Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Mathematics with Applications 48 (2004) 853–867
An International Journal
computers & mathematics with applications
www.elsevier.com/locate/camwa

Stable Computation of Multiquadric Interpolants for All Values of the Shape Parameter

B. FORNBERG*
University of Colorado, Department of Applied Mathematics
CB-526, Boulder, CO 80309, U.S.A.
fornberg@colorado.edu

G. WRIGHT†
University of Utah, Department of Mathematics
Salt Lake City, UT 84112, U.S.A.
wright@math.utah.edu

(Received March 2003; revised and accepted August 2003)

Abstract—Spectrally accurate interpolation and approximation of derivatives used to be practical only on highly regular grids in very simple geometries. Since radial basis function (RBF) approximations permit this even for multivariate scattered data, there has been much recent interest in practical algorithms to compute these approximations effectively.

Several types of RBFs feature a free parameter (e.g., c in the multiquadric (MQ) case $\phi(r) = \sqrt{r^2 + c^2}$). The limit of $c \to \infty$ (increasingly flat basis functions) has not received much attention because it leads to a severely ill-conditioned problem. We present here an algorithm which avoids this difficulty, and which allows numerically stable computations of MQ RBF interpolants for all parameter values. We then find that the accuracy of the resulting approximations, in some cases, becomes orders of magnitude higher than was the case within the previously available parameter range.

Our new method provides the first tool for the numerical exploration of MQ RBF interpolants in the limit of $c \to \infty$. The method is in no way specific to MQ basis functions and can—without any change—be applied to many other cases as well.

Keywords—Radial basis functions, RBF, Multiquadrics, Ill-conditioning.

1. INTRODUCTION

Linear combinations of radial basis functions (RBFs) can provide very good interpolants for multivariate data. Multiquadric (MQ) basis functions, generated by $\phi(r) = \sqrt{r^2 + c^2}$ (or in the notation used in this paper, $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$ with $\varepsilon = 1/c$), have proven to be particularly successful [1]. However, there have been three main difficulties with this approach: severe numerical ill-conditioning for a fixed N (the number of data points) and small ε, similar ill-conditioning

*This work was supported by NSF Grants DMS-9810751 (VIGRE), DMS-0073048, and a Faculty Fellowship from the University of Colorado at Boulder.
†This work was supported by an NSF VIGRE Graduate Traineeship under Grant DMS-9810751.

0898-1221/04/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
doi:10.1016/j.camwa.2003.08.010

problems for a fixed ε and large N, and high computational cost. This study shows how the first of these three problems can be resolved.

Large values of parameter ε are well known to produce very inaccurate results (approaching linear interpolation in the case of 1-D). Decreasing ε usually improves the accuracy significantly [2]. However, the direct way of computing the RBF interpolant suffers from severe ill-conditioning as ε is decreased [3]. Several numerical methods have been developed for selecting the "optimal" value of ε (e.g., [4–6]). However, because of the ill-conditioning problem, they have all been limited in the range of values that could be considered, having to resort to high-precision arithmetic, for which the cost of computing the interpolant increases to infinity as $\varepsilon \to 0$ (timing illustrations for this will be given later). In this study, we present the first algorithm which not only can compute the interpolant for the full range $\varepsilon \geq 0$, but it does so entirely without a progressive cost increase as $\varepsilon \to 0$.

In the highly special case of MQ RBF interpolation on an infinite equispaced Cartesian grid, Buhmann and Dyn [7] showed that the interpolants obtain spectral convergence for smooth functions as the grid spacing goes to zero (see, for example, [8] for the spectral convergence properties of MQ and other RBF interpolants for scattered finite data sets). Additionally, for an infinite equispaced grid, but with a fixed grid spacing, Baxter [9] showed the MQ RBF interpolant in the limit of $\varepsilon \to 0$ to cardinal data (equal to one at one data point and zero at all others) exists and goes to the multidimensional sinc function—just as the case would be for a Fourier spectral method. Limiting ($\varepsilon \to 0$) interpolants on scattered finite data sets were studied by Driscoll and Fornberg [10]. They noted that, although the limit usually exists, it can fail to do so in exceptional cases. The present numerical algorithm handles both of these situations. It also applies—without any change—to many other types of basis functions. The cases we will give computational examples for are listed in Table 1. Note that for all these cases, the limits of flat basis functions correspond to $\varepsilon \to 0$.

Table 1.

Name of RBF	Abbreviation	Definition
Multiquadrics	MQ	$\phi(r) = \sqrt{1+(\varepsilon r)^2}$
Inverse Quadratics	IQ	$\phi(r) = \dfrac{1}{1+(\varepsilon r)^2}$
Gaussians	GA	$\phi(r) = e^{-(\varepsilon r)^2}$

The main idea of the present method is to consider the RBF interpolant at a fixed $\underline{x}$

$$s(\underline{x},\varepsilon) = \sum_{j=1}^{N} \lambda_j \phi\left(\|\underline{x} - \underline{x}_j\|\right), \tag{1}$$

(where $\|\cdot\|$ is the two-norm) not only for real values of ε, but as an analytic function of a complex variable ε. Although not explicitly marked, λ_j and ϕ are now functions of ε. In the sections that follow, we demonstrate that in a relatively large area around $\varepsilon = 0$, $s(\underline{x},\varepsilon)$ will at worst have some isolated poles. It can, therefore, be written as

$$s(\underline{x},\varepsilon) = (\text{rational function in } \varepsilon) + (\text{power series in } \varepsilon). \tag{2}$$

The present algorithm numerically determines (in a stable way) the coefficients to the rational function and the power series. This allows us to use (2) for computing the RBF interpolant effectively numerically right down to $\varepsilon = 0$. The importance of this entirely new capability is expected to be as a tool to investigate properties of RBF approximations and not, at the present time, to interpolate any large experimental data sets.

Although not pursued here, there are a number of important and unresolved issues relating to the limit of the RBF interpolant as $\varepsilon \to 0$, for which the present algorithm will now allow

numerical explorations. For example, it was shown by Driscoll and Fornberg [10] that the limiting interpolant in 1-D is simply the Lagrange interpolating polynomial. This, of course, forms the foundation for finite-difference and pseudospectral methods. The equivalent limit ($\varepsilon \to 0$) can now be studied for scattered data in higher dimensions. This is a situation where, in general, there does not exist any unique lowest-degree interpolating polynomial and, consequently, spectral limits have not received much attention.

The rest of this paper is organized as follows. Section 2 introduces a test example which we will use to describe the new method. In Section 3, we illustrate the structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane (the distribution of poles etc.). Section 4 describes the steps in the numerical method, which then are applied to our test problem in Section 5. Section 6 contains additional numerical examples and comments. We vary the number of data points and also give numerical results for some other choices of RBFs. One of the examples we present there features a situation where $\varepsilon \to 0$ leads to divergence. We finish by giving a few concluding remarks in Section 7.

2. FIRST TEST PROBLEM

Figure 1 shows 41 data points $\underline{x}_j$ randomly scattered over the unit disk (in the $\underline{x}$-plane where $\underline{x}$ is a two-vector with components x_1, x_2). We let our data at these points be defined by the function

$$f(\underline{x}) = f(x_1, x_2) = \frac{59}{67 + (x_1 + 1/7)^2 + (x_2 - 1/11)^2}. \tag{3}$$

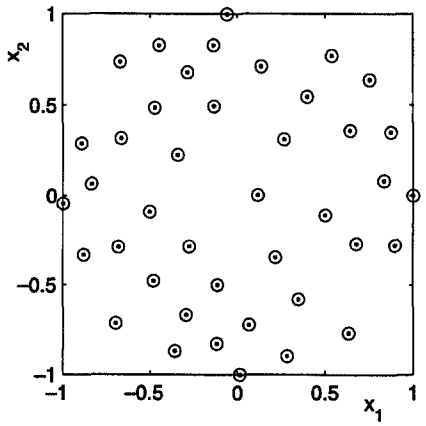


Figure 1. Distribution of 41 data points for use in the first test problem.

The task is to compute the MQ RBF interpolant (i.e., (1) with $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$) at some location $\underline{x}$ inside the unit disk. We denote the data by $y_j = f(\underline{x}_j)$, $j = 1, 2, \ldots, 41$. The immediate way to perform the RBF interpolation would be to first obtain the expansion coefficients λ_j by solving

$$[A(\varepsilon)] \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{41} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{41} \end{bmatrix}, \tag{4}$$

where the elements of $A(\varepsilon)$ are $a_{j,k} = \phi(\|\underline{x}_j - \underline{x}_k\|)$. The RBF interpolant, evaluated at $\underline{x}$, is then written as

$$s(\underline{x}, \varepsilon) = \sum_{j=1}^{41} \lambda_j \phi\left(\|\underline{x} - \underline{x}_j\|\right), \tag{5}$$

or equivalently

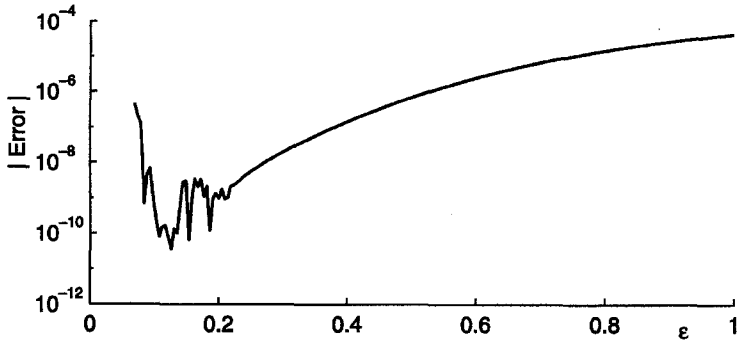


Figure 2. The error (in magnitude) as a function of ε in the interpolant $s(\underline{x}, \varepsilon)$ of (3) when $s(\underline{x}, \varepsilon)$ is computed directly using (6). We have chosen $\underline{x} = (0.3, -0.2)$.

$$s(\underline{x}, \varepsilon) = [B(\varepsilon)]\,[A(\varepsilon)]^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_{41} \end{bmatrix}, \tag{6}$$

where the elements of $B(\varepsilon)$ are $b_j = \phi(\|\underline{x} - \underline{x}_j\|)$.

Figure 2 shows the magnitude of the error $s(\underline{x}, \varepsilon) - f(\underline{x})$, where $\underline{x} = (0.3, -0.2)$ as a function of ε when computed directly via (4) and (5). This computation clearly loses its accuracy (in 64-bit floating point precision) when ε falls below approximately 0.2. The drop in error as ε approaches 0.2 (from above) suggests that computations for lower values of ε could be very accurate if the numerical instability could be overcome. The reason the onset of ill-conditioning occurs so far from $\varepsilon = 0$ is that the matrix $A(\varepsilon)$ approaches singularity very rapidly as ε decreases. Using Rouché's theorem, we can find that in this test case $\det(A(\varepsilon)) = \alpha \cdot \varepsilon^{416} + O(\varepsilon^{418})$ (where the coefficient α is nonzero). Regardless of this rapid approach to singularity, we usually find that $s(\underline{x}, \varepsilon)$ exists and is bounded as $\varepsilon \to 0$. This means an extreme amount of numerical cancellation occurs for small ε when evaluating $s(\underline{x}, \varepsilon)$.

In the notation of (6), our task is to then determine the row vector

$$[C(\varepsilon)] = [B(\varepsilon)]\,[A(\varepsilon)]^{-1}, \tag{7}$$

for all $\varepsilon \geq 0$. To do this, we need an algorithm which bypasses the extremely ill-conditioned direct formation of $A(\varepsilon)^{-1}$ and computation of the product $B(\varepsilon) \cdot A(\varepsilon)^{-1}$ for any values of ε less than approximately 0.3. The algorithm we present in Section 4 does this by directly computing $C(\varepsilon)$ around some circle in the complex ε-plane where $A(\varepsilon)$ is well conditioned. This will allow us to determine the coefficients in (2), and therefore, determine $s(\underline{x}, \varepsilon)$ for small ε-values.

Note that in practice, we often want to evaluate the interpolant at several points. This is most easily done by letting $B(\varepsilon)$ (and thus, $C(\varepsilon)$) in (7) contain several rows—one for each of the points.

3. TEST PROBLEM VIEWED IN A COMPLEX ε-PLANE

Figure 3 shows the $\log_{10}$ of the condition number of $A(\varepsilon)$ when ε is no longer confined to the real axis. We see that the ill-conditioning is equally severe in all directions as ε approaches zero in the complex plane. Furthermore, we note a number of sharp spikes. These correspond to complex ε-values for which $A(\varepsilon)$ is singular (apart from $\varepsilon = 0$, none of these can occur on the real axis according to the nonsingularity result by Micchelli [11]).

As stated in the Introduction, in a large area around $\varepsilon = 0$, $s(\underline{x}, \varepsilon)$ is a meromorphic function of ε. This can be shown by first noting that (7) can be rewritten as

$$C(\varepsilon) = \frac{1}{\det(A(\varepsilon))}\,[B(\varepsilon)]\,[\mathrm{adj}(A(\varepsilon))],$$

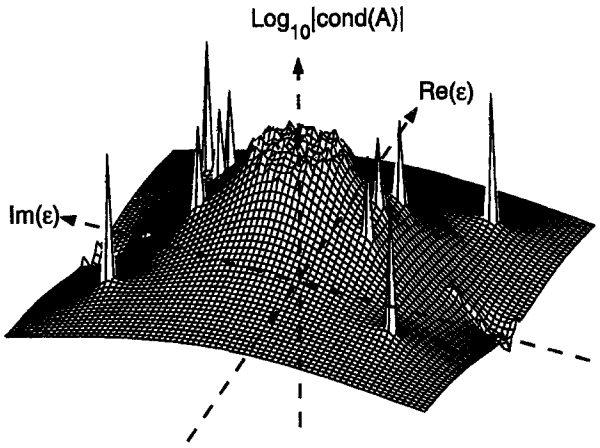


Figure 3. Logarithm (base 10) of the condition number for $A(\varepsilon)$ as a function of the complex variable ε. The domain of the plot is a square with sides of length $2 \cdot 0.75$ centered at the origin and the range of the plot varies from 0 to 10^{20}. Note near $\varepsilon = 0$ the $\log_{10}$ of the condition number of $A(\varepsilon)$ goes to infinity. However, due to numerical rounding, no values greater than 10^{20} were recorded.

where $\mathrm{adj}(A(\varepsilon))$ is the adjoint matrix of $A(\varepsilon)$. Now, letting $\Gamma_{j,k}(\varepsilon)$ be the cofactors of $A(\varepsilon)$, we have that $\Gamma_{j,k}(\varepsilon) = \Gamma_{k,j}(\varepsilon)$ for $j, k = 1, \ldots, N$ since $A(\varepsilon)$ is symmetric. Thus, expanding $\det(A(\varepsilon))$ also in cofactors, gives the following result for the j^{th} entry of $C(\varepsilon)$

$$C_j(\varepsilon) = \frac{\sum_{k=1}^{N} \phi\left(\|\underline{x} - \underline{x}_k\|\right) \Gamma_{k,j}(\varepsilon)}{\sum_{k=1}^{N} \phi\left(\|\underline{x}_j - \underline{x}_k\|\right) \Gamma_{k,j}(\varepsilon)}. \tag{8}$$

The numerator and denominator of (8) are analytic everywhere apart from the trivial branch point singularities of $\phi(r)$ on the imaginary axis. Thus, at every point apart from these trivial singularities, the numerator and denominator have a convergent Taylor expansion in some region. None of the trivial singularities can occur at $\varepsilon = 0$ since this would require $r = \infty$. Hence, there can only be a pole at $\varepsilon = 0$ if the leading power of ε in the denominator is greater than the leading power in the numerator. Remarkably, the leading powers are usually equal, making $\varepsilon = 0$ a removable singularity (in Section 6, we explore an example where this is not the case; a more extensive study on this phenomenon can be found in [12]). Apart from $\varepsilon = 0$ and the trivial singularities, the only singularity that can arise in $C_j(\varepsilon)$ is when $A(\varepsilon)$ is singular. Due to the analytic character of the numerator and denominator, this type of singularity can only be a pole (thus, justifying the analytic form stated (2)).

The structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane is shown in Figure 4. The lined area marks where the ill-conditioning is too severe for direct computation of $s(\underline{x}, \varepsilon)$ in 64-bit floating-point. The solid circles mark simple poles and the $\times$'s mark the trivial branch point singularities. The dashed line in Figure 4 indicates a possible contour (a circle) we can use in our method. Everywhere along such a circle, $s(\underline{x}, \varepsilon)$ can be evaluated directly with no particular ill-conditioning problems. Had there been no poles inside the circle, plain averaging of the $s(\underline{x}, \varepsilon)$-values around the circle would have given us $s(\underline{x}, 0)$.

It should be pointed out that if we increase the number of data points N too much (e.g., $N > 100$ in this example) the ill-conditioning region in Figure 4 will grow so that it contains some of the branch point singularities (starting at $\varepsilon = 0.5i$), forcing us to choose a circle that falls within this ill-conditioned region. However, we can still find $s(\underline{x}, \varepsilon)$ everywhere inside our circle for no worse conditioning than at ε just below 0.5.

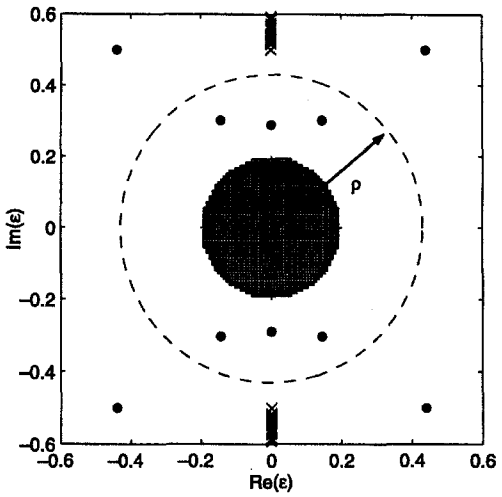


Figure 4. Structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane. The approximate area with ill-conditioning is marked with a line pattern; poles are marked with solid circles and branch points with ×'s.

To complete the description of our algorithm, we next discuss how to

- detect and compensate for the poles located inside our circle (if any), and
- compute $s(\underline{x}, \varepsilon)$ at any ε-point inside the circle (and not just at its center)

based only on ε-values around the circle.

4. NUMERICAL METHOD

We first evaluate $s(\underline{x}, \varepsilon)$ at equidistant locations around the circle of radius ρ that was shown in Figure 4, and then take the (inverse) fast Fourier transform (FFT) of these values. This produces the vector shown in Table 2 (here ordered as is conventional for the output of an FFT routine). From this (with $\varepsilon = \rho e^{i\theta}$), we have essentially obtained the Laurent expansion coefficients for $s(\underline{x}, \varepsilon)$. We can, thus, write

$$s(\underline{x}, \varepsilon) = \cdots + d_{-3}\varepsilon^{-3} + d_{-2}\varepsilon^{-2} + d_{-1}\varepsilon^{-1} + d_0 + d_1\varepsilon^1 + d_2\varepsilon^2 + d_3\varepsilon^3 + \cdots . \tag{9}$$

This expansion is convergent within some strip around the periphery of the circle. If there are no poles inside the circle all the coefficients in (9) with negative indices vanish, giving us the Taylor part of the expansion

$$s(\underline{x}, \varepsilon) = d_0 + d_1\varepsilon^1 + d_2\varepsilon^2 + d_3\varepsilon^3 + \cdots . \tag{10}$$

We can then use this to evaluate $s(\underline{x}, \varepsilon)$ numerically for any value of ε inside the circle.

The presence of any negative powers in (9) indicates that $s(\underline{x}, \varepsilon)$ has poles inside the circle. To account for the poles so that we can evaluate $s(\underline{x}, \varepsilon)$ for any value of ε inside the circle, we recast the terms with negative indices into Padé rational form. This is accomplished by first using the FFT data to form

$$q(\eta) = d_{-1}\eta + d_{-2}\eta^2 + d_{-3}\eta^3 + \cdots . \tag{11}$$

Next, we expand $q(\eta)$ in Padé rational form (see, for example, [13]), and then set

$$r(\varepsilon) = q\left(\frac{1}{\varepsilon}\right).$$

Table 2.

d_0	ρd_1	$\rho^2 d_2$	$\rho^3 d_3$	$\cdots$		$\cdots$	$\rho^{-3} d_{-3}$	$\rho^{-2} d_{-2}$	$\rho^{-1} d_{-1}$

Since $s(\underline{x}, \varepsilon)$ can only possess a finite number of poles inside the circle, the function $r(\varepsilon)$ together with (10) will entirely describe $s(\underline{x}, \varepsilon)$ in the form previously stated in (2)

$$s(\underline{x}, \varepsilon) = \{r(\varepsilon)\} + \left\{d_0 + d_1\varepsilon + d_2\varepsilon^2 + \cdots\right\}.$$

This expression can be numerically evaluated to give us $s(\underline{x}, \varepsilon)$ for any value of ε inside the circle.

An automated computer code needs to monitor several consequences of the fact that we are working with finite and not infinite expansions. These are as follows.

- $s(\underline{x}, \varepsilon)$ must be sampled densely enough so that the coefficients for the high negative and positive powers of ε returned from the FFT are small.
- When turning (11) into Padé rational form, we must choose the degrees of the numerator and denominator (which can be chosen to be equal) so that they match or exceed the total number of poles within our circle. (Converting the Padé expansion back to Laurent form and comparing coefficients offers an easy and accurate test that the degrees were chosen sufficiently high.)
- The circular path must be chosen so that it is inside the closest branch point on the imaginary axis (equal to i/D where D is the maximum distance between points), but still outside the area where direct evaluation of $s(\underline{x}, \varepsilon)$ via (6) is ill-conditioned.
- The circular path must not run very close to any of the poles.

The computations required of the method may appear to be specific to each evaluation point $\underline{x}$ that is used. However, it is possible to recycle some of the computational work needed for evaluating $s(\underline{x}, \varepsilon)$ at one $\underline{x}$ into evaluating $s(\underline{x}, \varepsilon)$ at new values of $\underline{x}$. For example, from (8), we know that the nontrivial pole locations of $s(\underline{x}, \varepsilon)$ are entirely determined by the data points $\underline{x}_j$. Thus, once $r(\varepsilon)$ has been determined for a given $\underline{x}$, we can reuse its denominator for evaluating $s(\underline{x}, \varepsilon)$ at other values of $\underline{x}$. This allows the interpolant to be evaluated much more cheaply at new values of $\underline{x}$.

It could conceivably happen that a zero in the denominator of (8) gets canceled by a simultaneous zero in the numerator for one evaluation point but not another. We have, however, only observed this phenomenon in very rare situations (apart from the trivial case when the evaluation point coalesces with one of the data points). Nevertheless, an automated code needs to handle this situation appropriately.

5. NUMERICAL METHOD APPLIED TO THE TEST PROBLEM

We choose for example $M = 128$ points around a circle of radius $\rho = 0.42$ (as shown in Figure 4). This requires just $M/4 + 1 = 33$ evaluations of $s(\underline{x}, \varepsilon)$ due to the symmetry between the four quadrants. We again take $\underline{x} = (0.3, -0.2)$. Following the inverse FFT (and after "scaling away" ρ), we cast the terms with negative indices to Padé rational form to obtain

$$r(\varepsilon) = \frac{-3.3297 \cdot 10^{-11} - 5.9685 \cdot 10^{-10}\varepsilon^2 - 1.8415 \cdot 10^{-9}\varepsilon^4 + 0 \cdot \varepsilon^6}{1.0541 \cdot 10^{-3} + 2.4440 \cdot 10^{-2}\varepsilon^2 + 2.2506 \cdot 10^{-1}\varepsilon^4 + \varepsilon^6}. \tag{12}$$

(The highest degree term in the numerator is zero because expansion (11) contains no constant term.) Combining (12) with the Taylor series approximation, we compute, for example, $s(\underline{x}, \varepsilon)$ at $\varepsilon = 0.1$

$$s(\underline{x}, 0.1) \approx \{r(0.1)\} + \left\{\sum_{k=0}^{32} d_{2k}(0.1)^{2k}\right\} \approx 0.87692244095761.$$

Note that the only terms present in the Taylor and Padé approximations are even, due to the four-fold symmetry of $s(\underline{x}, \varepsilon)$ in the complex ε-plane.

Table 3 compares the error in $s(\underline{x}, \varepsilon)$ when computed in standard 64-bit floating point with the direct method (6) and when computed with the present algorithm. The comparisons were made

Table 3. Comparison of the error in $s(\underline{x}, \varepsilon)$ when computed using the direct method and the Contour-Padé algorithm. For these comparisons, we have chosen $\underline{x} = (0.3, -0.2)$.

Magnitude of the error in $s(\underline{x}, \varepsilon)$ when computed using the direct method					
$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
∞	$3.9 \cdot 10^{-3}$	$1.0 \cdot 10^{-6}$	$4.9 \cdot 10^{-10}$	$1.4 \cdot 10^{-9}$	$4.6 \cdot 10^{-11}$

Magnitude of the error in $s(\underline{x}, \varepsilon)$ when computed using the Contour-Padé algorithm						
$\frac{M}{4}+1$	$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
33	...	$1.1 \cdot 10^{-13}$	$1.0 \cdot 10^{-13}$	$8.4 \cdot 10^{-14}$	$7.1 \cdot 10^{-14}$	$1.1 \cdot 10^{-13}$
65	...	$1.3 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.2 \cdot 10^{-13}$
129	...	$2.1 \cdot 10^{-13}$	$2.0 \cdot 10^{-13}$	$1.8 \cdot 10^{-13}$	$1.6 \cdot 10^{-13}$	$5.6 \cdot 10^{-14}$

Magnitude of the error $s(\underline{x}, \varepsilon) - f(\underline{x})$ when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm						
$\frac{M}{4}+1$	$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
33	$5.3 \cdot 10^{-11}$	$5.2 \cdot 10^{-11}$	$2.5 \cdot 10^{-11}$	$2.3 \cdot 10^{-12}$	$2.5 \cdot 10^{-13}$	$5.5 \cdot 10^{-9}$

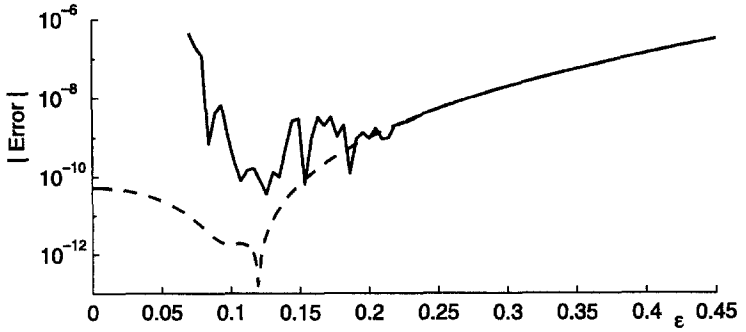


Figure 5. The error (in magnitude) as a function of ε in the interpolant $s(\underline{x}, \varepsilon)$ of (3). The solid line shows the error when $s(\underline{x}, \varepsilon)$ is computed using (6) and the dashed line shows the error when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm presented in Section 4. Again, we have chosen $\underline{x} = (0.3, -0.2)$.

with $s(\underline{x}, \varepsilon)$ computed via (6) with high-precision arithmetic, using 60 digits of accuracy. The last part of the table compares the error in the approximation of (3), when $s(\underline{x}, \varepsilon)$ is computed using the present algorithm.

Figure 5 graphically compares the results of the Contour-Padé algorithm using $M/4+1 = 33$ to those using the direct method (6). Like Table 3, the figure clearly shows that the Contour-Padé algorithm allows the RBF interpolant to be computed in a stable manner for the full range of ε. (The increased error in the results of the Contour-Padé algorithm as ε falls below 0.12 is not due to any loss in computational accuracy; it is a genuine feature of the RBF interpolant, and will be discussed in a separate study.)

Next, we compare the computational effort required to compute $s(\underline{x}, \varepsilon)$ using the direct method (6) and the Contour-Padé algorithm. To obtain the same level of accuracy (around 12 digits) with the direct method as the present algorithm provides requires the use of high-precision arithmetic. Table 4 summarizes the time required for computing the interpolant via the direct method using MATLAB's variable-precision arithmetic (VPA) package. Note that in this approach, changing ε will necessitate an entirely new calculation. All computations were done on a 500 MHz Pentium III processor.

Table 4.

ε	Digits needed	Time for finding λ_j	Time for evaluating $s(\underline{x}, \varepsilon)$ at each $\underline{x}$
10^{-2}	42	172.5 sec.	1.92 sec.
10^{-4}	74	336.3 sec.	2.09 sec.
10^{-6}	106	574.6 sec.	2.31 sec.
10^{-8}	138	877.1 sec.	2.47 sec.

Table 5.

Portion of the Algorithm	Time
Finding the expansion coefficients around the ε circle and the poles for the Padé rational form	0.397 sec.
Evaluating $s(\underline{x}, \varepsilon)$ at a new $\underline{x}$ value	0.0412 sec.
Evaluating $s(\underline{x}, \varepsilon)$ at a new ε value	0.0022 sec.

With the Contour-Padé algorithm, the problem can be done entirely in standard 64-bit floating point. A summary of the time required to compute the various portions of the algorithm using MATLAB's standard floating point is shown in Table 5. Note that these times hold true regardless of the value of ε.

6. SOME ADDITIONAL EXAMPLES AND COMMENTS

Looking back at the description of the Contour-Padé algorithm, we see that it only relied on computing $s(\underline{x}, \varepsilon)$ around a contour and was in no way specific to the MQ RBF. In the first part of this section, we present some additional results of the algorithm and make some comments not only for the MQ RBF, but also for the IQ and GA RBFs.

We consider RBF approximations of (3) sampled at the 62 data points $\underline{x}_j$ shown in Figure 6. To get a better idea of how the error behaves over the whole region (i.e., the unit disk), we compute the root-mean-square (RMS) error of the approximations over a dense set of points covering the region. In all cases, we use the Contour-Padé algorithm with $M = 512$ points around the contour.

Figure 7a shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the MQ RBF (we recall that the pole locations are entirely independent of $\underline{x}$). Unlike the example from Section 2, which resulted in six poles for $s(\underline{x}, \varepsilon)$, we see from the figure that the present example only results in two poles within the contour (indicated by the dashed line). Figure 8a compares the resulting

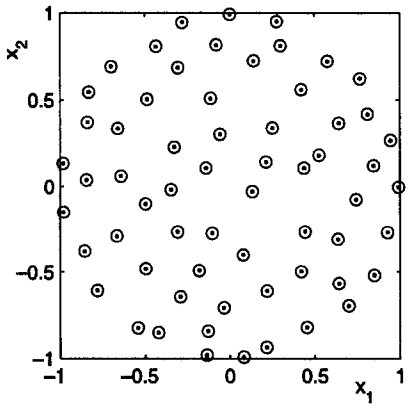


Figure 6. Distribution of 62 data points for use in the example from Section 6.

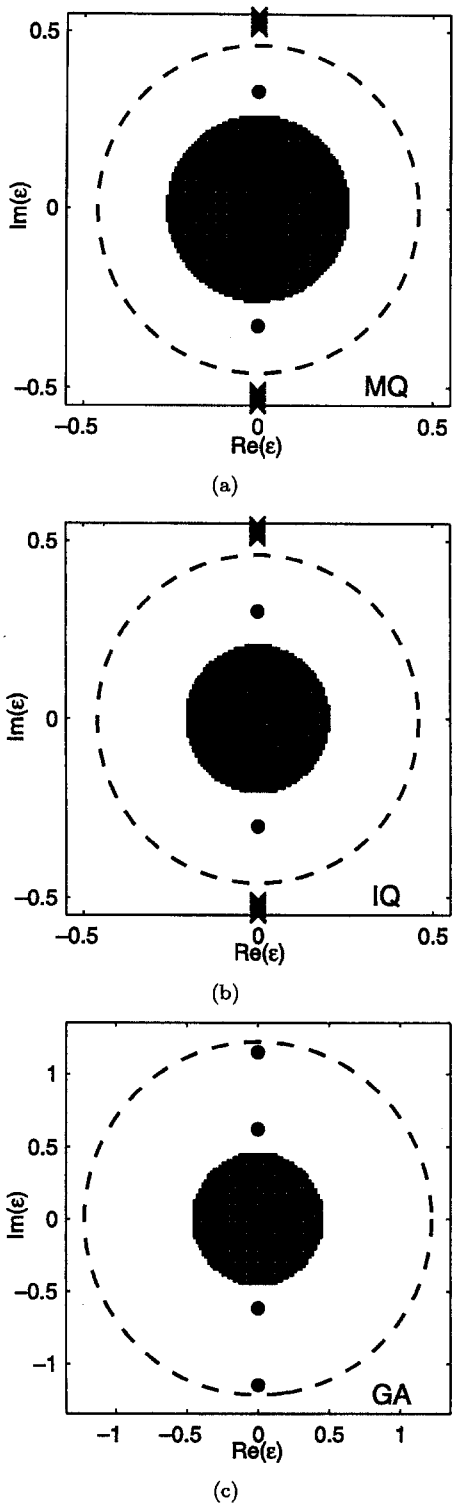
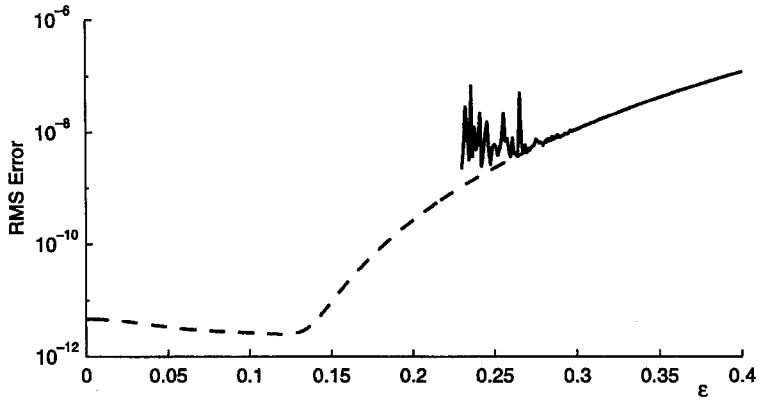
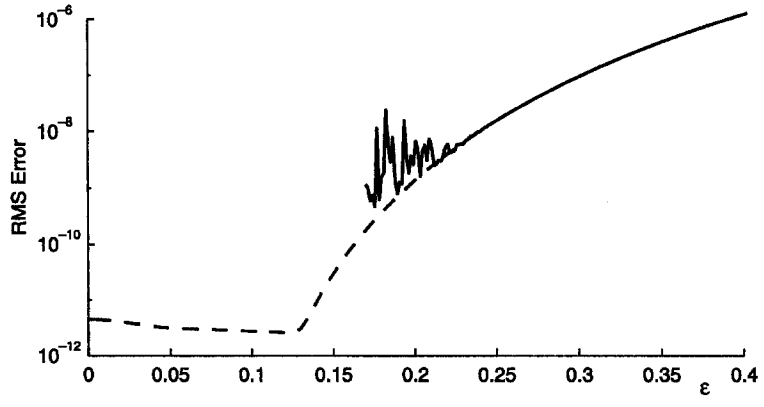


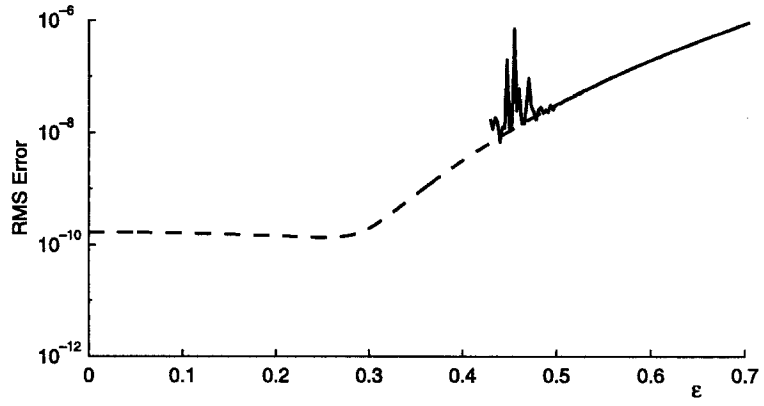
Figure 7. The structures of $s(\underline{x}, \varepsilon)$ in complex ε-plane for the 62 data points shown in Figure 6 in the case of (a) MQ RBF, (b) IQ RBF, and (c) GA RBF (note the different scale). The approximate region of ill-conditioning is marked with a line pattern, the poles are marked with solid circles, and singularities due to the basis functions themselves (i.e., branch points for the MQ RBF and poles for the IQ RBF) are marked with ×'s. The dashed lines indicate the contours that were used for computing $s(\underline{x}, \varepsilon)$ for each of the three cases.



(a) MQ RBF.



(b) IQ RBF.



(c) GA RBF.

Figure 8. The RMS error in the (a) MQ, (b) IQ, and (c) GA RBF approximations $s(\underline{x}, \varepsilon)$ of (3). The solid line shows the error when $s(\underline{x}, \varepsilon)$ is computed using (6) and the dashed line shows the error when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm. Note the different scale for the GA RBF results.

RMS error as a function of ε when the MQ RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm. The figure shows that the direct computation becomes unstable when ε falls below approximately 0.28. Most algorithms for selecting the optimal value of ε (based on RMS errors) would thus be limited from below by this value. However, the Contour-Padé algorithm allows us to compute the approximation accurately for every value of ε. As the figure shows, the true optimal value of ε is approximately 0.119. The

RMS error in the approximation at this value is approximately $2.5 \cdot 10^{-12}$, whereas the RMS error in the approximation at $\varepsilon = 0.28$ is approximately $6.0 \cdot 10^{-9}$.

Figure 7b shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the IQ RBF. We notice a couple of general differences between the structures based on the IQ RBF and MQ RBF. First, the IQ RBF leads to a slightly better conditioned linear system to solve. Thus, the approximate area of ill-conditioning is smaller. Second, the IQ basis function contains a pole, rather than a branch point, when $\varepsilon = \pm i/r$. Thus, for evaluation on the unit disk, there will be trivial poles (of unknown strengths) on the imaginary ε-axis that can never get closer to the origin than $\pm i/2$. For our 62 point distribution and for an evaluation point $\underline{x}$ that does not correspond to any of the data points, there could be up to $2 \cdot 62 = 124$ trivial poles on the imaginary axis. If we combine these with the nontrivial poles that arise from singularities in the $A(\varepsilon)$ matrix, this will be too many for the Contour-Padé algorithm to "pick up". So, as in the MQ case, the choice of our contour is limited by $1/D$, where D is the maximum distance between the points (e.g., $1/D = 1/2$ for evaluation on the unit disk). One common feature we have observed in the structures of $s(\underline{x}, \varepsilon)$ for the IQ and MQ cases is that the location of the poles due to singularities of the $A(\varepsilon)$ matrix are usually in similar locations (cf. the solid circles in Figures 7a and 7b).

Figure 8b compares the resulting RMS error as a function of ε when the IQ RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm. Again, we see that the direct computation becomes unstable when ε falls below approximately 0.21. This is well above the optimal value of approximately 0.122. Using the Contour-Padé algorithm, we find that the RMS error in the approximation at this value of ε is approximately $2.5 \cdot 10^{-12}$, whereas the RMS error at $\varepsilon = 0.21$ is approximately $2.5 \cdot 10^{-9}$.

Figure 7c shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the GA RBF. It differs significantly from the structures based on the IQ and MQ RBFs. The first major difference is that the GA RBF possesses no singularities in the finite complex ε-plane (it has an essential singular point at $\varepsilon = \infty$). Thus, the contour we choose is not limited by the maximum distance between the points. However, the GA RBF grows as ε moves farther away from the real axis. Thus, the contour we choose for evaluating $s(\underline{x}, \varepsilon)$ is limited by the ill-conditioning that arises for large imaginary values of ε. This limiting factor has significantly less impact than the "maximum distance" limiting factor for the MQ and IQ RBFs, and makes the Contour-Padé algorithm based on GA RBF able to handle larger data sets (for example, it can easily handle approximations based on 100 data points in the unit disk when the computations are done in standard 64-bit floating point). Indeed, Figure 7c shows that the contour we used for the GA RBF approximation is much farther away from the ill-conditioned region around $\varepsilon = 0$, than the corresponding contours for the MQ and IQ approximations. The second difference for the GA RBF is that it leads to a linear system that approaches ill-conditioning faster as ε approaches zero [3]. The final difference we note (from also looking at additional examples) is that the pole structure of $s(\underline{x}, \varepsilon)$ based on the GA RBF often differs quite significantly from those based on the MQ and IQ RBFs.

Figure 8c compares the resulting RMS error as a function of ε when the GA RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm. The figure shows that instability in the direct method arises when ε falls below 0.48. Again, this is well above the optimal value of $\varepsilon = 0.253$. The Contour-Padé algorithm produces an RMS error of approximately $1.4 \cdot 10^{-10}$ at this value, whereas the RMS error at $\varepsilon = 0.48$ is approximately $2.0 \cdot 10^{-8}$.

We next explore a case where the limit of $s(\underline{x}, \varepsilon)$ as $\varepsilon \to 0$ fails to exist. As was reported in [10], the 5×5 equispaced Cartesian grid over $[0,1] \times [0,1]$ leads to divergence in $s(\underline{x}, \varepsilon)$ of the type $O(\varepsilon^{-2})$. To see how the Contour-Padé algorithm handles this situation, we consider the 5×5 grid as our data points $\underline{x}_j$ and compute the MQ RBF approximation to (3) (although the choice of data values is irrelevant to the issue of convergence or divergence; as we know from (6), this depends only on the properties of the matrix $C(\varepsilon) = B(\varepsilon) \cdot A(\varepsilon)^{-1}$). Figure 9 shows a log $-$ log plot of RMS error where the MQ RBF approximation has been evaluated on a much denser grid over $[0,1] \times [0,1]$. In agreement with the high-precision calculations reported in [10], we again see

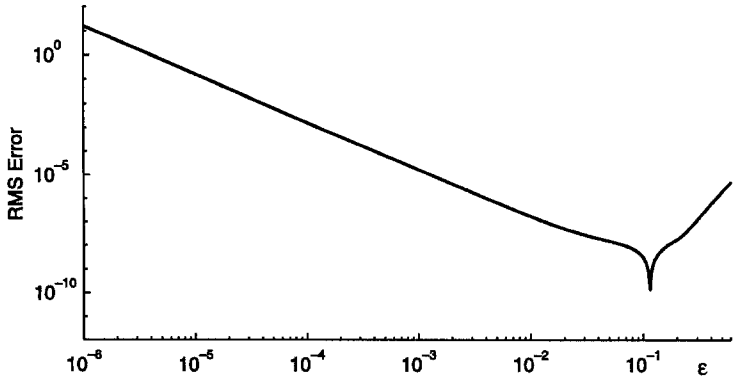


Figure 9. The RMS error in the MQ RBF approximation $s(\underline{x}, \varepsilon)$ of (3) for the case of a 5×5 equispaced Cartesian grid over $[0,1] \times [0,1]$.

a slow growth towards infinity for the interpolant. The reason is that this time there is a double pole right at the origin of the ε-plane (i.e., $\varepsilon = 0$ is not, in this case, a removable singularity). The Contour-Padé algorithm automatically handles this situation correctly, as Figure 9 shows.

To get a better understanding of how the interpolant behaves for this example, we use the algorithm to compute all functions $d_k(\underline{x})$ in the small ε-expansion

$$s(\underline{x}, \varepsilon) = d_{-2}(\underline{x})\varepsilon^{-2} + d_0(\underline{x}) + d_2(\underline{x})\varepsilon^2 + d_4(\underline{x})\varepsilon^4 + \cdots . \tag{13}$$

Figure 10 displays the first six $d_k(\underline{x})$-functions over the unit square. Note the small vertical scale on the figure for the $d_{-2}(\underline{x})$ function. This is consistent with the fact that divergence occurs only for small values of ε (cf. Figure 9). Each surface in Figure 10 shows markers (solid circles) at the 25 data points. Function $d_0(\underline{x})$ exactly matches the input function values at those points (and the other functions are exactly zero there). It also gives very accurate approximation to the actual function; the RMS error is $1.27 \cdot 10^{-8}$.

We omit the results for the IQ and GA RBF interpolants for this example, but note that the IQ also leads to divergence in $s(\underline{x}, \varepsilon)$ of the type $O(\varepsilon^{-2})$ (as reported in [10]), whereas the GA RBF actually leads to convergence.

We conclude this section with some additional comments about other techniques we tried related to computing the interpolant for small values of ε.

It is often useful (and sometimes necessary) to augment the RBF interpolant (1) with low-order polynomial terms (see, for example, [14]). The addition of these polynomial terms gives the RBF interpolation matrix (a slightly modified version of the $A(\varepsilon)$ matrix found in (4)) certain desirable properties, e.g., (conditional) positive or negative definiteness [11]. The Contour-Padé algorithm can—without any change—be used to compute the RBF interpolant also with the inclusion of these polynomial terms. We have found, however, that the behavior of the interpolant is not significantly affected by such variations. For example, we found that the pole structure of $s(\underline{x}, \varepsilon)$ is not noticeably affected, and there is no significant gain in accuracy at the "optimal" ε value (however, for larger values of ε, there can be some gains).

Since the RBF interpolant can usually be described for small values of ε by (13) but without the ε^{-2} term, one might consider using Richardson/Romberg extrapolation at larger values of ε to obtain the interpolant for $\varepsilon = 0$. However, this idea is not practical. Such extrapolation is only effective if the first few expansion terms strongly dominate the later ones. This would only be true if we are well inside the expansion's radius of convergence. As Figures 4 and 8 indicate, this would typically require computations at ε values that are too small for acceptable conditioning.

7. CONCLUDING REMARKS

The shape parameter ε in RBF interpolation plays a significant role in the accuracy of the interpolant. The highest accuracy is often found for values of ε that make the direct method of

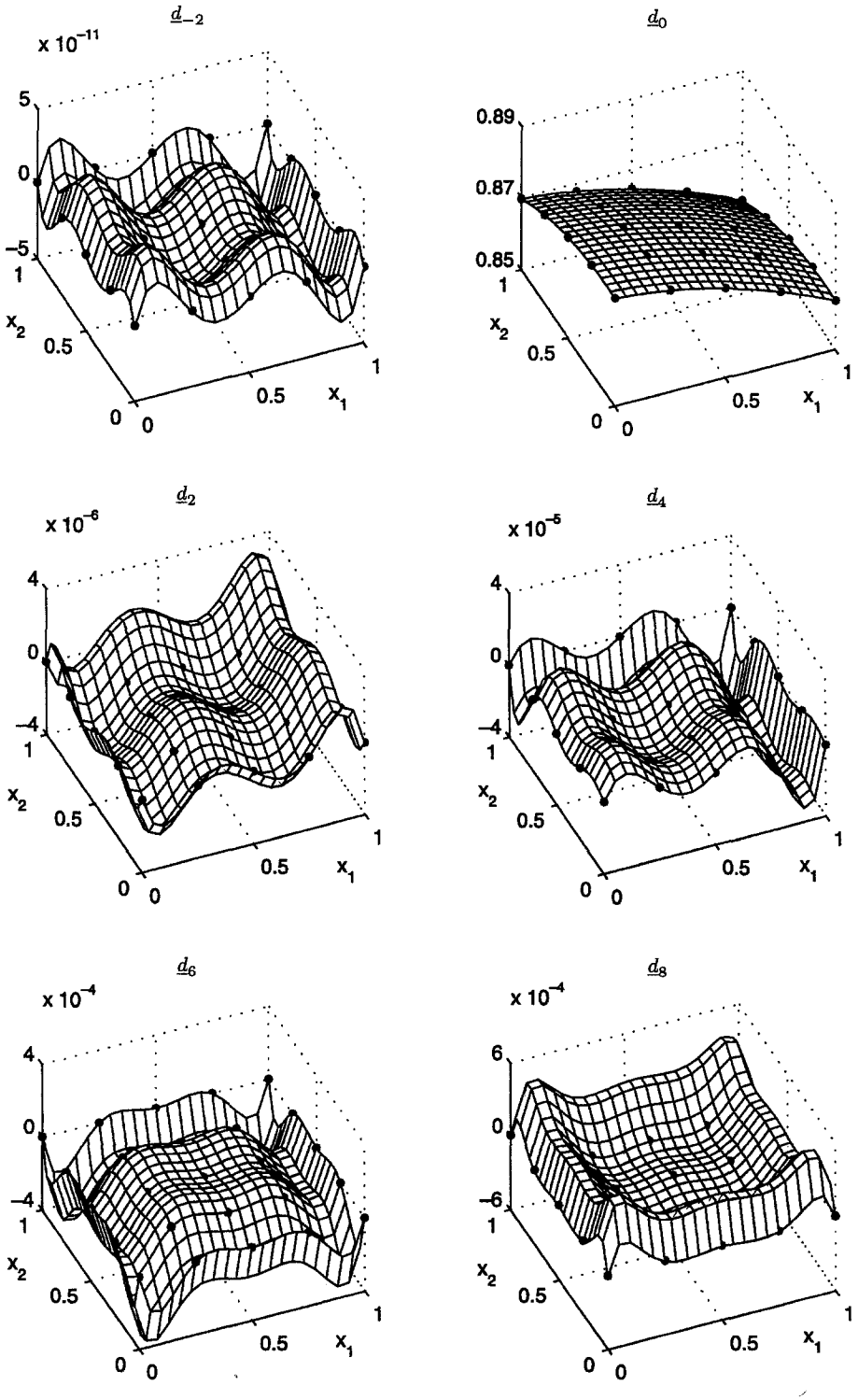


Figure 10. The first six terms from expansion (13) of $s(\underline{x}, \varepsilon)$. The solid circles represent the 5×5 equispaced Cartesian grid that served as the input data points for the interpolant.

computing the interpolant suffer from severe ill-conditioning. In this paper, we have presented an algorithm that allows stable computation of RBF interpolants for all values of ε, including the limiting case (if it exists) when the basis functions become perfectly flat. This algorithm has

also been successfully used in [15] for computing RBF based solutions to elliptic PDEs for the full range of ε-values.

The key to the algorithm lies in removing the restriction that ε be a real parameter. By allowing ε to be complex, we not only obtain a numerically stable algorithm, but we also gain a wealth of understanding about the interpolant, and we can use powerful tools to analyze it, such as Cauchy integral formula, contour integration, Laurent series, and Padé approximations.

REFERENCES

1. R. Franke, Scattered data interpolation: Tests of some methods, *Math. Comput.* **38**, 181–200, (1982).
2. W.R. Madych, Miscellaneous error bounds for multiquadric and related interpolants, *Computers Math. Applic.* **24** (12), 121–138, (1992).
3. R. Schaback, Error estimates and condition numbers for radial basis function interpolants, *Adv. Comput. Math.* **3**, 251–264, (1995).
4. R.E. Carlson and T.A. Foley, The parameter R^2 in multiquadric interpolation, *Computers Math. Applic.* **21** (9), 29–42, (1991).
5. T.A. Foley, Near optimal parameter selection for multiquadric interpolation, *J. Appl. Sci. Comput.* **1**, 54–69, (1994).
6. S. Rippa, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, *Adv. Comput. Math.* **11**, 193–210, (1999).
7. M.D. Buhmann and N. Dyn, Spectral convergence of multiquadric interpolation, In *Proceedings of the Edinburgh Mathematical Society, Volume 36*, pp. 319–333, Edinburgh, (1993).
8. J. Yoon, Spectral approximation orders of radial basis function interpolation on the Sobolev space, *SIAM J. Math. Anal.* **33** (4), 946–958, (2001).
9. B.J.C. Baxter, The asymptotic cardinal function of the multiquadric $\varphi(r) = (r^2+c^2)^{1/2}$ as $c \to \infty$, *Computers Math. Applic.* **24** (12), 1–6, (1992).
10. T.A. Driscoll and B. Fornberg, Interpolation in the limit of increasingly flat radial basis functions, *Computers Math. Applic.* **43** (3–5), 413–422, (2002).
11. C.A. Micchelli, Interpolation of scattered data: Distance matrices and conditionally positive definite functions, *Constr. Approx.* **2**, 11–22, (1986).
12. B. Fornberg, G. Wright and E. Larsson, Some observations regarding interpolants in the limit of flat radial basis functions, *Computers Math. Applic.* **47** (1), 37–55, (2004).
13. C.M. Bender and S.A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw-Hill, (1978).
14. I.R.H. Jackson, Radial basis functions: A survey and new results, University of Cambridge, Report No. DAMTP, NA16, (1988).
15. E. Larsson and B. Fornberg, A numerical study of some radial basis function based solution methods for elliptic PDEs, *Computers Math. Applic.* **46** (5/6), 891–902, (2003).