Applied Numerical Mathematics 130 (2018) 131–142

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

A rational radial basis function method for accurately resolving discontinuities and steep gradients

Scott A. Sarra *, Yikun Bai

Marshall University, United States of America

ARTICLE INFO

Article history:
Received 21 April 2017
Received in revised form 4 April 2018
Accepted 5 April 2018
Available online 11 April 2018

Keywords:
Radial Basis Functions
Rational interpolation
Time-dependent PDEs
Shocks
Gibbs phenomenon

ABSTRACT

Radial Basis Function (RBF) methods have become important tools for scattered data interpolation and for solving partial differential equations (PDEs) in complexly shaped domains. When the underlying function is sufficiently smooth, RBF methods can produce exceptional accuracy. However, like other high order numerical methods, if the underlying function has steep gradients or discontinuities the RBF method may/will produce solutions with non-physical oscillations. In this work, a rational RBF method is used to approximate derivatives of functions with steep gradients and discontinuities and to solve PDEs with such solutions. The method is non-linear and is more computationally expensive than the standard RBF method. A modified partition of unity method is discussed as an way to implement the rational RBF method in higher dimensions.

1. Introduction

Rational methods based on polynomials have a long history in approximation theory. Polynomial based rational methods are well-known for more accurately approximating functions with poles, steep gradients, and discontinuities than are polynomial methods. Examples of polynomial based rational methods are extensive and include [3,9,14]. Most polynomial based rational methods are of course tied to the same restrictive grids as are the algebraic or trigonometric polynomial methods that they are based on and thus are difficult or impossible to apply in complexly shaped domains. However, the recently described AAA algorithm [15] which employees polynomials appears much more flexible than previously described polynomial based rational methods and it is applicable in complexly shaped domains.

Previously described RBF methods that have a rational form include what the authors call a rescaled RBF method [4,12]. While the authors of [4,12] do not explicitly use the term rational, their scaled method has a rational form in which the denominator is the RBF interpolant to the constant function one and the numerator is the normal RBF interpolant. Reference [10] introduces a rational Radial Basis Function (RBF) method for the purpose of having a rational method that is applicable for the interpolation of scattered data points located in complexly shaped domains. In this work the method of [10] is extended to approximate derivatives and to solve partial differential equations with solutions featuring steep gradients, discontinuities, and shocks. In order to efficiently implement the method in higher dimensions, the method is localized via a modified partition of unity approach.

* Corresponding author.
E-mail address: sarra@marshall.edu (S.A. Sarra).

https://doi.org/10.1016/j.apnum.2018.04.001
0168-9274/

2. Radial basis function methods

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. The only restrictions on the center locations is that they must be unique. A RBF

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

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

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

where a is a vector of expansion coefficients. The inverse quadratic (IQ) RBF

$$\phi(r) = \frac{1}{1 + \varepsilon^2 r^2} \tag{3}$$

is used throughout in all examples. The IQ is a representative member of the class of strictly positive definite, global, infinitely differently RBFs that have a shape parameter. This class of RBF interpolates sufficiently smooth functions from the RBF's native space with exponential accuracy [6, chapters 14 and 15].

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

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

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

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

By linearity, the RBF interpolant can be differentiated as

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

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

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

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

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

Both equations (5) for the expansion coefficients and (10) for the differentiation matrix assume that the system matrix is invertible. The IQ system matrix is symmetric positive definite (SPD) and thus invertible. While invertible, the system matrix is typically very poorly conditioned. For a fixed set of centers, 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 of the RBF method and the condition number of the system matrix cannot both be kept small [23] when the standard basis functions are used. Other formulations, for example in references [7] and [5], of the RBF method that use a different basis that spans the same space as the standard basis but results in a better conditioned linear system do exist and are applicable is some applications. The methods are referred to as RBF-QR methods as a QR factorization is featured prominently in the approach. Reference [21] compares and

contrasts the application of the standard RBF method using both double and extended precision with the RBF-QR methods and applies the methods to a suite of numerical examples. For simplicity, this work only employs the standard RBF method. However, it is possible to replace the standard RBF method with the RBF-QR method in the rational RBF method that is described in this manuscript. Doing so may significantly improve the accuracy of the method but it would also add a great deal of complexity and increase computational expense of the method.

Recent monographs [2,6,22,28] on RBF methods can be consulted for more information.

3. A rational RBF method

The rational RBF method assumes an expansion of the form

$$\mathcal{R}_N f(x) = \frac{p(x)}{q(x)} \tag{11}$$

that is subject to the interpolation conditions

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

The numerator $p(x)$ and denominator $q(x)$ of the expansion are the standard RBF interpolants

$$p(x) = \sum_{k=1}^{N} a_k^p \phi\left(\left\|x - x_k^c\right\|_2, \varepsilon\right)$$

and

$$q(x) = \sum_{k=1}^{N} a_k^q \phi\left(\left\|x - x_k^c\right\|_2, \varepsilon\right)$$

to the vectors $\vec{p}$ and $\vec{q}$ which will be defined momentarily. In order for the rational interpolant to be uniquely defined, an extra condition is imposed which causes the native space semi-norms of the standard RBF interpolants $p(x)$ and $q(x)$ that are respectively the numerator and denominator of the rational interpolant to be minimized. That is, for example for the numerator $p(x)$, the quantity $(a^p)^* B a^p$ or equivalently $(\vec{p})^* B^{-1} \vec{p}$, is made as small as possible where B is the system matrix, a^p are the RBF expansion coefficients of $\vec{p}$, and $\vec{p}$ is the data being interpolated. The condition leads to a minimization problem with the solution $\vec{q}$ that is the eigenvector corresponding to the smallest eigenvalue of the eigenvalue problem

$$S\vec{q} = \lambda \vec{q} \tag{12}$$

where

$$S = diag\left(1/\left(\frac{f^2}{\|f\|_{\ell_2}^2} + 1\right)\right)\left(\frac{DB^{-1}D}{\|f\|_{\ell_2}^2} + B^{-1}\right) \tag{13}$$

and where $f = \left[f(x_1^c), \ldots, f(x_N^c)\right]$ contains the function values at the center locations, D is a diagonal matrix with f on the diagonal, B is the RBF system matrix, and $\|f\|_{\ell_2}^2 = \sum_{k=1}^{N} f_k^2$. Additionally in (13), the notation f^2 represents an elementwise squaring of the elements of the vector f and the division is elementwise as well. Since the RBF system matrix is SPD, so is the matrix S. After $\vec{q}$ is found, then the vector $\vec{p}$ is $\vec{p} = D\vec{q}$.

Once $\vec{p}$ and $\vec{q}$ are found, their standard RBF interpolants are formed by solving two linear systems, $Ba^p = \vec{p}$ and $Ba^q = \vec{q}$, for the expansion coefficients. The interpolants are evaluated by two matrix multiplications and the rational interpolant is then evaluated via

$$\mathcal{R}_N f(x) = \frac{Ha^p}{Ha^q}$$

where the division is element-wise and H is the RBF evaluation matrix (7).

The system matrix B may be very poorly conditioned. The system matrix may be efficiently regularized [19] by the method of diagonal increments (MDI) so that the condition number is reduced and so that it remains numerically SPD and can be factorized by a Cholesky factorization. Without MDI regularization, it is possible for a theoretically SPD but ill-conditioned matrix to not be SPD in floating point arithmetic which causes a Cholesky factorization to fail and forces the use of a more expensive LU factorization. MDI is a potential regularization tool for both the standard and rational RBF method. However, it has not been used to produce any of the numerical results in order to keep the focus on the rational method itself.

The rational interpolation method is summarized in Algorithm 1. Following the verbal description of each step is Matlab [13] code that executes the step.

Algorithm 1. ***Rational RBF interpolation.***

Inputs B, $N \times N$ system matrix based on centers x^c. H, $M \times N$ evaluation matrix based on evaluation points x. f, $N \times 1$ vector containing the function values at the center locations. ε, the RBF shape parameter. μ, an optional regularization parameter for the MDI.

Output f_r, a $M \times 1$ vector containing the values of the RBF rational interpolant evaluated at the evaluation points x.

1. *if $\mu > 0$ regularize B via the MDI by setting $B = B + \mu I$ where I is the $N \times N$ identity matrix.*
2. *Cholesky factorize B*
```
L = chol(B,'lower');
```
3. *Use the factorization to calculate the inverse of B*
```
Bi = L'\( L\eye(N) );
```
4. *Form the diagonal matrix D with f on the diagonal*
```
D = diag(f);
```
5. *Construct the matrix S given by equation (13)*
```
K = 1.0/sum(f.^2);
S = diag(1./( K*f.^2 + 1 ))*( K*D*Bi*D + Bi );
```
6. *Find the eigenvector q corresponding to the smallest eigenvalue of S*
```
opts.issym = 1;              % S is a real symmetric matrix
opts.isreal = 1;
[q,ew] = eigs(S,1,0,opts);
```
7. *Form the vector p*
```
p = D*q;
```
8. *Find the expansion coefficients of the standard RBF interpolants of p and q*
```
pAlpha = L'\( L\p );    % solve linear system B*pAlpha = p
qAlpha = L'\( L\q );    % solve linear system B*qAlpah = q
```
9. *Evaluate the rational interpolant at the evaluation points*
```
fr = (H*pAlpha)./(H*qAlpha);
```

In step 6 of Algorithm 1 the minimal eigenvector can be efficiently found using Arpack [11]. Arpack is available in major scientific software packages such as Matlab where it is available via the *eigs* function. The *eigs* function is passed two additional options to indicate that the problem is real and symmetric. Additionally, the *eigs* function uses a tolerance for determining the minimal eigenvector. The default value of the tolerance parameter is 1e-14 and this value was used in all cases. In all the numerical examples, the matrix S was slightly less ill-conditioned than the system matrix B, but S is still typically ill-conditioned. Despite the poor conditioning of S, the eigs function successfully calculated the minimal eigenvector in all examples.

Derivatives are calculated by applying the quotient rule to the rational interpolant. For instance, a partial derivative with respect to x based on the interpolant from step 9 of Algorithm 1 is calculated as

```
fx = ( (B*pAlpha).*(Hx*pAlpha) - (B*pAlpha).*(Hx*qAlpha) )./(B*qAlpha).^2;
```

where B is the system matrix and Hx is a derivative evaluation matrix (9).

Fig. 1 illustrates the results of using both the standard RBF (SRBF) method and the rational RBF (RRBF) method to interpolate a discontinuous step function. With a small number of centers (left image of Fig. 1) the SRBF method exhibits severe oscillations while the RRBF method only has a minor overshoot at the discontinuity. As N increases the RRBF method is both more accurate at the point of discontinuity and is significantly more accurate away from the discontinuity.

The results of Fig. 1 indicate that the rational RBF method may have advantages over the standard RBF method in approximating discontinuous functions, but both methods have sizable errors in the immediate neighborhood of the discontinuity. In order to illustrate the behavior of the two methods in approximating smooth functions, the methods are used to interpolate the function $f(x) = \exp(\sin(\pi x))$ on the interval $[-1, 1]$. Non-uniformly spaced centers that cluster mildly near the boundaries are used which are specified by the formula

$$x_k^c = -\frac{\arcsin\left[0.999\cos((k-1)\pi/(N-1))\right]}{\arcsin(0.999)}, \quad k = 1, 2, \ldots, N. \tag{14}$$

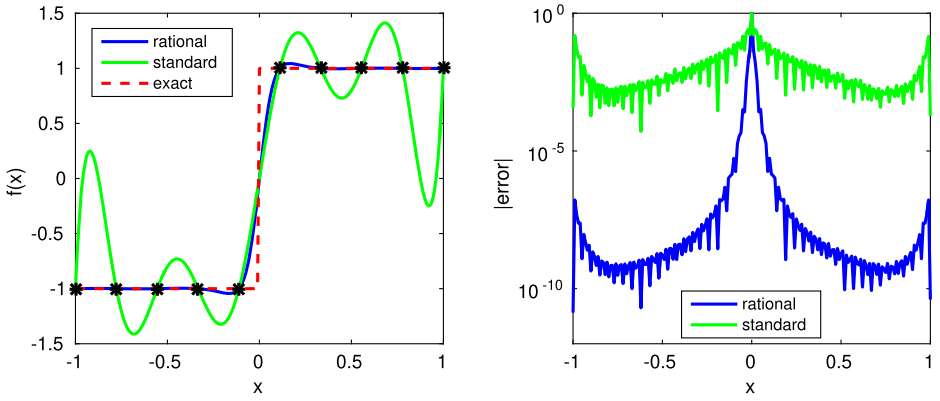

Fig. 1. Interpolation of a discontinuous function with the IQ RBF by the standard RBF method and the rational RBF method. Left: $N = 10$ evenly spaced centers, and shape parameter $\varepsilon = 0.45$. Right: pointwise errors for $N = 80$ and $\varepsilon = 3.5$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

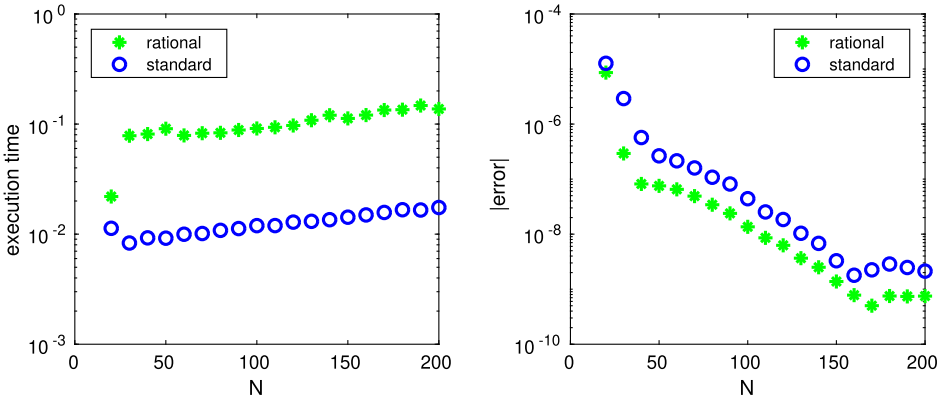

Fig. 2. Interpolation of the smooth function $f(x) = \exp(\sin(\pi x))$ by the standard and rational RBF methods. Left: execution time versus the number of centers N. Right: maximum errors versus N.

For each N, a shape parameter is used that results in a system matrix with a $\mathcal{O}\left(10^{16}\right)$ condition number. The interpolants are evaluated at 200 evenly spaced points. In the left image of Fig. 2 execution times versus N are shown indicating that for interpolation problems the rational method takes approximately 10 times longer to execute than does the standard method. In the right image of the figure the rational method is shown to be slightly more accurate than the standard method in this example over the entire range of N that is used.

In time-dependent PDE problems with a fixed set of centers and the same shape parameter for each time step, the setup phase can be calculated once before beginning time stepping, and then only steps from four on need to be completed during each time step. For differentiation both the standard and rational methods have the same flop count for the setup phase. The setup phase of the rational Algorithm 1 consists of steps one through three which have a combined dominant flop count term of $\frac{4}{3}N^3$. The setup phase of the standard method consists of factorizing the system matrix and then using the factorization to form the differentiation matrix. Next the first derivative of the function $f(x) = \exp(\sin(\pi x))$ is approximated by both methods using the same centers and shape parameters as the previous interpolation problem. The range of N is typical for 1d problems as well as for stencil or patch sizes of local methods in 2d or 3d. For each N the derivatives were calculated a thousand times and then the execution times were averaged. The right image of Fig. 3 shows the accuracy of the two methods versus N and the rational method is again slightly more accurate with each N. The left image of the figure shows the average execution times in which the rational method is significantly more computationally expensive. The differences in execution times are due to the standard method only requiring one matrix–vector multiplication whereas the rational method requires five matrix–vector multiplications, two forward substitutions, two back substitutions, and the solution of one eigenvalue problem. For smooth problems the small accuracy gains may not be worth the additional computation expense. However, for problems with discontinuities and steep fronts the rational method may be significantly more accurate and the additional computational expense may be justified.

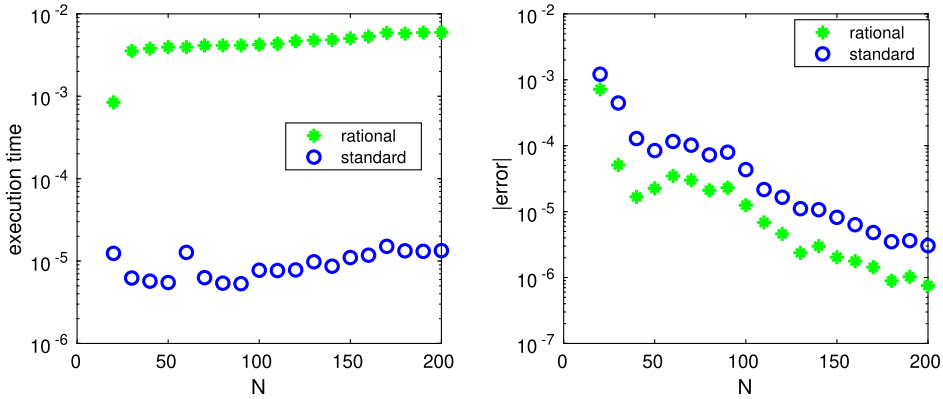

Fig. 3. First derivative approximation of the smooth function $f(x) = \exp(\sin(\pi x))$ by the standard and rational RBF methods. Left: execution time versus the number of centers N. Right: maximum errors versus N.

4. Efficient extension to higher dimensions

In order to efficiently implement the RRBF method in higher dimensional time-dependent PDE problems where derivatives may need to be evaluated thousands of times it is necessary to localize the method. The localization method used is a modified partition of unity method (PUM). In the context of scattered data interpolation, it appears that a partition of unity method was first described in [8] and was later analyzed in more detail in [27]. Reference [1] details the application of the PUM to the numerical solution of PDEs by the finite element method. Recently the PUM method has been used [17] to implement a local RBF collocation method based on the SRBF method for the numerical solution of PDEs.

A partition of unity method is implemented by constructing an overlapping covering $\{\Omega_i\}_{i=1}^M$ of the domain Ω. Each Ω_i in the covering is called a patch. For simplicity, it is assumed that the patches are circles with centers $c_i^0 = (x_i^0, y_i^0)$ and radii R_i. The left image in Fig. 10 displays an example cover of a complexly shaped domain. In general, it is not necessary that the patches be circular, as they may be shaped as squares, ellipses, etc. The PUM is applicable in higher dimensions as well, for example in 3d the patches could be cubes or spherical in shape. Associated with each center in Ω is the index function

$$P_i(x_k^c) = \left\{ i \,|\, x_k^c \in \Omega_i \right\} \quad i = 1, \ldots, M \tag{15}$$

which is used to keep track of how many patches that each center is located in.

Associated with the covering is a family of compactly supported, non-negative, continuous functions $\{w_i\}$ which are constructed via Shephard's method [24] as

$$w_i(x_k) = \frac{C_i(x_k^c)}{\sum_{j \in P_i(x_k^c)} C_j(x_k^c)} \quad i = 1, \ldots, M. \tag{16}$$

A possible choice for $C_i(x)$ in (16) is a compactly supported Wendland function [26] RBF such as

$$C(r) = (1-r)_+^4 (1+4r) \tag{17}$$

where

$$(1-r)_+^4 = \begin{cases} (1-r)^4 & 0 \le r < 1 \\ 0 & r > 1. \end{cases} \tag{18}$$

In this case, compact support on each patch is guaranteed if the Wendland functions (17) used to construct the weight functions (16) are specified as

$$C_i(x) = C\left(\frac{\|x - c_i\|_2}{R_i} \right).$$

As a result, the weight functions satisfy the partition of unity property

$$\sum_{i \in P_i(x_k^c)} w_i(x) = 1.$$

Additionally, $w_i(x) = 0$ if $i \notin P_i(x)$. A function f is approximated locally by s_i on each patch Ω_i. Then due to properties of the weight function the local approximants are put together as

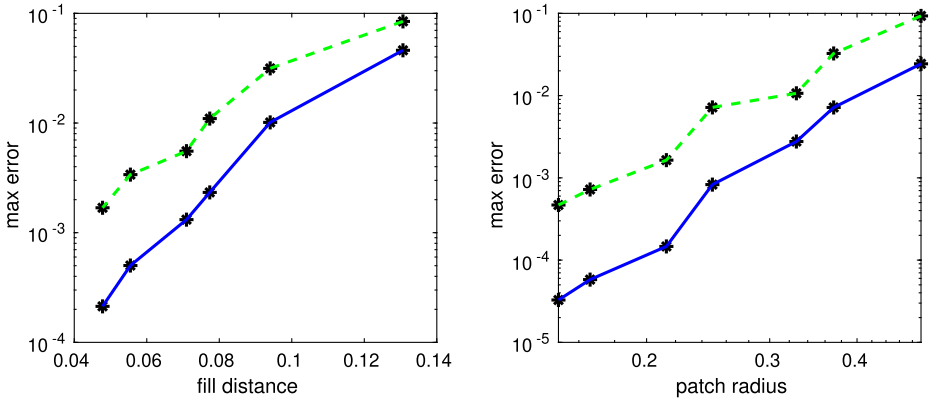

Fig. 4. RRBF partion of unity approximation of $\mathcal{D}_1 = f_x + f_y$ (solid line) and $\mathcal{D}_2 = f_{xx} + f_{yy}$ (dashed line) of the function $f(x, y) = e^{xy}$. Left: convergence trend with decreasing fill distance for a fixed number of patches and an increasing number of centers per patch. Right: convergence trend with an increasing number of patches with an approximately fixed number of centers per patch.

$$s(x) = \sum_{i \in P_i(x)} s_i(x) w_i(x). \tag{19}$$

The approximation at each x is a linear combination of the local approximations on each patch containing x with the largest weight being placed on patches where x is located near the center of the patch and less weight is placed on patch approximations where x is located near the boundary of a patch.

In a typical PUM approximation method such as [17], derivatives are approximated by applying a linear differential operator $\mathcal{D}$ to (19) as

$$\mathcal{D}s(x) = \sum_{i \in P_i(x)} \mathcal{D}\left[s_i(x) w_i(x)\right]. \tag{20}$$

As the order and complexity of $\mathcal{D}$ increase, the repeated use of the product rule can quickly result in an unwieldy expression containing a large number of derivatives to evaluate. Since the calculation of derivatives by the RRBF method already involves the quotient rule, a simplified PU method is used to reduce the computational cost of the method. In the simplified method, values of the weight function (16) still depend on the location of x within a patch, but then are considered constants rather than functions of x for approximating derivatives which the modified method calculates as

$$\mathcal{D}s(x) = \sum_{i \in P_i(x)} w_i \mathcal{D}s_i(x). \tag{21}$$

Derivative values at a point x are simply linear combinations of their values from each patch the point is contained in.

Fig. 4 examines the convergence of the RRBF PU method for approximating $\mathcal{D}_1 = f_x + f_y$ and $\mathcal{D}_2 = f_{xx} + f_{yy}$ of the smooth function $f(x, y) = e^{xy}$. The domain is a unit circle and the centers are quasi random Hammersley points [16]. The total number of radius 0.3736 circular patches is kept constant at sixteen and an increasing number of centers are added resulting in more centers per patch. The convergence can be examined in terms of the fill distance h which (left image of Fig. 4) is the radius of the largest possible empty ball that can be placed among the centers in any one patch. The convergence can also be examined as the number of patches is increasing (right image of Fig. 4) while the number of centers per patch is kept fixed (or approximately fixed as absolutely fixed is difficult with quasi random centers). As the patch radius is adjusted the ratio $radius/h$ is kept fixed at approximately 4.5. The convergence results can be compared to the convergence results of the RBF PU method in figure 10 of reference [17]. A theoretical analysis of the modified method as well as additional numerical results are needed in order for the modified method to be validated.

5. Numerical examples

Fig. 1 compared the results of interpolating a discontinuous function by the standard and rational RBF methods. The rational method can also be significantly more accurate in approximating continuous functions that have steep fronts or sharp gradients as is illustrated in section 5.1. Section 5.2 examines the solutions of the linear advection equation with a discontinuous initial condition, Burgers' equation with a solution featuring a steep front, and a nonlinear hyperbolic conservation law, inviscid Burgers' equation, with a smooth initial condition that develops a shock as the solution is advanced in time. Section 5.3 considers the interpolation of a 2d function with a steep front using scattered quasi random center locations. Finally, section 5.4 uses the modified PU method from section 4 to implement the RRBF method for the solution of the 2d Burgers' equation in a complexly shaped domain.

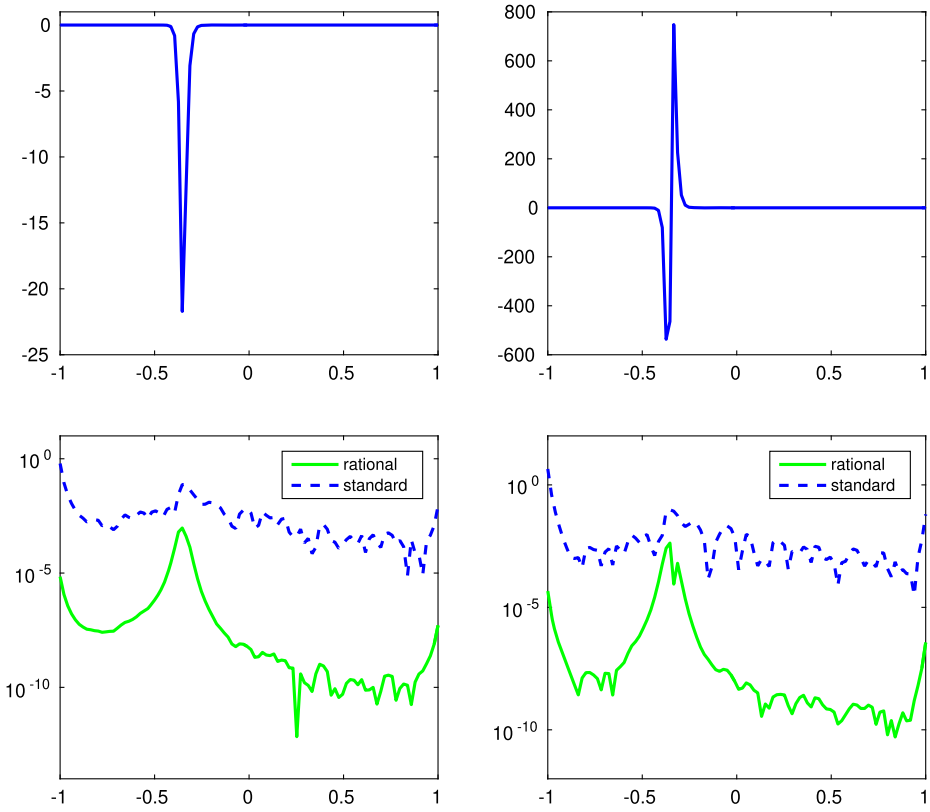

Fig. 5. Top left: first derivative of function (22) at $t = 0.53$. Top right: second derivative of function (22) at $t = 0.53$. Bottom left: first derivative relative pointwise errors. Bottom right: second derivative relative pointwise errors.

In all examples, the shape parameter was selected so that the condition number of the system matrix B (6) has a $\mathcal{O}\left(10^{16}\right)$ condition number. When implemented in double precision, which was used in all examples, a shape parameter selected in this way results in the RBF method producing the most accurate approximations. Both the standard and rational RBF methods invert the same system matrix and show the same sensitivity to changes in the shape parameter.

5.1. 1d interpolation and differentiation

The function

$$u(x,t) = \frac{0.1e^a + 0.5e^b + e^c}{e^a + e^b + e^c} \tag{22}$$

where $a = -(x + 0.5 + 4.95t)/(20\nu)$, $b = -(x + 0.5 + 0.75t)/(4\nu)$ and $c = -(x + 0.625)/(2\nu)$ is a solution to Burgers' equation (24) that is solved in section 5.2. With small values of ν, the function initially features two smaller steep fronts at $t = 0$. The two fronts merge into a larger steep front as t advances. The function is interpolated, and its first and second derivative are approximated at time $t = 0.53$ after the two small fronts have merged into a larger one by both the standard and rational RBF method. $N = 100$ uniformly spaced centers are used. A shape parameter of $\varepsilon = 4$ results in a system matrix with a $\mathcal{O}\left(10^{16}\right)$ condition number. Both methods factorize the same system matrix B so the conditioning issues of the two methods are the same. The standard method has relative maximum errors for interpolation and the first and second derivative respectively of 0.0290, 0.6195 and 4.4667 while the rational method errors are considerably smaller at 1.3762e-04, 9.3574e-04 and 0.0042. The derivatives and pointwise errors are shown in Fig. 5.

5.2. 1d time-dependent PDEs

After the time-dependent PDEs are discretized in space with a RBF method, the resulting semi-discrete system is advanced in time with an explicit fourth-order Runge–Kutta method with a constant time step size.

The first time-dependent problem is the advection equation

$$u_t + u_x = 0 \tag{23}$$

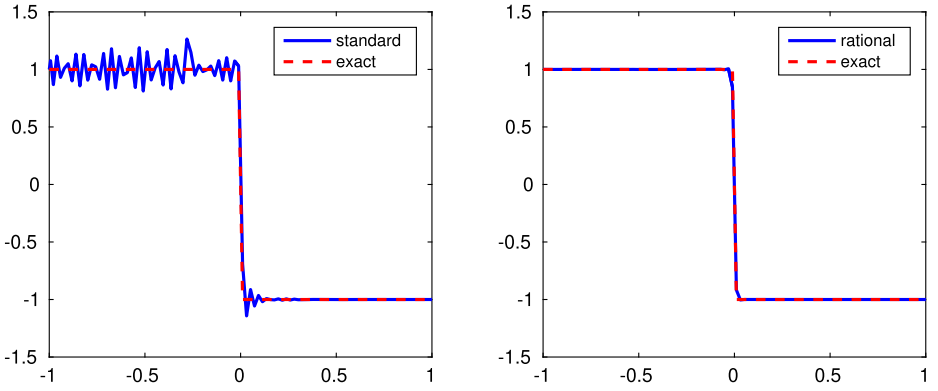

Fig. 6. Solution of the advection equation (23) at $t = 0.75$. Left: SRBF method. Right: RRBF method.

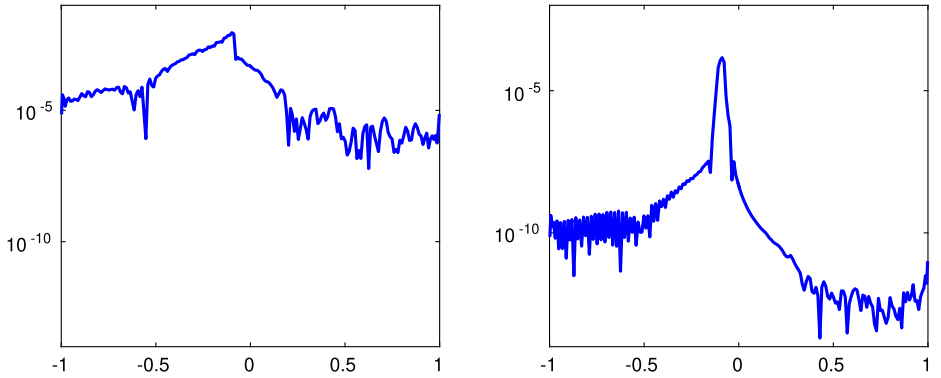

Fig. 7. Pointwise errors for the solution of the Burgers equation (24) at $t = 1$. Left: standard RBF method. Right: rational RBF method.

on the interval $\Omega = [-1, 1]$. The initial condition is a discontinuous step function and the boundary condition $u(-1, t) = 1$ is applied. As time advances the initial condition is propagated to the right with wave speed one. The non-uniformly spaced centers used for the problem cluster mildly around the boundaries and their location is given by the formula (14). The solution is advanced to time $t = 0.75$ by both the standard and rational RBF method and the result is illustrated in Fig. 6. Both methods use a shape parameter of $\varepsilon = 6$. The standard method solution exhibits spurious Gibbs oscillations whereas the rational solution does not.

Burgers' equation

$$u_t + uu_x = \nu u_{xx} \tag{24}$$

is solved on the interval $[-1, 1]$. The exact solution to the test problem is given by equation (22). The initial condition, $u(x, 0)$, and the boundary conditions $u(-1, t) = g_l(t)$ and $u(1, t) = g_r(t)$ are specified using the exact solution. The viscosity coefficient is taken as $\nu = 0.002$. The $N = 200$ centers are located according to equation (14) and a shape parameter of $\varepsilon = 8.5$ is used. The solution is advanced in time to $t = 1$ at which time the steep front is approaching the center of the domain. Fig. 7 shows the pointwise errors of the standard method in the left image and the rational method in the right image. The rational method is approximately two more decimal places accurate near the steep front and approximately five decimal places more accurate away from the front.

The next PDE problem is an nonlinear hyperbolic conservation law, inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0 \tag{25}$$

on the interval $[-1, 1]$. The smooth initial condition $u(x, 0) = \sin(\pi(x + 1))$ develops a discontinuity at time $t = \frac{1}{\pi}$ when a shock forms. Most high-order numerical methods such as the RBF method or the pseudospectral are unstable for this type of problem without some type of artificial viscosity [25] being added. With $N = 100$ centers located according to equation (14) and with shape parameter $\varepsilon = 6$ the standard method solution becomes unstable and blows up shortly after the shock forms in the solution. With the same settings, the RRBF method is able to accurately resolve the problem. The RRBF solution at time $t = 0.75$ for the problem is shown in the left image of Fig. 8 and the pointwise error is shown in the right image.

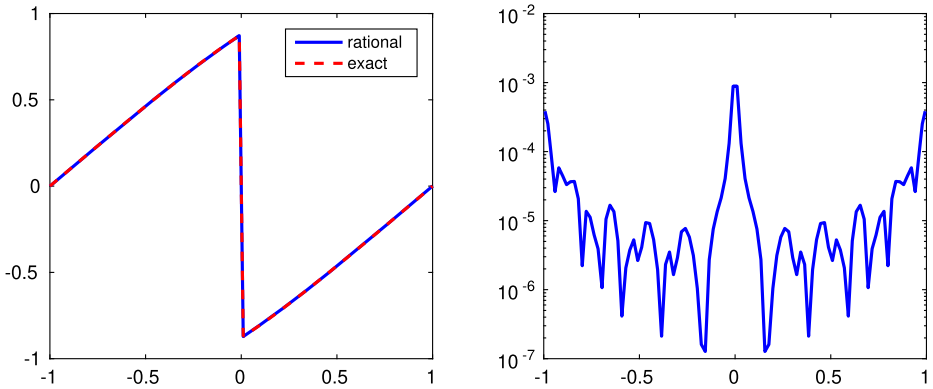

Fig. 8. Rational RBF solution of the inviscid Burgers equation (25) at $t = 0.75$. Left: the numerical versus the exact solution. Right: pointwise error from the solution in the left image.

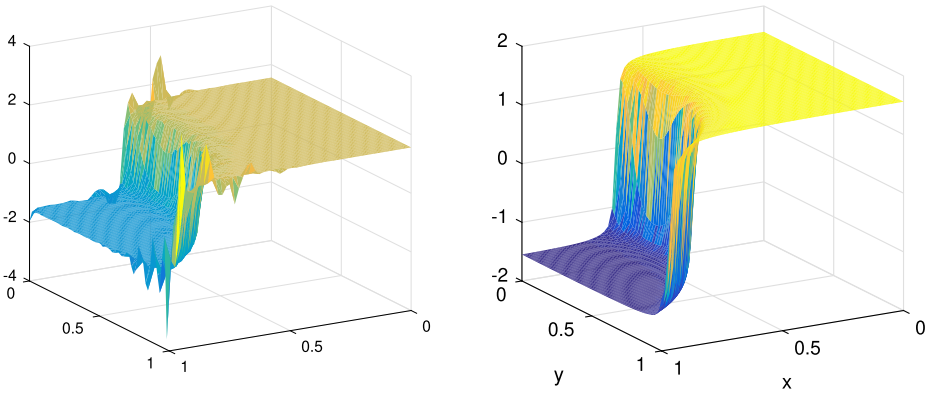

Fig. 9. Interpolation of function (26). Left: standard RBF method has non-physical oscillations near the steep front. Right: the rational RBF method produces a smooth solution with no visible oscillations.

5.3. 2d interpolation

This example interpolates a 2d function

$$f(x, y) = \arctan[125(\sqrt{(x-1.5)^2 + (y-0.25)^2} - 0.92)] \tag{26}$$

with a steep wave front located asymmetrically in the unit square. The function is interpolated on a set of centers consisting of $N = 5000$ quasi random Hammersley points. The interpolant is evaluated on a 3600 point uniformly spaced tensor product grid. A shape parameter of $\varepsilon = 8$ is used which results in the RBF system matrix having a $\mathcal{O}(10^{16})$ condition number. The standard method (left image of Fig. 9) has large visible oscillations and a 2.43 max error. The rational method (right image of Fig. 9) has no visible oscillations and a max error of 0.16.

5.4. 2d time-dependent PDEs

The rational method takes about 14 times longer to execute than the standard method on the 2d interpolation problem in section 5.3. While not a large issue for a one time interpolation, if the approximation is to be repeated a thousand of times as is the case of derivative approximation in a time-dependent PDE problem in more than one dimension the extended execution time may prohibit the rational RBF method from being implemented as a global method. The RRBF method can be implemented more efficiently in higher dimensions using the modified PU method described in section 4.

This example uses the modified PU approach to implement the RRBF method for the 2d Burgers' equation

$$u_t + uu_x + uu_y = \nu(u_{xx} + u_{yy}) \tag{27}$$

on scattered centers in the complexly shaped domain shown in the left image of Fig. 10 along with a PU covering consisting of 22 circular patches. A variety of coverings with various degrees of overlapping of patches may produce good results. Reference [17] recommends that the overlap of the patches be 20 percent of the distance between the centers of the patches. The exact solution of the problem is

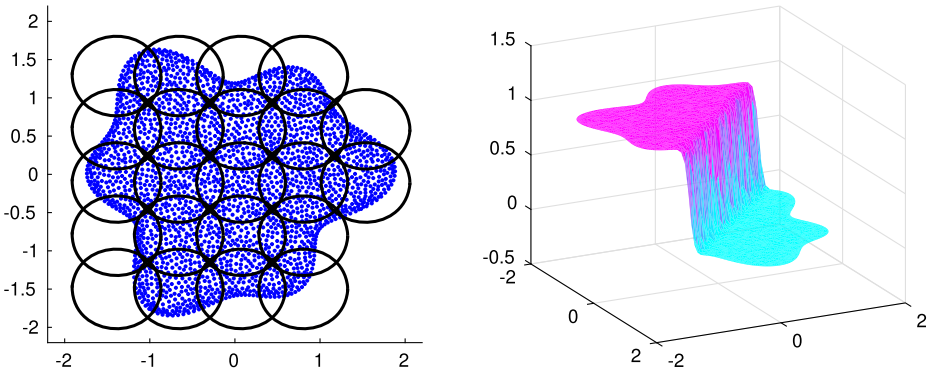

Fig. 10. Left: Centers and partion of unity cover for the solution of the 2d Burgers equation (27). Right: The rational RBF approximation of the solution of the 2d Burgers equation (27) at $t = 1$ with $\nu = 0.02$.

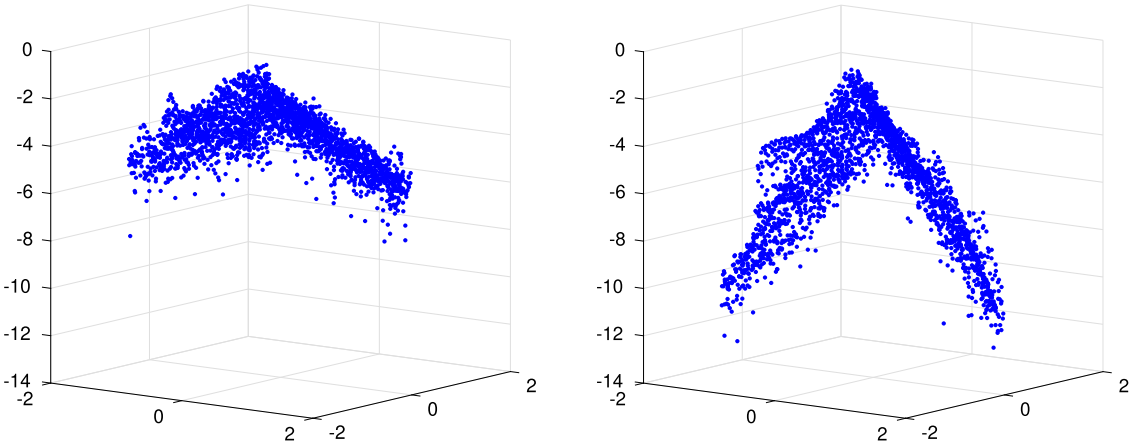

Fig. 11. 2d Burgers equation (27) absolute pointwise error on a log scale. Left: global SRBF method. Right: RRBF partion of unity method.

$$u(x, y, t) = \left(1 + e^{\frac{x+y-t+1}{2\nu}}\right)^{-1}. \tag{28}$$

The initial condition and Dirichlet boundary conditions are prescribed according to the exact solution.

The standard global RBF method and the local PU rational RBF method are both used to advance the PDE (27) with viscosity coefficient $\nu = 0.02$ to time $t = 1$. The maximum errors, which occur along the step front, are 0.028 for the standard global method and 0.020 for the local rational method. The rational method is significantly more accurate away from the steep front. The pointwise errors for each method are shown in Fig. 11. The rational method takes approximately 11 times longer to execute on this example. However, no attempt has been made yet to optimize the execution time of the RRBF PU method. Parts of the RRBF PU method could be implemented in parallel in order to realize shorter execution times.

6. Conclusions

The rational RBF method is capable of more accurately resolving problems featuring steep fronts and discontinuities than is the standard RBF method. In the numerical examples, the RRBF method was moderately more accurate in the neighborhood of a steep front or discontinuity and significantly more accurate away from the steep fronts and discontinuities. In one numerical example, the RRBF was able to accurately resolve the solution of a nonlinear hyperbolic conservation law without any added artificial viscosity.

When using a RBF, such as the IQ that has been used throughout, with a SPD system matrix the RRBF method is well-defined. In this case the RRBF method factorizes the system matrix B with a Cholesky factorization and finds the minimal eigenvector of a SPD matrix S. In time-dependent PDE problems where derivatives are calculated multiple times using the same set of centers and the same shape parameter, the system matrix B only needs to factorized once, B^{-1} only needs to be constructed once, and all derivative evaluation matrices only need to be constructed once in a set up phase of the algorithm.

The RRBF works with the same system matrix as does the SRBF method. Both methods are most accurate for values of the shape parameter that cause the system matrix to be ill-conditioned. The RRBF method can be regularized by the MDI in the same way that the SRBF is regularized in order to alleviate the conditioning problem.

A modified partition of unity approach has been used to more efficiently implement the method in two space dimensions. The modified method features a less computationally intensive approach to derivative approximation. While the modified PU method performed well on the numerical examples within it is not backed by theoretical support or extensive numerical examples. More work needs to be done in order for the modified method to be validated. The more computationally intensive standard PU derivative approximation approach (20) could be easily used with the rational method if a theoretically backed approach with a longer history of successful application is desired. With the partition of unity implementation of the rational RBF method, there is an opportunity for parallel implementation which could be used to significantly speed up the execution time of the method. An additional refinement that can be made to the RRBF PU approach is shape parameter selection on individual patches.

The class ***rbfRational*** which implements all the methods in this manuscript has recently been added to version 1.1 of the Matlab Radial Basis Function Toolkit (MRBFT) [20,18]. The ***rationalRbf*** folder in the examples directory of the MRBFT distribution contains scripts that carry out several of the examples in this manuscript.

References

[1] I. Babuka, J.M. Melenk, The partition of unity method, Int. J. Numer. Methods Eng. 40 (4) (1997) 727–758.
[2] M.D. Buhmann, Radial Basis Functions, Cambridge University Press, 2003.
[3] C.W. Clenshaw, K. Lord, Rational approximations from Chebyshev series, in: B.K.P. Scaife (Ed.), Studies in Numerical Analysis, Academic Press, 1974, pp. 95–113.
[4] S. Deparis, D. Forti, A. Quarteroni, A rescaled localized radial basis function interpolation on non-cartesian and nonconforming grids, SIAM J. Sci. Comput. 36 (6) (Jan. 2014) A2745–A2762.
[5] G. Fasshauer, M. McCourt, Stable evaluation of Gaussian RBF interpolants, SIAM J. Sci. Comput. 34 (2012) 737–762.
[6] G.E. Fasshauer, Meshfree Approximation Methods with Matlab, World Scientific, 2007.
[7] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM J. Sci. Comput. 33 (2011) 869–892.
[8] R. Franke, Locally determined smooth interpolation at irregularly spaced points in several variables, IMA J. Appl. Math. 19 (4) (1977) 471–482.
[9] J. Hesthaven, S. Kaber, L. Lurati, Padé–Legendre interpolants for Gibbs reconstruction, J. Sci. Comput. 28 (2–3) (2006) 337–359.
[10] S. Jakobsson, B. Andersson, F. Edelvik, Rational radial basis function interpolation with applications to antenna design, J. Comput. Appl. Math. (233) (2009) 889–904.
[11] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK Users Guide, Society for Industrial & Applied Mathematics (SIAM), Jan. 1998.
[12] S. De Marchi, A. Idda, G. Santin, A rescaled method for RBF approximation, in: Approximation Theory XV, San Antonio 2016, Springer International Publishing, 2017, pp. 39–59.
[13] MATLAB, version 9.1 (R2016b), The MathWorks Inc., Natick, Massachusetts, 2016.
[14] M.S. Min, S.M. Kaber, W.S. Don, Fourier–Padé approximations and filtering for spectral simulations of an incompressible Boussinesq convection problem, Math. Comput. 76 (2007) 1275–1290.
[15] Y. Nakatsukasa, O. Sete, L.N. Trefethen, The AAA algorithm for rational approximation, Preprint, https://arxiv.org/abs/1612.00337.
[16] H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods, CBMS-NSF, SIAM, Philadelphia, 1992.
[17] A. Safdari-Vaighani, A. Heryudono, E. Larsson, A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications, J. Sci. Comput. 64 (2) (Oct. 2014) 341–367.
[18] S.A. Sarra, A Matlab radial basis function toolkit with symmetry exploitation, regularization, and extended precision, http://www.scottsarra.org/rbf/rbf.html.
[19] S.A. Sarra, Regularized symmetric positive definite matrix factorizations for linear systems arising from RBF interpolation and differentiation, Eng. Anal. Bound. Elem. 44 (2014) 76–86.
[20] S.A. Sarra, The Matlab radial basis function toolbox, J. Open Res. Softw. 5 (March 2017).
[21] S.A. Sarra, S. Cogar, An examination of evaluation algorithms for the RBF method, Eng. Anal. Bound. Elem. 75 (Feb. 2017) 36–45.
[22] S.A. Sarra, E.J. Kansa, Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations, Adv. Comput. Mech. 2 (2009).
[23] R. Schaback, Error estimates and condition numbers for radial basis function interpolation, Adv. Comput. Math. 3 (1995) 251–264.
[24] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 1968 23rd ACM National Conference on, Association for Computing Machinery (ACM), 1968.
[25] E. Tadmor, Convergence of spectral methods for nonlinear conservation laws, SIAM J. Numer. Anal. 26 (1989) 30–44.
[26] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree, Adv. Comput. Math. 4 (1995) 389–396.
[27] H. Wendland, Fast evaluation of radial basis functions: methods based on partition of unity, in: Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, 2002, pp. 473–483.
[28] H. Wendland, Scattered Data Approximation, Cambridge University Press, 2005.