Numerical Algorithms manuscript No.
(will be inserted by the editor)

On Choosing "Optimal" Shape Parameters for RBF Approximation

Gregory E. Fasshauer · Jack G. Zhang

Received: date / Accepted: date

Abstract Many radial basis function (RBF) methods contain a free shape parameter that plays an important role for the accuracy of the method. In most papers the authors end up choosing this shape parameter by trial and error or some other ad-hoc means. The method of cross validation has long been used in the statistics literature, and the special case of leave-one-out cross validation forms the basis of the algorithm for choosing an optimal value of the shape parameter proposed by Rippa in the setting of scattered data interpolation with RBFs.

We discuss extensions of this approach that can be applied in the setting of iterated approximate moving least squares approximation of function value data and for RBF pseudo-spectral methods for the solution of partial differential equations. The former method can be viewed as an efficient alternative to ridge regression or smoothing spline approximation, while the latter forms an extension of the classical polynomial pseudo-spectral approach. Numerical experiments illustrating the use of our algorithms are included.

Keywords Radial basis functions · Approximate moving least squares · Shape parameter · Cross validation · Pseudo-spectral methods

Mathematics Subject Classification (2000) 65D05 · 65D15 · 65M70

1 Introduction

Many radial basis functions (RBFs) contain a free shape parameter ε that can be tuned by the user. In the traditional RBF approach (as described in Section 2 below) the problem boils down to the solution of a system of linear equations with a system matrix $\mathbf{A}$ whose condition number grows (often exponentially) as the shape parameter ε goes to zero. On the other hand, standard error estimates via the so-called power function indicate (exponentially) better accuracy for decreasing ε. This inter-dependence is known in the literature (see, e.g., [32]) as the *uncertainty* or *trade-off* principle and has been recognized as an important issue by many researchers.

Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, U.S.A.
E-mail: {fasshauer,zhangou}@iit.edu

More recent research conducted mostly by Fornberg and co-workers (see, e.g., [2, 15, 16, 22, 23]) has shown that one may be able to overcome the conditioning problems of the traditional RBF approach by using other techniques such as the Contour-Padé algorithm of [15]. This research has confirmed that — even when circumventing the ill-conditioning of the system matrix — there usually is a value of the shape parameter which results in optimal approximation errors.

Due to a connection between RBF interpolation and polynomial interpolation (also studied in some of the papers listed above) it is likely that RBF interpolation also suffers from a phenomenon that is similar to the well-known Runge phenomenon for polynomial interpolation, and that the choice of the shape parameter can alleviate this effect. Of course, other factors such as center placement are likely to play an important role in successfully dealing with this Runge phenomenon, also. Nevertheless, once a set of RBF centers has been chosen, it is of importance to find the corresponding optimal shape parameter ε.

In summary, regardless of whether one follows the traditional RBF approach (and therefore looks for a good balance between accuracy and stability), or whether one applies stabilization techniques such as the Contour-Padé algorithm (which is applicable only to rather small problems), the flexibility and potential for improved accuracy offered by the shape parameter present in many RBFs should be exploited by the user. It is somewhat ironic that this freedom is often viewed as a disadvantage since the user is forced to make a decision on the choice of the shape parameter. This leads to the fact that the authors of many papers end up choosing ε by a rather costly trial and error approach performing their numerical experiments over and over again until they end up with a satisfactory result. Alternatively, the shape parameter is picked by some (non-optimal) ad-hoc criterion. For example, in one of the earliest RBF papers on (inverse) multiquadric RBF interpolation in $\mathbb{R}^2$ Hardy [18] suggests the use of $\varepsilon = 1/(0.815d)$, where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$, and d_i is the distance from the data point $\mathbf{x}_i$ to its nearest neighbor. Franke [17] on the other hand recommends $\varepsilon = \frac{0.8\sqrt{N}}{D}$, where D is the diameter of the smallest circle containing all data points. Here ε is used as in the examples listed in Section 2 below, and thus may differ slightly from the discussion in the original papers.

The method of cross validation has long been used in the statistics literature, and the special case of leave-one-out cross validation (LOOCV) forms the basis of the algorithm for choosing an optimal value of the RBF shape parameter proposed by Rippa [27] in the setting of scattered data interpolation. We will review this algorithm below in Section 2 as it forms the starting point for our work.

In particular, we discuss extensions of Rippa's LOOCV algorithm that can be applied in the setting of iterated approximate moving least squares (AMLS) approximation and for RBF pseudo-spectral (PS) methods for the solution of partial differential equations. In Section 4 we will discuss how an LOOCV strategy can be used in the context of iterated AMLS approximation (which we review in Section 3) to find both the optimal number of iterations and the optimal shape parameter. As our numerical experiments presented in Section 5 will show, this iterative approach can be viewed as an efficient alternative to ridge regression or smoothing spline approximation. In Section 6 we switch to our second application and briefly review the RBF-PS method showing that it generalizes the classical polynomial pseudo-spectral method. Rippa's algorithm is modified in Section 7 to yield the optimal shape parameter for the RBF-PS approach, and some numerical experiments are included in Section 8.

2 Some Background Information

In the standard RBF interpolation problem we are given generally scattered data sites $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \Omega$ and associated real function values $f(\mathbf{x}_i)$, $i = 1, \ldots, N$. Here Ω is usually some bounded domain in $\mathbb{R}^s$. It is our goal to find a (continuous) function $P_f : \mathbb{R}^s \to \mathbb{R}$ that interpolates the given data, i.e., such that

$$P_f(\mathbf{x}_i) = f(\mathbf{x}_i), \qquad i = 1, \ldots, N. \tag{1}$$

In the RBF literature (see, e.g., [8,32]) one assumes that this interpolant is of the form

$$P_f(\mathbf{x}) = \sum_{j=1}^{N} c_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|), \tag{2}$$

where the basic function φ is (in this paper) assumed to be strictly positive definite and the coefficients $\mathbf{c} = [c_1, \ldots, c_N]^T$ are found by enforcing the interpolation constraints (1). This implies that

$$\mathbf{c} = \mathbf{A}^{-1}\mathbf{f},$$

where $\mathbf{A}_{ij} = \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$ and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$.

The kind of RBFs $\Phi = \varphi(\|\cdot\|)$ we will be mostly interested in are the Gaussians $\Phi(\mathbf{x}) = e^{-\varepsilon^2\|\mathbf{x}\|^2}$, Matérn functions such as

$$\begin{aligned}
\Phi(\mathbf{x}) &= e^{-\varepsilon\|\mathbf{x}\|}, \\
\Phi(\mathbf{x}) &= (1 + \varepsilon\|\mathbf{x}\|)e^{-\varepsilon\|\mathbf{x}\|}, \\
\Phi(\mathbf{x}) &= (3 + 3\varepsilon\|\mathbf{x}\| + \varepsilon^2\|\mathbf{x}\|^2)e^{-\varepsilon\|\mathbf{x}\|},
\end{aligned}$$

or the multiquadrics $\Phi(\mathbf{x}) = (1+\varepsilon^2\|\mathbf{x}\|^2)^\beta$, $\beta \notin 2\mathbb{N}$. While this latter family of functions is quite popular, only those multiquadrics for $\beta < 0$ are strictly positive definite. Other commonly used RBFs such as polyharmonic splines or compactly supported functions will not play a role in this paper.

Many nice properties of the RBF interpolant are emphasized in the RBF literature. For example, P_f is the minimum norm interpolant to f in a Hilbert space H, i.e.,

$$P_f = \operatorname{argmin}\left\{\|s\|_H \,:\, s \in H,\ s(\mathbf{x}_i) = f(\mathbf{x}_i),\ i = 1, \ldots, N\right\}.$$

Here H is actually a reproducing kernel Hilbert space with reproducing kernel $\varphi(\|\cdot - \cdot\|)$ — sometimes referred to as the *native space* of φ. Moreover, the interpolant P_f is even a best approximation to f in the sense that

$$\|f - P_f\|_H \le \|f - s\|_H$$

for all $s \in H_X = \{s = \sum_{j=1}^{N} a_j \varphi(\|\cdot - \mathbf{x}_j\|),\ \mathbf{x}_j \in X\}$.

Clearly, the space H varies with the choice of shape parameter ε present in all of the RBFs listed above, and one will want to find the "best" Hilbert space in which the interpolant P_f is optimal.

A popular strategy for estimating this shape parameter based on the given data $(\mathbf{x}_i, f(\mathbf{x}_i))$, $i = 1, \ldots, N$, is the method of *cross validation* well-known in statistics. In [27] an algorithm is described that corresponds to a variant of cross validation known as *"leave-one-out" cross validation* (LOOCV). In this algorithm an optimal value of ε is selected by minimizing a cost function that collects the errors for a sequence of

partial fits to the data. Even though the true error of the full RBF interpolant is not known in general, we can *estimate* this error by splitting the data set into two parts: one to base an approximation to the interpolant on, and the other to base an estimate for the error on. In LOOCV one splits off only a single data point at which to compute the error, and then bases the approximation to the interpolant on the remaining $N-1$ data points. This procedure is in turn repeated for each one of the N data points. The result is a vector of error estimates and the cost function that is used to find the "optimal" value of ε is provided by some norm of this error vector. One of the main contributions of [27] was to show that this procedure can be executed efficiently without having to compute all of the N partial interpolants based on subsets of $N-1$ data points (cf. formula (5) below).

Since the LOOCV method is based on errors computed on the given data, the predicted "optimal" shape parameter is usually close to the actual optimum value (which, of course, can only be found if we already know the function to be reconstructed).

For the following discussion we define

$$\mathbf{x}^{[k]} = [\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_N]^T,$$

the vector of datasites with the point $\mathbf{x}_k$ removed (indicated by the superscript $[k]$). Similarly, we define $\mathbf{f}^{[k]}$, $P_f^{[k]}$, $\mathbf{c}^{[k]}$ and all other quantities appearing in LOOCV-like algorithms later on.

Specifically, if $P_f^{[k]}$ is the partial RBF interpolant to the data $\mathbf{f}^{[k]}$, i.e.,

$$P_f^{[k]}(\mathbf{x}) = \sum_{j=1}^{N-1} \mathbf{c}_j^{[k]} \varphi(\|\mathbf{x} - \mathbf{x}_j^{[k]}\|),$$

and if e_k is the error estimator

$$e_k = f(\mathbf{x}_k) - P_f^{[k]}(\mathbf{x}_k),$$

then the quality of the overall fit to the entire data set will be determined by the norm of the vector of errors $\mathbf{e} = [e_1, \ldots, e_N]^T$ obtained by removing in turn each one of the data points and comparing the resulting fit with the (known) value at the removed point as described above. As mentioned earlier, the norm of $\mathbf{e}$ as a function of ε will serve as a *cost function* for the shape parameter. In principle any vector norm can be used. In [27] the author presented examples based on use of the ℓ_1 and ℓ_2 norms. We will base all of our numerical experiments on the use of the ℓ_2 norm.

According to the ideas presented so far the LOOCV algorithm for RBF interpolation can be summarized as follows:

Algorithm 1

Fix ε
For $k = 1, \ldots, N$
Let

$$P_f^{[k]}(\mathbf{x}) = \sum_{j=1}^{N-1} \mathbf{c}_j^{[k]} \varphi\left(\|\mathbf{x} - \mathbf{x}_j^{[k]}\|\right) \tag{3}$$

Compute the error estimator at the k^{th} data point

$$e_k = \left| f(\mathbf{x}_k) - P_f^{[k]}(\mathbf{x}_k) \right| \tag{4}$$

end
Form the cost vector $\mathbf{e} = [e_1, \ldots, e_N]^T$
The optimal ε is given by minimizing $\|\mathbf{e}\|$

While this naive implementation of the leave-one-out algorithm would be rather expensive (on the order of N^4), Rippa showed that the algorithm can be simplified to a single formula by which

$$e_k = \frac{c_k}{\mathbf{A}^{-1}_{kk}}, \tag{5}$$

where c_k is the k^{th} coefficient in the expansion of the interpolant P_f based on the *full* data set, and $\mathbf{A}^{-1}_{kk}$ is the k^{th} diagonal element of the inverse of the corresponding interpolation matrix.

Note that only a single interpolant (on the entire data set) needs to be computed for this formulation. Thus, this results in $O(N^3)$ computational complexity. Moreover, all entries in the error vector $\mathbf{e}$ can be computed in a single statement in MATLAB provided we vectorize the component formula (5) (see line 4 in Program 2 below). In order to determine a good value of the shape parameter as quickly as possible we can use the Matlab function `fminbnd` to find the minimum of the cost function for ε.

A possible implementation of the cost function in the form of the subroutine `CostEps.m` is displayed in Program 2.

Program 2 `CostEps.m`

```
1  function ceps = CostEps(ep,rbf,DM,rhs)
2  A = rbf(ep,DM);
3  invA = pinv(A);
4  errorvector = (invA*rhs)./diag(invA);
5  ceps = norm(errorvector);
```

Here `rbf` needs to provide a MATLAB function that can generate the interpolation matrix `A` based on a shape parameter `ep` and a matrix `DM` of all the pairwise distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ among the datasites. For a Gaussian kernel the function `rbf` could look like

```
rbf = @(ep,r) exp(-(ep*r).^2);
```

A possible calling sequence for the cost function `CostEps` is given by

```
[ep,fval] = fminbnd(@(ep) CostEps(ep,rbf,DM,rhs),minep,maxep);
```

where `minep` and `maxep` define the interval to search in for the optimal ε value.

3 Iterated Approximate MLS Approximation

In [9] it was shown that iterated *approximate moving least squares* (AMLS) approximation yields the RBF interpolant in the limit. The algorithm can be interpreted as a variant of iterative refinement. The AMLS method is based on quasi-interpolants of the form

$$Q_f(\mathbf{x}) = \sum_{j=1}^{N} f_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|),$$

where $f_j = f(\mathbf{x}_j)$ are the given data values, and the generating function $\varphi(\|\cdot\|)$ is required to satisfy a number of conditions. In order to guarantee a desired order of approximation φ has to satisfy certain continuous moment conditions. The number of vanishing moments determines the precise approximation order of the scheme. Moreover, the generating function φ needs to satisfy a mild decay condition. If no connection to RBF interpolation is desired, then φ is not subject to any further restrictions. However, if this connection is desired, then we need to ensure further that φ generates the same function space as the RBF, i.e., the function is strictly positive definite. Finally, we can ensure convergence of the iterative algorithm by an appropriate scaling of the generating function. Details of the basic AMLS method are provided in [8] and the iterative method is described in [9]. A specific family of generating functions that satisfy all of the requirements just mentioned is listed in (11) below. We note that, in general, φ need not be a radial function. However, in this paper we concentrate on the radial case.

The iterative algorithm proceeds as follows:

Algorithm 3

Compute the initial approximation

$$Q_f^{(0)}(\mathbf{x}) = \sum_{j=1}^{N} f_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|)$$

For $n = 1, 2, \ldots$

For $j = 1, \ldots, N$

Compute residuals at the data sites

$$r_j^{(n)} = f_j - Q_f^{(n-1)}(\mathbf{x}_j)$$

end

Compute the correction

$$u(\mathbf{x}) = \sum_{j=1}^{N} r_j^{(n)} \varphi(\|\mathbf{x} - \mathbf{x}_j\|)$$

Update $Q_f^{(n)}(\mathbf{x}) = Q_f^{(n-1)}(\mathbf{x}) + u(\mathbf{x})$

end

As pointed out above, we expect the sequence of approximants $\left\{Q_f^{(n)}\right\}$ to converge to the RBF interpolant P_f. For the following discussion it will be convenient to introduce some more abbreviations:

$$P_{\mathbf{f}} = [P_f(\mathbf{x}_1), \ldots, P_f(\mathbf{x}_N)]^T, \tag{6}$$

$$Q_{\mathbf{f}}^{(n)} = [Q_f^{(n)}(\mathbf{x}_1), \ldots, Q_f^{(n)}(\mathbf{x}_N)]^T. \tag{7}$$

With this additional notation one can derive an *explicit* formula for the iterated AMLS approximant as

$$Q_f^{(n)}(\mathbf{x}) = \sum_{j=1}^{N} \left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \mathbf{f} \right]_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|) \tag{8}$$

(see [9] for details). By evaluating the approximant on the data sites we obtain the vectorized expression

$$Q_{\mathbf{f}}^{(n)} = \mathbf{A} \sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f} = \left(\mathbf{I} - (\mathbf{I} - \mathbf{A})^{n+1} \right) \mathbf{f}, \tag{9}$$

where the latter equality holds due to

$$\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i = \mathbf{A}^{-1} \left(\mathbf{I} - (\mathbf{I} - \mathbf{A})^{n+1} \right),$$

which in turn is a consequence of

$$\sum_{i=0}^{\infty} (\mathbf{I} - \mathbf{A})^i = \mathbf{A}^{-1}, \tag{10}$$

the standard Neumann series for the matrix inverse.

The sufficient scaling condition mentioned above that guarantees convergence of the iteratively computed AMLS approximant to the RBF interpolant is

$$\max_{i=1,2,\ldots,N} \left\{ \sum_{j=1}^{N} |\mathbf{B}_{ij}| \right\} < 2,$$

where the matrix $\mathbf{B}$ is a scaled version of $\mathbf{A}$, i.e.,

$$\mathbf{B}_{ij} = \varepsilon^s \varphi \left(\frac{\varepsilon}{h} \|\mathbf{x}_i - \mathbf{x}_j\| \right).$$

Here h is a data-dependent scale parameter which we take to be $h = 1/(N^{1/s} - 1)$, i.e., for uniformly spaced data in $\mathbb{R}^s$ h behaves just like the fill distance (see [9]).

One of the features of the iterative approximate MLS approximation algorithm is that it automatically adapts to non-uniformly spaced data since it converges to the RBF interpolant which is known to work reasonably well with non-uniform data. Without the residual iteration it is well known that approximate MLS approximation does not perform well on scattered data unless a complicated non-uniform scaling is performed for the generating function (see, e.g., [5]).

Finally, the most critical point for our application is to actually have generating functions that satisfy both the continuous moment conditions required for the approximate MLS method and are strictly positive definite — as required for RBF interpolation.

The *Laguerre-Gaussians* given by

$$\Phi(\mathbf{x}) = e^{-\|\mathbf{x}\|^2} L_n^{s/2}(\|\mathbf{x}\|^2), \qquad \mathbf{x} \in \mathbb{R}^s, \tag{11}$$

with generalized Laguerre polynomials $L_n^{s/2}$ are strictly positive definite on $\mathbb{R}^s$. Specific examples are:

$$\begin{aligned}
\Phi(\mathbf{x}) &= \frac{1}{\sqrt{\pi^s}} e^{-\|\mathbf{x}\|^2}, \qquad \mathbf{x} \in \mathbb{R}^s, \\
\Phi(\mathbf{x}) &= \frac{1}{\pi} \left(2 - \|\mathbf{x}\|^2 \right) e^{-\|\mathbf{x}\|^2}, \qquad \mathbf{x} \in \mathbb{R}^2, \\
\Phi(\mathbf{x}) &= \frac{1}{\pi} \left(3 - 3\|\mathbf{x}\|^2 - \frac{1}{2}\|\mathbf{x}\|^4 \right) e^{-\|\mathbf{x}\|^2}, \qquad \mathbf{x} \in \mathbb{R}^2.
\end{aligned}$$

These functions provide approximation order $O(h^2)$, $O(h^4)$, and $O(h^6)$, respectively (see, e.g., [4]).

4 LOOCV for the Optimization of Iterated AMLS

As explained in Section 2, Rippa originally designed the leave-one-out cross-validation algorithm to optimize the shape parameter for a standard RBF interpolation problem.

We now propose two LOOCV schemes that have been adapted to the situation that arises in iterated AMLS approximation. Our task now becomes to find not only a good value of the shape parameter ε, but also a good stopping criterion that results in an optimal number of iterations. For the latter to make sense it needs to be noted that for noisy data the iteration acts like a noise filter. However, after a certain number of iterations the noise will begin to feed on itself and the quality of the approximant will degrade.

4.1 Direct LOOCV for Iterated AMLS Approximation

Since the basic Rippa algorithm for LOOCV was designed to deal with the interpolation setting we now convert the iterated AMLS approximation to a similar formulation. From (9) we know that

$$\mathbf{A} \sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f} = Q_{\mathbf{f}}^{(n)}. \tag{12}$$

This formulation is similar to the linear system for an interpolation problem with system matrix $\mathbf{A}$, coefficient vector $\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f}$, and right-hand side $Q_{\mathbf{f}}^{(n)}$. In this formulation the right-hand side vector is obtained by evaluating the quasi-interpolant at the data sites instead of by the given data values themselves. If we multiply both sides of (12) by

$$\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \right]^{-1} \mathbf{A}^{-1}$$

then we obtain

$$\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \right]^{-1} \left(\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f} \right) = \mathbf{f}, \tag{13}$$

where the right-hand side of (13) is itself a consequence of (12). Note that (13) now is in the form of a standard interpolation system with system matrix $\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \right]^{-1}$, the same coefficient vector $\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f}$ as above, and the usual right-hand side $\mathbf{f}$ given by the data.

As a consequence of the Neumann series expansion (10) it turns out that the system matrix $\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \right]^{-1}$ of (13) is an approximation to the RBF interpolation matrix $\mathbf{A}$. More details of the connection between these two interpolation matrices as it pertains to smoothing and preconditioning will be discussed elsewhere.

In light of the reformulation of the iterative AMLS approximation described above one will expect that an LOOCV optimization of the system (13) will yield good parameter values for the iterated AMLS scheme. This means that in our current setting the Rippa formula (5) applied to compute an error vector for $Q_f^{(n)}$ is given by

$$e_k = \frac{\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \, \mathbf{f} \right]_k}{\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \right]_{kk}}. \tag{14}$$

Note that — in contrast to the RBF interpolation setting — in (14) we do not have to compute a matrix inverse. In fact, the numerator and denominator in (14) can be accumulated iteratively. Namely, if we let $\mathbf{v}^{(0)} = \mathbf{f}$ then the recursion

$$\mathbf{v}^{(n)} = \mathbf{f} + (\mathbf{I} - \mathbf{A})\,\mathbf{v}^{(n-1)}$$

will generate the coefficient vector whose k^{th} component appears in the numerator of (14).

Moreover, the complexity of the matrix powers in the denominator can be reduced by using an eigen-decomposition, i.e., we first compute

$$\mathbf{I} - \mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1},$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, and the columns of $\mathbf{X}$ are given by the eigenvectors of $\mathbf{I} - \mathbf{A}$. At this point it is worthwhile to note that not only the matrix $\mathbf{A}$ is symmetric and positive definite (by our assumptions on the generating functions $\varphi(\|\cdot\|)$), but the matrix $\mathbf{I} - \mathbf{A}$ has the same properties due to the scaling assumption required for convergence of the iterated AMLS method (see [9]).

Then, starting with $\mathbf{M}^{(0)} = \mathbf{I}$ we can iteratively generate the denominator of (14) via

$$\mathbf{M}^{(n)} = \mathbf{\Lambda}\mathbf{M}^{(n-1)} + \mathbf{I} \tag{15}$$

so that, for any fixed n,

$$\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i\right] = \mathbf{X}\mathbf{M}^{(n)}\mathbf{X}^{-1}.$$

Note that the matrix-matrix product in (15) involves only diagonal matrices. Moreover, the diagonal elements of $\mathbf{X}\mathbf{M}^{(n)}\mathbf{X}^{-1}$ can also be obtained without full matrix-matrix multiplications since $\mathbf{M}^{(n)}$ is diagonal.

We summarize everything we derived for the direct LOOCV strategy of iterated AMLS approximation in

Algorithm 4

Fix ε. Perform an eigen-decomposition

$$\mathbf{I} - \mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}$$

Initialize $\mathbf{v}^{(0)} = \mathbf{f}$ and $\mathbf{M}^{(0)} = \mathbf{I}$

For $n = 1, 2, \ldots$

Perform the updates

$$\mathbf{v}^{(n)} = (\mathbf{I} - \mathbf{A})\,\mathbf{v}^{(n-1)} + \mathbf{f}$$
$$\mathbf{M}^{(n)} = \mathbf{\Lambda}\mathbf{M}^{(n-1)} + \mathbf{I}$$

Compute the cost vector $\mathbf{e}^{(n)}$ as the componentwise quotient of $\mathbf{v}^{(n)}$ and the diagonal of $\mathbf{X}\mathbf{M}^{(n)}\mathbf{X}^{-1}$ (cf. line 4 of the MATLAB Program 2)

If $\left\|\mathbf{e}^{(n)}\right\| - \left\|\mathbf{e}^{(n-1)}\right\| < tol$

Stop the iteration

end

end

An optimal value of the shape parameter ε is given by minimizing the norm of the cost vector $\mathbf{e}^{(n)}$ with respect to ε. This can again be done using the MATLAB function `fminbnd`. Note that an optimal stopping value for n is automatically generated by the above algorithm.

4.2 Iterative LOOCV for Iterated AMLS Approximation

A second — albeit different — LOOCV strategy for the iterated AMLS algorithm is to follow the basic leave-one-out paradigm of Algorithm 1. Since the quasi-interpolation approach does not involve the solution of any linear systems it is conceivable to consider a straightforward implementation. This leads to

Algorithm 5

Fix ε
For $k = 1, \ldots, N$
Let

$$Q_f^{(0)[k]}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{f}_j^{[k]} \varphi\left(\|\mathbf{x} - \mathbf{x}_j^{[k]}\|\right)$$

For $n = 1, 2, \ldots$
For $j = 1, \ldots, N$
Compute residuals at the data sites

$$r_j^{(n)[k]} = \mathbf{f}_j^{[k]} - Q_f^{(n-1)[k]}(\mathbf{x}_j^{[k]})$$

end
Compute the correction

$$u(\mathbf{x}) = \sum_{j=1}^{N} r_j^{(n)[k]} \varphi(\|\mathbf{x} - \mathbf{x}_j^{[k]}\|)$$

Update $Q_f^{(n)[k]}(\mathbf{x}) = Q_f^{(n-1)[k]}(\mathbf{x}) + u(\mathbf{x})$
Compute the error estimate for the k^{th} data point

$$e_k^{(n)[k]} = \left| f(\mathbf{x}_k) - Q_f^{(n)[k]}(\mathbf{x}_k) \right| \tag{16}$$

end
Form the cost vector $\mathbf{e}^{(n)} = \left[e_1^{(n)[1]}, \ldots, e_N^{(n)[N]}\right]^T$
If $\left\|\mathbf{e}^{(n)}\right\| - \left\|\mathbf{e}^{(n-1)}\right\| < tol$
stop the iteration
end
end

As in the previous algorithm, an optimal ε is given by minimizing $\|\mathbf{e}^{(n)}\|$. Note that this algorithm essentially performs N copies of Algorithm 3 inside the leave-one-out loop over k.

In order to speed up this computation and simplify computer programming we now derive a simpler formulation for this iterative LOOCV process. Clearly, the residual computation (16) can actually be performed at all data points besides the k^{th} one. Thus, we extend the notation for residuals to

$$e_j^{(n)[k]} = \left| f(\mathbf{x}_j) - Q_f^{(n)[k]}(\mathbf{x}_j) \right|, \qquad j, k = 1, \ldots, N, \tag{17}$$

and then form a residual *matrix* $\mathbf{E}^{(n)}$ with entries $\mathbf{E}^{(n)}_{jk} = e^{(n)[k]}_j$. Consequently, the cost vector $\mathbf{e}^{(n)}$ actually lies along the diagonal of $\mathbf{E}^{(n)}$.

A component-wise examination of the relation between the iterated vectors and matrices reveals the following iterative procedure for computing the residual matrix $\mathbf{E}^{(n)}$ (whose derivation is less obvious but quite straightforward to verify).

Algorithm 6

Fix ε
Initialize $\mathbf{E}^{(0)}_{jk} = f(\mathbf{x}_j), \qquad j,k = 1,\ldots,N$
For $n = 1, 2, \ldots$
Let $\mathbf{D}^{(n-1)} = \mathrm{diag}(\mathbf{E}^{(n-1)})$
Update

$$\mathbf{E}^{(n)} = \mathbf{E}^{(n-1)} - \mathbf{A}\left(\mathbf{E}^{(n-1)} - \mathbf{D}^{(n-1)}\right)$$

end

This formulation is simple enough to program but the computation requires a matrix multiplication with the interpolation matrix $\mathbf{A}$ during each iteration. Fortunately, the matrix multiplication can be avoided since we need only the diagonal entries of the residual matrix $\mathbf{E}^{(n)}$. This gives rise to the final version of the iterative version of the LOOCV algorithm for the iterated AMLS method.

Algorithm 7

Fix ε. Perform an eigen-decomposition

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}$$

Initialize

$$\mathbf{S}^{(0)} = \mathbf{X}^{-1}\mathbf{E}^{(0)}$$
$$\mathbf{D}^{(0)} = \mathrm{diag}(\mathbf{E}^{(0)})$$

For $n = 1, 2, \ldots$
Update

$$\mathbf{S}^{(n)} = (\mathbf{I} - \mathbf{\Lambda})\,\mathbf{S}^{(n-1)} + \mathbf{\Lambda}\mathbf{X}^{-1}\mathbf{D}^{(n-1)}$$

$\mathbf{D}^{(n)} = \mathrm{diag}\left(\mathbf{X}\mathbf{S}^{(n)}\right)$
Set the cost vector

$$\mathbf{e}^{(n)} = \mathbf{D}^{(n)}$$

If $\left\|\mathbf{e}^{(n)}\right\| - \left\|\mathbf{e}^{(n-1)}\right\| < tol$
stop the iteration
end
end

An optimal ε is given by minimizing $\|\mathbf{e}^{(n)}\|$. Use this ε and the corresponding stopping n to construct an iterative AMLS approximant $Q^{(n)}_f$. We point out that only matrix scalings are performed for the update step, and in order to obtain the diagonal matrix $\mathbf{D}^{(n)}$ one does not need to perform the full matrix-matrix product $\mathbf{X}\mathbf{S}^{(n)}$.

5 Numerical Examples I

The iterative AMLS method is particularly well suited for the approximation of noisy data. In the numerical experiments summarized in Table 1 we sampled a standard test function (similar to Franke's function) at different sets of N randomly distributed Halton points in the unit square $[0,1]^2$. Then we artificially added 3% (uniformly distributed) random noise to the function values in order to obtain our simulated noisy data. Gaussian generating functions were used for all of the computations.

In addition to the two LOOCV algorithms discussed above for iterated approximate MLS approximation we also include results for analogous algorithms applied to an iterated Shepard approximation scheme. In contrast to the approximate MLS methods, Shepard's method is a true MLS approximation method which preserves constants, i.e., if (non-noisy) data has been sampled from a constant function, then Shepard's method will reconstruct that function. While approximate MLS methods form only an approximate partition of unity, Shepard's method is exact.

For noisy data it does not make sense to apply standard RBF interpolation and therefore this method is not used here. Instead we use a ridge regression (or smoothing spline) approximation. In the ridge regression approach the coefficients $\mathbf{c}$ of the RBF expansion (2) are determined by minimizing the functional

$$\sum_{j=1}^{N} \left[P_f(\mathbf{x}_j) - f_j\right]^2 + \omega \mathbf{c}^T \mathbf{A} \mathbf{c}.$$

Here the first term measures the goodness of fit, and the second term is a smoothness measure (also known as the *native space norm* of the RBF interpolant). The smoothing (or regression) parameter $\omega \geq 0$ balances the trade-off between these two quantities. A zero value of ω corresponds to the interpolation setting, while a larger value will allow a smoothing effect. It is well known that the coefficients can be found by solving a regularized interpolation system, i.e.,

$$(\mathbf{A} + \omega \mathbf{I})\, \mathbf{c} = \mathbf{f}.$$

This method (and optimal choice of its parameters) was discussed in [10]. In that paper only the direct LOOCV strategy of Algorithm 4 was applied to the iterated AMLS method. We include those results in Table 1 for comparison with the iterative LOOCV strategy of Algorithm 7.

We can observe similar approximation errors for all methods. Also, the two different variants of LOOCV result in similar "optimal" values of the shape parameter ε and similar execution times for both the iterated AMLS and iterated Shepard methods. However, use of the iterated LOOCV strategy seems to lead to fewer (albeit costlier) iterations. Clearly, the ridge regression approach requires considerably more time than any of the iterative methods. This is easily explained since the iterative methods do not require solution of any linear systems.

6 The RBF-PS Approach to PDEs

The second situation in which we will determine an optimal value of the RBF shape parameter ε is within a pseudo-spectral (PS) approach to the solution of partial differential equations. Before we explain how to adapt the LOOCV method for this application we first give a brief introduction to the RBF-PS method.

Table 1 Performance of different LOOCV algorithms for the approximation of noisy data.

N		9	25	81	289	1089
AMLS Direct LOOCV	RMSerr	4.80e-3	1.53e-3	6.42e-4	4.39e-4	2.48e-4
	ε	1.479865	1.268158	0.911517	0.652600	0.46741
	no. iter.	7	6	6	4	3
	time	0.2	0.4	1.0	5.7	254
AMLS Iterative LOOCV	RMSerr	4.57e-3	1.63e-3	6.77e-4	4.22e-4	2.30e-4
	ε	1.482765	1.229673	0.871514	0.593223	0.383817
	no. iter.	7	3	3	2	2
	time	0.25	0.25	0.84	7.54	238.0
Shepard Direct LOOCV	RMSerr	5.65e-3	1.96e-3	8.21e-4	5.10e-4	2.73e-4
	ε	2.194212	1.338775	0.895198	0.656266	0.468866
	no. iter.	7	7	7	5	3
	time	0.2	0.4	2.1	7.0	225
Shepard Iterative LOOCV	RMSerr	5.51e-3	2.01e-3	8.15e-4	5.07e-4	2.77e-4
	ε	2.077628	1.314021	0.910505	0.634368	0.409473
	no. iter.	5	4	4	2	2
	time	0.39	0.36	0.86	7.75	332.9
Ridge	RMSerr	3.54e-3	1.62e-3	7.20e-4	4.57e-4	2.50e-4
	ε	2.083918	0.930143	0.704802	0.382683	0.181895
	ω	0.010000	0.010000	0.021131	0.037574	0.033610
	time	0.3	1.2	1.1	21.3	672

Polynomial pseudo-spectral methods are well known as highly accurate solvers for PDEs (see, e.g., [14,31]). We now describe a generalization that involves multivariate radial basis functions instead of univariate polynomials. Our approach does indeed generalize polynomials since a number of authors have shown recently (see, e.g., [1,2,29, 30]) that in the limiting case of "flat" basis functions, i.e., $\varepsilon \to 0$, the one-dimensional RBF interpolant yields a polynomial interpolant.

The basic idea behind any pseudo-spectral method is to expand the solution u of the partial differential equation in terms of smooth global basis functions, i.e.,

$$u(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi_j(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^s. \tag{18}$$

The smoothness and global support of the basis functions ϕ_j, $j = 1, \ldots, N$, are the main properties that ensure the spectral approximation order of the method. While the basis functions ϕ_j could be rather general we will use radial basis functions, i.e., $\phi_j = \varphi(\| \cdot - \mathbf{x}_j \|)$.

In the numerical examples below we will be presenting both elliptic and parabolic PDEs, but for the present discussion we can limit ourselves to the spatial part of the solution. Time dependence will be dealt with in the standard way.

Expansion (18) is exactly the same as that used for the well-known non-symmetric RBF collocation method or Kansa's method [20]. However, as the following discussion shows, Kansa's method and the RBF-PS approach are not identical. The difference is that for Kansa's one explicitly computes the expansion coefficients λ_j and then subsequently is able to evaluate the approximate PDE solution at an arbitrary point $\mathbf{x}$. In the RBF-PS approach one obtains values of the approximate solution at the

collocation points only, and the coefficients λ_j are never explicitly computed. Instead one works with a so-called differentiation matrix. This slightly different view of RBF collocation has been adopted recently by a number of authors (see, e.g., [6,7,26,28]). Details of this approach are now presented.

There are many similarities between the RBF-PS method we are about to describe and the RBF interpolation method reviewed at the beginning of this article. However, for PDEs we need to be able to represent values of derivatives of u. Therefore we require not only the expansion (18), but also

$$Lu(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j L\phi_j(\mathbf{x}), \tag{19}$$

which follows directly from (18) provided L is assumed to be a linear differential operator.

An important difference between the RBF-PS method and standard RBF interpolation is that for the PS approach we limit evaluation to the collocation points only. Thus, instead of attempting to obtain the function u defined in (18) for arbitrary $\mathbf{x}$-values we compute only the vector $\mathbf{u} = [u(\mathbf{x}_1), \ldots, u(\mathbf{x}_N)]^T$ of function values at the collocation points $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

Now, the key idea is to relate this vector of function values to a vector of "derivative" values $\mathbf{u}_L$ using a *differentiation matrix* $\mathbf{D}$, i.e.,

$$\mathbf{u}_L = \mathbf{D}\mathbf{u}. \tag{20}$$

In order to derive an expression for the differentiation matrix we first evaluate (18) at the collocation points. That yields

$$\mathbf{u} = \mathbf{A}\boldsymbol{\lambda}, \tag{21}$$

where the matrix $\mathbf{A}$ has entries $\mathbf{A}_{ij} = \phi_j(\mathbf{x}_i)$. This means that $\mathbf{A}$ is just an RBF interpolation matrix as used earlier, and use of the same notation is justified. Similarly, evaluation of (19) at the collocation points results in

$$\mathbf{u}_L = \mathbf{A}_L\boldsymbol{\lambda}, \tag{22}$$

where $\mathbf{A}_{L,ij} = L\phi_j(\mathbf{x}_i)$. We can solve (21) for $\boldsymbol{\lambda}$ (since the interpolation matrix $\mathbf{A}$ is known to be invertible) and then insert this into (22) to arrive at

$$\mathbf{u}_L = \mathbf{A}_L\mathbf{A}^{-1}\mathbf{u}.$$

Comparison of this expression with (20) yields the differentiation matrix in the form

$$\mathbf{D} = \mathbf{A}_L\mathbf{A}^{-1}.$$

Note that up to this point the discussion has been rather generic. No specific properties of the differential operator were required other than linearity (and as we will see in the examples in the next section nonlinear PDEs can also be solved by this approach). More importantly, we have up to now completely ignored the boundary conditions of the problem. Dirichlet conditions can be incorporated in a completely trivial manner since such a condition at a boundary collocation point corresponds to having a row of the identity matrix in $\mathbf{D}$. If the prescribed function value is even zero, then one can just delete the appropriate row and column from $\mathbf{D}$ (for more

details see, e.g., [8] or [31]). The implementation of other boundary conditions can be accomplished in a similar manner, but does require a little more care. We denote the differentiation matrix that incorporates the boundary conditions by $\mathbf{D}_\Gamma$, where Γ denotes the boundary of the domain Ω.

In the literature on RBF collocation methods one frequently encounters two different approaches: Kansa's non-symmetric method (see, e.g., [20]), and a symmetric method based on Hermite interpolation (see, e.g., [3]). As mentioned above, the present RBF-PS method has many similarities with the non-symmetric Kansa method. However, here the coefficient vector is never explicitly determined and one is not interested in the function u representing the solution of the PDE — only its function values $\mathbf{u}$ at the collocation points. Also, boundary conditions are handled differently in Kansa's method and in the RBF-PS approach.

The precise connection between the RBF-PS approach and the standard RBF collocation methods is discussed in [8]. It is shown there that for the non-symmetric RBF-PS method with Dirichlet boundary conditions we get

$$\mathbf{D}_\Gamma = \begin{bmatrix} \tilde{\mathbf{A}}_L \\ \tilde{\mathbf{A}} \end{bmatrix} \mathbf{A}^{-1},$$

where the block matrix that arises here is exactly the well-known collocation matrix from Kansa's method. As a consequence of this we cannot ensure general invertibility of the differentiation matrix $\mathbf{D}_\Gamma$ for the RBF-PS approach (cf. [19]).

On the other hand, it is possible to formulate an RBF-PS approach that is analogous to the symmetric RBF collocation method. In that case, however, the differentiation matrix (for a Dirichlet problem) would be of the form

$$\hat{\mathbf{D}}_\Gamma = \begin{bmatrix} \hat{\mathbf{A}}_{LL^*} & \hat{\mathbf{A}}_L \\ \hat{\mathbf{A}}_{L^*} & \hat{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{L^*} & \tilde{\mathbf{A}}^T \end{bmatrix}^{-1},$$

where the first block matrix is the collocation matrix that arises in the symmetric RBF collocation method, and the second block matrix is the transpose of the Kansa matrix (see [8] for details). As a consequence of this we know (from the standard RBF collocation approach) that the elliptic PDE $Lu = f$ can be solved. However, *formulation* of the differentiation matrix $\hat{\mathbf{D}}_\Gamma$ cannot be justified since it contains as one of its factors the inverse of the (transpose of the) Kansa matrix.

In summary, for elliptic PDEs one should use the symmetric collocation method, and for parabolic problems the (non-symmetric) RBF-PS approach outlined at the beginning of this section. However, since configurations of collocation points that lead to singular matrices are rare (cf. [19]) and since one can employ QR or SVD techniques to deal even with these situations (see, e.g., [24]), most people will prefer to use the simpler non-symmetric approach for both time-dependent and time-independent problems. We will follow this general trend with our numerical examples below.

7 Selecting a Good Shape Parameter for the RBF-PS Method

Due to the similarity of the RBF-PS method with the standard RBF interpolation problem we will be able to adapt Rippa's LOOCV algorithm [27] for determining an optimal shape parameter with only some minor modifications.

In the RBF interpolation setting the LOOCV algorithm targets solution of the underlying linear system $\mathbf{Ac} = \mathbf{f}$, where the entries of the matrix $\mathbf{A}$ depend on the RBF shape parameter ε which we seek to optimize. As stated in (5), the components of the cost vector are given by

$$e_k = \frac{c_k}{\mathbf{A}_{kk}^{-1}}.$$

According to the previous section the RBF-PS differentiation matrix is given by

$$\mathbf{D} = \mathbf{A}_L \mathbf{A}^{-1}.$$

Equivalently, we can write this as

$$\mathbf{A}\mathbf{D}^T = (\mathbf{A}_L)^T, \tag{23}$$

where we have taken advantage of the symmetry of the (RBF interpolation) matrix $\mathbf{A}$. We will use (23) as a basis for our LOOCV algorithm. The structure of this formula is just as that of the standard RBF interpolation problem discussed in [27]. Now, however, we are dealing with multiple systems of the form $A\mathbf{c} = \mathbf{f}$ having a common system matrix $\mathbf{A}$.

Therefore, the components of our cost *matrix* are given by

$$\mathbf{E}_{k\ell} = \frac{(\mathbf{D}^T)_{k\ell}}{\mathbf{A}_{kk}^{-1}}.$$

In MATLAB this can again be vectorized, so that we end up with a program very similar to Program 2.

Program 8 `CostEpsLRBF.m`

```
function ceps = CostEpsLRBF(ep,DM,rbf,Lrbf)
n = size(DM,2);
A = rbf(ep,DM);
rhs = Lrbf(ep,DM)';
invA = pinv(A);
errormatrix = (invA*rhs)./repmat(diag(invA),1,n);
ceps = norm(errormatrix(:));
```

The function `Lrbf` creates the matrix $\mathbf{A}_L$. For the Gaussian RBF and the Laplacian differential operator this could look like

```
Lrbf = @(ep,r) 4*ep^2*exp(-(ep*r).^2).*((ep*r).^2-1);
```

Note that for differential operators of odd order one will also have to provide a matrix of pairwise *differences*. For example, the partial derivative with respect to the x-coordinate for the Gaussian would be coded as

```
Lrbf = @(ep,r,dx) -2*dx*ep^2.*exp(-(ep*r).^2);
```

Here `dx` is the matrix containing the pairwise differences in the x-coordinates of the collocation points.

8 Numerical Examples II

Example 9 Our first example deals with the following Laplace equation

$$u_{xx} + u_{yy} = 0, \quad x, y \in (-1, 1)^2,$$

with piecewise defined boundary conditions

$$u(x,y) = \begin{cases} \sin^4(\pi x), & y = 1 \text{ and } -1 < x < 0, \\ \frac{1}{5}\sin(3\pi y), & x = 1, \\ 0, & \text{otherwise.} \end{cases}$$

This is the same problem as used in Program 36 of [31]. In our implementation we use the MATLAB code of [31] and simply replace the call to the subroutine `cheb` in that program with equivalent code that generates the RBF-PS differentiation matrices `D` and `D2`[1]. Implementation details are provided in [8].

We use the "cubic" Matérn RBF $\varphi(r) = (15 + 15\varepsilon r + 6(\varepsilon r)^2 + (\varepsilon r)^3)e^{-\varepsilon r}$ whose optimal shape parameter was determined to be $\varepsilon = 0.362752$ by the LOOCV algorithm. The spatial discretization consists of a tensor product of 25×25 Chebyshev points.

Figure 1 shows the solution obtained via the RBF-PS and Chebyshev pseudospectral methods, respectively. The qualitative behavior of the two solutions is very similar.

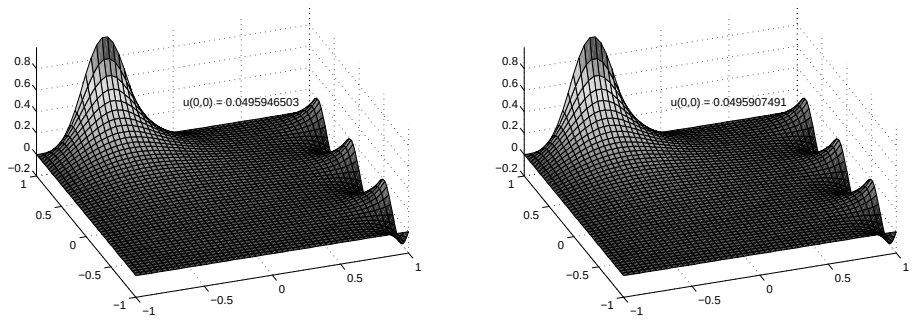

Fig. 1 Solution of the Laplace equation using a Chebyshev PS approach (left) and cubic Matérn RBFs with $\varepsilon = 0.362752$ (right) with 625 collocation points.

Note that for this type of elliptic problem we require inversion of the differentiation matrix. As pointed out at the end of the previous section we use the non-symmetric RBF-PS method even though this may not be warranted theoretically.

Example 10 As our second example we consider the 2-D Helmholtz equation (see Program 17 in [31])

$$u_{xx} + u_{yy} + k^2 u = f(x,y), \quad x, y \in (-1, 1)^2,$$

[1] For the RBF-PS approach this second derivative matrix has to be generated separately since, contrary to the polynomial case, we in general do not get the second derivative matrix as the square of the first derivative matrix (for details see [8])

with boundary condition $u = 0$ and

$$f(x, y) = \exp\left(-10\left[(y-1)^2 + (x - \frac{1}{2})^2\right]\right).$$

The solution of the Helmholtz equation for $k = 9$ with Gaussians using $\varepsilon = 2.549243$ and 625 collocation points placed on a Chebyshev tensor-product grid is displayed next to the Chebyshev pseudospectral solution of [31] in Figure 2.

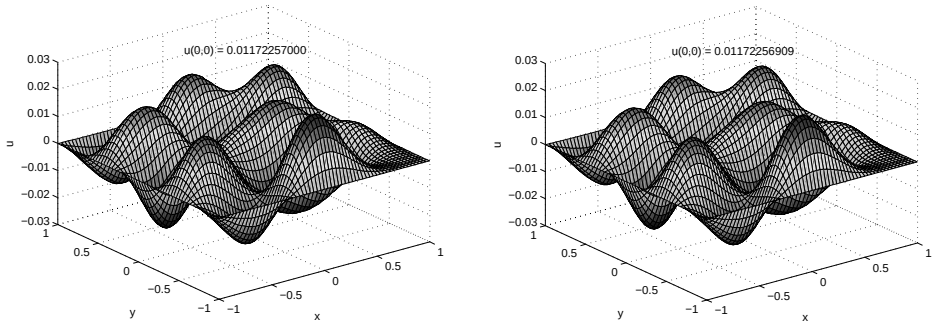

Fig. 2 Solution of 2-D Helmholtz equation with 625 collocation points using the Chebyshev pseudospectral method (left) and Gaussians with $\varepsilon = 2.549243$ (right).

Example 11 Our final example is the most challenging for the RBF-PS method. This example is based on Program 35 in [31] which is concerned with the solution of the nonlinear reaction-diffusion (or Allen-Cahn) equation. As mentioned earlier, this example shows that the RBF-PS method can be applied in a straightforward manner also to nonlinear problems by incorporating the nonlinearity into the time-stepping method (for details see either the original code in [31] or the RBF-PS modification in [8]). The PDE is of the form

$$u_t = \mu u_{xx} + u - u^3, \qquad x \in (-1, 1),\ t \geq 0,$$

with parameter μ, initial condition

$$u(x, 0) = 0.53x + 0.47 \sin\left(-\frac{3}{2}\pi x\right), \qquad x \in [-1, 1],$$

and non-homogeneous (time-dependent) boundary conditions

$$u(-1, t) = -1 \text{ and } u(1, t) = \sin^2(t/5).$$

The solution to this equation has three steady states ($u = -1, 0, 1$) with the two nonzero solutions being stable. The transition between these states is governed by the parameter μ. For our example displayed in Figure 3 we use $\mu = 0.01$, and the unstable state should vanish around $t = 30$.

The two plots in Figure 3 show the solution obtained via the Chebyshev pseudospectral method and via an RBF-PS approach based on the same "cubic" Matérn

functions used above. The spatial discretization is given by a (one-dimensional) grid of 21 Chebyshev points. This time the optimal shape parameter determined by the LOOCV algorithm is $\varepsilon = 0.350920$. We can see from the figure that the solution based on Chebyshev polynomials appears to be slightly more accurate since the transition occurs at a slightly later and correct time (i.e., at $t \approx 30$) and is also a little "sharper".

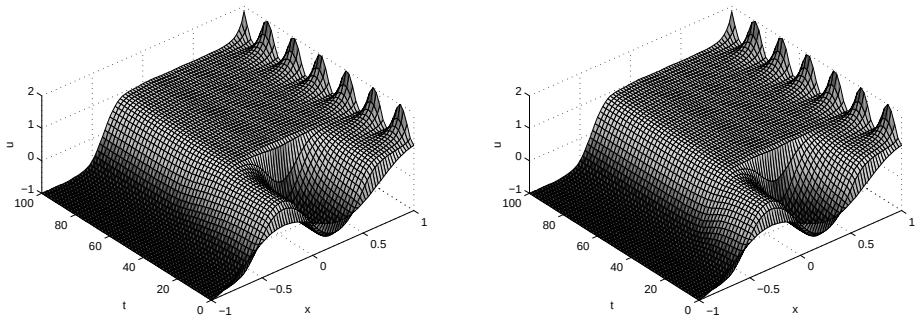

Fig. 3 Solution of the Allen-Cahn equation using the Chebyshev pseudospectral method (left) and a cubic Matérn functions (right) with 21 Chebyshev points.

9 Remarks and Conclusions

Two applications of leave-one-out cross validation for the determination of RBF shape parameters were discussed in this paper: iterated approximate moving least squares approximation, and an RBF pseudo-spectral method.

For the iterated AMLS method we suggested two different LOOCV algorithms: one based on an explicit formula for the iterated approximant (Algorithm 4), the other directly based on the iterative formulation of the approximant (Algorithm 7). We showed that both algorithms yield similar results in terms of accuracy and efficiency for reconstruction of noisy function data. Moreover, the execution time of our iterative algorithms performed favorably when compared to an RBF ridge regression (or smoothing spline) algorithm. The fundamental difference between our iterative algorithms and the more traditional smoothing spline approach is that we work with quasi-interpolants, and thus are able to avoid the solution of dense linear systems.

In the second part of the paper we showed how the RBF-PS method on the one hand generalizes the traditional polynomial pseudo-spectral approach, and on the other hand provides a standard framework in which to discuss as well as implement RBF collocation solutions of PDEs. It was pointed out that the RBF-PS approach is more efficient than Kansa's method for time-dependent PDEs. The shape parameter problem was addressed in this context, also, by providing an adaption of Rippa's LOOCV algorithm (see Program 8).

One important point that has not yet been addressed by our work is the computational cost of the RBF-PS method as compared to implementations of the traditional

polynomial PS method. At this point the RBF-based approach is clearly more expensive since it requires a matrix inversion to determine the differentiation matrix, and for polynomial PS methods the entries of the differentiation matrix are known explicitly (see, e.g., [31]). On the other hand, differentiation methods do not give rise to the most efficient implementation of PS methods anyway. Thus, the search for an efficient implementation of the RBF-PS method in the vain of FFT implementations for polynomial PS methods is still a wide open research area.

The RBF-PS method has also been applied successfully to a number of engineering problems (see, e.g., [11,12]). Of course, as mentioned earlier, Kansa's method has been popular since its introduction in [20] and countless papers exist in the literature dealing with all kinds of applications in science and engineering. An LOOCV algorithm for finding an "optimal" shape parameter similar to the one described in this paper was used with Kansa's method in [13].

Clearly, there is still much to be done regarding the optimal choice of RBF shape parameters. For example, many researchers have suggested the use of variable shape parameters (e.g., [21]). On the one hand, following this suggestion provides a clear potential for improved accuracy and stability of the RBF method. On the other hand, the use of variable shape parameters raises some challenging theoretical problems that have up to now remained unsolved. The choice of optimal RBF shape parameters that vary spatially with the centers of the basis functions was studied in the recent paper [16]. We are currently also working on this problem in connection with an SVD stabilization of the RBF interpolant. Another interesting application presently being investigated by us is the use of the iterated AMLS algorithm as a preconditioner for the standard RBF interpolant.

References

1. de Boor, C. On interpolation by radial polynomials. *Adv. Comput. Math.* **24** (2006), 143–153.
2. Driscoll, T. A. and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Comput. Math. Appl.* **43** (2002), 413–422.
3. Fasshauer, G. E. Solving partial differential equations by collocation with radial basis functions, in *Surface Fitting and Multiresolution Methods*. A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.). Vanderbilt University Press, Nashville, TN (1997), 131–138.
4. Fasshauer, G. E. Approximate moving least-squares approximation: A fast and accurate multivariate approximation method, in *Curve and Surface Fitting: Saint-Malo 2002*. A. Cohen, J.-L. Merrien, and L. L. Schumaker (eds.). Nashboro Press, Nashville (2003), 139–148.
5. Fasshauer, G. E. Toward approximate moving least squares approximation with irregularly spaced centers. *Computer Methods in Applied Mechanics & Engineering* **193** (2004), 1231–1243.
6. Fasshauer, G. E. RBF collocation methods and pseudospectral methods. Technical report, Illinois Institute of Technology, 2004.
7. Fasshauer, G. E. RBF collocation methods as pseudospectral methods, in *Boundary Elements XXVII*, A. Kassab, C. A. Brebbia, E. Divo, and D. Poljak (eds.). WIT Press, Southampton (2005), 47–56.
8. Fasshauer, G. E. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishers, Singapore, to appear.
9. Fasshauer, G. E. and J. G. Zhang. Iterated approximate moving least squares approximation. Proceedings of Meshless Methods Lisbon 2005, Springer, in press.
10. Fasshauer, G. E. and J. G. Zhang. Scattered data approximation of noisy data via iterated moving least squares. Proceedings of Curves and Surfaces Avignon 2006, Nashboro Press, in press.

11. Ferreira, A. J. M. and G. E. Fasshauer. Computation of natural frequencies of shear deformable beams and plates by an RBF-pseudospectral method. *Comput. Methods Appl. Mech. Engrg.* **196** (2006), 134–146.
12. Ferreira, A. J. M. and G. E. Fasshauer. Analysis of natural frequencies of composite plates by an RBF-pseudospectral method. *Composite Structures*, to appear.
13. Ferreira, A. J. M., G. E. Fasshauer, C. M. C. Roque, R. M. N. Jorge and R. C. Batra. Analysis of functionally graded plates by a robust meshless method. *J. Mech. Adv. Mater. & Struct.*, to appear.
14. Fornberg, B. *A Practical Guide to Pseudospectral Methods*. Cambridge Univ. Press (1998).
15. Fornberg, B. and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Comput. Math. Appl.* **47** (2004), 497–523.
16. Fornberg, B. and J. Zuev. The Runge phenomenon and spatially variable shape parameters in RBF interpolation. *Comput. Math. Appl.*, to appear.
17. Franke, R. Scattered data interpolation: tests of some methods. *Math. Comp.* **48** (1982), 181–200.
18. Hardy, R. L. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.* **76** (1971), 1905–1915.
19. Hon, Y. C. and R. Schaback. On nonsymmetric collocation by radial basis functions. *Appl. Math. Comput.* **119** (2001), 177–186.
20. Kansa, E. J. Multiquadrics — A scattered data approximation scheme with applications to computational fluid-dynamics — II: Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Applic.* **19** (1990), 147–161.
21. Kansa, E. J. and R. E. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Comput. Math. Applic.* **24** (1992), 99–120.
22. Larsson, E. and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic PDEs. *Comput. Math. Appl.* **46** (2003), 891–902.
23. Larsson, E. and B. Fornberg. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Comput. Math. Appl.* **49** (2005), 103–130.
24. Ling, L., R. Opfer and R. Schaback. Results on meshless collocation techniques. *Engineering Analysis with Boundary Elements* **30** (2006), 247–253.
25. Maz'ya, V. and G. Schmidt. On quasi-interpolation with non-uniformly distributed centers on domains and manifolds. *J. Approx. Theory* **110** (2001), 125–145.
26. Platte, R. B. and T. A. Driscoll. Eigenvalue stability of radial basis function discretizations for time-dependent problems. *Comput. Math. Appl.* **51** 8 (2006), 1251–1268.
27. Rippa, S. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Adv. Comput. Math.* **11** (1999), 193–210.
28. Sarra, S. A. Adaptive radial basis function methods for time dependent partial differential equations. *Appl. Numer. Math.* **54** (2005), 79–94.
29. Schaback, R. Multivariate interpolation by polynomials and radial basis functions. *Const. Approx.* **21** (2005), 293–317.
30. Schaback, R. Limit problems for interpolation by analytic radial basis functions. *J. Comput. Appl. Math.*, to appear.
31. Trefethen, L. N. *Spectral Methods in MATLAB*. SIAM, Philadelphia, PA (2000).
32. Wendland, H. *Scattered Data Approximation*. Cambridge University Press, Cambridge (2005).