuracy of methods $\boldsymbol{M}$ and $\boldsymbol{Q}$ at the expense of additional computer time.

The obvious drawback of method $\boldsymbol{X}$ is execution speed since quadruple and arbitrary precision are implemented in software. Currently a quadruple flop takes approximately 100 times a double flop as well as the additional overhead of setting up and storing custom data types. For small 1d problems, method $\boldsymbol{X}$ typically takes about 80 to 100 times longer to execute than does method $\boldsymbol{D}$ while for larger higher dimensional problems method $\boldsymbol{X}$ may take 200 to 300 times longer to execute than does method $\boldsymbol{D}$. Until the quadruple type is implemented in specialized hardware registers an option for increasing speed is GPU acceleration. While not currently implemented in the MCT, other extended precision floating point systems have seen benefits from GPU acceleration [1]. All problem domains that have been used have a type of symmetry that allow RBF methods using the standard basis (methods $\boldsymbol{D}$ and $\boldsymbol{X}$) to be implemented more efficiently. The symmetry can be exploited and the leading term in the

flop count of the associated linear algebra algorithms can be reduced by a factor of four [24]. The savings can somewhat alleviate the extra computational expense of method $\boldsymbol{X}$. For example, in the derivative approximation in section 4.3 symmetry can be exploited to reduce the performance penalty from 106 to 46 (implemented in ***gradientCircle_xc.m***). For larger N, the execution time reduction approaches a factor of 3.5.

The source code that implements all examples in this manuscript is available from the first author's website.

References

[1] GPUREC: supporting high precision on graphics processors. https://code.google.com/p/gpuprec/. 5

[2] Advanpix. Multiprecision computing toolbox for Matlab, version 3.9.9 for 64-bit Linux. http://www.advanpix.com/. 3.2

[3] M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, 2003. 2

[4] G. Fasshauer and M. McCourt. Stable evaluation of Gaussian RBF interpolants. *SIAM Journal on Scientific Computing*, 34:737–762, 2012. 3, 3.3.2

[5] G. E. Fasshauer. *Meshfree Approximation Methods with Matlab*. World Scientific, 2007. 2

[6] G. E. Fasshauer and M. McCourt. *Kernel-based Approximation Methods using Matlab*. World Scientific, 2015. 3.3.2

[7] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with Gaussian Radial Basis Functions. *SIAM Journal on Scientific Computing*, 33:869–892, 2011. 1, 3, 3.3.1, 4.4

[8] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, pages 181–200, 1982. 4

[9] L. Guttman. Enlargement methods for computing the matrix inverse. *The Annals of Mathematical Statistics*, 17(3):336–343, 1946. 2

[10] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(2):53–60, 1981. 2

[11] C.-S. Huang, C.-F. Leeb, and A.H.-D. Cheng. Error estimate, optimal shape factor, and high precision computation of multiquadric collocation method. *Engineering Analysis with Boundary Elements*, 31:614–623, 2007. 1

[12] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates. *Computers and Mathematics with Applications*, 19(8/9):127–145, 1990. 2

[13] E. Larsson. A MATLAB implementation of the RBF-QR method. http://www.it.uu.se/research/scicomp/software/rbf_qr. 1, 3.3.1

[14] M. McCourt. GaussQR: Stable Gaussian computation. http://math.iit.edu/~mccomic/gaussqr/. 1, 3.3.2, 4.3

[15] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909. 3.3.2

[16] J. Muller. *Handbook of Floating Point Arithemetic*. Birkhauser, 2010. 3.1, 3.2

[17] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. CBMS-NSF, SIAM, Philadelphia, 1992. 4

[18] W. Piegorsch and G. Casella. The early use of matrix diagonal increments in statistical problems. *SIAM Review*, 31:428–434, 1989. 2

[19] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. 3.3.2

[20] S. A. Sarra. A Matlab radial basis function toolkit with symmetry exploitation, regularization, and extended precision. http://www.scottsarra.org/rbf/rbf.html. 1

[21] S. A. Sarra. Radial basis function approximation methods with extended precision floating point arithmetic. *Engineering Analysis with Boundary Elements*, 35:68–76, 2011. 1

[22] S. A. Sarra. Regularized symmetric positive definite matrix factorizations for linear systems arising from RBF interpolation and differentiation. *Engineering Analysis with Boundary Elements*, 44:76–86, 2014. 2, 3, 3.1

[23] S. A. Sarra. The Matlab radial basis function toolkit. *Journal of Open Research Software, under review*, 2016. 1, 3.2, 4.3

[24] S. A. Sarra. Radial basis function methods - the case of symmetric domains. *Submitted to Numerical Methods for Partial Differential Equations*, 2016. 5

[25] S. A. Sarra and E. J. Kansa. Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations. *Advances in Computational Mechanics*, 2, 2009. 1, 2

[26] S. A. Sarra and D. Sturgill. A random variable shape parameter strategy for radial basis function approximation methods. *Engineering Analysis with Boundary Elements*, 33:1239–1245, 2009. 2

[27] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3:251–264, 1995. 2

[28] G. Stewart. *Afternotes on Numerical Analysis*. SIAM, 1996. 4.5

[29] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005. 2