ELSEVIER

Engineering Analysis with Boundary Elements 31 (2007) 614–623

ENGINEERING ANALYSIS with BOUNDARY ELEMENTS

www.elsevier.com/locate/enganabound

Error estimate, optimal shape factor, and high precision computation of multiquadric collocation method

C.-S. Huang[a,*], C.-F. Lee[b], A.H.-D. Cheng[c]

[a]Department of Applied Mathematics and National Center for Theoretical Sciences, National Sun Yat-sen University, Kaohsiung 804, Taiwan, ROC
[b]Department of Applied Mathematics, National Sun Yat-sen University, Kaohsiung 804, Taiwan, ROC
[c]Department of Civil Engineering, University of Mississippi, P.O. Box 1848, University, MS 38677, USA

Received 30 May 2006; accepted 17 November 2006
Available online 12 March 2007

Abstract

Multiquadric (MQ) collocation method is highly efficient for solving partial differential equations due to its exponential error convergence rate. A special feature of the method is that error can be reduced by increasing the value of shape constant c in the MQ basis function, without refining the grid. It is believed that in a numerical solution without roundoff error, infinite accuracy can be achieved by letting $c \to \infty$. Using the arbitrary precision computation, this paper tests the above conjecture. A sharper error estimate than previously obtained is presented. A formula for a finite, optimal c value that minimizes the solution error for a given grid size is obtained. Using residual errors, constants in error estimate and optimal c formula can be obtained. These results are supported by numerical examples.

PACS: 02.70.Jn; 02.60.−x; 02.30.Jr

Keywords: Multiquadric collocation method; Meshless method; Error estimate; Exponential convergence; Optimal shape factor; Arbitrary precision computation

1. Introduction

Multiquadric (MQ) collocation method, first pioneered by Kansa [1], is a highly efficient numerical method for solving partial differential equations (PDEs). The advantages of the method, as laid out in Cheng et al. [2] and Cheng and Cabral [3], are:

(1) The method is "meshless" or "element-free", thus simplifies the geometric representation of the solution domain and eliminates the need of constructing and booking the element connectivity. This feature is particularly handy when moving boundary is involved, which may require frequent remeshing.
(2) It has the highest error convergence rate, the exponential convergence, $\varepsilon \sim \mathcal{O}(\lambda^{h^{-1}})$, where h is the maximum grid spacing and $0<\lambda<1$ is a constant, in contrast to the algebraic convergence of kth order Galerkin finite element $\mathcal{O}(h^k)$, or the superconvergence of h–p version finite element, $\mathcal{O}(h^{k+p})$, $p = 1, 2$.
(3) Although exponential error convergence can also be achieved by spectral methods, such as the Fourier series, Chebyshev polynomial [4], and other orthogonal basis function based collocation methods, these methods are applicable only to regular domain shapes such as rectangular or circular. MQ collocation method, on the other hand, uses scattered nodes, which is highly efficient in form fitting.
(4) Similar to an h–p finite element, the h–c MQ collocation offers two ways to reduce solution error—either refining the grid size h, or increase the value of the shape factor c. The c-scheme can achieve better solution accuracy *without* incurring additional computation cost.
(5) The collocation method allows certain classes of ill-posed boundary value problems to be solved *without* iteration [3]. The traditional technique works only with well-posed boundary conditions, which are prescribed by trial and error through optimization and iteration.

*Corresponding author.
E-mail address: huangcs@math.nsysu.edu.tw (C.-S. Huang).

0955-7997/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.enganabound.2006.11.011

In a number of numerical methods that uses global shape functions, such as the method of fundamental solution (MFS) and the MQ collocation method, it has been observed that, as the basis (shape) functions (Green's function or radial basis function) become flatter and flatter, more and more accurate solution is obtained. In the case of MFS, the radius of the enclosing circle, on which fundamental solutions are distributed, should be increased to as large as possible [5]. For MQ collocation, the shape factor c in the basis function $\sqrt{r^2+c^2}$ should be increased to its limit. In both cases, as the accuracy improves, the condition number of solution matrix becomes huge. Evidently, better solutions are obtained as the matrix becomes more singular. Ultimately, though, the roundoff error dominates and the matrix solution becomes unstable; at that point, the solution breaks down.

For the case of using MQ and Gaussian basis functions for the interpolation of multivariate functions (not for solving PDE), Madych [6] derived the estimate of approximation error as $\varepsilon\sim\mathcal{O}(e^{ac}\lambda^{ch^{-1}})$, where $0<\lambda<1$ and $a>0$ are real, positive constants. Ignoring the e^{ac} factor, it often has been argued that if computation can be carried out without roundoff error, infinite accuracy can be achieved by pushing $c\to\infty$, without refining the interpolation grid! For the case of solving PDE, Cheng et al. [2] provided the empirically established error estimate for solving Poisson equation as $\varepsilon\sim\mathcal{O}(\lambda^{c^{1/2}h^{-1}})$, which leads to the same conclusion as $c\to\infty$.

It seems that by using a global shape function, whose "flatness" can be adjusted using a shape constant, the MQ and other similar collocation methods offer a highly efficient numerical algorithm that can achieve an accuracy that is beyond the reach of the tradition methods, such as finite element and finite difference, which use compactly supported, low-degree polynomials as interpolants. It also seems that, when using MQ collocation, we should push the c value to as large as possible to achieve more and more accurate solution, limited only by the ill-conditioning of the solution matrix. Quite a few studies [7–13] have exploited this property using a variety of techniques, ranging from seeking a formula that determines the localized optimal c value based on local curvature, improving the matrix condition number by pre-conditioning, to extracting the singularity of the solution matrix. Various degrees of success have been reported.

This paper takes a step back to examine the widely adopted assumption—when roundoff error can be overcome, the optimal c value is $c\to\infty$. The statement that without roundoff error, infinite accurate solution can be obtained with a coarse grid is counterintuitive. For a given grid, does there exist a maximum and optimal value for c, whose value should not be increased unless the grid is refined? To answer the above question, we use the arbitrary precision computation capability of *Mathematica* to examine the above conjecture. First, we establish a new error estimate for solving Poisson equation, $\varepsilon\sim\mathcal{O}(e^{ac^{3/2}}\lambda^{c^{1/2}h^{-1}})$, $0<\lambda<1$ and $a>0$, which can be compared to the earlier one reported in Cheng et al. [2], $\mathcal{O}(\lambda^{c^{1/2}h^{-1}})$. It turns out that the first part of error estimate, $e^{ac^{3/2}}$, plays an important role: as c increases, this part grows, while the second part, $\lambda^{c^{1/2}h^{-1}}$, decreases. Hence, there exists a finite c value at which ε is minimum. This optimal value is found to be $c_{\max}=-\ln\lambda/3ah$. The above result suggest that for a given h, there is an upper bound for c. Once that value is reached, further reduction of error can be accomplished only by refining the grid (decreasing h). If the optimal value $c=c_{\max}$ can be found and used in the computation, the effective error for grid refinement then becomes $\varepsilon\sim\mathcal{O}(\gamma^{h^{-3/2}})$, where $0<\gamma<1$. Although this is not infinite accuracy without grid refinement, it is nevertheless a very strong convergence rate with respect to h.

To determine $c_{\max}$, knowledge about the constants λ and a in the error estimate is needed. In a real world problem, error is not known because the true solution is not given. Without data, λ and a cannot be determined. This difficulty is overcome by utilizing the residual error, which is a good measure of the error trend, but not the error magnitude. Using residual errors corresponding to a number of c and h values, these two constants λ and a can be estimated by data fitting.

The above theoretical result for optimal c is based on infinite precision computation. In a finite precision computation tool such as C^{++} program, the precision is limited to double precision (16 digits). In that case, there exist two limits for increasing c values—c is either restricted by it upper bound $c_{\max}$, or constrained by too large a condition number, whichever comes first. Experiences tell us the latter one comes first. The interplay between these two limits are examined by double and quadruple precision with C^{++} program.

The paper is organized as follows. In the next section, we review the radial basis function collocation method for solving PDE. In Section 3, we implement the arbitrary precision computation of *Mathematica* to solve boundary value problems of Poisson equation to demonstrate that there exists an upper bound for c, beyond which the solution error increases. Section 4 establishes a new error estimate, from which we obtain a formula for optimal c in Section 5. Section 6 introduces an *a posteriori* error estimate based on residual errors, and formula for estimating constants in the error estimate. The trade off between accuracy and efficiency using double and quadruple precision computation by C^{++} program is explored in Section 7. Section 8 concludes the paper.

2. Radial basis function collocation method

Solving PDEs by collocation of global shape functions is a long existing practice. We briefly introduce the procedures here. Given the governing equation

$$\mathfrak{L}\{u(\mathbf{x})\}=f(\mathbf{x}),\quad \mathbf{x}\in\Omega, \tag{1}$$

subject to the boundary condition

$$\mathfrak{B}\{u(\mathbf{x})\} = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \tag{2}$$

In the above, $\mathfrak{L}$ is a partial differential operator, $\mathfrak{B}$ is a mixed boundary operator of order lower than $\mathfrak{L}$, Ω is the solution domain, and Γ is the solution boundary. We seek the approximate solution $\hat{u}$ of (1) and (2) in the form

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \phi(r_i), \tag{3}$$

where ϕ is a radial basis function, $r_i = \|\mathbf{x} - \mathbf{x}_i\|$ is the Euclidean norm between two points, $\mathbf{x}_i$ is the center of the radial basis function, and α_i are constants to be determined by collocation. In the present paper, the inverse MQ (IMQ) is used:

$$\phi = \frac{1}{\sqrt{r_i^2 + c^2}}, \tag{4}$$

where c is a shape factor that controls the flatness (or steepness) of the shape function; as c gets larger, the function becomes less steep.

Now take a set of n_1 distinct points $\mathbf{x}_j \in \Omega$ and a set of n_2 distinct points $\mathbf{x}_j \in \Gamma$, and assume that $\mathfrak{L}$ and $\mathfrak{B}$ are linear operators. Substituting (3) into (1) and (2), we have

$$\sum_{i=1}^{n} \alpha_i \mathfrak{L}\{\phi(r_{ij})\} = f(\mathbf{x}_j), \quad \mathbf{x}_j \in \Omega, \ j = 1, 2, \ldots, n_1, \tag{5a}$$

$$\sum_{i=1}^{n} \alpha_i \mathfrak{B}\{\phi(r_{ij})\} = g(\mathbf{x}_j), \quad \mathbf{x}_j \in \Gamma, \ j = n_1 + 1, \ldots, n_1 + n_2, \tag{5b}$$

where $r_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|$. If $n_1 + n_2 = n$, then we have a linear system that can be solved for the coefficients α_i, $i = 1, \ldots, n$,

$$[\mathbf{A}]\{\alpha\} = \{F\} \iff \{\alpha\} = [\mathbf{A}]^{-1}\{F\}, \tag{6}$$

where $\mathbf{A}$ is a square coefficient matrix, $\alpha = \lfloor \alpha_1, \ldots, \alpha_n \rfloor^{\mathrm{T}}$, and $F = \lfloor f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{n_1}), g(\mathbf{x}_{n_1+1}), \ldots, g(\mathbf{x}_n) \rfloor^{\mathrm{T}}$. If $n_1 + n_2 > n$, the overdetermined linear system can be solved in the least square sense. For nonlinear operators, linearization and iterative techniques can be applied for the solution. Once the coefficients α_i are solved, the approximate solution at any $\mathbf{x} \in \Omega$ is given by the interpolation formula (3).

3. Arbitrary precision computation

As claimed by Madych [6] and subsequent studies, as the shape factor $c \to \infty$, the approximation error $\varepsilon \to 0$, limited only by the roundoff error. This suggests that if we can carry out the computation with infinite precision, we can push the c value larger and larger without limit to obtain better and better solutions. This conjecture is tested here using the arbitrary precision computation capability of *Mathematica*.

We utilize the example in Cheng et al. [2] involving the solution of a Poisson equation in two dimensions

$$\begin{aligned} \nabla^2 u(x, y) = & -\frac{751\pi^2}{144} \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4} \\ & + \frac{7\pi^2}{12} \cos\frac{\pi x}{6} \cos\frac{7\pi x}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4} \\ & + \frac{15\pi^2}{8} \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \cos\frac{3\pi y}{4} \cos\frac{5\pi y}{4}, \\ & (x, y) \in [0, 1]^2, \end{aligned} \tag{7}$$

subject to the Dirichlet type boundary conditions

$$u(0, y) = 0, \tag{8a}$$

$$u(1, y) = \sin\frac{\pi}{6} \sin\frac{7\pi}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4}, \tag{8b}$$

$$u(x, 0) = 0, \tag{8c}$$

$$u(x, 1) = \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \sin\frac{3\pi}{4} \sin\frac{5\pi}{4}. \tag{8d}$$

The exact solution of this problem is

$$u(x, y) = \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4}. \tag{9}$$

Relatively coarse grids with uniform spacing of $h = 0.2$ and 0.1 are used to distribute the scattered nodes for collocation. The computation is carried out using 100 digits precision. Multiple cases are run by varying c value from 0.1 to 100 with fixed grid. To assess the solution error, we devise two measures, the maximum error, and the root mean square (rms) error

$$\varepsilon_{\max} = \frac{\max\{|u(\mathbf{x}_i) - \hat{u}(\mathbf{x}_i)|; \ i = 1, \ldots, N\}}{|u_{\max}|}, \tag{10a}$$

$$\varepsilon_{\mathrm{rms}} = \frac{\sqrt{(1/N)\sum_{i=1}^{N} [u(\mathbf{x}_i) - \hat{u}(\mathbf{x}_i)]^2}}{|u_{\max}|}, \tag{10b}$$

where u is the exact solution, $\hat{u}$ is the MQ (approximate) solution, $\mathbf{x}_i$ are observation points uniformly distributed over the domain, N is a large number, typically taken as $100n$, with n the number of collocation nodes. We note that $\varepsilon_{\max}$ is the maximum error found in the solution region, normalized by the maximum value of u, $u_{\max}$, and $\varepsilon_{\mathrm{rms}}$ is an average error over the domain. The results are reported in Table 1.

We first examine the upper part of Table 1 based on $h = 0.2$, that is, a grid of 6×6 nodes over the domain of $[0, 1] \times [0, 1]$. We observe that the minimum $\varepsilon_{\max}$ is found at $c = 1.2$, and the minimum $\varepsilon_{\mathrm{rms}}$ is at $c = 1.4$. These values are compared to $c_{\max} = 2.0$, as predicted by formula (15) to be presented later, which is slightly off. The maximum and rms errors based on the optimal c's are 1.9% and 0.4%, respectively. If we examine the solution variability as shown in Fig. 1, we can see the result is quite amazing. For

Table 1
Error trend by increasing c based on *Mathematica* with 100 digits precision

h	c	ε_{max}	ε_{rms}	Condition number
0.2	0.1	4.36×10^{-01}	1.40×10^{-01}	$4.95 \times 10^{+02}$
0.2	1.1	2.49×10^{-02}	9.08×10^{-03}	$8.89 \times 10^{+07}$
0.2	1.2[a]	1.92×10^{-02}	6.93×10^{-03}	$2.94 \times 10^{+08}$
0.2	1.3	1.94×10^{-02}	5.12×10^{-03}	$9.22 \times 10^{+08}$
0.2	1.4[a]	1.99×10^{-02}	4.24×10^{-03}	$2.76 \times 10^{+09}$
0.2	1.5	2.08×10^{-02}	4.94×10^{-03}	$7.92 \times 10^{+09}$
0.2	2.0[b]	3.37×10^{-02}	1.85×10^{-02}	$8.49 \times 10^{+11}$
0.2	3.0	9.64×10^{-02}	5.84×10^{-02}	$1.09 \times 10^{+15}$
0.2	10.0	6.10×10^{-01}	4.19×10^{-01}	$6.38 \times 10^{+24}$
0.2	100.0	1.11×10^{0}	7.82×10^{-01}	$9.15 \times 10^{+42}$
0.1	0.1	8.67×10^{-02}	2.89×10^{-02}	$2.19 \times 10^{+03}$
0.1	2.5	6.88×10^{-06}	1.74×10^{-06}	$2.88 \times 10^{+27}$
0.1	4.0[a]	1.88×10^{-06}	6.23×10^{-07}	$6.40 \times 10^{+34}$
0.1	4.1[a,b]	2.21×10^{-06}	6.09×10^{-07}	$1.57 \times 10^{+35}$
0.1	10.0	1.5×10^{-04}	1.11×10^{-04}	$4.82 \times 10^{+49}$
0.1	100.0	6.24×10^{0}	4.56×10^{0}	$3.49 \times 10^{+87}$

[a] c value where minimum error is observed.
[b] c_{max} as estimated by (15).

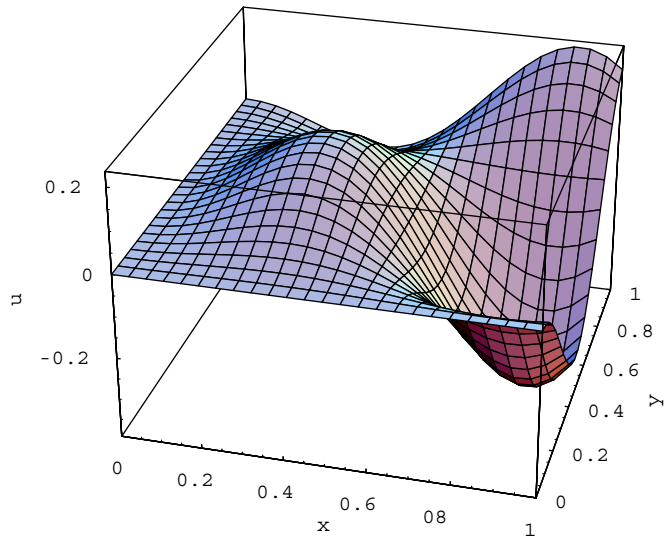

Fig. 1. Exact solution of u, based on (9).

such a coarse grid, $h = 0.2$, we can easily miss peaks and valleys between collocation nodes. The solution based on such limited data is nevertheless interpolated quite accurately. The MQ has lived up to its reputation as a highly accurate interpolation method without the need of grid refinement.

Close examination of Table 1 shows that there indeed is an upper bound for c, beyond which the error increases. We observe from the last column of Table 1 that the condition number for all cases are much less than 10^{100}; hence the roundoff error of the 100-digit computation is not the cause for the deterioration of solution.

Next, we refine the grid spacing to $h = 0.1$, and present the result as the second half of Table 1. We observe that optimal c values are found at $c = 4.0$ and 4.1 (while (15) predicts $c_{max} = 4.1$), where the maximum and rms errors are, respectively, 0.00019% and 0.000061%. This shows that the optimal c is indeed a function of h. By refining h, it is possible to push c to larger values and achieve better accuracy. Another amazing fact is the rate of error convergence—by merely halving the grid size, the error drops four orders of magnitude! This is the power of the exponential convergence. This type of performance cannot be accomplished by the piecewise, low degree polynomial interpolation of the finite element and finite difference methods.

4. Error estimate

The previously established error estimate in Cheng et al. [2], $\varepsilon \sim \mathcal{O}(\lambda^{c^{1/2}h^{-1}})$, does not predict the error behavior as observed in the preceding section. With the high precision computation, we are able to obtain more data points, particularly in the large c value range. Hence the numerical experiment is again performed to find out empirically the error estimate.

Following the lead of Madych [6] and Cheng et al. [2], we speculate that the error estimate takes the form of

$$\varepsilon \sim \mathcal{O}(e^{ac^{k_3}} \lambda^{c^{k_1} h^{k_2}}), \tag{11}$$

where k_1, k_2 and k_3 are exponents associated with solving PDE with Laplacian operator and are independent of the boundary value and geometry of the problem, $a > 0$ and $0 < \lambda < 1$, on the other hand, are constants dependent on the boundary value and geometry.

To find the exponent k_2, we can fix c and use three grid spacings h_1, h_2 and h_3, such that $h_2 = b_1 h_1$ and $h_3 = b_2 h_1$, where b_1, b_2 are real numbers. Solving these three cases we obtain the rms errors ε_1, ε_2, and ε_3. Utilizing two data points, we can show that

$$\log \frac{\varepsilon_1}{\varepsilon_2} = c^{k_1} (\log \lambda) h_1^{k_2} (1 - b_1^{k_2}), \tag{12}$$

and similarly for that ratio $\varepsilon_1/\varepsilon_3$. Dividing the two expressions, we obtain

$$\frac{\log(\varepsilon_1/\varepsilon_2)}{\log(\varepsilon_1/\varepsilon_3)} = \frac{1 - b_1^{k_2}}{1 - b_2^{k_2}}. \tag{13}$$

The above equation allows us to determine k_2. Using three data points corresponding to $h = \frac{1}{10}$, $\frac{1}{20}$, and $\frac{1}{28}$, we find rather precisely that $k_2 \approx -1$.

Next, we use a large data set obtained by letting $h = \frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{20}$, and c from 0.1 to 5.0 with increment 0.1, to perform nonlinear fitting while fixing $k_2 = -1$. We obtain $k_1 \approx \frac{1}{2}$ and $k_3 \approx \frac{3}{2}$. We also obtain $a \approx 0.72$ and $\ln \lambda \approx -0.896$ ($\lambda \approx 0.408$), which are problem geometry and boundary value dependent. Therefore, we conclude that the error

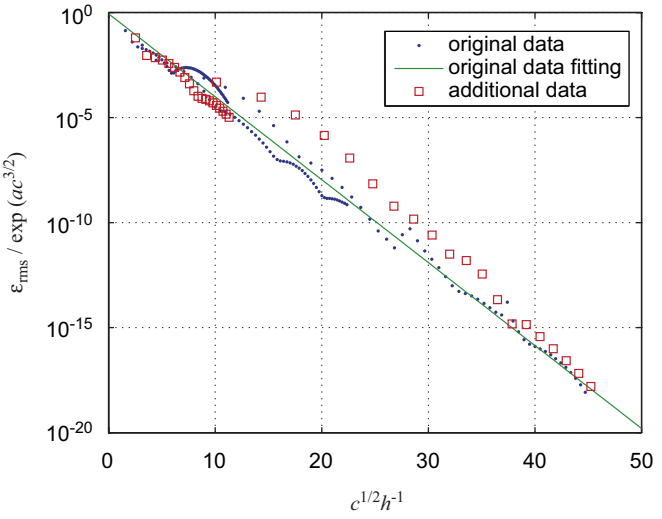

Fig. 2. Fitting for error estimate for IMQ solution of Poisson equation: composite plot of a large number of cases with different h and c values.

estimate is given by

$$\varepsilon \sim \mathcal{O}(e^{ac^{3/2}} \lambda^{c^{1/2}h^{-1}}). \tag{14}$$

We now plot the above data sets, together with more sets obtained by fixing h at $\frac{1}{8}$ and $\frac{1}{32}$ and c varying from 0.1 to 2.0 with increment 0.1, in Fig. 2. The data are plotted as $\varepsilon/e^{ac^{3/2}}$ versus $c^{1/2}h^{-1}$ on a semi-log scale. To avoid cluttering of the data, only the RMS errors are plotted. The data fitting based on (14) is presented as the straight line. The large number of data are fitted reasonable well.

It should be remarked that in the earlier investigation of error estimate [2], the c values were kept relatively small due to the lack of high precision computation capability; hence there was not sufficient variability in the e^{ac} term for its detection. The existence of the e^{ac} factor is of ultimate importance in the determination of an optimal c value, which is demonstrated in the next section.

5. Optimal shape factor

The new error estimate (14) contains an increasing part and a decreasing part with respect to increasing c, implying the existence of a c value where the error is minimum. This value is easily found by differentiating (14) with respect to c and set it to zero, and we find

$$c_{max} = -\frac{\ln \lambda}{3ah}, \tag{15}$$

which is denoted as c_{max} because our intention is to increase c to as large as possible. Note that the second derivative of (14) with respect to c at c_{max} is positive if $a>0$, $c>0$ and $0<\lambda<1$, this guarantees the local minimal of (14) at c_{max}. When c attains such value, the minimum error that can be accomplished with the given grid size h is obtained by substituting (15) into (14), and we have

$$\varepsilon \sim \mathcal{O}(\gamma^{h^{-3/2}}), \tag{16}$$

where

$$\gamma = (\lambda e^{-\ln \lambda/3})^{\sqrt{-\ln \lambda/3a}} \tag{17}$$

is a constant and $0<\gamma<1$. Therefore, the convergence rate is exponential to the power of $h^{-3/2}$, if maximum c can be used all the time. To have an idea of this convergence rate, we use the constants a and λ obtained in the current numerical example and calculate $\gamma = 0.68$. If we start the solution with a coarse mesh $h = 0.2$ and then half it to $h = 0.1$, (16) predicts an error reduction of 2500 fold! This is just about what we observe in Table 1. If we half the grid again from $h = 0.1$ to 0.05 and are able to use $c = c_{max}$, the error should reduces another 5×10^9-fold.

The validity of formulas (15) and (16) can be tested. On a grid $h = 0.05$, (15) predicts a $c_{max} = 8.3$ and (16) predicts an error magnitude of $\mathcal{O}(10^{-15})$. The numerical results are presented in Table 2 in terms of maximum error and rms error. It is easy to see that the numerical data match (15). We observe that both the c_{max} and the error magnitude are predicted very well.

We next present the maximum and rms errors of the $h = 0.05$ case for a full range of c values in Fig. 3, in semi-log scale. We note that diamond symbols represent maximum errors, squares are for rms errors, and the curve is computed from (14). First we observe that the error

Table 2
Error for $h = \frac{1}{20}$ near optimal c value

c	ε_{max}	ε_{rms}
7.0	2.22×10^{-15}	7.86×10^{-16}
7.5	1.91×10^{-15}	9.60×10^{-16}
7.7	2.37×10^{-15}	9.26×10^{-16}
8.0	2.88×10^{-15}	8.87×10^{-16}
8.5	3.58×10^{-15}	1.06×10^{-15}
9.0	3.75×10^{-15}	41.2×10^{-15}

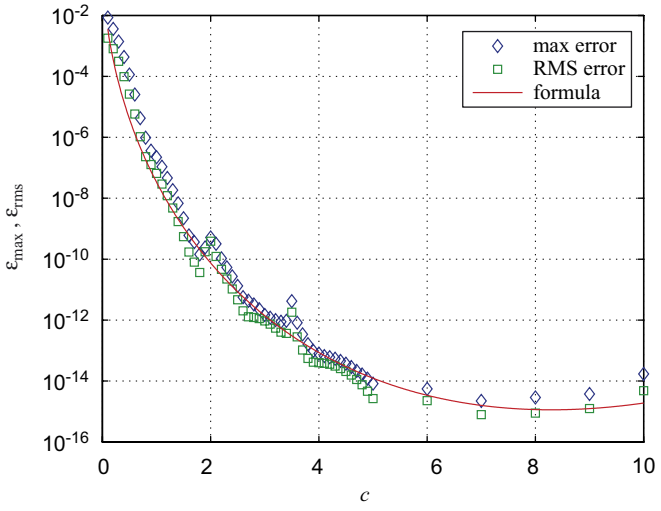

Fig. 3. Maximum and rms error for $h = \frac{1}{20}$.

drops steeply as c is increased from 0.1 to 4.0. This is accomplished by just reading different c values as the input data for the program, without any other manipulation. As c continues to increase, the rate of drop decreases, and indeed a lower bound for error is observed. It is also of interest to point out that the theoretical curve in Fig. 3 is produced using the λ and a values obtained from the best-fit of data using c value only up to 5, yet the curve in Fig. 3 is extrapolated to $c = 10$. It shows that formula (14) is fairly robust in predicting the error.

So far we have used only one example. To further test the error estimate, a second example is introduced. We consider

$$\nabla^2 u(x,y) = -8\pi^2 \sin(2\pi x)\cos(2\pi y), \quad (x,y) \in \Omega, \tag{18}$$

where $\Omega = [0,1]^2$, with the boundary condition

$$u(x,y) = 0, \quad (x,y) \in \partial\Omega. \tag{19}$$

The exact solution is

$$u(x,y) = \sin(2\pi x)\cos(2\pi y). \tag{20}$$

First, we use data with $h = \frac{1}{5}$ and $\frac{1}{10}$, and c varying from 0.1 to 5.0 with increment 0.1 to find $a = 0.838$ and $\ln\lambda = -0.907$, while k_1, k_2 and k_3 remain the same as before. We then use these data to predict $c_{\max}$ using (15) for the case with $h = \frac{1}{20}$. We calculate $c_{\max} = 7.2$, which is again an extrapolation from the data. From this $c_{\max}$ we calculate $\varepsilon \sim \mathcal{O}(4 \times 10^{-15})$. Table 3 gives the numerical results. We again observe that these two formulas predict quite well both the $c_{\max}$and the errors.

Table 3
Error for the second example, (18) and (19), with $h = \frac{1}{20}$

c	$\varepsilon_{\max}$	ε_{rms}
7.0	1.92×10^{-14}	6.82×10^{-15}
7.5	4.78×10^{-15}	1.70×10^{-15}
8.0	1.25×10^{-15}	5.14×10^{-16}
8.5	5.14×10^{-15}	2.09×10^{-15}
9.0	2.42×10^{-15}	1.14×10^{-15}

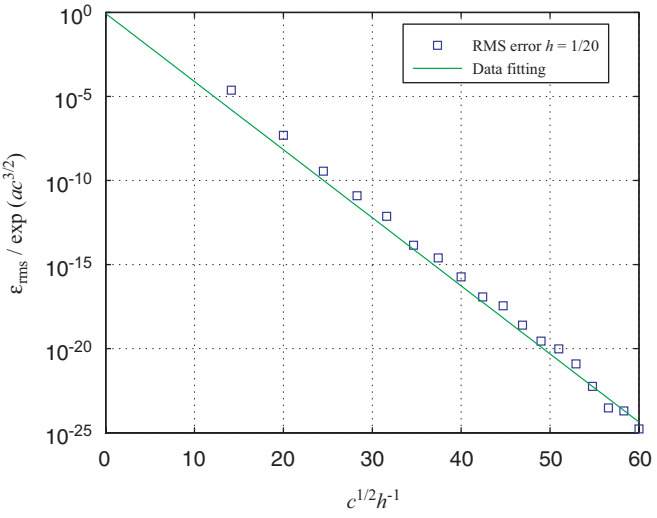

Fig. 4. Validating (14), second example.

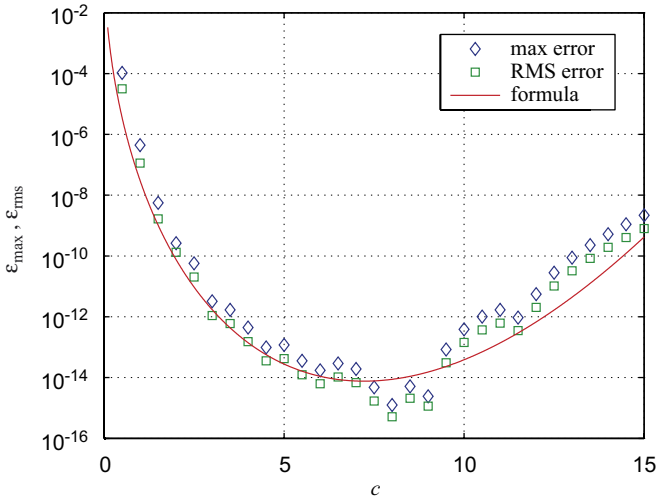

Fig. 5. Validating (15), second example.

In Fig. 4 we plot the rms error for the $h = \frac{1}{20}$ case, normalized by $e^{ac^{3/2}}$, versus $c^{1/2}h^{-1}$, on semi-log scale. In Fig. 5, $\varepsilon_{\max}$ and ε_{rms} are plotted versus c. The straight line and curve in these figures are produced using coefficients generated from best fit of data from the $h = \frac{1}{5}$ and $\frac{1}{10}$ cases. The data presented are for $h = \frac{1}{20}$. Hence we again observe the good prediction capability of the formula.

6. Residual error

The excellent predictability of error and $c_{\max}$ demonstrated in the preceding section is somewhat deceiving, because to find these constants, we have the *actual* error data to use in the fitting. In a real problem, we have no knowledge about the exact solution; hence we do not have error data to use at all. We need to find an alternative to the above procedure.

One way to estimate error is to utilize the residual error. The collocation solution enforces the governing equation and boundary conditions on a set of selected points, following (5a) and (5b). Solution between the collocation nodes are interpolated by the approximate solution (3). Hence if we check the *residual* at a node $\mathbf{x}_i$ not belonging to the collocation set,

$$\varepsilon_r(\mathbf{x}_i) = \mathfrak{L}\{u(\mathbf{x}_i)\} - f(\mathbf{x}_i) \tag{21}$$

it is generally not zero. Residual error, which can be obtained *a posteriori* from the numerical solution without the knowledge of exact solution, has been widely used as an error estimator in adaptive FEM. It has been proven in Hsiao et al. [14] that the approximation error of a solution is bounded from above and below by the residual error with some constant multiplication factors independent of the solution. The bounding constants are usually not

known. Hence residual error can be used as a good indication of error trend, but it does not give the error magnitude!

Based on the above description, it seems reasonable to write the estimate of residual error following (14) as

$$\varepsilon_r \sim C\varepsilon, \tag{22}$$

where C is a constant of unknown order of magnitude. For a given grid h, we can perform two computations using two different c values, c_i and c_{i+1}, to obtain the residual errors $\varepsilon_r(c_i)$ and $\varepsilon_r(c_{i+1})$. With two such data, it is easy to show that their ratio gives the following linear equation in the two unknowns a and $\ln\lambda$

$$\ln\frac{\varepsilon_r(c_i)}{\varepsilon_r(c_{i+1})} = (c_i^{3/2} - c_{i+1}^{3/2})a + \frac{(c_i^{1/2} - c_{i+1}^{1/2})}{h}\ln\lambda. \tag{23}$$

It seems that three such computations with different c's can form two equations for the determination of a and $\ln\lambda$. In actual practice, it is better to obtain a larger number of data points to perform least square fit. These "data points" can be obtained using a coarser grid. The obtained constants can then be used to determine the $c_{\max}$ value for a finer grid. As demonstrated in the examples in the preceding section, these constants were predicted rather well, despite that a different, coarser grid was used.

For the current problems, we perform the following steps. First, we fix the grid to $h = \frac{1}{10}$ and vary c from 0.1 to 8 to obtain a number of data points. The constants a and $\ln\lambda$ obtained from the least square fit are then used in (15) to determine $c_{\max}$ for a finer grid $h = \frac{1}{20}$. From these procedures, we obtain $c_{\max} = 8.8$ and 8.1, respectively, for the two examples, defined by (7) and (8) and (18) and (19). Hence it is recommended that these $c_{\max}$ values be used for the finer grid $h = \frac{1}{20}$ to get the optimal accuracy. In fact, these $c_{\max}$ values can be compared with those obtained using real error data, not residual error data, as $c_{\max} = 8.3$

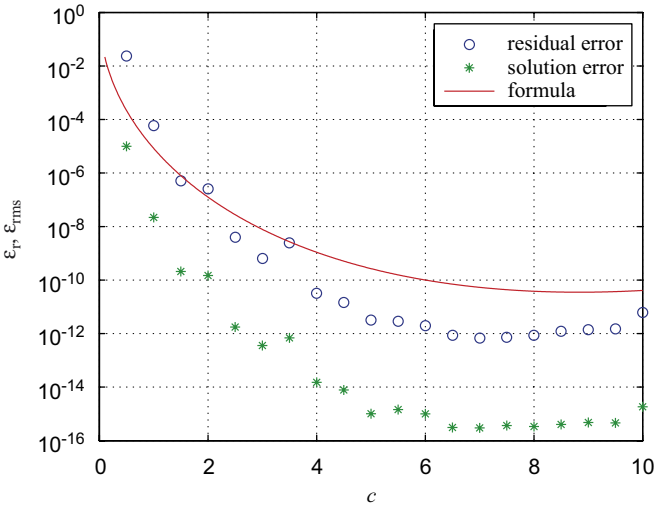

Fig. 6. Real and residual error near optimal c value, for problem defined by (7) and (8).

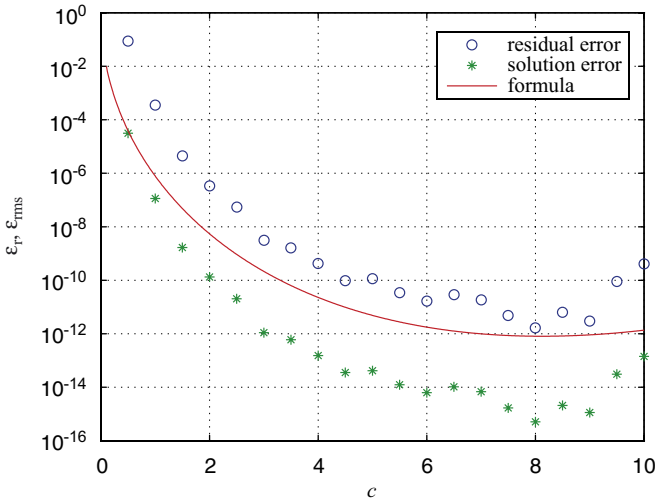

Fig. 7. Real and residual error near optimal c value, for problem defined by (18) and (19).

and 7.2. We observe that the residual error results tend to overestimate the optimal c values, but are reasonable.

To have a clear view of the performance of residual error based result, we present in Figs. 6 and 7 the error versus c around the optimal value for $h = \frac{1}{20}$, for the two examples. Although in a real case the real error is not available, for comparison purpose, we plot both the real error and the residual error in these figures. Only the rms errors are presented. First, we observe that these two type of errors are indeed of different order of magnitude, but seem to have the same trend versus the shape factor c. The curves based on formula (22) and $h = \frac{1}{10}$ data give a reasonable fit of the residual error of $h = \frac{1}{20}$, although better fit can be obtained if the $h = \frac{1}{20}$ data are used. The key is, realizing that the real error is not available, the purpose of this exercise is to see whether $c_{\max}$ can be approximately located using the residual error data. These numerical experiments seem to support this procedure.

7. Finite precision computation

In order to more accurately derive the error estimate of MQ collocation, all computations up to this point are performed using 100 significant digits. In reality, however, numerical methods are based on programming languages such as Fortran and C, in which only the single (seven significant digits) and double (16 digits) precision floating point computation is available. The impact of the limited precision on the stability of collocation method needs to be examined.

Generally speaking, due to global nature of the basis function, the solution matrix of the collocation method is ill conditioned. The truncation error caused by the limited precision can make the matrix solution unstable. In Table 4 we examine the problem defined by (7) and (8) using the grid size $h = 0.1$. The problem is solved using both 100 digits and 16 digits. The c value is increased from 0.1 to 2.0,

Table 4
Comparison of error between 100 and 16 significant digit computation, $h = 0.1$

c	ε_{max} 100 digits	ε_{rms} 100 digits	Condition number	ε_{max} 16 digits	ε_{rms} 16 digits
0.1	8.67×10^{-2}	2.89×10^{-2}	2.19×10^{03}	8.67×10^{-2}	2.89×10^{-2}
0.3	1.91×10^{-2}	4.44×10^{-3}	1.13×10^{05}	1.91×10^{-2}	4.44×10^{-3}
0.5	8.31×10^{-3}	1.99×10^{-3}	5.61×10^{08}	8.31×10^{-3}	1.99×10^{-3}
0.7	2.56×10^{-3}	6.21×10^{-4}	3.08×10^{11}	2.56×10^{-3}	6.21×10^{-4}
1.0	3.16×10^{-4}	6.30×10^{-5}	1.36×10^{15}	3.16×10^{-4}	6.30×10^{-5}
1.1	1.74×10^{-4}	5.03×10^{-5}	1.66×10^{16}	1.73×10^{-4}	5.02×10^{-5}
1.3	1.28×10^{-4}	4.16×10^{-5}	1.72×10^{18}	1.31×10^{-4}	4.17×10^{-5}
1.5	9.74×10^{-5}	2.71×10^{-5}	1.45×10^{20}	1.01×10^{-4}	3.06×10^{-5}
1.8	4.59×10^{-5}	1.23×10^{-5}	3.42×10^{22}	1.08×10^{-4}	3.05×10^{-5}
2.0	2.49×10^{-5}	7.26×10^{-6}	1.51×10^{24}	1.23×10^{-3}	3.56×10^{-4}

while c_{max} as previously reported is 4.1 for this case. In Table 4 we also report the matrix condition number. We observe that for the double precision case, the condition number reached 10^{16} when $c = 1.1$. When c is pushed to 1.3, the condition number exceeds the allowable limit of 10^{16} for 16-digit precision, to reach 10^{18}. We observe that the result for double precision computation is nevertheless as good as the 100-digit case. When c is further increased, the double precision results cease to improve and soon becomes unstable. Hence in the finite precision computation, the barrier for ultimate accuracy with a fixed grid is not c_{max}, rather it is the condition number of the matrix.

A few more observations can be made in Table 4. First, although the c-scheme with double precision is limited by the condition number such that the ultimate accuracy using c_{max} cannot be realized, the accuracy obtained with the relatively coarse mesh $h = 0.1$ is nevertheless quite good, with $\varepsilon_{rms} \sim \mathcal{O}(10^{-5})$. Second, somewhat mysteriously, it has often been observed that good results can be obtained with condition number exceeding the allowable magnitude. This can be explained by the concept of "effective condition number".

The traditional definition of condition number was given in Wilkinson [15] and became widely used; see, e.g. Atkinson [16]. For solving the linear algebraic equation $\mathbf{Ax} = \mathbf{b}$ resulting from elliptic equations, the traditional condition number is defined as Cond. $= \sigma_1/\sigma_n$, where σ_1 and σ_n are, respectively, the maximal and the minimal singular values of matrix $\mathbf{A}$. The condition number is used to provide the bounds of the relative errors from the perturbation of both $\mathbf{A}$ and $\mathbf{b}$. However, in actual applications, we only deal with a certain vector $\mathbf{b}$, and the real relative errors may be smaller, or even much smaller. Such a case was first studied in Chan and Foulser [17] and subsequently applied to a boundary value problem in Christiansen and Hansen [18], who proposed the so-called effective condition number. Applying to collocation method, Li et al. [19] have demonstrated that the effective condition number can be a few orders of magnitude smaller than the traditional one, which is consistent with what is observed here.

It remains to discuss the strategy of computation when the optimal c cannot be attained. As recommended in Cheng et al. [2], the condition number should be monitored when c value is being increased. As a most conservative measure, results should be abandoned if the condition number exceed 10^d, where d is the number of significant digits used in computation. Or, less conservatively, the effective condition number can be calculated and monitored. A better and more direct indicator for the failure of computation is the residual error. As mentioned above, residual error gives the correct trend of error. When residual error becomes unstable, it is an indication that the quality of solution has deteriorated. As demonstrated in Cheng et al. [2], the residual error can be calculated on a small number of control points and still provide a good error trend. A combination of the above checks can provide a safe way to increase c value to obtain optimal accuracy for a given grid. Once that accuracy is reached, further improvement of accuracy can be achieved only by refining the grid. The effective convergence rate of this scheme, though, is only $\varepsilon \sim \mathcal{O}(h^{-1/2})$, not as good as the $\mathcal{O}(h^{-3/2})$ for high precision computation.

The power of high precision computation as demonstrated in the preceding sections has prompted the idea of exploring the arbitrary precision computation capability of C^{++} programming. In the High-Precision Software Directory, Bailey et al. [20] provide public domain software for converting a standard C^{++} program to one that can perform arbitrary precision computation. In this paper, we utilize the QD (double–double and quad double) package. This package provides numeric types of twice the precision of IEEE double (106 mantissa bits, or approximately 32 decimal digits) and four times the precision of IEEE double (212 mantissa bits, or approximately 64 decimal digits) and basic arithmetic operations (add, subtract, multiply, divide, square root) and common transcendental functions such as the exponential, logarithm, trigonometric and hyperbolic functions. We use two data types, double precision (with 16 significant digits) and quadruple precision (with 32 significant digits). The resulting linear system (6) is solved

Table 5
CPU time (in s) for double (16 digits) and quadruple (32 digits) precision, for $h = \frac{1}{10}$

c	1.0	1.5	2.0	3.0	4.0
CPU dbl prec.	0.081	×	×	×	×
CPU quad prec.	3.27	3.29	3.26	3.25	×
$\varepsilon_{\max}$	3.2×10^{-4}	9.7×10^{-5}	2.5×10^{-5}	5.6×10^{-6}	—
ε_{rms}	6.3×10^{-5}	2.7×10^{-5}	7.3×10^{-6}	2.5×10^{-6}	—
Cond. number	1.4×10^{15}	1.2×10^{20}	1.2×10^{24}	1.9×10^{30}	6.4×10^{34}

Failed cases are marked by '×'.

Table 6
CPU time (in s) for double precision, for $h = \frac{1}{40}$

c	0.1	0.3	0.4
CPU dbl prec.	45.7	45.8	×
$\varepsilon_{\max}$	1.0×10^{-3}	9.0×10^{-6}	—
ε_{rms}	2.4×10^{-4}	2.0×10^{-6}	—
Cond. number	6.5×10^{8}	7.3×10^{19}	2.7×10^{20}

Failed case is marked by '×'.

by the partial pivoting Gaussian elimination method [21]. The code is implemented on a microcomputer with Pentuim M 1.5 GHz processor and 1.25 Gb memory. The computed results for the problem defined by (7) and (8) are presented in Tables 5 and 6.

In Table 5 we fix the grid to $h = \frac{1}{10}$ and vary c starting from 1.0. These cases are run using both the double precision (16 digits) and quadruple precision (32 digits). We observe that the double precision case succeeded only for $c = 1.0$, and failed for greater c values. This is anticipated as the condition numbers become too large for it to handle. The quadruple precision, on the other hand, can handle up to $c = 3.0$, but failed at $c = 4.0$ (while (15) predicts $c_{\max} = 4.1$). It is not surprising that the quadruple precision allows the use of larger c and gives better result. The question is, what is the amount of CPU time needed to gain this accuracy? As demonstrated in the table, switching from double to quadruple precision increases the CPU time by about 40-fold. Table 5 also shows that changing c value basically has no effect on the CPU time.

The question that whether higher precision computation is worthwhile can be answered by solving the following problem. We solve the same problem above using double precision only. Once the maximum allowable c is reached, the only way to push for better accuracy is through grid refinement. In Table 6 we show result of computation based on the grid size $h = \frac{1}{40}$. We observe that we are able to achieve comparable accuracy as the $h = \frac{1}{10}$ case using quadruple precision. The CPU time for the $h = \frac{1}{40}$ case, however, is about 14-fold of the $h = \frac{1}{10}$ and quadruple precision case. Based on the above investigation, the conclusion and recommendation is that, it is better to use a relatively coarse grid and high precision computation to push c value to as large as possible, but no larger than $c_{\max}$, than to use grid refinement to obtain comparable accuracy.

8. Conclusion

In this paper we have examined a fundamental question of the MQ collocation method: to what limit we can push the shape factor c larger to obtain better accuracy solution of PDE without the need of refining grid? The following results were obtained:

(1) Using the arbitrary precision capability of *Mathematica*, we are able to establish a new error estimate for solving Poisson equation, $\varepsilon \sim \mathcal{O}(e^{ac^{3/2}} \lambda^{c^{1/2}h^{-1}})$, refined from the earlier result by Cheng et al. [2].
(2) The new error estimate predicts a critical c value, $c_{\max} = -\ln \lambda / 3ah$, that minimizes the solution error.
(3) When $c_{\max}$ is used, the effective error convergence rate for refining the mesh is $\mathcal{O}(\gamma^{h^{-3/2}})$.
(4) Formula for utilizing residual error to find the constants a and λ in the estimate of $c_{\max}$ has been established.
(5) In finite precision computation, such as double precision, condition number, effective condition number, and residual error should be used to monitor the quality of the solution and to prevent too large a c value being used.
(6) To achieve optimal accuracy and efficiency in solving elliptic boundary value problems, it is better to use a relatively coarse grid and high precision computation capability of C^{++} to push c value to as large as possible, but no larger than $c_{\max}$, than to use grid refinement to obtain comparable accuracy. Since the arbitrary precision computation capability for C^{++} [20] is relatively new, further investigation in this direction is needed.

References

[1] Kansa EJ. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics, I. Comput Math Appl 1990;19:127–45.
[2] Cheng AHD, Golberg MA, Kansa EJ, Zammito G. Exponential convergence and h–c multiquadric collocation method for partial differential equations. Numer Methods Partial Differential Equations 2003;19(5):571–94.

[3] Cheng AHD, Cabral JJSP. Direct solution of ill-posed boundary value problems by radial basis function collocation method. Int J Numer Methods Eng 2005;64(1):45–64.
[4] Boyd JP. Chebyshev and Fourier spectral methods. 2nd ed. New York: Dover; 2001.
[5] Bogomolny A. Fundamental solutions method for elliptic boundary value problems. SIAM J Numer Anal 1985;22:644–69.
[6] Madych WR. Miscellaneous error bounds for multiquadric and related interpolators. Comput Math Appl 1992;24:121–38.
[7] Brown D, Ling L, Kansa EJ, Levesley J. On approximate cardinal preconditioning methods for solving PDEs with radial basis functions. Eng Anal Bound Elem 2005;29(4):343–53.
[8] Fornberg B, Wright G, Larsson E. Some observations regarding interpolants in the limit of flat radial basis functions. Comput Math Appl 2004;47(1):37–55.
[9] Fornberg B, Wright G. Stable computation of multiquadric interpolants for all values of the shape parameter. Comput Math Appl 2004;48(5–6):853–67.
[10] Kansa EJ, Carlson RE. Improved accuracy of multiquadric interpolation using variable shape parameters. Comput Math Appl 1992;24:99–120.
[11] Kansa EJ, Hon YC. Circumventing the ill-conditioning problem with multiquadric radial basis functions: applications to elliptic partial differential equations. Comput Math Appl 2000;39:123–37.
[12] Larsson E, Fornberg B. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. Comput Math Appl 2005;49(1):103–30.
[13] Ling L, Hon YC. Improved numerical solver for Kansa's method based on affine space decomposition. Eng Anal Boun Elem 2005;29(12):1077–85.
[14] Hsiao GC, Kleinman RE, Li RX, van den Burg PM. Residual error—a simple and sufficient estimate of actual error in solutions of boundary integral equations. In: Grilli S, Brebbia CA, Cheng AH-D, editors. Computational engineering with boundary elements, vol. 1: fluid and potential problems. Southampton: Computational Mechanics Publication; 1990. p. 73–83.
[15] Wilkinson JH. The algebraic eigenvalue problems. Oxford: Oxford University Press; 1965.
[16] Atkinson KE. An introduction to numerical analysis. New York: Wiley; 1989.
[17] Chan TF, Fousler DE. Effectively well-conditioned linear system. SIAM J Sci Stat Comput 1988;9(6):963–9.
[18] Christiansen S, Hansen PC. The effective condition number applied to error analysis of certain boundary collocation methods. J Comput Appl Math 1994;54(1):15–36.
[19] Li ZC, Lu ZZ, Hu HY, Cheng AHD. Trefftz and collocation method. Southampton: WIT Press, in press.
[20] Bailey DH, Hida Y, Jeyabalan K, Li XS, Thompson B. High precision software directory, ⟨http://crd.lbl.gov/~dhbailey/mpdist/⟩.
[21] Golub GH, van Loan CF. Matrix computations. 2nd ed. Baltimore: Johns Hopkins University Press; 1989.