Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 53 (2007) 927–939
An International Journal
computers & mathematics
with applications
www.elsevier.com/locate/camwa

Adaptive residual subsampling methods for radial basis function interpolation and collocation problems

Tobin A. Driscoll*, Alfa R.H. Heryudono

Department of Mathematical Sciences, University of Delaware, Newark, DE, 19716, United States

Received 31 January 2006; received in revised form 1 June 2006; accepted 15 June 2006

Abstract

We construct a new adaptive algorithm for radial basis functions (RBFs) method applied to interpolation, boundary-value, and initial-boundary-value problems with localized features. Nodes can be added and removed based on residuals evaluated at a finer point set. We also adapt the shape parameters of RBFs based on the node spacings to prevent the growth of the conditioning of the interpolation matrix. The performance of the method is shown in numerical examples in one and two space dimensions with nontrivial domains.

Keywords: Adaptive; Radial basis functions; Interpolation; Collocation; Residual subsampling

1. Introduction

Radial basis function (RBF) methods have been praised for their simplicity and ease of implementation in multivariate scattered data approximation [1], and they are becoming a viable choice as a method for the numerical solution of partial differential equations (PDEs) [2–5]. Compared to low-order methods such as finite differences, finite volumes, and finite elements, RBF-based methods offer numerous advantages, such as no need for a mesh or triangulation, simple implementation, dimension independence, and no staircasing or polygonization for boundaries. Moreover, depending on how the RBFs are chosen, high-order or spectral convergence can be achieved [6,7].

In problems that exhibit high degrees of localization in space and/or time, such as steep gradients, corners, and topological changes resulting from nonlinearity, adaptive methods may be preferred over fixed grid methods. The goal is to obtain a numerical solution such that the error is below a prescribed accuracy with the smallest number of degrees of freedom. Since RBF methods are completely meshfree, requiring only interpolation nodes and a set of points called *centers* defining the basis functions, implementing adaptivity in terms of refining and coarsening nodes is very straightforward.

In recent years several researchers have incorporated RBF methods in several adaptive schemes in a number of time-independent and time-dependent settings, either in a primary or supporting role. In [8] Bozzini et al. combine B-spline techniques with RBFs, specifically scaled multiquadrics, as an adaptive method for interpolation. Instead of

* Corresponding author.
E-mail addresses: driscoll@math.udel.edu (T.A. Driscoll), heryudon@math.udel.edu (A.R.H. Heryudono).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2006.06.005

using piecewise linear spline interpolants as in standard B-spline techniques, scaled multiquadrics provide smoother interpolants and maintain superior shape preserving properties (positivity, monotonicity, and convexity). Schaback et al. [9] and Hon et al. [10] propose adaptive methods based on a greedy algorithm and best n-term approximation using compactly supported RBFs for interpolation and collocation problems. The convergence of their methods is known to be linear. In problems with a boundary layer Hon [11] proposes an adaptive technique using multiquadrics. An indicator based on the weak formulation of the governing equation is used to adaptively re-allocate more points to the boundary layer. In the time-dependent case Behrens et al. [12,13] combine an adaptive semi-Lagrangian method with local thin plate splines interpolation. The interpolation part itself plays a crucial role in efficiency, approximation quality, and rules of adaptivity of the method. Their adaptive method performs well on nonlinear transport equations. In [14] Sarra modifies a simple moving grid algorithm, which was developed for use with low-order finite difference method, to RBF methods for time-dependent PDEs. The method is essentially the method of lines with RBFs implementation of the equidistribution of arclength algorithm in space.

The purpose of this article is to present a new method for adaptive RBF in time-independent problems, specifically in interpolation and boundary-value problems, based on *residual subsampling*. We start with nonoverlapping boxes, each of which contains an active center. Once an interpolant has been computed for the active center set, the residual of the resulting approximation is sampled on a finer node set in each box. Nodes from the finer set are added to or removed from the set of centers based on the size of the residual of the interpolation or the PDE at those points. The interpolant is then recomputed using the new active center set for a new approximation. The choice of shape parameters of RBFs is also crucial to the method. The procedure for adapting the shape parameters will be explained in Section 3.1.

Our method is easily implemented in any number of dimensions and with any geometry. We demonstrate the effectiveness of this technique for interpolation in one dimension, and in square and nontrivial 2D regions. We also show that it can be applied to linear and nonlinear boundary-value problems. Finally, we show that the method may also be effective for time-dependent problems, either as a time-step post-processor or by operating directly in space–time.

2. Radial basis functions

Given data at nodes $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in d dimensions, the basic form for an RBF approximation is

$$F(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi_{\epsilon_j}(\|\mathbf{x} - \mathbf{x}_j\|), \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean distance between two points and $\phi(r)$ is defined for $r \geq 0$. Given scalar function values $f_i = f(\mathbf{x}_i)$, the expansion coefficients λ_j are obtained by solving a system of linear equation

$$\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \tag{2}$$

where the *interpolation matrix* A satisfies $a_{ij} = \phi_{\epsilon_j}(\|\mathbf{x}_i - \mathbf{x}_j\|)$. Common choices of $\phi(r)$ are:

- Infinitely smooth with a free parameter: Multiquadrics ($\phi_\epsilon(r) = \sqrt{1 + (\epsilon r)^2}$) and Gaussian ($\phi_\epsilon(r) = \mathrm{e}^{-(\epsilon r)^2}$);
- Piecewise smooth and parameter-free: Cubic ($\phi(r) = r^3$), thin plate splines ($\phi(r) = r^2 \ln r$);
- Compactly supported piecewise polynomials with free parameter for adjusting the support: Wendland functions [15].

For many choices of ϕ with mild restrictions, constant shape parameters, and possibly including minor modifications to (1), the *interpolation matrix* A is guaranteed to be nonsingular [1,16]. Under certain conditions, the infinitely smooth radial functions exhibit exponential or spectral convergence as a function of center spacing, while the piecewise smooth types give algebraic convergence [6,7,15,17]. The appeal of compactly supported functions is that they lead naturally to banded interpolation matrices, although experience has shown that the matrices need to be large for good accuracy.

In this article we focus on the popular choice of multiquadrics. Multiquadrics respond rather sensitively to the shape parameter ϵ. For example, in one dimension, the limit $\epsilon \to \infty$ produces piecewise linear interpolation, and the 'flat limit' $\epsilon \to 0$ produces global polynomial interpolation [18]. Hence smaller values of ϵ are associated with more accurate approximation. Many rules of thumb for parameter selection are known from experience and theory [19–21].

The use of node-dependent shape parameters ϵ unfortunately breaks the symmetry of the interpolation matrix A as well as the proof of its nonsingularity. In practice, however, these properties do not prevent A from becoming so ill-conditioned as to be essentially singular, and we observe the benefits of varying ϵ as described below to be substantial.

In addition to the advantages mentioned in the previous section, RBF methods have serious drawbacks. As the number of centers grow, the method needs to solve a relatively large algebraic system. Moreover, severe ill-conditioning of the interpolation matrix A causes instability that makes spectral convergence difficult to achieve. This behavior is manifested as a classic accuracy and stability trade-off or uncertainty principle; for instance, the condition number of A, $\kappa(A)$, grows exponentially with N [22]. For small N, it is possible to compute the coefficient vector λ accurately, using complex contour integration [23], but the robustness of this method for large N is not yet tested. If one wishes to use an iterative method to solve for the interpolation coefficients, ill conditioning can also create a serious convergence issue. In such cases good preconditioners are needed [24].

Nor does all of the potential instability result from linear algebra. Node and center locations also play a crucial role through the classical problem of interpolation stability, as measured by Lebesgue constants and manifested through the Runge phenomenon. It is clear, for example by comparison to the flat limit of polynomial interpolation, that spectral convergence results such as those cited above must be limited to certain classes of functions that are well-behaved beyond analyticity in the domain of approximation. This is thoroughly described in [25] for the special case of Gaussian RBFs in one dimension. Just as in polynomial potential theory, for the limit $N \to \infty$ an interpolated function must be analytic in a complex domain larger than the interval of approximation, unless a special node density is used. This density clusters nodes toward the ends of the interval, in much the same way as nodes based on Jacobi polynomials do, in order to avoid Runge oscillations. The existence and construction of stable node sets in higher dimensions and general geometries, in particular with regards to proper clustering near the boundary, remains a very challenging problem [26].

3. Residual subsampling method

In this section, we describe a simple, easily implemented method for adapting RBF centers and parameters based on the results of the interpolation process. We call our method *residual subsampling*. In order to introduce the method, we pick a simple 1D interpolation problem using Runge function $f(x) = (25x^2 + 1)^{-1}$ on the interval $[-1, 1]$. First, we generate an initial discretization using N equally spaced points and find the RBF approximation of the function. Next, we compute the interpolation error at points halfway between the nodes. Points at which the error exceeds a threshold θ_r are to become centers, and centers that lie between two points whose error is below a smaller threshold θ_c are removed. The two end points are always left intact. The shape parameter of each center is chosen based on the spacing with nearest neighbors, and the RBF approximation is recomputed using the new center set. In short, the adaptation process follows the familiar paradigm of solve–estimate–refine/coarsen until a stopping criterion is satisfied. In MATLAB, this problem can be coded in modest number of lines.

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

refine = true;
while any(refine)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;

  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda - f(y));

  coarsen = resid(1:N-2) < thetac & resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
  refine = resid > thetar; x = sort([x;y(refine)]);
end
```

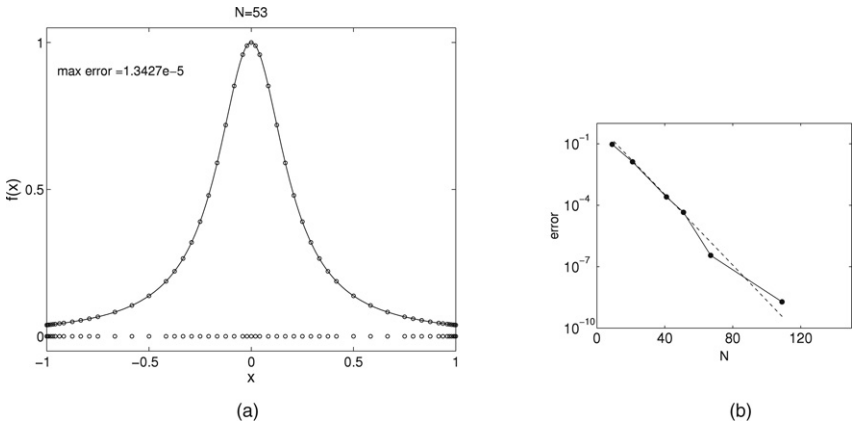


Fig. 1. Left: Runge function with final RBF node distribution. Right: Error convergence of residual subsampling method compared to standard Chebyshev interpolation (dashes).

A version of the code with graphics can be downloaded from [27]. Fig. 1 shows the final result obtained from running the MATLAB code. In this case, the adaptative iteration automatically clusters nodes near the boundaries to counter oscillations while also resolving regions of high activity adequately. The convergence is evidently spectral and virtually identical to Chebyshev. Although the adaptive method above seems adequate for our example, in the rest of the paper we describe and use a slightly different version that is more easily generalized to higher dimensions.

3.1. Residual subsampling for interpolation

We begin with the problem of interpolating a known function $f(\mathbf{x})$. The method begins with a coarse collection of nonoverlapping regular boxes in R^d that cover the domain Ω of interest. Each initial box has an RBF center at its midpoint, and we call such boxes *center boxes*. Each of the 2^d child boxes arising from the natural subdivision (bisection) of a center box is a *check box*, and the midpoint of each check box is a residual *check point*. Henceforth discussion will proceed in $d = 2$ dimensions for concreteness.

After initial boxes are established, the next stage is geometric refinement. Any center box lying entirely outside Ω is deleted, and any lying entirely inside is accepted without refinement. For each center box that intersects the boundary of Ω, we divide into two cases:

- Case 1: If the midpoint of the center box is outside Ω, its associated check points are tested for membership in Ω. For any check points which lie inside Ω, the associated boxes are promoted to center boxes (themselves acquiring four check points each). For those which lie outside Ω, the test-and-promote step is applied to its children. These can be done recursively for up to three generations or even more. At the bottom of the recursion, center boxes whose midpoints are outside Ω are simply deleted.
- Case 2: If the midpoint of the center box is inside Ω, its box is accepted. However, we do not stop here since further refinement may be needed. We again test its check points for membership in Ω. For any check points which lie inside, we do nothing. For those which lie outside, we promote its boxes to center boxes and treat them as case 1.

To summarize, the geometric adaptation is nothing more than finding a set of center boxes, whose midpoints are inside Ω, which more or less represent the domain Ω. An example of geometric refinement up to first generation for a test domain Ω is illustrated in Fig. 2.

This simple method has proved sufficient in our tests so far but is probably not optimal. Given initial boxes, a robust method for finding the best initial geometric refinement is still under investigation. However, the tremendous flexibility of RBFs allows for all kinds of tweaks to the process without causing any conceptual difficulty. Indeed, in order to improve the resolution of the geometry (particularly for the boundary-value problems shown in the next section), we place additional RBF centers along $\partial\Omega$. This can be done using centers that remain fixed throughout the rest of the solution process, or by applying the 1D form of our adaptive method to a parametrization of the

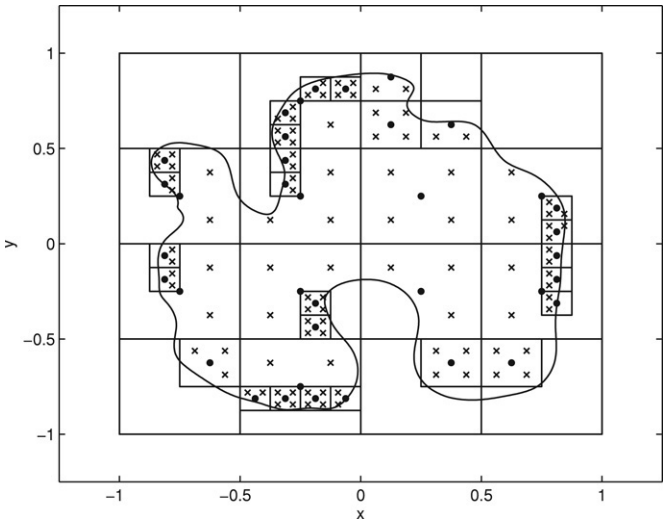


Fig. 2. An example of geometric refinement for some domain Ω. Initially, coarse collection consisting of 4×4 nonoverlapping boxes is used to cover the domain and then the domain is refined up to first generation. Accepted RBF centers are shown as dots, and interior residual check points are crosses.

boundary. Another tweak of geometric refinement process is to use RBF centers outside Ω [28]. However, a different implementation of the method will be required in order to place centers outside the domain since we have to separate interpolation nodes and centers. There is no obvious advantage for doing so but it will be an interesting topic for future study.

After geometric refinement, the main iteration begins. An RBF interpolant is created using the current set of centers from the center boxes. The residual η (in this case, the interpolation error) of the interpolant is then evaluated at the check points, and adaptation decisions are made. At any check points where the residual exceeds a predefined threshold θ_r, we convert that check box to a center box and add its children as check points. This is the refinement portion of the algorithm. We also allow coarsening of the centers: for any center whose child check points all have residuals less than a smaller threshold θ_c, the check boxes are discarded and the center box itself becomes a check box.

Once the refinement and coarsening decisions have been made, the RBF interpolant is recomputed and the process is repeated. Note that the four sibling check points are tested independently for addition to the set of centers. The quadtree-based refinement we describe here is the simplest way to implement multiscale adaptivity and it generalizes to higher dimensions effortlessly.

We also adapt the shape parameter ϵ. At the top level, each center box is assigned a value of parameter, say ϵ_g, chosen according to the coarse node spacing. Whenever a check box is converted to a center box, its value of ϵ is twice its parent's value, since centers at this level are twice as close together. In addition, the parent's value is also doubled. In some sense, the shape of the basis functions is kept constant on all scales as defined by local node spacing. This choice of ϵ refinement turns out to be quite important to the algorithm; when we try to increase it more slowly, in order to get spectral accuracy as we descend in length scales, the condition number of the interpolation matrix climbs until numerical singularity, and the method often fails. With our choice, though, the conditioning usually remains essentially constant after a couple of iterations. The choice of ϵ_g is also important: when it is smaller, indicating more global/spectral behavior, the refinement is less obviously tied to features in f; when it is larger, the nodes cluster in an intuitively clear way, though they may also be more numerous.

3.2. Residual subsampling for boundary-value problems

Our adaptation method can be easily extended to solve linear or nonlinear boundary-value problems. In place of an interpolation matrix A, we have a more general matrix that collocates at the interior centers and the boundary conditions at boundary centers. For instance, in Poisson's equation with zero boundary conditions, the matrix

becomes

$$A\boldsymbol{\lambda} = \begin{bmatrix} \Delta\phi_{\epsilon_1}(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \dots & \Delta\phi_{\epsilon_N}(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \vdots & & \vdots \\ \Delta\phi_{\epsilon_1}(\|\mathbf{x}_k - \mathbf{x}_1\|) & \dots & \Delta\phi_{\epsilon_N}(\|\mathbf{x}_k - \mathbf{x}_N\|) \\ \phi_{\epsilon_1}(\|\mathbf{x}_{k+1} - \mathbf{x}_1\|) & \dots & \phi_{\epsilon_N}(\|\mathbf{x}_{k+1} - \mathbf{x}_N\|) \\ \vdots & & \vdots \\ \phi_{\epsilon_1}(\|\mathbf{x}_N - \mathbf{x}_1\|) & \dots & \phi_{\epsilon_N}(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \\ \lambda_{k+1} \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} g(\mathbf{x}_1) \\ \vdots \\ g(\mathbf{x}_k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{3}$$

where $\{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ and $\{\mathbf{x}_{k+1}, \dots, \mathbf{x}_N\}$ are the set of interior and boundary centers respectively and Δ is operating on the first variable. Once the system of linear equations has been solved for the RBF expansion coefficients, the residual of the PDE is evaluated at the check points as before. For instance, in Poisson's equation, we use

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_M \end{bmatrix} = \begin{bmatrix} \left| \sum_{j=1}^{N} \lambda_j \Delta\phi_{\epsilon_j}(\|\mathbf{y}_1 - \mathbf{x}_j\|) - g(\mathbf{y}_1) \right| \\ \vdots \\ \left| \sum_{j=1}^{N} \lambda_j \Delta\phi_{\epsilon_j}(\|\mathbf{y}_M - \mathbf{x}_j\|) - g(\mathbf{y}_M) \right| \end{bmatrix} \tag{4}$$

to measure solution quality at the set of check points $\{\mathbf{y}_1, \dots, \mathbf{y}_M\}$. As an alternative, one can instead use

$$\tilde{\boldsymbol{\eta}} = \begin{bmatrix} \tilde{\eta}_1 \\ \vdots \\ \tilde{\eta}_M \end{bmatrix} = \begin{bmatrix} \eta_1 \times S_1 \\ \vdots \\ \eta_M \times S_M \end{bmatrix}, \tag{5}$$

where S_i denotes the area (in 2D) of the check box in which $\mathbf{y}_i$ is midpoint. $\tilde{\eta}$ is intended to mimic the 1-norm of the residual, whereas η is more naturally related to the ∞-norm. Typically, η leads to more stringent conditions on the solution than $\tilde{\eta}$ does. We do not treat either η or $\tilde{\eta}$ as a rigorous estimate of the error, but as an indicator to refine or coarsen centers.

4. Numerical experiments

We now present the results of numerical experiments in 1 and 2 dimensions. All programs are run in MATLAB on Linux P4 running at 1.4 GHz. The top-level global parameter ϵ_g is chosen in the interval $[1, 10^3]$. The choice of θ_r and θ_c depends on tested problems. Most examples took under 30 s of CPU time to converge.

4.1. Interpolation

In order to be consistent, we modified the MATLAB code given in the previous section to use the 1D implementation of the box-based method. As models for 1D interpolation, functions with steep gradient and corner features were chosen. A finer grid containing 5001 points in the domain is used to measure the errors. Table 1 shows the adaptive steps of interpolating $\tanh(60x - .01)$ with $\epsilon_g = 4$. The adaptive iteration converges in 7 iterations with $N = 129$ final total centers. Table 1 shows that the conditioning of the interpolation matrix grows quickly and then levels off, a pattern that is repeated in all of our experiments. Given initial boxes, it is still not clear how to choose the global ϵ_g optimally. With $\epsilon_g = 10$ we end with $N = 110$ centers after 10 iterations. Fig. 3 shows how the nodes distribute themselves near the steep gradient, and also how, as θ_r is reduced, the errors in our method compare very favorably to standard Chebyshev interpolation.

In Fig. 4 we present results for the C_0 function $f(x) = |x + .04|$. While the convergence is not spectral, it is considerably better than for unadapted Chebyshev interpolants. Even though the strengths of RBFs are usually not evident in 1D, our results are more than satisfactory.

Table 1
Iterative progress of the refinement algorithm for interpolation of $\tanh(60x - 0.1)$

It	N	N_c	N_r	$\kappa(A)$	$\|.\|_\infty$
1	11	0	18	5.090e+02	7.9211e−01
2	29	0	34	2.632e+04	4.1501e−01
3	63	0	31	4.545e+05	1.2544e−01
4	94	3	30	4.330e+06	1.1129e−02
5	121	4	12	3.962e+07	2.6766e−04
6	129	2	2	3.180e+08	5.7980e−05
7	129	0	0	3.038e+08	2.5329e−05

N_r, N_c = Number of centers to be added/removed respectively.

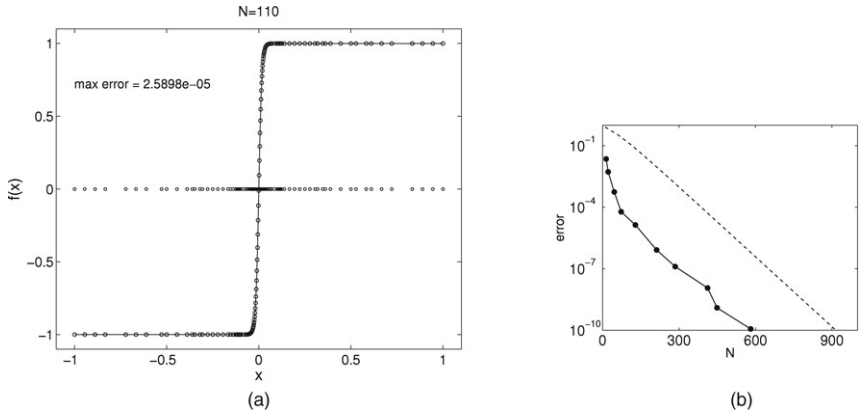


Fig. 3. Left: $\tanh(60x - .01)$. Right: Error convergence of residual subsampling method compared to standard Chebyshev interpolation (dashes).

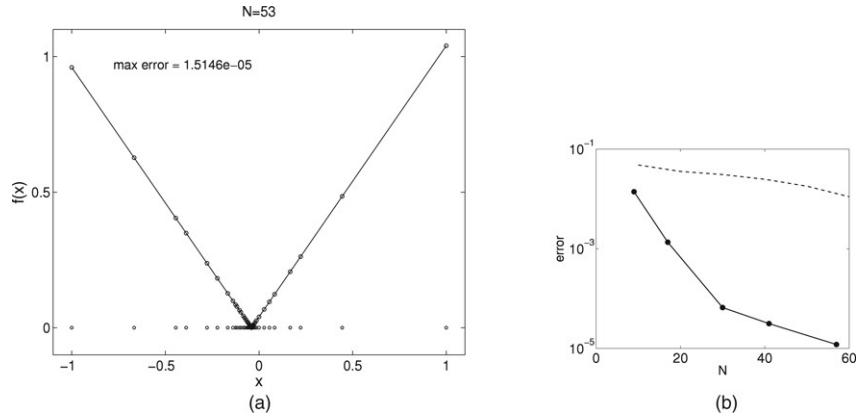
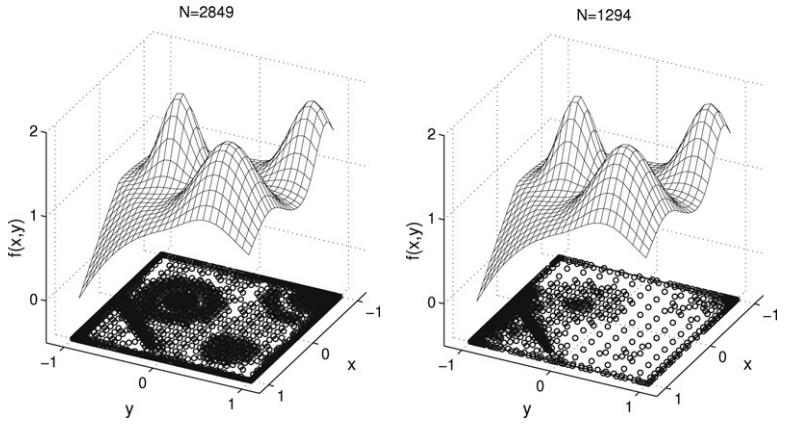


Fig. 4. Left: $|x + .04|$, the box-based method adaptively refines centers around corners. Right: Error convergence of residual subsampling method compared to standard Chebyshev interpolation (dashes).

Now we move on to 2D cases. In addition to adapting interior points, we also adapt points on the boundary. Our first test is with the Franke-type function [29] in $[-1, 1]^2$, which has been used many times to test for RBF interpolation:

$$f(x, y) = \mathrm{e}^{-0.1(x^2+y^2)} + \mathrm{e}^{-5((x-0.5)+(y-0.5)^2)} + \mathrm{e}^{-15((x+0.2)^2+(y+0.4)^2)} + \mathrm{e}^{-9((x+0.8)^2+(y-0.8)^2)}, \tag{6}$$

modified by introducing a singularity of lower-order derivatives along the line $y - x = -1$ by taking $f(x, y) - (y - x + 1)y$ instead of $f(x, y)$ for $y - x < -1$, see [9]. Only six iterations are needed in reaching the desired residuals 8.82×10^{-4} in interior and 4.24×10^{-5} on boundary for both $\epsilon_g = 10^2$ and $\epsilon_g = 4$. As can be seen from Fig. 5, the two versions look quite different. When $\epsilon_g = 10^2$, the multiquadrics are far from spectral and the final



(a) $\epsilon_g = 10^2$. (b) $\epsilon_g = 4$.

Fig. 5. Local and global features of modified Franke function.

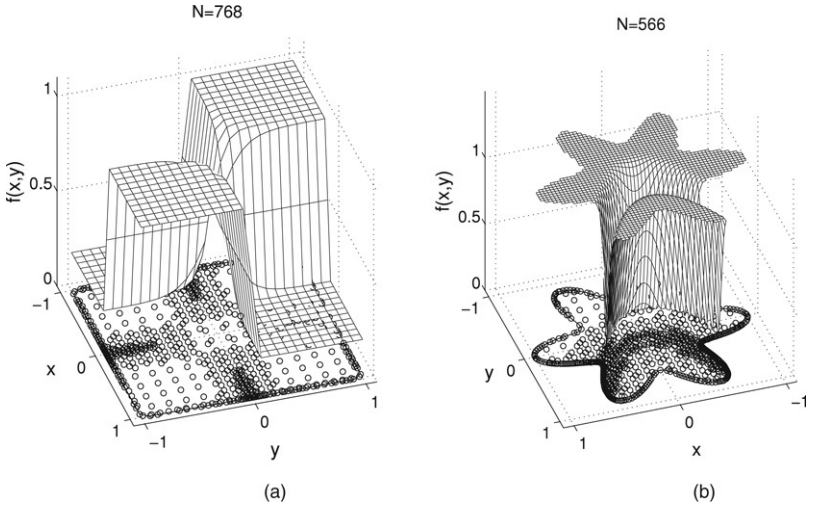


Fig. 6. Centers are located neatly around regions with steep gradients.

$N = 2849$ centers are distributed in accordance with the local function behavior. On the other hand, $\epsilon_g = 4$ tends to be more global and the $N = 1294$ centers are less clearly correlated to local features of f. Moreover, the savings are also significant. From our numerical experiments, the results are basically comparable to those of [9] whether boundary adaptation is used or not. Our second and third tests are $f(x, y) = -0.4 \tanh(50xy) + 0.6$ in $[-1, 1]^2$, and $f(x, y) = -e^{-60(y-2x^2)^2} + 1.2$ in a smooth flower-like region. Again, the adaptive method performs well, locating centers neatly near the rapid variation regions. See Fig. 6. In both cases, fewer than 10 iterations are needed with 768 total centers for the first case and 566 centers for the flower region to reduce the residuals to 1% in the interior.

4.2. Boundary-value problems

We start with stationary Burgers' equation on [0, 1] with Robin condition, as given in [30]:

$$\begin{aligned} &\nu u_{xx} - u u_x = 0 \\ &\nu u_x(0) - \kappa(u(0) - \alpha) = 0 \\ &\nu u_x(1) + \kappa(u(1) + \alpha) = 0, \end{aligned}$$

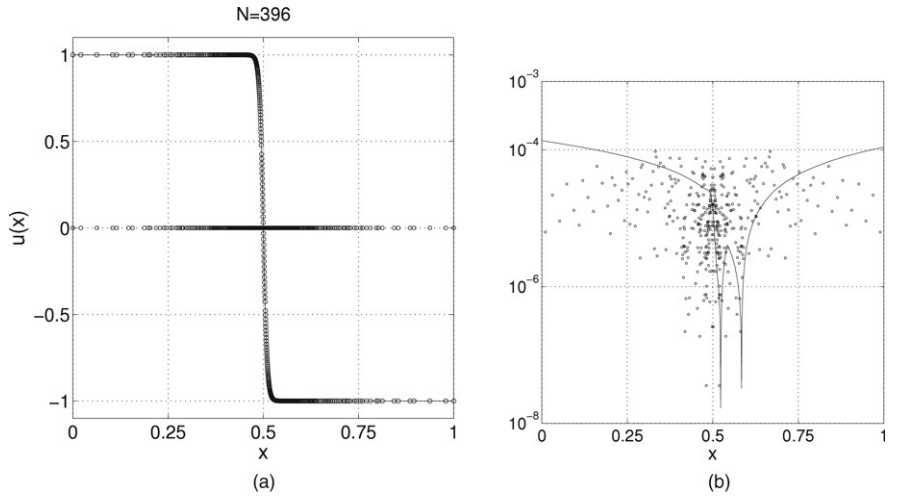


Fig. 7. Left: Numerical solution of stationary viscous Burgers' equation with Robin condition. Right: Solid lines are the absolute error with respect to the exact solution and dots are pointwise residuals of the PDE at check points.

which has nontrivial solution

$$u(x) = -\beta \tanh\left(\frac{1}{2}\beta\nu^{-1}\left(x - \frac{1}{2}\right)\right), \tag{7}$$

where β satisfies the transcendental equation

$$-\frac{1}{2}\beta^2 \operatorname{sech}^2\left(\frac{1}{4}\beta\nu^{-1}\right) + \kappa\left[\alpha - \beta\tanh\left(\frac{1}{4}\beta\nu^{-1}\right)\right] = 0. \tag{8}$$

For our test, we choose $\nu = 5 \times 10^{-3}$, $\alpha = 1$, and $\kappa = 2$. Since the equation is nonlinear, we use *fsolve* in MATLAB to solve the resulting system of nonlinear algebraic equations. A good initial starting guess is needed to solve the nonlinear equation. However, one can choose the initial guess by continuation from the solution obtained with higher ν. Adaptation takes fewer than 10 iterations to converge and we end up having 396 centers with maximum residual less than 10^{-4}, as shown in Fig. 7.

In the stationary Allen–Cahn equation, given by

$$\begin{aligned} &\nu u_{xx} + u - u^3 = 0 \quad x \in (-1, 1)\\ &u(-1) = -1\\ &u(1) = 1, \end{aligned}$$

we choose $\nu = 10^{-4}$. In order to approximate the starting initial guess for *fsolve*, we continue from $\nu = 10^{-2}$ and then $\nu = 10^{-3}$. The final total number of centers at $\nu = 10^{-4}$ is 132, as shown in Fig. 8. By comparison, `bvp4c` in MATLAB at the same tolerance uses 270 nodes for the same error.

As a final test of boundary-value model problems, we consider the Poisson equation in 2D

$$\begin{cases} \Delta u = f & \text{in } \Omega,\\ u = u_o & \text{on } \partial\Omega, \end{cases} \tag{9}$$

where $f \in C^\infty(\Omega)$. We choose boundary data and f such that

$$u(x, t) = \mathrm{e}^{-10(x^2+y^2)} \tag{10}$$

is the exact solution in $[-1, 1]^2$. As error indicator in the interior, we use the area-based residual function. The numerical solution is then compared to the exact solution on a finer grid in $[-1, 1]^2$ using 4000 points. By taking $\epsilon_g = 1$, only 4 iterations are needed to achieve maximum error 4.54×10^{-5} with 260 centers.

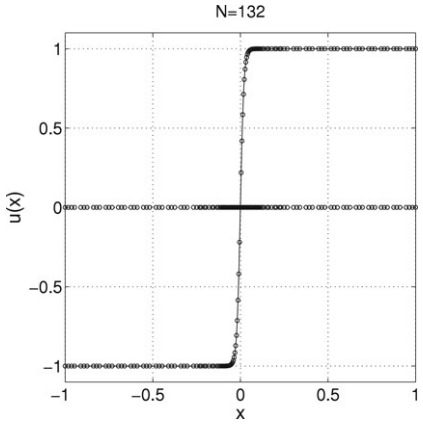


Fig. 8. Numerical solution of stationary Allen–Cahn with parameter $\nu = 10^{-4}$.

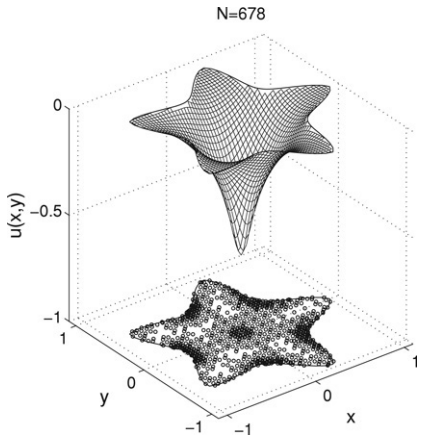


Fig. 9. Numerical solution of Poisson's equation in starfish-like domain.

For a more complicated domain, like the smooth "starfish" in Fig. 9, we use forcing term $f(x, y) = 100\mathrm{e}^{-200(x^2+y^2)}$. With $\epsilon_g = 4$, Fig. 9 shows the solution after 3 iterations. Note that we measure the Dirichlet boundary residuals as well as Laplace residuals inside the domain. Usually these residuals differ in orders of magnitude. Following [10], we calculate their relative magnitude and introduce a weighted maximum norm which often puts a larger weight on the quality of the boundary values.

The final test is the L-shaped domain with the forcing term

$$f(x, y) = -10\mathrm{e}^{-10((x-0.1)^2+(y-0.1)^2)}$$

close to the corner, as shown in Fig. 10. The iteration stops successfully with $N = 2100$. We do not regard automatic adaptation as the ideal method for a singularity due to a geometric corner. One would probably do much better with an a priori adaptation or by including appropriate singular functions in the interpolant [31].

4.3. Time-dependent problems

One way to use our adaptation in a time-dependent problem is to alternate time stepping with adaptation. In this model, an initial condition at $t = 0$ is used to adaptively select the RBF nodes and shape parameters. These parameters define a standard method-of-lines semi-discretization that can be used to advance the discrete solution up to a predetermined time $t = \tau$. Now that one has values at the RBF nodes that implicitly determine an RBF interpolant approximating the solution at $t = \tau$, we can apply the residual subsampling algorithm to the interpolant to derive a new set of RBF nodes and shape parameters. This is treated like a new initial state for further time stepping, etc. The

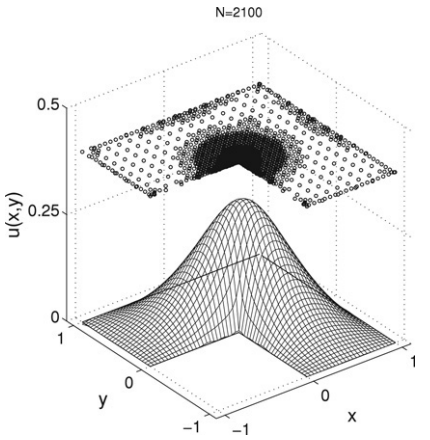


Fig. 10. Numerical solution of Poisson's equation in L-shaped domain.

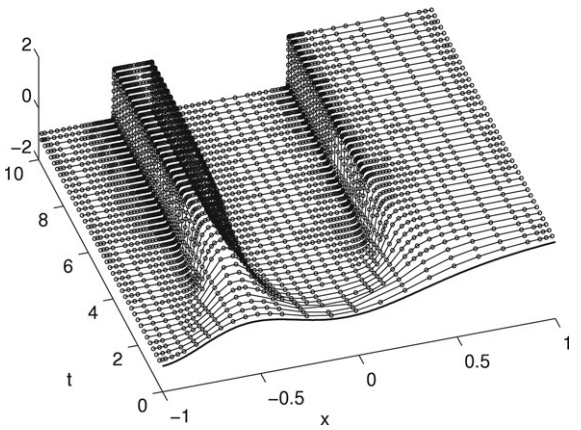


Fig. 11. Solution of the Allen–Cahn equation (11) using a model of alternating time stepping and adaptation by residual subsampling of the approximate solution. Each line in the figure represents a time at which adaptation has occurred. The nodes clearly adapt themselves to emerging steep gradients.

idea is to pick τ large enough to avoid excessive adaptation steps, while keeping it small enough that the adaptation can keep up with emerging or changing features in the solution.

Fig. 11 shows the result of this alternating model for the Allen–Cahn equation,

$$u_t - u(1-u^2) = \nu u_{xx}, \tag{11}$$

with $\nu = 10^{-6}$, $u(\pm 1, t) = \pm 1$, and $u(x,0) = 0.6x + 0.4\sin\left[\frac{\pi}{2}(x^2 - 3x - 1)\right]$. The solution is advanced in time using fourth-order Runge–Kutta with step size 0.01, and adaptation occurs every 25 time steps with thresholds $\theta_r = 200\theta_c = 10^{-3}$. The number of RBF nodes starts with 27 and gradually grows to a maximum of 178 at $t = 8.25$. As the figure shows, the nodes are able to track the developing steep gradients quite efficiently.

Our final example is the moving front problem given by the Burgers' equation

$$\begin{aligned} &\nu u_{xx} - u u_x = u_t, \quad 0 < x < 1 \\ &u(0,t) = u(1,t) = 0 \\ &u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \end{aligned}$$

This problem is known to be a common test problem for many adaptive methods [32]. The solution generates a steep front with width $O(\nu)$ moving towards $x = 1$ which also decays with time due to the boundary conditions.

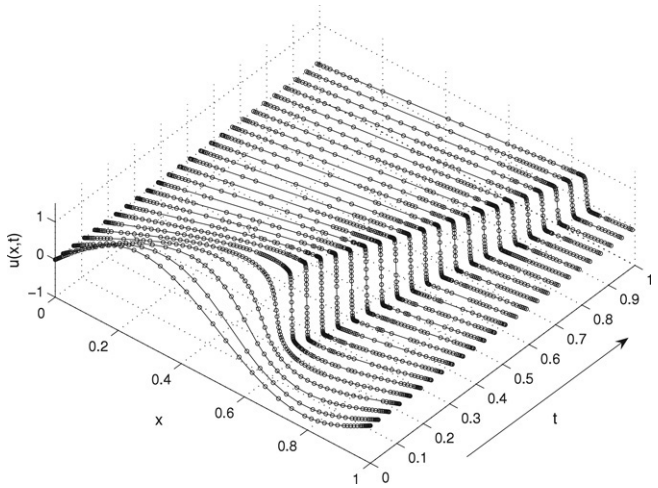


Fig. 12. Solution of the Burgers' equation using a model of alternating time stepping and adaptation by residual subsampling of the approximate solution. Each line in the figure represents a time at which adaptation has occurred.

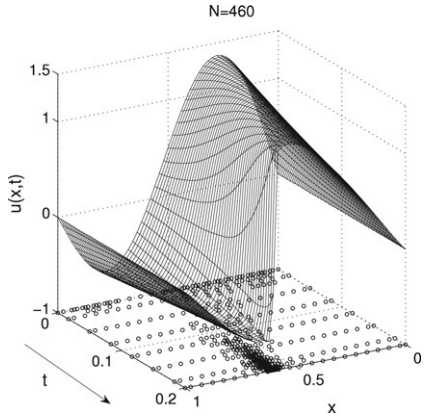


Fig. 13. Numerical solution of time-dependent Burgers' equation using direct implementation of residual subsampling method in space–time.

Our numerical experiment was run with $\nu = 10^{-3}$. The solution is advanced in time using MATLAB built-in stiff odesolver ode15s. As Fig. 12 shows, the nodes are able to track the developing moving front. As an alternative to step-and-adapt method, our numerical experiment can be run directly in space–time. In other words, we treat this space–time problem as a regular 2D case. The choice of the distance function is the Euclidean distance in space–time given by

$$r = \sqrt{(x - x_j)^2 + (t - t_j)^2}. \tag{12}$$

To start with, we compute Burgers' equation in the space–time slab $[0, 1] \times [0, 2]$. We apply boundary adaptation except at points with $t = 0.2$ and the area-based residuals are used as error indicator. We use *fsolve* in MATLAB to solve the nonlinear equations with $u(x, t) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$, where $t = [0, 0.2]$ as the surface of the very first initial guess. Even though *fsolve* has a built-in feature based on finite difference to calculate the Jacobian of the nonlinear algebraic equations, in this case we are able to use the exact Jacobian. The residuals are less than 5×10^{-4} in fewer than 10 adaptive iterations. In Fig. 13, we can see that the RBF centers are adaptively located following the path of the moving front problem.

The process of marching in time can be done in a straightforward manner. Once we found all information such as centers, shape parameters, and values at current space–time slab, we simply apply the same adaptive technique to the next slab by using the values at final previous time as a new initial condition. In this preliminary form, the application

of residual subsampling in time-dependent problems is very promising. However, the stability and robustness of the method in time-dependent cases are still under research.

References

[1] M.D. Buhmann, Radial Basis Functions: Theory and Implementations, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 12, Cambridge University Press, Cambridge, 2003.
[2] E.J. Kansa, Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates, Comput. Math. Appl. 19 (8–9) (1990) 127–145.
[3] E.J. Kansa, Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl. 19 (8–9) (1990) 147–161.
[4] C. Franke, R. Schaback, Solving partial differential equations by collocation using radial basis functions, Appl. Math. Comput. 93 (1) (1998) 73–82.
[5] E. Larsson, B. Fornberg, A numerical study of some radial basis function based solution methods for elliptic PDEs, Comput. Math. Appl. 46 (5–6) (2003) 891–902.
[6] A.H.-D. Cheng, M.A. Golberg, E.J. Kansa, G. Zammito, Exponential convergence and *h-c* multiquadric collocation method for partial differential equations, Numer. Methods Partial Differential Equations 19 (5) (2003) 571–594.
[7] M. Buhmann, N. Dyn, Spectral convergence of multiquadric interpolation, Proc. Edinb. Math. Soc. (2) 36 (2) (1993) 319–333.
[8] M. Bozzini, L. Lenarduzzi, R. Schaback, Adaptive interpolation by scaled multiquadrics, Adv. Comput. Math. 16 (4) (2002) 375–387.
[9] R. Schaback, H. Wendland, Adaptive greedy techniques for approximate solution of large RBF systems, Numer. Algorithms 24 (3) (2000) 239–254.
[10] Y.C. Hon, R. Schaback, X. Zhou, An adaptive greedy algorithm for solving large RBF collocation problems, Numer. Algorithms 32 (1) (2003) 13–25.
[11] Y.C. Hon, Multiquadric collocation method with adaptive technique for problems with boundary layer, Internat. J. Appl. Sci. Comput. 6 (3) (1999) 173–184.
[12] J. Behrens, A. Iske, M. Käser, Adaptive meshfree method of backward characteristics for nonlinear transport equations, in: Meshfree Methods for Partial Differential Equations (Bonn, 2001), in: Lect. Notes Comput. Sci. Eng., vol. 26, Springer, Berlin, 2003, pp. 21–36.
[13] J. Behrens, A. Iske, Grid-free adaptive semi-Lagrangian advection using radial basis functions, Comput. Math. Appl. 43 (3–5) (2002) 319–327.
[14] S.A. Sarra, Adaptive radial basis function methods for time dependent partial differential equations, Appl. Numer. Math. 54 (1) (2005) 79–94.
[15] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 (4) (1995) 389–396.
[16] C.A. Micchelli, Interpolation of scattered data: Distance matrices and conditionally positive definite functions, Constr. Approx. 2 (1) (1986) 11–22.
[17] J. Yoon, Spectral approximation orders of radial basis function interpolation on the Sobolev space, SIAM J. Math. Anal. 33 (4) (2001) 946–958 (Electronic).
[18] T.A. Driscoll, B. Fornberg, Interpolation in the limit of increasingly flat radial basis functions, Comput. Math. Appl. 43 (3–5) (2002) 413–422.
[19] E.J. Kansa, R.E. Carlson, Improved accuracy of multiquadric interpolation using variable shape parameters, Comput. Math. Appl. 24 (12) (1992) 99–120.
[20] R.E. Carlson, T.A. Foley, The parameter $\mathbf{R}^2$ in multiquadric interpolation, Comput. Math. Appl. 21 (9) (1991) 29–42.
[21] S. Rippa, An algorithm for selecting a good value for the parameter *c* in radial basis function interpolation, Adv. Comput. Math. 11 (2–3) (1999) 193–210.
[22] R. Schaback, Error estimates and condition numbers for radial basis function interpolation, Adv. Comput. Math. 3 (3) (1995) 251–264.
[23] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (5–6) (2004) 853–867.
[24] R.K. Beatson, J.B. Cherrie, C.T. Mouat, Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration, Adv. Comput. Math. 11 (2–3) (1999) 253–270.
[25] R.B. Platte, T.A. Driscoll, Polynomials and potential theory for Gaussian radial basis function interpolation, SIAM J. Numer. Anal. 43 (2) (2005) 750–766 (Electronic).
[26] B. Fornberg, T.A. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Comput. Math. Appl. 43 (3–5) (2002) 473–490.
[27] T.A. Driscoll, http://www.math.udel.edu/~driscoll/index.html.
[28] A.I. Fedoseyev, M.J. Friedman, E.J. Kansa, Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary, Comput. Math. Appl. 43 (3–5) (2002) 439–455.
[29] R. Franke, Scattered data interpolation: Tests of some methods, Math. Comp. 38 (157) (1982) 181–200.
[30] L.G. Reyna, M.J. Ward, On the exponentially slow motion of a viscous shock, Comm. Pure Appl. Math. 48 (2) (1995) 79–120.
[31] R.B. Platte, T.A. Driscoll, Computing eigenmodes of elliptic operators using radial basis functions, Comput. Math. Appl. 48 (3–4) (2004) 561–576.
[32] W. Huang, Y. Ren, R.D. Russell, Moving mesh methods based on moving mesh partial differential equations, J. Comput. Phys. 113 (2) (1994) 279–290.