Chapter 8

Some Issues Related to Practical Implementations

In this chapter we will collect some information about issues that are important for the practical use of radial basis function and moving least squares methods. These issues include stability and conditioning of radial basis function interpolants, the *trade-off principle* which explains the trade-off between achievable convergence rates and numerical stability or efficiency, as well as algorithms for fast solution and evaluation of radial basis interpolants and moving least squares approximants.

8.1 Stability and Conditioning of Radial Basis Function Interpolants

A standard criterion to measure the numerical stability of an approximation method is its condition number. In particular, for radial basis function interpolation we need to look at the condition number of the interpolation matrix A with entries $A_{ij} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$. For any matrix A the ℓ_2-condition number of A is given by

$$\operatorname{cond}(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of A. If we concentrate on positive definite matrices A, then we can also take the ratio

$$\frac{\lambda_{\max}}{\lambda_{\min}}$$

of largest and smallest eigenvalues as an indicator for the condition number of A.

What do we know about these eigenvalues? First, Gershgorin's Theorem says that

$$|\lambda_{\max} - A_{ii}| \le \sum_{\substack{j=1 \\ j\neq i}}^{N} |A_{ij}|.$$

Therefore,

$$\lambda_{\max} \le N \max_{i,j=1,\ldots,N} |A_{ij}| = N \max_{\boldsymbol{x}_i,\boldsymbol{x}_j \in \mathcal{X}} \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j),$$

which, since Φ is strictly positive definite, becomes

$$\lambda_{\max} \le N\Phi(\mathbf{0})$$

by the properties of positive definite functions listed in Theorem 1.2.6. Now, as long as the data are not too wildly distributed, N will grow as $h_{\mathcal{X},\Omega}^{-s}$ which is acceptable. Therefore, the main work in establishing a bound for the condition number of A lies in finding lower bounds for $\lambda_{\min}$ (or correspondingly upper bounds for the norm of the inverse $\|A^{-1}\|_2$). This is the subject of several papers by Ball, Narcowich, Sivakumar and Ward [19, 479, 481, 482, 483] who make use of a result by Ball [18] on eigenvalues of distance matrices. Ball's result follows from the Rayleigh quotient, which gives the smallest eigenvalue of a positive definite matrix as

$$\lambda_{\min} = \inf_{\boldsymbol{c}\in\mathbb{R}^N\setminus\mathbf{0}} \frac{\boldsymbol{c}^T A\boldsymbol{c}}{\boldsymbol{c}^T\boldsymbol{c}}.$$

This leads to the following bound for the norm of the inverse of A.

Lemma 8.1.1 *Let $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$, be distinct points in $\mathbb{R}^s$ and let $\Phi:\mathbb{R}^s\to\mathbb{R}$ be either strictly positive definite or strictly conditionally negative definite of order one with $\Phi(\mathbf{0})\le 0$. Also, let A be the interpolation matrix with entries $A_{ij}=\Phi(\boldsymbol{x}_i-\boldsymbol{x}_j)$. If the inequality*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j A_{ij} \ge \theta\|\boldsymbol{c}\|_2^2$$

is satisfied whenever the components of $\mathbf{c}$ satisfy $\sum_{j=1}^N c_j = 0$, then

$$\|A^{-1}\|_2 \le \theta^{-1}.$$

Note that for positive definite matrices the Rayleigh quotient implies $\theta=\lambda_{\min}$ which shows why lower bounds on the smallest eigenvalue correspond to to upper bounds on the norm of the inverse of A. In order to obtain the bound for conditionally negative matrices the Courant-Fischer Theorem 3.1.2 needs to be employed.

Narcowich and Ward establish bounds on the norm of the inverse of A in terms of the *separation distance* of the data sites

$$q_{\mathcal{X}} = \frac{1}{2}\min_{i\ne j}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2.$$

We can picture $q_{\mathcal{X}}$ as the radius of the largest ball that can be placed around every point in $\mathcal{X}$ such that no two balls overlap (see Figure 8.1).

The derivation of these bounds is rather technical, and for details we refer to either the original papers by Narcowich, Ward and co-workers, the more recent paper [557] by Schaback (who uses a slightly simpler strategy), or Wendland's book [634]. We now list several bounds as derived in [634].

Examples: In the examples below the explicit constants

$$M_s = 12\left(\frac{\pi\Gamma^2(\frac{s+2}{2})}{9}\right)^{1/(s+1)} \le 6.38s \quad\text{and}\quad C_s = \frac{1}{2\Gamma(\frac{s+2}{2})}\left(\frac{M_s}{\sqrt{8}}\right)^s$$

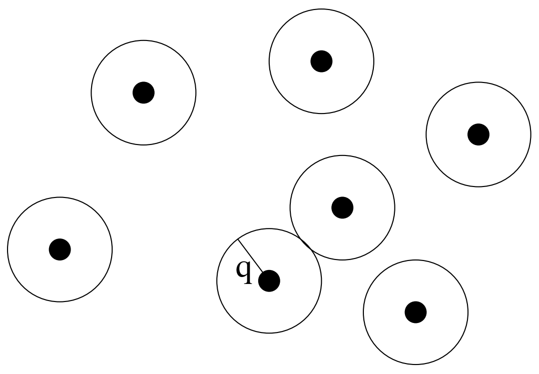

Figure 8.1: The separation distance $q_{\mathcal{X}}$ for a set of data sites in $\mathbb{R}^2$.

are used. The upper bound for M_s can be obtained using Stirling's formula (see, e.g., [634]).

1. For Gaussians $\Phi(\boldsymbol{x}) = e^{-\alpha\|\boldsymbol{x}\|^2}$ one obtains
$$\lambda_{\min} \geq C_s(2\alpha)^{-s/2} e^{-40.71 s^2/(q_{\mathcal{X}}^2\alpha)} q_{\mathcal{X}}^{-s}.$$
2. For (inverse) multiquadrics $\Phi(\boldsymbol{x}) = \left(\|\boldsymbol{x}\|^2 + \alpha^2\right)^\beta$, $\beta \in \mathbb{R} \setminus \mathbb{N}_0$ one obtains
$$\lambda_{\min} \geq C(\alpha, \beta, s) q_{\mathcal{X}}^{\beta - \frac{s}{2} + \frac{1}{2}} e^{-2\alpha M_s / q_{\mathcal{X}}}$$
with another explicitly known constant $C(\alpha, \beta, s)$.
3. For thin plate splines $\Phi(\boldsymbol{x}) = (-1)^{k+1}\|\boldsymbol{x}\|^{2k} \log \|\boldsymbol{x}\|$, $k \in \mathbb{N}$, one obtains
$$\lambda_{\min} \geq C_s c_k (2M_s)^{-s-2k} q_{\mathcal{X}}^{2k}$$
with another explicitly known constant c_k.
4. For the powers $\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta/2 \rceil}\|\boldsymbol{x}\|^\beta$, $\beta > 0$, $\beta \notin 2\,\mathbb{N}$, one obtains
$$\lambda_{\min} \geq C_s c_\beta (2M_s)^{-s-\beta} q_{\mathcal{X}}^{\beta}$$
with another explicitly known constant c_β.
5. For the compactly supported functions $\Phi_{s,k}(\boldsymbol{x}) = \varphi_{s,k}(\|\boldsymbol{x}\|)$ of Section 4 one obtains
$$\lambda_{\min} \geq C(s,k) q_{\mathcal{X}}^{2k+1}$$
with a constant $C(s,k)$ depending on s and k.

By providing matching *lower* bounds for $\|A^{-1}\|_2$ Schaback [547] showed that the upper bounds on the norm of the inverse obtained earlier by Narcowich, Ward and others are near optimal.

For the infinitely smooth functions of Examples 1 and 2 we see that, for a fixed shape parameter α, the lower bound for $\lambda_{\min}$ goes exponentially to zero, and therefore the condition number of the interpolation matrix A grows exponentially, as the separation distance $q_{\mathcal{X}}$ decreases. This shows that, if one adds more interpolation points in order to improve the accuracy of the interpolant (within the same domain Ω), then the problem becomes increasingly ill-conditioned. Of course one would always expect this to happen, but here the ill-conditioning grows primarily due to the decrease in the separation distance $q_{\mathcal{X}}$, and not to the increase in the number N of data points. We will come back to this observation when we discuss a possible change of basis in Section 8.4.

On the other hand, if one keeps the number of points (or at least the separation distance) fixed and instead increases (reduces) the value of α for Gaussians (multiquadrics), then the condition number of A is improved. This corresponds to the *stationary* approximation setting (which we did not discuss in detail earlier). In this case it is possible to show that the upper bound for the error estimate increases, i.e., the accuracy of the interpolant deteriorates. Conversely, one can attempt to improve the accuracy of a radial basis function interpolant by decreasing (increasing) α for Gaussians (multiquadrics). However, this is only possible at the cost of numerical instability (ill-conditioning of A). This is to be expected since for small (large) values of α the Gaussians (multiquadrics) more and more resemble a constant function, and therefore the rows (as well as columns) of the matrix A become more and more alike, so that the matrix becomes almost singular – even for well separated data sites. In the literature this phenomenon has been referred to as *trade-off* or (*uncertainty*) *principle* (see, e.g., the paper [549] by Schaback). The relation between the power function (as part of the upper bound on the approximation error) and minimal eigenvalue (as part of the measure of the condition number) is derived below.

Remarks:

1. This trade-off has led a number of people to search for an "optimal" value of the shape parameter, i.e., a value that yields maximal accuracy, while still maintaining numerical stability. For example, in his original work on (inverse) multiquadric interpolation in $\mathbb{R}^2$ Hardy [286] suggested using $\alpha = 0.815d$, where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$, and d_i is the distance from $\boldsymbol{x}_i$ to its nearest neighbor. Later Franke [231] suggested using $\alpha = 1.25\frac{D}{\sqrt{N}}$, where D is the diameter of the smallest circle containing all data points. Foley [221] based his strategy for finding a good value for α on the observation that that good value was similar for multiquadrics and inverse multiquadrics. Other studies were reported in [102] and [103]. A more recent algorithm was proposed by Rippa in [531]. He suggests a variant of cross validation known as "leave-one-out" cross validation. This method is rather popular in the statistics literature where it is also known as PRESS (Predictive REsidual Sum of Squares). In this algorithm an "optimal" value of α is selected by minimizing the least squares error for a fit based on the data for which one of the centers was "left out". A similar strategy was proposed earlier in [262] for the solution of elliptic partial differential equations via the dual reciprocity method based on multiquadric interpolation.

2. More recently, Fornberg and co-workers have investigated the dependence of the stability on the values of the shape parameter α in a series of papers (e.g., [159, 228, 361]). On the one hand, they suggest a way of stably computing very accurate (inverse) multiquadric and Gaussian interpolants (with extreme values

of α) by using a complex Contour-Padé integration algorithm. This algorithm is rather expensive, and so far only applicable for problems involving no more than 100 centers. On the other hand, Fornberg and co-workers as well as Schaback [558] have shown that in the limiting case of the shape parameter α, i.e., with very "flat" basis functions, the infinitely smooth radial basis function interpolants approach multivariate polynomial interpolants. Therefore, Fornberg and his co-workers suggest using radial basis functions as a generalization of spectral methods (applicable also in the case of scattered data) for the numerical solution of partial differential equations. This approach was also taken recently by Sarra [543].

We can observe that for the functions with finite smoothness (as in Examples 3–5) the lower bounds for $\|A^{-1}\|_2$ are of the same order as the upper bounds for the power function. The following theorem therefore shows that the order of both of these bounds is optimal. The theorem also provides the foundation for the trade-off principle referred to above.

Theorem 8.1.2 *Let $u_j^*(\boldsymbol{x})$, $j = 1, \ldots, N$, denote the cardinal functions for interpolation with the strictly positive definite function Φ as explained in Chapter 5, and let $\lambda_{\boldsymbol{x}}$ be the minimal eigenvalue of the $(N+1) \times (N+1)$ matrix $A_{\boldsymbol{x}}$ with entries $A_{\boldsymbol{x},ij} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$, $i, j = 0, 1, \ldots, N$, where we have added the evaluation point to the set of centers, i.e., $\boldsymbol{x}_0 = \boldsymbol{x}$. Then*

$$\lambda_{\boldsymbol{x}}^{-1} P_{\Phi,\mathcal{X}}^2(\boldsymbol{x}) \geq 1 + \sum_{j=1}^{N} \left[u_j^*(\boldsymbol{x})\right]^2.$$

Proof: The definition of the power function (with standard interpolation matrix A, and right-hand side vector $\boldsymbol{b}(\boldsymbol{x}) = [\Phi(\boldsymbol{x} - \boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x} - \boldsymbol{x}_N)]^T$, see Section 5.3) yields

$$\begin{aligned} P_{\Phi,\mathcal{X}}^2(\boldsymbol{x}) &= (\boldsymbol{u}^*(\boldsymbol{x}))^T A \boldsymbol{u}^*(\boldsymbol{x}) - 2(\boldsymbol{u}^*(\boldsymbol{x}))^T \boldsymbol{b}(\boldsymbol{x}) + \Phi(\boldsymbol{0}) \\ &= \sum_{j=1}^{N}\sum_{k=1}^{N} u_j^*(\boldsymbol{x}) u_k^*(\boldsymbol{x}) \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) - 2\sum_{j=1}^{N} u_j^*(\boldsymbol{x}) \Phi(\boldsymbol{x} - \boldsymbol{x}_j) + \Phi(\boldsymbol{x} - \boldsymbol{x}). \end{aligned}$$

If we define $u_0^*(\boldsymbol{x}) = -1$ and $\boldsymbol{x}_0 = \boldsymbol{x}$, then

$$P_{\Phi,\mathcal{X}}^2(\boldsymbol{x}) = \sum_{j=0}^{N}\sum_{k=0}^{N} u_j^*(\boldsymbol{x}) u_k^*(\boldsymbol{x}) \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k).$$

Finally, by using the Rayleigh quotient to get

$$\lambda_{\boldsymbol{x}} \leq \frac{\boldsymbol{c}^T A_{\boldsymbol{x}} \boldsymbol{c}}{\boldsymbol{c}^T \boldsymbol{c}}$$

for the (augmented) coefficient vector $\boldsymbol{c} = (\boldsymbol{u}_{\boldsymbol{x}}^*(\boldsymbol{x})) \in \mathbb{R}^{N+1}$ and $(N+1) \times (N+1)$ matrix $A_{\boldsymbol{x}}$, we have

$$P_{\Phi,\mathcal{X}}^2(\boldsymbol{x}) \geq \lambda_{\boldsymbol{x}} \sum_{j=0}^{N} \left[u_j^*(\boldsymbol{x})\right]^2,$$

and the statement follows by splitting off the $j = 0$ term again. $\square$

Theorem 8.1.2 not only implies the *uncertainty principle* (or *trade-off principle*) [549]

$$\lambda_{\boldsymbol{x}} \le P^2_{\Phi,\mathcal{X}}(\boldsymbol{x}) \quad \text{or} \quad \lambda_{\min} \le \min_{j=1,\ldots,N} P^2_{\Phi,\mathcal{X}\setminus\{\boldsymbol{x}_j\}}(\boldsymbol{x}),$$

which gives the power function as an upper bound for the smallest eigenvalue and vice versa. The theorem also provides an upper bound on the Lebesgue function, i.e.,

$$\sum_{j=1}^{N} \left|u_j^*(\boldsymbol{x})\right|^2 \le \frac{P^2_{\Phi,\mathcal{X}}(\boldsymbol{x})}{\lambda_{\boldsymbol{x}}} - 1.$$

Since the power function can be bounded in terms of the fill distance $h_{\mathcal{X},\Omega}$, and the minimum eigenvalue in terms of the separation distance $q_{\mathcal{X}}$, we see that for quasi-uniformly distributed data, i.e., if $h_{\mathcal{X},\Omega} \simeq q_{\mathcal{X}}$, the Lebesgue function will not grow too rapidly.

There is also a trade-off principle for compactly supported functions. This was explained theoretically as well as illustrated with numerical experiments by Schaback [553]. The consequences are as follows. In the case of stationary interpolation, i.e., if we scale the support size of the basis functions proportional to the fill distance $h_{\mathcal{X},\Omega}$, then the "bandwidth" of the interpolation matrix A is constant. This means we can apply numerical algorithms (e.g., conjugate gradient) that can be performed in $\mathcal{O}(N)$ computational complexity. The method is numerically stable, but there will be essentially no convergence (see Table 8.1). In the non-stationary case, i.e., with fixed support size, the bandwidth of A increases as $h_{\mathcal{X},\Omega}$ decreases. This results in convergence (i.e., the error decreases) as we showed in Chapter 5, but the interpolation matrices will become more and more dense as well as ill-conditioned. Therefore, this approach is not very efficient (see Table 8.2).

In Tables 8.1 and 8.2 we illustrate this behavior. We use the compactly supported function $\varphi_{3,1}(r) = (1-r)_+^4\,(4r+1)$ to interpolate Franke's function

$$\begin{aligned} F(x,y) \quad = \quad & \frac{3}{4}\left[\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right)\right] \\ & + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - (9y-3)^2\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned}$$

on a grid of equally spaced points in the unit square $[0,1]^2$. In the stationary case (Table 8.1) the support of the basis function is scaled to contain 25 grid points. Therefore, the "bandwidth" of the interpolation matrix A is kept constant (at 25), so that A is very sparse for finer grids. We can observe convergence for the first few iterations, but once an ℓ_2-error of approximately 2×10^{-3} is reached, there is no further improvement. This behavior is not yet fully understood. However, it is similar to what happens in the *approximate approximation* method of Maz'ya (see, e.g., [434]). The rate listed in the table is the exponent β of the observed ℓ_2-convergence rate $\mathcal{O}(h^\beta)$. The % nonzero column indicates the sparsity of the interpolation matrices, and the time is measured in seconds.

In the non-stationary case (Table 8.2) we used the basis function without adjusting its support size. This is the situation to which the error bounds of Chapter 5 apply.

Mesh	ℓ_2-error	rate	% nonzero	time
3×3	2.367490e-01		100	0
5×5	6.572754e-02	1.849	57.8	0
9×9	1.740723e-02	1.917	23.2	0
17×17	2.362950e-03	2.881	7.47	1
33×33	2.060493e-03	0.198	2.13	1
65×65	2.012010e-03	0.034	0.06	11
129×129	2.007631e-03	0.003	0.01	158

Table 8.1: 2D stationary interpolation with $\varphi(r) = (1-r)_+^4(4r+1)$, 25 points in support.

Mesh	ℓ_2-error	rate
3×3	2.407250e-01	
5×5	7.101748e-02	1.761
9×9	1.833534e-02	1.954
17×17	1.392914e-03	3.718
33×33	3.050789e-04	2.191
65×65	9.314516e-06	5.034

Table 8.2: 2D non-stationary interpolation with $\varphi(r) = (1-r)_+^4(4r+1)$, unit support.

We have convergence – although it is not obvious what the rate might be. However, the matrices become increasingly dense. Therefore, Table 8.2 is missing the entry for the 129×129 case, and even though no times are provided in that table, the time for the 65×65 case is already more than 20 minutes on a standard desktop PC.

8.2 Multilevel Interpolation and Approximation

In order to overcome the problems with both approaches for interpolation with compactly supported radial functions described above, Schaback suggested using a multilevel stationary scheme. This scheme was implemented first by Floater and Iske [217] and later studied by a number of other researchers (see, e.g., [115, 209, 281, 297, 318, 478, 630].

The basic idea of the multilevel interpolation algorithm is to scale the size of the support of the basis function with $h_{\mathcal{X},\Omega}$, but to interpolate to residuals on progressively refined sets of centers. This method has all of the combined benefits of the methods described earlier: it is computationally efficient (can be performed in $\mathcal{O}(N)$ operations), well-conditioned, and convergent.

An algorithm for multilevel interpolation is as follows:

Algorithm: (Multilevel interpolation)

1. Create nested point sets $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_K = \mathcal{X} \subset \mathbb{R}^s$, and initialize $\mathcal{P}f(\boldsymbol{x}) = 0$.

2. For $k = 1, 2, \ldots, K$ do

Mesh	ℓ_2-error	rate	% nonzero	time
3×3	2.367490e-01		100	0
5×5	6.665899e-02	1.828	57.8	0
9×9	2.087575e-02	1.675	23.2	0
17×17	1.090837e-04	4.258	7.47	0
33×33	1.497227e-04	2.865	2.13	6
65×65	5.313053e-05	1.495	0.06	37
129×129	1.112638e-05	2.256	0.01	212

Table 8.3: 2D (stationary) multilevel interpolation with $\varphi(r) = (1-r)_+^4(4r+1)$.

(a) Solve $u(\boldsymbol{x}) = f(\boldsymbol{x}) - \mathcal{P}f(\boldsymbol{x})$ on $\mathcal{X}_k$.

(b) Update $\mathcal{P}f(\boldsymbol{x}) = \mathcal{P}f(\boldsymbol{x}) + u(\boldsymbol{x})$.

The representation of the update u at step k is of the form

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{X}_k} c_j^{(k)} \varphi\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|}{\rho_k}\right)$$

with $\rho_k \simeq h_{\mathcal{X}_k,\Omega}$. This requires the solution of a linear system whose size is determined by the number of points in $\mathcal{X}_k$.

In the numerical example listed in Table 8.3 we again use the compactly supported function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$ and Franke's function.

The initial scale ρ_1 was chosen so that the basis function was supported on $[-2, 2]$. Subsequent scales were successively divided by 2 – just as the fill distance of the computational grids $\mathcal{X}_k$. The rate listed in the table is the exponent β of the observed ℓ_2-convergence rate $\mathcal{O}(h^\beta)$. The % nonzero column indicates the sparsity of the interpolation matrices, and the time is measured in seconds.

So far there are only limited theoretical results concerning the convergence of this multilevel algorithm. Narcowich, Schaback and Ward [478] show that a related algorithm (in which additional boundary conditions are imposed) converges at least linearly, and Hartmann analyzed the multilevel algorithm in his Ph.D. thesis [297]. He showed at least linear convergence for multilevel interpolation on a regular lattice for various radial basis functions. Similar results are obtained by Hales and Levesley [281] for polyharmonic splines, i.e., thin plate splines and powers. The main difficulty in proving the convergence of the multilevel algorithm is the fact that the approximation space changes from one level to the next. The approximation spaces are not nested (as they usually are for wavelets). This means that the native space norm changes from one level to the next. Hales and Levesley avoid this problem by scaling the (uniformly spaced) data instead of the basis functions. Then the fact that polyharmonic splines are in a certain sense harmonic (see Section 8.4) simplifies the analysis. This fact was also used by Wendland [634] to prove linear convergence for multilevel (scattered data) interpolation based on thin plate splines.

The same basic multilevel algorithm can also be used for other approximation methods. In [203] the idea was applied to moving least squares methods and approximate moving least squares methods. Tables 8.4 and 8.5 illustrate the effect of

Mesh	Shepard			linear precision		
	ℓ_2-error	rate	time	ℓ_2-error	rate	time
3×3	2.737339e-01		7	2.749670e-01		14
5×5	1.100713e-01	1.314	7	1.033060e-01	1.412	13
9×9	5.393041e-02	1.029	5	5.242492e-02	0.979	9
17×17	1.507797e-02	1.839	3	1.502361e-02	1.803	5
33×33	4.124059e-03	1.870	3	4.111092e-03	1.870	4
65×65	1.061904e-03	1.957	2	1.047348e-03	1.973	3
129×129	2.628645e-04	2.014	2	2.628645e-04	1.994	3

Table 8.4: 2D MLS approximation with weight $\varphi(r) = (1-r)_+^4(4r+1)$.

the multilevel algorithm for Shepard's method and a moving least squares approximation with linear precision, both based on the compactly supported weight function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$. This experiment was conducted with a mollified Franke function f on the unit square $[0,1]^2$, i.e.,

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

The support scaling was as in the previous multilevel example.

One can observe that the basic Shepard's method actually performs much better than the predicted $\mathcal{O}(h)$ (see Table 8.4). Notice that the multilevel algorithm (illustrated in Table 8.5) improves the accuracy considerably at very little extra cost. It is interesting to note that this effect is much more pronounced for computations in $\mathbb{R}^2$ than in $\mathbb{R}$ (cf. [202]). The times listed in Tables 8.4 and 8.5 are due only to the evaluation on a very fine evaluation mesh since the method was coded so that no linear systems had to be solved. This means that the Lagrange multipliers for the case of linear precision were determined explicitly by solving the 3×3 Gram system analytically (cf. (7.5)). The resulting generating functions 7.6) were directly coded into the program.

There seems to be no theoretical investigation of the convergence properties of the multilevel algorithm for moving least squares approximation.

8.3 Preconditioning

In the first section of this chapter we noted that the system matrices arising in scattered data interpolation with radial basis functions tend to become very ill-conditioned as the minimal separation distance $q_{\mathcal{X}}$ between the data sites $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, is reduced. Therefore it is natural to devise strategies to prevent such instabilities by either preconditioning the system, or by finding a better basis for the approximation space we

Mesh	Shepard			linear precision		
	ℓ_2-error	rate	time	ℓ_2-error	rate	time
3×3	2.737339e-01		7	2.749670e-01		14
5×5	1.076424e-01	1.347	7	1.013114e-01	1.440	12
9×9	3.909725e-02	1.461	5	4.308322e-02	1.234	9
17×17	7.327282e-03	2.416	3	8.549613e-03	2.333	6
33×33	9.545860e-04	2.940	2	8.937409e-04	3.258	4
65×65	1.424136e-04	2.745	2	9.896052e-05	3.175	3
129×129	3.946680e-05	1.851	2	1.361339e-05	2.872	2

Table 8.5: 2D multilevel MLS approximation with $\varphi(r) = (1-r)_+^4(4r+1)$.

are using. The former approach is standard procedure in numerical linear algebra, and in fact we can use any of the well-established methods (such as preconditioned conjugate gradient iteration) to improve the stability and convergence of the interpolation systems that arise for strictly positive definite functions. In particular, the sparse systems that arise in (multilevel) interpolation with compactly supported radial basis functions can be efficiently solved with the preconditioned conjugate gradient method, and in fact the examples reported in the previous section were implemented using the conjugate gradient method with a diagonal (Jacobi) preconditioner.

The idea of using a more stable basis is well known from univariate polynomial and spline interpolation. The Lagrange basis functions for univariate polynomial interpolation are of course the ideal basis if we are interested in stably solving the interpolation equations since the resulting interpolation matrix is the identity matrix (which is certainly much better conditioned than, e.g., the Vandermonde matrix that we get if we use a monomial basis). Similarly, B-splines give rise to diagonally dominant, sparse system matrices which are much easier to deal with than the matrices we would get if we were to represent a spline interpolant using the alternative truncated power basis. Both of these examples are studied in great detail in standard numerical analysis texts (see, e.g., [350]) or in the literature on splines (see, e.g., [574]). We will address an analogous approach for radial basis functions in the next section.

Before we describe any of the specialized preconditioning procedures for radial basis function interpolation matrices we give two examples presented by Jackson [326] to illustrate the effects of and motivation for preconditioning in the context of radial basis functions.

8.3.1 Two Simple Examples

Example 1: (Uniform data) Let $s = 1$ and $\varphi(r) = r$ with no polynomial terms added. As data sites we choose $\mathcal{X} = \{1, 2, \dots, 10\}$. This leads to the system matrix

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & \dots & 9 \\ 1 & 0 & 1 & 2 & \dots & 8 \\ 2 & 1 & 0 & 1 & \dots & 7 \\ 3 & 2 & 1 & 0 & \dots & 6 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 9 & 8 & 7 & 6 & \dots & 0 \end{bmatrix}$$

with ℓ_2-condition number $\text{cond}(A) \approx 67$. Instead of solving the linear system $A\boldsymbol{c} = \boldsymbol{y}$, where $\boldsymbol{y} = [y_1, \dots, y_{10}]^T \in \mathbb{R}^{10}$ is a vector of given real numbers (data values), we can find a suitable matrix B to premultiply both sides of the equation such that the system is simpler to solve. Ideally, the new system matrix BA should be the identity matrix, i.e., B should be an approximate inverse of A. Thus, having found an appropriate matrix B, we must now solve the linear system $BA\boldsymbol{c} = B\boldsymbol{y}$. For the matrix A above we can choose a preconditioner B as

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{2} & -1 & \frac{1}{2} & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

This leads to the following preconditioned system matrix BA in the system

$$\begin{bmatrix} 0 & 1 & 2 & \dots & 8 & 9 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 9 & 8 & 7 & \dots & 1 & 0 \end{bmatrix} \boldsymbol{c} = B\boldsymbol{y},$$

which is almost an identity matrix. Now $\text{cond}(BA) \approx 45$.

The motivation for this choice of B is the following. The function $\varphi(r) = r$ or $\Phi(x) = |x|$ is a fundamental solution of the Laplacian Δ, i.e.,

$$\Delta\Phi(x) = \frac{d^2}{dx^2}|x| = \frac{1}{2}\delta_0(x),$$

where δ_0 is the Dirac delta function. Thus, B is chosen as a discretization of the Laplacian with special choices at the endpoints of the data set.

Remark: One can also verify that the function $\Phi(x) = |x|$ minimizes the functional

$$\int_{\mathbb{R}} \left[f'(x)\right]^2 dx$$

over all functions interpolating data sampled from a function in the space

$$F = \{f \in C(\mathbb{R}), f' \in L_2(\mathbb{R})\}.$$

Therefore, the radial basis function $\varphi(r) = r$ is a linear (natural) spline. An analogous argument can be used to show that the function $\varphi(r) = r^3$ in $\mathbb{R}$ is nothing but a cubic natural spline. This is in agreement with the variational theory presented earlier according to which every radial basis function represents the minimum norm interpolant in its native space.

Example 2: (Nonuniform data) For nonuniformly distributed data we can use a different discretization of the Laplacian Δ for each row of B. To see this, let $s = 1$, and $\mathcal{X} = \{1, \frac{3}{2}, \frac{5}{2}, 4, \frac{9}{2}\}$, and again consider the radial function $\varphi(r) = r$. Then

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{3}{2} & 3 & \frac{7}{2} \\ \frac{1}{2} & 0 & 1 & \frac{5}{2} & 3 \\ \frac{3}{2} & 1 & 0 & \frac{3}{2} & 2 \\ 3 & \frac{5}{2} & \frac{3}{2} & 0 & \frac{1}{2} \\ \frac{7}{2} & 3 & 2 & \frac{1}{2} & 0 \end{bmatrix}$$

with $\operatorname{cond}(A) \approx 18.15$, and if we choose

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -\frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{5}{6} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & -\frac{4}{3} & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

based on second-order backward differences of the points in $\mathcal{X}$, then the preconditioned system to be solved becomes

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{3}{2} & 3 & \frac{7}{2} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{7}{2} & 3 & 2 & \frac{1}{2} & 0 \end{bmatrix} \boldsymbol{c} = B\boldsymbol{y}.$$

Once more, this system is almost trivial to solve and has an improved condition number of $\operatorname{cond}(BA) \approx 8.94$.

8.3.2 Early Preconditioners

Ill-conditioning of the interpolation matrices was identified as a serious problem very early, and Nira Dyn along with some of her co-workers (see, e.g., [172], [173], [178], or [180]) provided some of the first preconditioning strategies tailored especially to radial basis functions.

For the following discussion we consider the general interpolation problem which includes polynomial reproduction. Therefore, we have to solve the following system of linear equations

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix}, \tag{8.1}$$

with the individual pieces given by $A_{jk} = \varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|)$, $j,k = 1,\ldots,N$, $P_{j\ell} = p_\ell(\boldsymbol{x}_j)$, $j = 1,\ldots,N$, $\ell = 1,\ldots,M$, $\boldsymbol{c} = [c_1,\ldots,c_N]^T$, $\boldsymbol{d} = [d_1,\ldots,d_M]^T$, $\boldsymbol{y} = [y_1,\ldots,y_N]^T$, and $\boldsymbol{0}$ a zero vector of length M with $M = \dim\Pi^s_{m-1}$. Here, as discussed earlier, φ should be strictly conditionally positive definite of order m and radial on $\mathbb{R}^s$ and the set $\mathcal{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ should be $(m-1)$-unisolvent.

The preconditioning scheme proposed by Dyn and her co-workers is a generalization of the simple differencing scheme discussed above. It is motivated by the fact that the polyharmonic splines

$$\varphi(r) = \begin{cases} r^{2k-s}\log r, & s \text{ even}, \\ r^{2k-s}, & s \text{ odd}, \end{cases}$$

$2k > s$, are fundamental solutions of the k-th iterated Laplacian in $\mathbb{R}^s$, i.e.,

$$\Delta^k \varphi(\|\boldsymbol{x}\|) = c\delta_{\boldsymbol{0}}(\boldsymbol{x}),$$

where $\delta_{\boldsymbol{0}}$ is the Dirac delta function, and c is an appropriate constant. For the (inverse) multiquadrics $\varphi(r) = (r^2+\alpha^2)^{\pm 1/2}$, which are also discussed in the papers mentioned above, application of the Laplacian yields a similar limiting behavior, i.e.,

$$\lim_{r\to\infty} \Delta^k \varphi(r) = 0,$$

and for $r \to 0$

$$\Delta^k \varphi(r) \gg 1.$$

One now wants to discretize the Laplacian on the (irregular) mesh given by the (scattered) data sites in $\mathcal{X}$. To this end Dyn, Levin, and Rippa [180] suggest the following procedure for the case of scattered data interpolation over $\mathbb{R}^2$.

1. Start with a triangulation of the set $\mathcal{X}$, e.g., the *Delaunay triangulation* will do. This triangulation can be visualized as follows.
   (a) Begin with the points in $\mathcal{X}$ and construct their *Dirichlet tesselation.* The Dirichlet tile of a particular point $\boldsymbol{x}$ is that subset of points in $\mathbb{R}^2$ which are closer to $\boldsymbol{x}$ than to any other point in $\mathcal{X}$. The left part of Figure 8.2 shows the Dirichlet tesselation for a given set of 6 points.
   (b) Construct the Delaunay triangulation, which is the dual of the Dirichlet tesselation, i.e., connect all strong neighbors in the Dirichlet tesselation, i.e., points whose tiles share a common edge. The right part of Figure 8.2 shows the corresponding Delaunay triangulation of the 6 points.

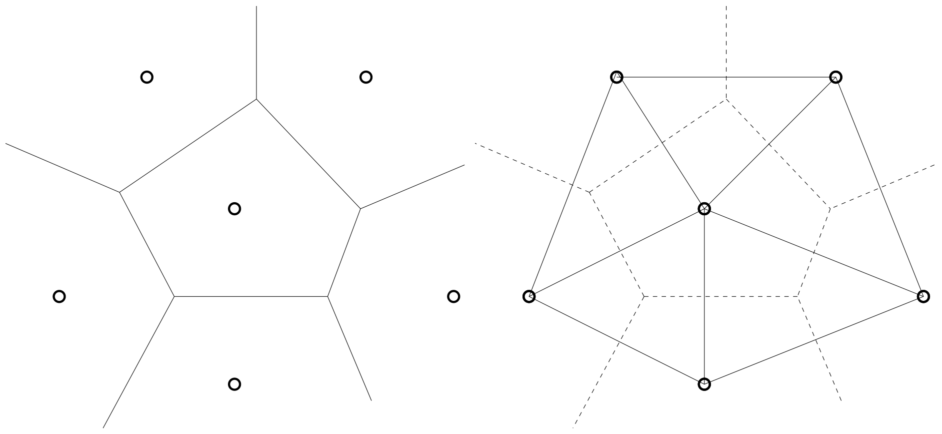
Figure 8.2: Dirichlet tesselation (left) and corresponding Delaunay triangulation (right) of the points $\circ$.

2. Discretize the Laplacian on this triangulation. In order to also take into account the boundary points Dyn, Levin and Rippa instead use a discretization of an iterated Green's formula which has the space Π^2_{m-1} as its null space. The necessary partial derivatives are then approximated on the triangulation using certain sets of vertices of the triangulation. (3 points for first order partials, 6 for second order).

The discretization described above yields the matrix $B = (b_{ji})_{j,i=1}^N$ as the preconditioning matrix in an analogous to the previous section. We now obtain

$$(BA)_{jk} = \sum_{i=1}^{N} b_{ji}\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_k\|) \approx \Delta^m \varphi(\| \cdot - \boldsymbol{x}_k\|)(\boldsymbol{x}_j), \quad j,k = 1,\dots,N, \tag{8.2}$$

which has the property that the entries close to the diagonal are large compared to those away from the diagonal, which decay to zero as the distance between the two points involved goes to infinity. Since the part $BP = 0$ by step 2 of the construction, one must now solve the system

$$\begin{aligned} BA\boldsymbol{c} &= B\boldsymbol{y} \\ P^T\boldsymbol{c} &= 0. \end{aligned}$$

Actually, the system $BA\boldsymbol{c} = B\boldsymbol{y}$ is singular, but it is shown in the paper [180] that the additional constraints $P^T\boldsymbol{c} = 0$ guarantee existence of a unique solution. Furthermore, the coefficients $\boldsymbol{d}$ in the original expansion of the interpolant s can be obtained by solving

$$P\boldsymbol{d} = \boldsymbol{y} - A\boldsymbol{c},$$

i.e., by fitting the polynomial part of the expansion to the residual $\boldsymbol{y} - A\boldsymbol{c}$.

The approach just described leads to localized basis functions ψ which are linear combinations of the original basis functions φ. More precisely,

$$\psi_j(\boldsymbol{x}) = \sum_{i=1}^{N} b_{ji}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \approx \Delta^m \varphi(\| \cdot - \boldsymbol{x}_j\|)(\boldsymbol{x}), \tag{8.3}$$

φ	N	Grid I orig.	Grid I precond.	Grid II orig.	Grid II precond.
TPS	49	1181	4.3	1885	3.4
	121	6764	5.1	12633	3.9
MQ	49	7274	69.2	17059	222.8
	121	10556	126.0	107333	576.0

Table 8.6: Condition numbers without and with preconditioning.

where the coefficients b_{ji} are those determined in the discretization above.

Remarks:

1. The localized basis functions ψ_j, $j = 1, \ldots, N$, (see (8.3)) can be viewed as an alternative (better conditioned) basis for the approximation space spanned by the functions $\varphi_j = \varphi(\| \cdot - \boldsymbol{x}_j \|)$.

2. In [180] it is described how the preconditioned matrices can be used efficiently with various iterative schemes such as Chebyshev iteration or a version of the conjugate gradient method. The authors also mention smoothing of noisy data, or low-pass filtering as another application for this preconditioning scheme.

The effectiveness of the above preconditioning strategy was illustrated with some numerical examples in [180]. We list some of their results in Table 8.6. Thin plate splines and multiquadrics were tested on two different data sets (grid I and grid II) in $\mathbb{R}^2$. The shape parameter α for the multiquadrics was chosen to be the average mesh size. A linear term was added for thin plate splines, and a constant for multiquadrics.

One can see that the most dramatic improvement is achieved for the thin plate splines. This is to be expected since the method is tailored to them. As noted earlier, for the multiquadrics an application of the Laplacian does not yield the delta function, but for values of r close to zero gives just relatively large values.

Remarks:

1. Another early preconditioning strategy was suggested by Powell [516]. Powell uses Householder transformations to convert the matrix of the interpolation system (8.1) to a symmetric positive definite matrix, and then uses the conjugate gradient method. However, Powell reports that this method is not particularly effective for large thin plate spline interpolation problems in $\mathbb{R}^2$.

2. Baxter [27, 30] discusses the use of a preconditioned conjugate gradient method for solving the interpolation problem in the case when Gaussians or multiquadrics are used on a regular grid. The resulting matrices are Toeplitz matrices, and a large body of literature exists for dealing with this special case (see, e.g., [110]).

8.3.3 Preconditioned GMRES via Approximate Cardinal Functions

More recently, Beatson, Cherrie and Mouat [34] have proposed a preconditioner for the iterative solution of radial basis function interpolation systems using the GMRES method of Saad and Schultz [538]. The GMRES method is a general purpose iterative solver that can be applied to nonsymmetric (nondefinite) systems. For fast convergence the matrix should be preconditioned such that its eigenvalues are clustered around 1 and away from the origin. Obviously, if the basis functions for the radial basis function space were cardinal functions, then the matrix would be the identity matrix with all its eigenvalues equal to 1. Therefore, the GMRES method would converge in a single iteration. Consequently, the preconditioning strategy for the GMRES method is to obtain a preconditioning matrix B that is close to the inverse of A.

Since it is too expensive to find the true cardinal basis (this would involve at least as much work as solving the interpolation problem), the idea pursued in [34] (and suggested earlier in [36, 46]) is to find *approximate* cardinal functions similar to the functions ψ_j in the previous subsection. Now, however, there is also an emphasis on efficiency, i.e., we are interested in *local* approximate cardinal functions, if possible. Several different strategies were suggested in [34]. We will now explain the basic idea.

Given the centers $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the j-th approximate cardinal function is given as a linear combination of the basis functions $\varphi_i = \varphi(\|\cdot - \boldsymbol{x}_i\|)$, where i runs over (some subset of) $\{1, \ldots, N\}$, i.e.,

$$\psi_j = \sum_{i=1}^{N} b_{ji} \varphi(\|\cdot - \boldsymbol{x}_i\|) + p_j, \tag{8.4}$$

where (for the conditionally positive definite case) p_j is a polynomial in Π^s_{m-1} and the coefficients b_{ji} satisfy the usual conditions

$$\sum_{i=1}^{N} b_{ji} p_j(\boldsymbol{x}_i) = 0 \qquad \text{for all } p_j \in \Pi^s_{m-1}. \tag{8.5}$$

The key feature in designing the approximate cardinal functions is to have only a few $n \ll N$ coefficients in (8.4) to be nonzero. In that case the functions ψ_j are found by solving small $n \times n$ linear systems, which is much more efficient than dealing with the original $N \times N$ system. For example, in [34] the authors use $n \approx 50$ for problems involving up to 10,000 centers. The resulting preconditioned system is of the same form as the earlier preconditioner (8.2), i.e., we now have to solve the preconditioned problem

$$(BA)\boldsymbol{c} = B\boldsymbol{y},$$

where the entries of the matrix BA are just $\psi_j(\boldsymbol{x}_k)$, $j, k = 1, \ldots, N$.

The simplest strategy for determining the coefficients b_{ji} is to select the n nearest neighbors of $\boldsymbol{x}_j$, and to find b_{ji} by solving the (local) cardinal interpolation problem

$$\psi_j(\boldsymbol{x}_i) = \delta_{ij}, \qquad i = 1, \ldots, n,$$

subject to the moment constraint (8.5) listed above. Here δ_{ij} is the Kronecker-delta, and the points $\boldsymbol{x}_i$ are the nearest neighbors selected above.

φ	N	unprecond.	local precond.	local precond. w/special
TPS	289	4.005e06	1.464e03	5.721e00
	1089	2.753e08	6.359e05	1.818e02
	4225	2.605e09	2.381e06	1.040e06
MQ	289	1.506e08	3.185e03	2.639e02
	1089	2.154e09	8.125e05	5.234e04
	4225	3.734e10	1.390e07	4.071e04

Table 8.7: Condition numbers without and with preconditioning.

This basic strategy is improved by adding so-called *special points* that are distributed (very sparsely) throughout the domain.

A few numerical results for thin plate spline and multiquadric interpolation in $\mathbb{R}^2$ from [34] are listed in Table 8.7. The condition numbers are ℓ_2-condition numbers, and the points were randomly distributed in the unit square. The "local precond." column uses the $n = 50$ nearest neighbors to determine the approximate cardinal functions, whereas the right-most column uses the 41 nearest neighbors plus 9 special points placed uniformly in the unit square. The effect of the preconditiong on the performance of the GMRES algorithm was, e.g., a reduction from 103 to 8 iterations for the 289 point data set for thin plate splines, or from 145 to 11 for multiquadrics.

Remark: An extension of the ideas of Beatson, Cherrie and Mouat [34] to linear systems arising in the collocation solution of partial differential equations (see Chapter 9) was explored in Mouat's Ph.D. thesis [468] and also in the recent paper by Ling and Kansa [395].

8.4 Change of Basis

Another idea that can be used to obtain a "better" basis for conditionally positive definite radial basis functions is closely connected to finding the reproducing kernel of the associated native space. Since we did not elaborate on the construction of the native spaces for conditionally positive definite functions earlier, we will now present the relevant formulas (without going into the details). In particular, for polyharmonic splines we will be able to find a basis that is in a certain sense *homogeneous*, and therefore the condition number of the related interpolation matrix will depend only on the number N of data points, but *not* on their separation distance.

This approach was suggested by Beatson, Light and Billings [42], and has its roots in work by Sibson and Stone [582].

Let Φ be a strictly conditionally positive definite kernel of order m, and $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega \subset \mathbb{R}^s$ be an $(m-1)$-unisolvent set of centers. Then the reproducing kernel for the native space $\mathcal{N}_\Phi(\Omega)$ is given by

$$K(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \Phi(\boldsymbol{x}, \boldsymbol{y}) - \sum_{k=1}^{M} p_k(\boldsymbol{x}) \Phi(\boldsymbol{x}_k, \boldsymbol{y}) - \sum_{\ell=1}^{M} p_\ell(\boldsymbol{y}) \Phi(\boldsymbol{x}, \boldsymbol{x}_\ell)$$

$$+\sum_{k=1}^{M}\sum_{\ell=1}^{M} p_k(\boldsymbol{x})p_\ell(\boldsymbol{y})\Phi(\boldsymbol{x}_k,\boldsymbol{x}_\ell) + \sum_{\ell=1}^{M} p_\ell(\boldsymbol{x})p_\ell(\boldsymbol{y}),$$

where the points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M$ are an $(m-1)$-unisolvent subset of $\mathcal{X}$ and the polynomials p_k, $k=1,\ldots,M$, form a *cardinal* basis for Π^s_{m-1} whose dimension is $M=\binom{s+m-1}{m-1}$, i.e.,

$$p_\ell(\boldsymbol{x}_k)=\delta_{k,\ell}, \qquad k,\ell=1,\ldots,M.$$

An immediate consequence is that we can express the radial basis function interpolant to values of some function f given on $\mathcal{X}$ in the form

$$\mathcal{P}f(\boldsymbol{x})=\sum_{j=1}^{N} c_j K(\boldsymbol{x},\boldsymbol{x}_j), \qquad \boldsymbol{x}\in\mathbb{R}^s.$$

The coefficients c_j are determined by satisfying the interpolation conditions

$$\mathcal{P}f(\boldsymbol{x}_i)=f(\boldsymbol{x}_i), \qquad i=1,\ldots,N.$$

We will see below (in Tables 8.8 and 8.9) that this basis already performs "better" than the standard basis $\{\Phi(\cdot,\boldsymbol{x}_1),\ldots,\Phi(\cdot,\boldsymbol{x}_N)\}$ if we keep the number of centers fixed, and vary only their separation distance.

To obtain the homogeneous basis referred to above we modify K by subtracting the tensor product polynomial, i.e.,

$$\kappa(\boldsymbol{x},\boldsymbol{y})=K(\boldsymbol{x},\boldsymbol{y})-\sum_{\ell=1}^{M} p_\ell(\boldsymbol{x})p_\ell(\boldsymbol{y}).$$

Now, if $\boldsymbol{y}$ is one of the points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M$ in the $(m-1)$-unisolvent subset of $\mathcal{X}$ mentioned above, then

$$\begin{aligned}\kappa(\cdot,\boldsymbol{y}) &= \Phi(\cdot,\boldsymbol{y})-\sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k,\boldsymbol{y})-\sum_{\ell=1}^{M} p_\ell(\boldsymbol{y})\Phi(\cdot,\boldsymbol{x}_\ell)+\sum_{k=1}^{M}\sum_{\ell=1}^{M} p_k(\cdot)p_\ell(\boldsymbol{y})\Phi(\boldsymbol{x}_k,\boldsymbol{x}_\ell)\\ &= \Phi(\cdot,\boldsymbol{y})-\sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k,\boldsymbol{y})-\Phi(\cdot,\boldsymbol{y})+\sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k,\boldsymbol{y})=0.\end{aligned}$$

This means that the functions $\kappa(\cdot,\boldsymbol{x}_j)$, $j=1,\ldots,N$, cannot be used as a basis of our approximation space. However, it turns out that the matrix C with entries $C_{i,j}=\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$, $i,j=M+1,\ldots,N$, is positive definite, and therefore we obtain the following basis

$$\{p_1,\ldots,p_M\}\cup\{\kappa(\cdot,\boldsymbol{x}_{M+1}),\ldots,\kappa(\cdot,\boldsymbol{x}_N)\},$$

and the interpolant can be represented in the form

$$\mathcal{P}f(\boldsymbol{x})=\sum_{j=1}^{M} d_j p_j(\boldsymbol{x})+\sum_{k=M+1}^{N} c_k\kappa(\boldsymbol{x},\boldsymbol{x}_k), \qquad \boldsymbol{x}\in\mathbb{R}^s.$$

Spacing h	Standard matrix	Reproducing kernel	Homogeneous matrix
1/8	3.5158e03	1.8930e04	7.5838e03
1/16	3.8938e04	2.6514e05	1.1086e05
1/32	5.1363e05	4.0007e06	1.6864e06
1/64	7.6183e06	6.2029e07	2.6264e07

Table 8.8: Condition numbers for different thin plate spline bases on $[0,1]^2$ with increasing number of points and varying separation distance.

Since the polynomials p_k are cardinal on $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M\}$ the coefficients are determined by solving the linear system

$$\begin{bmatrix} I & 0 \\ P^T & C \end{bmatrix} \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{c} \end{bmatrix} = \boldsymbol{y}, \tag{8.6}$$

with I an $M \times M$ identity matrix, C as above, $P_{ij} = p_j(\boldsymbol{x}_i)$, $j = 1,\ldots,M$, $i = M+1,\ldots,N$, $\boldsymbol{c} = [c_{M+1},\ldots,c_N]^T$, $\boldsymbol{d} = [d_1,\ldots,d_M]^T$, and the right-hand side $\boldsymbol{y} = [f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_M), f(\boldsymbol{x}_{M+1}),\ldots,f(\boldsymbol{x}_N)]^T$. The identity block (cardinality of the polynomial basis functions) implies that the coefficient vector $\boldsymbol{d}$ is given by

$$d_j = f(\boldsymbol{x}_j), \qquad j = 1,\ldots,M,$$

and therefore the system (8.6) can be solved as

$$C\boldsymbol{c} = \tilde{\boldsymbol{y}} - P^T\boldsymbol{d},$$

where $\tilde{\boldsymbol{y}} = [f(\boldsymbol{x}_{M+1}),\ldots,f(\boldsymbol{x}_N)]^T$ and the matrix C is symmetric and positive definite. Finally, for polyharmonic splines, the ℓ_2-condition number of the matrix C is invariant under a uniform scaling of the centers, i.e., if $C^h = (\kappa(h\boldsymbol{x}_i, h\boldsymbol{x}_j))$, then $\operatorname{cond}(C^h) = \operatorname{cond}(C)$. This is proved to varying degrees in the paper [42] by Beatson, Light and Billings, the book [634] by Wendland, and the paper [320] by Iske.

We close with some numerical experiments from [42]. They use thin plate splines in $\mathbb{R}^2$. In the first experiment (illustrated in Table 8.8) the problem is formulated on the unit square $[0,1]^2$. Here both the number of points and the separation distance vary from one row in the table to the next. The three different columns list the ℓ_2-condition numbers of the interpolation matrix for the three different approaches mentioned above, i.e., using the standard basis consisting of functions $\Phi(\cdot,\boldsymbol{x}_j)$ and monomials, using the reproducing kernels $K(\cdot,\boldsymbol{x}_j)$, and using the matrix C. The three polynomial cardinal functions are based on the three corners $(0,0)$, $(0,1)$, and $(1,0)$. With this setup all three methods perform comparably.

In the second experiment (shown in Table 8.9) the number of points is kept fixed at 5×5 equally spaced points. However, the domain is scaled to the square $[0,a]^2$ with scale parameter a, so that only the separation distance $q_{\mathcal{X}}$ changes from one row to the next. Now, clearly the two new methods show less dependence on the separation distance, with the homogeneous matrix C being completely insensitive as claimed earlier.

Remark: Iske takes advantage of the scale invariance of polyharmonic splines (and thin plate splines in particular) in the construction of a numerical multiscale solver for transport problems (see, e.g., [47]).

Scale parameter	Standard matrix	Reproducing kernel	Homogeneous matrix
0.001	2.4349e08	8.4635e08	5.4938e02
0.01	2.4364e06	8.4640e06	5.4938e02
0.1	2.5179e04	8.5134e04	5.4938e02
1.0	3.6458e02	1.3660e03	5.4938e02
10	1.8742e06	1.2609e03	5.4938e02
100	1.1520e11	1.1396e05	5.4938e02
1000	3.4590e15	1.1386e07	5.4938e02

Table 8.9: Condition numbers for different thin plate spline bases on $[0, a]^2$ with fixed number of points and varying separation distance.

8.5 Special Numerical Algorithms

Since the use of radial basis functions for interpolation of scattered data leads to (large) linear systems that are frequently ill-conditioned it is important to devise algorithms that can

1. efficiently solve the interpolation system (preferably in $\mathcal{O}(N)$ operations), and
2. efficiently evaluate a radial basis function expansion once its coefficients have been determined (preferably in a constant number of operations – independent of N).

The second goal is also important for approximation via the moving least squares method or by quasi-interpolation.

All of the work described below is very recent, and it is quite likely that much more insight can be gained, and many improvements are still possible.

8.5.1 Iterative Algorithms

This section is based on the contents of the papers [562, 563] by Schaback and Wendland, the book [634] by Wendland, and the papers [212, 213] by Faul and Powell.

We concentrate on systems for strictly positive definite functions (variations for strictly conditionally positive definite functions also exist). One of the central ingredients is the native space inner product discussed in Chapter 5. As always we assume that our data sites are $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, but now we also consider a second set $\mathcal{Y} \subseteq \mathcal{X}$.

If we consider $\mathcal{P}_{\mathcal{Y}} f$ to be the interpolant to f on $\mathcal{Y} \subseteq \mathcal{X}$, then $\langle f - \mathcal{P}_{\mathcal{Y}} f, \mathcal{P}_{\mathcal{Y}} f \rangle_{\mathcal{N}_\Phi(\Omega)} = 0$ and we obtain the energy split (see Corollary 5.5.3)

$$\|f\|^2_{\mathcal{N}_\Phi(\Omega)} = \|f - \mathcal{P}_{\mathcal{Y}} f\|^2_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_{\mathcal{Y}} f\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

One possible point of view is now to consider an iteration on residuals. To this end we start with our desired interpolant $r_0 = \mathcal{P}_{\mathcal{X}} f$ on the entire set $\mathcal{X}$, and an appropriate sequence of sets $\mathcal{Y}_k$, $k = 0, 1, \ldots$ (we will discuss some possible choices later). Then, we iteratively define

$$r_{k+1} = r_k - \mathcal{P}_{\mathcal{Y}_k} r_k, \qquad k = 0, 1, \ldots. \tag{8.7}$$

Now, the energy splitting identity with $f = r_k$ gives us

$$\|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_k - \mathcal{P}_{\mathcal{Y}_k} r_k\|^2_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_{\mathcal{Y}_k} r_k\|^2_{\mathcal{N}_\Phi(\Omega)} \tag{8.8}$$

or, using the iteration formula (8.7),

$$\|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} + \|r_k - r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)}. \tag{8.9}$$

Therefore,

$$\begin{aligned}
\sum_{k=0}^{K} \|\mathcal{P}_{\mathcal{Y}_k} r_k\|^2_{\mathcal{N}_\Phi(\Omega)} &= \sum_{k=0}^{K} \|r_k - r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{k=0}^{K} \left\{ \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} - \|r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} \right\} \\
&= \|r_0\|^2_{\mathcal{N}_\Phi(\Omega)} - \|r_{K+1}\|^2_{\mathcal{N}_\Phi(\Omega)} \le \|r_0\|^2_{\mathcal{N}_\Phi(\Omega)},
\end{aligned}$$

which shows that the sequence of partial sums is monotone increasing and bounded, and therefore convergent – even for a poor choice of the sets $\mathcal{Y}_k$. If we can show that the residuals r_k converge to zero, then we would have that the iteratively computed approximation

$$s_{K+1} = \sum_{k=0}^{K} \mathcal{P}_{\mathcal{Y}_k} r_k = \sum_{k=0}^{K} (r_k - r_{k+1}) = r_0 - r_{K+1} \tag{8.10}$$

converges to the original interpolant $r_0 = \mathcal{P}_\mathcal{X} f$.

Remark: While this residual iteration algorithm has some structural similarities with the multilevel algorithm of Section 8.2 we now are considering a way to efficiently compute the interpolant $\mathcal{P}_\mathcal{X} f$ on some fine set $\mathcal{X}$ based on a single function Φ, whereas earlier, our final interpolant was the result of using the spaces $\bigcup_{k=1}^{K} \mathcal{N}_{\Phi_k}(\Omega)$, where Φ_k was an appropriately scaled version of the basis function Φ. And the goal there was to approximate f.

In order to prove convergence of the residual iteration let us assume that we can find sets of points $\mathcal{Y}_k$ such that at step k at least some fixed percentage of the energy of the residual is picked up by its interpolant, i.e.,

$$\|\mathcal{P}_{\mathcal{Y}_k} r_k\|^2_{\mathcal{N}_\Phi(\Omega)} \ge \gamma \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} \tag{8.11}$$

with some fixed $\gamma \in (0, 1]$. Then (8.9) and the iteration formula (8.7) imply

$$\|r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} - \|\mathcal{P}_{\mathcal{Y}_k} r_k\|^2_{\mathcal{N}_\Phi(\Omega)},$$

and therefore

$$\|r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} \le \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} - \gamma \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} = (1-\gamma) \|r_k\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

Applying this bound recursively yields (see [562])

Theorem 8.5.1 *If the choice of sets $\mathcal{Y}_k$ satisfies (8.11), then the residual iteration (8.10) converges linearly in the native space norm, and after K steps of iterative refinement there is an error bound*

$$\|r_0 - s_K\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_K\|^2_{\mathcal{N}_\Phi(\Omega)} \le (1-\gamma)^K \|r_0\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

This theorem has various limitations. In particular, the norm involves the function Φ which makes it difficult to find sets $\mathcal{Y}_k$ that satisfy (8.11). Moreover, the native space norm of the initial residual r_0 is not known, either. Therefore, using an equivalent discrete norm on the set $\mathcal{X}$, Schaback and Wendland establish an estimate of the form

$$\|r_0 - s_K\|^2_{\mathcal{X}} \le \frac{C}{c}\left(1 - \delta\frac{c^2}{C^2}\right)^{K/2} \|r_0\|^2_{\mathcal{X}},$$

where c and C are constants denoting the norm equivalence, i.e.,

$$c\|s\|_{\mathcal{X}} \le \|s\|_{\mathcal{N}_\Phi(\Omega)} \le C\|s\|_{\mathcal{X}}$$

for any $s \in \mathcal{N}_\Phi(\Omega)$, and where δ is a constant analogous to γ (but based on use of the discrete norm $\|\cdot\|_{\mathcal{X}}$ in (8.11)).

In [563] a basic version of this algorithm – where the sets $\mathcal{Y}_k$ consist of a single point – is described and tested. The resulting approximation yields the *best K-term approximation* to the interpolant. This idea is related to the concept of *greedy approximation algorithms* (see, e.g., [607]) and *sparse approximation* (see, e.g., [252]).

If the set $\mathcal{Y}_k$ consists only of a single point $\boldsymbol{y}_k$, then the partial interpolant $\mathcal{P}_{\mathcal{Y}_k} r_k$ is particularly simple, namely

$$\mathcal{P}_{\mathcal{Y}_k} r_k = \beta\Phi(\cdot, \boldsymbol{y}_k)$$

with

$$\beta = \frac{r_k(\boldsymbol{y}_k)}{\Phi(\boldsymbol{y}_k, \boldsymbol{y}_k)}.$$

The point $\boldsymbol{y}_k$ is picked to be the point in $\mathcal{X}$ where the residual is largest, i.e., $|r_k(\boldsymbol{y}_k)| = \|r_k\|_\infty$. For this choice of "set" $\mathcal{Y}_k$ we certainly satisfy the constraint (8.11). Moreover, the interpolation problem is (approximately) solved without having to invert any linear systems. The algorithm can be summarized as

Algorithm (Greedy one-point algorithm)

```
Input data locations $\mathcal{X}$, associated values of $f$, tolerance $\epsilon > 0$

Set initial residual $r_0 = \mathcal{P}_{\mathcal{X}} f$, initialize $s_0 = 0$, $e = \infty$, $k = 0$

Choose starting point $\boldsymbol{y}_k \in \mathcal{X}$

While $e > \epsilon$ do

    Set $\beta = \frac{r_k(\boldsymbol{y}_k)}{\Phi(\boldsymbol{y}_k, \boldsymbol{y}_k)}$
    For $1 \le i \le N$ do
```

```
            $r_{k+1}(\boldsymbol{x}_i) = r_k(\boldsymbol{x}_i) - \beta\Phi(\boldsymbol{x}_i, \boldsymbol{y}_k)$
            $s_{k+1}(\boldsymbol{x}_i) = s_k(\boldsymbol{x}_i) + \beta\Phi(\cdot, \boldsymbol{y}_k)$
            Find $e = \max_{\mathcal{X}} |r_{k+1}|$ and the point $\boldsymbol{y}_{k+1}$ where it occurs
        end
        Increment $k = k + 1$
    end
```

Remarks:

1. One advantage of this very simple (but fairly slow) algorithm is that no linear systems need to be solved. Nor are any matrix-vector multiplications required. This can be beneficial for problems that are very large (and possibly ill-conditioned), since in that situation the conjugate gradient method (which does use matrix-vector multiplications) may take very long.

2. For practical situations, e.g., for smooth radial basis functions and densely distributed points in $\mathcal{X}$ the convergence can be rather slow. In order to speed up the algorithm one should couple it with an algorithm that efficiently evaluates the residuals. If the basis functions are compactly supported, then a fast tree code algorithm can be used. Otherwise, fast multipole or fast Fourier transforms for non-uniform data can be used (see below for more details on these methods).

3. Schaback and Wendland [563] extend the simple greedy algorithm described above to a version that adaptively uses basis functions of varying scales.

Another iterative algorithm was suggested by Faul and Powell [212, 213]. From our earlier discussions we know that it is possible to express the radial basis function interpolant in terms of cardinal functions $u_j^*(\boldsymbol{x})$, $j = 1, \ldots, N$, i.e.,

$$\mathcal{P}f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j^*(\boldsymbol{x}).$$

The basic idea of the Faul-Powell algorithm is to use *approximate* cardinal functions instead. Of course, this will only give an approximate value for the interpolant, and therefore an iteration on the residuals is suggested to improve the accuracy of this approximation. As done several times before, the approximate cardinal functions ψ_j, $j = 1, \ldots, N$, are determined as linear combinations of the basis functions $\Phi(\cdot, \boldsymbol{x}_i)$, i.e.,

$$\psi_j = \sum_{i \in \mathcal{L}_j} b_{ji} \Phi(\cdot, \boldsymbol{x}_i), \tag{8.12}$$

where $\mathcal{L}_j$ is an index set consisting of n ($n \approx 50$) indices that are used to determine the approximate cardinal function. For example, the n nearest neighbors of $\boldsymbol{x}_j$ with some additional special points (as in Section 8.3.3) will do. For every $j = 1, \ldots, N$, the coefficients b_{ji} found as solution of the linear system

$$\psi_j(\boldsymbol{x}_k) = \delta_{jk}, \qquad k \in \mathcal{L}_j. \tag{8.13}$$

These approximate cardinal functions are computed in a pre-processing step.

In its simplest form the residual iteration can be formulated as

$$\begin{aligned} s^{(0)}(\boldsymbol{x}) &= \sum_{j=1}^{N} f(\boldsymbol{x}_j)\psi_j(\boldsymbol{x}) \\ s^{(k+1)}(\boldsymbol{x}) &= s^{(k)}(\boldsymbol{x}) + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - s^{(k)}(\boldsymbol{x}_j) \right] \psi_j(\boldsymbol{x}), \qquad k = 0, 1, \ldots. \end{aligned}$$

Instead of adding the contribution of all approximate cardinal functions at the same time, this is done in a three-step process in the Faul-Powell algorithm. To this end index sets $\mathcal{L}_j$, $j = 1, \ldots, N-n$, are chosen so that $\mathcal{L}_j \subseteq \{j, j+1, \ldots, N\}$ making sure that $j \in \mathcal{L}_j$. Also, one needs to ensure that the corresponding centers form an $(m-1)$-unisolvent set.

Now, in the first step we define $s_0^{(k)} = s^{(k)}$, and then iterate

$$s_j^{(k)} = s_{j-1}^{(k)} + \theta_j^{(k)} \psi_j, \qquad j = 1, \ldots, N-n, \tag{8.14}$$

with

$$\theta_j^{(k)} = \frac{\langle \mathcal{P}f - s_{j-1}^{(k)}, \psi_j \rangle_{\mathcal{N}_\Phi(\Omega)}}{\langle \psi_j, \psi_j \rangle_{\mathcal{N}_\Phi(\Omega)}}. \tag{8.15}$$

The stepsize $\theta_j^{(k)}$ is chosen so that the native space best approximation to the residual $\mathcal{P}f - s_{j-1}^{(k)}$ from the space span$\{\psi_j\}$ is added. Using the representation (8.12) of ψ_j in terms of the basis $\{\Phi(\cdot, \boldsymbol{x}_i) : i = 1, \ldots, N\}$, the reproducing kernel property of Φ, and the (local) cardinality property (8.13) of ψ_j we can calculate

$$\begin{aligned} \langle \psi_j, \psi_j \rangle_{\mathcal{N}_\Phi(\Omega)} &= \langle \psi_j, \sum_{i \in \mathcal{L}_j} b_{ji} \Phi(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \sum_{i \in \mathcal{L}_j} b_{ji} \langle \psi_j, \Phi(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \sum_{i \in \mathcal{L}_j} b_{ji} \psi_j(\boldsymbol{x}_i) = b_{jj}. \end{aligned}$$

Similarly, we get

$$\begin{aligned} \langle \mathcal{P}f - s_{j-1}^{(k)}, \psi_j \rangle_{\mathcal{N}_\Phi(\Omega)} &= \langle \mathcal{P}f - s_{j-1}^{(k)}, \sum_{i \in \mathcal{L}_j} b_{ji} \Phi(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \sum_{i \in \mathcal{L}_j} b_{ji} \langle \mathcal{P}f - s_{j-1}^{(k)}, \Phi(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \sum_{i \in \mathcal{L}_j} b_{ji} \left(\mathcal{P}f - s_{j-1}^{(k)} \right)(\boldsymbol{x}_i) \\ &= \sum_{i \in \mathcal{L}_j} b_{ji} \left(f(\boldsymbol{x}_i) - s_{j-1}^{(k)}(\boldsymbol{x}_i) \right). \end{aligned}$$

Therefore (8.14) and (8.15) can be written as

$$s_j^{(k)} = s_{j-1}^{(k)} + \frac{\psi_j}{b_{jj}} \sum_{i \in \mathcal{L}_j} b_{ji} \left(f(\boldsymbol{x}_i) - s_{j-1}^{(k)}(\boldsymbol{x}_i) \right), \qquad j = 1, \ldots, N-n.$$

In the second step the residual is interpolated on the remaining n points (collected via the index set $\mathcal{L}^*$). Thus, we find a function $\sigma^{(k)}$ such that

$$\sigma^{(k)}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) - s^{(k)}_{N-n}(\boldsymbol{x}_i), \qquad i \in \mathcal{L}^*,$$

and the approximation is updated, i.e.,

$$s^{(k+1)} = s^{(k)}_{N-n} + \sigma^{(k)}.$$

In the third step the residuals are updated, i.e.,

$$r_i^{(k+1)} = f(\boldsymbol{x}_i) - s^{(k+1)}(\boldsymbol{x}_i), \qquad i = 1, \ldots, N.$$

The outer iteration (on k) is now repeated unless the largest of these residuals is small enough.

We can summarize this algorithm as

Algorithm (Faul-Powell algorithm)

```
Input data locations $\mathcal{X}$, associated values of $f$, tolerance $\epsilon > 0$
Set $k = 0$ and $s_0^{(k)} = 0$
Compute residuals $r_i^{(k)} = f(\boldsymbol{x}_i) - s^{(k)}(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, and set $e = \max_{i=1,\ldots,N} |r_i^{(k)}|$.
While $e > \epsilon$ do
    For $1 \le j \le N - n$ do
        Update
            $s_j^{(k)} = s_{j-1}^{(k)} + \frac{\psi_j}{b_{jj}} \sum_{i \in \mathcal{L}_j} b_{ji} \left( f(\boldsymbol{x}_i) - s_{j-1}^{(k)}(\boldsymbol{x}_i) \right)$
    end
    Solve the interpolation problem
        $\sigma^{(k)}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) - s^{(k)}_{N-n}(\boldsymbol{x}_i), \qquad i \in \mathcal{L}^*$
    Update the approximation
        $s_0^{(k+1)} = s^{(k)}_{N-n} + \sigma^{(k)}$
    Compute new residuals and new value for $e$
    Increment $k = k + 1$
end
```

Faul and Powell prove that this algorithm converges to the solution of the original interpolation problem. Similar to some of the other algorithms (greedy one-point or preconditioned GMRES) one needs to make sure that the residuals are evaluated efficiently by using a fast multipole expansion, fast Fourier transform, or compactly supported functions. Since the approximate cardinal functions can be computed in a preprocessing step this evaluation along with the determination of the sets $\mathcal{L}_j$ is the most expensive operation in the algorithm.

Remark: In its most basic form the Krylov subspace algorithm of Faul and Powell can also be explained via a dual approach to the greedy residual iteration algorithm of Schaback and Wendland. Instead of defining appropriate sets of points $\mathcal{Y}_k$, in the Faul and Powell algorithm one picks certain subspaces S_k of the native space. In particular, if S_k is the one-dimensional space $S_k = \text{span}\{\psi_k\}$ (where ψ_k is a local Lagrange function as in Section 8.3.3) we get the algorithm described above. For more details see [563].

8.5.2 Fast Fourier Transforms

In the recent papers [354, 490, 508] by Kunis, Nieslony, Potts and Steidl use of the fast Fourier transform at nonuniformly spaced points was suggested as an efficient way to solve and evaluate radial basis function problems. The software package NFFT by the authors is available for free download [353]. A discussion of the actual NFFT software would go beyond the scope of this manuscript. Instead, we briefly describe how to use NFFTs and FFTs to evaluate expansions of the form

$$\mathcal{P}f(\boldsymbol{y}_j) = \sum_{k=1}^{N} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k) \tag{8.16}$$

simultaneously at many evaluation points $\boldsymbol{y}_j$, $j = 1, \ldots, M$. Direct summation requires $\mathcal{O}(MN)$ operations, while it can be shown that use of the NFFT reduces the cost to $\mathcal{O}(M + N)$ operations. Therefore, as is always the case with fast Fourier transforms, use of the algorithm will pay off for sufficiently many evaluations.

In their papers Nieslony, Potts and Steidl distinguish between kernels Φ that are singular and those that are non-singular. Singular kernels are C^∞ everywhere except at the origin and include examples such as

$$\frac{1}{r},\ \frac{1}{r^2},\ \log r,\ r^2 \log r,$$

where $r = \|\cdot\|$. Non-singular kernels are smooth everywhere such as Gaussians and (inverse) multiquadrics. We will restrict our discussion to this latter class.

The basic idea for the following algorithm is remarkably simple. It relies on the fact that the exponential $e^{-\alpha(\boldsymbol{y}_j - \boldsymbol{x}_k)}$ can be written as $e^{-\alpha \boldsymbol{y}_j} e^{\alpha \boldsymbol{x}_k}$. Moreover, the method applies to arbitrary kernels (which is in strong contrast to the fast multipole type methods discussed in the next section). One starts out by approximating the (arbitrary, but smooth) kernel using standard Fourier series, i.e.,

$$\Phi(\boldsymbol{x}) \approx \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \boldsymbol{x}}$$

with index set $I_n = \left[-\frac{n}{2}, \frac{n}{2}\right)^s$. The coefficients $b_{\boldsymbol{\ell}}$ are found by the discrete inverse Fourier transform

$$b_{\boldsymbol{\ell}} = \frac{1}{n^s} \sum_{\boldsymbol{k} \in I_n} \Phi\left(\frac{\boldsymbol{k}}{n}\right) e^{-2\pi i \boldsymbol{k}\boldsymbol{\ell}/n}.$$

Numerically, this task is accomplished with software for the standard (inverse) FFT (e.g., [245]).

Remark: Note that this definition of the Fourier transform (as well as the one below) is different from the one used in Chapter 2. However, in order to stay closer to the software packages, we adopt the notation used there.

Therefore,

$$\begin{aligned}
\mathcal{P}f(\boldsymbol{y}_j) &\approx \sum_{k=1}^{N} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell}(\boldsymbol{y}_j - \boldsymbol{x}_k)} \\
&= \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{2\pi i \boldsymbol{\ell}(\boldsymbol{y}_j - \boldsymbol{x}_k)}
\end{aligned}$$

Now, the exponential is split using the above mentioned property, i.e.,

$$\mathcal{P}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{-2\pi i \boldsymbol{\ell}\boldsymbol{x}_k} e^{2\pi i \boldsymbol{\ell}\boldsymbol{y}_j}.$$

This, however, can be viewed as a fast Fourier transform at non-uniformly spaced points, i.e.,

$$\mathcal{P}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} c_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell}\boldsymbol{y}_j}.$$

where the coefficients $c_{\boldsymbol{\ell}} = b_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}$ with

$$a_{\boldsymbol{\ell}} = \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{-2\pi i \boldsymbol{\ell}\boldsymbol{x}_k}$$

which is nothing but an inverse discrete Fourier transform at non-uniformly spaced points. These latter two transforms are dealt with numerically using the NFFT software.

Together, for the case of non-singular kernels Φ we have the following algorithm.

Algorithm (Fast Fourier transform evaluation)

```
For $\boldsymbol{\ell} \in I_n$
    Compute the coefficients
        $$b_{\boldsymbol{\ell}} = \frac{1}{n^s} \sum_{\boldsymbol{k} \in I_n} \Phi\left(\frac{\boldsymbol{k}}{n}\right) e^{-2\pi i \boldsymbol{k}\boldsymbol{\ell}/n}$$
    by inverse FFT.
```

```
        Compute the coefficients
$$a_{\boldsymbol{\ell}} = \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{-2\pi i \boldsymbol{\ell} \boldsymbol{x}_k}$$
        by inverse NFFT.
        Compute the coefficients $c_{\boldsymbol{\ell}} = a_{\boldsymbol{\ell}} b_{\boldsymbol{\ell}}$.
    end
    For $1 \le j \le M$
        Compute the values
$$\mathcal{P}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} d_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \boldsymbol{y}_j}$$
        by NFFT.
    end
```

Remarks:

1. In the papers [354, 490, 508] the authors also suggest a special boundary regularization in case the kernel does not decay fast enough, i.e., the kernel is large near the boundary of the domain.

2. Of course, this method will only provide an approximation of the expansion (8.16) and error estimates are provided in the literature (see, e.g., [490]).

3. While we only illustrated the use of (N)FFTs for the evaluation of radial sums it should be clear that this method can also be coupled with the algorithms of the previous sections (such as preconditioned GMRES, the "greedy" algorithm, or the Faul-Powell algorithm) to efficiently solve radial basis function interpolation systems.

A few examples of the use of fast Fourier transforms for the evaluation of approximate moving least squares approximations (quasi-interpolants) are given in Figures 8.3–8.5. The graphs on the left indicate ℓ_∞ approximation errors for a Franke-type function. The graphs on the right show the execution times in seconds for direct summation (solid lines) and FFT summations (dashed lines). The colors correspond to the three different types of kernels listed in Table 8.10 below. The red curves correspond to the Gaussians (listed in the $\mathcal{O}(h^2)$ column), green curves to the function in the $\mathcal{O}(h^4)$ column (Gaussian multiplied by a linear Laguerre polynomial), and blue curves to those in the $\mathcal{O}(h^6)$ column (Gaussian multiplied by a quadratic Laguerre polynomial).

The evaluation of the results listed in Figures 8.3–8.5 occurs at 10,001, 16,641, and 2,146,689 randomly distributed points in the unit square, respectively. The 3D experiments show that there is a cross-over value of about 1,000 evaluations at which the FFT approach becomes faster than the direct approach. For the one and two-dimensional experiments this cross-over point occurs much earlier and is not detectable in the figures.

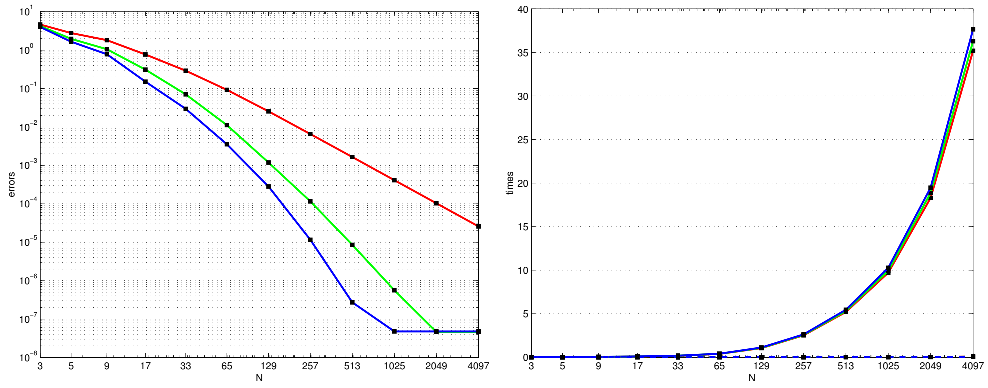

Figure 8.3: Convergence and execution times for 1D example.

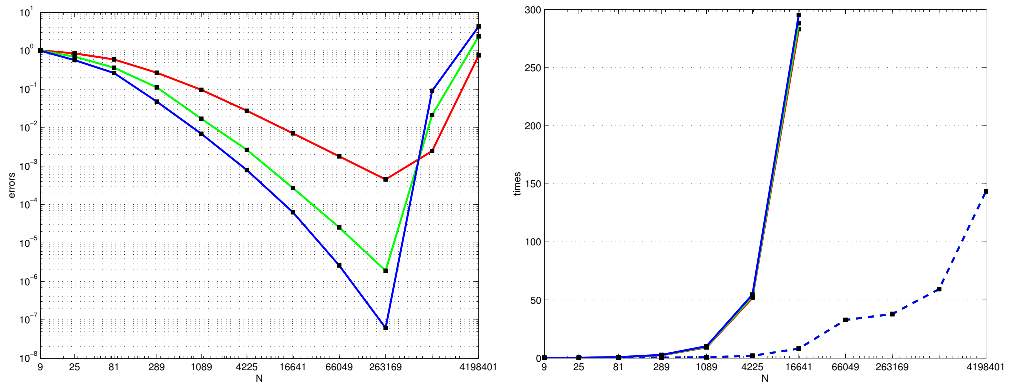

Figure 8.4: Convergence and execution times for 2D example.

The polynomial terms in Table 8.10 are given by generalized Laguerre polynomials with radial arguments. In general one can show (see, e.g., [434]) that if $L_d^{s/2}$ is used to denote the generalized Laguerre polynomial of degree d, then the smooth function f in $\mathbb{R}^s$ can be approximated with approximate approximation order $\mathcal{O}(h^{2d+2})$ by an expansion of the form

$$\mathcal{P}f(\boldsymbol{x}) = \frac{1}{(\pi\mathcal{D})^{s/2}} \sum_{k=1}^{N} f(\boldsymbol{x}_k) L_d^{s/2}\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_k\|^2}{\mathcal{D}h^2}\right) \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_k\|^2}{\mathcal{D}h^2}\right).$$

Here $\mathcal{D}$ is a parameter that controls a so-called *saturation error*, i.e., the predicted approximation order is achieved only up to some user-controllable threshold (and therefore referred to as *approximate approximation*). This threshold is clearly visible in the convergence graphs.

8.5.3 The Fast Multipole Method

Another quite popular strategy for dealing with fast summation problems is known as the *fast multipole method*. This method was first suggested by Greengard and Rokhlin in 1987 (see, e.g., the original paper [269], the popular discussion [268], or the short

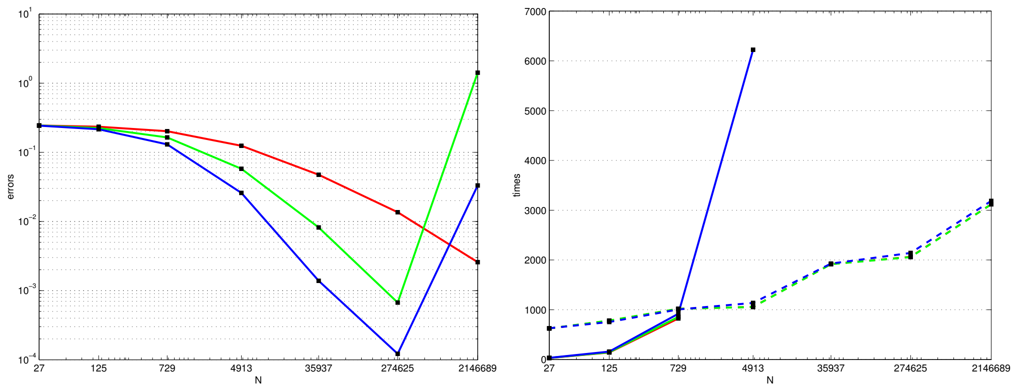

Figure 8.5: Convergence and execution times for 3D example.

s	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$
1	$e^{-\lvert x\rvert^2}$	$\left(\frac{3}{2}-\lvert x\rvert^2\right)e^{-\lvert x\rvert^2}$	$\left(\frac{15}{8}-\frac{5}{2}\lvert x\rvert^2+\frac{1}{2}\lvert x\rvert^4\right)e^{-\lvert x\rvert^2}$
2	$e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(2-\lVert \boldsymbol{x}\rVert^2\right)e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(3-3\lVert \boldsymbol{x}\rVert^2+\frac{1}{2}\lVert \boldsymbol{x}\rVert^4\right)e^{-\lVert \boldsymbol{x}\rVert^2}$
3	$e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(\frac{5}{2}-\lVert \boldsymbol{x}\rVert^2\right)e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(\frac{35}{8}-\frac{7}{2}\lVert \boldsymbol{x}\rVert^2+\frac{1}{2}\lVert \boldsymbol{x}\rVert^4\right)e^{-\lVert \boldsymbol{x}\rVert^2}$

Table 8.10: Generating functions for approximate MLS approximation in $\mathbb{R}^s$.

course tailored to radial basis functions [37]). It has quickly become very popular in the computational sciences. The breakthrough accomplishment of this algorithm was the ability to perform fast evaluations of sums of the type

$$\mathcal{P}f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \Phi(\boldsymbol{x}, \boldsymbol{x}_k), \qquad \boldsymbol{x} \in \mathbb{R}^s .$$

In particular, M such evaluations can be performed in $\mathcal{O}(M \log N)$ (or even $\mathcal{O}(M)$) operations instead of the standard $\mathcal{O}(MN)$ operations. The nonuniform fast Fourier transform of the previous section was able to do this also, and in a fairly general way for a very large class of kernels Φ, but the fast multipole method is a little older and may be more efficient since special expansions are used that are chosen with the particular kernel Φ in mind. We will now outline the basic idea of the *fast Gauss transform* [270]. This transform can be applied directly to the approximate moving least squares approximands based on Gaussians used in the previous section (see the numerical experiments reported in Table 8.11 below). The higher-order kernels consisting of Gaussians times Laguerre polynomials, however, require a completely new derivation.

Thus, using the abbreviation $\rho = \sqrt{\mathcal{D}}h$, we are now interested in a fast summation technique for M evaluations of the Gaussian quasi-interpolant (or *discrete Gauss transform*)

$$\mathcal{G}f(\boldsymbol{y}_j) = \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{-\lVert(\boldsymbol{y}_j - \boldsymbol{x}_k)/\rho\rVert^2}, \qquad j = 1, \ldots, M. \tag{8.17}$$

In [270] such an algorithm was developed, and in [591] a modification was suggested to cover also the case of variable scales ρ_k as needed for use with quasi-interpolation at scattered sites or with variable scales.

One of the central ingredients for the fast Gauss transform are the *multivariate Hermite functions*

$$h_{\boldsymbol{\alpha}}(\boldsymbol{x}) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} e^{-\|\boldsymbol{x}\|^2}, \tag{8.18}$$

which are related to the (multivariate) Hermite polynomials via

$$H_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{i=1}^{s} H_{\alpha_i}(x_i) = e^{\|\boldsymbol{x}\|^2} h_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{8.19}$$

(see, e.g., the univariate formula (6.1.3) in [5]). It is beneficial that the Hermite functions can be evaluated recursively using the (univariate) recurrence relation

$$\begin{aligned} h_{n+1}(x) &= 2xh_n(x) - 2nh_{n-1}(x), \qquad n = 1, 2, \ldots, \\ h_0(x) &= e^{-|x|^2}, \; h_1(x) = 2xe^{-|x|^2}, \end{aligned}$$

which follows immediately from (8.19) and the recursion relation for Hermite polynomials (see, e.g., formula (6.1.10) in [5]).

The algorithm of Greengard and Strain is based on three basic expansions which we list below as Theorems 8.5.2–8.5.4 (see [270, 271]). The main effect of these expansions is the fact that the variables $\boldsymbol{y}_j$ and $\boldsymbol{x}_k$ will be separated (this is the fundamental "trick" used with all fast summation algorithms). This will allow for the pre-computation and storage of certain *moments* below.

The first step in the algorithm is to scale the problem to the unit box $[0,1]^s$, and then subdivide the unit box into smaller boxes B and C which usually coincide. They can, however, also differ. The boxes B contain the *sources* $\boldsymbol{x}_k$, and the boxes C the *targets* $\boldsymbol{y}_j$. For each source box B one then determines its *interaction region* $IR(B)$. The interaction region of B is a set of nearest neighbors of B such that the error of truncating the sum over all boxes is below a certain threshold. Due to the fast decay of the Gaussians it is suggested (see [271]) to use the 9^s nearest neighbors for single precision and the 13^s nearest neighbors for double precision.

Theorem 8.5.2 *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ which lie in a box B with center $\boldsymbol{x}_B$ and side length ρ, and let $\boldsymbol{y}_C$ be the center of the target box C $(\in IR(B))$ of radius r_c containing the targets $\boldsymbol{y}_j$. Then the Gaussian field due to the sources in B,*

$$\mathcal{G}^{(B)} f(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) e^{-\|(\boldsymbol{y}_j - \boldsymbol{x}_k)/\rho\|^2},$$

has the following Taylor expansion about $\boldsymbol{y}_C$:

$$\mathcal{G}^{(B)} f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} A_{\boldsymbol{\alpha}}^{(B)} \left(\frac{\boldsymbol{y}_j - \boldsymbol{y}_C}{\rho} \right)^{\boldsymbol{\alpha}}, \tag{8.20}$$

where the coefficients $A_{\boldsymbol{\alpha}}^{(B)}$ are given by

$$A_{\boldsymbol{\alpha}}^{(B)} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) h_{\boldsymbol{\alpha}} \left(\frac{\boldsymbol{x}_k - \boldsymbol{y}_C}{\rho} \right).$$

The error $E_T(p)$ due to truncating the series (8.20) after the p-th order terms satisfies the bound

$$|E_T(p)| = |\sum_{\boldsymbol{\alpha}>p} A_{\boldsymbol{\alpha}}^{(B)} \left(\frac{\boldsymbol{y}_j - \boldsymbol{y}_C}{\rho}\right)^{\boldsymbol{\alpha}}| \le (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!}^s} \left[\frac{\left(\frac{r_c}{\rho}\right)^{p+1}}{1-\frac{r_c}{\rho}}\right]^s ,$$

where $F^{(B)} = \sum_{k\in I_B} |f(\boldsymbol{x}_k)|$.

Remark: Here we used the multi-index notation $\boldsymbol{\alpha} \ge 0$ to denote the constraints $\alpha_i \ge 0$ for all $i = 1, \ldots, s$. More generally, for some integer p $\boldsymbol{\alpha} \ge p$ if $\alpha_i \ge p$ for all $i = 1, \ldots, s$. This implies $\boldsymbol{\alpha} > p$ for some integer p if $\boldsymbol{\alpha} \ge p$ and $\alpha_i > p$ for some i. We also use $\boldsymbol{\alpha} \ge \boldsymbol{\beta}$ if $\alpha_i \ge \beta_i$ for all $i = 1, \ldots, s$.

The expansion (8.20) will be used in the case when the source box B contains relatively few sources, but the target box C contains many targets.

By reversing the role of the Hermite functions and the shifted monomials one can write a single Gaussian as a multivariate Hermite expansion about a point $\boldsymbol{z}_0 \in \mathbb{R}^s$, i.e.,

$$e^{-\|(\boldsymbol{y}_j-\boldsymbol{x}_k)/\rho\|^2} = \sum_{\boldsymbol{\alpha}\ge 0} \frac{1}{\boldsymbol{\alpha}!} \left(\frac{\boldsymbol{x}_k - \boldsymbol{z}_0}{\rho}\right)^{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{z}_0}{\rho}\right). \tag{8.21}$$

This is used in

Theorem 8.5.3 *(Far-field expansion) Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ which lie in a box B with center $\boldsymbol{x}_B$ and side length ρ. Then the Gaussian field due to the sources in B,*

$$\mathcal{G}^{(B)} f(\boldsymbol{y}_j) = \sum_{k\in I_B} f(\boldsymbol{x}_k) e^{-\|(\boldsymbol{y}_j-\boldsymbol{x}_k)/\rho\|^2},$$

is equal to an Hermite expansion about $\boldsymbol{x}_B$:

$$\mathcal{G}^{(B)} f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha}\ge 0} B_{\boldsymbol{\alpha}}^{(B)} h_{\boldsymbol{\alpha}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{x}_B}{\rho}\right). \tag{8.22}$$

The moments $B_{\boldsymbol{\alpha}}^{(B)}$ are given by

$$B_{\boldsymbol{\alpha}}^{(B)} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k\in I_B} f(\boldsymbol{x}_k) \left(\frac{\boldsymbol{x}_k - \boldsymbol{x}_B}{\rho}\right)^{\boldsymbol{\alpha}}.$$

The error $E_H(p)$ due to truncating the series (8.22) after p-th order terms satisfies the bound

$$|E_H(p)| = |\sum_{\boldsymbol{\alpha}>p} B_{\boldsymbol{\alpha}}^{(B)} h_{\boldsymbol{\alpha}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{x}_B}{\rho}\right)| \le (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!}^s} \left[\frac{\left(\frac{r_c}{\rho}\right)^{p+1}}{1-\frac{r_c}{\rho}}\right]^s .$$

Theorem 8.5.3 is used when B contains many sources, but C only few targets. Finally, in the case when both B and C contain relatively many points we use

Theorem 8.5.4 *(Translation operation) Let the sources $\boldsymbol{x}_k$ lie in a box B with center $\boldsymbol{x}_B$ and side length ρ and let $\boldsymbol{y}_j$ be an evaluation point in a box C with center $\boldsymbol{y}_C$. Then the corresponding truncated Hermite expansion (8.22) can be expanded as a Taylor series of the form*

$$\mathcal{G}^{(BC)} f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\beta}\geq 0} C_{\boldsymbol{\beta}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{y}_C}{\rho}\right)^{\boldsymbol{\beta}}. \tag{8.23}$$

The coefficients $C_{\boldsymbol{\beta}}$ are given by

$$C_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{\boldsymbol{\alpha}\leq p} B_{\boldsymbol{\alpha}}^{(B)} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\frac{\boldsymbol{x}_B - \boldsymbol{y}_C}{\rho}\right),$$

with $B_{\boldsymbol{\alpha}}^{(B)}$ as in Theorem 8.5.3. The error $E_T(p)$ due to truncating the series (8.23) after p-th order terms satisfies the bound

$$|E_T(p)| = |\sum_{\boldsymbol{\beta}>p} B_{\boldsymbol{\beta}}^{(B)} \left(\frac{\boldsymbol{x} - \boldsymbol{y}_C}{\rho}\right)^{\boldsymbol{\beta}}| \leq (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!}^s} \left[\frac{\left(\frac{r_c}{\rho}\right)^{p+1}}{1 - \frac{r_c}{\rho}}\right]^s.$$

Theorem 8.5.4 is based on the multivariate Taylor series expansion of the Hermite functions $h_{\boldsymbol{\alpha}}$

$$h_{\boldsymbol{\alpha}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{x}_B}{\rho}\right) = \sum_{\boldsymbol{\beta}\geq 0} \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \left(\frac{\boldsymbol{y}_j - \boldsymbol{y}_C}{\rho}\right)^{\boldsymbol{\beta}} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\frac{\boldsymbol{x}_B - \boldsymbol{y}_C}{\rho}\right).$$

Remarks:

1. The error estimates in the original paper on the fast Gauss transform [270] were incorrect. In the mean time a number of other authors have provided alternate error bounds in their papers (see, e.g., [31, 225, 271, 637]).

2. For 1D calculations on the order of $p = 20$ terms are required to achieve double precision accuracy. For the 2D one can get by with a smaller value of p (about 15), but the number of terms is of course much higher (on the order of p^s for s-dimensional problems).

The basic outline of the algorithm is as follows:

Algorithm:

1. If necessary, scale the problem so that the coarsest box $B_0 = [0, 1]^s$. Subdivide B_0 into smaller boxes with side length ρ parallel to the axes. Assign each source $\boldsymbol{x}_k$ to the box B in which it lies and each evaluation point $\boldsymbol{y}_j$ to the box C in which it lies.

2. Choose p so that the truncation error satisfies the desired accuracy, and for each box B compute and store the coefficients (or moments)

$$B_{\boldsymbol{\alpha}}^{(B)} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k\in I_B} f(\boldsymbol{x}_k) \left(\frac{\boldsymbol{x}_k - \boldsymbol{x}_B}{\rho}\right)^{\boldsymbol{\alpha}}, \qquad \boldsymbol{\alpha} \leq p,$$

of its Hermite expansion (8.22).

3. For each evaluation box C, determine its interaction region $IR(C)$.

4. For each evaluation box C transform all Hermite expansions in source boxes within the interaction region $IR(C)$ into a single Taylor expansion using (8.23), i.e.,

$$\mathcal{G}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta} \le p} C_{\boldsymbol{\beta}} \left(\frac{\boldsymbol{y}_j - \boldsymbol{y}_C}{\rho} \right)^{\boldsymbol{\beta}},$$

where

$$C_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{B \in IR(C)} \sum_{\boldsymbol{\alpha} \le p} B_{\boldsymbol{\alpha}}^{(B)} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\frac{\boldsymbol{x}_B - \boldsymbol{y}_C}{\rho} \right).$$

For a small number of points direct summation is more efficient than the fast transform. For the case $s = 1$ the fast Gauss transform should be preferable to direct summation for $N \approx 1000$ (with $M \approx 2000$ evaluation points). The break-even point in $\mathbb{R}^2$ is at about $N = 12000$ (with $M = 24000$). In particular, in $\mathbb{R}^3$, it is rather disappointing that the break-even point may not occur until about $N = 270000$ data sites (with $M = 2.16 \times 10^6$ evaluation points). This makes fast Gauss transform in its basic form virtually unusable for 3D applications.

Note that this algorithm does not use a hierarchical decomposition of space as is typical for so-called *tree codes*, as well as many other more general fast multipole algorithms. In this algorithm the interaction region is determined simply based on the fast decay of the Gaussian.

Clearly, the most work is involved in step 4. The performance of this step can be improved by using *plane wave* expansions to diagonalize the translation operators (see [271]). In order to keep matters as simple as possible, we will not discuss this feature.

A more complete algorithm (designed for radial basis function interpolation) has been developed by Beatson and co-workers (see, e.g., [43, 137]).

The numerical experiments in Table 8.11 were conducted by performing quasi-interpolation of the form

$$\mathcal{Q}_h f(x) = \mathcal{D}^{-1/2} \sum_{k=1}^{N} f(x_k) \psi \left(\frac{x - x_k}{\sqrt{\mathcal{D}} h} \right),$$

with a Gaussian ψ on $N = 2^\kappa + 1$ equally spaced points in $[0, 1]$ with the mollified test function

$$f(x) = 15 e^{\frac{-.25}{.25-(x-.5)^2}} \left[\frac{3}{4} e^{-(x-2)^2/4} + \frac{3}{4} e^{-(x+1)^2/49} + \frac{1}{2} e^{-(x-7)^2/4} - \frac{1}{5} e^{-(x-4)^2} \right].$$

All errors were computed on $M = 524289$ equally spaced points in $[0, 1]$. In the "order" column we list the number $order = \ln(e_{\kappa-1}/e_\kappa)/\ln 2$ corresponding to the exponent in the $\mathcal{O}(h^{order})$ notation. Other parameters were $\mathcal{D} = 4$, and the default values for the code of [225] (i.e., $R = 0.5$). All times are measured in seconds.

The asterisk * on the entries in the lower part of the direct column indicate estimated times. The fast Gauss transform yields a speedup of roughly a factor of 300. Another way to interpret these results is that for roughly the same amout of work we can obtain

	direct			fast			
N	ℓ_∞ error	order	time	ℓ_∞ error	order	time	speedup
5	3.018954e-00		1.93	5.495125e-00		1.07	1.80
9	2.037762e-00	0.57	3.40	2.037762e-00	1.43	5.31	0.64
17	9.617170e-01	1.08	6.39	9.617170e-01	1.08	5.33	1.20
33	3.609205e-01	1.41	12.28	3.609205e-01	1.41	5.35	2.30
65	1.190192e-01	1.60	24.72	1.190192e-01	1.60	5.39	4.59
129	3.354132e-02	1.83	53.38	3.354132e-02	1.83	5.46	10.14
257	8.702868e-03	1.95	113.35	8.702868e-03	1.95	5.61	20.20
513	2.196948e-03	1.99	226.15	2.196948e-03	1.99	5.94	38.07
1025			450*	5.505832e-04	2.00	6.67	67.47
2049			900*	1.377302e-04	2.00	7.87	114.36
4097			1800*	3.443783e-05	2.00	10.56	170.45
8193			3600*	8.609789e-06	2.00	15.78	228.14
16385			7200*	2.152468e-06	2.00	26.27	274.08
32769			14400*	5.381182e-07	2.00	47.39	303.86
65537			28800*	1.345296e-07	2.00	89.91	320.32
131073			57600*	3.363241e-08	2.00	174.74	329.63
262145			115200*	8.408103e-09	2.00	343.59	335.28

Table 8.11: 1D quasi-interpolation using fast Gauss transform.

an answer which is about 100000 times more accurate. The $\mathcal{O}(h^2)$ convergence of the Gaussian quasi-interpolant is perfectly illustrated by the entries in the "order" columns.

An alternative to fast multipole methods are so-called *fast tree codes*. These kind of algorithms originated in computational chemistry. We recommend recent papers by Krasny and co-workers (e.g., [160, 385]). The advantage of these kinds of methods is that they make use of standard Taylor expansions instead of the specialized expansions (such as, e.g, in terms of Hermite functions, spherical harmonics, spherical Hankel functions, plane waves, or hypergeometric functions [137]). This simplifies their implementation. However, their convergence properties are probably not as good as for fast mutipole expansions.

We now present a very general discussion of fast summation via Taylor expansions. The presentation of this material is motivated by the work of Krasny and co-workers (see, e.g., [160, 385]) as well as the algorithm for the fast Gauss transform reviewed above. Since we are interested in many evaluations of our quasi-interpolants (or other radial basis function expansion), we split the set of M evaluation points $\boldsymbol{y}_j$ into groups (contained in boxes C with centers $\boldsymbol{y}_C$). We also split the N data locations $\boldsymbol{x}_k$ into boxes B with center $\boldsymbol{x}_B$, and use the index set I_B to denote the points in B.

In order to set the stage for a fast summation of the quasi-interpolant

$$\begin{aligned}\mathcal{Q}f(\boldsymbol{y}_j) &= \sum_{k=1}^{N} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)\\ &= \sum_{B}\sum_{k\in I_B} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k) \qquad (8.24)\end{aligned}$$

with generating function Φ we require the multivariate Taylor expansion of Φ about a point $\boldsymbol{z}_0 \in \mathbb{R}^s$, i.e.,

$$\Phi(\boldsymbol{z}) = \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{z}_0} \frac{(\boldsymbol{z}-\boldsymbol{z}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}, \tag{8.25}$$

where $\boldsymbol{\alpha}$ is a multi-index. Now – as for the fast Gauss transform – we consider three basic expansions.

Theorem 8.5.5 *(Taylor Series Expansion about Centers of Target Boxes) Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ which lie in a box B with center $\boldsymbol{x}_B$, and let $\boldsymbol{y}_C$ be the center of the target box C containing an evaluation point $\boldsymbol{y}_j$. Then the quasi-interpolant due to sources in B*

$$\mathcal{Q}^{(B)} f(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) \Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be written as a Taylor expansion about $\boldsymbol{y}_C$:

$$\mathcal{Q}^{(B)} f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} A_{\boldsymbol{\alpha}}^{(B)} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}},$$

where

$$A_{\boldsymbol{\alpha}}^{(B)} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k)$$

with $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C - \boldsymbol{x}_k}$.

Proof: We combine the contribution for the source box B of (8.24) with (8.25), and let $\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0 = \boldsymbol{y}_C - \boldsymbol{x}_k$. Then (8.24) becomes

$$\mathcal{Q}^{(B)} f(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C - \boldsymbol{x}_k} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C - \boldsymbol{x}_k}$ we can rewrite this as

$$\mathcal{Q}^{(B)} f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} A_{\boldsymbol{\alpha}}^{(B)} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}},$$

where

$$A_{\boldsymbol{\alpha}}^{(B)} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k).$$

□

Example : If we take $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ then

$$T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = h_{\boldsymbol{\alpha}}(\boldsymbol{y}_C - \boldsymbol{x}_k) = h_{\boldsymbol{\alpha}}(\boldsymbol{x}_k - \boldsymbol{y}_C),$$

and Theorem 8.5.5 is equivalent to Theorem 8.5.2 given above.

Remark: We can see that the Taylor expansion has allowed us to separate the evaluation points $\boldsymbol{y}_j$ from the data points $\boldsymbol{x}_k$.

Theorem 8.5.6 *(Reversed Taylor Series Expansion about Centers of Source Boxes) Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ which lie in a box B with center $\boldsymbol{x}_B$. Then the quasi-interpolant due to sources in B*

$$\mathcal{Q}^{(B)}f(\boldsymbol{y}_j) = \sum_{k\in I_B} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be written as a reversed Taylor expansion about $\boldsymbol{x}_B$:

$$\mathcal{Q}^{(B)}f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha}\geq 0} B_{\boldsymbol{\alpha}}^{(B)} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B),$$

with the moments $B_{\boldsymbol{\alpha}}^{(B)}$ *given by*

$$B_{\boldsymbol{\alpha}}^{(B)} = \frac{1}{\boldsymbol{\alpha}!}\sum_{k\in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}},$$

and $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_j-\boldsymbol{x}_B}$.

Proof: We combine the contribution for the source box B of (8.24) with (8.25), and let $\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0 = \boldsymbol{y}_j - \boldsymbol{x}_B$. Then (8.24) becomes

$$\mathcal{Q}^{(B)}f(\boldsymbol{y}_j) = \sum_{k\in I_B} f(\boldsymbol{x}_k)\sum_{\boldsymbol{\alpha}\geq 0} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_j-\boldsymbol{x}_B}(-1)^{|\boldsymbol{\alpha}|}\frac{(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_j-\boldsymbol{x}_B}$ we can reverse the role of the Taylor coefficients and the polynomials to write this as

$$\mathcal{Q}^{(B)}f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha}\geq 0} B_{\boldsymbol{\alpha}}^{(B)} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B),$$

with

$$B_{\boldsymbol{\alpha}}^{(B)} = \frac{1}{\boldsymbol{\alpha}!}\sum_{k\in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}}.$$

$\square$

Example: Using $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ this is equivalent to Theorem 8.5.3.

Remark: The moments can be pre-computed and stored during the setup phase of the algorithm.

Theorem 8.5.7 *(Conversion of Taylor Series Expansions about Source Centers to Series about Target Centers) Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ which lie in a box B with center $\boldsymbol{x}_B$, and let $\boldsymbol{y}_C$ be the center of the target box C containing $\boldsymbol{y}_j$. Then a fast summation formula for the quasi-interpolant*

$$\mathcal{Q}f(\boldsymbol{y}_j) = \sum_{k=1}^{N} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be given as an expansion about $\boldsymbol{y}_C$*:*

$$\mathcal{Q}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta}\le p} C_{\boldsymbol{\beta}}(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\beta}},$$

where

$$C_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta}\le p} \sum_{B} T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) B_{\boldsymbol{\alpha}}^{(B)},$$

$T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}+\boldsymbol{\beta}|} D^{\boldsymbol{\alpha}+\boldsymbol{\beta}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C-\boldsymbol{x}_B}$*, and the moments* $B_{\boldsymbol{\alpha}}^{(B)}$ *are as in Theorem 8.5.6.*

Proof: We combine (8.24) with (8.25), and now replace $\boldsymbol{z}$ by $\boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0$ by $\boldsymbol{y}_C - \boldsymbol{x}_B$. Then (8.24) becomes

$$\mathcal{Q}f(\boldsymbol{y}_j) = \sum_{B} \sum_{k\in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha}\ge 0} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C-\boldsymbol{x}_B} \frac{(\boldsymbol{y}_j - \boldsymbol{x}_k - (\boldsymbol{y}_C - \boldsymbol{x}_B))^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C-\boldsymbol{x}_B}$ along with the multivariate binomial theorem we can rewrite this as

$$\begin{aligned}
\mathcal{Q}f(\boldsymbol{y}_j) &= \sum_{B} \sum_{k\in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha}\ge 0} (-1)^{|\boldsymbol{\alpha}|} \frac{T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B)}{\boldsymbol{\alpha}!} \sum_{\boldsymbol{\beta}\le\boldsymbol{\alpha}} \binom{\boldsymbol{\alpha}}{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}} (\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}} \\
&= \sum_{\boldsymbol{\alpha}\ge 0} \sum_{B} (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta}\le\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} \sum_{k\in I_B} f(\boldsymbol{x}_k) \frac{(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}}}{\boldsymbol{\beta}!}.
\end{aligned}$$

In fact, we can introduce the moments of Theorem 8.5.6 and write

$$\mathcal{Q}f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha}\ge 0} \sum_{B} (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta}\le\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} B_{\boldsymbol{\beta}}^{(B)},$$

where

$$B_{\boldsymbol{\beta}}^{(B)} = \frac{1}{\boldsymbol{\beta}!} \sum_{k\in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}}.$$

A fast algorithm is now obtained by truncating the infinite series after the p-th order terms, i.e.,

$$\mathcal{Q}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\alpha}\le p} \sum_{B} (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta}\le\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} B_{\boldsymbol{\beta}}^{(B)}.$$

Using the fact that

$$\sum_{\boldsymbol{\alpha}\le p} a_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}\le\boldsymbol{\alpha}} b_{\boldsymbol{\alpha}-\boldsymbol{\beta}} = \sum_{\boldsymbol{\alpha}\le p} b_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\alpha}\le\boldsymbol{\beta}\le p} a_{\boldsymbol{\beta}} = \sum_{\boldsymbol{\alpha}\le p} b_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta}\le p} a_{\boldsymbol{\alpha}+\boldsymbol{\beta}},$$

which can be verified by a simple rearrangement of the summations and an index transformation, we obtain (interchanging the role of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) the following fast summation formula:

$$\mathcal{Q}f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta}\le p} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta}\le p} (-1)^{|\boldsymbol{\alpha}|} \frac{1}{\boldsymbol{\beta}!} \sum_{B} (-1)^{|\boldsymbol{\alpha}+\boldsymbol{\beta}|} T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) B_{\boldsymbol{\alpha}}^{(B)} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\beta}}.$$

This is equivalent to the statement of the theorem. □

Example: Using $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ this is almost equivalent to Theorem 8.5.3. However, our alternate formula is more efficient since only Hermite functions up to order p are required (as opposed to order $2p$ in the Greengard/Strain version). This gain is achieved by using the binomial theorem instead of a second Taylor expansion (the Hermite series expansion used in the traditional fast Gauss transform is equivalent to a Taylor expansion).

Remarks:

1. Note that the Taylor coefficients $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B)$ depend only on the box centers $\boldsymbol{y}_C$ and $\boldsymbol{x}_B$.

2. In order to make the algorithm efficient one will use a decision rule (as in Strain's code for the fast Gauss transform) to decide when to use which of the three expansions. Error estimation is very similar to Greengard/Strain. The only difference is that one needs bounds on the Taylor coefficients instead of the Hermite functions.

3. In order to adapt this fast transform to Gauss-Laguerre generating functions of the previous sections (or any other generating function) one needs to compute the required Taylor coefficients. This is a task that goes beyond the scope of this manuscript.

8.6 Domain Decomposition

Finally, another method commonly used to deal with large computational problems is the *domain decomposition* method. The domain decomposition method is frequently implemented on parallel computers in order to speed up the computation even more. A standard reference (based mostly on finite difference and finite element methods) is the book by Smith, Bjørstad and Gropp [584]. For radial basis functions there is a recent paper by Beatson, Light and Billings [42].

The main aim of the paper [42] is to solve the radial basis function interpolation problem discussed multiple times in previous sections. In particular, a so-called *multiplicative Schwarz* algorithm (which is analogous to Gauss-Seidel iteration) is presented, and linear convergence of the algorithm is proved. A section with numerical experiments reports results for an *additive Schwarz* method (which is analogous to Jacobi iteration).

In particular, the authors implemented polyharmonic radial basis functions, and used the scale invariant basis discussed in Section 8.4.

The classical additive Schwarz algorithm is usually discussed in the context of partial differential equations, and it is known that one should add a coarse level correction in order to ensure convergence and to filter out some of the low-frequency oscillations (see, e.g., [584]).

In [42] a two-level additive algorithm for interpolation problems was presented. One begins by subdividing the set on interpolation point $\mathcal{X}$ into M smaller sets $\mathcal{X}_i$, $i = 1, \ldots, M$, whose pairwise intersection is non-empty. The points that belong to one set $\mathcal{X}_i$ only are called *inner points* of $\mathcal{X}_i$. Those points in the intersection of more than one set need to be assigned in some way as inner points to only one of the subsets $\mathcal{X}_i$ so that the collection of all inner points yields the entire set $\mathcal{X}$. This corresponds to the concept of *overlapping domains*. One also needs to choose a coarse grid $\mathcal{Y}$ that contains points from all of the inner point sets.

In the setup phase of the algorithm the radial basis function interpolation matrices for the smaller problems on each of the subsets $\mathcal{X}_i$, $i = 1, \ldots, M$, are computed and factored. At this point one can use the homogeneous basis of Section 8.4 to ensure numerical stability. Now the algorithm proceeds as follows:

Algorithm:

```
Input: Data $f$, point sets $\mathcal{X}_i$ and factored interpolation matrices $A_i$, $i = 1, \ldots, M$,
tolerance $\epsilon$

Initialize $r = f$, $s = 0$

While $\|r\| > \epsilon$ do

    For $i = 1$ to $m$ (i.e., for each subset $\mathcal{X}_i$) do
        Determine the coefficients $\boldsymbol{c}_i$ of the interpolant to the residual $r|_{\mathcal{X}_i}$ on
        $\mathcal{X}_i$.
    end
    Make $\boldsymbol{c}$ orthogonal to $\Pi^s_{m-1}$.
    Assemble an intermediate approximation $s_1 = \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j)$.
    Compute the residual on the coarse grid, i.e.,
        $r_1 = r - s_1|_{\mathcal{Y}}$.
    Interpolate to $r_1$ on the coarse grid $\mathcal{Y}$ using a radial basis function expansion
    $s_2$.
    Update $s = s + s_1 + s_2$.
    Reevaluate the global residual $r = f - s$ on the whole set $\mathcal{X}$

end
```

Remarks:

1. In [42] it is proved that a multiplicative version of this algorithm converges at least linearly. However, the additive version can be more easily implemented on a parallel computer.

2. If strictly positive definite kernels such as Gaussians are used, then it is not necessary to make the coefficients $\boldsymbol{c}$ orthogonal to polynomials.

3. As in many algorithms before, the evaluation of the residuals needs to be made "fast" using either a fast multipole method or a version of the fast Fourier transform.

4. In the case of very large data sets it may be necessary to use more than two levels so that one ends up with a *multigrid* algorithm.

5. The authors of [42] report having solved interpolation problems with several millions of points using the domain decomposition algorithm above.

6. A number of other papers discussing domain decomposition methods for radial basis functions have recently appeared in the literature (see, e.g., [166, 306, 313, 379, 396, 647]). However, most of these papers contain little theory, focussing mostly on numerical experiments.