Preconditioning of Radial Basis Function Interpolation Systems via Accelerated Iterated Approximate Moving Least Squares Approximation

Gregory E. Fasshauer and Jack G. Zhang

Abstract The standard approach to the solution of the radial basis function interpolation problem has been recognized as an ill-conditioned problem for many years. This is especially true when infinitely smooth basic functions such as multiquadrics or Gaussians are used with extreme values of their associated shape parameters. Various approaches have been described to deal with this phenomenon. These techniques include applying specialized preconditioners to the system matrix, changing the basis of the approximation space or using techniques from complex analysis. In this paper we present a preconditioning technique based on residual iteration of an approximate moving least squares quasi-interpolant that can be interpreted as a change of basis. In the limit our algorithm will produce the perfectly conditioned cardinal basis of the underlying radial basis function approximation space. Although our method is motivated by radial basis function interpolation problems, it can also be adapted for similar problems when the solution of a linear system is involved such as collocation methods for solving differential equations.

Keywords: Preconditioning methods · Radial Basis Functions · Accelerated Iterated least squares

1 Motivation

The solution of ill-conditioned problems has been a major challenge throughout the history of numerical analysis affecting computational accuracy, complexity and stability. One should avoid dealing with ill-conditioning whenever possible. However, there are cases when finding a theoretically good solution may initially lead

G.E. Fasshauer
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL 60616, USA, e-mail: fasshauer@iit.edu

© Springer Science + Business Media B.V. 2008

to an ill-conditioned situation. *Radial basis function* (RBF) methods have various nice features such as a certain insensitivity to the dimension of the problem domain and ease of implementation (see e.g. [7, 16, 31]). Therefore they have recently gained a great deal of attention and have been implemented in various applications such as multidimensional scattered data interpolation and the numerical solution of differential equations by collocation (see, e.g. [6, 13, 15]). In their standard formulation RBF methods often involve the solution of linear systems whose system matrices usually are full and severely ill-conditioned – especially if certain popular radial basic functions such as Gaussians or inverse multiquadrics [3, 4, 21] are used.

In this paper we extend our earlier work on iterated approximate moving least squares (IAMLS) approximation [18] by providing an accelerated version of the residual iteration algorithm that serves as a preconditioner for RBF systems and can be interpreted as a change of basis procedure. In the limit our algorithm will produce the perfectly conditioned cardinal basis of the underlying radial basis function approximation space. Although our method is motivated by radial basis function interpolation problems, it can also be adapted to similar problems involving the solution of a linear system such as collocation methods for solving differential equations.

The paper is organized as follows. In the remainder of this section we will illustrate how ill-conditioning may happen for RBF interpolation systems by describing the RBF interpolation problem and its solution approach. In Sect. 2 we provide a quick review of some standard preconditioning techniques, and relate our method to polynomial preconditioners. In Sect. 3 we go into the details of our iterated approximate MLS preconditioner including the acceleration procedure. Sect. 4 contains the numerical algorithm we have implemented in MATLAB®. We end the paper with a presentation and discussion of some numerical experiments in Sect. 5.

1.1 Conditioning of RBF Interpolation

Given $\{(x_j, f_j) : \ j = 1, 2, \ldots, N\}$, with datasites $x_j \in \mathbb{R}^s$ and values $f_j = f(x_j) \in \mathbb{R}$ assumed to come from some unknown smooth function f, we are to construct a continuous function $\mathscr{P}_f$ such that

$$\mathscr{P}_f(x_j) = f_j, \quad \text{for } j = 1, 2, \ldots, N. \tag{1}$$

Due to the *Mairhuber-Curtis* result for multi-dimensional interpolation problems (details may be found in, e.g. [16] or [31]) the function $\mathscr{P}_f$ is usually assumed to be a linear expansion of shifts of a real-valued radial basic function ϕ, that is

$$\mathscr{P}_f(\cdot) := \sum_{j=1}^{N} c_j \phi(\cdot - x_j). \tag{2}$$

Thus, condition (1) leads to the following linear system

$$\mathbf{A}\mathbf{c} = \mathbf{f}, \tag{3}$$

with the interpolation matrix $\mathbf{A}_{ij} := \phi(x_i - x_j)$ for $i, j = 1, \ldots, N$, the coefficient vector $\mathbf{c} := [c_1, \ldots, c_N]^T$, and the right-hand side $\mathbf{f} := [f_1, \ldots, f_N]^T$. If ϕ is taken to be an s-variate *strictly positive definite* function, then $\mathbf{A}$ is guaranteed to be invertible and thus (3) has a unique solution $\mathbf{c} = \mathbf{A}^{-1}\mathbf{f}$.

The definition of ϕ will sometimes involve a shape or scaling factor ε, e.g., the Gaussian basic function is given by $\phi(\cdot) = e^{-\varepsilon^2\|\cdot\|^2}$, where $\|\cdot\|$ is usually the Euclidean norm. In this paper, we restrict our attention only to fixed constant $\varepsilon > 0$. An important issue that researchers have been working on for some time is to optimize the RBF interpolant with respect to the shape parameter ε of a certain chosen radial basis (see e.g. [19, 22, 23, 30]). That is, to find ε_{opt} so that

$$\varepsilon_{opt} = \underset{\varepsilon>0}{\operatorname{argmin}}\{\|f - \mathscr{P}_f\|\}.$$

Such an optimization usually involves both a theoretical and a numerical component. Of these at least the latter is related to the conditioning of the problem. This may be demonstrated by the following simple typical examples. The left part of Fig. 1 shows an error plot for 1D interpolation problems with Gaussian RBFs (as defined above), and the right part contains analogous results for the inverse multiquadrics $\phi(\cdot) := \frac{1}{(1+\varepsilon^2\|\cdot\|^2)}$. Both sets of results were obtained using the test function $f(x) = x(1-x)$ on $[0,1]$ with 101 data points $x_j = 0.01(j-1),\ j = 1, \ldots, 101$. The solution $\mathbf{c} = \mathbf{A}^{-1}\mathbf{f}$ was computed by the default built-in solver in MATLAB®.

The main trend of the curves shows that the optimal ε value seems to fall in an interval where the accuracy behavior of the interpolant is extremely unstable. If the same experiment is performed with a slightly different data point resolution or different ε resolution, the main trend of the resulting curves remains similar, but the oscillatory segments change unpredictably and nowhere match each other. This sawtooth instability has not yet been well understood although it has been recognized

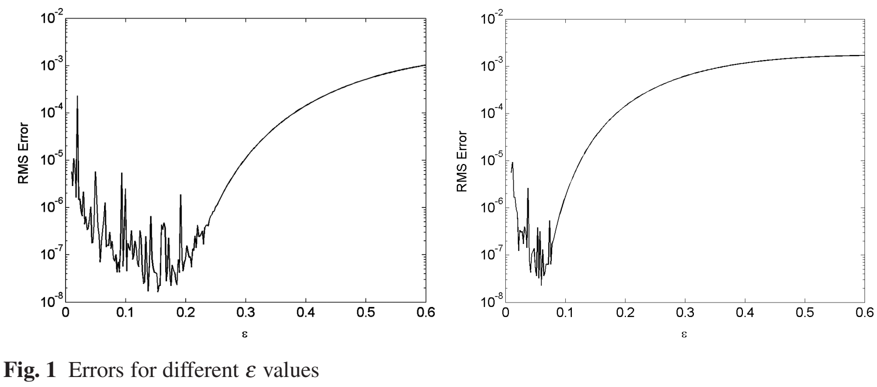


Fig. 1 Errors for different ε values

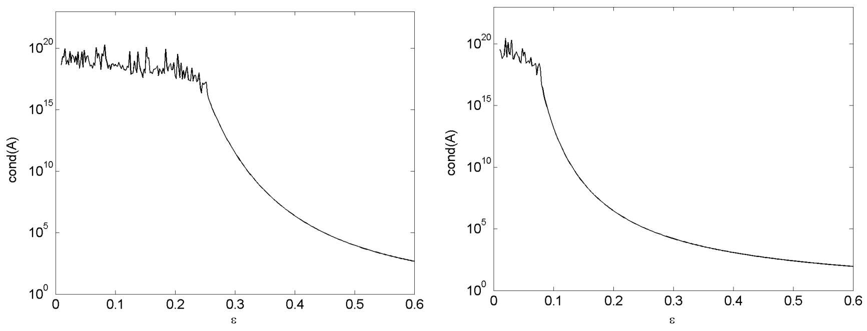


Fig. 2 cond(A) by MATLAB®

that RBF interpolants in 1-D in general converge to the Lagrange interpolating polynomial which in turn gives rise to the *Runge phenomenon* (see [11, 22]). However, it is reasonable to believe that there may be some kind of connection between this instability and the conditioning of the associated interpolation matrix **A**. When **A** is ill-conditioned, i.e., its condition number $\kappa(\mathbf{A})$ is large, then most of standard linear system solvers may become unreliable because the solution **c** found by these solvers loses a significant amount of accuracy. The next two figures (Fig. 2) are estimated condition number curves corresponding to the set of results recorded in Fig. 1.

As we have seen in the examples above, the smallest error in RBF interpolation is associated with the choice of a basic function that has a rather flat shape (i.e., ε is small), and therefore the corresponding interpolation matrix is dense and close to singular (due to almost parallel rows or columns of the interpolation matrix). This does not necessarily mean that we have to give up on the solution space spanned by such a basis because it is conceivable that there exists a "better" basis for the same linear space. Various techniques have been proposed to deal with this (seemingly) ill-conditioned problem. One obvious strategy is to apply specialized preconditioners to the system matrix. A number of papers exist on this subject starting from work of Dyn and co-workers in the mid 1980s (see, e.g. [13, 14]) or the more recent papers [2, 3, 6, 25, 26]. Another approach is to introduce a new – hopefully better – basis of the approximation space (see, e.g. [4]). Complex analysis techniques in the form of a Contour-Padé algorithm were suggested in [21], and numerical linear techniques based on the QR or singular value decompositions have also been proposed [9, 20, 24, 27].

Our approach to dealing with the ill-conditioned standard RBF basis is via a preconditioning algorithms based on our earlier work on iterated *approximate moving least-squares* (AMLS) approximation [18]. We will show that iteration on the AMLS residuals can effectively reduce the condition number of the linear system of the RBF interpolant. In fact, it reflects a change of basis which generates approximate cardinal functions. A partial theoretical justification for this change-of-basis approach (at least for RBFs with finite smoothness) is provided by the recent paper [10] where the authors show the stability of the radial basis function *space* – even if the standard *basis* may be ill-conditioned. As a result of our algorithm the eigenvalues of the preconditioned system are tightly clustered around unity, and it is known that such well conditioned linear system allow most Krylov solvers to converge quickly and also yield accurate and reliable solutions.

2 A Short Review of Preconditioning Techniques

A classical approach to overcome the difficulties associated with the ill-conditioning of systems of linear algebraic equations is to find an appropriate preconditioning matrix $\mathbf{P}$ so that $\kappa(\mathbf{PA}) \ll \kappa(\mathbf{A})$ or $\kappa(\mathbf{AP}) \ll \kappa(\mathbf{A})$. The ideal preconditioner is given by $\mathbf{A}^{-1}$ itself. Of course, use of $\mathbf{P} = \mathbf{A}^{-1}$ is impractical.

As indicated by the two different notations used above, there are different ways to apply the preconditioning. One is to left-multiply $\mathbf{P}$ to the original linear system, i.e., to consider

$$\mathbf{PAc} = \mathbf{Pf}. \tag{4}$$

Then, the solution is given as $\mathbf{c} = (\mathbf{PA})^{-1}(\mathbf{Pf})$ which is theoretically equal to the solution $\mathbf{c} = \mathbf{A}^{-1}\mathbf{f}$ given by (3) but expected to be numerically more accurate since $\mathbf{PA}$ is better conditioned than $\mathbf{A}$. However, an undesired phenomenon may occur. The relative residual $\frac{\|\mathbf{PAc} - \mathbf{Pf}\|}{\|\mathbf{Pf}\|}$ may be small partially because $\|\mathbf{P}\|$ is very large in magnitude. Thus, the absolute residual $\|\mathbf{Ac} - \mathbf{f}\|$ may not be guaranteed to be small. Moreover, if we use the coefficient vector $\mathbf{c}$ thus obtained to construct the approximant $\mathscr{P}_f$ and then evaluate it at a new set of points $y_i \in \mathbb{R}^s,\ i = 1, \ldots, M$, then the resulting values of $\mathscr{P}_f(y_i)$ are often inaccurate. This phenomenon was observed in some of our numerical examples.

A second preconditioning strategy is to change the original linear system to

$$\mathbf{APc} = \mathbf{f}, \tag{5}$$

that is

$$\mathbf{c} = (\mathbf{AP})^{-1}\mathbf{f}. \tag{6}$$

Use of the right-preconditioned system (5) is equivalent to reformulating the RBF interpolant $\mathscr{P}_f$ in (2) as

$$\mathscr{P}_f(\cdot) := \sum_{j=1}^{N} c_j \gamma_j(\cdot), \tag{7}$$

where the set $\{\gamma_j(\cdot)\}$ represents a new basis (since $\mathbf{P}$ is usually non-singular) of the space spanned by the original basis set $\{\phi(\cdot - x_j)\}$. This change of basis is provided by the transformation

$$\Gamma(\cdot) := \begin{bmatrix} \gamma_j(\cdot) \\ \vdots \\ \gamma_N(\cdot) \end{bmatrix} = \mathbf{P}^T \begin{bmatrix} \phi(\cdot - x_1) \\ \vdots \\ \phi(\cdot - x_N) \end{bmatrix} =: \mathbf{P}^T \Phi(\cdot). \tag{8}$$

As noted earlier, when $\mathbf{P} = \mathbf{A}^{-1}$, $\Gamma(\cdot)$ becomes the cardinal basis of span$\{\Phi(\cdot)\}$, i.e., $\gamma_j(x_i) = \delta_{ij}$. In that case the two basis sets are related as

$$\Phi(\cdot)^T = \Gamma(\cdot)^T \mathbf{A} \text{ or } \Phi(\cdot)^T \mathbf{A}^{-1} = \Gamma(\cdot)^T, \text{ (since } \mathbf{P} = \mathbf{A}^{-1}).$$

Note that the notation used here may appear to be a less natural one. However, we feel compelled to use it since we are working with right preconditioning defined via (4).

The evaluation of the resulting interpolant $\mathscr{P}_f$ formulated in (7) can be put in the following matrix-vector notation. Define an evaluation matrix

$$\mathbf{B}_{ij} := \phi(y_i - x_j),\ i = 1,\dots,M,\ j = 1,\dots,N.$$

Then the evaluation vector

$$\mathbf{y} = \left[\mathscr{P}_f(y_1),\dots,\mathscr{P}_f(y_M)\right]^T$$

is given as

$$\mathbf{y} = \mathbf{BPc}. \tag{9}$$

Note that the two linear systems defined in (4) and (5) are different in general, meaning that their coefficient vectors are not equal. However, we use the same notation $\mathbf{c}$ for convenience. In the next section we start our discussion of the construction of the preconditioner $\mathbf{P}$.

As an additional reason for using the right-preconditioning scheme we mention a classic preconditioning technique known as *polynomial preconditioning*. This technique is related to the method we are going to describe. According to Benzi [5] the idea to precondition a linear system goes back to Cesari in 1937 [8]. In fact, Cesari used a low degree polynomial $p(\mathbf{A})$ in $\mathbf{A}$. However, Benzi also states that polynomial preconditioners for Krylov subspace methods came into vogue in the late 1970s but are currently out of favor because of their limited effectiveness and robustness, especially for nonsymmetric problems. In the classic formulation only polynomial preconditioners of low degree (2–16) were suggested for practical use (see [1]). As we will see later, our method can stably lift the polynomial to a much higher degree.

3 Preconditioning by Iterated AMLS

Iterated approximate MLS approximation is based on the concept of approximate approximation first suggested by Maz'ya in the early 1990s [28]. The iterated AMLS approximation starts with the definition of a quasi-interpolant. Then, a sequence of approximants is constructed by adding residuals computed on the data sites to the previous approximant [18].

3.1 Iterated AMLS

The formulation of this approach can be summarized as

$$\mathscr{Q}_f^{(n)}(\cdot) := \left[\Phi(\cdot)^T \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}, \quad n=0,1,2,\ldots, \tag{10}$$

where $\Phi(\cdot)$ is the vector of original basis functions as defined in (8). Denote

$$\Gamma(\cdot)^T = \Phi(\cdot)^T \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k.$$

Clearly, $\Gamma(\cdot)$ is also a vector of functions. Moreover, its entries are linear combinations of $\phi(\cdot - x_j)$ so that it corresponds to a change of basis for $\text{span}\{\phi(\cdot - x_j)\}$. This follows since it can be shown that the transformation matrix $\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k$ has full rank.

It is known that the truncated Neumann series $\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k$ converges to $\mathbf{A}^{-1}$ if and only if $\|\mathbf{I}-\mathbf{A}\| < 1$. Thus $\mathscr{Q}_f^{(n)} \to \mathscr{P}_f$ and $\Gamma(\cdot)$ converges to a cardinal basis as $n \to \infty$. If we denote the truncated Neumann series by $\mathbf{P}^{(n)}$ then $\mathbf{P}^{(n)}\mathbf{A} = \mathbf{A}\mathbf{P}^{(n)} \to \mathbf{I}$ as $n \to \infty$ (the equality holds because $\mathbf{P}^{(n)}$ is a polynomial of $\mathbf{A}$).

We summarize some of the main properties of the iterated AMLS method, while more details are presented in [18]. Iterated AMLS can be used to compute

- An approximate inverse of $\mathbf{A}$
$$\mathbf{A}^{-1} \approx \mathbf{P}^{(n)}$$
- Approximate expansion coefficients for the standard RBF interpolation problem (3)
$$\mathbf{c} \approx \mathbf{P}^{(n)}\mathbf{f}$$
- An approximate cardinal basis $\Gamma(\cdot)$ by
$$\Gamma(\cdot)^T = \Phi(\cdot)^T\mathbf{P}^{(n)}$$

In all of these formulations

$$\mathbf{P}^{(n)} = \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k \tag{11}$$

and $\mathbf{A}$ denotes the original interpolation matrix with entries $\phi(x_i - x_j)$. Note that $\mathbf{A}$ is symmetric.

We formalize and prove the last of these statements.

Theorem 1. *The n-th iterated quasi-interpolant can be written as*

$$\mathscr{Q}_f^{(n)}(\cdot) = \Phi(\cdot)^T \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k \mathbf{f} =: \Gamma(\cdot)^T\mathbf{f},$$

i.e., $\{\gamma_1(\cdot),\dots,\gamma_N(\cdot)\}$ *provides a new — approximately cardinal — basis for* $\operatorname{span}\{\phi(\cdot-x_1),\dots,\phi(\cdot-x_N)\}$.

Proof. We use induction on n. By definition we have

$$\mathscr{Q}_f^{(n+1)}(\cdot) = \mathscr{Q}_f^{(n)}(\cdot) + \sum_{j=1}^{N}\left[f(x_j) - \mathscr{Q}_f^{(n)}(x_j)\right]\phi(\cdot - x_j).$$

Next, using the induction hypothesis yields

$$\begin{aligned}\mathscr{Q}_f^{(n+1)}(\cdot) &= \Phi(\cdot)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\mathbf{f} + \sum_{j=1}^{N}\left[f(x_j) - \Phi(x_j)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\mathbf{f}\right]\phi(\cdot - x_j)\\ &= \Phi(\cdot)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\mathbf{f} + \Phi(\cdot)^T\left[\mathbf{I} - \mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}.\end{aligned}$$

If we simplify further we obtain

$$\begin{aligned}\mathscr{Q}_f^{(n+1)}(\cdot) &= \Phi(\cdot)^T\left[\mathbf{I} - \sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^{k+1}\right]\mathbf{f}\\ &= \Phi(\cdot)^T\left[\sum_{k=0}^{n+1}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f} = \Gamma(\cdot)^T\mathbf{f}\end{aligned}$$

which completes the proof. □

Clearly, $\mathbf{P}^{(n)}$ can be used as a preconditioner for the interpolation problem discussed in previous section.

3.2 Accelerating Convergence of the Iterations

As described earlier, preconditioning by iterated AMLS requires both the coefficient vector $\mathbf{c}$ and the preconditioning matrix $\mathbf{P}^{(n)}$ to be explicitly computed. This requires expensive matrix-matrix multiplication. So, it is desirable to find a computational algorithm that can reduce the operation count and thus increase numerical accuracy.

Writing out (10) for $n = 0$ and $n = 1$, we can see

$$\mathscr{Q}_f^{(0)}(\cdot) = \Phi(\cdot)^T\mathbf{f}, \tag{12}$$

$$\mathscr{Q}_f^{(1)}(\cdot) = \Phi(\cdot)^T(\mathbf{I} + (\mathbf{I}-\mathbf{A}))\mathbf{f} = \Phi(\cdot)^T(2\mathbf{I}-\mathbf{A})\mathbf{f}, \tag{13}$$

where the matrix $\mathbf{A}_{ij} = \phi(x_i - x_j)$ arises from the evaluation of the approximant $\mathscr{Q}_f^{(0)}$ at the data sites as required for the residual calculation.

The iterative process (10) can be accelerated via the following scheme. Take

$$\tilde{\mathscr{Q}}_f^{(0)} := \mathscr{Q}_f^{(n)} = \left[\boldsymbol{\Phi}(\cdot)^T \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}, \tag{14}$$

and perform one iteration following the pattern (12–13). Then we have

$$\begin{aligned}\tilde{\mathscr{Q}}_f^{(1)}(\cdot) &= \left[\boldsymbol{\Phi}(\cdot)^T \sum_{k=0}^{n} (\mathbf{I}-\mathbf{A})^k\right]\left[2\mathbf{I}-\mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}\\ &= \boldsymbol{\Phi}(\cdot)^T\left[\sum_{k=0}^{2n+1}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}\\ &= \mathscr{Q}_f^{(2n+1)}(\cdot). \end{aligned}\tag{15}$$

Surely, the acceleration (14) and (15) can be performed continuously and consecutively starting as early as from the beginning of the original iteration.

This is formalized in

Theorem 2. *Acceleration of the iterated approximate MLS method (10) is achieved with*

$$\tilde{\mathscr{Q}}_f^{(n)}(\cdot) = \boldsymbol{\Phi}(\cdot)^T\left[\sum_{k=0}^{2^n-1}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}, \quad n=0,1,2,\ldots. \tag{16}$$

Proof. As in Theorem 1 we get

$$\mathscr{Q}_f^{(n+1)}(\cdot) = \boldsymbol{\Phi}(\cdot)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\mathbf{f} + \boldsymbol{\Phi}(\cdot)^T\left[\mathbf{I}-\mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}.$$

According to the acceleration strategy explained above we now replace $\boldsymbol{\Phi}(\cdot)^T$ by its iterated version $\boldsymbol{\Phi}^{(n)}(\cdot)^T$. That yields

$$\begin{aligned}\tilde{\mathscr{Q}}_f^{(n+1)}(\cdot) &= \boldsymbol{\Phi}(\cdot)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\mathbf{f} + \boldsymbol{\Phi}^{(n)}(\cdot)^T\left[\mathbf{I}-\mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}\\ &= \boldsymbol{\Phi}^{(n)}(\cdot)^T\left[2\mathbf{I}-\mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}\\ &= \boldsymbol{\Phi}(\cdot)^T\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\left[2\mathbf{I}-\mathbf{A}\sum_{k=0}^{n}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f}\\ &= \boldsymbol{\Phi}(\cdot)^T\left[\sum_{k=0}^{2n+1}(\mathbf{I}-\mathbf{A})^k\right]\mathbf{f} = \boldsymbol{\Phi}^{(2n+1)}(\cdot)^T\mathbf{f} = \mathscr{Q}_f^{(2n+1)}(\cdot).\end{aligned}$$

We are done by observing that the upper index of summation satisfies $\tilde{a}_{n+1} = 2\tilde{a}_n + 1$, $\tilde{a}_0 = 0$, i.e., $\tilde{a}_n = 2^n - 1$. □

Clearly, $\left\{\tilde{\mathscr{Q}}_f^{(n)}\right\}$ inherits all convergence properties of $\left\{\mathscr{Q}_f^{(n)}\right\}$ but with a faster speed of convergence.

Now, we update the notation for the preconditioner (11) to the accelerated version

$$\mathbf{P}^{(n)} := \sum_{k=0}^{2^n-1} (\mathbf{I}-\mathbf{A})^k. \tag{17}$$

In the next section we will describe how this reduction of operations may be carried out and moreover how matrix-matrix multiplications can actually be avoided during the iterations.

4 Computational Algorithm

According to the general preconditioning strategies outlined in Section 1 we have:

1. For the system (4), i.e., $\mathbf{PAc} = \mathbf{Pf}$ we can proceed as follows:
   - $\mathbf{P}^{(0)} = \mathbf{I}$
   - For $k = 1, 2, \ldots, n$, $\mathbf{P}^{(k)} = \left(2\mathbf{I} - \mathbf{P}^{(k-1)}\mathbf{A}\right)\mathbf{P}^{(k-1)}$
   - Use a standard linear solver to compute $\mathbf{c} = \left(\mathbf{P}^{(n)}\mathbf{A}\right)^{-1}\left(\mathbf{P}^{(n)}\mathbf{f}\right)$
   - Evaluate $\mathbf{y} = \mathbf{Bc}$
2. For the system (5), i.e., $\mathbf{APc} = \mathbf{f}$ we can proceed as:
   - $\mathbf{P}^{(0)} = \mathbf{I}$
   - For $k = 1, 2, \ldots, n$, $\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)}\left(2\mathbf{I} - \mathbf{AP}^{(k-1)}\right)$
   - Use a standard linear solver to compute $\mathbf{c} = \left(\mathbf{AP}^{(n)}\right)^{-1}\mathbf{f}$
   - Evaluate $\mathbf{y} = \left(\mathbf{BP}^{(n)}\right)\mathbf{f}$

For the reasons stated in Sect. 1.1 we use only the second preconditioning strategy. Note that we use $\left(\mathbf{BP}^{(n)}\right)$ to indicate that this quantity will be computed first since it will give a better accuracy based on our experimental observation. The specific computational algorithm is listed below. Note that this algorithm needs only one matrix diagonal decomposition. No further matrix-matrix multiplications are needed during the iterations.

Algorithm 1

- *Perform the eigen-decomposition* $\mathbf{A} = \mathbf{X}\boldsymbol{\Lambda}^{(0)}\mathbf{X}^{-1}$ *and initialize* $\mathbf{P}^{(0)} = \mathbf{I}$
- *For* $k = 1, 2, \ldots, n$, $\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)}\left(\mathbf{I} - \boldsymbol{\Lambda}\mathbf{P}^{(k-1)}\right)$
- *Update the preconditioner* $\mathbf{P}^{(n)} \leftarrow \mathbf{XP}^{(n)}\mathbf{X}^{-1}$

- *Compute* $\mathbf{c} = \left(\mathbf{AP}^{(n)}\right)^{-1}\mathbf{f}$
- *Evaluate* $\mathbf{y} = \left(\mathbf{BP}^{(n)}\right)\mathbf{c}$

Note that the diagonalization of $\mathbf{A}$ provides theoretical equivalences for ordering or arranging the computation in Algorithm 1. For example, it is not necessary to actually compute $\left(\mathbf{AP}^{(n)}\right)^{-1}$ since $\left(\mathbf{AP}^{(n)}\right)$ was already given in the form of a diagonal decomposition. Thus its inverse may be easily obtained via its diagonal decomposition. However, these different arrangements may yield different computational accuracies and it is not clear which arrangement is best.

Finally, it should be clear that this preconditioning method is not necessarily restricted to RBF interpolation. Indeed, it could be applied to generic linear systems as long as the system matrix satisfies the convergence requirements stated in [18].

5 Numerical Experiments and Discussion

5.1 The Basic Functions Used in Our Experiments

The experiments presented in this section use shifts of normalized radial functions such as Laguerre-Gaussians and generalized inverse multiquadrics defined on $[0,1]^2$. The following is proved in [32]:

Theorem 3.

1. *Let*
$$\psi(t) = \frac{1}{\sqrt{\pi^s}} e^{-t} L_d^{s/2}(t),$$
where $L_d^{s/2}(\cdot)$ *are generalized Laguerre polynomials of order* $s/2$ *and degree* d*. This will yield the family of* Laguerre-Gaussians.
2. *Let*
$$\psi(t) = \frac{1}{\pi^{s/2}} \frac{1}{(1+t)^{2d+s}} \sum_{j=0}^{d} \frac{(-1)^j (2d+s-j-1)!(1+t)^j}{(d-j)!\,j!\,\Gamma(d+s/2-j)},$$
which gives rise to generalized inverse multiquadrics.

In either case the function $\phi(x) = \psi\left(\|x\|^2\right)$ *is strictly positive definite in* $\mathbb{R}^s$ *and satisfies the continuous moment conditions*
$$\int_{\mathbb{R}^s} x^\alpha \phi(x)dx = \delta_{\alpha,0}, \quad 0 \le |\alpha| \le 2d+1$$
of order $2d+1$.

The specific examples of Laguerre-Gaussians and generalized inverse multiquadrics for space dimension $s = 1,2,3$ and degree $d = 0,1,2$ used in some of our numerical experiments are listed in Tables 1 and 2.

Table 1 Examples of Laguerre-Gaussians: $\phi(x) = e^{-\|x\|^2} \times$ table entry

$s \diagdown d$	0	1	2
1	$\frac{1}{\sqrt{\pi}}$	$\frac{1}{\sqrt{\pi}}\left(\frac{3}{2} - \|x\|^2\right)$	$\frac{1}{\sqrt{\pi}}\left(\frac{15}{8} - \frac{5}{2}\|x\|^2 + \frac{1}{2}\|x\|^4\right)$
2	$\frac{1}{\pi}$	$\frac{1}{\pi}\left(2 - \|x\|^2\right)$	$\frac{1}{\pi}\left(3 - 3\|x\|^2 + \frac{1}{2}\|x\|^4\right)$
3	$\frac{1}{\pi^{3/2}}$	$\frac{1}{\pi^{3/2}}\left(\frac{5}{2} - \|x\|^2\right)$	$\frac{1}{\pi^{3/2}}\left(\frac{35}{8} - \frac{7}{2}\|x\|^2 + \frac{1}{2}\|x\|^4\right)$

Table 2 Examples of generalized inverse multiquadrics

$s \diagdown d$	0	1	2
1	$\frac{1}{\pi}\frac{1}{1+\|x\|^2}$	$\frac{1}{\pi}\frac{\left(3-\|x\|^2\right)}{\left(1+\|x\|^2\right)^3}$	$\frac{1}{\pi}\frac{\left(5-10\|x\|^2+\|x\|^4\right)}{\left(1+\|x\|^2\right)^5}$
2	$\frac{1}{\pi}\frac{1}{\left(1+\|x\|^2\right)^2}$	$\frac{2}{\pi}\frac{\left(2-\|x\|^2\right)}{\left(1+\|x\|^2\right)^4}$	$\frac{3}{\pi}\frac{\left(3-6\|x\|^2+\|x\|^4\right)}{\left(1+\|x\|^2\right)^6}$
3	$\frac{4}{\pi^2}\frac{1}{\left(1+\|x\|^2\right)^3}$	$\frac{4}{\pi^2}\frac{\left(5-3\|x\|^2\right)}{\left(1+\|x\|^2\right)^5}$	$\frac{8}{\pi^2}\frac{\left(7-14\|x\|^2+3\|x\|^4\right)}{\left(1+\|x\|^2\right)^7}$

In our experiments we combine the basic functions with a shape scaling factor ε which has a strong influence on the condition number of $\mathbf{A}$ (as already illustrated in Sect. 1.1 and Fig. 1). Also, a multivariate spacing factor h is used in our experiments which is taken to be the average of the data point spacing, i.e., for 2D experiments with N points in $[0,1]^2$ we have $h = \frac{1}{\sqrt{N}-1}$. As a result we end up with, e.g., a scaled Gaussian basis function of the form

$$\phi(\cdot - x_j) = \frac{\varepsilon^2}{\pi} e^{-\varepsilon^2 \|(\cdot - x_j)/h\|^2}.$$

The use of h in the definition of the basis functions usually appears in the context of *stationary* (with h) and *non-stationary* (without h) approximation. When the domain of the problem is taken to be the unit cube $[0,\ 1]^s$, the standard approximate MLS formulation must be in stationary form which will then lead to a convergence scenario with a so-called *saturation error* [28, 29]. A similar phenomenon is also observed in standard RBF interpolation [16, 31]. Although standard RBF interpolation can be formulated in the non-stationary setting, it is observed that as the number of data points gets denser, the interpolation matrix becomes more ill-conditioned and therefore the solution becomes increasingly inaccurate and unstable [16]. In light of such numerical difficulties, the use of h reduces the effect on the conditioning of the interpolation matrix caused by the number of data points since then $\phi(x_i - x_j)$ is (approximately, if the x_j are not evenly spaced) invariant. Hence, conditioning of the interpolation matrix will *roughly* only depend on the shape parameter

ε as long as the distribution of data points is reasonably even. We use the word "*roughly*" because condition numbers for far apart numbers of data points are still significantly different based on our experiments even with fixed ε and in the stationary setting. Certainly, h may be combined with ε and associated with the data points x_j (see [12]). In such a case distinguishing the two scaling factors becomes trivial.

5.2 Effects of the Preconditioner

As mentioned earlier, the standard (left) preconditioning method defined in (4) is often unreliable. Thus, we only present results for the right preconditioning method defined in (5) and carried out by Algorithm 1.

Figure 3 demonstrates how the condition number of the system matrix changes during the preconditioning iterations. The right plot is a zoom-in of the left one. As described earlier, the preconditioner $\mathbf{P}^{(n)}$ is a truncated Neumann series which can be viewed as a polynomial preconditioner. This technique is simple and easy to implement. However, when the interpolation matrix $\mathbf{A}$ is rather ill-conditioned (which is often true), direct computation with products of $\mathbf{A}$ will rapidly lose its accuracy. Hence, the polynomial preconditioner is likely to become useless (cf. the earlier insights reported in the literature [1]). We reach a similar conclusion based on our experiments. Direct computation of $\mathbf{AP}^{(n)}$ is extremely inaccurate and unstable especially in the beginning of the iterations (i.e., with low-degree polynomial preconditioners). In a striking contrast to this, the accelerated computation is stable and yields a satisfactory condition number drop. However, as n increases (corresponding to polynomial degrees of $2^n - 1$), the accelerated computation may still gather enough numerical error so that the convergence of the Neumann series (i.e., convergence of the iterated AMLS algorithm) are destroyed.

A more comprehensive series of condition number drops is presented in Fig. 4. The left plot uses Laguerre-Gaussians (with $\varepsilon = 0.2, 0.3, 0.4, d = 0, 1, 2$) and the right one uses the generalized inverse multiquadrics (with $\varepsilon = 0.008, 0.08, 0.25$, $d = 0, 1, 2$). It can be observed that corresponding functions behave similarly.

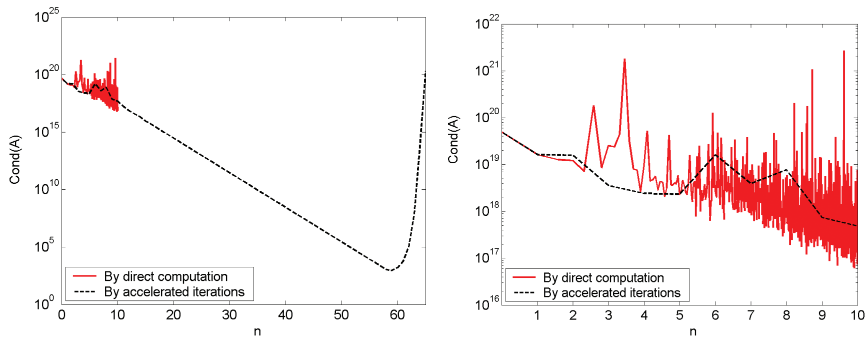
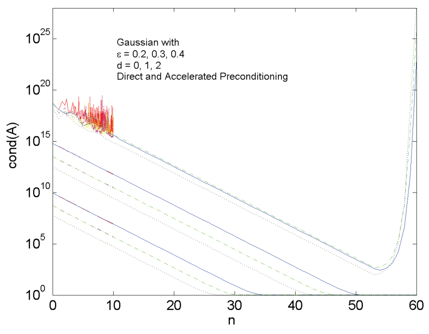


Fig. 3 Condition numbers with accelerated (black/dashed) and with standard polynomial (red/solid) preconditioning for $N = 289$ Halton points in $[0, 1]^2$



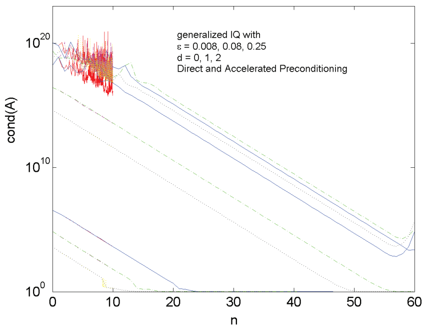
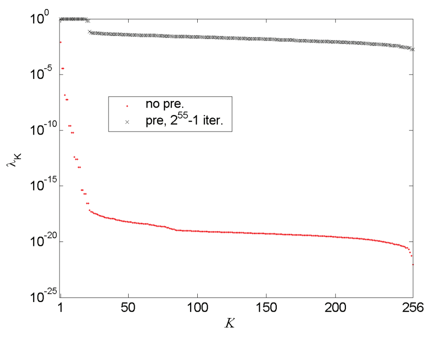


Fig. 4 Condition numbers with accelerated preconditioning, $N = 289$ Halton points



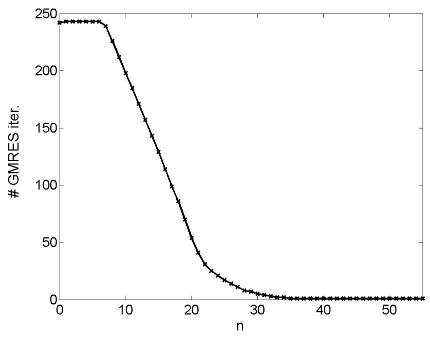


Fig. 5 Eigenvalue distribution and GMRES convergence, $N = 256$ uniform points

Generalized inverse multiquadrics and Laguerre-Gaussians seem to behave similarly with generalized inverse multiquadrics being slightly better conditioned overall.

Figure 5 shows an example of the eigenvalue clustering achieved by the accelerated preconditioning (left plot), and an example of the improvements in GMRES convergence (right figure) for the solution of a test problem based on data sampled from the 2D modified Franke function g defined on $[0,1]^2$ via

$$\begin{aligned} f(x_1,x_2) &= \frac{3}{4}\left[\exp\left(-\frac{(9x_1-2)^2}{4}-\frac{(9x_2-2)^2}{4}\right)\right. \\ &\quad \left.+\exp\left(-\frac{(9x_1+1)^2}{49}-\frac{(9x_2+1)^2}{10}\right)\right] \\ &\quad +\frac{1}{2}\exp\left(-\frac{(9x_1-7)^2}{4}-(9x_2-3)^2\right) \\ &\quad -\frac{1}{5}\exp\left(-(9x_1-4)^2-(9x_2-7)^2\right) \\ g(x_1,x_2) &= 15f(x_1,x_2)\exp\left(\frac{-1}{1-4(x_1-1/2)^2}\right)\exp\left(\frac{-1}{1-4(x_2-1/2)^2}\right). \end{aligned}$$

Table 3 Condition number comparison with [3]

Cond. no.	289		1,089		4,225	
	No pre	Pre	No pre	Pre	No pre	Pre
MQ [3]	1.506(8)	5.742(1)	2.154(9)	2.995(3)	3.734(10)	4.369(4)
TPS [3]	4.005(6)	3.330(0)	2.753(8)	1.411(2)	2.605(9)	2.025(3)
Gauss	8.796(9)	1.000(0)	6.849(10)	1.000(0)	7.632(10)	1.000(0)
IQ	1.186(8)	1.000(0)	4.284(8)	1.000(0)	1.082(9)	1.000(0)

Table 4 GMRES iteration counts as compared with [3]

No.	289		1,089		4,225	
GMRES iter.	No pre	Pre	No pre	Pre	No pre	Pre
MQ [3]	145	8	>150	15	>150	28
TPS [3]	103	5	145	6	>150	9
Gauss	>150	2	>150	2	>150	2
IQ	>150	2	>150	2	>150	2

In Table 3 we list a set of comparisons to results of the local cardinal basis method presented in [3]. Our experiments listed in Tables 3 and 4 employ 2D Halton points, a value of $n = 40$ for the accelerated preconditioner and shape parameters of $\varepsilon = 0.4$ for the Gaussian and $\varepsilon = 0.2$ for the inverse quadratic basis. Results for higher-order Laguerre-Gaussians and higher-order generalized inverse multiquadrics are similar and therefore omitted.

5.3 A Stopping Criterion

Now it is time to ask the question how to determine the number of iterations (or polynomial degree) n (or $2^n - 1$) used with the preconditioner. It is clear that, as the iteration goes on, the preconditioner $\mathbf{P}^{(n)}$ changes from $\mathbf{I}$ to $\mathbf{A}^{-1}$, that is, $\kappa(\mathbf{P}^{(0)}) = 1$ and $\kappa(\mathbf{P}^{(\infty)}) = \kappa(\mathbf{A})$, while $\kappa(\mathbf{AP}^{(0)}) = \kappa(\mathbf{A})$ and $\kappa(\mathbf{AP}^{(\infty)}) = 1$. Thus, considering only solution of the linear system, we would like to have $\kappa(\mathbf{AP}^{(n)})$ as small as possible.

However, since $\mathbf{P}^{(n)}$ will also be used for the evaluation of the interpolant $\mathscr{P}_f$ at the point set $\{y_j\}$ it is desired that $\kappa(\mathbf{P}^{(n)})$ should be kept small so as to obtain evaluation accuracy. Hence, we suggest that the iteration stops when

$$\kappa\left(\mathbf{P}^{(n)}\right) = \kappa\left(\mathbf{AP}^{(n)}\right). \tag{18}$$

Let σ_{max} and σ_{min} be the largest and smallest singular values of $\mathbf{A}$. Recall that ϕ is strictly positive definite, that is, $\mathbf{A}$ is positive definite and symmetric and $\|\mathbf{I} - \mathbf{A}\|_2 < 1$. Thus, $0 < \sigma_{min} < \sigma_{max} < 1$. It can be verified via singular value decomposition for **A** that

$$\kappa\left(\mathbf{P}^{(n)}\right) = \frac{\sum_{k=0}^{2^n-1}(1-\sigma_{min})^k}{\sum_{k=0}^{2^n-1}(1-\sigma_{max})^k} = \frac{1-(1-\sigma_{min})^{2^n}}{1-(1-\sigma_{max})^{2^n}}\frac{\sigma_{max}}{\sigma_{min}}, \tag{19}$$

and

$$\kappa\left(\mathbf{AP}^{(n)}\right) = \frac{1-(1-\sigma_{max})^{2^n}}{1-(1-\sigma_{min})^{2^n}}. \tag{20}$$

Thus, the ideal number of iterations can be estimated by solving the nonlinear equation

$$\frac{1-(1-\sigma_{max})^{2^n}}{1-(1-\sigma_{min})^{2^n}} = \sqrt{\frac{\sigma_{max}}{\sigma_{min}}} \tag{21}$$

for n. Note that for a symmetric positive definite **A** its eigen-decomposition in Algorithm 1 is identical to its singular value decomposition. Thus, there is no extra computation needed for finding σ_{max} and σ_{min}. If Shepard's method is used, then **A** is just a product of a symmetric positive definite matrix and a diagonal matrix (performing row or column scaling). Thus, with a little adaption the formulation that we have discussed will still be applicable.

In Fig. 6 and Table 5 we present a set of error comparisons obtained with this optimally stopped preconditioning algorithm. Both the original system and the iteratively preconditioned system (computed by Algorithm 1 with the suggested optimal number of iterations) are solved by a MATLAB® GMRES method with

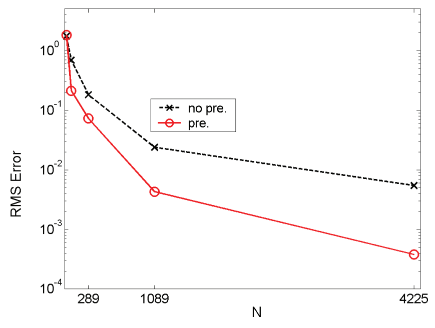


Fig. 6 Error drop comparison with automatic stopping criterion. N uniform points in 2D

Table 5 "Optimal" number of preconditioning iterations, n, for the results shown in Fig. 6

N	289			1,089			4,225		
$\kappa(\mathbf{A})$	1.4e+11			2.1e+12			4.8e+12		
	RMSerr	GMRES	n	RMSerr	GMRES	n	RMSerr	GMRES	n
No pre	7.2e−2	>10		2.4e−2	>10		5.5e−3	>10	
pre	1.8e−1	>10	20	4.3e−3	6	20	3.8e−4	1	21

default settings, that is, $\mathbf{c} = \text{gmres}(\mathbf{A},\mathbf{b})$ in MATLAB® syntax for $\mathbf{Ac} = \mathbf{b}$, and $\mathbf{c} = \text{gmres}(\mathbf{AP},\mathbf{b})$ for $\mathbf{APc} = \mathbf{b}$. When $N = 4{,}225$ the condition number $\kappa(\mathbf{A}) \approx 10^{12}$, i.e., $\kappa(\mathbf{AP}) \approx \kappa(\mathbf{P}) \approx 10^6$, and the preconditioning algorithm terminates after $n = 21$ iterations. With our preconditioning the GMRES method converges within the default maximum number of iterations for a default tolerance while it does not converge without preconditioning. Note that the error drop without preconditioning is also reasonably stable although it is larger than that achieved with preconditioning. This happens because that lack of exactness at data sites is not necessarily reflected in the global accuracy of the solution.

5.4 Concluding Remarks

We have demonstrated that the proposed accelerated preconditioning method is effective and easy to implement. The diagonalization performed in the acceleration Algorithm 1 improves the speed of computation without contributing any extra numerical inaccuracy. The accuracy of $\mathbf{P}^{(n)}$ as a preconditioner is actually immaterial as long as $\kappa\left(\mathbf{AP}^{(n)}\right) \ll \kappa(\mathbf{A})$. However, since the accuracy of the evaluation (via $\mathbf{BP}^{(n)}$ or $\mathbf{BP}^{(n)}\mathbf{c}$) also depends highly on $\kappa\left(\mathbf{P}^{(n)}\right)$ it is clear that for extremely ill-conditioned problems (with $\kappa(\mathbf{A}) > 10^{20}$) this preconditioning method will not work very well.

Based on the numerical experiments we performed in MATLAB®, our preconditioning method works efficiently and accurately when $\kappa\left(\mathbf{AP}^{(n)}\right)$ is in the order of $10^{12} \sim 10^{14}$. Thus it has certain advantages over most of the standard MATLAB® solvers.

When $\kappa(\mathbf{A})$ exceeds 10^{20}, $\kappa\left(\mathbf{AP}^{(n)}\right)$ can still be significantly reduced, but then $\mathbf{P}^{(n)}$ becomes very ill-conditioned. Also, in such a case, a non-linear solver with higher precision is required to solve (21) for n.

Finally, recall that our preconditioning process starts with a well-formulated quasi-interpolant. Thus, the method can also give good performance in situations where interpolation is not required or preferred, such as, for example, optimized smooth approximation of noisy data (see [17]).

References

1. S. F. Ashby, T. A. Manteuffel, and J. S. Otto, "A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems", *SIAM J. Sci. Statist. Comput.* Vol. 13, pp. 1–29, 1992.
2. B. J. C. Baxter, "Preconditioned conjugate gradients, radial basis functions, and Toeplitz matrices", *Comput. Math. Appl.* Vol. **43**, pp. 305–318, 2002.

3. R. K. Beatson, J. B. Cherrie, and C. T. Mouat, "Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration", *Adv. Comput. Math.* Vol. **11**, pp. 253–270, 1999.
4. R. K. Beatson, W. A. Light, and S. Billings, "Fast solution of the radial basis function interpolation equations: domain decomposition methods", *SIAM J. Sci. Comput.* Vol. **22**, pp. 1717–1740, 2000.
5. M. Benzi, "Preconditioning techniques for large linear systems: a survey", *J. Comput. Phys.* Vol. **182**, pp. 418–477, 2002.
6. D. Brown, L. Ling, E. Kansa, and J. Levesley, "On approximate cardinal preconditioning methods for solving PDEs with radial basis functions", *Eng. Anal. Bound. Elem.* Vol. **29**, pp. 343–353, 2005.
7. M. D. Buhmann, "Radial Basis Functions", Cambridge University Press, New York, 2003.
8. L. Cesari, "Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive", *Ricerca Sci., Roma* Vol. **2** 8_{I}, pp. 512–522, 1937.
9. C. S. Chen, H. A. Cho and M. A. Golberg, "Some comments on the ill-conditioning of the method of fundamental solutions", *Eng. Anal. Bound. Elem.* Vol. **30**, pp. 405–410, 2006.
10. S. De Marchi and R. Schaback, "Stability of Kernel-Based Interpolation", preprint, 2007.
11. T. A. Driscoll and B. Fornberg, "Interpolation in the limit of increasingly flat radial basis functions", *Comput. Math. Appl.*, Vol. **43**, pp. 413–422, 2002.
12. T. A. Driscoll and A. R. H. Heryudono, "Adaptive residual subsampling methods for radial basis function interpolation and collocation problems", *Comput. Math. Appl.*, Vol. **53**, pp. 927–939, 2007.
13. N. Dyn, "Interpolation of scattered data by radial functions", in *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic New York, pp. 47–61, 1987.
14. N. Dyn, D. Levin, and S. Rippa, "Numerical procedures for surface fitting of scattered data by radial functions", *SIAM J. Sci. Statist. Comput.* Vol. **7**, pp. 639–659, 1986.
15. G. E. Fasshauer, "Solving partial differential equations by collocation with radial basis functions", in *Surface Fitting and Multiresolution Methods*, A. Le Mehaute, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, TN, pp. 131–138, 1997.
16. G. E. Fasshauer, "Meshfree approximation methods with MATLAB", *Interdisciplinary Mathematical Sciences*, Vol. **6**, World Scientific Publishers, New York, 2007.
17. G. E. Fasshauer and J. G. Zhang, "Scattered data approximation of noisy data via iterated moving least squares", in *Curve and Surface Fitting: Avignon 2006*, T. Lyche, J. L. Merrien and L. L. Schumaker (eds.), Nashboro Press, Brentwood, TN, pp. 150–159, 2007.
18. G. E. Fasshauer and J. G. Zhang, "Iterated approximate moving least squares approximation", in *Advances in Meshfree Techniques*, V. M. A. Leitao, C. Alves and C. A. Duarte (eds.), Springer, Singapore, pp. 221–240, 2007.
19. G. E. Fasshauer and J. G. Zhang, "On choosing "optimal" shape parameters for RBF approximation", *Numer. Algorithms* Vol. **45**, pp. 345–368, 2007.
20. B. Fornberg and C. Piret, "A stable algorithm for flat radial basis functions on a sphere", *SIAM J. Sci. Comp.* Vol. **30**, pp. 60–80, 2007.
21. B. Fornberg and G. Wright, "Stable computation of multiquadric interpolants for all values of the shape parameter", *Comput. Math. Appl.* Vol. **47**, pp. 497–523, 2004.
22. B. Fornberg and J. Zuev, "The Runge phenomenon and spatially variable shape parameters in RBF interpolation", *Comput. Math. Appl.* Vol. **54**, pp. 379–398, 2007.
23. E. J. Kansa and R. E. Carlson. "Improved accuracy of multiquadric interpolation using variable shape parameters", *Comput. Math. Applic.* Vol. **24**, pp. 99–120, 1992.
24. C.-F. Lee, L. Ling and R. Schaback, "On convergent numerical algorithms for unsymmetric collocation", *Adv. Comp. Math*, to appear.
25. L. Ling and E. J. Kansa, "Preconditioning for radial basis functions with domain decomposition methods", *Math. Comput. Model.* Vol. **40**, pp. 1413–1427, 2004.
26. L. Ling and E. J. Kansa, "A least-squares preconditioner for radial basis functions collocation methods", *Adv. Comp. Math.* Vol. **23**, pp. 31–54, 2005.
27. L. Ling and R. Schaback, "Stable and convergent unsymmetric meshless collocation methods", *SIAM J. Numer. Anal.*, to appear.

28. V. Maz'ya, "A new approximation method and its applications to the calculation of volume potentials. Boundary point method", in *DFG-Kolloquium des DFG-Forschungsschwerpunktes "Randelementmethoden"*, 1991.
29. V. Maz'ya and G. Schmidt, "On quasi-interpolation with non-uniformly distributed centers on domains and manifolds", *J. Approx. Theory*, Vol. **110**, pp. 125–145, 2001.
30. S. Rippa, "Algorithm for selecting a good value for the parameter *c* in radial basis function interpolation", *Adv. Comput. Math.* Vol. **11**, pp. 193–210, 1999.
31. H. Wendland, Scattered Data Approximation, Cambridge University Press, Cambridge, 2005.
32. J. G. Zhang, "Iterated Approximate Moving Least-Squares: Theory and Applications", *Ph.D. Dissertation, Illinois Institute of Technology*, 2007.