Radial Basis Functions in Scientific Computing

by

Leevan Ling

B.Sc. (Hons), Simon Fraser University, 1997

M.Sc., Simon Fraser University, 1999

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in the Department
of
Mathematics

SIMON FRASER UNIVERSITY

August 2003

APPROVAL

Name: Leevan Ling

Degree: Doctor of Philosophy

Title of Thesis: Radial Basis Functions in Scientific Computing

Examining Committee: Dr. T. C. Brown
Chair

Dr. M. R. Trummer, Senior Supervisor

Dr. R. D. Russell

Dr. S. J. Ruuth

Dr. M. C. Kropinski, Internal Examiner

Dr. E. J. Kansa, External Examiner

Date Approved: April 25, 03

PARTIAL COPYRIGHT LICENCE

Title of Thesis/Project/Extended Essay

Radial Basis Functions in Scientific Computing

Author:

_____ _____
(signature)

_____ _____
(name)

April 25, 2003
(date)

Abstract

Radial basis functions (RBFs) have originally been used as an effective tool to interpolate data. The basis functions (all translations of one radial symmetric function) can be centered at arbitrary points in the domain of interest, hence providing a truly meshfree approximation, i.e., a wireframe parametric surfaces is not required. For certain types of RBFs exponential convergence has been shown.

This thesis deals with various aspects of using RBFs in scientific computing problems. The first part treats the interpolation problem: instead of solving the interpolation equations, however, a quasi-interpolation procedure provides an approximate interpolation formula without solving any matrix system. We propose a multilevel scheme for achieving better smoothness of an existing univariate quasi-interpolation scheme for RBF, and prove its convergence. Then we extend the results to higher dimensions, and study a boundary padding technique that improves accuracy.

Over the last decade or so, researchers have investigated the use of RBFs in the numerical solutions of ordinary (ODEs) and partial differential equations (PDEs). In the second part of this thesis we introduce a new RBF collocation method for singularly perturbed boundary value problems (we implement this method for two-point boundary value problems). Our method can achieve high accuracy even for extremely thin layers, $\epsilon = \mathcal{O}(10^{-12})$. We then modify our scheme to become adaptive, which results in a very robust method without the need for fine tuning parameters.

In general, when using RBFs to solve PDEs, the resulting matrices are full and severely ill-conditioned. Therefore, applications of RBF-PDE were once restricted to problems of moderate size solved with direct methods. The third part of this thesis

addresses this problem with a preconditioning technique based on approximate cardinal basis functions (ACBFs). We then couple the ACBF preconditioning technique with the domain decomposition method (DDM) which allows one to solve large-scale PDE problems in parallel.

Acknowledgments

I am grateful for the support and guidance of my supervisor Manfred Trummer. There are many people within the Department of Mathematics at Simon Fraser University that make my education fruitful and enjoyable. I want to thank Robert Russell, Steven Ruuth, Mary Catherine Kropinski, Cecil Graham, Tao Tang, and Anadijiban Das. I also thank Diane Pogue for her administrative help.

I say a big thank you to Edward Kansa at the Lawrence Livermore National Laboratory who invited me to the field of radial function. I wish to thank Rick Beatson at the University of Canterbury, Anne Greenbaum at the University of Washington, and Jichun Li at the University of Nevada in Las Vegas for their valuable comments on the fast matrix-vector multiplication methods, on the QR updating scheme, and on the domain decomposition methods, respectively. Furthermore, I thank the RBF researchers who have also played a part to this thesis; in particular, Benny Y. C. Hon at the City University of Hong Kong, and C. S. Chen at the University of Nevada in Las Vegas.

I would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada and the Department of Mathematics for financial support. Thanks also to the Simon Fraser University for financial support in the form of Faculty of Sciences Graduate Fellowships and a President's Ph.D. Research Stipend.

This thesis contains results from six peer-refereed journal articles; to date,

- L. Ling, A Univariate Quasi Multiquadric Interpolation with Better Smoothness, *Computers and Mathematics with Applications*, To Appear.

- L. Ling, A Multidimensional Quasi-Interpolation formula with Dimension-splitting

Multiquadric Basis, *Applied Mathematics and Computation*, In Review.

- L. Ling, and M. R. Trummer, Multiquadric Collocation Method With Integral Formulation For Boundary Layer Problems, *Computers and Mathematics with Applications*, To Appear.

- L. Ling, and M. R. Trummer, Adaptive Multiquadric Collocation For Boundary Layer Problems, *Mathematics of Computation*, In Review.

- L. Ling, and E. J. Kansa, A Least-Squares Preconditioner For Radial Basis Functions Collocation Methods, *Advances In Computational Mathematics*, To Appear.

- L. Ling, and E. J. Kansa, Preconditioning for Radial Basis Functions with Domain Decomposition Methods, *Mathematical and Computer Modelling*, In Review.

Material in chapter 2 is joint work with Manfred Trummer; that in chapter 3 is joint work with Edward Kansa. I express my thanks to them.

Dedication

To my dearest wife-to-be Juliana,
my parents, and grandparents
&
In the memory of my three loved dogs:
Dodo, Kimble, and Fancy.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Background

1.1 RBF Interpolation

Over the last 30 years, many researchers have shown a great deal of interest in radial basis functions (RBFs).

Radial basis functions have been used for interpolation problems as well as for numerically solving differential equations.

Definition 1.1.1 *If* $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ *and* $\phi(x) = \phi(y)$ *whenever* $\|x\| = \|y\|$, *then* ϕ *is called a* radial basis function.

There are an infinite number of possible univariate RBFs. Denoting r by the Euclidean distance between any pair of points in the domain Ω, some of the more commonly employed RBFs are:

$\phi(r) = r^2 \log(r)$	thin-plate spline	$SCPD2$
$\phi(r) = -r^\beta,\ j \in (0,2),$	power spline	$SCPD1$
$\phi(r) = \sqrt{r^2 + c^2},$	Hardy's multiquadric	$SCPD1$
$\phi(r) = 1/\sqrt{r^2 + c^2},$	inverse multiquadric	$SCPD0$
$\phi(r) = e^{-cr},$	exponential spline	$SCPD0$
$\phi(r) = e^{-cr^2},$	Gaussian spline	$SCPD0$
$\phi(r) = e^{-cr} K_\nu(cr),$	Matern spline	$SCPD0$
$\phi(r) = (1-r)_+^m p(r)$	CS-RBF spline	$SCPD0$

Here K_ν is the modified Bessel function of order ν , and $p(r)$ is a polynomial of the Wendland [103] compactly supported (CS-RBF) spline. The error estimate for CS-RBF can be found in [105].

The following definition deals with the positive definiteness of $\phi(\|x_i - x_j\|)$.

Definition 1.1.2 *A function* $\phi \in \mathbb{C}^0 : \mathbb{R}^+ \to \mathbb{R}$ *is* strictly conditionally positive definite of order m *(SCPDm) on* $\mathbb{R}^d$ *if for every set of distinct data points* $\{x_1, \ldots, x_N\} \subset \mathbb{R}^d$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j \phi(\|x_i - x_j\|) > 0$$

for all non-zero $\{\lambda_1, \ldots, \lambda_N\}$ *satisfying,*

$$\sum_{i=1}^{N} \lambda_i p(x_i) = 0,$$

for all polynomials p degree less than m.

The idea of RBFs is to use linear combinations of translates of one function $\phi(r)$ of one real variable, centered at "data centres" $x_k \in \mathbb{R}^d$, to approximate an unknown function:

$$s(x) = \sum_{k=1}^{n} \lambda_k \phi(\|x - x_k\|) + p(x), \tag{1.1}$$

where $p \in \Pi^d_{m-1}$ is an appended polynomial of degree $m-1$ or less in $\mathbb{R}^d$. The data centres x_k can be chosen arbitrarily in the domain of interest, hence creating a truly meshfree method; a wireframe parametric surface or connections between data centers is not required.

The interpolation problem is to find λ_k and $p(\cdot)$ such that the interpolant $s(x)$ passes through all data, $s(x_i) = f_i$. In matrix form,

$$\begin{bmatrix} A_\phi & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \gamma \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{1.2}$$

where $(A_\phi)_{ij} = \phi(x_i - x_j)$, $Pij = p_j(x_i), j = 1, \ldots, \dim(\Pi^d_{m-1})$, $\{p_j\}$ is a basis for Π^d_{m-1}, and γ is the expansion coefficients of $p(x)$ with respect to the basis for

Π^d_{m-1}. When the degree of the appended polynomial is chosen to be the order of strictly conditionally positiveness definite, then the interpolation problem (1.2) is always solvable for every set of distinct set of points unisolvent for Π^d_{m-1}.

Definition 1.1.3 *Let $X = \{x_1, ..., x_N\}$ be a subset of $\mathbb{R}^d$ and let $p \in \Pi^d_{m-1}$ be any polynomial of degree $m-1$ or less. The set X is said to be* unisolvent *if $p(x_j) = 0$ for all j implies that $p \equiv 0$.*

A set is unisolvent for Π^d_0 if all data points are distinct; a set is unisolvent for Π^d_1 if not all elements sit on a line, and so on. For compact notation, let Π^d_{-1} denotes the null space. Now we present a theorem of RBFs solvability.

Theorem 1.1.1 *If the radial basis function ϕ is SCPDm, then the interpolation matrix in (1.2) is non-singular, and the interpolation problem (1.1) has a unique solution for every set of distinct data points unisolvent for Π^d_{m-1}.*

Therefore, a SCPD0 or SCPD1 RBF requires a distinct set of data points. The matrix A_ϕ resulting from a SCPD0 RBF is symmetric positive definite. A SCPD1 RBF requires any set of data points that is not co-linear in order to *guarantee* an invertible matrix system.

In [42], Franke's numerical experiments compared 29 interpolation methods with analytic two-dimensional test functions. One of the most powerful method is the RBF method based on multiquadric (MQ) basis function

$$\phi(r) = \sqrt{r^2 + c^2}, \tag{1.3}$$

suggested by R. L. Hardy [52]. Madych and Nelson [78, 79] showed that interpolation with the MQ basis is exponentially convergent based on reproducing kernel Hilbert spaces. Wu and Schaback [108] use a different technique to prove the same results. Their technique is general enough to handle the case of interpolation with the power spline and the thin plate spline. Buhmann also showed the spectral (i.e., exponentially) convergence property of the MQ basis in [17, 18]. Since the Hilbert space is small when the radial basis function is smooth, the function being interpolated has to

be extremely smooth for the error estimates to apply. Yoon [109] showed that the MQ basis function method converges exponentially in a Sobolov space. Similar results for the thin plate spline (TPS) can be found in [104]. This was verified numerically by Fedoseyev et al. [35]. MQ interpolation becomes increasingly accurate as the shape parameter c in (1.3) increases [79]; the convergence rate goes as $\nu^{\frac{c}{h}}$, where $\nu < 1$ and h is the average distance between pairs of data centres. Carlson and Foley [21] show that the location of the data points has less effect on accuracy of the optimal solution, but the choice of shape parameter is playing an important role. The price for this increased accuracy is usually ill-conditioning of the associated linear systems which need to be solved: the "uncertainty relation" [91, 92, 93] of Schaback.

To specify the MQ basis, one must specify the centres x_j and the MQ shape parameter c_j. Most authors use the collocation points as centres for the MQ. Fornberg et al. [39] have worked on different data centers and collocation points placements near boundary. Too large or too small shape parameters c_j make the MQ basis too flat or too peaked, respectively, and should both be avoided. Hardy [52] suggested a constant shape parameter $c = 0.815 \times \text{mean}(d_j)$, where d_j is the distance from the j^{th} point to its nearest neighbour. Moody and Darken [86] suggested simple varying shape parameters $c_j = \theta\, d_j$, where $\theta > 0$ is a constant factor. Various other nonconstant shape parameters are employed by Kansa et al. [69, 87] and Hon et al. [61, 62]. Wang and Liu [101] studied the effect of shape parameters on the numerical accuracy of MQ.

We list some recent work on handling the RBF's ill conditioning problem. Beatson, Cherrie, and Mouat [7] recast the RBF basis in terms of a set of basis functions into a better conditioned one based on the far field expansions of RBFs. This results in approximate cardinal functions and lowers the computational cost of solving the RBF interpolation problem to $\mathcal{O}(N \log N)$ operations. Interested readers can find more studies of cardinal functions of MQ by Buhmann and Micchelli [19], and by Baxter [5]. Another important work of Beatson and co-workers [7, 8, 11, 13] is the fast matrix-vector product algorithm. The set up cost is only $\mathcal{O}(N \log N)$. The evaluation of the MQ expansion at any $\mathbf{x} \in \mathbb{R}^d$ requires $\mathcal{O}(\log N)$ operations, that is much faster than the $\mathcal{O}(N)$ operations for direct evaluation. This greatly improves the efficiency of all

iterative methods designed for MQ interpolation problems. Mouat [85] showed the condition number for the preconditioned MQ interpolation matrix with 3200 scattered centers in $\mathbb{R}^3$ is of order $\mathcal{O}(10^2)$, whereas that of the unpreconditioned matrix is of order $\mathcal{O}(10^8)$.

Fornberg and co-workers [40, 72] have demonstrated extra-ordinary convergence rates with the C^∞ RBFs such as multiquadrics and Gaussians in the limit as $c \to \infty$. They show that such convergence is not possible with piecewise smooth RBFs such as the thin plate splines and the r^{2k+1} splines. Near the critical value of the shape parameter (at which rounding error starts to affect accuracy badly), they permit the shape parameter to become complex, and find that branch points and removable poles cause singularities. They removed such singularities by Cauchy-Padé contour integration. In [72], Larsson and Fornberg solved the Poisson equation with remarkable accuracy. Their MSR errors were on the order of 10^{-12} with relatively few data centers.

Baxter [6] employed the well developed Toeplitz theory to get an iterative solver for gridded data. On such data, the resulting interpolation matrix would enjoy (block) Toeplitz structure. The approximate inverse of the (MQ) interpolation matrix is constructed by the inverse of its m-by-m principle matrix. In the same article, a problem with $N = 32768$ and $m = 9$ is solved with the conjugate gradient method to obtain impressive results.

Besides direct interpolation, the applications of RBF interpolation include finding the global minimum of a continuous nonconvex function on a compact subset [51], accelerating a simulated annealing method [1], estimating the global minimizer with a SQP algorithm [65, 66], medical imaging [22], surface reconstruction [25, 26, 27], and more.

1.2 RBF Collocation methods

The history of using RBF in solving partial differential equations (PDEs) is relatively short. In 1990, Kansa [68, 69] pioneered the use of RBF for the numerical solution of the Navier-Stokes equations of fluid flow. Applications of RBF-PDE arise commonly in various ordinary and partial differential equations. To name a few, they include

initial value problems [61], the nonlinear Burgers' equation [62], operator splitting-RBF [4], boundary layer problems [56, 77], surface wind field approximation [54], biphasic and triphasic models of mixtures [59, 60], the shallow water equation [58, 106, 110], and financial mathematics [63, 80].

To introduce RBF collocation methods, we consider a PDE in the form of

$$\begin{aligned} \mathcal{L}u &= f(x) \quad \text{in } \Omega \subset \mathbb{R}^d, \\ \mathcal{B}u &= g(x) \quad \text{on } \partial\Omega, \end{aligned} \tag{1.4}$$

where d is the dimension, $\partial\Omega$ denotes the boundary of the domain Ω, $\mathcal{L}$ is the differential operator that operates on the interior, and $\mathcal{B}$ is an operator that specifies the boundary conditions of Dirichlet, Neumann or mixed type . Both f and g are given functions mapping $\mathbb{R}^d \to \mathbb{R}$.

Using Kansa's asymmetric multiquadric collocation method, similar to (1.1), the unknown PDE solution u is approximated by RBFs in the form

$$u \approx U(x) = \sum_{k=1}^{N} \lambda_k \, \phi_k(x) + p(x), \tag{1.5}$$

where $\phi_k(x) = \phi(\|x - x_k\|)$, and ϕ can be any radial basis function, $p(\cdot) \in \Pi_{m-1}^d$ is a polynomial of degree $m-1$ or less, and $\|\cdot\|$ indicates the Euclidean norm. Let $\{ (x_j, u_j) \}_{j=1}^N$ be the N collocation points in $\Omega \cup \partial\Omega$. We assume the collocation points are arranged in such a way that the first N_I points and the last N_B points are in Ω and on $\partial\Omega$, respectively. To solve for the $N+m$ unknown coefficients $\lambda = [\lambda_1, \ldots, \lambda_N]^T$ and $p(\cdot)$ in (1.5), $N+m$ linearly independent equations are needed. Ensuring that $U(x)$ satisfies (1.4) at the collocation points results in a good approximation of the solution u. The first N equations are given by

$$\begin{aligned} \sum_{k=1}^{N} \lambda_k \mathcal{L}\phi_k(x_i) + \mathcal{L}p(x) &= f(x_i) \quad \text{for } i = 1, \ldots, N_I, \\ \sum_{k=1}^{N} \lambda_k \mathcal{B}\phi_k(x_i) + \mathcal{B}p(x) &= g(x_i) \quad \text{for } i = N_I+1, \ldots, N_I+N_B. \end{aligned} \tag{1.6}$$

The last m equations could be obtained by imposing some extra condition on $p(\cdot)$.

Rewriting (1.6) in matrix form, we have

$$\begin{bmatrix} A\mathcal{L} & P_{\mathcal{L}} \\ A\mathcal{B} & P_{\mathcal{B}} \\ P^T & 0 \end{bmatrix} \lambda = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix}, \tag{1.7}$$

and

$$\begin{aligned} (A_{\mathcal{L}})_{ik} &= \mathcal{L}\phi(x_i - x_k), \quad \text{for } i = 1, \ldots, N_I, \\ (A_{\mathcal{B}})_{ik} &= \mathcal{B}\phi(x_i - x_k), \quad \text{for } i = N_I + 1, \ldots, N_I + N_B. \end{aligned} \tag{1.8}$$

This method is often named the asymmetric collocation method. The matrix given by (1.7) and (1.8) is generally non-symmetric and full; this system of equations is known to be ill-conditioned when N or c becomes large. The symmetric collocation approach can be found in [34]. The idea is to modify the RBF basis in (1.5) using the differential operator $\mathcal{L}$ and $\mathcal{B}$. Since higher order derivatives are employed in the numerical formulation, generally speaking, the symmetric method is less accurate than Kansa's method, and is therefore less popular.

There is ample evidence that RBFs, especially MQ-RBFs, offer many computational advantages over traditional methods. Some of the advantages of RBFs are their truly meshfree nature and their very high order rates of convergence, see Golberg, Chen, and Karrur [48], Cheng et al. [24], Fedoseyev [35]. That is, for small to moderate sized problems, RBFs often outperform traditional methods. In [46], Golberg and Chen observe competitive results of RBFs with 60 points versus the linear FEM with 71000 points. The main concern is whether RBFs can be computationally efficient with large scale, and geometrically complex problems. Thus, there is a reluctance to embrace this method because of the perceived problems associated with ill-conditioning and the operation counts associated with very wide or full matrices.

The problem of ill-conditioned large systems of equations is not unique to RBFs. Compactly supported methods such as finite differences, finite elements, and finite volumes also suffer from ill-conditioning when the dimension of the system becomes very large. However, their ill conditioning grows slower than for global RBF. The problem of ill conditioning forces one to use direct methods instead of iterative methods to solve the resulting RBF-PDE matrix systems. With the Gaussian elimination

methods, the number of operations required is $\mathcal{O}(N^3)$. Kansa and Hon [70] showed that a truncated MQ expansion yielding sparse matrices was capable of yielding very good accuracy in the solution of elliptic PDE problems, although direct methods were still used over each subdomain.

If a domain is decomposed into many semi-independent subdomains containing $N_k \ll N$ points, the condition number as well as the operation count in each subdomain is considerably reduced. There are basically four ways to overcome the ill-conditioning problems, see Smith, Bjørstad, and Gropp [96]: (i) preconditioning, (ii) domain decomposition methods (DDM) using overlapping or non-overlapping schemes, (iii) combinations of domain decomposition and preconditioning, and (iv) removal of singularities by Padé-contour integration developed by Fornberg and co-workers [38, 40, 41]. Hardy [53] showed that very accurate results could be obtained without resorting to global expansions; Hardy and co-workers were the first to use the domain decomposition method (DDM) in a variety of image reconstruction problems.

In [55], Hon and Schaback provide counterexamples to show that a general proof of nonsingularity for unsymmetric collocation is impossible. However, their numerical evidences show that cases of singularity are rare. It takes more than 7000 random samples to find a counterexample. The existence of singular counterexamples does not reject the importance of RBF collocation methods. One can slightly move one or a few data points, or change the value of the shape parameter, if near-singularity is detected.

Franke and Schaback [43] give the first theoretical foundation for RBFs with the error estimate,

$$\|u - U\|_{\infty,\Omega} \leq \|g - U\|_{\infty,\partial\Omega} + C \cdot \sqrt{\text{vol}(\Omega)}\|f - \mathcal{L}U\|_{\infty,\Omega}.$$

Then, in [44] they show that $\|u - U\| \leq \mathcal{O}(h^{\rho - q/2})$, where ρ measures the regularity of the solution, q is even and denotes the order of the differential operator $\mathcal{L}$.

1.3 About this thesis

This dissertation deals with various aspects of using RBFs in scientific computing problems.

We begin with a study of the quasi-interpolation problem. Instead of solving the interpolation equations, a quasi-interpolation procedure provides an approximate interpolation formula without solving any matrix system. In chapter 2, we propose a multilevel scheme for achieving better smoothness of an existing univariate quasi-interpolation scheme for MQ-RBF, and prove its convergence. Then we extend the scheme to higher dimensions. We show that the technique of boundary padding can improve accuracy. The relationship between smoothness and derivative approximation is also studied.

In chapter 3, we focus on the numerical solutions of boundary layer problems. We concentrate on the case where these layers occur near the boundary, although our method can be applied to problems with interior layers. One technique to deal with the increased resolution requirements in these layers is the use of domain transformations. A coordinate stretching based transform allows one to move collocation points into the layer, a requirement to resolve the layer accurately. Previously such transformations have been studied in the context of finite difference and spectral collocation methods. We use radial basis functions (RBF) to solve the boundary value problem. Specifically, we present a collocation method based on multiquadric (MQ) functions with an integral formulation combined with a coordinate transformation. We find that our scheme is ultimately more accurate than a recently proposed adaptive MQ scheme.

The advantages of radial basis functions also allow our scheme to become adaptive relatively easily. We introduce a new error indicator function which accurately captures the regions of the interval with insufficient resolution. This indicator is used to adaptively add data centres and collocation points. The method allows us to resolve extremely thin layers accurately with fairly few basis functions. The adaptive scheme is very robust, and reaches high accuracy even when parameters in our coordinate stretching technique are not chosen optimally.

Our third research topic studies different preconditioning techniques for solving partial differential equations (PDEs) with radial basis functions collocation methods. In chapter 3 we present a simple preconditioning scheme that is based upon constructing least-squares approximate cardinal basis functions (ACBFs) from linear combinations of the RBF-PDE matrix elements. The ACBFs transform a badly conditioned linear system into one that is well conditioned, allowing us to solve for the expansion coefficients iteratively, so we can reconstruct the unknown solution everywhere on the domain. Our preconditioner requires $\mathcal{O}(mN^2)$ flops to set up, and $\mathcal{O}(mN)$ storage locations, where m is a user defined parameter of order 10. For the two dimensional MQ-RBF basis with the shape parameter $c \sim 1/\sqrt{N}$, the number of iterations required for convergence is of order 10 for large values of N, making this a very attractive approach computationally.

The idea of the classical alternating Schwarz algorithm was first published by Schwarz [94] in 1870. The application of DDM includes finite difference methods, finite element methods, finite volume methods, and spectral methods for linear or nonlinear problems, and other applications. Readers should refer to the proceedings of the annual domain decomposition meetings for the most recent developments. Recently, as interest in RBFs increased, researchers began to combine the DDM with RBF methods.

Hardy [53] was the first to use DDM on interpolation and approximation problems. In fact one of Hardy's most impressive discoveries was that excellent accuracy did not depend on global expansions. Kansa [69] used DDM on a hyperbolic problem; later Dubal [28] applied DDM with the MQ RBF to solve one-dimensional problems. Kansa and Hon [70] found that the fineness of the domain cuts had a big impact on the ill-conditioning problem and achieved better accuracy. Advances on coupling overlapping domain decomposition and RBF methods can be found in [107, 111].

RBF-PDE solution methods that incorporate the newly developed least-squares preconditioners permitting GMRES iteration, domain decomposition methods, a controlled degree of coefficient matrix sparsity, and fast multipole expansions under certain circumstances can outperform traditional mesh-based methods that give rise to sparse matrices. The question is whether a more complex, very high or exponentially

convergent method requiring a much coarser discretization is more efficient than a very low order sparse scheme. A thorough study to answer this question has yet to be undertaken.

We couple the ACBF preconditioning technique with a domain decomposition method (DDM), namely, the classic alternating Schwarz algorithm. The truly mesh-free RBFs collocation methods allow the Schwarz algorithm to be applied to both matching and non-matching nodes without any modification. Our method allows the dimension of the matrix to be reduced and therefore allows the ACBFs preconditioner to work more efficiently. We experiment with different implementations of the ACBF-DDM scheme and provide numerical results for $N > 10,000$ nodes. Our algorithm not only helps reducing the overall work of solving RBF-PDE systems, but we also show that the ACBF-DDM method is more efficient than solving the global RBF-PDE problem with ACBFs preconditioning alone. In particular, efficiency can be improved by using more subdomains.

Chapter 2

On Quasi Interpolations

2.1 Univariate Quasi-interpolation formula

The standard formula for the interpolation of a function $f \in C^1 : [a, b] \rightarrow \mathbb{R}$ on scattered points and data $\{(x_j, f_j\})_{j=0}^n$ where

$$a = x_0 < x_1 < \ldots < x_n = b, \quad h = \max_{0 \leq j \leq n} (x_j - x_{j-1}), \tag{2.1}$$

has the form

$$f(x_k) = \mathcal{Q}f(x_k) \tag{2.2}$$

where

$$\mathcal{Q}f(x) = \sum_{k=1}^{n} \lambda_k \, \chi(x - x_k), \tag{2.3}$$

and $\chi(\cdot)$ is an interpolation kernel, for all $0 \leq k \leq n$.

In this chapter, we focus on the quasi-interpolation method. A weaker form of (2.2), known as *quasi-interpolation*, holds only for polynomials of degree d, i.e.,

$$p_d(x_k) = \mathcal{Q}p_d(x_k), \quad \forall \; p_d \in \Pi_d, \tag{2.4}$$

for all $0 \leq k \leq n$. Beatson and Powell [14] first proposed a quasi-interpolation formula where $\chi(x)$ in (2.3) is a linear combination of the MQ defined in (1.3). Their formula requires derivative values of f at endpoints, which is not convenient for practical

purposes. Wu and Schaback [102] proposed another quasi-interpolation formula with modification at the endpoints. Given data $\{(x_j, f_j)\}_{j=0}^{n}$, Wu-Schaback's formula is

$$\mathcal{L}_{\mathcal{D}} f(x) = f_0\alpha_0(x) + f_1\alpha_1(x) + \sum_{j=2}^{n-2} f_j\alpha_j(x) + f_{n-1}\alpha_{n-1}(x) + f_n\alpha_n(x), \tag{2.5}$$

where

$$\begin{aligned}
\alpha_0(x) &= \frac{1}{2} + \frac{\phi_1(x) - (x - x_0)}{2(x_1 - x_0)}, \\
\alpha_1(x) &= \frac{\phi_2(x) - \phi_1(x)}{2(x_2 - x_1)} - \frac{\phi_1(x) - (x - x_0)}{2(x_1 - x_0)}, \\
\alpha_j(x) &= \frac{\phi_{j+1}(x) - \phi_j(x)}{2(x_{j+1} - x_j)} - \frac{\phi_j(x) - \phi_{j-1}(x)}{2(x_j - x_{j-1})}, \\
\alpha_{n-1}(x) &= \frac{(x_n - x) - \phi_{n-1}(x)}{2(x_n - x_{n-1})} - \frac{\phi_{n-1}(x) - \phi_{n-2}(x)}{2(x_{n-1} - x_{n-2})}, \\
\alpha_n(x) &= \frac{1}{2} + \frac{\phi_{n-1}(x) - (x_n - x)}{2(x_n - x_{n-1})}.
\end{aligned} \tag{2.6}$$

The interpolation kernel α_i is chosen so that the quasi-interpolation formula (2.5) can exactly reproduce the constant and linear functions.

The main advantage of this formula is that it does not require solving any linear system. Instead, the formula uses the given function values f_j at x_j as its coefficients. The drawback is that instead of $c = \mathcal{O}(h)$, one needs to use a smaller shape parameter c in order to achieve quadratic convergence resulting in less smoothness. Therefore, we propose a scheme using the same basis (2.6) which allows a shape parameter $c = \mathcal{O}(h)$. Moreover, we study the convergence rate of our proposed scheme.

For the readers' convenience, we quote some of Wu-Schaback's theorems here without proofs.

Theorem 2.1.1 [102, Theorem 2] *If the data $f(x_j)_{j=0}^{n}$ stem from a convex (concave, linear) function, then the quasi-interpolant $\mathcal{L}_{\mathcal{D}} f(x)$ as defined by (2.5) and (2.6) is a convex (concave, linear) function.*

Theorem 2.1.2 [102, Theorem 3] *The quasi-interpolation $\mathcal{L}_{\mathcal{D}} f(x)$ is monotonicity preserving.*

Theorem 2.1.3 [102, Theorem 4 & corollary] *For $f \in C^2[a,b]$ the quasi-interpolant $\mathcal{L}_{\mathcal{D}}f(x)$ defined on the points (2.1) satisfies an error estimate of type*

$$\|f - \mathcal{L}_{\mathcal{D}}f(x)\|_\infty \leq K_1 h^2 + K_2 ch + K_3 c^2 \log h$$

for $h \to 0$ with suitable positive constants K_1, K_2, and K_3 independent of h and c. The quasi-interpolant $\mathcal{L}_{\mathcal{D}}f(x)$ can have $\mathcal{O}(h^2)$ error only if at least

$$c^2|\log c| = \mathcal{O}(h^2).$$

Furthermore, Wu and Schaback concluded that no improvement towards $\mathcal{O}(h^2)$ convergence is possible just by changes of end conditions or knot placements, provided that the α_j for $2 \leq j \leq n-2$ in (2.6) are used in the interior of the domain.

2.1.1 The multilevel quasi-interpolation in 1-D

Given a quasi-interpolation problem defined by (2.4) with $d = 1$ on scattered points (2.1), we will present a quasi MQ interpolation, denoted by $\mathcal{L}_{\mathcal{R}}$ in the form of (2.5) with the same basis as in (2.6) but using coefficients other than f_j. For any function $f \in C^1 : [a,b] \to \mathbb{R}$, and data $\{(x_j, f_j\})_{j=0}^n$, if we pick $m \approx \frac{n}{2}$ and define an index

$$0 = k(0) < k(1) < \ldots < k(m) = n, \tag{2.7}$$

we have another quasi-interpolation problem based on a smaller set of data, namely

$$\{(x_{k(j)},\ f_{k(j)})\}_{j=0}^m. \tag{2.8}$$

We denote by $\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}$ the quasi-interpolation $\mathcal{L}_{\mathcal{D}}$ defined by (2.5) and (2.6) applied to data (2.8) on data points $x_{k(j)}$. Let the residual or error function after this coarse quasi-interpolation be

$$\mathcal{E}(x) = f(x) - \mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}f(x). \tag{2.9}$$

Using (2.9), we can define another quasi-interpolation problem on the original data points x_j. The data being interpolated are

$$\{(x_j,\ \mathcal{E}_j)\}_{j=0}^n. \tag{2.10}$$

Since the function $\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}f(x)$ is defined on $\mathbb{R}$ and the function values $f_j = f(x_j)$ are known, the values $\mathcal{E}_j = \mathcal{E}(x_j)$ can be computed for all $0 \leq j \leq n$.

Similarly, let $\mathcal{L}_{\mathcal{D}\{x_j\}}\mathcal{E}(x)$ be the quasi-interpolation applied to data (2.10) on the original points x_j. We denote our new (level-1) quasi-MQ interpolation operator by

$$\mathcal{L}_{\mathcal{R}}f(x) = \mathcal{L}_{\mathcal{D}\{x_j\}}\mathcal{E}(x) + \mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}f(x). \tag{2.11}$$

Theorem 2.1.4 *The quasi-interpolation $\mathcal{L}_{\mathcal{R}}$ preserve monotonicity.*

Proof: Applying theorem 2.1.2 proves the assertion. ∎

By rewriting (2.5) and (2.6), we get

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}f(x) &= \frac{1}{2}\left\{\left[\frac{f_1 - f_0}{x_1 - x_0}\right](x - x_0) + \sum_{j=1}^{n-1}\left[\frac{f_{j+1} - f_j}{x_{j+1} - x_j} - \frac{f_j - f_{j-1}}{x_j - x_{j-1}}\right]\phi_j(x)\right. \\
&\qquad \left. + \left[\frac{f_n - f_{n-1}}{x_n - x_{n-1}}\right](x - x_n) + [f_0 + f_n]\cdot 1\right\}. \\
&= \sum_{j=1}^{n-1}\frac{1}{2}\left[\frac{f_{j+1} - f_j}{x_{j+1} - x_j} - \frac{f_j - f_{j-1}}{x_j - x_{j-1}}\right]\phi_j(x) \\
&\qquad + \frac{1}{2}\left[f_0 + f_n - \frac{f_1 - f_0}{x_1 - x_0}\cdot x_0 - \frac{f_n - f_{n-1}}{x_n - x_{n-1}}\cdot x_n\right]\cdot 1 \\
&\qquad + \frac{1}{2}\left[\frac{f_1 - f_0}{x_1 - x_0} + \frac{f_n - f_{n-1}}{x_n - x_{n-1}}\right]x.
\end{aligned} \tag{2.12}$$

Note that it is cheaper to implement (2.12) instead of (2.5) to evaluate the interpolant.

An important work of Beatson and coworkers [8, 9, 11, 13] is the fast matrix-vector product algorithm. If c is constant and $0 < c \leq h$ for all ϕ, the cost of evaluating the MQ function at a large number of different evaluation points y through

$$s(y) = \sum_{j=0}^{n} d_j\ \phi(y - x_j),$$

is $\mathcal{O}(\log n)$ plus set-up cost using the fast multipole method for large n. This is much faster than the $\mathcal{O}(n)$ flop for direct evaluation of $s(y)$.

To have $\mathcal{L}_{\mathcal{R}}$ as efficient as $\mathcal{L}_{\mathcal{D}}$, we must use the same shape parameter c in both $\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}f(x)$ and $\mathcal{L}_{\mathcal{D}\{x_j\}}\mathcal{E}(x)$. Doing so will avoid introduction of a new basis. As

Algorithm 1 Pseudo-code for quasi MQ interpolation $\mathcal{L}_{\mathcal{R}}$.

INPUT: $x \in \mathbb{R}^{n+1}$, $f \in \mathbb{R}^{n+1}$

OUTPUT: $\lambda \in \mathbb{R}^{n+1}$

Find separating distance $h = \max_{1 \le i \le n} (x_i - x_{i-1})$

Pick a shape parameter $c = \mathcal{O}(h)$

Find coarse data $X = (\downarrow 2)(x)$, $F = (\downarrow 2)(f)$

Find coefficients $\lambda = \text{Coef}(F)$

Find error $\mathcal{E} = f - \mathcal{L}_{\mathcal{D}\{X\}} f(x)$

Update coefficients $\lambda \longleftarrow \text{Coef}(\mathcal{E}) + (\uparrow 2)(\lambda)$,

indicated earlier, this results in an update of the undetermined coefficients for the $n+1$ basis functions

$$[\phi_1, \ldots, \phi_{n-1}, 1, x], \tag{2.13}$$

in (2.12). One may repeat this procedure to construct higher level schemes. We denote the level-2 scheme, with two updates in the undetermined coefficients, by $\mathcal{L}_{\mathcal{R}^2}$. This quasi MQ interpolation will be studied numerically in Section 2.1.3.

The pseudo-code of finding the undetermined coefficients of the quasi MQ interpolation $\mathcal{L}_{\mathcal{R}}$ is given in algorithm 1 where $(\downarrow 2)$ is any function that downsamples $\{x_j\}_{j=0}^{n}$ to $\{x_{k(j)}\}_{j=0}^{m}$ and $(\uparrow 2)$ upsamples the coefficients of $\mathcal{L}_{\mathcal{R}\{x_{k(j)}\}} f(x)$, $\lambda \in \mathbb{R}^{m+1}$, to the corresponding position with respect to (2.13). Coefficients of unused basis functions are zero. The function "*Coef*" maps the input data to the coefficients defined by (2.12). Only minor modifications to the code of Wu-Schaback's formula are required. All steps in algorithm 1 cost $\mathcal{O}(n)$ flops except the step "*find error*" which costs $n \cdot \mathcal{O}(\log n)$ flops to evaluate all the coefficients of the quasi-interpolation $\mathcal{L}_{\mathcal{R}\{x_{k(j)}\}} f$

using the fast multipole methods. Thus the setup cost for $\mathcal{L}_{\mathcal{R}}$ is $\mathcal{O}(n \log n)$. Once the coefficients are determined, one can always differentiate or integrate the functions (2.13) in order to approximate the derivative or the integral of $f(x)$.

2.1.2 The accuracy of the quasi-interpolation $\mathcal{L}_{\mathcal{R}}$

Suppose a quasi-interpolation satisfies (2.4) for some d. Then the error bound for the approximation of a smooth L_2-function $g(x)$ is given by:

$$\|g - \mathcal{Q}g\| \leq K(\chi) \cdot h^{d+1} \cdot \|g^{(d+1)}\|, \tag{2.14}$$

where $g^{(d+1)}$ denotes the $(d+1)^{st}$ derivative of g and $K(\chi)$ is a constant depending only on the interpolation kernel χ used in (2.3). This concept is introduced by Strang and Fix [36]; also see [99] for more recent developments.

In this section, we show that for $f \in C^2[a, b]$ the quasi-interpolation $\mathcal{L}_{\mathcal{R}}$ defined by (2.11) enjoys a convergence rate faster than the existing method defined by (2.5) and (2.6), and it allows one to use $c = \mathcal{O}(h)$. When both $|f''(a)|$ and $|f''(b)| \ll \|f''\|_\infty$, the convergence rate of our method will increase, which is numerically verified in section 2.1.3. We extend the proof of results in [102].

From (2.12), we can easily see that $\mathcal{L}_{\mathcal{D}}$ reproduces constant functions and linear functions exactly. Besides (2.14), theorem 2.1.3 suggests for a smooth L_2-function $f(x)$, we have

$$\|f(x) - \mathcal{L}_{\mathcal{D}} f(x)\|_\infty \leq \|f''(x)\|_\infty (\tilde{K}_1 h^2 + \tilde{K}_2 ch + \tilde{K}_3 c^2 \log h). \tag{2.15}$$

By (2.9), (2.11), and (2.15), we have

$$\begin{aligned} \|f(x) - \mathcal{L}_{\mathcal{R}} f(x)\|_\infty &= \|f(x) - \mathcal{L}_{\mathcal{D}\{x_j\}} \mathcal{E}(x) - \mathcal{L}_{\mathcal{D}\{x_{k(j)}\}} f(x)\| \\ &= \|\mathcal{E}(x) - \mathcal{L}_{\mathcal{D}\{x_j\}} \mathcal{E}(x)\| \\ &\leq K(c, h_2) \cdot \|\mathcal{E}''(x)\|_\infty, \end{aligned}$$

where

$$K(c, h) = \tilde{K}_1 h^2 + \tilde{K}_2 ch + \tilde{K}_3 c^2 \log h,$$

c is the shape parameter of $\phi_j(x)$ defined by (1.3), and h_2 denotes the maximum separating distance of the data points $x_{k(j)}$. Since we have $h_2 \approx 2h$ as we assume roughly half of the data points are used in the coarse quasi-interpolation, the terms $\mathcal{O}(h_2)$ are also $\mathcal{O}(h)$. We use the notation $\mathcal{O}(h)$ instead of $\mathcal{O}(h_2)$ throughout the section. Also, note that there exists a point $x^* \in [a,b]$ such that

$$\|\mathcal{E}''(x)\|_\infty = \left| f''(x^*) - \left(\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}} f\right)''(x^*) \right|. \tag{2.16}$$

Since $\phi_j \in C^\infty$, the function $\mathcal{E} \in C^2[a,b]$. Replacing $\mathcal{L}_\mathcal{D}$ by $\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}}$ in (2.12) and differentiating with respect to x twice we get

$$\begin{aligned} \left(\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}} f\right)''(x) &= \frac{1}{2}\sum_{j=1}^{m-1} \left[\frac{f_{k(j+1)} - f_{k(j)}}{x_{k(j+1)} - x_{k(j)}} - \frac{f_{k(j)} - f_{k(j-1)}}{x_{k(j)} - x_{k(j-1)}} \right] \phi''_{k(j)}(x) \\ &= \sum_{j=1}^{m-1} f''_{k(j)} \frac{\phi''_{k(j)}(x)}{2} \cdot h_2 + \mathcal{O}(h^2), \end{aligned} \tag{2.17}$$

where the index $k(j)$ is defined by (2.7). By the symmetry of the MQ function $\phi''_x(\eta) = \phi''_\eta(x)$. Applying the Trapezoidal integration formula to the summation in (2.17), we obtain

$$\begin{aligned} \sum_{j=1}^{m-1} f''_{k(j)} \frac{\phi''_{k(j)}(x)}{2} \cdot h_2 &= -\frac{h_2}{4}\left[f''_{k(1)} \phi''_{x_{k(1)}}(x) + f''_{k(m-1)} \phi''_{x_{k(m-1)}}(x) \right] \\ &\quad + \int_{x_{k(1)}}^{x_{k(m-1)}} f''(\eta) \frac{\phi''_x(\eta)}{2} d\eta + \mathcal{O}(h^2). \end{aligned} \tag{2.18}$$

Combining (2.17) and (2.18), we have

$$\left(\mathcal{L}_{\mathcal{D}\{x_{k(j)}\}} f\right)''(x^*) = \int_{x_{k(1)}}^{x_{k(m-1)}} f''(\eta) \frac{\phi''_{x^*}(\eta)}{2} d\eta + B(f)O(h) + O(h^2), \tag{2.19}$$

where

$$B(f) = f''_a \phi''_a(x) + f''_b \phi''_b(x).$$

We define $\sigma(x) = \text{sign}(f''(x))$, and we assume f is neither a constant nor a linear polynomial (i.e., f'' is not identically equal to zero).

Case 2.1.1 *General case*

In the first case, we look at the integral in (2.19). Assume $\sigma(x^*) = +1$, *and* $x^* \in (a, b)$. *(There is no loss in generality. If* $x^* = a$ *or* $x^* = b$, *replace* x^* *in (2.16) by* $z^* \pm \sqrt{c}$, *respectively. Then,* $\mathcal{E}''(x^*) = \mathcal{E}''(z^*) + \mathcal{O}(\sqrt{c})$.*) It is easy to verify that* $\phi''(x)$ *is a strictly positive decaying function, and for any shape parameter* c,

$$\int_{-\infty}^{\infty} \frac{\phi''_{x^*}(\eta)}{2} d\eta = 1, \quad \forall\ x \in \mathbb{R}.$$

For any x^*, *the function* $\frac{1}{2}\phi''_{x^*}(\eta)$ *becomes a nonnegative density function as* $c \to 0$. *By splitting the interval into* $|\eta - x^*| \le \sqrt{c}$ *and* $|\eta - x^*| > \sqrt{c}$, *One can show that*

$$\begin{aligned} \int_{x_{k(1)}}^{x_{k(m-1)}} f''(\eta) \frac{\phi''_{x^*}(\eta)}{2} d\eta &= \left(f''(x^*) + \mathcal{O}(\sqrt{c})\right) \frac{1}{\sqrt{1+c}} + \mathcal{O}(c) \\ &= f''(x^*) + \mathcal{O}(\sqrt{c}). \end{aligned} \tag{2.20}$$

Therefore, we get

$$\|f(x) - \mathcal{L}_{\mathcal{R}} f(x)\|_\infty \le K(c, h_2) \cdot \left[\mathcal{O}(\sqrt{c}) + B(f)\mathcal{O}(h) + \mathcal{O}(h^2)\right].$$

We conclude our results with the following theorem.

Theorem 2.1.5 *For* $f \in C^2[a, b]$ *the quasi-interpolant* $\mathcal{L}_{\mathcal{R}} f(x)$ *converges to* $f(x)$ *at a speed of* $O(h^{2.5} \log h)$ *under the* ℓ_∞*-norm provided* $c = \mathcal{O}(h)$.

Corollary 2.1.1 *If the data* $f(x_j)_{j=0}^n$ *stem from a strictly convex (strictly concave, linear) function, then the quasi-interpolant* $\mathcal{L}_{\mathcal{R}} f(x)$ *is a strictly convex (strictly concave, linear) function for* h *small enough.*

Proof: *The linear case is trivial as* $\mathcal{L}_{\mathcal{D}}$ *interpolates linear function exactly. Suppose* f *is strictly convex and is not linear. The results above not only hold for* x^*, *but also for all* $x \in [a, b]$. *Thus, by (2.20)*

$$\mathcal{E}''(x) \ge f''(x)\left(1 - \frac{1}{\sqrt{1+c}}\right) + \frac{h_2}{4} B(f) + \mathcal{O}(h^2) > 0,$$

which is also strictly positive provided h *is small enough. Applying theorem 2.1.1 proves the corollary.* ∎

Case 2.1.2 *Special case*

This case includes functions whose second derivative at both boundaries are significantly smaller than their ℓ_∞-norm. Assume $x^* \in (a+\Delta,\, b-\Delta)$. Again we assume $\sigma(x^*) = +1$. Then the term $B(f)$ can be rewritten as

$$[f''(a) + f''(b)] \cdot \mathcal{O}\left(\frac{c^2}{\Delta^3}\right) \cdot \mathcal{O}(h),$$

and becomes one of the higher order terms. Furthermore, the integral in (2.19) can now be estimated by splitting the interval into $|\eta - x^*| \le L$ and $|\eta - x^*| > L$ where $\sqrt{c} \le L < \Delta$; i.e., we have more room to obtain a tighter bound than (2.20).

This claim seems to be counter-intuitive. The function f may behave badly inside the domain, $\mathcal{L}_\mathcal{R} f$ converges to f faster than theorem 2.1.5 predicts as long as $f''(a)$ and $f''(b)$ are relatively small compared with $\|f''\|_\infty$. Note that a higher convergence rate is not equivalent to better accuracy. When f is smooth, the error will be small in general. This will be studied numerically in Section 2.1.3.

On the other hand, case 2.1.2 reflects the spectral convergence property of MQ. The condition of f'' on boundaries implies that the linear basis can approximate f relatively closely at boundaries. The convergence rate is then determined by the interior MQ basis.

2.1.3 Numerical examples

In this section, we compare the rate of convergence of our formula $\mathcal{L}_\mathcal{R}$ with the one of the Wu-Schaback's formula $\mathcal{L}_\mathcal{D}$, and the level-2 scheme $\mathcal{L}_{\mathcal{R}^2}$. We present results for the functions listed in figure 2.1. Test functions are normalized so that their ranges are of $\mathcal{O}(1)$. We are interested in how the MQ shape parameter affects the convergence rate and the accuracy. Over

$$\frac{c}{h} = 0.1r, \quad 1 \le r \le 40,$$

both convergence rate, and errors for 2^{-11} are reported graphically. For figure 2 to figure 7, we use the dot ($\cdot$), the plus sign ($+$), and the cross ($\times$) to denote the $\mathcal{L}_\mathcal{D}$, $\mathcal{L}_\mathcal{R}$, and $\mathcal{L}_{\mathcal{R}^2}$ quasi-interpolation, respectively.

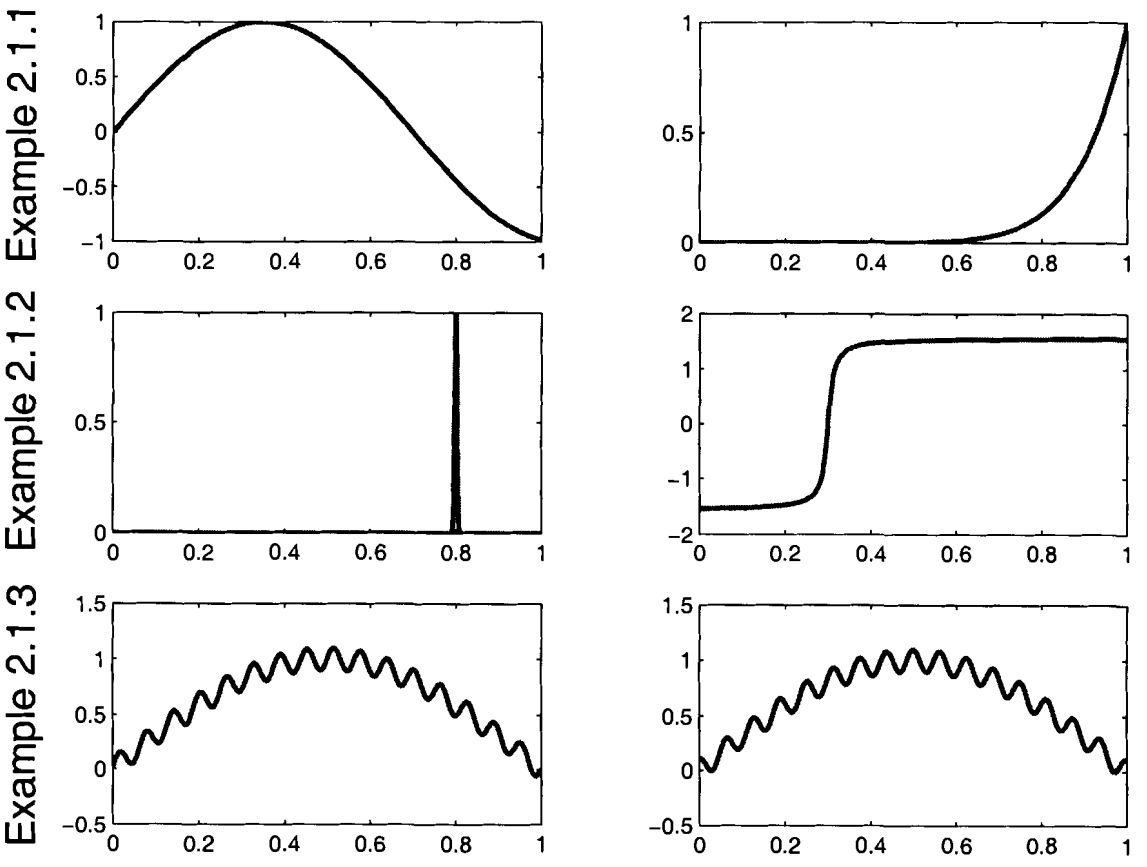

Figure 2.1: Test functions for the new quasi MQ interpolation scheme.

We compute the quasi-interpolants on $[0, 1]$ with equal spacing $h = 2^{-10}$ and $h = 2^{-11}$. For both h's, all three quasi-interpolants are evaluated at 4096 points to compute errors. The ℓ_∞-norm error is taken to be the maximum absolute error at the points of evaluation. The convergence rate in theorem 2.1.5 for these particular h's is 2.36.

In the last example, we compare the traditional MQ-RBF interpolation and our quasi-interpolation scheme on small number of points.

Example 2.1.1 (General case) *The first example demonstrates the result in theorem 2.1.5. We report the results for*

$$f_1(x) = \sin(4.5x),$$

and

$$f_2(x) = x^9.$$

The first function has maximum second derivative inside the domain, $|f_1''(b)| \approx 0.978\|f_1''\|$. *For the second function we have* $\|f_2''\|_\infty = f_2''(b)$. *Convergence rates and errors of the interpolants are in figure 2.2 and figure 2.3.*

The convergence rate of $\mathcal{L}_\mathcal{D}$ *is consistently below* 2 *except at the last tested ratio,* $c/h = 0.1$. *This verifies theorem 2.1.3 that* $\mathcal{L}_\mathcal{D}$ *converges quadratically only if* $c^2|\log c| = \mathcal{O}(h^2)$. *Furthermore, its errors decrease monotonically. Using small* c *is a trade-off of smoothness to achieve accuracy. This behavior reflects the fact that* $\mathcal{L}_\mathcal{D}$ *is constructed based on the piecewise linear interpolation by* $|x|$. *The errors of using* $\mathcal{L}_\mathcal{D}$ *are a magnitude larger than the other two methods for all tests. For all tested functions,* $\mathcal{L}_\mathcal{D}$ *behaves similarly.*

The results of $\mathcal{L}_\mathcal{R}$ *are promising. For both* f_1 *and* f_2, *the convergence rates reach* $\mathcal{O}(h^{2.08})$ *when* $c = 0.8h$. *Despite the slower than predicted converge rates, they are approximately quadratic for all tested* c *between* $0.1h$ *to* $4h$.

Lastly, the level-2 scheme, $\mathcal{L}_{\mathcal{R}^2}$, *behaves similarly to* $\mathcal{L}_\mathcal{R}$. *This suggests that the lower bound of* $\mathcal{L}_{\mathcal{R}^2}$*'s convergence rate is the same as* $\mathcal{L}_\mathcal{R}$; *i.e., theorem 2.1.5 applies to* $\mathcal{L}_{\mathcal{R}^2}$ *too. Note that the errors of* $\mathcal{L}_{\mathcal{R}^2}$ *are smaller than the errors of* $\mathcal{L}_\mathcal{R}$ *for* $c \geq 2h$. *This behavior can be observed for all functions* $f \in C^1$. *In this case, one may want to spend* $\mathcal{O}(n \log n)$ *extra flops for an extra update to get an even higher degree of smoothness.*

Example 2.1.2 (Special case) *In this example, we use*

$$f_3(x) = \exp\left(-2^{16}(x - 0.8)^2\right),$$

and

$$f_4(x) = \arctan\left(100(x - 0.3)\right),$$

to mimic impulses and shocks, respectively. From what we have learned from case 2.1.2 in section 2.1.2, although these functions have very rapid local behavior, their second derivatives are almost locally supported and are nearly zero at both boundaries. We numerically verify our claim.

The results are shown in figure 2.4 and figure 2.5. The maximum rates of convergence under the ℓ_∞-norm are 2.80 *and* 3.02 *occurring at* $c = 0.9h$ *and* $c = 1.2h$ *for* f_3 *and* f_4*, respectively. Both tests clearly show a much faster convergence rate than for the functions in example 2.1.1 when* $c > h$*. The minimum error occurs around* $c = h$ *for both tests: they are* $c = .6h$ *and* $c = 0.9h$ *for* f_3 *and* f_4*, respectively.*

As in example 2.1.1, we see that the optimal c for $\mathcal{L}_{\mathcal{R}^2}$ *is between* $2.3h$ *and* $3h$ *in terms of the rate of convergence under the* ℓ_∞*-norm. From the result for* f_4*, we can see that* $\mathcal{L}_{\mathcal{R}^2}$ *is capable of* $O(h^{3.5})$ *convergence under the* ℓ_∞*-norm when* $c = 3h$*; see figure 2.5.*

Lastly, the derivative values at the boundaries do not affect the convergence rate of $\mathcal{L}_{\mathcal{D}}$*.*

Example 2.1.3 (Effect of second derivative at boundaries) *Again we examine the claim in case 2.1.2 of section 2.1.2. The functions tested are*

$$f_5(x) = \sin(\pi x) + 0.1\sin(32\pi x),$$

and

$$f_6(x) = \sin(\pi x) + 0.1\cos(32\pi x).$$

Both functions are oscillating at the same frequency. We construct f_5 *to have vanishing second derivatives at the boundaries.*

As suggested, the results are dramatically different. The behavior of $\mathcal{L}_{\mathcal{R}} f_6$ *is similar to the functions in example 2.1.1; whereas that of* $\mathcal{L}_{\mathcal{R}} f_5$ *is similar to the functions in example 2.1.2. In terms of errors, the minimum* 6.98×10^{-6} *for* $\mathcal{L}_{\mathcal{R}} f_5$ *is also a magnitude smaller than the* 3.09×10^{-5} *for* $\mathcal{L}_{\mathcal{R}} f_6$*.*

Example 2.1.4 (Comparing with traditional MQ) *In many practical applications, there are only few experimental data available. In this example, we compare our quasi-interpolation schemes with the traditional MQ-RBF interpolation (with appended constant) on a smaller set of data points applying to two smooth test functions*

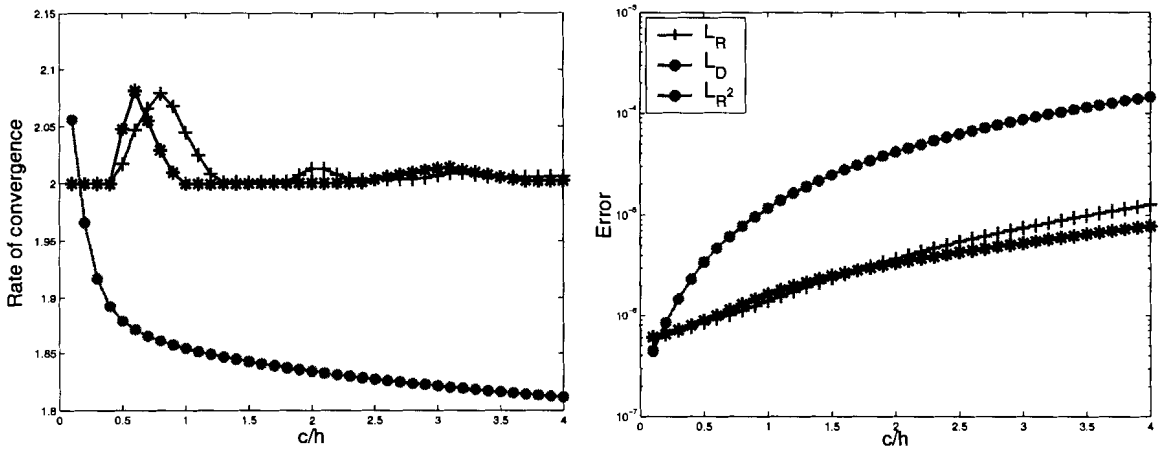

Figure 2.2: Rate of convergence and error for test function $f_1(x) = \sin(4.5x)$

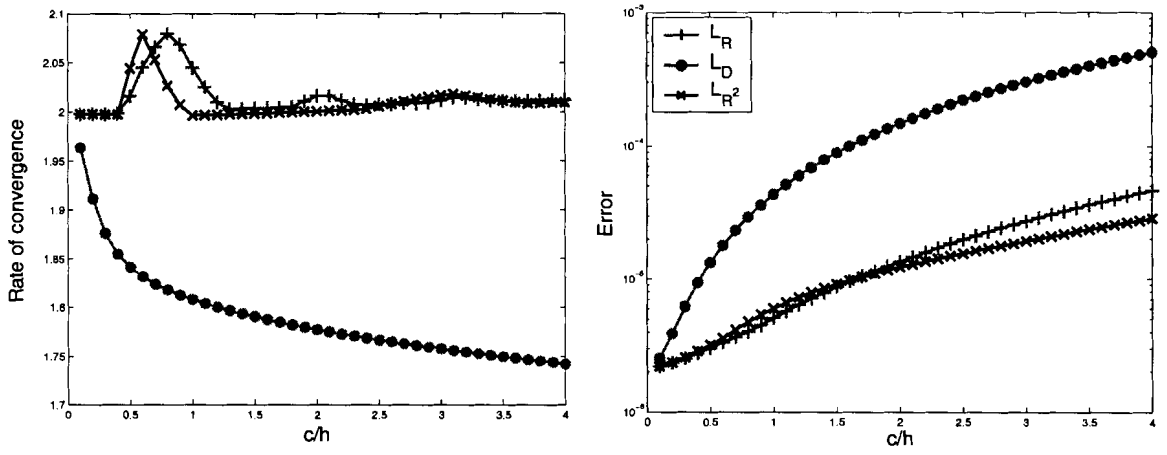

Figure 2.3: Rate of convergence and error for test function $f_2(x) = x^9$

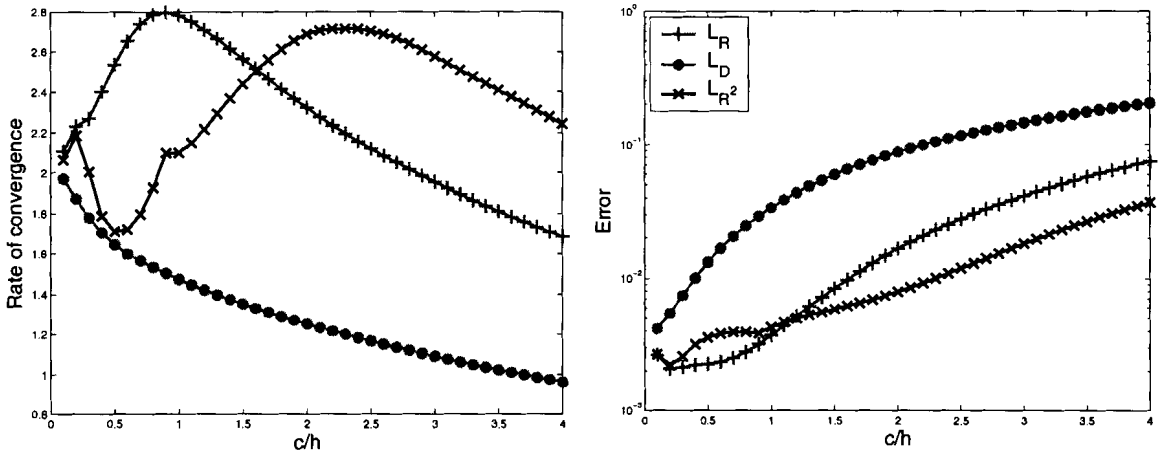

Figure 2.4: Rate of convergence and error for test function $f_3(x) = \exp(-2^{16}(x-0.8)^2)$

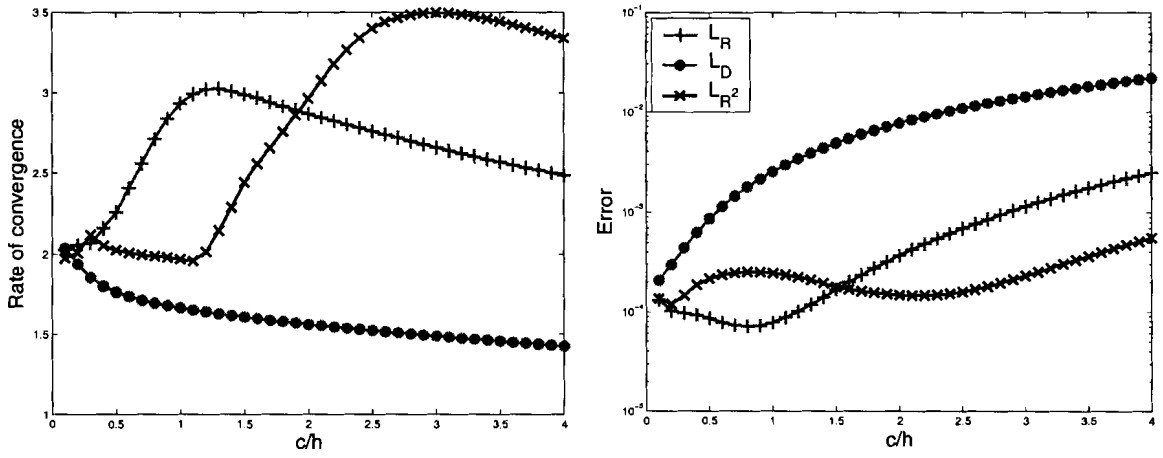

Figure 2.5: Rate of convergence and error for test function $f_4(x) = \arctan(100(x-0.3))$

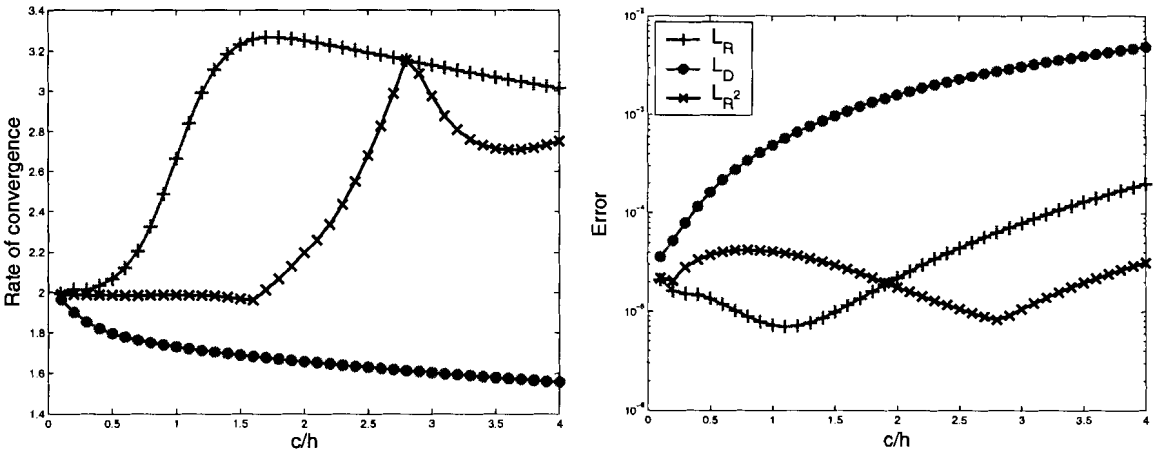

Figure 2.6: Rate of convergence and error for test function $f_5(x) = \sin(\pi x) + 0.1\sin(32\pi x)$

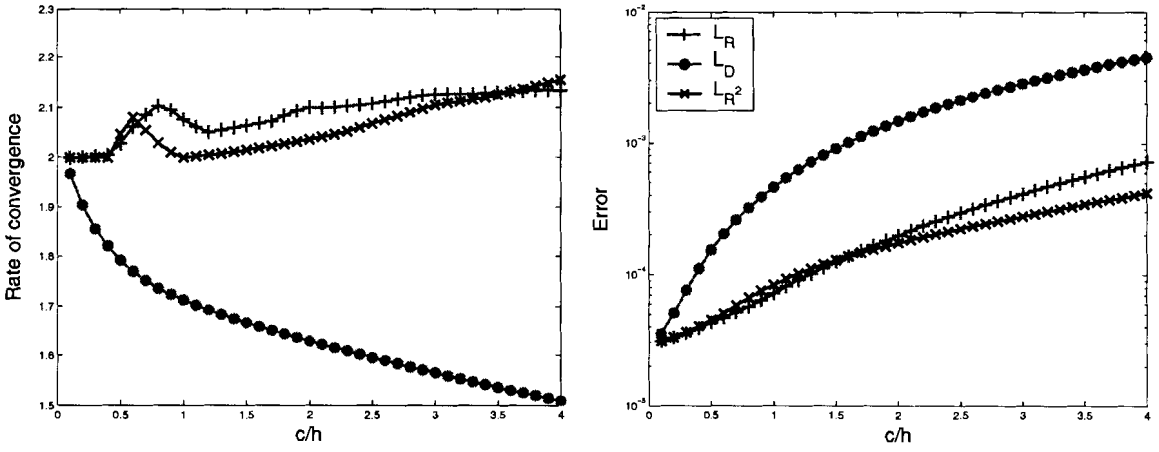

Figure 2.7: Rate of convergence and error for test function $f_6(x) = \sin(\pi x) + 0.1\cos(32\pi x)$

	$\mathcal{L}_{\mathcal{D}}$	$\mathcal{L}_{\mathcal{R}}$	$\mathcal{L}_{\mathcal{R}^2}$	MQ
$n = 13$	1.49(-1)	4.58(-2)	3.49(-2)	1.55(-2)
29	3.79(-2)	8.20(-3)	6.69(-3)	6.55(-3)
57	1.17(-2)	1.99(-3)	1.70(-3)	3.26(-3)
113	3.48(-3)	4.88(-4)	4.28(-4)	1.63(-3)

Table 2.1: Maximum errors of our quasi-interpolation schemes and the traditional MQ-RBF applying to f_1 with different number of data points.

	$\mathcal{L}_{\mathcal{D}}$	$\mathcal{L}_{\mathcal{R}}$	$\mathcal{L}_{\mathcal{R}^2}$	MQ
$n = 13$	1.82(-1)	1.12(-1)	9.74(-2)	4.79(-2)
29	6.02(-2)	2.59(-2)	2.18(-2)	1.64(-2)
57	2.14(-2)	6.89(-3)	5.83(-3)	7.38(-3)
113	7.05(-3)	1.76(-3)	1.51(-3)	3.47(-3)

Table 2.2: Maximum errors of our quasi-interpolation schemes and the traditional MQ-RBF applying to f_2 with different number of data points.

f_1 and f_2 introduced in example 2.1.1. The maximum errors are evaluated at 2^{12} equally spaced points, and the shape parameters used in this example are $c = h$.

We summarize the results in table 2.1 and table 2.2, for f_1 and f_2, respectively. In both tables, we underline the results of quasi-interpolations that outperform traditional MQ-RBF. From table 2.1 and table 2.2, Wu-Schaback's formula $\mathcal{L}_{\mathcal{D}}$ is less accurate than the traditional MQ-RBF for all n. When $n = 23$, we see that both the level-1 and level-2 multilevel scheme $\mathcal{L}_{\mathcal{R}}$ and the $\mathcal{L}_{\mathcal{R}^2}$ have errors of the same magnitude as the traditional MQ-RBF, and they are more accurate than traditional MQ-RBF when $n = 57$ and $n = 113$. Our multilevel quasi-interpolation scheme is a very attractive alternative to the traditional MQ-RBF not only in terms of efficiency but also in terms of accuracy.

2.1.4 Discussion

For all examples in Section 2.1.3, $c = h$ appears to be a good shape parameter for $\mathcal{L}_{\mathcal{D}}$. For all test functions, both convergence rates and errors are optimized for c between $0.5h$ and $1.5h$. The level-2 scheme is a good choice if one desires an even higher degree

of smoothness. Its shape parameter should be chosen between $2h$ to $3h$.

We remark that we used two fixed data points to estimate the convergence rate of our scheme with different ratio of c/h. The convergence rates of $\mathcal{L}_{\mathcal{D}}$ in the ℓ_2-norm are consistently better than in the ℓ_∞-norm. For example 2.1.1, we obtain a consistent $\mathcal{O}(h^{2.5})$ convergence rate for $1.4h \leq c \leq 4h$ for both functions. For the other functions in example 2.1.2 and example 2.1.3, faster than cubic convergence rates are obtained.

2.2 Multidimensional Interpolation

Because the methods are matrix-free and fast, quasi-interpolation in higher dimensions are even more attractive in practice. There are no shortage of engineering problems that can be solved once a robust multidimensional quasi-interpolation scheme is successfully developed. In this paper, we suggest a possible extension of the multilevel quasi-interpolation scheme using the dimension-splitting MQ (DSMQ) basis that is readily preformed on parallel computers. We show that the cost of finding the coefficients for the DSMQ basis is $3dN$ on $\mathbb{R}^d$, and the work of direct evaluation of the quasi-interpolant can be reduced from $11N^2$ in 2D and $16N^2$ in 3D to $\approx 2N$. A boundary padding technique is employed to improve accuracy. Examples in 2D and 3D are both given.

In [10], a quasi-interpolation formula for thin-plate splines on a square was proposed by Beatson and Light. Moreover, Li, Ho and Chen [76] introduced various quasi-interpolations in high dimensional case using radial function. While one could develop a quasi-interpolation formula using the MQ basis on arbitrary finite subsets of $\mathbb{R}^d$, it is much convenient to relax some of the conditions. In this section, we extend the Wu-Schback [102] univariate quasi-interpolation formula to two dimension on rectangular grid. The extension to 3D or higher dimension is straightforward. Then a multilevel formulation will be given in section 2.3.

Similar to the case of Lagrange interpolation, one could consider the basis functions $\alpha_i(x)$ in (2.6) of Wu-Schaback's formula (2.5) as the fundamental functions of the

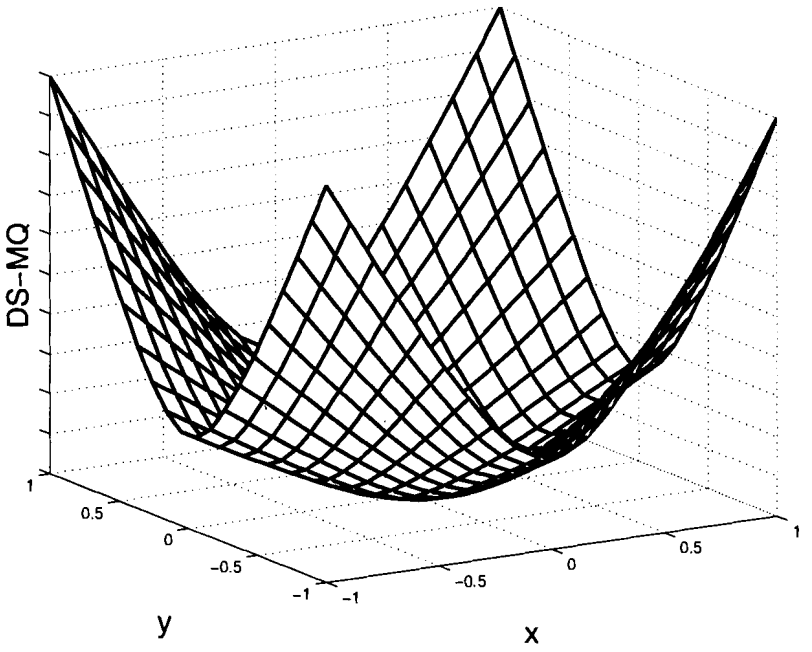

Figure 2.8: A dimension-splitting MQ (DSMQ) basis centered at the origin.

quasi-interpolation with MQ basis. Instead of the two dimensional MQ basis

$$\sqrt{(x-x_i)^2+(y-y_j)^2+c^2},$$

our two-dimensional quasi-interpolation scheme uses the *dimension-splitting* multi-quadric (DSMQ) basis,

$$\sqrt{(x-x_i)^2+c^2}\cdot\sqrt{(y-y_j)^2+c^2}, \tag{2.21}$$

as the interior basis. Figure 2.8 shows an example of a DSMQ. Readers are referred to Mayers et al. [88] for the studies the positive definiteness, product-sum, integrated product, and the integrated product-sum of the DS-RBF.

Given data $\{x_i, y_j, F_{ij}\}$ for $i = 0, \ldots, n$, and $j = 0, \ldots, m$. Let the 1D MQ basis in the y variable be

$$\psi_j(y) = \sqrt{(y-y_j)^2+c^2},$$

and define a set of basis functions or interpolation kernel by

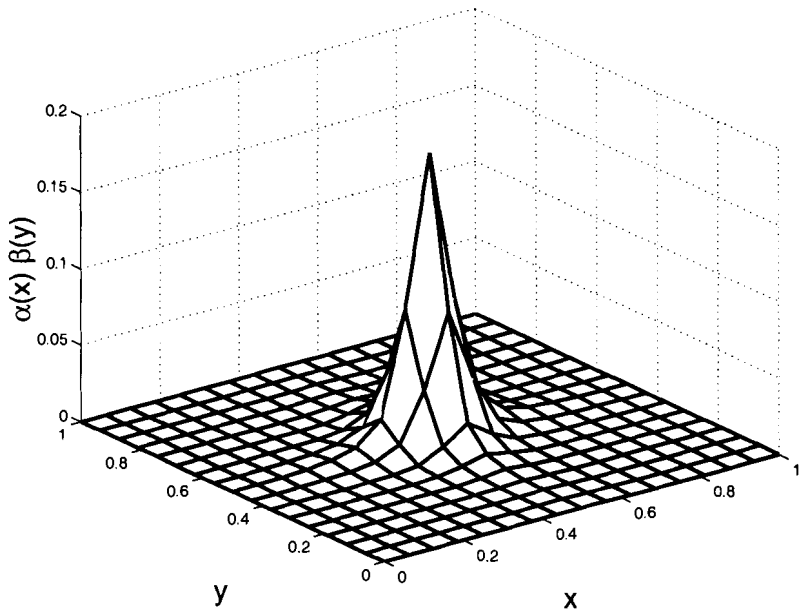

Figure 2.9: A fundamental functions of the 2-D quasi-MQ interpolation, $\alpha(x)\beta(y)$.

$$
\begin{aligned}
\beta_0(y) &= \frac{1}{2} + \frac{\psi_1(y) - (y - y_0)}{2(y_1 - y_0)}, \\
\beta_1(y) &= \frac{\psi_2(y) - \psi_1(y)}{2(y_2 - y_1)} - \frac{\psi_1(y) - (y - y_0)}{2(y_1 - y_0)}, \\
\beta_j(y) &= \frac{\psi_{j+1}(y) - \psi_j(y)}{2(y_{j+1} - y_j)} - \frac{\psi_j(y) - \psi_{j-1}(y)}{2(y_j - y_{j-1})}, \quad \text{for } 2 \le j \le m-2, \\
\beta_{m-1}(y) &= \frac{(y_m - y) - \psi_{m-1}(y)}{2(y_m - y_{m-1})} - \frac{\psi_{m-1}(y) - \psi_{m-2}(y)}{2(y_{m-1} - y_{m-2})}, \\
\beta_m(y) &= \frac{1}{2} + \frac{\psi_{m-1}(y) - (y_m - y)}{2(y_m - y_{m-1})},
\end{aligned}
\tag{2.22}
$$

as in the univariate case appling to the variable y. The (level-0) two dimensional quasi-interpolation formula is then given by

$$
\mathcal{L}_0 F(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} F_{ij} \alpha_i(x) \beta_j(y), \tag{2.23}
$$

where the function $\alpha_i(x)$ is given by (2.6). Note that the product $\alpha_i(x)\beta_j(y)$ for $i = 2, \ldots, n-2$, and $j = 2, \ldots, m-2$ results in the DSMQ basis as defined in (2.21).

2.2.1 Implementation for parallel computations

Using the DSMQ basis allows one to decouple problems in higher space dimensions into many one dimensional problems. In this section, we provide a cost analysis of finding coefficients of our quasi-interpolant, and a direct but faster way to evaluate the resulting quasi-interpolant.

Finding coefficients of DSMQ

Let the basis $\phi_j(x) = \sqrt{(x-x_j)^2+c^2}$ be the 1D MQ basis for $1 \le j \le n-1$, coupled with polynomials, $\phi_n(x) \equiv 1$ and $\phi_{n+1}(x) = x$. We first consider the univariate formula (2.12) as a changing basis from $\{\alpha_i\}$ to $\{\phi_i\}$,

$$\mathcal{L}_0 F(x) = \sum_{j=0}^{n} F_j\, \alpha_j(x) = \sum_{j=0}^{n} \lambda_j\, \phi_j(x),$$

Then (2.12) can be expressed in matrix form as

$$(T_{\alpha\to\phi})\, F = \Lambda, \tag{2.24}$$

where $F = [F_0, F_1, \cdots, F_n]^T$ and $\Lambda = [\lambda_0, \lambda_1, \cdots, \lambda_n]^T$ are column vectors of length $n_x := n+1$. The $n_x \times n_x$ matrix $T_{\alpha\to\phi}$ is given by

$$T_{\alpha\to\phi} = \frac{1}{2}\left[\begin{array}{cccccccc} D_1 & E_2 & D_2 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & D_2 & E_3 & D_3 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & D_{n-2} & E_{n-1} & D_{n-1} & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & D_{n-1} & E_n & D_n \\ \hline 1+x_0D_1 & -x_1D_1 & 0 & \cdots & \cdots & 0 & x_nD_n & 1-x_nD_n \\ -D_1 & D_1 & 0 & \cdots & \cdots & 0 & -D_n & D_n \end{array}\right] \tag{2.25}$$

where $D_j = 1/(x_j - x_{j-1})$ for $1 \le j \le n$, and $E_j = -D_j - D_{j-1}$ for $2 \le j \le n$. The top part of $T_{\alpha\to\phi}$ is an upper-triangular banded-diagonal matrix. The last two

rows of $T_{\alpha\to\phi}$ has only four nonzero elements. By exploring the sparsity of $T_{\alpha\to\phi}$ or considering $T_{\alpha\to\phi}$ as a filter, finding all coefficients for $\{\phi_j\}$ from F costs $\approx 3n_x$ flops. The benefit of using $\{\phi_j\}$ as the basis is that the evaluation of the quasi-interpolant is more efficient.

To compute the basis transform in 2D, from $\{\alpha_i \otimes \beta_j\}$ to $\{\phi_i \otimes \psi_j\}$, we consider the problem as n_x one-dimensional problems of length n_y, and n_y one dimensional problems of length n_x, where n_x and n_y are the total number of data points in the x- and y-direction, respectively. The cost of the basis transform is therefore $n_x(3n_y) + n_y(3n_x) = 6n_xn_y = 6N$ in 2D, where N is the total number of data points.

Suppose we store the function values F_{ij} in a two dimensional array of size $n_x \times n_y$, denoted by F. Then we can rewrite the 2D formula (2.23) as

$$\mathcal{L}_0 F(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} [(T_{\alpha\to\phi})\, F\, (T_{\beta\to\psi})^T]_{ij}\ \phi_i(x)\,\psi_j(y), \tag{2.26}$$

where we define the set of 1D DSMQ basis $\{\psi(y)\}$ in the same way as the $\{\phi(x)\}$ but in the y-direction. To state (2.26) in word, each column of F_{ij} is updated with the action of $T_{\alpha\to\phi}$, then each resulting row is updated with the action of $T_{\beta\to\psi}$. Since the computations applied on each column of F_{ij}, (or each row of $(T_{\alpha\to\phi})F_{ij}$) are independent of the other columns (or rows), the evaluation of the function values F_{ij} if applicable, can be performed in parallel. The performance would be optimized if both n_x and n_y are multiples of the number of available processors.

A similar argument can be applied to 3D if we store the function values F_{ijk} in a 3D array. It is easy to verify that the cost of transform is $9N$.

Evaluating quasi-interpolant

We provide details in two space dimension; the generalization to higher dimension is again straightforward. Suppose the coefficients λ_{ij} for the DSMQ quasi-interpolation is found,

$$\mathcal{L}_0 F(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} \lambda_{ij}\,\phi_i(x)\,\psi_j(y), \tag{2.27}$$

where $\lambda_{ij} = [(T_{\alpha\to\phi})\, F\, (T_{\beta\to\psi})^T]_{ij}$. To evaluate (2.27), for each data point (or for each i and j), one needs to evaluate $\phi_i(x)\,\psi_j(y)$ at all data points (x, y) that costs $9N$ flops; premultiplying λ_{ij} costs N flops; lastly, the cumulative sum requires another N additions to update. We estimate the cost of evaluating (2.27) to be $11N^2$ in 2D, and $16N^2$ in 3D. In this section, we show that it is, in fact, cheaper than evaluating a 2D MQ expansion directly due to the special structure of our scheme. We reduce the evaluation cost from $11N^2$ (2D) or $16N^2$ (3D) to $2N^2$.

We introduce more notations: let Φ_{ij}, and Ψ_{ij} be the $n_x \times n_y$ matrices with the values of $\phi_i(X)$ and $\psi_j(Y)$ evaluated at all data points X, Y (stored also as $n_x \times n_y$ matrices). Let $\circledast$ denotes the element-by-element multiplication operator on arrays of the same size. The quasi-interpolant (2.27) evaluated at all data centers can be rewritten as

$$\mathcal{L}_0 F(X,Y) = \sum_{i=0}^{n}\sum_{j=0}^{m}(\lambda_{ij}\,\Phi_{ij}) \circledast \Psi_{ij} = \sum_{i=0}^{n}\sum_{j=0}^{m}\Phi_{ij} \circledast (\lambda_{ij}\,\Psi_{ij}). \tag{2.28}$$

Since our scheme works on rectangular grid, both Φ_{ij} and Ψ_{ij} are rank one matrices,

$$\begin{aligned}\Phi_{ij} &= \phi_i(x_*)\cdot \mathbb{1}_{n_y}^T \\ \Psi_{ij} &= \mathbb{1}_{n_x}\cdot \psi_i(y_*)^T,\end{aligned}$$

where $\mathbb{1}_{\cdot}$ is a vector of all ones with the indicated length, $x_* = [x_0, \ldots, x_n]^T$, and $y_* = [y_0, \ldots, y_m]^T$. Therefore,

$$\Phi_{ij} \circledast \Psi_{ij} = (\phi_i(x_*)\cdot \mathbb{1}_{n_y}^T) \circledast (\mathbb{1}_{n_x}\cdot \psi_i(y_*)^T) = \phi_i(x_*)\cdot \psi_i(y_*)^T. \tag{2.29}$$

Beatson's fast multipole method can be employed to evaluate the n_x and n_y 1D MQ basis functions, $\phi_i(x)$ and $\psi_i(y)$ in the x- and y-direction, has an one-time set up cost of $4n_x \log n_x + 4n_y \log n_y$ flops. The details of the multipole method can be found in [8, 9].

Computing the rank one matrix in (2.29) before multiplying λ_{ij} would cost $N = n_x n_y$ work for the scalar-matrix multiplication; instead, a scale-vector multiplication only costs $\min(n_x, n_y)$. For each inner iteration, we shall premultiply λ_{ij} with either the vector $\phi_i(x_*)$ or $\psi_i(y_*)$ as in (2.28) whichever is shorter in length, i.e., we compute

Algorithm 2 Pseudo-code for the 2D multilevel quasi DSMQ interpolation.

INPUT: $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, $F \in \mathbb{R}^{(n_x)\times(n_y)}$

OUTPUT: $\lambda \in \mathbb{R}^{(n_x)\times(n_y)}$

Find separating distance $h = \min\left(\max_{1\le i\le n_x}(x_i - x_{i-1}), \max_{1\le j\le n_y}(y_j - y_{j-1})\right)$

Pick a shape parameter $c = \mathcal{O}(h)$

Find coarse data $x_{1/2} = (\downarrow 2)(x)$, $y_{1/2} = (\downarrow 2)(y)$, $F_{1/2} = (\downarrow 2)(F)$

Find coefficients $\lambda = \mathrm{Coef}(F_{1/2})$,

Find error $\mathcal{E} = F - \mathcal{L}_{0[1/2]}F(x, y)$

Update coefficients $\lambda \longleftarrow \mathrm{Coef}(\mathcal{E}) + (\uparrow 2)(\lambda)$,

either $[\lambda_{ij}\,\phi_i(x_*)] \cdot \psi_i(y_*)^T$ or $\phi_i(x_*) \cdot [\lambda_{ij}\,\psi_i(y_*)]^T$. Lastly, the work of updating the cumulative sum by this rank one matrix is $2N$ flops.

The total cost of evaluating a 2D DSMQ quasi-interpolant is reduced to

$$[4n_x \log n_x + 4n_y \log n_y] + N \cdot [\min(n_x, n_y) + 2N] \approx 2N^2.$$

By constructing a 3D array of values using three vectors of 1D MQ, one can easily shows that the same cost estimate $\approx 2N^2$ also holds in 3D.

2.3 Multilevel Interpolation

The idea of multilevel scheme is common in numerical analysis that has been used in many areas of applied mathematics. For instance, multilevel schemes with compactly supported radial function can be found in [37, 23]. Our extension to multilevel is

similar to the univariate case given in the previous section. For the level-1 scheme, we first quasi-interpolate on (roughly) half of the data points of size $(n_x/2) \times (n_y/2)$, then compute the residual on all data points, and finally quasi-interpolate the residual to update the corresponding coefficients of the quasi-interpolant.

The pseudo-code of finding the undetermined coefficients of the multilevel quasi DSMQ interpolation is given in algorithm 2 where $(\downarrow 2)$ is any function that downsamples a given vector to about half of its original size by keeping every second element and $(\uparrow 2)$ is the inverse of $(\downarrow 2)$ that upsamples by inserting zero coefficients between input elements. The function "*Coef*" maps the input data to the coefficients defined by (2.23).

It is easy to verify that the level-0 scheme (2.23) reproduces constant and linear polynomials exactly. According to Strang and Fix [36], (2.23) will enjoy a quadratic convergence with respect to h asymptotically. In fact, one could apply theorem 2.1.5 dimension by dimension, and show that the level-1 quasi-interpolant $\mathcal{L}_0 F$ converges to $F : \mathbb{R}^d \rightarrow \mathbb{R}$ at a speed of $\mathcal{O}(d \cdot h^{2.5} \log h)$ provided $c = \mathcal{O}(h)$. Similar to the univariate case, the second partial derivative values of F at boundary will have an effect on the convergence rate. Therefore, the convergence rate of algorithm 2 will be affected by rotation in general cases.

2.3.1 Boundary padding

From (2.5), we see that the constant and linear polynomials are employed at the boundary and the point next to the boundary. Information from x_0, x_1, x_{n-1}, and x_n are associated with these (appended) polynomials. It is reasonable to extend the boundary outwards with some *padding points* so that the centers of the polynomial parts are pushed out of the region of interest. In fact, Hon [57] removes the polynomial part in a quasi-interpolation formula completely by estimating the error of the truncated part. This extra degree of accuracy does not come freely: using more padding points means more work. But we are going to see that only a small amount of padding is necessary to achieve the near-optimal accuracy.

The function values of F at the out of boundary padding points could be computed

using the formula of F, i.e., we solve the same quasi-interpolation problem on a slightly expanded domain, but evaluate the quasi-interpolant only on the original domain in which each data point will be associated with an infinite differentiable DSMQ basis. In practice, if a closed form of the function F is not available, some simple extrapolation scheme could be used to approximate those values. These techniques are being studied widely in different areas of image processing, for instance see [29, 30, 90]. Since our scheme treats each dimension separately, one only need to extrapolate the out of boundary values of F dimension by dimension. Complicated schemes are not necessary. We have seen that having a small second derivative helps speeding up the rate of convergence of the multilevel scheme. This suggests a linear extrapolation, that results in a small second (partial) derivative at the new extended boundary, would be sufficient.

Using the multilevel scheme allows one to use a larger shape parameter c while keeping a high convergence rate and high accuracy. The resulting approximations also enjoy higher degree of smoothness. Since the constant and linear polynomial parts in (2.23) vanish or reduce in order under differentiation, boundary padding is highly recommended when one needs to use quasi-interpolation to approximate derivative values.

2.3.2 Numerical examples

Since one cannot arbitrary reduce h (especially in higher dimensions), we focus on the accuracy and estimated convergence rate of our scheme in this section. We first give an example to study the convergence rate of our level-1 scheme in 2D for various ratio of c/h. Then we look at the residual functions of two different shape parameters: small parameters results in better point-wise accuracy; whereas, large shape parameter results in higher degree of smoothness. Lastly, we show some results in 3D. The maximum errors (MAX) of all examples are taken to be the maximum absolute difference between the quasi-interpolant and the original function.

Example 1: Convergence rate.

Firstly, we look at the convergence behaviour for different $c = \mathcal{O}(h)$. We consider a 2D function

$$F(x, y) = \left(x - \frac{1}{2}\right)^2 \sin y, \qquad (x, y) \in [0, 1]^2, \tag{2.30}$$

on the unit square. The function (2.30) has both maximum x and y second partial derivatives along the boundary: F_{xx} obtains its maximum along $y = 1$ and F_{yy} is maximized at $(x, y) = (0, 1)$ and $(1, 1)$.

The shape parameter of DSMQ is related directly to the univariate case: $c = \mathcal{O}(h)$, where h denotes the node spacing.

We quasi-interpolate on 51×51 and 101×101 equally spaced data points to estimate the convergence rate (in figure 2.10), and report the ℓ_∞-norm error measured on the fine spacings (in figure 2.11). In both figures, the direct application of formula (2.23) is denoted by level-0; and level-1 denotes one quasi-interpolation update on the residual.

Except for the first tested c/h ratio, the convergence rates of the level-1 scheme are all above 2 and seem to be increasing as c/h increases. The maximum error of the level-1 scheme increases slower than that of the level-0 as the ratio c/h increases; i.e., using the multilevel scheme can achieve a higher degree of smoothness.

In general, the level-0 schemes in all dimensions output the coefficients for the DSMQ basis that depend only on the input function values of F, but are independent of the choice of shape parameters. Using our multilevel scheme, each computation of the residual function provides a feedback that reflect the influence of the chosen shape parameter. Thus, the level-1 scheme not only shows faster convergence, but also better accuracy.

Example 2: Smoothness.

From figure 2.10, the maximum errors of the level-1 scheme are 1.9×10^{-4} and 5.1×10^{-4} when $c = h$ and $c = 2h$, respectively. Using a larger shape parameter results in less accurate solutions, but the larger shape parameter results in a quasi-interpolant with

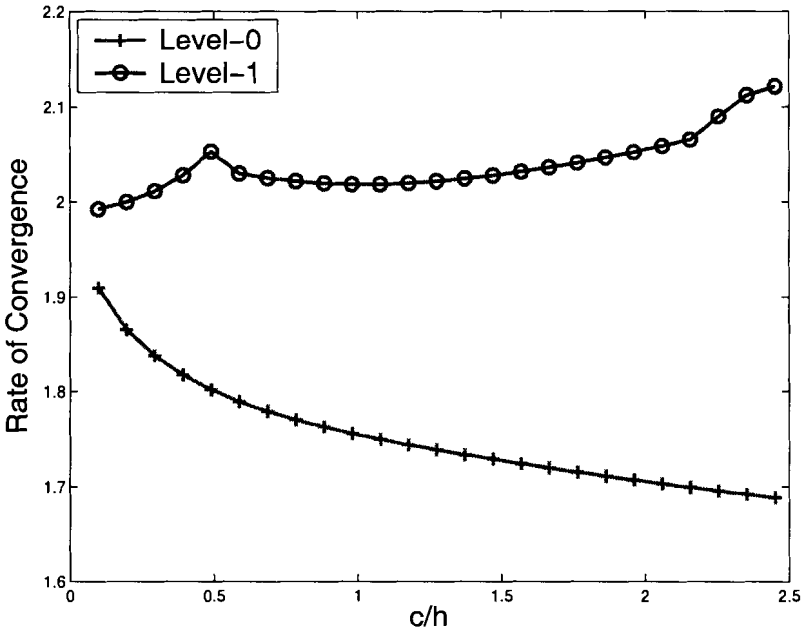

Figure 2.10: Estimated convergence rate of the 2-D quasi interpolation, without boundary padding, on function (2.30) as a function of c/h.

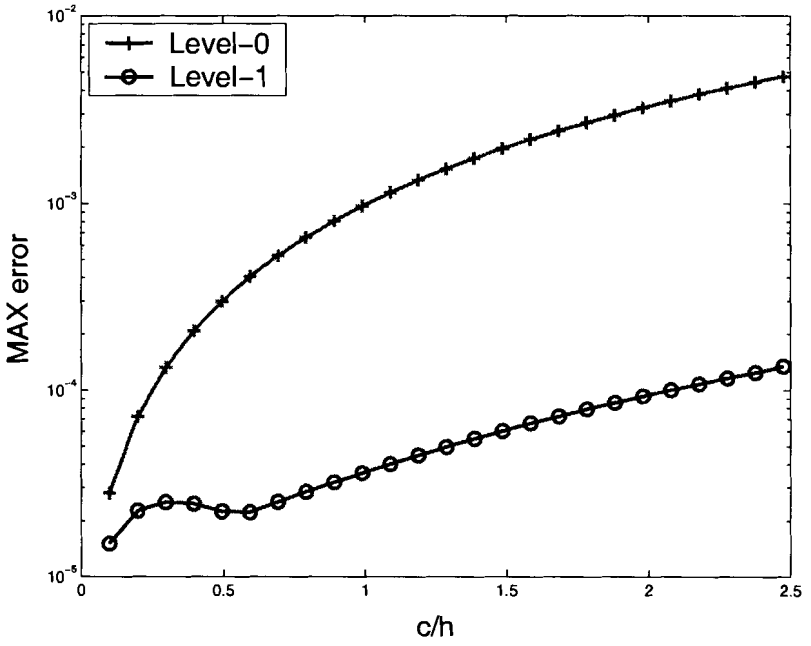

Figure 2.11: Maximum error corresponding to figure 2.10.

No. of padding	0	1	2	3	4	5
MAX error	3.59(-5)	1.02(-5)	1.01(-5)	9.94(-6)	1.00(-5)	9.89(-6)

Table 2.3: Maximum error of the level-1 scheme with different amount of paddings.

higher order of smoothness. We show the residual functions of both cases in figure 2.12 and figure 2.13. The residual function of $c = h$ is very oscillatory, but that of $c = 2h$ is significantly smoother as one expected.

Example 3: Boundary padding.

For the function F defined by (2.30), we look at the maximum error of the level-1 scheme with different amount of paddings using 101×101 equally spaced data points. The shape parameters are kept constant for all cases, $c = h$.

The values outside the original domain, at the out of boundary padding points, are computed using (2.30).

The results are given in table 2.3. To obtain the benefit of boundary padding, one only needs to have one padding point in order to achieve the near-optimal accuracy. By having one padding point, the maximum error reduces by a factor of 3.5.

Example 4: Approximating derivatives

Our next example considers the following function

$$F(x, y) = x(x-1)\sin(\pi y), \qquad (x, y) \in [0, 1]^2, \tag{2.31}$$

whose second partial derivatives obtain a maximum in the interior. We extend the boundary outwards with 3 padding points, and study the convergence of the level-1 scheme $\mathcal{L}_1 F$, and its derivatives $\frac{\partial}{\partial x}\mathcal{L}_1 F$, $\frac{\partial^2}{\partial x^2}\mathcal{L}_1 F$, and $\frac{\partial^2}{\partial x \partial y}\mathcal{L}_1 F$.

Figure 2.14 shows the convergence rates for all cases. With boundary padding, the convergence rate of $\mathcal{L}_1 F$ increases slowly with the ratio c/h, and does not oscillate as in the case of no padding.

If one differentiates the quasi-interpolation formula (2.23) once, the resulting formula can exactly reproduce constant functions but not all linear functions. Thus the

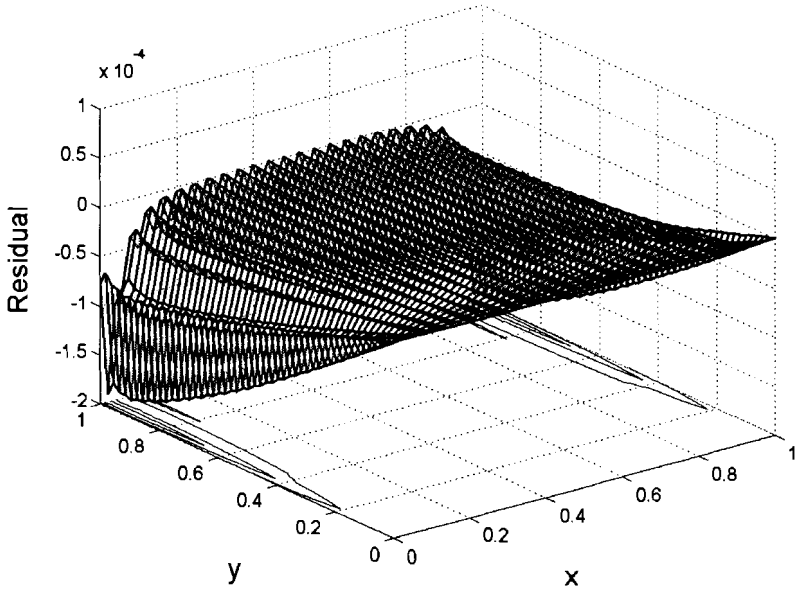

Figure 2.12: Residual function of the level-1 scheme applying to (2.30) with $c = h$.

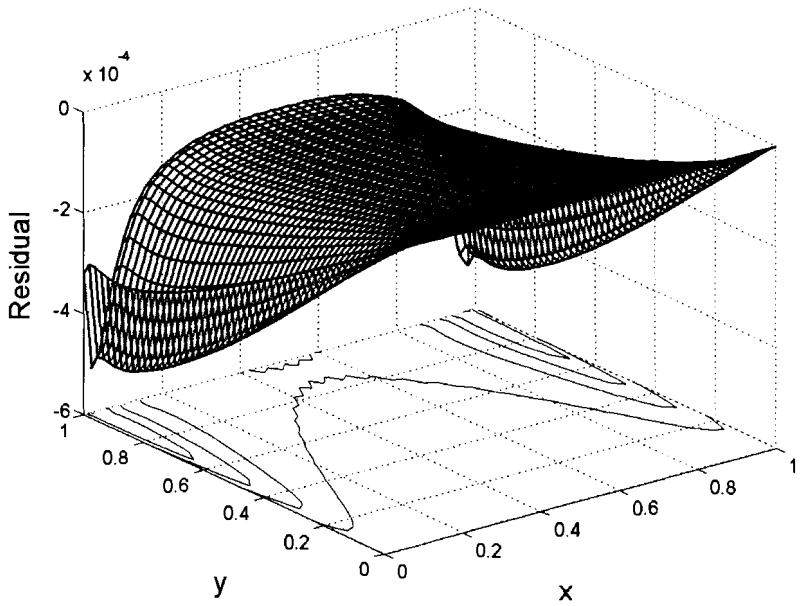

Figure 2.13: Residual function of the level-1 scheme applying to (2.30) with $c = 2h$.

convergence rate of $\frac{\partial}{\partial x}\mathcal{L}_1 F$, the $\frac{\partial}{\partial x}$-derivative of the level-0 quasi-interpolant, is expected to be linear according to [36]. The convergence rates of $\frac{\partial}{\partial x}\mathcal{L}_1 F$ and $\frac{\partial^2}{\partial x \partial y}\mathcal{L}_1 F$ behave similarly. Both are very high when c/h is small, and rapidly slows down as c/h increases. Note that when $c = h$, both approximations $\frac{\partial}{\partial x}\mathcal{L}_1 F$ and $\frac{\partial^2}{\partial x \partial y}\mathcal{L}_1 F$ converge roughly at quadratic speed.

In the case of no boundary padding, we consistently observe a superlinear convergence for $\frac{\partial}{\partial x}\mathcal{L}_1 F$ and $\frac{\partial^2}{\partial x \partial y}\mathcal{L}_1 F$. Since the quasi-interpolation formula (2.23) reproduces all function values in Π_1^2, it can only reproduce constant functions after a partial differentiation or a mixed differentiation. We therefore expect a drop of one order in the convergence rate.

After differentiating the quasi-interpolation formula (2.23) twice with respect to x (or y), the new formula no longer preserves constant functions, we therefore observe almost no convergence in the approximation of $\frac{\partial^2}{\partial x^2}\mathcal{L}_1 F$, see figure 2.14.

From figure 2.15, we see that the approximation of $\frac{\partial^2}{\partial x^2}\mathcal{L}_1 F$ depends only on the smoothness; we obtained a better approximation of $\frac{\partial^2}{\partial x^2}\mathcal{L}_1 F$ as the ratio c/h increases.

Example 5: 3D functions.

Our last example shows some results in three dimension. We consider the following functions,

$$\begin{aligned} F_1(x,y,z) &= x^2 y^2 z^2, \\ F_2(x,y,z) &= (x-0.5)^2 \sin(y) \cos(z), \\ F_3(x,y,z) &= \sin(\pi x) \sin(\pi y) \sin(\pi z), \\ F_4(x,y,z) &= \sin(\pi x y z). \end{aligned}$$

We quasi-interpolate using the level-0, level-1, and level-2 schemes on both $N = 11^3$ and $N = 23^3$ equally spaced data points. Three boundary padding points are used for all tests. The results for $N = 11^3$ are listed in table 2.4. Our level-1 and level-2 with $N = 11^3$ data points is sufficient to accurately approximate the slowly varying function F_1 and F_2 for both values of c. Halving the data point spacing allows us to

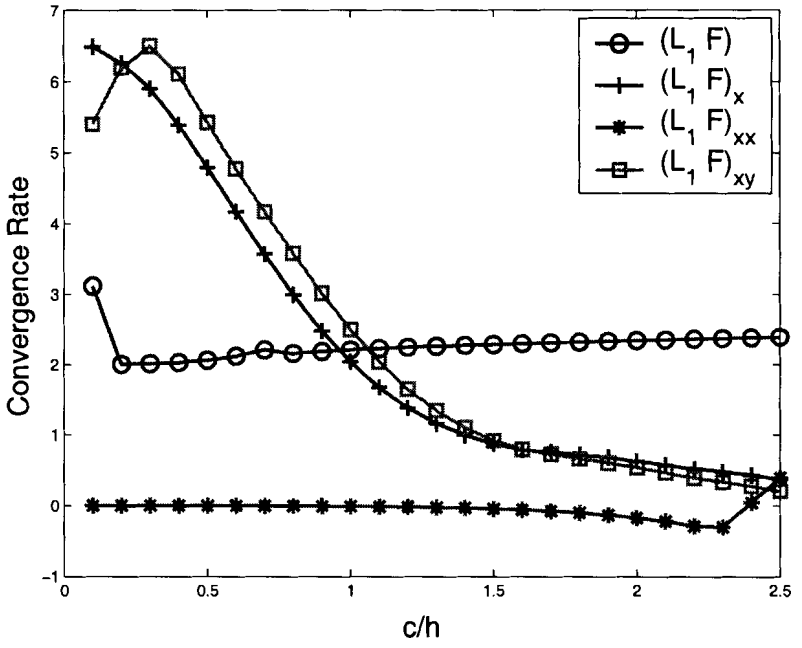

Figure 2.14: Estimated convergence rate of the 2-D quasi interpolation, with 3 points boundary padding, on function (2.31) as a function of c/h.

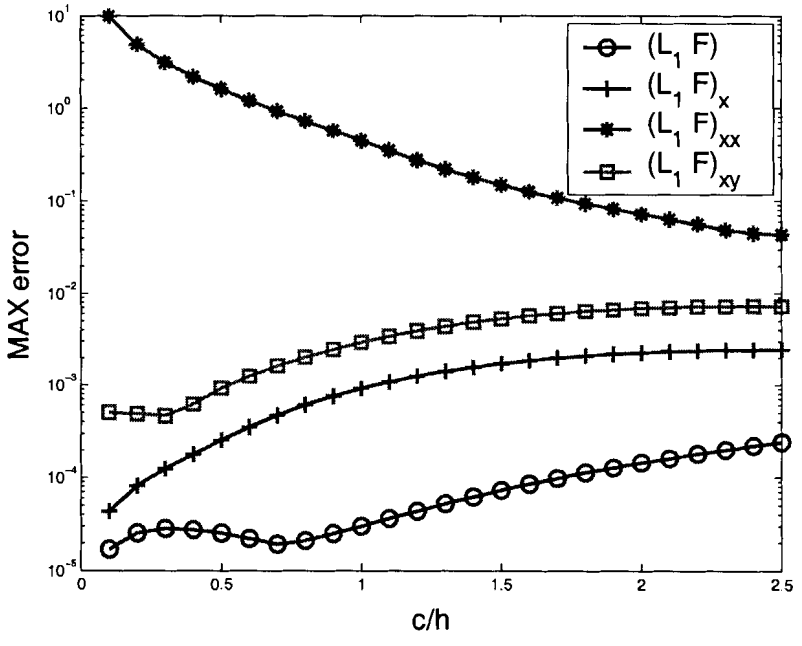

Figure 2.15: Maximum error corresponding to figure 2.14.

$N = 11^3$	$c = h$			$c = 2h$		
	Level-0	Level-1	Level-2	Level-0	Level-1	Level-2
F_1	9.57(-2)	3.90(-3)	4.27(-3)	3.02(-1)	2.36(-2)	3.79(-3)
F_2	2.76(-2)	3.36(-3)	1.86(-3)	7.97(-2)	1.81(-2)	4.38(-3)
F_3	3.01(-1)	9.04(-2)	2.11(-2)	6.60(-1)	4.27(-1)	1.42(-1)
F_4	1.85(-1)	5.98(-2)	2.09(-2)	4.45(-1)	2.42(-1)	1.06(-1)

Table 2.4: Maximun error of 3D multilevel quasi-interpolants with $c = h$ and $c = 2h$ with $N = 11^3$ and 3 boundary padding points.

$N = 23^3$	$c = h$			$c = 2h$		
	Level-0	Level-1	Level-2	Level-0	Level-1	Level-2
F_1	2.14(-2)	1.53(-3)	1.15(-3)	6.68(-2)	6.20(-3)	1.34(-3)
F_2	6.85(-3)	5.14(-4)	4.15(-4)	2.15(-2)	2.40(-3)	5.59(-4)
F_3	9.38(-2)	9.72(-3)	6.44(-3)	2.63(-1)	6.35(-2)	5.76(-3)
F_4	5.35(-2)	5.71(-3)	3.57(-3)	1.56(-1)	3.01(-2)	6.59(-3)

Table 2.5: Maximun error of 3D multilevel quasi-interpolants with $c = h$ and $c = 2h$ with $N = 23^3$ and 3 boundary padding points.

capture the other two functions F_3 and F_4 accurately. The results of $N = 23^3$ are shown in table 2.5.

2.4 Chapter summary

In this chapter we propose a multilevel univariate quasi multiquadric interpolation scheme. It is practical as it does not require derivative values of the function being interpolated. It has a higher degree of smoothness than the original one-level formula as it allows a shape parameter $c = \mathcal{O}(h)$. Our quasi interpolation costs $\mathcal{O}(n \log n)$ flops to set up. It preserves strict convexity and monotonicity. When $c = \mathcal{O}(h)$, we prove the proposed scheme converges with a rate of $\mathcal{O}(h^{2.5} \log h)$. Furthermore, if both $|f''(a)|$, and $|f''(b)|$ are relatively small compared with $\|f''\|_\infty$, the convergence rate will increase. We verify numerically that $c = h$ is a good shape parameter to use for our method, hence we need not find the optimal parameter. For all test functions, both convergence speed and error are optimized for c between $0.5h$ and $1.5h$. Our method can be generalized to a multilevel scheme; we include the numerical results for

the three-level scheme. The shape parameter of the three-level scheme can be chosen between $2h$ to $3h$.

We then propose a simple algorithm to extend the univariate formula to multi-dimensional using the dimension-splitting MQ basis. We show that the technique of boundary padding can improve accuracy, especially for the approximation of derivatives. the level-1 quasi-interpolant $\mathcal{L}_{\mathcal{D}}F$ converges to $F : \mathbb{R}^d \rightarrow \mathbb{R}$ at a speed of $\mathcal{O}(d \cdot h^{2.5} \log h)$ provided $c = \mathcal{O}(h)$. The relationship between smoothness and derivative approximation is also studied. In two dimension, our scheme can accurately approximate F, F_x, F_y and F_{xy}.

Chapter 3

On Boundary Layer Problems

3.1 The Boundary Layer Problems

We consider the singularly perturbated two-point boundary value problem (BVP)

$$\begin{aligned} &\epsilon u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \quad \forall x \in [a,\ b], \\ &u(a) = \alpha, \quad u(b) = \beta, \end{aligned} \tag{3.1}$$

where $\epsilon > 0$ denotes a fixed (small) constant. In many applications (3.1) possesses boundary layers, i.e., regions of rapid change in the solution near the endpoints with widths $o(1)$ as $\epsilon \to 0$. When we solve (3.1) with a collocation method using roughly N collocation points, then, with a very small parameter ϵ in (3.1), large N is required to obtain accurate solutions. For good resolution of the numerical solution at least one of the collocation points must lie in the boundary layer. For example, if $\epsilon \ll 1$ and the problem possesses a boundary layer of width $\mathcal{O}(\epsilon)$, then on a uniform grid with $\mathcal{O}(N^{-1})$ spacing between the points we would need $N = \mathcal{O}(\epsilon^{-1})$, which is not practical in most cases. Therefore, most numerical methods use specially designed grids that contain more points in and around the (boundary) layer. For instance, Miller et al. [84] successfully developed an upwind and central difference scheme on a piecewise uniform mesh. Gartland [45] and Vulanovic [100] suggested exponentially distributed grid points. All of these special meshes have a limitation: collocation points (or data centers) must not coincide numerically.

Integration is a smoothing operation in which the convergence rate is expected to accelerate in line with the convergence rate estimates of Madych and Nelson [78, 79]. Mai-Duy and Tran-Cong [81, 82] obtained very impressive computational results with singly or doubly integrated MQ-RBFs that they cast as a minimization of the least-squares residual errors to find the expansion coefficients of an RBF-PDE problem. The application of the volume integrated MQ-RBFs to the solution of PDEs has recently been investigated by Kansa et al. [71]. Since the derivatives of a solution to (3.1) change dramatically within a very narrow (boundary) layer, a better approximation to the solution's derivatives will result in better accuracy. Instead of using MQs to approximate $u(x)$, the *integral formulation* uses MQs to approximate the second derivative $u''(x)$. Details are given in Section 3.3.

Hon [56] suggested an adaptive technique using an *"a posteriori"* indicator based on the weak formulation of the governing equation to add collocation points where necessary. The indicator I_i is given by

$$I_i = \int_{x_{i-1}}^{x_{i+1}} [\epsilon u'' + p(x)u' + q(x)u - F]\, w_i\, dx, \tag{3.2}$$

for a BVP of the form (3.1), where u is the current numerical approximation to the solution, and

$$w_i = w_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \forall x \in (x_{i-1}, x_i), \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \forall x \in (x_i, x_{i+1}). \end{cases}$$

An extra data point is added at both mid-points of the intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ if $|I_i - I_\mu| \geq 2I_\sigma$ where I_μ and I_σ denote the mean and the standard deviation of these indicators I_i. We refer to the original article for details.

The present chapter combines the transformation technique used in [97] with the MQ integral formulation for solving problems with boundary layers. The proposed scheme does not suffer from the problem of data points coinciding, because the computational domain is on a fixed, well separated grid. Furthermore, the meshfree MQs allow a simple modification for one-sided boundary layer problems. Numerical comparisons with Tang's and Trummer's PSC methods and Hon's adaptive MQ scheme given in Section 3.4 show that our proposed scheme can achieve better accuracy for

problems with very thin layers, and is applicable to various boundary value problems. Next, we propose an error indicator in order to modify our scheme to become adaptive.

3.2 Transformations

As mentioned in Section 3.1, at least one of the collocation points should lie in the boundary layer in order to resolve the layer. Tang and Trummer [97] introduced a sequence of variable transformations that guarantee that there are some collocation points within distance ϵ from the boundaries ± 1 even for $\epsilon \ll 1$ and moderately small values of $N \approx 100$.

3.2.1 The transformed equations

We transform the singularly perturbed linear BVP (3.1) via the variable transformation $x \mapsto \xi(x)$ into the new BVP

$$\begin{aligned} &\epsilon v''(\xi) + P(\xi)v'(\xi) + Q(\xi)v(\xi) = F(\xi), \quad \forall \xi \in [a,b], \\ &v(a) = \alpha, \quad v(b) = \beta, \end{aligned} \tag{3.3}$$

where v is the transplant of u, $v(\xi) = u(x(\xi))$. Throughout the chapter we refer to x and ξ as the problem variable and the computational variable, respectively. The transformed coefficients are given by

$$\begin{aligned} P(\xi) &= \frac{p(x)}{\xi'(x)} + \epsilon \frac{\xi''(x)}{\xi'(x)^2}, \\ Q(\xi) &= \frac{g(x)}{\xi'(x)^2}, \quad F(\xi) = \frac{f(x)}{\xi'(x)^2}, \end{aligned} \tag{3.4}$$

where $x = x(\xi)$. For computational efficiency the two quantities $1/\xi'(x)$ and $\xi''(x)/[\xi'(x)]^2$ ought to be easy to calculate.

3.2.2 The iterated SINE-transform g_m

Without loss of generality we assume $[a,\ b] = [-1,\ 1]$. For $m \geq 1$ we consider the one-to-one mapping $x(\xi) = g_m(\xi)$ given by

$$g_0(\xi) = \xi,\ \ g_m(\xi) = \sin\left(\frac{\pi}{2} g_{m-1}(\xi)\right),\ \ m \geq 1. \tag{3.5}$$

It is shown in [97] that for the Chebyshev points $\xi_j = \{\cos(j\pi/N)\}_{j=0}^{N}$,

$$\begin{aligned} g_m(\xi_0) - g_m(\xi_1) &= g_m(\xi_{N-1}) - g_m(\xi_N) \\ &= \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{m+1}} (1 + \mathcal{O}(N^{-2})). \end{aligned} \tag{3.6}$$

This indicates that the transformed BVP can handle extremely thin boundary layers with a fairly small number of collocation points. For $m = 1$, 2 and 3 (corresponding to one, two and three SINE transformations), the distance between each boundary point and its nearest interior point is $\mathcal{O}(N^{-4})$, $\mathcal{O}(N^{-8})$ and $\mathcal{O}(N^{-16})$, respectively.

It is easy to show that for uniformly distributed collocation points $\xi_j = \left\{\frac{2}{N} j - 1\right\}_{j=0}^{N}$, the estimate becomes

$$g_m(\xi_0) - g_m(\xi_1) = g_m(\xi_{N-1}) - g_m(\xi_N) = \mathcal{O}(N^{-2^m}). \tag{3.7}$$

Although one may use the SINE-transformed points as collocation points for the original differential equation (3.1), the estimates (3.6) and (3.7) suggest that some of these points near the boundaries will numerically coincide after just a few transforms.

For the transformations (3.5), the quantity $1/\xi'(x)$ is given by

$$\frac{1}{\xi'(x)} = \prod_{k=0}^{m-1} \left(\frac{\pi}{2} \cos\left(\frac{\pi}{2} g_k(\xi)\right)\right),\ \ m \geq 1, \tag{3.8}$$

and $H_m = \xi''(x)/[\xi'(x)]^2$ can be computed using the recursion

$$H_0 = 0,$$

$$H_k = \frac{\pi}{2} \tan\left(\frac{\pi}{2} g_{m-k}^{-1}(x)\right) + \frac{\pi}{2} \cos\left(\frac{\pi}{2} g_{m-k}^{-1}(x)\right) H_{k-1},\ \ k = 1, \ldots, m. \tag{3.9}$$

The interested reader is referred to the original article for details. Note that the data centres in the computational variable ξ are well separated; on the other hand, in

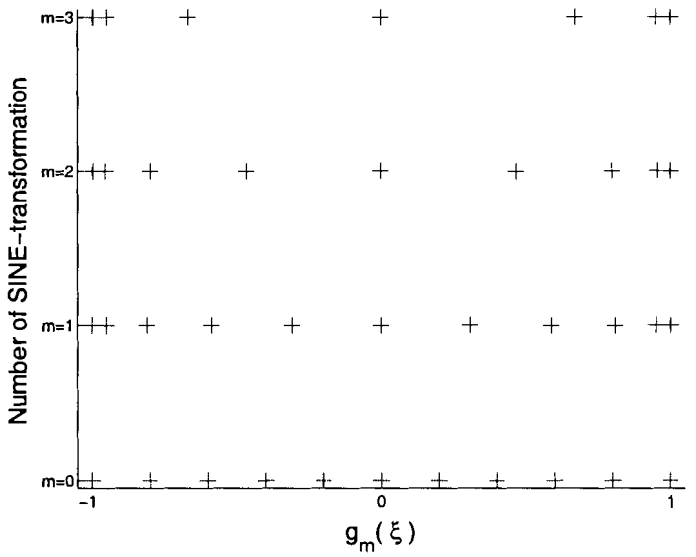

Figure 3.1: Effect of SINE-transforms on regularly spaced data centre.

the original problem variable x, data centres may coincide numerically. This does not appear to give rise to any numerical difficulty; in practice, our calculations have been successful even with "large" values of m, such as $m = 7$ (see Table 3.1 for the spacing of collocation points near the boundary). Once we pick the computational data points ξ_j, the problem variable x_j is given by (3.5). The x_j are only used when evaluating the transformation coefficients $1/\xi'(x)$ and $\xi''(x)/[\xi'(x)]^2$ using (3.8) and (3.9). The numerical coincidence of x_j is reflected in the coefficients of the transformed BVP, P and Q in (3.3) and (3.4), but not in the collocation points and data centres in the *computational* domain.

The transform defined by (3.5) maps $[-1, 1]$ onto $[-1, 1]$. Each level of the transform pushes the data centres further away from the origin towards ± 1, see figure 3.1. Note that the iterated SINE-transform g_m also maps the intervals $[-1, 0]$ and $[0, 1]$ onto themselves. Unlike the PSC method, the RBF method is "meshfree". If the singularly perturbed BVP contains only one layer on the left (or right), we shall translate the problem domain to $[-1, 0]$ (or $[0, 1]$, respectively). Unnecessary collocation points in the smooth region can be avoided without any modification to the scheme itself.

3.3 The MQ-scheme and the Int-MQ-scheme

For any given non-negative integer m, and for N distinct points

$$a = \xi_1 < \xi_2 < \cdots < \xi_N = b,$$

one can find the transformed BVP (3.3) as described in Section 3.2.

3.3.1 Asymmetric multiquadrics collocation

The basic idea of the *MQ-scheme* is to approximate the unknown function $v(\xi)$ in the transformed BVP (3.3) with Hardy's MQ basis,

$$v(\xi) = \sum_{j=1}^{N} \lambda_j \, \phi_j(\xi) + \sum_{\ell=1}^{M} \gamma_\ell \, \xi^{\ell-1}, \tag{3.10}$$

where M is some positive integer, and $\phi_j(\xi) = \sqrt{(\xi - \xi_j)^2 + c_j^2}$ is Hardy's MQ basis function.

In (3.10) there are $N+M$ undetermined coefficients λ_j and γ_ℓ. Since the MQ basis function is continuously differentiable, the derivatives of $v(\xi)$ can be approximated by differentiating the right hand side of (3.10) and using

$$\phi_j'(\xi) = (\xi - \xi_j)\,[\phi_j(\xi)]^{-1}, \quad \text{and} \quad \phi_j''(\xi) = c_j^2\,[\phi_j(\xi)]^{-3}.$$

We choose $M = 2$ so that we have enough degrees of freedom in the numerical approximation $v(\xi)$ to enforce all collocation and boundary conditions. Collocating the transformed BVP (3.3) at all N grid points provides N linear equations:

$$\begin{aligned} \sum_{i=1}^{N} \lambda_i \, [\epsilon \, \phi_i''(\xi_j) \;\; + \;\; & P(\xi_j)\,\phi_i'(\xi_j) + Q(\xi_j)\,\phi_i(\xi_j)] \\ & + \gamma_1 \, Q(\xi_j) + \gamma_2 \, [P(\xi_j) + \xi_j \, Q(\xi_j)] = F(\xi_j), \end{aligned}$$

for $1 \le j \le N$. The remaining $M = 2$ equations are obtained by substituting (3.10) into the boundary conditions in (3.3). This yields

$$\sum_{i=1}^{N} \lambda_i \phi_i(a) + \gamma_1 + \gamma_2 \, a \;\; = \;\; \alpha,$$

$$\sum_{i=1}^{N} \lambda_i \phi_i(b) + \gamma_1 + \gamma_2 b \;=\; \beta.$$

These $N+M$ equations determine the $N+M$ coefficients λ_j and γ_ℓ in (3.10), and therefore the solution $v(\xi)$ (or, equivalently, $u(x)$). The resulting matrix system is of the form

$$\begin{bmatrix} \ell_{11} & \cdots & \ell_{NN} & Q(\xi_1) & r_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \ell_{N1} & \cdots & \ell_{NN} & Q(\xi_N) & r_N \\ \phi_1(a) & \ldots & \phi_N(a) & 1 & a \\ \phi_1(b) & \ldots & \phi_N(b) & 1 & b \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} F(\xi_1) \\ \vdots \\ F(\xi_N) \\ \alpha \\ \beta \end{bmatrix},$$

where $\ell_{ij} = \epsilon\, \phi_i''(\xi_j) + P(\xi_j)\, \phi_i'(\xi_j) + Q(\xi_j)\, \phi_i(\xi_j)$, and $r_i = P(\xi_i) + \xi_i\, Q(\xi_i)$.

3.3.2 Recasting the linear system in integral formulation

The methodology of the integral formulation (*Int-MQ-scheme*) is similar, except we integrate the MQ basis functions twice to obtain the new basis functions

$$\Phi_j(\xi) = \frac{1}{6}\phi_j(\xi)(\xi - \xi_j) + \frac{c^2}{2}[(\xi - \xi_j)\ln(\phi_j(\xi) - (\xi - \xi_j)) - \phi_j(\xi)],$$

whose derivatives are

$$\begin{aligned} \Phi_j'(\xi) &= \frac{1}{2}(\xi - \xi_j)\phi_j(\xi) + \frac{c^2}{2}\ln(\phi_j(\xi) + (\xi - \xi_j)), \\ \Phi_j''(\xi) &= \phi_j(\xi). \end{aligned}$$

In other words, Hardy's MQ basis is used to approximate $v''(\xi)$ instead of $v(\xi)$. Since the basis employed here need more work to evaluate than that in the MQ-scheme, the set up cost of the Int-MQ scheme is more expansive. The expansion of the unknown function $v(\xi)$ is now of the form

$$v(\xi) = \sum_{j=1}^{N} \lambda_j\, \Phi_j(\xi) + \sum_{j=1}^{M} \gamma_\ell\, \xi^{\ell-1}. \tag{3.11}$$

We again pick $M = 2$ and apply the same collocation procedure as for the MQ-scheme. The resulting matrix system is of the form

$$\begin{bmatrix} \ell_{11} & \cdots & \ell_{NN} & Q(\xi_1) & r_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \ell_{N1} & \cdots & \ell_{NN} & Q(\xi_N) & r_N \\ \Phi_1(a) & \ldots & \Phi_N(a) & 1 & a \\ \Phi_1(b) & \ldots & \Phi_N(b) & 1 & b \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} F(\xi_1) \\ \vdots \\ F(\xi_N) \\ \alpha \\ \beta \end{bmatrix}, \tag{3.12}$$

where $\ell_{ij} = \epsilon\,\Phi_i''(\xi_j) + P(\xi_j)\,\Phi_i'(\xi_j) + Q(\xi_j)\,\Phi_i(\xi_j)$, and $r_i = P(\xi_i) + \xi_i\,Q(\xi_i)$. The linear system (3.12) determines all unknown coefficients in (3.11).

The shape parameter affects the performance of the Int-MQ-scheme. The Int-MQ-scheme obtains a better approximation to the solution by integrating twice. Considering this approximation as an interpolation problem for v'' suggests that the shape parameter should be chosen in the same fashion as in the previous section. We shall see in section 3.4 that despite relatively minor differences, a much improved numerical result can be achieved by the Int-MQ-scheme in comparison to the MQ-scheme.

3.4 Working with Int-MQ Scheme

We use uniformly distributed data centres for all examples in this section, without any intention of optimizing the distribution of data centres (or collocation points). Although higher accuracy may be expected by increasing the value of the MQ shape parameter c, or by using various values of c at different centres, a simple shape parameter is used for testing purposes. We use the shape parameter $c_j = 0.815\,d_j$ for the MQ basis (1.3), a constant shape parameter for uniformly distributed data centres. For small ϵ, solving the BVP directly on $\{x_j\}$ without using a transform appears hopeless.

Currently, we do not have a way of finding the optimal value of m *a-priori*. Estimating the number of data points inside a boundary layer of width ϵ provides, however, a fairly good guess for m. We choose m guided by the number of computational points in the interval $[-1,\, g_m^{-1}(\epsilon)]$. The inverse of the SINE-transform, $h_m = g_m^{-1}$, is given

N	16	32	64	128	256	512
$m = 1$	1.04(-03)	2.51(-04)	6.20(-05)	1.53(-05)	3.82(-06)	9.55(-07)
$m = 3$	1.17(-12)	4.02(-15)	1.47(-17)	5.59(-20)	2.15(-22)	8.33(-25)
$m = 5$	1.88(-48)	2.63(-58)	4.76(-68)	9.78(-78)	2.14(-87)	4.83(-97)

Table 3.1: Estimated distance between the boundary points and their nearest interior points for $2N - 1$ regularly spaced data centres on $[-1, 1]$ under m SINE-transforms.

by the recursion

$$h_0(\xi) = \xi, \quad h_m(\xi) = \frac{2}{\pi} \arcsin(h_{m-1}(\xi)), \ m \geq 1.$$

Table 3.1 lists the estimated distance (using (3.7)) between the boundary points and their nearest interior points for $2N - 1$ regularly spaced data centres on $[-1, 1]$ under m SINE-transforms. This allows us to estimate the number of necessary SINE-transforms; for example, we do not expect that one SINE-transform ($m = 1$) will successfully capture a boundary layer with $\epsilon = 10^{-9}$.

In all tables, $d_0.d_1d_2(e)$ represents the number $d_0.d_1d_2 \times 10^e$. All codes are written in Matlab 6.0, and are executed in double precision arithmetic ($\epsilon_{machine} = 2.22 \times 10^{-16}$) on a 4×500MHz Alpha machine with 2GB RAM. The matrix equations (3.12) are solved by the Matlab built-in *left matrix divide* function. In this section we do not employ iterative techniques or preconditioning. Readers interested in aspects of efficiently solving linear systems of equations arising from radial basis functions are referred to Beatson [7], chapter 4 of this thesis, and Mouat [85].

In our results we report the ℓ_∞-norm error. The ℓ_∞-norm error provides an accurate measure of how well the boundary layer is being captured numerically. In our examples, the ℓ_2-norm errors are, in general, $2 - 3$ orders of magnitude smaller than the ℓ_∞-norm error due to the fact that collocation points are densely distributed in the boundary layer where the errors of the numerical approximation are relatively large.

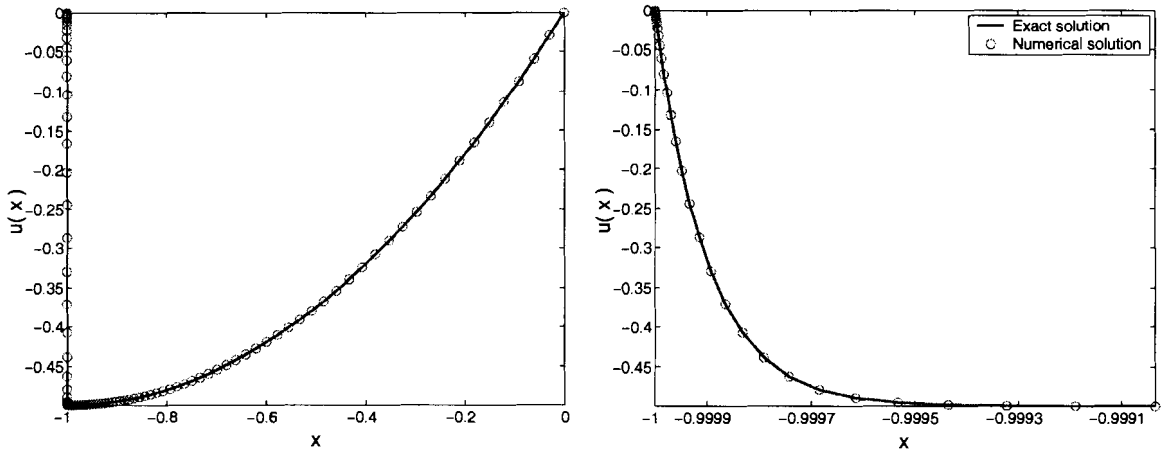

Figure 3.2: Numerical solution of (3.13) for Int-MQ-scheme with, $\epsilon = 10^{-4}$, $N = 128$, $m = 3$; —— exact solution, o o o approximation.

3.4.1 Numerical comparison of the MQ-schemes

We compare the Int-MQ and MQ methods with SINE-transformations. Previous studies by other authors suggest the integral formulation can achieve better accuracy for various RBF-PDE problems. Our results show the Int-MQ-scheme to be more accurate than the original MQ-scheme.

The same shape parameters is used for both methods and all values of N. The BVP has variable coefficients and the solution develops one boundary layer on the left. The equation, whose domain is shifted to $[-1, 0]$, is

$$\epsilon u''(x) + u'(x) = x + 1 + \epsilon, \tag{3.13}$$

where f is chosen such that the function

$$u(x) = \frac{1}{2}(e^{-\frac{x+1}{\epsilon}} + x^2 + 2x), \tag{3.14}$$

is an exact solution of the differential equation (3.13). The boundary conditions are $u(-1) = 0$, $u(0) = 0$. Note that the "exact solution" (3.14) will satisfy these boundary conditions to machine precision for all values of $\epsilon \leq 0.3$. The separation distance of the N problem variables x_j on the shifted domain is equivalent to the distance for $2N-1$ problems variables on $[-1, 1]$. See table 3.1 for the estimated distance between the boundary points and their nearest interior points.

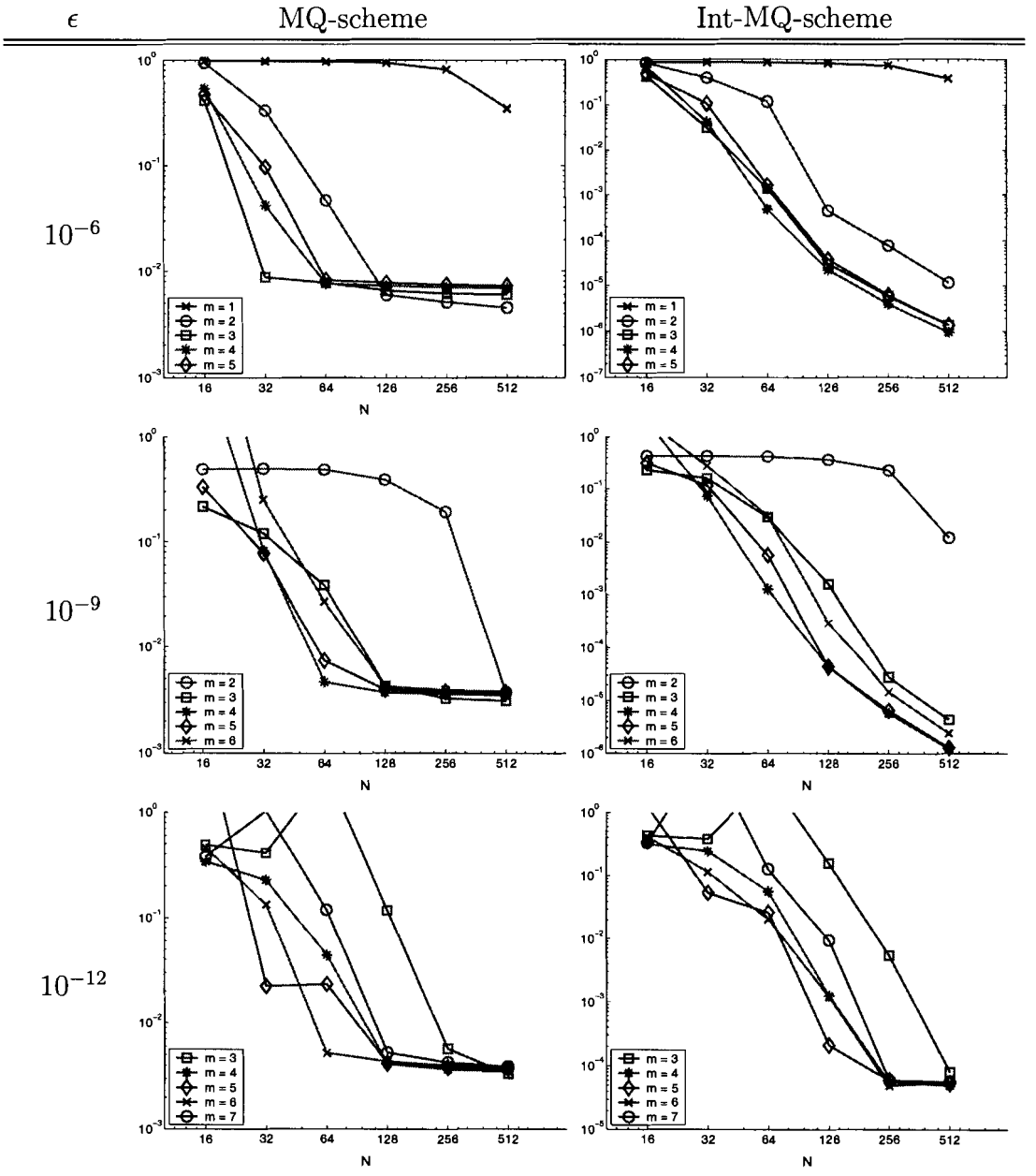

Figure 3.3: The ℓ_∞-norm errors of the MQ-scheme (left) and the Int-MQ-scheme (right) to (3.13), with different m SINE-transforms, and with $\epsilon = 10^{-6}$, $\epsilon = 10^{-9}$, and $\epsilon = 10^{-12}$, as a function of N.

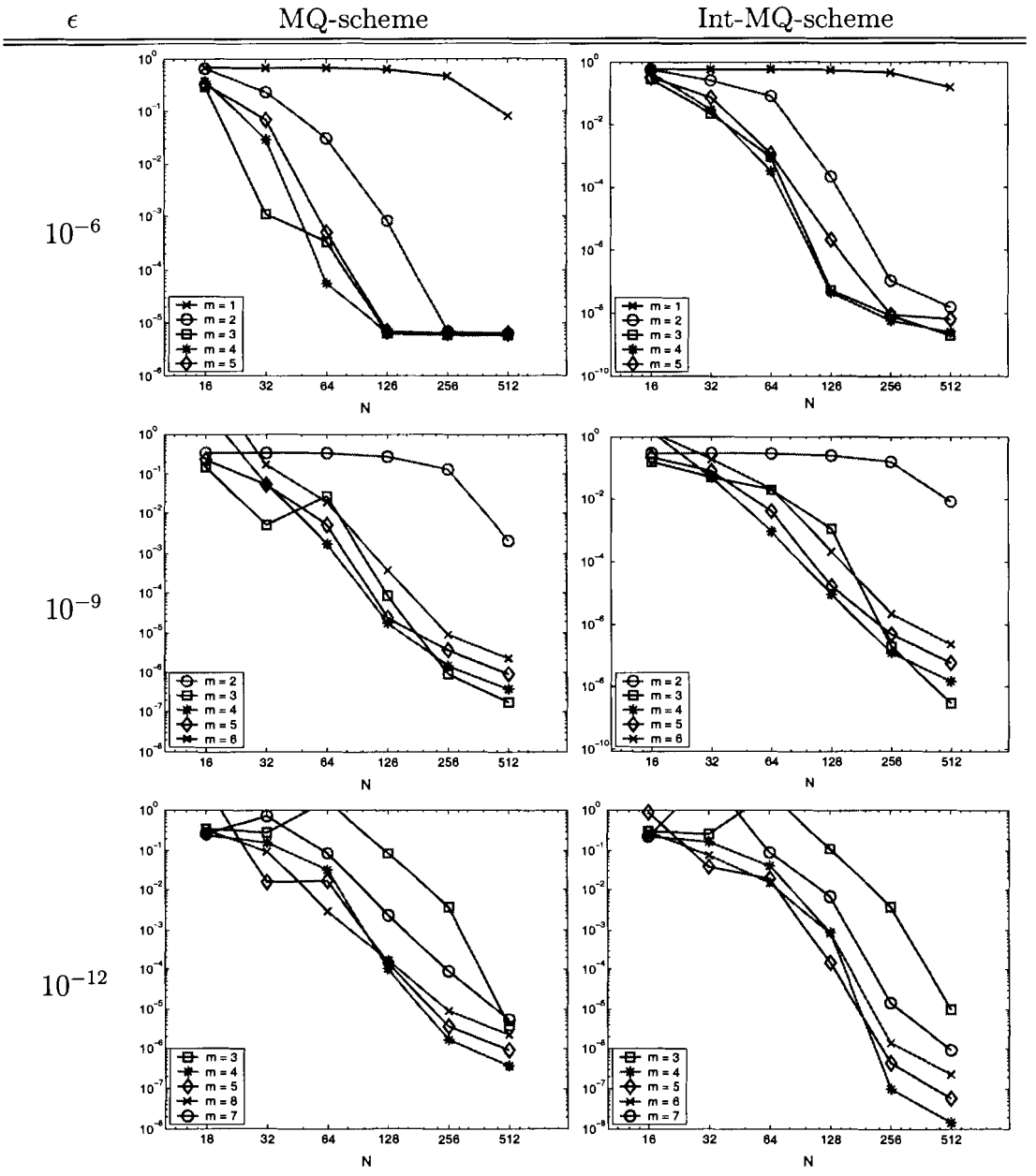

Figure 3.4: The ℓ_2-norm errors of the MQ-scheme (left) and the Int-MQ-scheme (right) to (3.13), with different m SINE-transforms, and with $\epsilon = 10^{-6}$, $\epsilon = 10^{-9}$, and $\epsilon = 10^{-12}$, as a function of N.

This is a difficult problem, since high resolution of the boundary layer is needed to capture the solution in the middle of the interval correctly. Figure 3.2 shows the solution for $\epsilon = 10^{-4}$, $N = 128$, and $m = 3$ SINE-transforms. The plot on the right is a magnification of the boundary layer. The SINE-transform moves more points into the boundary layer. We have a coarser grid spacing on the region where the solution is smooth.

We solve (3.13) with both the MQ-scheme and the Int-MQ-scheme. Figure 3.3 and figure 3.4 show the ℓ_∞-norm and the ℓ_2-norm errors for $\epsilon = 10^{-6}$, $\epsilon = 10^{-9}$, and $\epsilon = 10^{-12}$, respectively, with different values of m. The ℓ_∞-norm errors in figure 3.3 give a measure of how well our numerical solution approximate the exact solution in the boundary layer, whereas the ℓ_2-norm errors in figure 3.4 tell us about the accuracy at the interior smooth transition region.

In general, more SINE-transforms are required for smaller ϵ in order to capture the layer which gets thinner as ϵ gets smaller. We also see that the Int-MQ-scheme captures the boundary layer more accurately than the MQ-scheme. For $N = 512$, the ℓ_∞ errors of the most accurate numerical solution (among all the tested m values) of the Int-MQ-scheme are 4.7489×10^{-7}, 1.2356×10^{-6}, and 4.9541×10^{-5}, for $\epsilon = 10^{-6}$, $\epsilon = 10^{-9}$, and $\epsilon = 10^{-12}$, respectively. These errors occur when $m = 4$ for the Int-MQ-scheme. By contrast, the error of the MQ-scheme ceases to reduce and remains nearly constant at $\sim 3 \times 10^{-3}$ as N increases, for all tested ϵ.

When we "under-transform," and have too few points in the boundary layer, the error will become large due to the undersampling in the boundary layer. On the other hand, when we "over-transform", and have too many points in the boundary layer, the errors will become large due to the undersampling in the smooth region.

3.4.2 Numerical comparison with the PSC scheme

The Chebyshev spectral collocation (pseudospectral) method using the transform in [97] yields very accurate approximations for moderately small ϵ. Its error, however, increases rapidly as ϵ becomes smaller, see figure 3.5. When $N = 256$, the error of the numerical approximation can be as small as about 10^{-14}, but it is around 10^{-2}

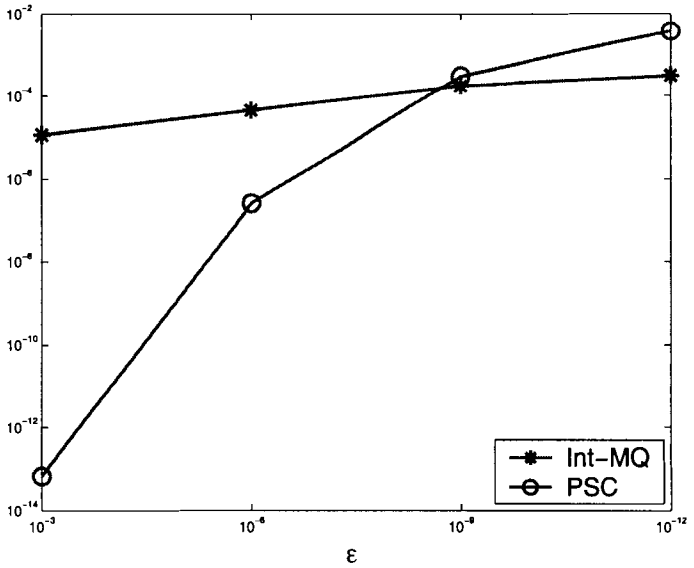

Figure 3.5: ℓ_∞-norm error, with optimal m, to the quasi-linear BVP in Example 1 of [97], for $N = 256$, as a function of ϵ.

for $\epsilon = 10^{-12}$. As seen in section 3.4.1, the accuracy of the Int-MQ-scheme is not very sensitive to the magnitude of ϵ when m is properly chosen.

Although the Int-MQ-scheme may not be always as accurate as the PSC method, it is more flexible. Since there is no restriction on how we discretize the ξ variable, an adaptive technique can be easily employed directly with our scheme. Extra points can be added to the existing grids. Doing this with the PSC is certainly not straightforward. Such flexibility plays an important role in developing an adaptive numerical scheme for time dependent BVPs whose solutions develop layers.

3.4.3 Numerical comparison with Hon's adaptive scheme

We consider the example found in [56]:

$$\begin{aligned} &\epsilon u''(x) + u'(x) = 0\ , \quad \forall x \in [0,1], \\ &u(0) = 0, \quad u(1) = 1. \end{aligned} \tag{3.15}$$

The BVP has one boundary layer on the left; we apply the Int-MQ-scheme on a shifted version of (3.15). Table 3.2 lists the results of the adaptive scheme found in

Number of Points	50	103	153	185	198	250	300
Int-MQ, $m = 3$	1.0(-1)	3.0(-2)	1.2(-4)	6.0(-5)	5.4(-5)	2.5(-5)	1.7(-5)
Int-MQ, $m = 4$	2.5(-2)	1.0(-4)	3.2(-5)	1.7(-5)	1.6(-4)	8.6(-6)	5.7(-6)
Adaptive scheme	2.3(+0)	6.8(-1)	5.4(-1)	2.2(-2)	6.0(-3)	1.4(-3)	9.2(-4)

Table 3.2: Numerical errors for (3.15) when $\epsilon = 10^{-8}$.

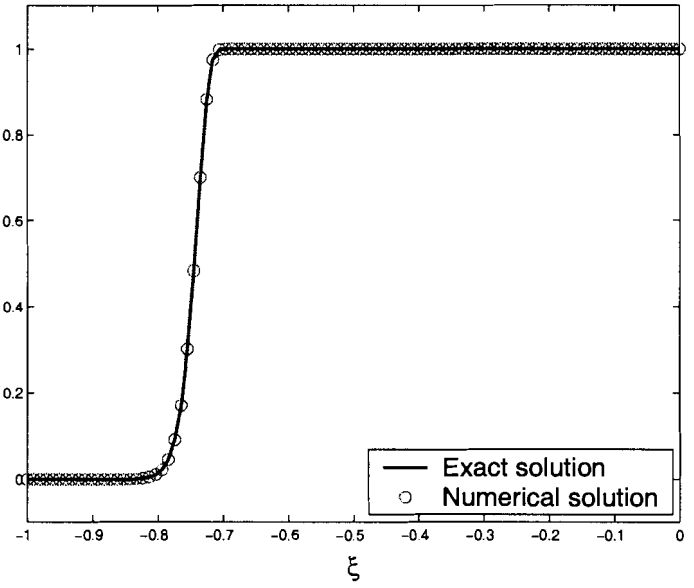

Figure 3.6: Numerical solution to (3.15) for $\epsilon = 10^{-8}$, $N = 103$ and $m = 4$ as a function of the computation variable ξ.

[56], and our Int-MQ-scheme results for $m = 3$, and $m = 4$. Our method shows better accuracy for all N. More importantly, the numerical solution of the Int-MQ-scheme with $m = 4$ and $N = 103$ is more accurate than that of the adaptive scheme with all tested N, which go as high as $N = 300$. Furthermore, our computational cost is considerably less than for the adaptive scheme in [56], which requires solving a linear system in each iteration.

Figure 3.6 shows the plot of the solution for $\epsilon = 10^{-8}$, $N = 103$ and $m = 4$ using the Int-MQ-scheme. Note that the plot is against the computational variable ξ. The boundary layer is stretched to $-1 \leq \xi \leq -0.7$. From figure 3.7, we see that the maximum error occurs in the boundary layer, around $\xi = -0.72$.

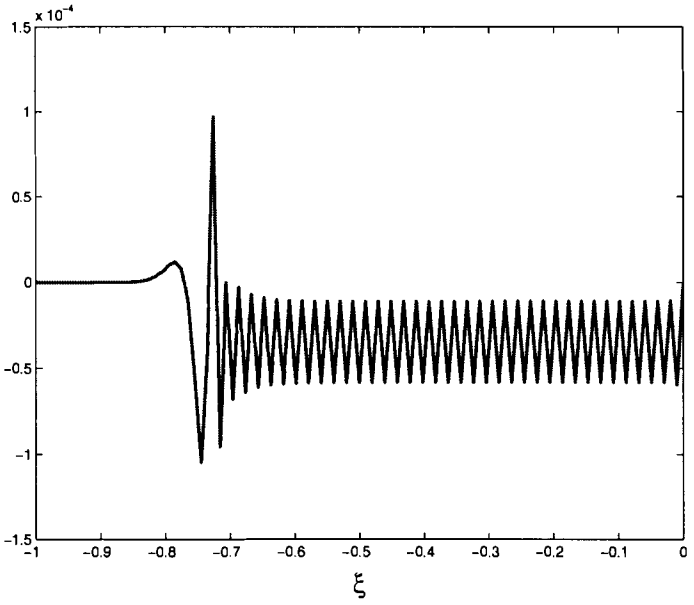

Figure 3.7: Error function corresponding to figure 3.6.

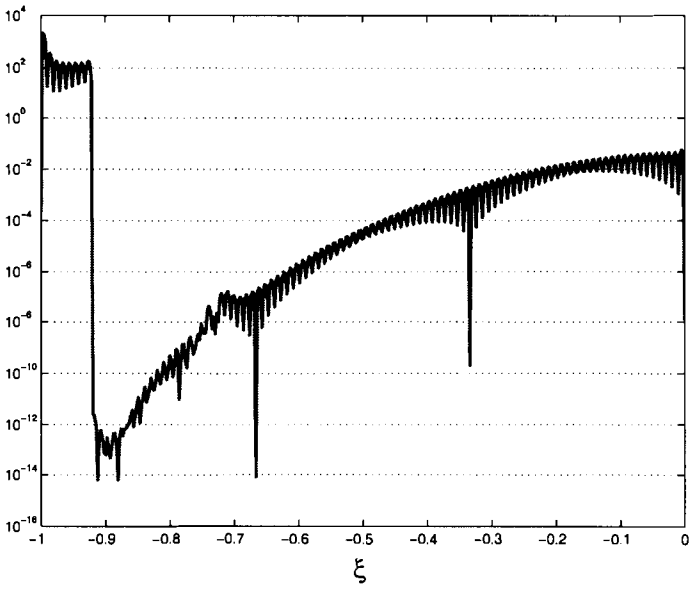

Figure 3.8: Weak error corresponding to figure 3.6.

Since the Int-MQ basis functions are also C^∞, the indicator (3.2) would apply. The weak error of the transformed BVP given by

$$\epsilon v'' + Pv' + Qv - F,$$

is straightforward to compute, see figure 3.8. The weak error is small on the interval $[-0.9, 0]$. Comparing figure 3.7 and figure 3.8, we see that (3.2) does not appear to be a good indicator for the Int-MQ-scheme: the region of large *weak error*, $-1 \leq \xi \leq -0.9$, indicated by (3.2) does not match with the region of large *error*, $-0.8 \leq \xi \leq -0.7$. We are currently developing an indicator that is suitable for the Int-MQ-scheme.

3.4.4 Discussion

Numerical results show that the integral multiquadric (Int-MQ) formulation with transformations is a very promising method for (boundary) layer problems. Even our non-adaptive method shows higher accuracy than a recently proposed adaptive multiquadric scheme. Our method uses transformations to allow a large number of collocation points in the thin boundary layer.

It is well known that the order of approximation of a smooth function is reduced by the order of differentiation. Therefore, we achieve a better numerical approximation to the solution by twice integrating the MQ basis functions. Our integral formulation appears to be superior to other RBF methods.

Although we only experimented with boundary layer problems, our scheme is capable of solving interior layer problems as well. Our method can solve this internal layer problem when combined with the overlapping domain decomposition method (DDM); see [12, 28, 70, 107, 111] for recent developments. Without loss of generality, consider an internal layer problem defined on $[-1, 1]$. Locating the layer(s) exactly is usually the tough challenge. Suppose the position of the interior layer has been located approximately by some indicator. For simplicity, suppose there is one internal layer and it is located around $x = L$. We decompose the computational domain $\Omega = [-1, 1]$ into $\Omega_1 = [-1, L + w]$ and $\Omega_2 = [L - w, 1]$, where w is the width of the overlapping region chosen wide enough to cover the layer. The BVPs defined on Ω_1 and Ω_2 are then shifted and solved by our method.

The flexibility of radial basis functions allows our method to become adaptive much easier than the spectral collocation method. The authors are continuing to study different indicators for the Int-MQ-scheme. A suitable indicator should add or redistribute data centres where necessary, but should also automatically apply transforms if and when needed.

3.5 The Adaptive scheme

In this section, we propose a new indicator that appears to work well with our scheme.

We will use our indicator to add points in the regions of the domain where the indicator is "large"; this region should correspond to the region where the (unknown) error in our solution is also large. To design such an indicator, we look at two of the main sources of error. Since the solution of the transformed BVP (3.3) still changes rapidly in the region corresponding to the boundary layer of the original BVP (3.1), the error is usually large and more data points are needed to capture the solution, see figure 3.9. On the other hand, when the number of collocation points used in the approximation (1.5) is small on some region, it is common to observe oscillations, and therefore a large error, see figure 3.10. In both cases, we expect a close relation between the error and the second derivative values: rapid changes and oscillations result in a large value of the second derivative. Since all the derivatives of the MQ basis are globally defined, we could define an indicator using the second derivative value evaluated at the midpoint of two nearby data points:

$$I_i^{(1)} = \frac{d^2v}{d\xi^2}\left(\frac{\xi_i + \xi_{i-1}}{2}\right), \quad i = 1, \ldots, N. \tag{3.16}$$

One obvious problem with this indicator is that the sum of $I_i^{(1)}$ will grow without bound as N increases. A bounded version of (3.16) should mimic the integral of the third derivative of the numerical solution:

$$I_i^{(2)} = (\xi_i - \xi_{i-1}) \cdot \frac{d^3v}{d\xi^3}\left(\frac{\xi_i + \xi_{i-1}}{2}\right), \quad i = 1, \ldots, N. \tag{3.17}$$

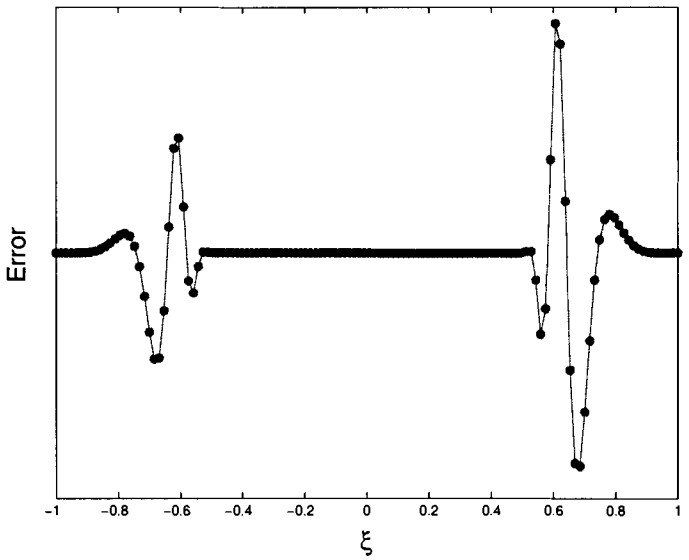

Figure 3.9: Since the solution changes rapidly in the boundary layer, the error is usually large in comparison to the error on the smooth region.

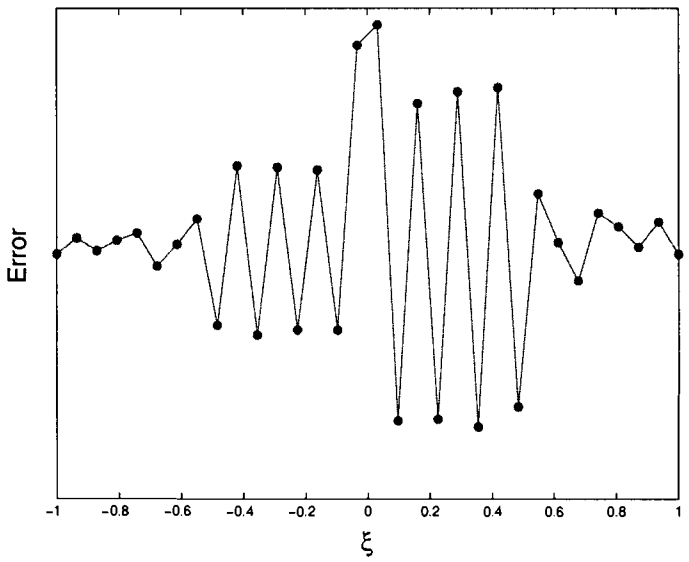

Figure 3.10: When collocation points are not dense enough, the numerical solution oscillates in the smooth region.

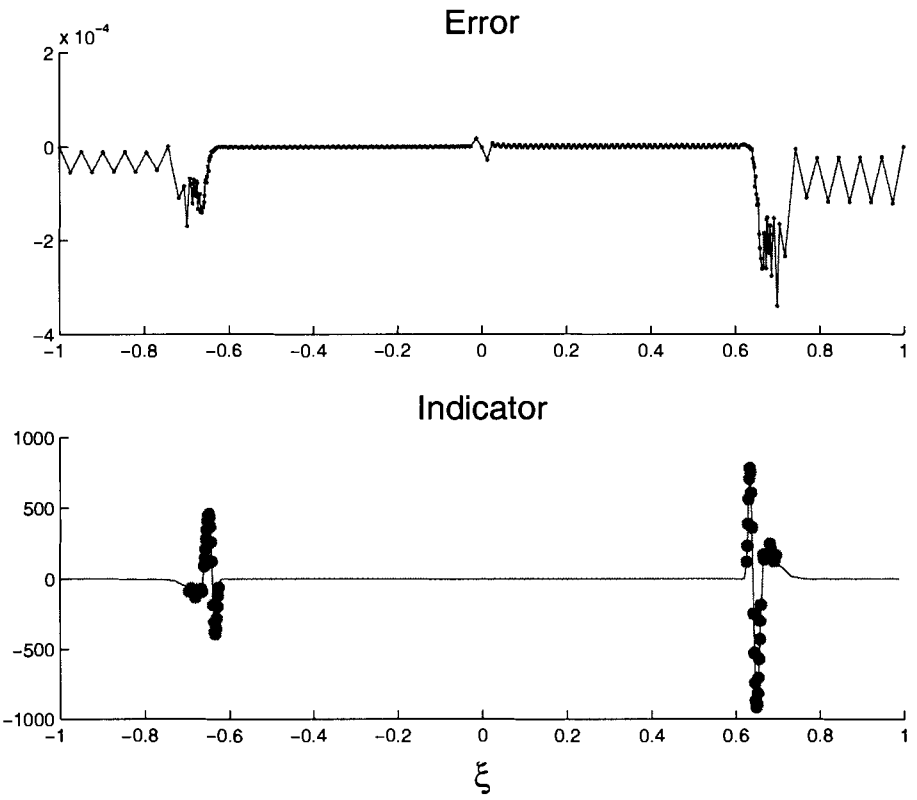

Figure 3.11: The error function and the indictor function of an adaptive scheme using $I^{(2)}$ after a few iterations. The big bullets shown in the plots of the indicator are points that will be added in the next iteration.

The second indicator works well when one starts with a sufficiently large number of data points, and lets the adaptive scheme run only for a small number of iterations. Proceeding with the adaptive iteration for a longer time using $I^{(2)}$ will lead to a situation as shown in figure 3.11: all new points will eventually be added near the boundary layer. From the plot of the error, we see that past iterations added most of the points to the boundary layer around $\xi = \pm 0.7$, and the region in between. However, only very few data points are located near the boundary of the interval $[-1, 1]$. The undersampling near the boundary causes the numerical solution to oscillate, and therefore adversely affects the accuracy of the scheme. We fix this problem with a new indicator.

On the other hand, the second indicator $I^{(2)}$ is successful in adding points into the region with the largest error, namely the boundary layer. A robust adaptive scheme, however, should also have some control over the ratio between the maximum and minimum separating grid spacing. Hence, we modify (3.17) to obtain our new indicator

$$J_i = (\xi_i - \xi_{i-1})^{\rho(k)} \cdot \frac{d^3v}{d\xi^3}\left(\frac{\xi_i + \xi_{i-1}}{2}\right), \quad i = 1, \ldots, N, \tag{3.18}$$

where k is the iteration counter for our adaptive scheme, and $\rho(\cdot)$ is any non-decreasing function.

The indicator $I^{(2)}$ is a special case of (3.18) with $\rho(k) \equiv 1$. For any constant $\rho(k) > 1$, a new balance or equilibrium between grid spacing and solution derivative will be achieved. Note that the actual values of each J_i is unimportant to the scheme. It is the relative magnitude of J_i that controls the addition of new data points. The interesting case is when $\rho(k)$ is an increasing function in k. The indicator J_i behaves similarly to $I_i^{(2)}$ when k is small. However, the problem shown in figure 3.11 will be corrected as k gets large. Suppose $\xi_i - \xi_{i-1}$ is large in comparison to the other separation distance, the term $(\xi_i - \xi_{i-1})^{\rho(k)}$ assigns a large values to J_i and the adaptive scheme will add an extra point to the midpoint of ξ_{i-1} and ξ_i. The adaptive scheme eventually captures the large grid spacing. Although a boundary layer problem usually has very large (third) derivative values near the layer, these values are considered to be fixed and finite. In the limit of $k \to \infty$, the adaptive scheme will always end up computing the solution on an equally spaced set of data points in exact arithmetic. In general, a rapidly growing $\rho(k)$ will favor equally distributed grids; whereas a slowly increasing $\rho(k)$ will tend to add more points to the boundary layer. For simplicity, all our computations in the next section use J_i with $\rho(k) = k$.

	Error: Non-adaptive			*Error: Adaptive*		
m	N	ℓ_∞-*norm*	ℓ_2-*norm*	N	ℓ_∞-*norm*	ℓ_2-*norm*
4	512	$8.14e-04$	$8.85e-07$	407	$2.00e-04$	$1.15e-05$
5	512	$6.09e-04$	$3.05e-05$	305	$1.54e-04$	$7.62e-06$
6	512	$1.56e-03$	$9.30e-05$	257	$2.24e-04$	$2.19e-06$
7	512	$2.30e-03$	$6.30e-04$	430	$1.84e-04$	$8.12e-05$

Table 3.3: Errors for a boundary layer problem with $\epsilon = 10^{-12}$. Comparison of the non-adaptive method with $N = 512$ and adaptive method.

3.5.1 Numerical results

Example 3.5.1 (Boundary layer problem) *We first demonstrate the robustness of our adaptive scheme with the example found in [97]. The BVP with variable coefficients is given by*

$$\epsilon u''(x) - xu'(x) - u(x) = f(x), \quad x \in [-1, 1],$$

where

$$f(x) = \left(\frac{x+1}{\epsilon} - 1\right) e^{-(x+1)/\epsilon} + 2\left(\frac{x-1}{\epsilon} + 1\right) e^{(x-1)/\epsilon}$$

is chosen such that the function

$$u(x) = e^{-(x+1)/\epsilon} + 2\,e^{(x-1)/\epsilon}$$

is an exact solution of the BVP. Boundary conditions are given by the values of the exact solution. This is not an easy test problem; besides exhibiting two boundary layers, it also features a turning point at $x = 0$.

We use the indicator J with $\rho(k) = k$ to solve the BVP with $\epsilon = 10^{-12}$. The number m always refers to the number of SINE-transforms applied to obtain (3.3). The MQ shape parameter is set to $c = 0.815h_i$, where h_i is the minimum distance between the center ξ_i and its nearby centers. An extra point $(\xi_i + \xi_{i+1})/2$ will be added to the set of data points if $|I_i - I_\mu| \geq \theta\, I_\sigma$, where I_μ and I_σ denote the mean and the standard deviation of these indicators I_i. In our tests we use $\theta = 0.5$. The adaptive iteration starts with $N = 40$ data points and stops when the ℓ_2-norm difference of two consecutive solution is less than δ_a with $\delta_a = 10^{-4}$.

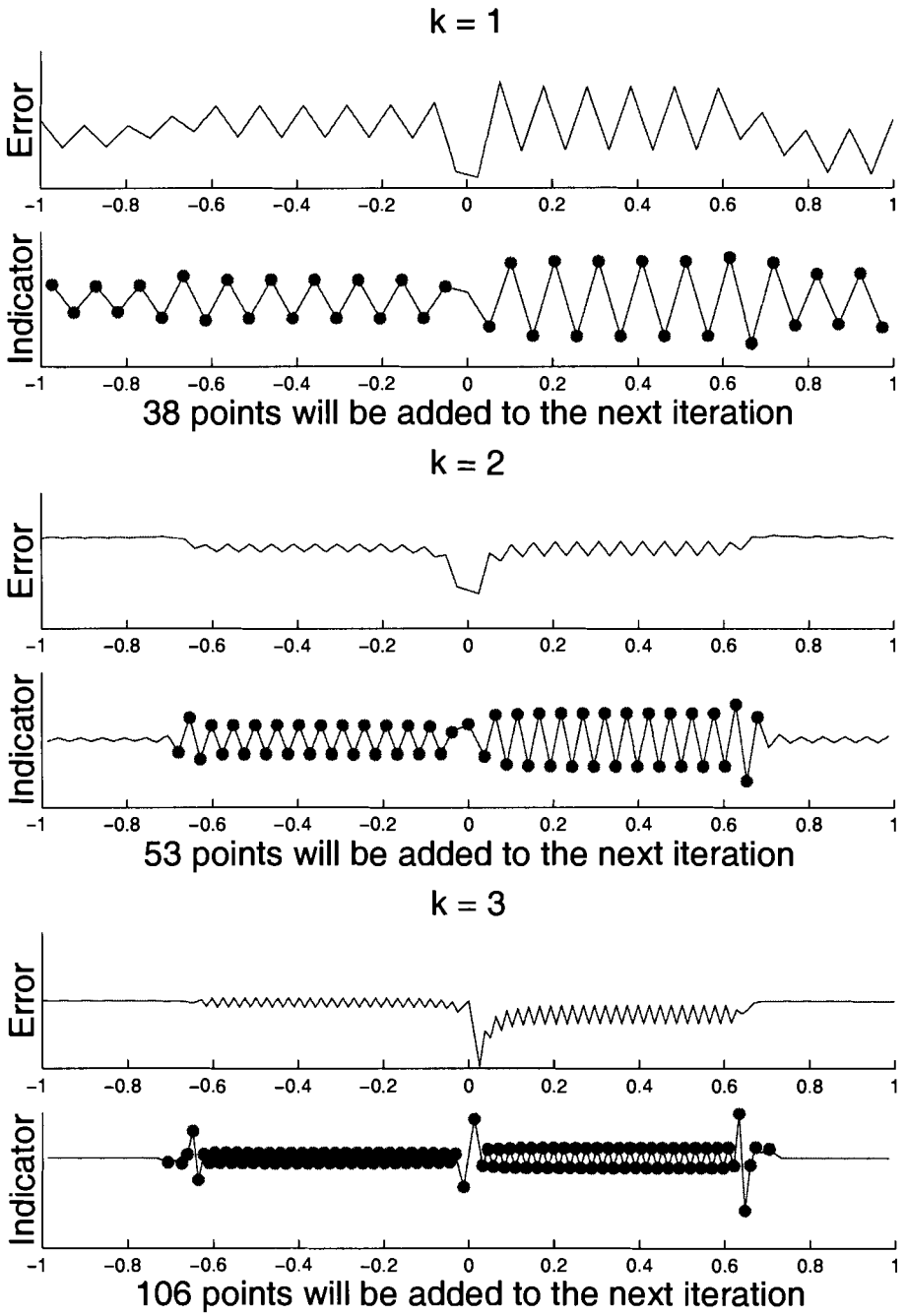

Figure 3.12: Errors and indicators J after $1-3$ adaptive iterations with $\rho(k) = k$, $\epsilon = 10^{-12}$ and $m = 5$.

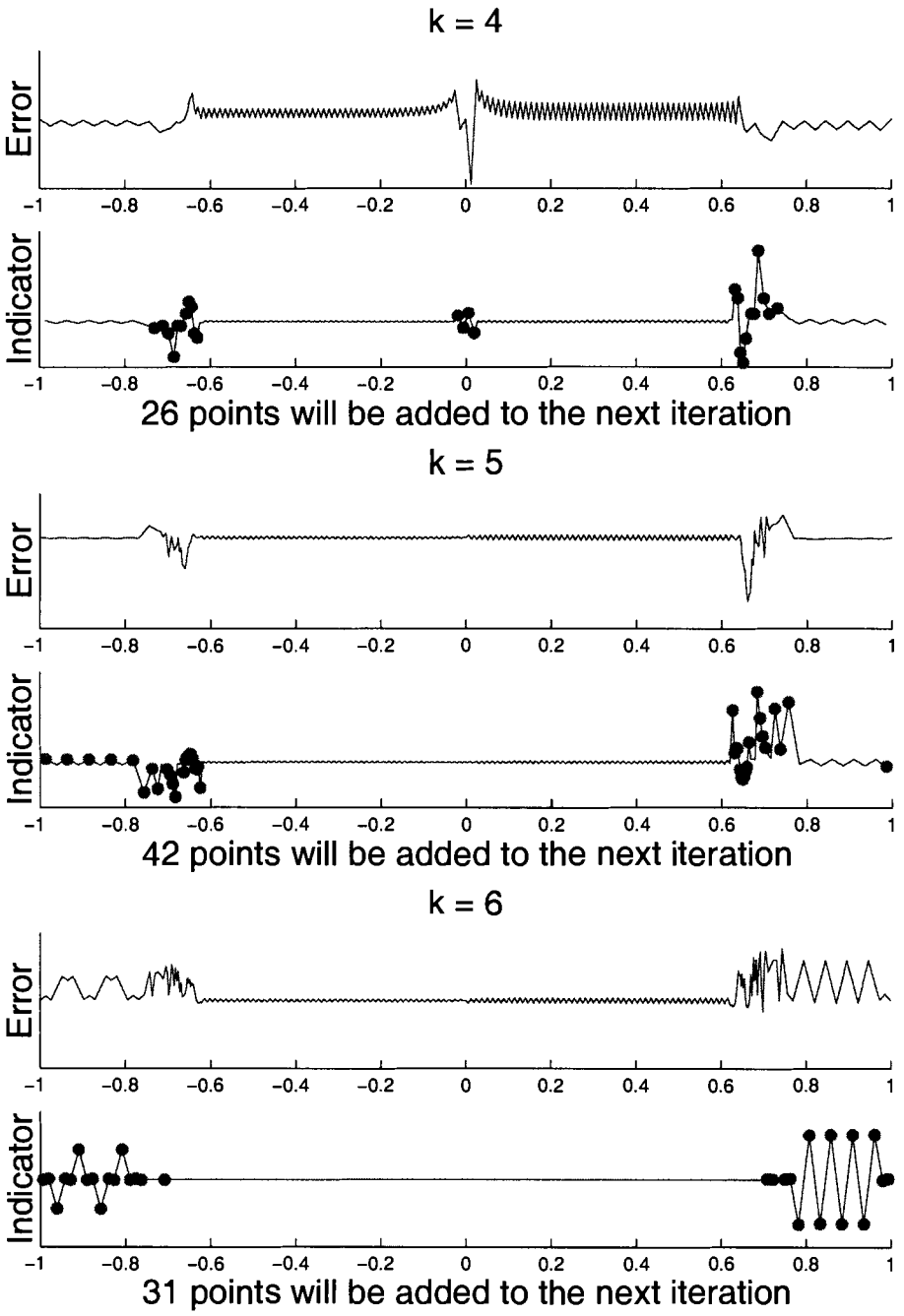

Figure 3.13: Errors and indicators J after $4-6$ adaptive iterations with $\rho(k) = k$, $\epsilon = 10^{-12}$ and $m = 5$.

In figures 3.12 and 3.13 the exact error function and the indicator for all adaptive iterations with $m = 5$ are provided. The big bullets (•) shown in the plots of the indicator are points what will be added in the next adaptive iteration. Initially, at $k = 1$, the number of data points, $N = 40$, is too small for the boundary layer problem with such a thin layer. Our scheme refines the set of data points throughout the whole computational domain. For $k = 2$ and $k = 3$, extra points are added only to the interior. At the fourth adaptive iteration, extra data points are added in the boundary layers, because of the large value of the third derivative, and near the origin, where relatively wide spacing of the data centres is causing an oscillation. The different behaviour of $I^{(2)}$ and J is obvious at the last two adaptive steps. Comparing to figure 3.11, J will add points near the boundaries where the error is large. In all cases, our indicator correctly indicates the regions of large error. Details are shown in table ??.

Example 3.5.2 (Exponentially ill-conditioning problem) *Our next example consider the equation,*

$$\epsilon u''(x) - xu'(x) + u(x) = 0, \quad x \in [-1, 1], \tag{3.19}$$

with

$$u(-1) = 1, \qquad u(1) = 2,$$

whose exact solution is

$$u(x) = \frac{1}{2}x + \frac{3}{2}\frac{\sqrt{-\frac{x^2\pi}{2\epsilon}}erf\left(-\frac{x^2}{2\epsilon}\right) + e^{(x^2/2\epsilon)}}{\sqrt{-\frac{\pi}{2\epsilon}}erf\left(-\frac{1}{2\epsilon}\right) + e^{(1/2\epsilon)}}.$$

This problem contains an eigenvalue of the order $e^{-1/2\epsilon}$, and is exponentially ill conditioned. An analysis of this problem can be found in [50]. In [74], the problem with $\epsilon = 1/70$ is solved with an adaptive methods; the obtained final error is 2.2×10^{-2} under the ℓ_2-norm. We implement an extra stopping criteria, as safeguard, to terminate the iteration and discard the newest numerical solution if divergence is detected.

We solve the ill-conditioned problem (3.19) with our adaptive scheme for $\epsilon = 1/70$, $\epsilon = 10^{-4}$, and $\epsilon = 10^{-8}$. The scheme begins with 11 equally spaced points, and $\theta = 0.5$.

$m = 4$ / k	N	*Error in* ℓ_∞*-norm*	*Error in* ℓ_2*-norm*	$\|v_k - v_{k-1}\|_2$
1	40	$1.03e+00$	$6.07e-01$	$--$
2	78	$1.68e+00$	$1.18e+00$	$1.01e+00$
3	154	$1.40e-01$	$7.81e-02$	$1.37e+00$
4	287	$3.47e-03$	$1.57e-03$	$8.20e-02$
5	366	$2.50e-04$	$1.44e-05$	$1.34e-03$
6	407	$2.00e-04$	$1.15e-05$	$8.09e-06$
$m = 5$ / k	N	*Error in* ℓ_∞*-norm*	*Error in* ℓ_2*-norm*	$\|v_k - v_{k-1}\|_2$
1	40	$8.43e-01$	$8.70e-01$	$--$
2	78	$1.22e+00$	$1.04e+00$	$6.98e-01$
3	131	$4.88e-02$	$2.62e-02$	$1.20e+00$
4	237	$9.42e-04$	$4.66e-04$	$2.66e-02$
5	263	$2.82e-04$	$1.12e-05$	$4.11e-04$
6	305	$1.54e-04$	$7.62e-06$	$3.63e-06$
$m = 6$ / k	N	*Error in* ℓ_∞*-norm*	*Error in* ℓ_2*-norm*	$\|v_k - v_{k-1}\|_2$
1	40	$1.22e+00$	$1.38e+00$	$--$
2	78	$6.60e+00$	$6.88e+00$	$6.88e+00$
3	115	$2.51e-01$	$1.56e-01$	$8.06e+00$
4	185	$2.78e-02$	$8.32e-03$	$1.58e-01$
5	206	$4.38e-04$	$3.04e-05$	$7.44e-03$
6	257	$2.24e-04$	$2.19e-06$	$1.32e-05$
$m = 7$ / k	N	*Error in* ℓ_∞*-norm*	*Error in* ℓ_2*-norm*	$\|v_k - v_{k-1}\|_2$
1	40	$2.86e+01$	$3.54e+01$	$--$
2	78	$6.78e+00$	$7.55e+00$	$3.00e+01$
3	105	$6.84e-02$	$1.23e-02$	$8.97e+00$
4	161	$5.40e-01$	$3.82e-01$	$3.67e-01$
5	202	$9.75e-02$	$6.68e-02$	$4.03e-01$
6	252	$7.38e-03$	$6.22e-04$	$6.61e-02$
7	302	$1.69e-04$	$9.11e-05$	$5.81e-04$
8	430	$1.84e-04$	$8.12e-05$	$7.69e-05$

Table 3.4: Detail data of the adaptive iteration for $\epsilon = 10^{-12}$ *with different* m.

ϵ	1/70	1/70	1/70	10^{-4}	10^{-4}	10^{-8}	10^{-8}
m	*0*	*1*	*2*	*2*	*3*	*3*	*4*
#iterations	*8*	*4*	*4*	*7*	*6*	*7*	*7*
N	*71*	*33*	*39*	*179*	*95*	*313*	*155*
Final error	*3.9(-3)*	*7.8(-3)*	*4.6(-3)*	*2.7(-4)*	*7.5(-4)*	*6.6(-3)*	*6.1(-2)*

Table 3.5: Performance of adaptive scheme on an exponential ill-conditioning problem.

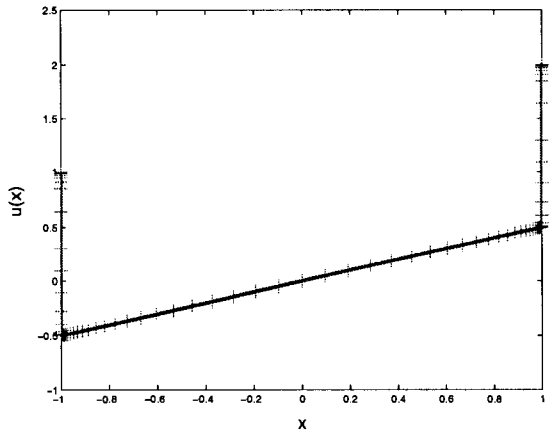

Figure 3.14: Numerical solution (+) and exact solution (−) of (3.19) with $m = 3$ and $\epsilon = 10^{-4}$ in the problem's variable.

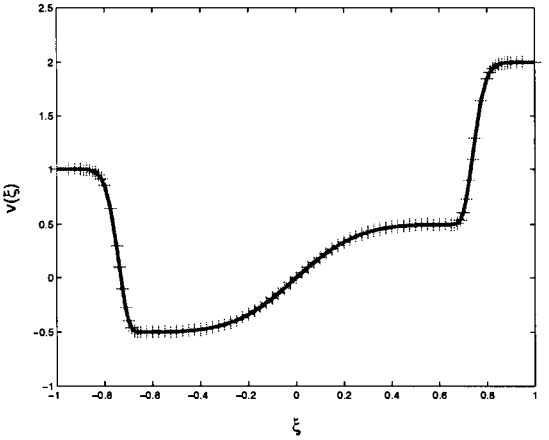

Figure 3.15: Numerical solution (+) and exact solution (−) of (3.19) with $m = 3$ and $\epsilon = 10^{-4}$ in the transformed variable.

The finial error is in the ℓ_2-norm. Numerical solution of $\epsilon = 10^{-4}$ and $m = 3$ in the problem's variable x, and transformed variable ξ are shown in figures 3.14 and 3.15. Figure 3.15 reveals the difficulty coming from the interior transition region. We summarize the results in table 3.5. Our adaptive scheme shows clear advantage over the method in [74] on this ill-conditioning problem in terms of accuracy and the ability to solve problems with tiny ϵ. On the other hand, we emphasize that the method found in [74] could apply to a broader range of problems because of their domain decomposition feature; i.e., adaptive method in [74] can capture highly oscillatory region actually.

Example 3.5.3 (Interior layer problem) *Although the SINE-transforms are designed to work specifically on solving boundary layer problems, our adaptive scheme could also be used to solve interior layer problems by giving up the transform. We*

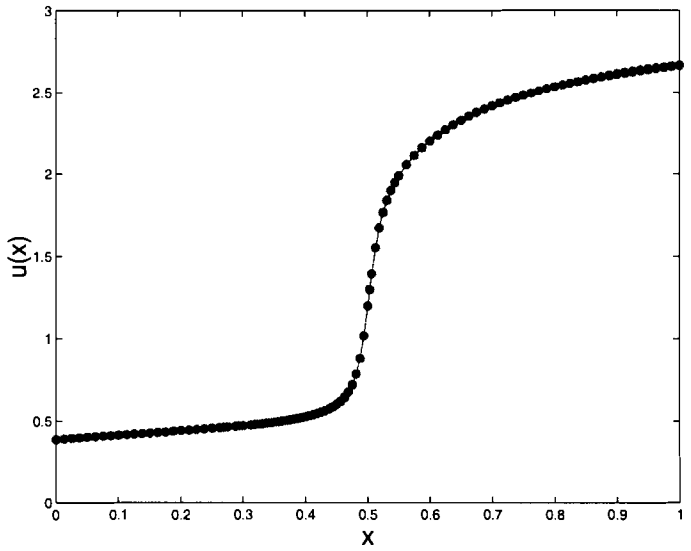

Figure 3.16: 103 *refined data points on the solution of an internal layer problem.*

solve the example in [56] where the coefficients functions of (3.1) are given by

$$p(x) = 4(x - 0.5)(1 + 0.3121(x - 1.5)),$$

and

$$q(x) = -1 - 0.2764(x - 0.5).$$

The exact solution is chosen to be

$$u(x) = \frac{\left[\sqrt{0.291(x-0.5)^2+\epsilon} + (x - 0.5)\right]}{(0.291(x-0.5)^2+\epsilon)^{0.375}} + e^{-x^2/2},$$

and the right hand side function $f(x)$ in (3.1) can be obtained from the exact solution. Without the help of coordinate stretching, our scheme is not capable of dealing with extremely small boundary layers; here, we solve this internal layer problem with $\epsilon = 10^{-4}$. No SINE-transform, $m = 0$, is applied. We start our adaptive scheme with 21 *data points and iteratively refine it to* 103 *data points after* 6 *iterations for the numerical solutions. The corresponding errors in the ℓ_∞-norm and ℓ_2-norm are* 5.84×10^{-5} *and* 2.85×10^{-5}, *respectively.*

3.6 Chapter summary

Singularly perturbed boundary value problems often have solutions with very thin layers in which the solution changes rapidly. This chapter concentrates on the case where these layers occur near the boundary, although our method can be applied to problems with interior layers. One technique to deal with the increased resolution requirements in these layers is the use of domain transformations. A coordinate stretching based transform allows one to move collocation points into the layer, a requirement to resolve the layer accurately. Previously such transformations have been studied in the context of finite difference and spectral collocation methods. In this paper we use radial basis functions (RBF) to solve the boundary value problem. Specifically, we present a collocation method based on multiquadric (MQ) functions with an integral formulation combined with a coordinate transformation. We find that our scheme is ultimately more accurate than a recently proposed adaptive MQ scheme. The RBF scheme is also amenable to adaptivity.

We then developed an adaptive scheme based on a multiquadric collocation method with a variable transform and integral formulation to solve boundary layer problems. Taking advantage of the truly meshfree nature of RBF methods we easily add extra data points where necessary. The differentiability of the MQ basis allows us to design a suitable indicator using the derivative values of the numerical solution. The adaptive scheme makes the tuning of the parameter m (which controls the variable transformation) unnecessary, and therefore increases the robustness of our solution scheme.

Chapter 4

On Preconditioners and DDMs

4.1 Preconditioner for RBFs Collocation Methods

A major class of preconditioners for the RBF interpolation problem is the null space method, see [85, 95]. Such a method relies on the fact that most popular RBFs are strictly conditionally positive definite of order m (SCPDm), see definition 1.1.2. The resulting preconditioners are of size $(N-m)$-by-$(N-m)$. The preconditioned systems are positive definite and are commonly solved by the conjugate gradient method. Bozzini, Lenarduzzi, and Schaback [15] use increasing annihilation orders of B-splines to improve the decay rates thereby improving the convergence rates of multiquadrics and polyharmonic splines. However, this class of preconditioner cannot be extended to the application of PDEs since the resulting RBF-PDE coefficient matrix does not enjoy the SCPD property.

Another important preconditioning technique is the approximate cardinal basis function (ACBF) approach. In this approach, the preconditioner has the same dimension as the original matrix. At a point x_i, an ACBF is merely a linear combination of the neighboring RBFs. Beatson et al. [7] lower the computational cost of solving the RBF interpolation problem to $\mathcal{O}(N \log N)$ operations. Faul improved the efficiency by combining the above ideas with Jacobi iteration and avoiding the unnecessary evaluation of residuals, see [31, 33]. Interested readers can find more studies of cardinal

functions of MQ by Buhmann and Micchelli [19], and Baxter [5].

Consider a PDE of the form (1.4),

$$\begin{aligned} \mathcal{L}u &= f(x) \quad \text{in } \Omega \subset \mathbb{R}^d, \\ \mathcal{B}u &= g(x) \quad \text{on } \partial\Omega, \end{aligned}$$

The unknown PDE solution u is approximated by RBFs as in (1.5) with $p(\cdot) \equiv 0$. To solve for the N unknown coefficients $\alpha = [\alpha_1, \ldots, \alpha_N]^T$, N linearly independent equations are needed. These equations can be obtained by choosing N distinct collocation points $X = \{x_1, \ldots, x_N\}$ on both $\Omega \setminus \partial\Omega$ and $\partial\Omega$. We assume the points (centers) are arranged in such a way that the first N_I points and the last N_B points are in $\Omega \setminus \partial\Omega$ and $\partial\Omega$, respectively. The centers x_k used in (1.5) are often chosen as collocation points. Other strategies of choosing X were studied by Fornberg et al. [39]. Ensuring that $U(x)$ satisfies (1.4) at the collocation points results in a good approximation of the solution u. The equations are

$$\begin{aligned} \sum_{k=1}^{N} \alpha_k \mathcal{L}\phi(x_i - x_k) &= f(x_i) \quad \text{for } x_i \subset X \cap (\Omega \setminus \partial\Omega), \\ \sum_{k=1}^{N} \alpha_k \mathcal{B}\phi(x_i - x_k) &= g(x_i) \quad \text{for } x_i \subset X \cap \partial\Omega. \end{aligned} \tag{4.1}$$

Rewriting (4.1) in matrix form, we have

$$A\alpha = b, \quad \text{where } A = \begin{bmatrix} A_{\mathcal{L}} \\ A_{\mathcal{B}} \end{bmatrix}, \; b = \begin{bmatrix} f(x_i) \\ g(x_i) \end{bmatrix}, \tag{4.2}$$

and

$$\begin{aligned} (A_{\mathcal{L}})_{ik} &= \mathcal{L}\phi(x_i - x_k), \quad x_i \subset (\Omega \setminus \partial\Omega), \quad x_k \in X, \\ (A_{\mathcal{B}})_{ik} &= \mathcal{B}\phi(x_i - x_k), \quad x_i \subset \partial\Omega, \qquad\quad x_k \in X. \end{aligned} \tag{4.3}$$

This method is often named the asymmetric collocation method. The symmetric collocation approach can be found in [34]. The matrix given by (4.2) and (4.3) is generally non-symmetric and full; this system of equations is known to be very ill-conditioned when N or c becomes large.

In this chapter we present a new preconditioner designed to work on the asymmetric collocation method for various RBF-PDE applications. This preconditioner is

sufficiently general that it may be applied to elliptic, hyperbolic or parabolic PDEs in which RBFs are used as the spatial approximation scheme. Furthermore, the preconditioning technique is coupled with the domain decomposition method.

4.2 The approximate cardinal basis function preconditioner

Given a matrix system $A\alpha = b$ as in (4.2), we construct a left-hand preconditioner W, then we solve the equivalent system

$$WA\alpha = Wb, \tag{4.4}$$

for the undetermined coefficients α.

In this section, we introduce a preconditioner based on the least-squares construction of the approximate cardinal basis functions (ACBFs). An ideal ACBF is equivalent to a delta-function, $\delta(x_i)$ that is one at it center, x_i, and zero everywhere else. In such a case, the matrix, WA, in (4.4) becomes the identity matrix, and α is its solution. However, the preconditioner would need to be exactly the matrix inverse. Since preconditioners should be inexpensive to construct, we choose to satisfy this cardinal condition in the least-squares sense.

4.2.1 The left-hand preconditioner, W

Let ϕ_I and ϕ_B denote the N_I and N_B MQ-RBFs whose center is in Ω and in $\partial\Omega$, respectively. Assume the differential operators $\mathcal{L}$ and $\mathcal{B}$ are continuous, or piece-wise continuous across different subdomains. The rows of A form a set of smooth basis functions in our spline space. From (4.3), they are given by

$$\{\Psi_i(x)\}_{i=1}^{N} = \left\{ \{\mathcal{L}\phi_I(x_i - x)\}_{i=1}^{N_I} \cup \{\mathcal{B}\phi_B(x_i - x)\}_{i=1}^{N_B} \right\}. \tag{4.5}$$

Each column of A has contributions from N_B entries of $\mathcal{B}\phi_B$ and N_I entries of $\mathcal{L}\phi_I$. It is very likely that such an arbitrary column will exhibit a jump discontinuity at

the boundary-interior interface. Therefore, we manipulate the rows of the RBF-PDE matrix A that is a discrete version of (4.5).

The strategy for constructing the *left*-hand preconditioner W is as follows. For each center x_i, select a small subset of centers or *support* of size $m \ll N$ indicated by the index set

$$\mathcal{S}_i = \left[s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{m)}\right], \quad i \in \mathcal{S}_i, \tag{4.6}$$

such that we try to enforce the condition

$$\sum_{j \in \mathcal{S}_i} w_j \Psi_j(x) = \delta_i(x), \quad \text{for all } x \in X, \tag{4.7}$$

where $\delta_i(x)$ is one at x_i and zero elsewhere. The choice of this index set, $\mathcal{S}_i$, will be deferred to a later discussion. Also, (4.7) yields N equations for $m < N$ unknowns. If (4.7) is satisfied for all $x \in X$, we obtain the inverse of A. Instead, we *best-fit* (4.7) in the least-squares sense. Let B_i be the m-by-N matrix formed by selecting m rows of A from the index set $\mathcal{S}_i$ in (4.6), i.e.,

$$B_i = \begin{bmatrix} A(s_i^{(1)},1) & A(s_i^{(1)},2) & \ldots & \ldots & A(s_i^{(1)},N) \\ A(s_i^{(2)},1) & A(s_i^{(2)},2) & \ldots & \ldots & A(s_i^{(2)},N) \\ \vdots & \vdots & & & \vdots \\ A(s_i^{(m)},1) & A(s_i^{(m)},2) & \ldots & \ldots & A(s_i^{(m)},N) \end{bmatrix} \in \mathbb{R}^{m \times N}.$$

Then, we can rewrite (4.7) in matrix form

$$B_i^T w_i = e_i, \tag{4.8}$$

where $w_i^T = [w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(m)}]$ are the non-zero elements of the i-th row of W, and e_i is the i-th N-by-1 standard vector. Each center x_i is associated with a row of the preconditioner W that approximates the i-th ACBF using m different rows of A. Since B_i has full rank m, $B_i B_i^T$ is a nonsingular $m \times m$ matrix. Thus, w_i can be uniquely determined.

Our preconditioner W will have $w_i^T P$ as its rows where P is the m-by-N permutation matrix that maps the elements of w_i from $\mathcal{S}_i$ back to the corresponding global

index set of the i-th row of W, or

$$W_{ij} = \begin{cases} w_i^{(k)} & \text{if } j = s_i^{(k)} \text{ for } k = 1, \ldots, m, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

We obtain the preconditioner W after solving N least-squares problems. Each row of W will operate on the correct rows of A and the preconditioned matrix WA will have the ACBFs as its rows. In general, W will not be constructed explicitly using (4.9). Instead, each w_i and $\mathcal{S}_i$ are stored for later use which cost $2mN$ to store (mN floating-point numbers for w_i and mN integers for $\mathcal{S}_i$).

We designed the preconditioner W so that its construction is simple with relatively few operations. The preconditioner is applicable to the non-symmetric RBF-PDE matrix problem. Our studies focus upon MQ-RBFs. But the preconditioning techniques are general enough to be applied on other RBFs. These preconditioners do not make use of the far-field expansion and can be used on shape parameters that vary locally in the domain. Although our preconditioners are not intended for the interpolation problem, they should have good performance for interpolation as well, as demonstrated later in the section. Unlike the null space methods, our preconditioning technique will transform the symmetric interpolation matrix to an non-symmetric matrix. In general, the eigenvalues of the preconditioned system will be in the complex plane.

4.2.2 Solving the least-squares problems

We consider an *overdetermined* linear system of the form (4.8). The least-squares problem has a unique solution w that minimizes $\|B_i^T w - e_i\|_2$. One solution method for a least-squares problem is to solve the corresponding local* normal equation (L-NE). Multiplying B_i from the left to (4.8) gives the normal equations,

$$B_i B_i^T w_i = B_i e_i. \tag{4.10}$$

By symmetry, the computation of the normal matrix $B_i B_i^T$ requires $m^2 N$ flops. Using Gaussian elimination to solve (4.10) requires $m^3/3$ flops. The overall operation count

*We reserve the term *local-* (or L-) to indicated that the method is employed in the construction of the preconditioner.

is $\sim m^2N + \frac{1}{3}m^3$ flops. This method is not always stable in the presence of round-off errors because the condition number of the normal matrix is twice as large as the original.

The other popular methods are the QR factorization and the SVD method that solve (4.8) directly. Their overall costs are $\sim 2m^2N - \frac{2}{3}m^3$ flops and $\sim 2m^2N + 11m^3$ flops, respectively. Lawson and Hanson [73] claim that for a fixed machine precision, a wider class of least-squares problems can be solved using the QR factorization. Trefethen and Bau [98, lecture 11] suggest that the QR factorization should be used for the typical problem, and that the SVD method should be used if the matrix B_i is nearly rank-deficient. The SVD approach can also be used to solve rank-deficient least-squares problem, see [49, chapter 5].

The construction of our preconditioner W requires solutions of N least-squares problems. Using any of the three above-mentioned methods would result in a set up cost of $\mathcal{O}(m^2N^2 + m^3N)$ flops. The term $O(m^2N^2)$ greatly limits the size of support that we can use to construct the ACBFs. We show that the set up cost can be reduced to $\mathcal{O}(mN^2 + m^3N)$.

Local-normal equation

We first note that the $m \times m$ normal matrix $B_iB_i^T \in \mathbb{R}^{m\times m}$ on the left of (4.10) is a submatrix of AA^T, namely the $(\mathcal{S}_i, \mathcal{S}_i)$ elements of AA^T, i.e.,

$$B_iB_i^T = \begin{bmatrix} AA^T(s_i^{(1)}, s_i^{(1)}) & AA^T(s_i^{(1)}, s_i^{(2)}) & \dots & AA^T(s_i^{(1)}, s_i^{(m)}) \\ AA^T(s_i^{(2)}, s_i^{(1)}) & AA^T(s_i^{(2)}, s_i^{(2)}) & \dots & AA^T(s_i^{(2)}, s_i^{(m)}) \\ \vdots & \vdots & \ddots & \vdots \\ AA^T(s_i^{(m)}, s_i^{(1)}) & AA^T(s_i^{(m)}, s_i^{(2)}) & \dots & AA^T(s_i^{(m)}, s_i^{(m)}) \end{bmatrix}. \tag{4.11}$$

We obtain a considerable amount of storage and computational savings by observing that only $\mathcal{O}(mN)$ elements of AA^T are needed for all N normal matrices. We define the *relevant elements* of AA^T to be only those elements required in the construction of the normal matrix $B_iB_i^T$ corresponding to each x_i (the expensive $O(m^2N)$ step). Referring to (4.11), we see that the $m \times m$ elements of AA^T correspond to a particular normal equation, $B_iB_i^T$. The union of all the relevant elements $i = 1, 2, \cdots, N$

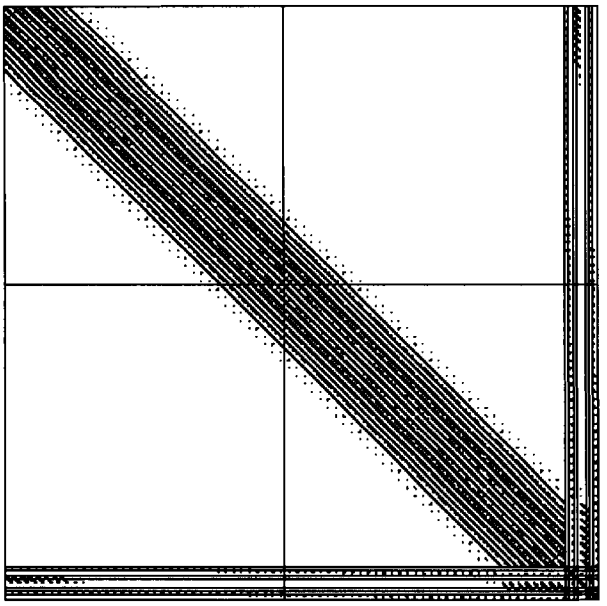

Figure 4.1: Relevant elements ($\sim 3mN$) of AA^T necessary for the construction of all $B_i B_i^T$ matrices.

is depicted in figure 4.1. Due to symmetry, only half of these relevant elements of AA^T need to be computed and stored.

Since the cost of finding any element of AA^T is $\mathcal{O}(N)$ flops, the set up of all normal equations costs $\mathcal{O}(mN^2)$ flops. Then we solve all the N normal equations, each of size m-by-m (4.10), using L-NE to find all rows of W with a cost $\mathcal{O}(m^3)$ flops per row. Note this step can be readily performed in parallel. The total cost of constructing W is therefore $\mathcal{O}(mN^2) + \mathcal{O}(m^3 N)$ flops.

Suppose we evaluate all the relevant elements in the lower triangular portion of AA^T. To find all relevant elements in the i-th column of AA^T, we need a few more rows of A below row i. If storage is not a limiting factor, it is optional to store these rows for later use. Note that after we obtain all the relevant elements in i-th column of AA^T, the i-th row of A will not be used again and can be deleted from memory. Furthermore, the vector on the right of (4.10) consists of the $\mathcal{S}_i$ elements of the i-th row of A, i.e.,

$$Be_i = \left[\; A(s_i^{(1)}, i) A(s_i^{(2)}, i) \ldots A(s_i^{(m)}, i) \; \right]^T .$$

Forming the right-hand vector does not require any extra computations.

A very important work of Beatson and coworkers [7, 8, 11, 13] is the fast RBF matrix-vector product algorithm. The set up cost is only $\mathcal{O}(N \log N)$ flops. The evaluation of the MQ expansion at any $x \in \mathbb{R}^d$ costs $\mathcal{O}(N \log N)$ flops, which is much faster than the $\mathcal{O}(N^2)$ flops for direct matrix-vector multiplication. This algorithm is also applicable for the RBF-PDE matrix A and its transpose A^T. This algorithm can be used to reduce the cost of each iteration.

(Optional) Local-QR and local-SVD

In this section, we provide a more stable algorithm to solve the least-squares problem (4.8) based on local-QR factorization or local-SVD instead of solving the normal equations (4.10) with L-NE. The cost of constructing the preconditioner becomes $\mathcal{O}(mm_l{}^{1/2} N^2)$. Similar to the L-NE approach, the main idea relies on recycling information from the previous least-squares problem.

We adopt the Lawson-Hanson-Chan algorithm for finding the SVD factorization of an N-by-m matrix M, see [98] for details. In phase one, we begin by computing the reduced QR factorization that requires $2m^2N - \frac{2}{3}m^3$ flops. Then, $M = QR$. In phase two, the Golub-Kahan bidiagonalization scheme is applied to the resulting upper triangular matrix of the QR factorization, $R = \mathcal{U}\Sigma\mathcal{V}^T$, that require $\frac{8}{3}m^3$ flops. The SVD factorization of M is given by $(QU)\Sigma V^T$.

Since the same special index set is used for all centers, we place the special points at the beginning of all index sets $\mathcal{S}_i$ defined by (4.6). Next, suppose that the QR factorization of $B_{i-1}^T = Q_{i-1}R_{i-1}$ is known, and we want to construct B_i^T with minimal extra flops. Without loss of generality, we assume that the centers x_{i-1} and x_i are close together.

We define a non-commutative operator $\ominus$ on the index set such that $\mathcal{A} \ominus \mathcal{B}$ is the set difference of $\mathcal{A}$ and $\mathcal{B}$, i.e., the elements that exist in set $\mathcal{A}$ but do not exist in the set $\mathcal{B}$. To reuse the QR factorization of B_{i-1}^T, we need to remove the columns that correspond to the index $\mathcal{S}_{i-1} \ominus \mathcal{S}_i$ and replace them with the new columns corresponding to the $\mathcal{S}_i \ominus \mathcal{S}_{i-1}$ columns of A^T. Assume that there are $K_1 = |\mathcal{S}_i \ominus \mathcal{S}_{i-1}|$ columns that need to be replaced, and K_2 columns in B_{i-1}^T that need to be reordered. Let $\mathcal{P}$

Algorithm 3 Pseudo-code for local-QR and local-SVD.

1. Reorder the last K_2 columns of B_{i-1}^T, using the permutation matrix $\mathcal{P}$, i.e.,
$$B_{i-1}^T \mathcal{P} = Q_{i-1}(R_{i-1}\mathcal{P}).$$
2. Compute the LU factorization with partial (or complete) pivoting of the m-by-m matrix,
$$R_{i-1}\mathcal{P} = P_{lu}^T LU.$$
Then we have
$$B_{i-1}^T \mathcal{P} = (Q_{i-1}P_{lu}^T L)U.$$
At this point, we obtain a new QR factorization of B_{i-1}^T whose first $m - K_1$ columns are in common with that of B_i^T.
3. Let Q_i and R_i be the first $m - K_1$ columns of $Q_{i-1}P_{lu}^T L$ and U, respectively.
4. Continue the Gram-Schmidt orthogonalization process on the columns of A^T indexed by $\mathcal{S}_i \ominus \mathcal{S}_{i-1}$, and complete the QR factorization $B_1^T = Q_i R_i$.
5. (a) If L-QR is used to solve $B_i^T w_i = e_i$, we solve the upper-triangular system $R_i w_i = Q_i^T e_i$ for w_i.
   (b) If L-SVD is used, we first compute the SVD factorization of the m-by$m$ matrix $R_i = \mathcal{U}\Sigma\mathcal{V}^T$. Then $w_i = \mathcal{V}\Sigma^{-1}\mathcal{U}^T Q_i^T e_i$.

denote the m-by-m permutation matrix that reorders the last K_2 columns of B_{i-1}^T so that the right-most K_1 columns of $B_{i-1}^T\mathcal{P}$ are not reused by B_i^T. In general we have $K_1 < K_2 \leq m_l$. We outline the procedure thusly in algorithm 3.

The computational effort involved in this procedure is dominated by step 3 and step 4.

Due to the ordering of $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$, we know that the matrices $\mathcal{P}$, P_{lu}, and L are all of the form $\begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}$, where 0 denotes the zero block matrix, X is the permutation matrix in both $\mathcal{P}$ and P_{lu}. X is a lower triangular matrix in L. The first m_s columns of Q_i that correspond to the special points remain unchanged. We only need to compute $K_2 - K_1$ columns of $Q_{i-1}P_{lu}^T L$ that are linear combinations of the K_1 to

$K_2 - K_1$ columns of the matrix Q_i in step 3. The cost of updating these orthonormal vectors of length N is $\sim K_1(2K_2 - K_1)N$ flops.

After step 3, we are K_1 steps away from completing the QR factorization of B_i^T. In step 3 the QR factorization is completed with another $\sim 2(2m - K_1)K_1N$ flops. All other steps of the algorithm require either $\mathcal{O}(m^3)$ flops or less.

To obtain an upper bound for the estimated flop count, we estimate $K_1 \sim m_l^{1/2}$ in two-dimensions, and pick the upper bound $K_2 = m_l$. Then the overall cost of solving one least-squares problem using local QR or local SVD is

$$\sim 2{m_l}^{1/2}\,(m_l + 2m)N + \mathcal{O}(m^3).$$

This estimate suggests that we should use more special points and fewer local points. A look-ahead algorithm for the ordering of the index set can further reduce the overall effort by minimizing K_2. Both L-QR and L-SVD have the same $\mathcal{O}(m^3)$ term in the overall cost estimate; but the constant of L-QR is smaller than L-SVD. We periodically restart the procedure presented above to avoid the accumulation of round-off errors[†].

4.2.3 The Role of Special Points in the Construction of the Cardinal Basis Functions

We adopted the strategy in [7, 85] for choosing the index set, $\mathcal{S}_i$. The subset is a combination of local and special points. We pick $\mathcal{S}_i$ to be the m_l local nearest centers to x_i that can be done efficiently in $\mathcal{O}(N \log N)$ flops, and m_s special points where $m_l + m_s = m$.

The choice of special points is another important issue. In general we want the special points to control the shapes and ensure the linear independence of all the ACBFs in the domain Ω. When the nearest neighbor subset is unbalanced about x_i, the least-squares ACBFs centered at different points near the boundary points may become too similar in shape, have considerable overlap, and become considerably less linear independent without the presence of special points. Fig 4.2 (a) shows one such example.

[†]QR-factorization could be used in step 2 instead of the LU-factorization; however, the cost of step 3 becomes $\mathcal{O}(m_l^2 N)$.

Assume that the centers have equal spacing, h. The ACBFs centered at $x_1 = 0h$ and $x_2 = 1h$, respectively are similar in shape and have considerable overlap. Two or more nearly degenerate ACBFs typically result in one or more eigenvalues clustered around zero in the preconditioned matrix system. To solve this problem, we use the last point $x_N = 1$ as a special point, see Fig 4.2 (b). The shapes of the two modified ACBFs are much more distinct with smaller overlap, their peaks are closer to one, and they decay faster toward zero.

We offer the following explanation why the special points have such an important influence. An ACBF can be constructed from a discrete Laplacian, see [7, 85]. Consider the discrete second derivative in one dimension, using a three-point stencil scheme evaluated at x_1 and x_2 of the same support, the two ACBFs will have exactly the same approximation, and are degenerate.

Our situation is similar to this trivial case. When an unbalanced local neighborhood is used to construct the ACBF at x_i, the resulting new ACBF is similar (although not exactly the same) to the nearby ACBFs at x_{i-1} and x_{i+1}. Whenever two or more ACFBs are similar in shape and have considerable overlap, such a situation leads to ill-conditioning.

We pick a set of universal special points that controls the distinctness of each ACBF for each x_i in $\Omega \cup \partial\Omega$. A given center, x_i, could be very near or could be one of the special points. But the remainder of the special points would be relatively far away and would help constructing a good approximation to an ACBF. Although we have focused our attention on the unit square, we can find special points on arbitrarily shaped domains in two and three dimensions. If Ω is convex, the first special point will be that center closest to the centroid of the domain. Pick centers on $\partial\Omega$ that are furthest away from the centroid. Continue computing the distances between the existing special points and index the furthest boundary center as a special point if the candidate is not *too close* to the existing special points. Then we start bisecting the distances between the pairs of candidate special points with the largest separation distances. We can continue this procedure to obtain any desired number of special points.

For example if the domain is contained in the unit square $[0, 1]^2$, we want to have at

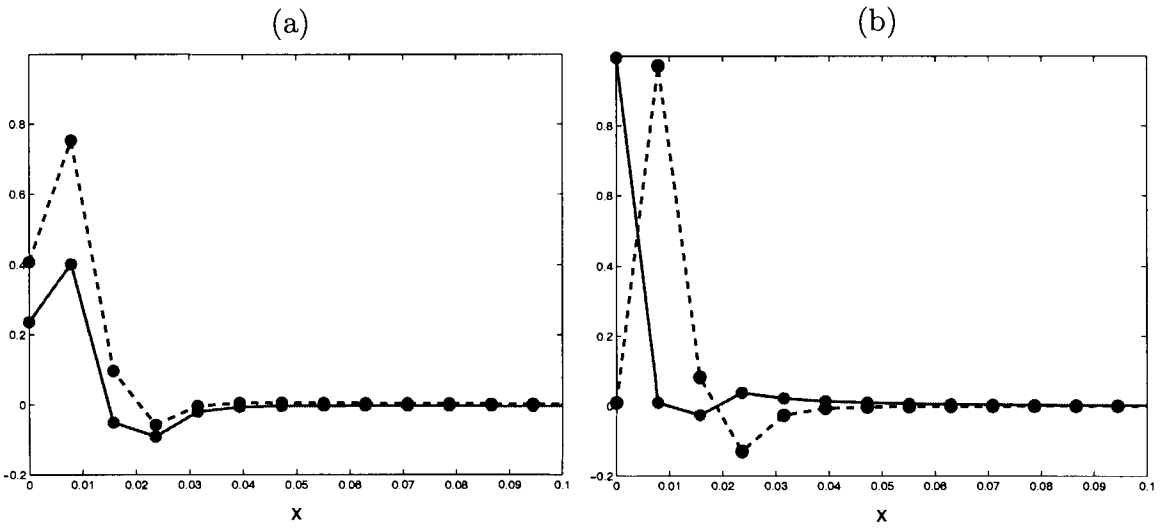

Figure 4.2: Effects of special points: (a) 4 local centers; (b) 4 local centers + 1 special point. The solid and dashed line are the ABCFs centered at x_1 and x_2, respectively. The dots indicate the location of centers.

least four special points that are chosen to be the centers closest to the four corners $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ respectively. If nine special points are desired, one would chose the centers that are closest to centroid, $(0.5,0.5)$ and the four corners. By bisecting the distances between special points that are furthest apart (i.e. the corners), we would find the following special points: $(0.5,0)$, $(0.5,1)$, $(0,0.5)$, $(1,0.5)$, $(0.5,0.5)$. This will give a set of nine special points.

If appended polynomials are used in the expansion of solution (1.5), we treat and refer to the rows of polynomial as *special.*

4.2.4 (Optional) The right-hand preconditioner, V

Once the left-hand preconditioner W is found, one can apply a similar process to the columns of WA to construct a right-hand preconditioner $V \sim (WA)^{-1}$. We define C_i to be the submatrix of the $\mathcal{S}_i$-th columns of WA. If the same subset $\mathcal{S}_i$ used in the construction of W is re-used for every center, we avoid extra computation and

storage. For

$$C_i = \begin{bmatrix} WA(1, s_i^{(1)}) & WA(1, s_i^{(2)}) & \dots & WA(1, s_i^{(m)}) \\ WA(2, s_i^{(1)}) & WA(2, s_i^{(2)}) & \dots & WA(2, s_i^{(m)}) \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ WA(N, s_i^{(1)}) & WA(N, s_i^{(2)}) & \dots & WA(N, s_i^{(m)}) \end{bmatrix} \in \mathbb{R}^{N \times m},$$

the corresponding least-squares problem is $C_i\, v_i = e_i$. The normal equations for each column of V is $C_i^T C_i\, v_i = C_i^T e_i$ for $i = 1, \dots, N$ where $v_i = [v_i^{(1)}, v_i^{(2)}, \dots, v_i^{(m)}]$ are the non-zero elements of the i-th column of V. Our right-hand preconditioner V will have Pv as its columns where P is again the same permutation matrix used in the construction of W, or

$$V_{ij} = \begin{cases} v_j^{(k)} & \text{if } i = s_j^{(k)} \text{ for } k = 1, \dots, m, \\ 0 & \text{otherwise.} \end{cases}$$

Similar to W, the preconditioner V only has m nonzero elements in each column; the matrix-vector product of V can be performed in $\mathcal{O}(mN)$ flops. The set up cost of V is more expensive. To solve its corresponding normal equations, we need to find $C_i^T C_i$ that is a submatrix of $(WA)^T(WA)$. Each relevant element costs $\mathcal{O}(mN + N)$ flops to compute. The total set up cost is then $\mathcal{O}(m^2N^2)$ flops and the storage requirement remains $\mathcal{O}(mN^2)$ locations.

4.3 Working with ACBF Preconditioners

4.3.1 Interpolation with various shape parameters

Figure 4.3 shows the effectiveness of both preconditioners applied to the interpolation problem. We randomly generated 289 centers on $[0, 1]^2$, see Figure 4.3 (a). Special points are denoted by the ($\times$) and ($\star$). The MQ shape parameters are chosen to be the minimum separation distance to the closest center, $h_{\min}$, and this example is a variable shape parameter problem.

The small eigenvalues of A can be found in Figure 4.3 (b). First, we used 12 local points and 4 special points (all $\times$). All the complex eigenvalues of WA and WAV are plotted, see Figure 4.3 (c) and (d), respectively. After the application of W, all the real parts of the eigenvalues clearly cluster around 1. When both preconditioners W and V are applied, we get an even better clustering about the real value of 1. This clustering of eigenvalues about one would allow any Krylov subspace iteration to converge quickly.

Figure 4.3 (e) shows the eigenvalues distributions when 18 local points and 9 special points (both $\times$ and $\star$) are used. Using a larger support, m, results in a denser clustering about the real value of 1 as expected. The cost of computing W with this larger $m = 18 + 9$ is much lower than the cost of forming V in the previous case with $m = 12 + 4$. We conclude that the right-hand preconditioner V is not practical unless its set up cost can be reduced.

4.3.2 RBF-PDE test problems

In this section we will study the performance of our preconditioners applied to PDEs. All codes are written in Matlab 6.0 executed with double precision arithmetic ($\epsilon_{machine} = 2.22 \times 10^{-16}$) and executed on a 4 $\times$ 500MHz Alpha machine with 2GB RAM . The normal equations (4.10) are solved by the Matlab built in *left matrix divide* function.

Although the condition number for the matrix A is not as informative here as for the symmetric case, it will be reported for completeness. We measure the performance in terms of the number of iterations required for GMRES (without restart) to converge within a 10^{-6} tolerance. Both the mean square residual (MSR) errors

$$\text{MSR} = \sqrt{\sum_i \frac{(U(\xi_i) - u(\xi_i))^2}{N}}, \tag{4.12}$$

and the maximum (MAX) errors

$$\text{MAX} = \max_i |U(\xi_i) - u(\xi_i)|, \tag{4.13}$$

are used to measure the accuracy of the numerical solution. In (4.12) and (4.13),

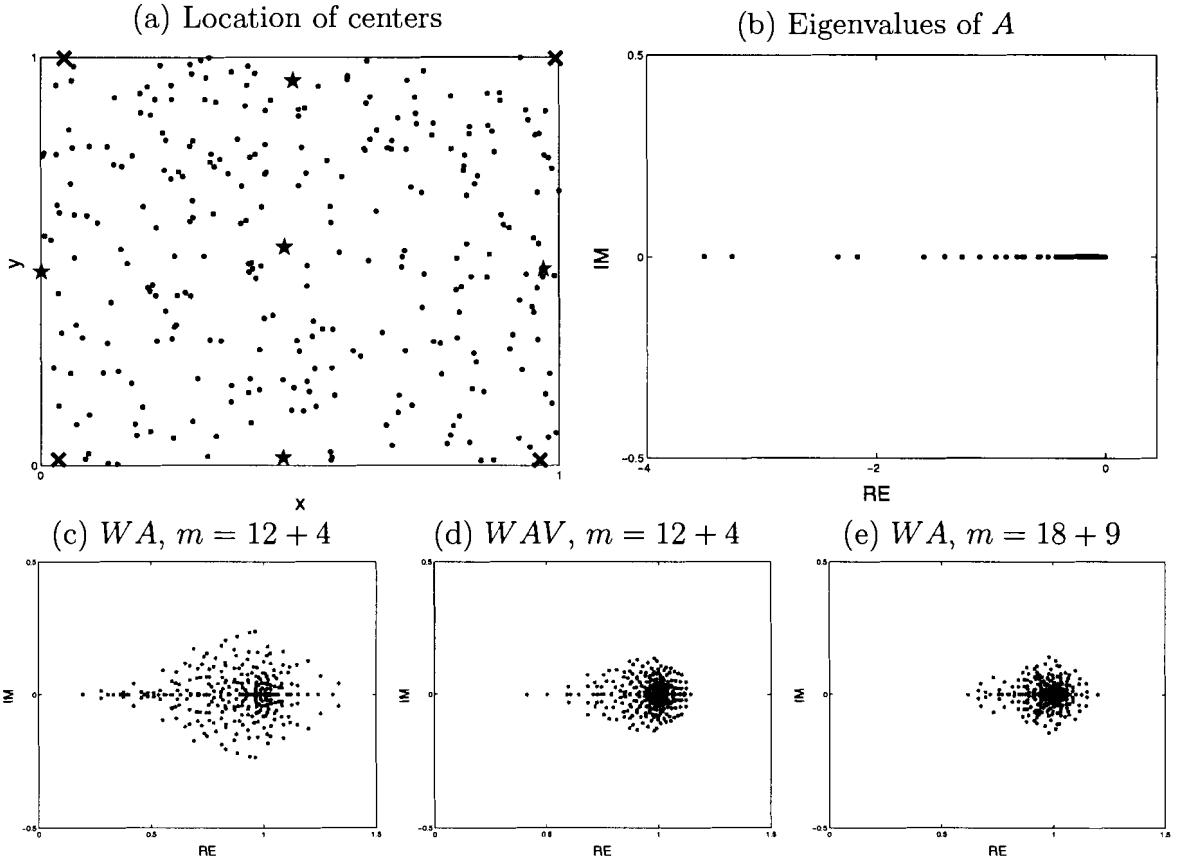

Figure 4.3: MQ-RBF interpolation: (a) 289 random centers on $[0, 1]^2$ with $c = h_{\min}$; (b) Eigenvalues (real) of A around the origin. All the complex eigenvalues of the preconditioned system: (c)WA, and (d) WAV are for $m = 12 + 4$, and (e)WA are for $m = 18 + 9$.

U is the approximate solution and u is the exact solution. In all tables, $d_0.d_1d_2(e)$ represents the number $d_0.d_1d_2 \times 10^e$.

We set up a boundary value problem for the Poisson equation in the unit square $[0, 1]^2$,

$$\begin{aligned} \nabla^2 u(x,y) &= \frac{-751\pi^2}{144} \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4} \\ &\quad + \frac{7\pi^2}{12} \cos\frac{\pi x}{6} \cos\frac{7\pi x}{4} \sin\frac{3\pi y}{4} \sin\frac{5\pi y}{4} \\ &\quad + \frac{15\pi^2}{8} \sin\frac{\pi x}{6} \sin\frac{7\pi x}{4} \cos\frac{3\pi y}{4} \cos\frac{5\pi y}{4}, \end{aligned} \tag{4.14}$$

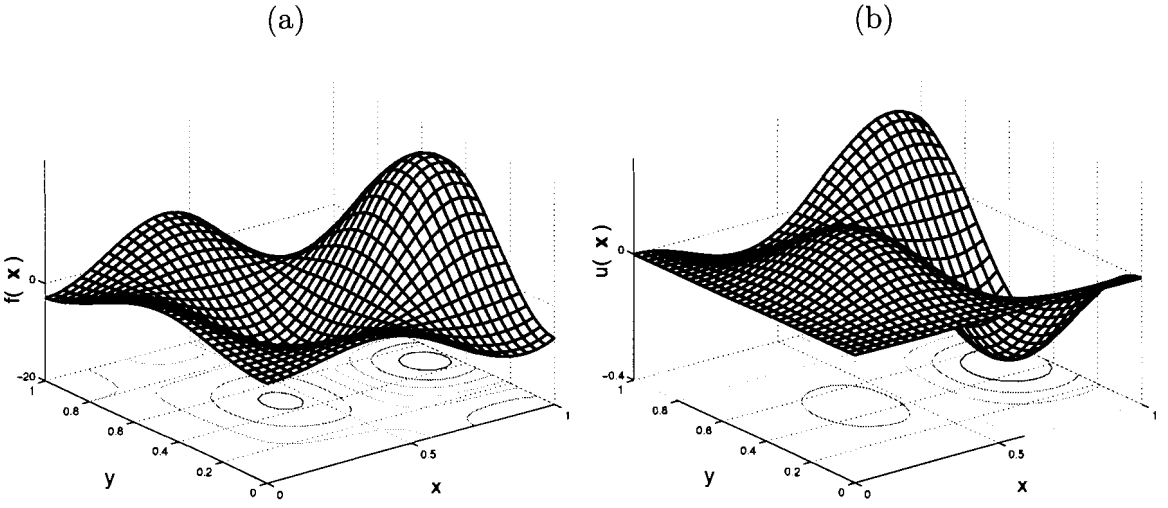

Figure 4.4: (a) the left hand function $f(x)$, and (b) the exact solution $u(x)$ of our test problem.

which has exact solution

$$u(x,y) = \sin\frac{\pi x}{6}\sin\frac{7\pi x}{4}\sin\frac{3\pi y}{4}\sin\frac{5\pi y}{4}. \tag{4.15}$$

We assume Dirichlet boundary conditions on all sides of the unit square whose values are given by (4.15). Figures 4.4 (a) and (b) plot the right hand side function of (4.14) and the exact solution (4.15), respectively.

4.3.3 Performance of the preconditioners for $c = \frac{1}{\sqrt{N}}$

Regularly spaced data center distribution

As an initial test, we solved (4.14) on equally spaced $N = n \times n$ centers where $n = 8j + 1$ for $j = 2, \ldots, 9$. A constant shape parameter $c = \frac{1}{\sqrt{N}}$ is used in each case. We used the L-NE scheme to construct the preconditioner W. For shape parameters in this size range, the L-SVD scheme is not necessary. Both the mrs and max errors are calculated by interpolating the solution onto a 81-by-81 grid and comparing the numerical solution with the exact solution. In this example, we used 50 local points and 9 special points for the construction of the preconditioners. We define itr_c as the number of GMRES iterations required to achieve convergence within the specified

N	Condition Number		Errors		Iterations	
	A	WA	MSR	MAX	Actual	Relative
289	2.49(5)	4.07(0)	6.85(-3)	6.19(-3)	8	$0.0277N$
625	9.10(5)	1.10(1)	2.44(-3)	3.16(-3)	12	$0.0192N$
1089	2.23(6)	1.99(1)	1.20(-3)	1.97(-3)	17	$0.0156N$
1681	4.46(6)	3.07(1)	6.99(-4)	1.33(-3)	22	$0.0131N$
2401	7.82(6)	4.37(1)	4.51(-4)	1.04(-3)	27	$0.0112N$
3249	1.25(7)	5.88(1)	3.13(-4)	8.25(-4)	31	$0.0095N$
4225	1.89(7)	7.65(1)	2.28(-4)	6.60(-4)	36	$0.0085N$
5329	2.70(7)	9.67(1)	1.73(-4)	5.37(-4)	40	$0.0075N$

Table 4.1: Numerical results for a regularly spaced data center distribution for $c = \frac{1}{\sqrt{N}}$, $m_l = 50$, and $m_s = 9$.

tolerance, 1×10^{-6}. For all tested N's, the solution errors agree with each other to at least 3 significant digits whether the system is solved by GMRES or Gaussian elimination. Only one set of errors is reported. The results are summarized in table 4.1.

As N increases so do the condition numbers of the A matrices. The condition numbers of the preconditioned systems are between $\mathcal{O}(1)$ to $\mathcal{O}(10)$. Since the size of $\mathcal{S}_i$ is kept constant, the set up cost remains constant at $\mathcal{O}(N^2)$ flops. On the other hand, a relatively small support is used to approximate the ACBFs for larger N. The number of iterations required for GMRES to converge, itr_c, and the condition number of preconditioned system slowly increase with increasing N. Although it is not remaining constant, we clearly see that smaller and smaller fractions of the N iterations are required as N increases. We plot the ratio of the estimated flops for GMRES to converge over the flops required by direct method, G-GE, in figure 4.5. We see that the overall flop ratio of our scheme requiring $\mathcal{O}(N^2)$ flops to the G-GE scheme requiring $\mathcal{O}(N^3)$ flops is approaching $35.47\, N^{-0.89}$.

Scattered data centers

Our preconditioners can work equally well on a scattered set of centers. The same computations are repeated on random sets of interior centers of total size 5329. The x-

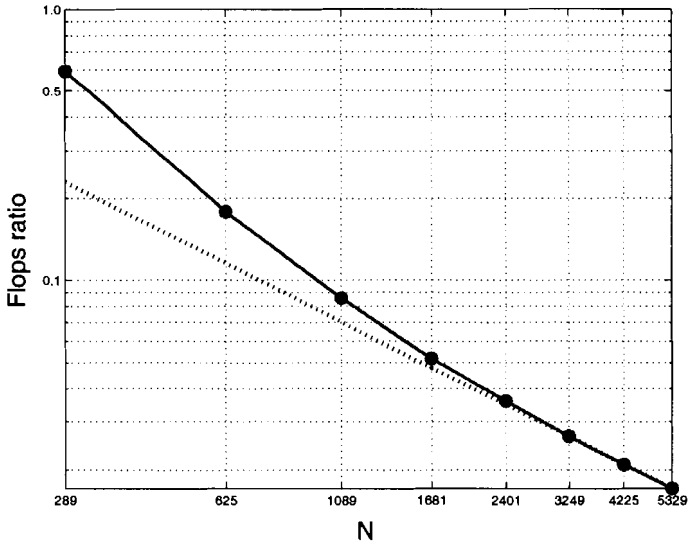

Figure 4.5: The ratio of the estimated flops for GMRES to converge over the flops required by direct method, G-GE. The ratio is approaching $35.47N^{-0.89}$ (dotted line) as N increases.

and y-coordinates of the random interior centers were generated separately using the Matlab built-in *rand* function calls. We did not re-order the centers using any sorts, or enforce any pre-set minimum separation distance. The shape parameter is chosen to be the minimum separation distance to the closest center. The minimum and maximum separation distances of the random centers are 1.73×10^{-8} and 5.26×10^{-4}, respectively. The average separation distance is 6.47×10^{-5}.

The condition numbers are 3.71×10^{7} and 1.50×10^{2} for the original and preconditioned matrices, respectively. GMRES takes 48 iterations to converge for this problem. The MSR and MAX errors are 2.15×10^{-3} and 5.81×10^{-3}, respectively. The condition numbers and number of iterations required for GMRES to converge are larger than in the comparable regular center distribution case. This is likely due to the fact that we did not enforce a bound on the minimum separation distance between the centers. Since the centers are placed randomly without any intention of optimization, the resulting solutions of random centers are also less accurate than in the regular case.

4.3.4 Range of workable MQ shape parameters

Madych [79] has shown the convergence rate of MQ can be accelerated either by increasing c or decreasing h. For a fixed h or N, the errors decrease monotonically as c increases reaching a minimum at a critical value of c. Using global Gaussian elimination (G-GE), the system (4.2) will have different minimum errors depending upon whether the arithmetical operations are performed in single or double precision. Although double precision arithmetic permits one to use a much larger value of c, the accumulation of round-off errors for increasing c eventually becomes so large that even double precision arithmetic cannot prevent numerical instabilities. Beyond this critical value, the solutions become unstable, and the errors rise dramatically.

Fornberg and co-workers, [39, 40, 72] have demonstrated extra-ordinary convergence rates with the C^{∞} RBFs such as multiquadrics and Gaussians in the limit as $c \rightarrow \infty$. They show that such convergence is not possible with piecewise smooth RBFs such as thin plate splines and r^{2k+1} splines. Near the critical value of the shape parameter, they permit the shape parameter to become complex, and find that branch points and removable poles cause singularities. They removed such singularities by using Cauchy-Pade contour integration. In [72], Larrson and Fornberg solved the Poisson equation achieving remarkable accuracy. Their MSR errors were on the order of 1×10^{-12} with relatively few data centers.

Increasingly larger c for a fixed N

We fix $N = 1089$ and $m_s = 9$. We let the shape parameter vary $c = \sqrt{j/N}$ for $j \in \mathbb{N}$ and performed a numerical search for the shape parameter at which the preconditioned GMRES fails to converge within N iterations. We studied three cases: $m_l = 10$, $m_l = 30$ and $m_l = 50$. This study permits one to determine whether our preconditioning technique is suitable for a given problem. We report the results graphically. All results can be categorized into two parts: (i) the smooth region for small values of $c \leq c^{\star}$, and (ii) the oscillatory region for large values of $c > c^{\star}$ due to round-off error, where $c^{\star}$ depends on N and m in general. In this set of numerical experiments, preconditioners are constructed using the L-NE approach.

For all tested values of the shape parameter, we plot the condition numbers of the unpreconditioned matrix A and of the preconditioned matrices in figure 4.6 (a). The number of iterations required for the GMRES method to converge within tolerance are shown in figure 4.6 (b). For $c = \sqrt{24/N}$ and $c = \sqrt{25/N}$, GMRES fails to converge after N iterations or stagnates for $m = 30 + 9$ and $m = 50 + 9$. As we continue the numerical experiments, we find that for $m = 10 + 9$ GMRES fails to converge when $c = \sqrt{50/N} \sim 1.23$. The number of iterations required for GMRES to converge increase as c increases. The decreasing convergence rate with increasing c is caused by the increasing ill-conditioning of the normal equations (4.10), in which case, the preconditioner W does not approximate the ACBF well due to round-off errors. We show the condition number of the most ill-conditioned normal equation in figure 4.6 (d).

Brown and Walker [16] find that GMRES breaks down without determining a solution through rank deficiency, or determines a solution without break down, but breaks down at the next step due to degeneracy of the Krylov subspace.

When GMRES does converge, both the MSR and MAX errors agree with the corresponding results from G-GE. The critical shape parameters are $c^\star \sim \sqrt{9/N}$ and $c^\star \sim \sqrt{17/N}$ for the $m = 30+9$ and $50+9$ cases, respectively. After $c^\star$, the condition numbers of the preconditioned system no longer increase smoothly with c but oscillate instead. In the case of $m = 10 + 9$, the round-off errors do not have any effect on those results for all tested c. Taking accuracy and computational cost into account, for $N = 1089$, our preconditioning technique can efficiently solve (1.4) for $c \leq \sqrt{10/N}$ with $\sim 0.1N$ GMRES iterations.

As for the previous example, we compare the total flops for solving the linear system with the preconditioned GMRES and G-GE. The results are shown in figure 4.7. Although using $m = 50 + 9$ allows GMRES to converge quicker in comparison to the other support sizes, we see from figure 4.7 that it is $m = 30+9$ that gives the best overall performance in the measure of total cost. In this example, our preconditioned scheme requires less computational efforts than G-GE to solve linear systems as long as GMRES converges. In particular, our preconditioning scheme requires less than 18% of the effort required by G-GE to solve the RBF-PDE problem for $c < 0.1$. In

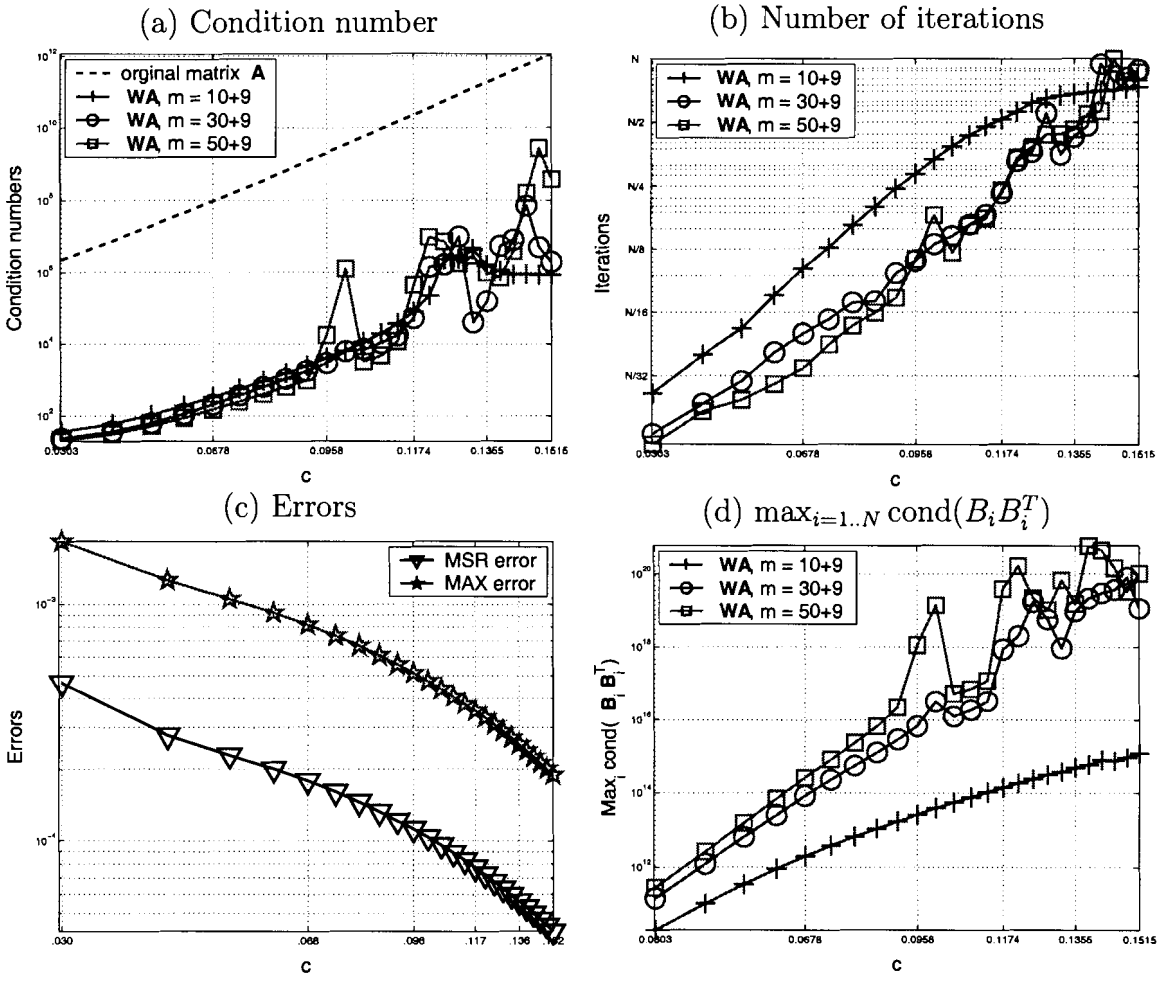

Figure 4.6: (a) Condition numbers of the original RBF-PDE matrix for $N = 1089$ and the preconditioned matrices as a function of $c = \sqrt{\frac{i}{N}}$. (b) The corresponding number of iterations GMRES required to converge within a 10^{-6} tolerance. (c) The resulting MSR errors and MAX errors of the preconditioned RBF-PDE solutions with different constant shape parameters on a log-log scale. (d) The condition number of the most ill-conditioned normal equation in the construction of preconditioner.

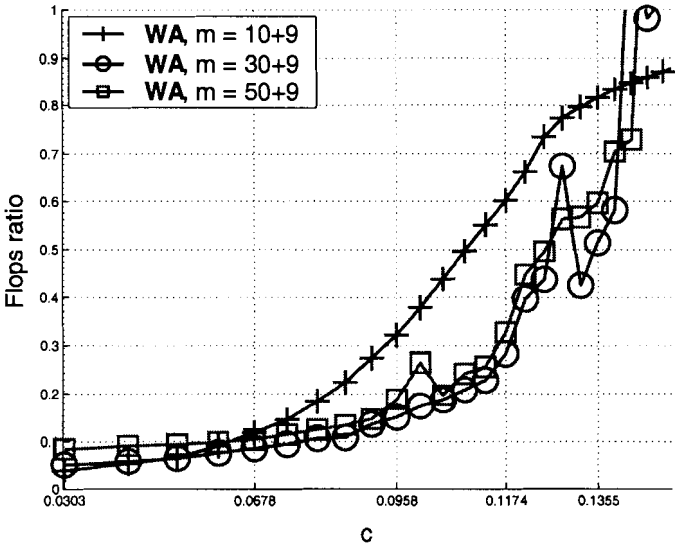

Figure 4.7: Estimated ratio of flops for preconditioned GMRES to converge over that the flops required by the direct method, G-GE. The preconditioned GMRES is more efficient than the direct method even when a larger number of iterations are required for the ill-conditioned matrices.

the case of $c < 0.05$, using the support of size $m = 10 + 9$ and $30 + 9$ one can solve the problem with a similar overall effort.

Increasingly larger N for a fixed c

Ill-conditioning will also arise if one uses a large number of knots, N. In this test, we ran the calculations at a fixed $c = 0.1$, and solved (4.14) on equally spaced $N = n \times n$ centers where $n = 8j+1$ for $j \in \mathbb{N}$ until GMRES fails to converge. The preconditioner W is constructed using 50, 30, or 10 local and 9 special points as in section 4.3.4. The condition numbers and the number of iterations for GMRES to converge to a 10^{-6} tolerance are shown in figure 4.8 (a) and (b), respectively. Results are not shown if GMRES fails to converge.

From the condition number of the most ill-conditioned normal equations in the construction of the preconditioner, see figure 4.8 (d), we see that if $\max_{i=1..N} \text{cond}(B_i B_i^T) > \mathcal{O}(10^{16})$, the L-NE approach does not perform as well as it should. Using a larger

support may result in slower convergence. If $\max_{i=1..N} \text{cond}(B_i B_i^T) > \mathcal{O}(10^{20})$ the preconditioner breaks down and GMRES does not converge.

We know that L-SVD is more stable than L-NE when we are dealing with nearly rank-deficient least-squares problems. But, the set up cost of the preconditioner constructed using L-SVD is higher, see section 4.2.2. If L-SVD is used to solve the corresponding least-squares problems (4.10), GMRES only fails to converge when $N = 4225$ and $m = 50 + 9$. The relative ratio of overall costs using L-SVD is shown in figure 4.9 (b); that of L-NE is shown in figure 4.9 (a) for comparison.

For a given problem, we can divide the shape parameter into three regions:

1. Small c: our preconditioner has excellent performance using L-NE.

2. Moderate c: L-NE starts to break down due the round-off errors. One should consider switching to L-SVD.

3. Large c: Our preconditioning technique is no longer applicable. Other numerical techniques should be used instead; e.g. domain decomposition.

4.3.5 Discussion

We have developed an effective preconditioner scheme for the asymmetric collocation scheme in which radial basis functions (RBFs) are used to solve partial differential equations (PDEs) problems. This method works well for elliptic, hyperbolic and parabolic PDEs and for the volume integral PDE formulation [71].

An ideal approximate cardinal basis function (ACBF) is equivalent to a delta function, $\delta(x_i)$ that is one at it center, x_i, and zero everywhere else. We start from the coefficient matrix, A, that is constructed from the PDE boundary condition and interior operators acting upon the RBFs. For each center x_i, we select a set containing special points and local neighboring centers of size $m \ll N$. The inclusion of special points in this subset that ensures each ACBF is more distinct in shape.

We find the set of weights w_i at each center by solving each least-squares problem to construct the global preconditioner W. The set of least-squares problems is solved

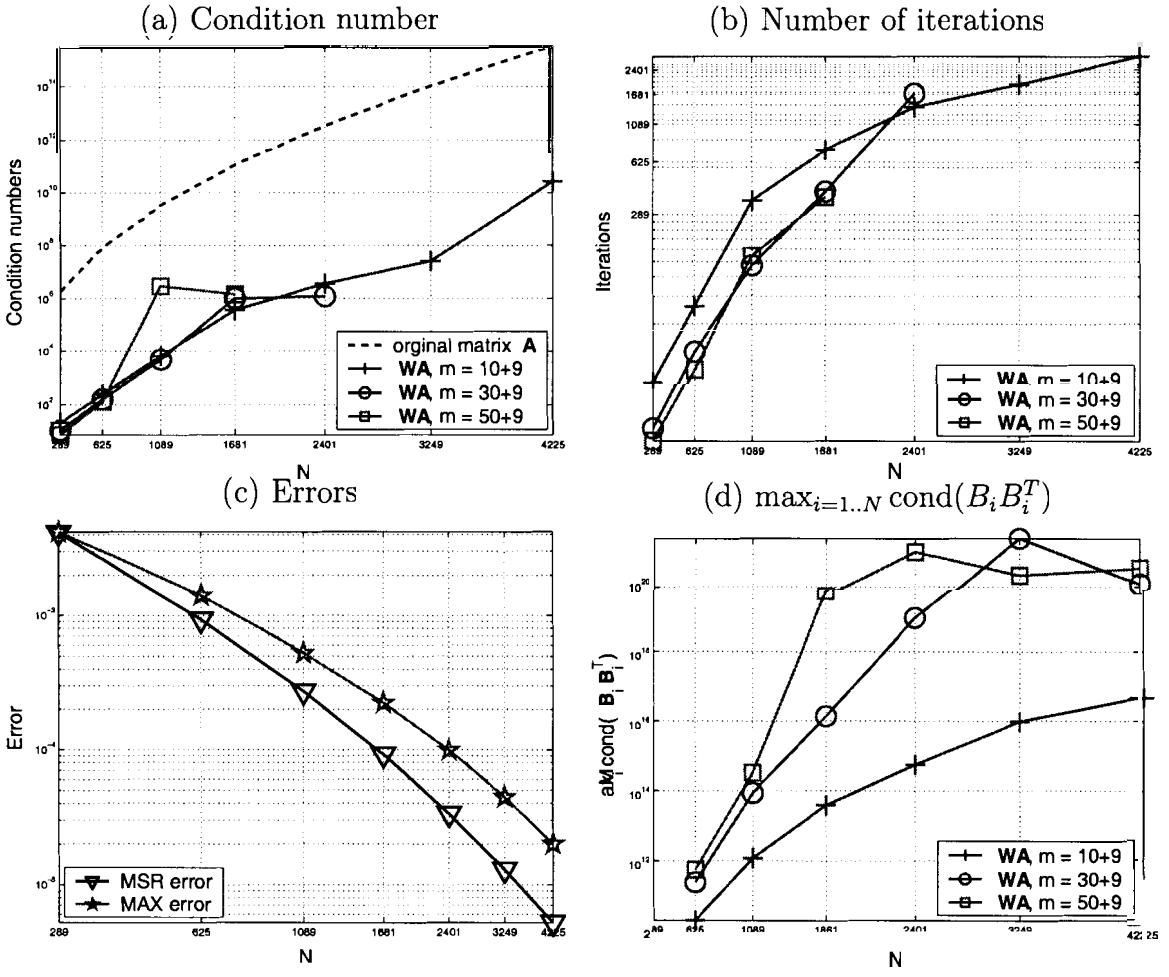

Figure 4.8: (a) Condition numbers of the original RBF-PDE matrix for $c = 0.1$ and the preconditioned matrices as a function of N. (b) The corresponding number of GMRES iterations required to converge within a 10^{-6} tolerance. (c) The resulting MSR errors and MAX errors of the preconditioned RBF-PDE solutions with different constant shape parameters on a log-log scale. (d) The condition number of the most ill-conditioned normal equation used in the construction of the preconditioner.

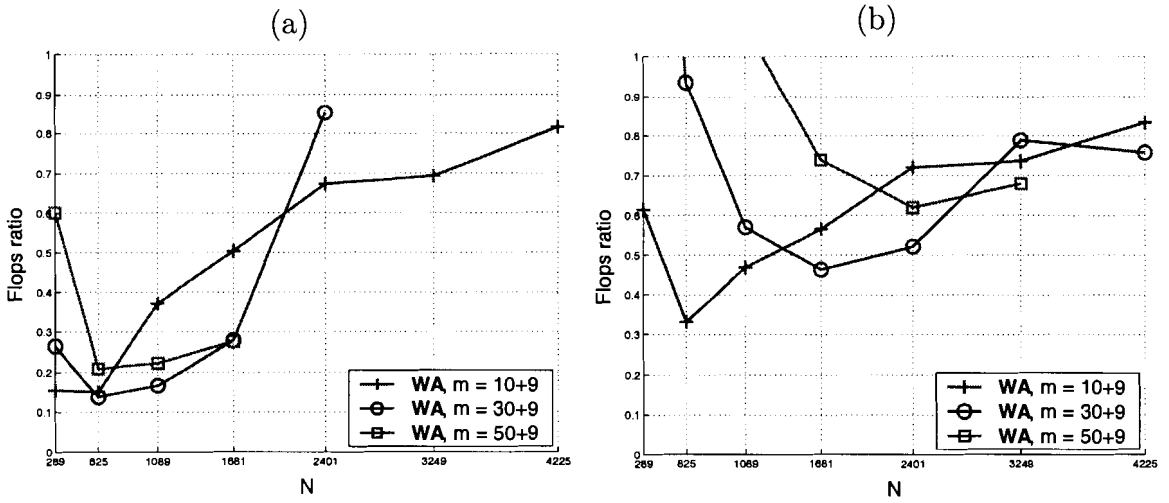

Figure 4.9: Ratio of the estimated flops for preconditioned GMRES to converge over the flops required by the direct method, G-GE. The preconditioner W is constructed using (a) L-NE and (b) L-SVD.

using either a local normal equation approach (L-NE), a local QR factorization (L-QR), or a local singular value decomposition (L-SVD) method.

The preconditioner W is constructed from the sets of weights w_i that transforms the elements of the RBF-PDE coefficient matrix A into ACBFs. The preconditioner W that acts as an approximation to A^{-1} clusters the real part of the complex eigenvalues of WA about 1 permitting the solution of the RBF expansion coefficients to be solved by the very efficient GMRES iteration scheme. With the set of expansion coefficients determined, we reconstruct the numerical solution u over the domain, Ω.

Our intent is to be able to obtain max and mean root square errors with our preconditioning scheme that are less than or equal to the same errors obtained by global Gaussian elimination, but requiring far fewer flops. In its present form, our preconditioner represents a major breakthrough in the computational efficiency of the RBF-PDE method. However, we make the caveat that this computational efficiency holds as long as safe values of the shape parameter are used to prevent severe ill-conditioning in both the global RBF-PDE coefficient matrix A and the normal equation matrices $B_i B_i^T$ that are used to construct the preconditioner W.

The numerical solutions obtained from GMRES iteration on an elliptic PDE test

problem found in [24] are reported. We tested the ACBF method for various size problems and values of the MQ shape parameter $c(N)$ for regularly spaced and scattered center distributions. We found that the shape parameter plays a very important role.

We explain why the preconditioning technique fails to work for large values of the shape parameters and N. The ACBFs become less independent when c or N increases as reflected in the condition number of A. When more local points are used, the normal equations used to find W increases in rank and (4.10) becomes more ill-conditioned. Since the elements of the preconditioners cannot be found accurately using L-NE, the condition numbers of the preconditioned systems WA start to oscillate with increasing c reflecting the effect of round-off errors. One possible way to solve the least-squares problems (4.8) is using L-SVD instead of the L-NE. Using L-SVD, the number of iterations and condition number increase smoothly as c or N increases. L-SVD yields more robust preconditioners when c is large. However, the number of operations required to construct the preconditioner becomes $\mathcal{O}(mm_l{}^{1/2}N^2)$ flops.

A large shape parameter is desired for faster convergence rates and higher accuracy according to the theoretical studies of [79]. One way to extend the use of our preconditioning technique to solve the RBF-PDE systems with large c is to use better-approximated ACBFs by using a larger m. As shown in section 4.3.4 and section 4.3.4, our preconditioner will eventually break down if one keeps increasing c or N.

The other attractive possibility of using larger values of c and/or N is to combine our preconditioner with the domain decomposition method (DDM), [12, 70, 75, 64, 111]. DDM splits Ω into K overlapping or nonoverlapping subdomains, each of which contains $N_k \ll N$ centers. The smaller dimension coefficient matrices are orders of magnitude better conditioned than the corresponding global coefficient matrix. Smith, Bjørstad, and Gropp [96] present a nice overview of the techniques used for overlapping and nonoverlapping methods that are applicable in the efficient solution methods for PDE problems on parallel machines that are readily transferable to RBF-PDE methods.

The set up cost of our preconditioner is shown in section 4.2. For any fixed value of

m, the B_i is the $m \times N$ matrix formed by selecting m rows of A from the index set $\mathcal{S}_i$ in (4.6). With DDM, the cost of constructing the preconditioners for each subdomain Ω_k containing $N_k \ll N$ centers is cheaper than finding the preconditioners for the original problem in which all N centers are used. The global storage requirements remain the same. Since the search of a local neighborhood is performed only within each subdomain, i.e. a smaller set of centers, the search time is reduced. The set up for the construction of each $(B_i)_k$ is less than that of the global B_i's. In each subdomain, the set up cost now becomes $\mathcal{O}(m^2 N_k^2 + m^3 N_k)$ flops.

With DDM, both the original and preconditioned systems on the subdomains are better conditioned when we are dealing with $N_k < N$ centers instead of N centers. Furthermore, combining our preconditioning technique with DDM will permit the use of a larger shape parameter by increasing the m/N ratio on each subdomain keeping the number of flops within an economical limit.

We acknowledge that our preconditioning scheme requires more development, but it still performs very efficiently with the caveat that we restrict the condition number of global coefficient matrix A by using safe values of $c \sim h_{\min}$. Then the GMRES iteration scheme converges within $\mathcal{O}(10)$ iterations, see section 4.3.3. The current preconditioner is two to three orders more efficient than global Gaussian elimination methods, and GMRES is N times more efficient than G-GE as N increases.

For cases where the condition number of A is about $\mathcal{O}(10^{10})$ to $\mathcal{O}(10^{13})$ and $1089 \leq N \leq 1681$, our preconditioning scheme only requires approximately 25% of the flops of the G-GE method. For highly ill-conditioned problems, see section 4.3.4, the preconditioned GRMES is able to converge for A having a condition number as high as $\mathcal{O}(10^{15})$, but GMRES now requires about 80% of the number of flops compared to G-GE.

In the next investigation, we will combine overlapping domain decomposition methods (DDM) with the least-squares ACBF method. We obtained considerable efficiency in our scheme by using the fast matrix-vector multiplication scheme of [7, 8]. However, we intend to obtain even better efficiencies when we implement the fast multipole expansion method of Beatson and Newsam[13] and combine our ACBF with domain decomposition similar to work of Beatson, Light, and Billings [12]. Another

promising approach would be to combine the ideas presented by Bozzini et al. [15]; by constraining the ACBFs to have higher order vanishing moments, the ACBFs decay very rapidly obviating the need to include the majority of the column elements of A. Such a combination is especially attractive for large-scale problems of engineering interest for the following reasons:

- No labor intensive mesh generation is required.
- The RBF-PDE collocation method is very simple to implement.
- The rate of convergence of MQ-RBF methods is exponential compared to the linear or quadratic convergence rates of the traditional methods.
- We will optimize the flop rates for the GMRES method using similar fast methods as Beatson and co-workers.
- As the number of subdomain increase, the partitioned matrix A_k associated with each subdomain becomes better conditioned, permitting larger c values with the accompanying faster convergence and coarser discretization requirements.
- With the work load distributed among many processors, the turn-around time to finish a run diminishes, permitting a larger number of parametric studies to be completed.

This DDM-ACBF approach can be very relevant when combined with Fornberg's method. We intend to provide persons who are interested in using asymmetric RBF-PDE methods various options to solve their problems. It is hoped that our work will dispel the perception that only finite difference, element, and volume methods are computationally efficient methods to solve PDE problems in view of their slow convergence rates.

4.4 Coupling with DDMs

Suppose we partition the original domain Ω into two subdomains Ω_1 and Ω_2 such that $\Omega_1 \cap \Omega_2 \neq \emptyset$ and $\Omega_1 \cup \Omega_2 = \Omega$. We employ the same notation as in Smith, Bjørstad and

Gropp [96]. The domains Ω, Ω_1, and Ω_2 are open and do not include their boundaries; let $\partial\Omega$, $\partial\Omega_1$, and $\partial\Omega_2$ denotes their respective boundaries. The artificial boundary Γ_i of Ω_i for $i = 1, 2$ is defined to be the part of $\partial\Omega_i$, the boundary of Ω_i, that is also interior to Ω. The rest of the boundary $\partial\Omega_i \backslash \Gamma_i$ is referred to as the natural boundary. The *classical alternating Schwarz algorithm* is given by

$$\begin{cases} \mathcal{L}U_1^n &= f(x) & \text{in } \Omega_1, \\ \mathcal{B}U_1^n &= g(x) & \text{on } \partial\Omega_1 \backslash \Gamma_1, \\ U_1^n &= U_2^{n-1}|_{\Gamma_1} & \text{on } \Gamma_1, \end{cases} \qquad (4.16)$$

$$\begin{cases} \mathcal{L}U_2^n &= f(x) & \text{in } \Omega_2, \\ \mathcal{B}U_2^n &= g(x) & \text{on } \partial\Omega_2 \backslash \Gamma_2, \\ U_2^n &= U_1^n|_{\Gamma_2} & \text{on } \Gamma_2, \end{cases}$$

for $n = 1, 2, \ldots$, where U_i^n is the numerical solution on the closure of the subdomain $\overline{\Omega}_i = \Omega_i \cup \partial\Omega_i$, $i = 1, 2$ after n iterations. For each subdomain, the same PDE is solved in the interior Ω_i and on the natural boundary $\partial\Omega_i \backslash \Gamma_i$. A Dirichlet condition is imposed on the artificial boundary Γ_i so that numerical solution on subdomain Ω_i matches the newest approximation on Γ_i. The notation $U_i^n|_D$ denotes the values of U_i^n restricted to the domain D.

Throughout the section, we use the notation U_i^n to indicate the approximate solution on subdomain Ω_i after n DDM iterations. Let α_i^n to be the RBF coefficients corresponding to U_i^n. vector α_i^n. Lastly, the symbol Δ is reserved to the operator that computes the maximum difference between two vectors: $\Delta U_i^n := \|U_i^n - U_i^{n-1}\|_\infty$, and $\Delta\alpha_i^n := \|\alpha_i^n - \alpha_i^{n-1}\|_\infty$. For simplicity, the superscript n will be omitted if it is clear from the context.

4.4.1 Using GMRES with DDM

The idea of the classical alternating Schwarz algorithm was first published by Schwarz [94] in 1870. The application of DDM includes for finite difference methods, finite element methods, finite volume methods, and spectral methods on linear or nonlinear

problems, and other applications. Readers should refer to the proceedings of the annual domain decomposition meetings for the most recent developments. Recently, as the interest in RBFs became more attractive, researchers began to combine the DDM with RBF methods. We combine the RBF methods and the ACBF preconditioning technique with DDM, denoted by ACBF-DDM in short. For an $N \times N$ matrix, the performance of the ACBF preconditioner depends on the condition of the matrix and the ratio between the size of support of the ACBF and of the matrix, m/N. For any fixed number of data points N in Ω, each subdomain will have a smaller number of data points $N_i < N$. Hence, as the ratio m/N_i increases, so does the performance of the ACBF preconditioner. Combining the DDM with the ACBF preconditioner is therefore expected to outperform the DDM using direct solver methods. In section 4.4.1, we studied different implementations of the ACBF-DDM scheme. Numerical results are reported in section 4.4.2.

For testing purposes, we set up a boundary value problem for the Poisson equation (4.14). We assume Dirichlet boundary conditions on all sides of the unit square whose values are given by (4.15). In this section, we apply the classical alternating Schwarz algorithm on overlapping matching subdomains using GMRES with the ACBF preconditioner.

The original domain Ω is spanned with 33×33 equally spaced knots that are, and other divided into four subdomains each of which contains 18×18 equally spaced knots with a uniform knot spacing, $h = 1/32$. The intersection region between adjacent subdomains will contain varying numbers of overlapping knots denoted by w (in knots). The four subdomains are given by

$$\Omega_1 = \{\, (x,y) \,:\, x \in (0,\ 1/2+wh),\ y \in (0,\ 1/2+wh) \,\},$$
$$\Omega_2 = \{\, (x,y) \,:\, x \in (1/2-wh,\ 1),\ y \in (0,\ 1/2+wh) \,\},$$
$$\Omega_3 = \{\, (x,y) \,:\, x \in (0,\ 1/2+wh),\ y \in (1/2-wh,\ 1) \,\},$$
$$\Omega_4 = \{\, (x,y) \,:\, x \in (1/2-wh,\ 1),\ y \in (1/2-wh,\ 1) \,\}.$$

In figure 4.10, we show the knot distributions of Ω_1 and Ω_4 with $w = 1$. The global matrix without preconditioning has a condition number of $\mathcal{O}(10^6)$. Note that the RBF-PDE matrix is not symmetric, and the condition number is not as informative

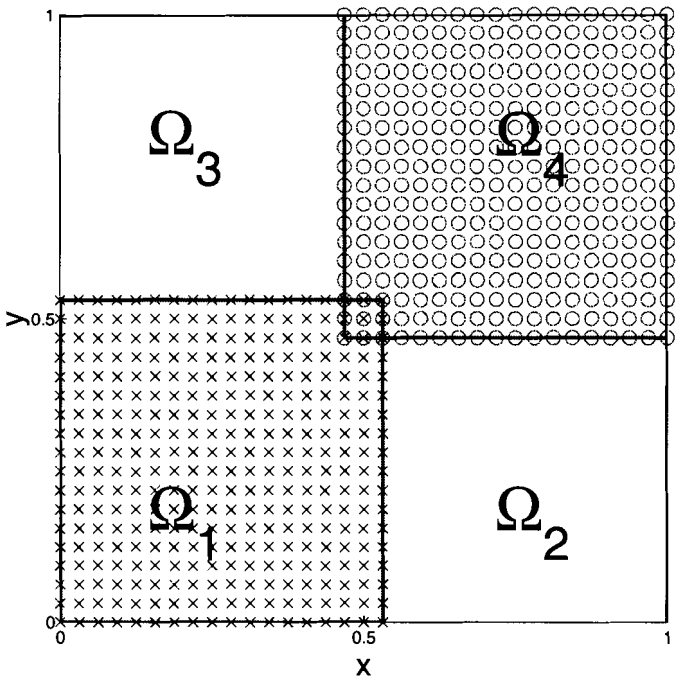

Figure 4.10: Overlapping grid of Ω_1 and Ω_4.

as for the symmetric case. More importantly, using the ACBF preconditioning technique reduces the number of GMRES iterations required to converge from hundreds to tens of iterations.

We use the MQ RBFs with a constant shape parameter $c = 1/33$. The ACBF preconditioners on each subdomain are constructed using a support size of $m = 10 + 9^{\ddagger}$. The condition numbers of the subdomains' RBF-PDE matrices are of the order $\mathcal{O}(10^5)$, and that of preconditioned matrices is 12. Efficiency is measured by the total number of GMRES iterations instead of the number of DDM iterations. The relation between τ_{DDM} and τ_{GMRES} is being studied. Furthermore, two update strategies of the function values on the artificial boundary are also studied experimentally.

[‡]10 local support and 9 special support

Algorithm 4 First ACBF-DDM algorithm

Initialization:

Tolerance: τ

DDM iteration:

While $\Delta\alpha_i > \tau$ on some subdomains Ω_i

For each subdomain Ω_i

Solve PDE system (4.17) with GMRES

Update $U_j|_{\Gamma_j}$ on neighboring artificial boundary Γ_j

EndFor

EndWhile

Relationship between τ_{DDM} and τ_{GMRES}

The numerical implementation of the classical alternating Schwarz algorithm involves a user defined DDM tolerance, τ_{DDM}. The DDM iteration stops when the maximum changes of the numerical approximations on the artificial boundary ΔU_i^n is less than τ_{DDM} on all subdomains Ω_i. When GMRES is used to solve each linear system in (4.17) instead of the direct method, a GMRES tolerance τ_{GMRES} is necessary to control the stopping criteria of GMRES.

One could obtain a simple algorithm by solving the linear systems in (4.17) using the preconditioned GMRES iteration scheme. In order to accelerate convergence, the latest approximation of the unknown RBF coefficients α_i on Ω_i should be used as the initial guess of the current GMRES iteration on Ω_i.

Lemma 4.4.1 *For any linear system $\Phi\alpha = b$, let $\|\cdot\|$ denote an arbitrary vector norm, κ denote the condition number of Φ, and $\tau > 0$. If $\|\delta\alpha\| \leq \tau$, then $\|\delta b\| \leq \nu\,\kappa\,\tau$, for some $\alpha > 0$.*

Proof: By definition,

$$\kappa = \sup_{\delta\alpha} \frac{\|\delta b\|}{\|\delta\alpha\|} \Big/ \frac{\|b\|}{\|\alpha\|} .$$

The lemma follows by letting $\nu = \|b\|/\|\alpha\|$ and replacing sup by an inequality. ∎

Although lemma 4.4.1 provides a relationship between τ_{DDM} and τ_{GMRES}, information about the exact solution is necessary to evaluate the constant $\rho = \|b\|/\|\alpha\|$. Moreover, lemma 4.4.1 suggests that when the change in $\|b\|$ is sufficiently small such that the initial guess of GMRES is within the desired tolerance, GMRES will return the initial guess without iterating. Therefore, the DDM iteration will have two equivalent consecutive approximations and will stop iterating. Thus, the role of τ_{DDM} is less significant.

We solve the test problem with different values of $\tau = \tau_{\text{DDM}} = \tau_{\text{GMRES}}$. Our first stopping criterion depends on the values of RBF coefficients instead of the approximated value on the artificial boundary. Iteration will stop if $\Delta\alpha_i^n < \tau$. In other words, the DDM iteration will stop when all RBF coefficients fail to be modified by an amount τ. The maximum error on the artificial boundary among all subdomains is shown in figure 4.11 with different values of τ as a function of total GMRES iterations required for convergence. For completeness, the number of DDM iterations required to converge are 3, 7, 10, 18, and 27 for tolerances 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, and 10^{-7}, respectively.

As one would expect, using a smaller tolerance would achieve more accurate solutions after the DDM iteration cycle converges. On the other hand, using a smaller tolerance does require more GMRES iterations or more computational effort to achieve a certain prescribed accuracy. The question is to find the best outcome with the least amount of computational effort. This reflects the fact that more GMRES iterations are necessary to achieve a smaller tolerance. This observation leads us to develop the algorithms in the section 4.4.1.

Algorithm-A and algorithm-B

From section 4.4.1, we see that an efficient algorithm should start with a relatively large value of τ_{DDM}. As the approximation converges, a smaller tolerance is desired so that the errors continue to be reduced.

Our first algorithm, *algorithm-A*, is outlined in algorithm 5. Algorithm-A recursively stores the newest approximation on all artificial boundaries before calling

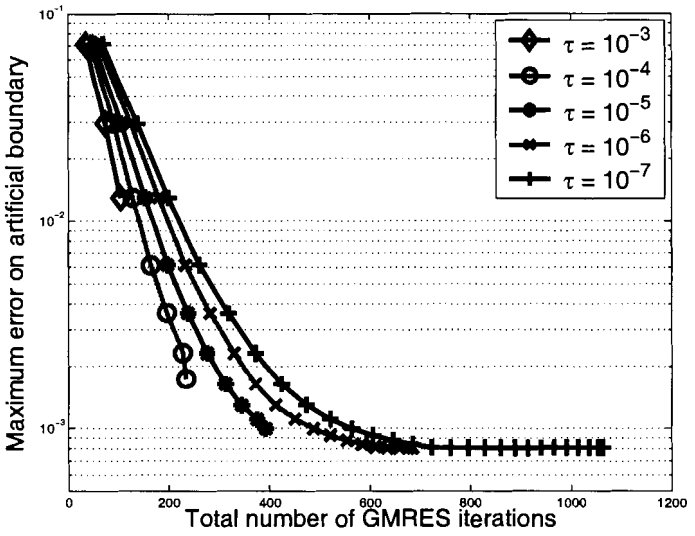

Figure 4.11: Error behavior on the artificial boundary with different τ_{GMRES}.

algorithm 4 with successively smaller tolerances. Since the function values on an artificial boundary are updated after each completion of algorithm 4, the DDM iteration cannot terminate within the call of algorithm 4.

We solve the test problem starting with $\tau_{\text{GMRES}} = 10^{-2}$. We use a reducing factor of $\rho = 10$, and $\min \tau_{\text{GMRES}} = 10^{-12}$. The maximum differences of the most recent two consecutive function values on all artificial boundaries, $\|W_i - U_i|_{\Gamma_i}\|$, is used as the *measure of convergence.*

In figure 4.12, this measure of convergence is shown against the total number of flops after each inner for-loop in algorithm 5.

Within each call of algorithm 4, the new approximation on the artificial boundaries is converging to another fixed point that differs from the stored value W_i. The measure of convergence of algorithm-A resembles an approximate step-function whose jumps occur after τ_{GMRES} is reduced in value.

To smooth out the curve $\|W_i - U_i|_{\Gamma_i}\|$, we suggest *algorithm-B* in which the function values on an artificial boundary are updated immediately within algorithm 4 if the GMRES iterates on some subdomains, see algorithm 6. Furthermore, an if-condition is added to allow early termination. This modification permits the DDM iteration

Algorithm 5 Algorithm-A

Initialization:

DDM tolerance: τ_{DDM}

Initial GMRES tolerance: τ_{GMRES}

Reducing factor for τ_{GMRES}: $\rho > 1$

Iteration:

While $\|W_i - U_i|_{\Gamma_i}\|_\infty > \tau_{\text{DDM}}$ on some subdomains Ω_i

Store approximations on artificial boundaries: $W_i \leftarrow U_i|_{\Gamma_i}$

Call algorithm 4 with $\tau = \tau_{\text{GMRES}}$

$\tau_{\text{GMRES}} \leftarrow \tau_{\text{GMRES}}/\rho$

EndWhile

to terminate within algorithm 4 and hence avoid unnecessary GMRES iterations. Although the measure of convergence of algorithm-B still contains jumps, it does behave more similarly to the direct solver version of DDM, see figure 4.12. It is clear from figure 4.12 that algorithm-B will allow the DDM iteration to terminate earlier for any give τ_{DDM}. As an example, if $\tau_{\text{DDM}} = 10^{-4}$, 630 and 481 GMRES iterations are need for algorithm-A and algorithm-B to converge, respectively. Their corresponding maximum errors are 2.441×10^{-3} and 2.668×10^{-3}. If $\tau_{\text{DDM}} = 10^{-6}$, these values are 1144 and 2.427664×10^{-3} for algorithm-A, and 1022 and 2.427552×10^{-3} for algorithm-B, respectively.

Finally, we want to mention that for both algorithm-A and algorithm-B, a minimum value of τ_{GMRES} and a maximum number of iterations should be imposed to avoid an infinite loop. For simplicity, we skip the details in the *initializations*. Some variables need to be initialized correctly for the *iteration* to start. Algorithm-B will be used for all the tests in the remainder of the chapter.

Width of the overlapping region

It is commonly known that increasing the width w of the overlapping region will speed up the DDM convergence rate. In this section, we will verify this behavior with the

Algorithm 6 Algorithm-B

Initialization:

DDM tolerance: τ_{DDM}

Initial GMRES tolerance: τ_{GMRES}

Reducing factor for τ_{GMRES}: $\rho > 1$

Iteration:

While $\|W_i - U_i|_{\Gamma_i}\|_\infty > \tau_{\text{DDM}}$ on some subdomains Ω_i

While $\Delta\alpha_i > \tau_{\text{GMRES}}$ on some subdomains Ω_i

For each subdomain Ω_i

Solve the PDE system (4.17) with GMRES

Update $U_j|_{\Gamma_j}$ on neighboring artificial boundary Γ_j

EndFor

If GMRES iterates *on some subdomains*

Store approximations on artificial boundaries: $W_i \leftarrow U_i|_{\Gamma_i}$

If $\|W_i - U_i|_{\Gamma_i}\| < \tau_{\text{DDM}}$ on all subdomains Ω_i

Break (DDM converged)

EndIf

Else (i.e., GMRES returns initial guess *on all subdomains*)

Break inner while-loop

EndIf

EndWhile

$\tau_{\text{GMRES}} \leftarrow \tau_{\text{GMRES}}/\rho$

EndWhile

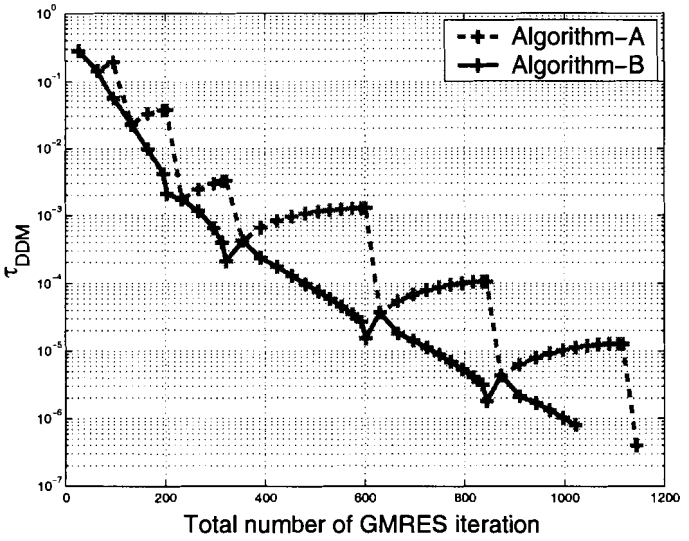

Figure 4.12: Measure of convergence as a function of total flops.

GMRES version of DDM. We solve the test problem with 1 to 7 knots overlapping in each direction, and with $\tau_{\text{DDM}} = 10^{-4}$. The total number of GMRES iterations required for convergence, total number of DDM iterations, and the smallest τ_{GMRES} used in algorithm-B for different overlapping widths are listed in table 4.2.

In terms of the number of DDM iterations, faster convergence is achieved with wider overlap. Also, we can see a trend of faster convergence in terms of the number of GMRES iterations required, see figure 4.13. Increasing the width of the overlap does not necessarily accelerate the convergence. Convergence also depends upon the smallest τ_{GMRES} used in algorithm-B. In general, we see that if algorithm-B terminates with a larger τ_{GMRES}, the total effort will be less. Furthermore, we also see an increase in accuracy as the overlapping width increases; from 2 to 7 overlapping knots per row or column. The maximum errors are 2.668×10^{-3}, 2.421×10^{-3}, 2.345×10^{-3}, 2.223×10^{-3}, 2.149×10^{-3}, 2.062×10^{-3}, and 2.022×10^{-3}, respectively.

4.4.2 Numerical examples

In all the numerical examples in this section, we use algorithm-B with $\tau_{\text{GMRES}} = 10^{-2}$ and a GMRES tolerance reducing factor $\rho = 10$. We continue to study (4.14) as

Width of overlapping, w	1	2	3	4	5	6	7
Total GMRES iterations	481	268	254	331	283	191	196
Total DDM iteration	17	10	10	9	9	5	5
Smallest τ_{GMRES} used	10^{-6}	10^{-5}	10^{-5}	10^{-6}	10^{-6}	10^{-5}	10^{-5}

Table 4.2: Convergence behavior with different overlapping widths with $\rho = 10$.

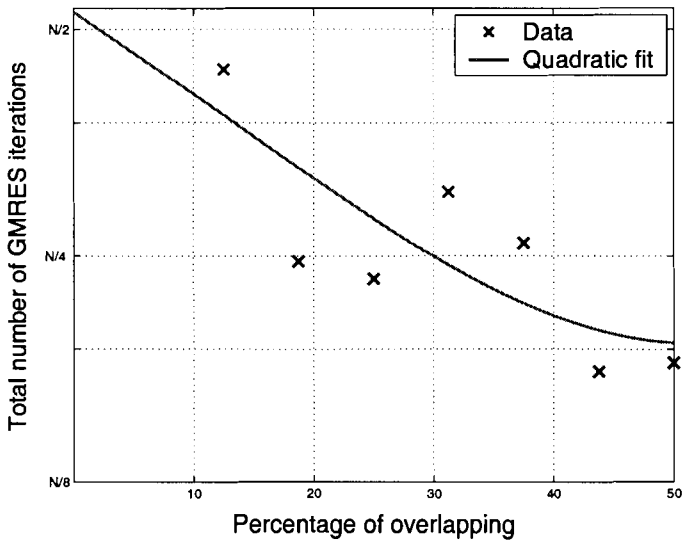

Figure 4.13: Total effort as a function of overlapping width, where $N = 1089$.

an example of DDM with matching nodes. Next we consider a PDE with variable coefficients. Then we show the results of another example used by Smith et al. [96] with the basis function r^5 for the non-matching case.

Example 4.4.1 *The MQ shape parameter for each run is* $c^2 = 1/N_{total}$ *where* N_{total} *denotes the number of nodes in the union of all subdomains' nodes. We concentrate* $N_{total} = 73 \times 73 = 5329$ *and* $N_{total} = 109 \times 109 = 11881$. *We partition the domain* Ω *into* 4, 9, 16, *and* 36 *subdomains. The ACBF preconditioners are constructed using a support size of* $m = 10 + 9$ *as in the last section. The width of overlapping is around* 10% *to the width of the subdomains. One disadvantage of the matching nodes method is that we are not able to pick arbitrarily the width of overlap.*

We terminate algorithm-B when $\tau_{\text{DDM}} = 10^{-3}$ *and* $\tau_{\text{DDM}} = 10^{-4}$. *For both cases, the number of iterations required for algorithm-B to converge are shown in table 4.3 and*

table 4.4, respectively. The average *number of (GMRES) iterations is defined to be the average number of GMRES iterations required per DDM iteration per subdomain whose value is also expressed as a factor of* $\overline{N_i}$. *The term* $\overline{N_i}$ *is the average number of nodes in all subdomain, and is given by*

$$\overline{N_i} = \frac{1}{(No.\ of\ subdomains)} \cdot \sum_i N_i,$$

which gets larger when more subdomains or wider overlappings are used. When more subdomains are used, the ratio m/N_i *increases as the dimension of each matrix decreases, and we expect better performance of the ACBF preconditioning technique on each subdomain. The average number of GMRES iterations required for convergence is therefore reduced. In general, using more subdomains requires more iterations for both the DDM and the GMRES cycles to achieve convergence. However, note that the matrix-vector product becomes cheaper as the size of the subdomain is reduced; the total number of GMRES is not an absolute measure of the cost of algorithm-B. We summarize by figure 4.14: the ratio of estimate flops required for algorithm-B to converge over the flops required by the direct method. The overall cost estimate of algorithm-B consists of the set up of ACBF preconditioners:* $m \cdot \sum N_i^2 + m^3 \cdot \sum N_i$, *plus the cost estimate of the matrix-vector multiplications in the GMRES iterations. The cost of direct method is simply chosen to be* N^3. *Using* $m = 50 + 9$, *Ling and Kansa [?] obtained an estimated flop ratio* $35.47N^{-0.89}$ *(that becomes* 0.171 *and* 0.0084 *for* $N_{total} = 5329$ *and* $N_{total} = 11881$, *respectively) when applying the ACBF preconditioner directly for the global RBF-PDE matrix of (4.14). Figure 4.14 shows that our ACBF-DDM method is even more attractive than using ACBF without DDM in all test cases. One could improve the efficiency of algorithm-B by using more subdomains. Hence, this trend appears to be very attractive for massively parallel computers.*

Example 4.4.2 *With a similar set up as in example 4.4.1, we consider a PDE with variable coefficients,*

$$\frac{\partial}{\partial x}\left[a(x,y)\frac{\partial u}{\partial x}\right] + \frac{\partial}{\partial y}\left[b(x,y)\frac{\partial u}{\partial y}\right] = f(x,y), \tag{4.17}$$

N_{total}	*Number of subdomains*	*Number of iterations*		
		DDM	*GMRES*	*Average*
73×73	$(2,2)$	*7*	*490*	$17.5 = .0095\overline{N_i}$
73×73	$(3,3)$	*16*	*1118*	$7.76 = .0092\overline{N_i}$
73×73	$(4,4)$	*27*	*2407*	$5.57 = .0115\overline{N_i}$
73×73	$(6,6)$	*36*	*4564*	$3.52 = .0132\overline{N_i}$
109×109	$(2,2)$	*8*	*991*	$31.0 = .0076\overline{N_i}$
109×109	$(3,3)$	*13*	*1601*	$13.7 = .0068\overline{N_i}$
109×109	$(4,4)$	*27*	*4930*	$11.4 = .0099\overline{N_i}$
109×109	$(6,6)$	*37*	*8116*	$6.09 = .0122\overline{N_i}$

Table 4.3: Number of iterations required by solving (4.14) using algorithm-B with $\tau_{\text{DDM}} = 10^{-3}$ *on large scale problems.*

N_{total}	*Number of subdomains*	*Number of iterations*		
		DDM	*GMRES*	*Average*
73×73	$(2,2)$	*8*	*530*	$16.6 = .0090\overline{N_i}$
73×73	$(3,3)$	*26*	*1808*	$7.73 = .0092\overline{N_i}$
73×73	$(4,4)$	*43*	*4123*	$5.99 = .0124\overline{N_i}$
73×73	$(6,6)$	*59*	*8379*	$3.94 = .0148\overline{N_i}$
109×109	$(2,2)$	*10*	*1242*	$31.1 = .0076\overline{N_i}$
109×109	$(3,3)$	*22*	*2854*	$14.4 = .0071\overline{N_i}$
109×109	$(4,4)$	*45*	*7518*	$10.4 = .0090\overline{N_i}$
109×109	$(6,6)$	*47*	*11082*	$6.55 = .0131\overline{N_i}$

Table 4.4: Number of iterations obtained by solving (4.14) using algorithm-B with $\tau_{\text{DDM}} = 10^{-4}$ *on large scale problems.*

for $(x,y) \in [0,1]^2$, *where*

$$a(x,y) = 2 - x^2 - y^2, \qquad b(x,y) = e^{x-y},$$

and

$$f(x,y) = -xe^{x-y}(1-x)(3-2y) + 2y(1-y)(3x^2 + y^2 - x - 2).$$

The exact solution to (4.17) is given by

$$u(x,y) = xy(1-x)(1-y).$$

We report the convergence results in table 4.5 for $\tau_{\text{DDM}} = 10^{-4}$.

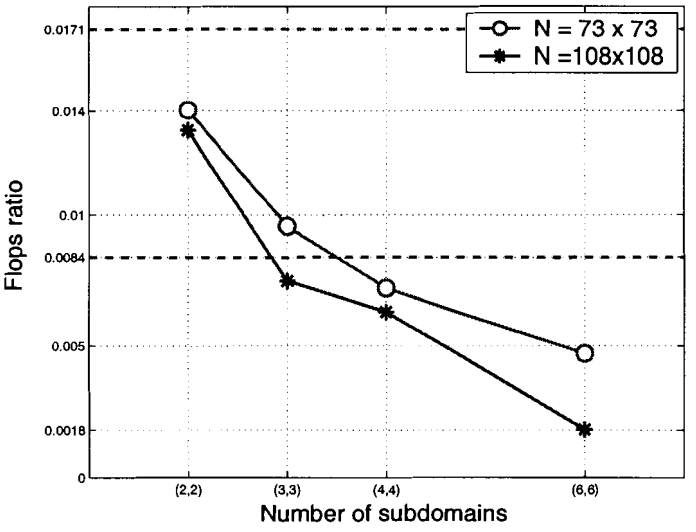

Figure 4.14: The ratio of estimated flops required for algorithm-B to converge over the flops required by the direct method.

Applying algorithm-B to a variable coefficients problem does not seems to affect the convergence behavior in terms of the average GMRES iterations. For the case of $(6, 6)$ *subdomains, we see a relatively larger number of DDM and GMRES iterations are required for convergence compared to the first example. This may due to the inefficient data communication between subdomains when too many subdomains are employed.*

Example 4.4.3 *In the last example, we consider the Poisson equation,*

$$\begin{aligned} \triangle u &= -xe^y \quad in\ \Omega, \\ u &= -xe^y \quad on\ \partial\Omega, \end{aligned} \tag{4.18}$$

where Ω *is a nontrivial domain: a union of square and a circle centered at* $(2, 2)$*, see figure 4.15 (LEFT). We apply algorithm-B on two overlapping subdomains with non-matching nodes. Collocation points are placed irregularly throughout each subdomain. An example of collocation points placement is shown in figure 4.15 (RIGHT). In general, the number of iterations required for convergence of the classical alternating Schwarz methods depends only on the geometric overlap, not the number of nodes or mesh refinement. We show that algorithm-B also shares a similar property.*

N_{total}	*Number of subdomains*	*Number of iterations*		
		DDM	*GMRES*	*Average*
73×73	$(2,2)$	*12*	*858*	$17.9 = .0097\overline{N_i}$
73×73	$(3,3)$	*25*	*2157*	$9.59 = .0104\overline{N_i}$
73×73	$(4,4)$	*34*	*4230*	$7.78 = .0141\overline{N_i}$
73×73	$(6,6)$	*65*	*11233*	$4.80 = .0180\overline{N_i}$
109×109	$(2,2)$	*9*	*1154*	$32.1 = .0078\overline{N_i}$
109×109	$(3,3)$	*24*	*3724*	$17.2 = .0085\overline{N_i}$
109×109	$(4,4)$	*43*	*8306*	$12.1 = .0105\overline{N_i}$
109×109	$(6,6)$	*63*	*19826*	$8.74 = .0152\overline{N_i}$

Table 4.5: Number of iterations obtained by solving (4.17) using algorithm-B with $\tau_{\text{DDM}} = 10^{-4}$ *on large scale problems.*

Instead of the MQ RBF, we use the basis function $\phi = r^5$ *to solve the problem; the same problem is studied by Li and Hon [75]. We use* 50 *local points, and* 13 *and* 9 *special points on the square and circle, respectively, to construct the ACBF preconditioner. We pick* $\tau_{\text{DDM}} = 10^{-4}$. *Since the* r^5 *basis function does not have a parameter to counter the ill conditioning effect with the increasing number of nodes, the condition of the RBF-PDE matrix increases rapidly with* N.

On two subdomains, our method can efficiently *handle the problem with around* 1000 *nodes on each subdomain despite the ill conditioning problem. With* 146 *nodes on the square and* 144 *nodes on the circle, the condition number reduces from* $\mathcal{O}(10^7)$ *to the order of* 1 *on both subdomains. With* 1099 *nodes on the square and* 1078 *nodes on the circle, the condition number increases to* $\mathcal{O}(10^{10})$ *and the preconditioner only helps reducing it to the order of* 100. *Table 4.6 shows that the number of DDM iterations performed within algorithm-B is roughly constant. However, the number of GMRES iterations required for convergence increases as* N, *or the condition number, increases. On average, we need* 6 *to* 31 *GMRES iterations per subdomain per DDM iteration.*

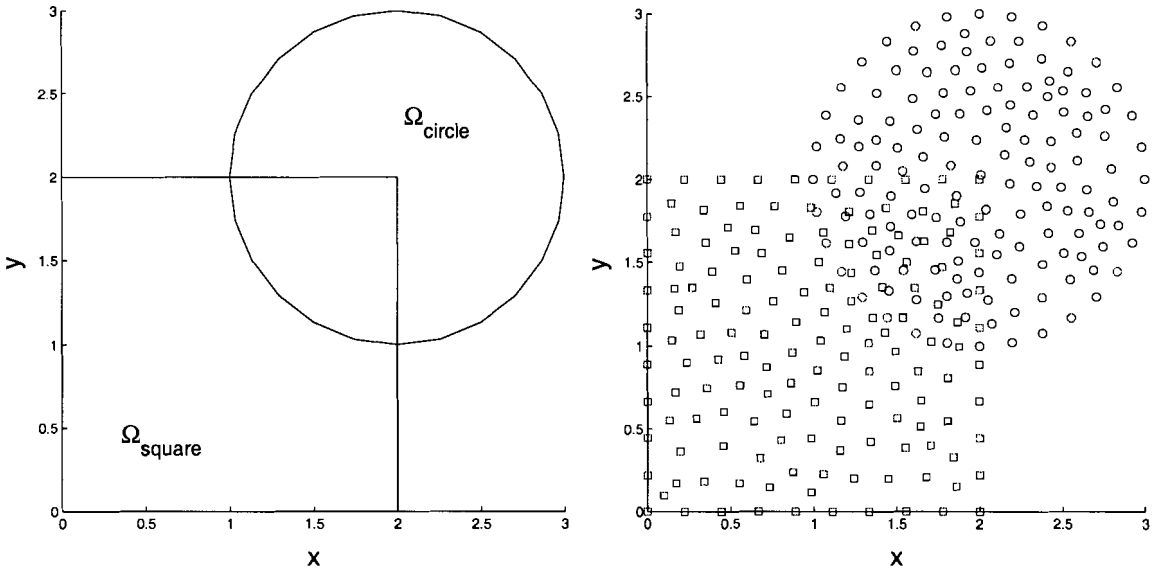

Figure 4.15: Random placement of collocation points on two subdomains that is a union of a square and a circle for the Poisson equation (4.18).

4.4.3 Discussion

We experimented with different implementations of the coupling of the approximate cardinal basis functions (ACBFs) preconditioning technique and the classic alternating Schwarz algorithm [96, 94]. The truly meshfree radial basis functions (RBFs) collocation methods allow the Schwarz algorithm to be applied on both matching and non-matching nodes without any modification.

Using DDM allows the rank of each subdomain matrix to be reduced and therefore allows the ACBFs preconditioner to work more efficiently.

Our algorithm not only helps reducing the overall computation effort of solving RBF-PDE systems, but we show it is more efficient than solving the global RBF-PDE problem with ACBFs preconditioning alone. In particular, the efficiency can be improved by using more subdomains.

Because RBF methods possess higher order convergence rates than the standard methods, they require orders of magnitude less discretization. Also, as the spatial dimension increases, the connectivity requirements increase the bandwidth of the sparse

No. of nodes		*Max. error*		*No. of iteration*		
Square	*Circle*	*Square*	*Circle*	*DDM*	*GMRES*	*Average*
146	*144*	4.7×10^{-2}	5.9×10^{-2}	*9*	*124*	$6.89 = .0475\overline{N_i}$
320	*354*	4.4×10^{-3}	7.2×10^{-3}	*11*	*389*	$17.7 = .0525\overline{N_i}$
497	*544*	3.6×10^{-3}	4.3×10^{-3}	*11*	*721*	$32.8 = .0630\overline{N_i}$
1099	*1078*	1.5×10^{-3}	1.4×10^{-3}	*8*	*1281*	$80.1 = .0736\overline{N_i}$

Table 4.6: Number of iterations and maximum error obtained by algorithm-B with $\phi = r^5$.

system of equations using traditional methods. Note that special preconditioning methods and domain decomposition are also used to solve large PDE systems.

To state categorically that the traditional methods giving rise to sparse systems will always be more efficient than RBF methods that give rise to either broad-banded sparse or full systems remains an open question that is problem dependent. With the combination of the ACBF preconditioner scheme on multiple overlapping subdomains, we have demonstrated that we can achieve a significant reduction in the total number of flops to solve a very large system of equations, especially when implemented on massively parallel computers. We believe additional reductions in the total number of flops required to achieve convergence can be achieved by combining the scheme presented here with fast multipole expansions, control of higher moments of the ACBF expansions, and a better understanding of conditions under which RBF expansions can be truncated as discussed by Hardy[53]. This section has not addressed every detail by which RBF-PDE solutions can be made very competitive, but has shown that the hybrid scheme combining the ACBF preconditioners with DDM has demonstrated that criticism of RBF methods as being too ill conditioned and too costly to implement is no longer valid.

4.5 Chapter summary

Although meshless radial basis function (RBF) methods applied to partial differential equations (PDEs) are not only simple to implement and enjoy exponential convergence rates as compared to standard mesh-based schemes, the system of equations required

to find the expansion coefficients are typically badly conditioned and expensive using the global Gaussian elimination (G-GE) method requiring $\mathcal{O}(N^3)$ flops.

We present a simple preconditioning scheme that is based upon constructing least-squares approximate cardinal basis functions (ACBFs) from linear combinations of the RBF-PDE matrix elements. The ACBFs transforms a badly conditioned linear system into one that is very well conditioned, allowing us to solve for the expansion coefficients iteratively so we can reconstruct the unknown solution everywhere on the domain. Our preconditioner requires $\mathcal{O}(mN^2)$ flops to set up, and $\mathcal{O}(mN)$ storage locations where m is a user define parameter of order 10. For the 2D MQ-RBF with the shape parameter $c \sim 1/\sqrt{N}$, the number of iterations required for convergence is of order 10 for large values of N, making this a very attractive approach computationally.

As the shape parameter increases, our preconditioner will eventually be affected by the ill conditioning and round-off errors, and thus becomes less effective. We tested our preconditioners on increasingly larger c and N. A more stable construction scheme is available with a higher set up cost.

In this chapter, we also combine the RBF methods and the ACBF preconditioning technique with DDM, denoted by ACBF-DDM in short. For an $N \times N$ matrix, the performance of the ACBF preconditioner depends on the condition of the matrix and the ratio between the size of support of the ACBF and of the matrix, m/N. For any fixed number of data points N in Ω, each subdomain will have a smaller number of data points $N_i < N$. Hence, as the ratio m/N_i increases, so does the performance of the ACBF preconditioner. Combining the DDM with the ACBF preconditioner is therefore expected to outperform the DDM using direct solver methods. In section 4.4.1, we studied different implementations of the ACBF-DDM scheme. Numerical results are reported in section 4.4.2.

Chapter 5

Future Studies

Quasi-Interpolation

The study of quasi-interpolation could play an important role when coupled with the method of fundamental solutions (MFS). For an inhomogeneous differential equation $\mathcal{L}u = f$, our quasi-interpolation scheme allows us to rewrite f in terms of the dimension-splitting (DSMQ) basis. Before applying MFS, one needs to transform the inhomogeneous equation to a homogeneous one.

After we obtain the quasi-interpolant for the right hand function f, the problem of finding a particular solution is transformed into a series of subproblems: solving $\mathcal{L}u =$ DSMQ. This technique of finding a particular solution involves symbolic calculations which result in accurate and efficient algorithms. Similar technique is used by Golberg et al. [47] with Lagrange polynomials instead of DSMQ. They are able to find a particular solution for Poisson's equation or inhomogeneous Helmoholtz-type equation. We believe our quasi-interpolation formula could be used on similar problems for solving PDEs.

Boundary Layer Problems

We gave an example of solving interior layer problems with our RBF scheme; however, our method is not designed for such problem. Like most of the existing schemes, we see that our scheme fails to capture small layer when dealing with interior layer problems. One of our goals is to develop a new coordinate

condensing transform that allows us to handle interior layers as efficient as what we have now for the boundary layer problems. For example, the inverse of the SINE-transform could be a good choice. On the other hand, the exact location of the interior layer may not be trivial. A sophisticated adaptive scheme is required to capture an interior layer.

We would like to extend our scheme to two dimension. Two dimensional versions of the SINE-transform and indicator are necessary. In particular, extending the indicator, that used in our adaptive scheme which employs information from the third derivatives of the numerical approximation, is not trivial.

ACBF Preconditioners

For simplicity, assume our domain is $\Omega = [0,1]^2$ and a total number of N regularly spaced centers is used to discretize the domain. Let us consider one of the subdomains, Ω_k, used in the DDM. Suppose there are N_k centers in $\Omega_k = [a, a+L] \times [b, b+L]$, and assume that the shape parameter c is fixed. A simple coordinate transformation $(x,y) \mapsto \left(\frac{x-a}{L}, \frac{y-b}{L}\right)$ maps Ω_k to $\tilde{\Omega}_k = [0,1]^2$. For $\tilde{\Omega}_k$, the corresponding normalized shape parameter $\tilde{c} = c/L > c$. From the ratio of the area over the number of centers, we know $1/N = L^2/N_k$. Eliminating L, we can rewrite this relationship as

$$\tilde{c} \Big/ \left(\frac{1}{\sqrt{N_k}}\right) = c \Big/ \left(\frac{1}{\sqrt{N}}\right). \tag{5.1}$$

Suppose, for a given problem, one desires to use a shape parameter $c = j/\sqrt{N}$ for some $j > 1$. The corresponding normalized shape parameter for $\tilde{\Omega}_k$ is given by $\tilde{c} = j/\sqrt{N_k}$ by (5.1). We see that DDM helps solving the ill conditioning problem by reducing the rank of the matrices but not by reducing the shape parameter.

One not only needs to split the original data points into smaller sets as in the DDM, but the area/volume of the subdomains also needs to be roughly the same as the original domain in order to achieve benefits from both the reducing rank and the reducing shape parameter. A possible direction is to couple ACBF with

a multilevel (ML) scheme. The idea of ML allows one to deal with a smaller number of data points at a time. Most importantly, these data points are still distributed throughout the original domain and thus have a smaller normalized shape parameter. ML is not new to RBF. Floater and Iske [32] use a ML interpolation method for scattered data. Narcowich et al. [89] provide some theoretical results about the Floater-Iske method. Iske and Levesley [67] proposed an ML scheme that relies on an adaptive domain decomposition strategy. Chen et al. [23] apply an ML scheme to solve elliptic problems.

There is still much room for researches into the theories and applications of RBFs. Our final goal is to provide a numerically efficient and stable scheme to solve the RBF-PDE problem with a large shape parameter and a large number of unknowns.

Bibliography

[1] P. Alotto, A. Caiti, G, Molinari, and M. Repetto, A Multiquadrics-Based Algorithm for the Acceleration of Simulated Annealing Optimization Procedures, *IEEE Trans. on Magnetics*, **32**, (3), 1198-1201, (1996).

[2] U. M. Ascher, J. Christiansen, and R. D. Russell, A Collocation Solver for Mixed Order Systems of Boundary Value Problems, *Math. Comp.*, **33**, 659-679, (1979).

[3] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, New Jersey, (1988).

[4] K. Balakrishnan, R. Sureshkumar, and P. A. Ramachandran, An Operator Splitting-Radial Basis Function Method for the Solution of Transient Nonlinear Poisson Problems, *Comput. Math. Appl.*, **43**, 289-304, (2002).

[5] B. Baxter, The Asymptotic Cardinal Function of the Multiquadratic $\phi(R) = (R^2 + C^2)^{1/2}$ as $c \to \infty$, *Comput. Math. Appl.*, **24**, (12), 1-6, (1992).

[6] B. Baxter, Preconditioned Conjugate Gradients, Radial Basis Functions, and Toeplitz Matrices, *Comput. Math. Appl.*, **43**, (3-5), 305-318, (2002).

[7] R. K. Beatson, J. B. Cherrie, and C. T. Mouat, Fast Fitting of Radial Basis Functions: Methods Based on Preconditioned GMRES Iteration, *Adv. Comput. Math.*, **11**, 253-270, (1999).

[8] R. K. Beatson, J. B. Cherrie, and G. N. Newsam, Fast Evaluation of Radial Basis Functions: Methods for Generalized Multiquadrics in $\mathbb{R}^N$, *SIAM J. Sci. Comput.*, **23**, (5), 1549-1571, (2002).

[9] R. K. Beatson and L. Greengard, A Short Course on Fast Multipole Methods, in *Wavelets, Multilevel Methods and Elliptic PDEs*, (M. Ainsworth, J. Levesley, W. Light, and M. Marletta, Eds.), 1-37, Oxford University Press, New York, (1997).

[10] R. K. Beatson, and W. A. Light, Quasi-Interpolation by Thin-Plate Splines on a Square, *Constr. Approx.*, **9**, 407-433, (1993).

[11] R. K. Beatson, and W. A. Light, Fast Evaluation of Radial Basis Function: Methods for Two-Dimensional Polyharmonic Splines, *IMA J. Numer. Anal.*, **17**, 343-372, (1997).

[12] R. K. Beatson, W. A. Light, S. Billings, Fast Solution of the Radial Basis Function Interpolation Equations: Domain Decomposition Methods, *SIAM J. Sci. Comput.*, **22**, 1717-1740, (2000).

[13] R. K. Beatson, and G. N. Newsam, Fast Evaluation of Radial Basis Functions: I, *Comput. Math. Appl.*, **24**, (12), 7-19, (1992).

[14] R. K. Beatson and M. J. D. Powell, Univariate Multiquadric Approximation: Quasi-Interpolation to Scattered Data, *Constr. Approx..*, **8**, (3), 275-288, (1992).

[15] M. Bozzini, L. Lenarduzzi, and R. Schaback, Kernel B-Splines and Adaptive Interpolation, *Manuscript*, (2002).

[16] P. N. Brown and H. F. Walker, GMRES on (Nearly) Singular Systems, *SIAM J. Matrix Anal. Appl.*, **18**, 37-51, (1997).

[17] M. D. Buhmann, Multivariate Cardinal Interpolation with Radial-Basis Functions, *Constr. Approx.*, **6**, (3), 225-255, (1990).

[18] M. D. Buhmann, Multivariate Interpolation in Odd-Dimensional Euclidean Spaces Using Multiquadrics, *Constr. Approx.*, **6**, (1), 21-34, (1990).

[19] M. D. Buhmann, and C. A. Micchelli, Multiquadric Interpolation Improved, *Comput. Math. Appl.*, **24**, (12), 21-25, (1992).

[20] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Series of Computational Physics, Springer-Verlag, Heidelberg, Berlin, New York, (1988).

[21] R. E. Carlson, and T. A. Foley, The Parameter $\mathbb{R}^2$ in Multiquadric and Related Interpolations, *Comput. Math. Appl.*, **21**, 29-42, (1991).

[22] J. C. Carr, W. R. Fright, and R. K. Beatson, Surface Interpolation with Radial Basis Functions for Medical Imaging, *IEEE Trans. Medical Imaging*, **16**, 96-107, (February 1997).

[23] C. S. Chen, M. Ganesh, M. A. Golberg, and A. H. D. Cheng, Multilevel Compact Radial Functions Based Computational Schemes for Some Elliptic Problems, *Comput. Math. Appl.*, **43**, 359-378, (2002).

[24] A. H. D. Cheng, M. A. Golberg, E. J. Kansa, and T, Zammito, Exponential Convergence and H-C Multiquadric Collocation Method for Partial Differential Equations, *Num. Meth. for Partial Differential Equations*, (To Appear).

[25] H. Q. Dinh, G. Turk, and G. Slabaugh, Reconstructing Surfaces by Volumetric Regularization Using Radial Basis Functions, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**, (10), 1358-1371, (October 2002).

[26] H. Q. Dinh, G. Turk, and G. Slabaugh, Reconstructing Surfaces Using Anisotropic Basis Functions, *International Conference on Computer Vision, the Eighth IEEE International Conference on Computer Vision*, (2001).

[27] H. Q. Dinh and G. Turk, Graphics Reconstructing Surfaces by Volumetric Regularization, *Visualization, and Usability Tech Report*, GVU-00-26.

[28] M. R. Dubal, Domain Decomposition and Local Refinement for Multiquadric Approximations, *J. Appl. Sci. Comput.* , **1**, 146-171, (1994).

[29] E. A. Edirisinghe, J. Jiang, and C. Grecos, Object Boundary Padding Technique for Improving MPEG-4 Video Compression Efficiency, *IEEE Electronic Letters*, **35**, (17), 1453-1455, (1999).

[30] E. A. Edirisinghe, J. Jiang, and C. Grecos, A Shape Adaptive Padding Technique for MPEG-4, *IEEE Trans. on Consumer Electronics*, **46**, (3), 514-520, (2000).

[31] A. C. Faul, *Iterative Techniques for Radial Basis Function Interpolation*, Ph.D. Thesis, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, England, (2001).

[32] M. S. Floater, and A. Iske, Multistep, Scattered Data Interpolation Using Compactly Supported Radial Basis Functions, *J. Comput. Appl. Math.*, **73**, (5), 65-78, (1996).

[33] A. C. Faul, and M. J. D. Powell, Krylov Subspace Methods for Radial Basis Function Interpolation, *Numerical Analysis 1999* (Dundee), 115-141, Chapman & Hall/Crc Res. Notes Math., 420, Chapman & Hall/Crc, Boca Raton, Fl, (2000).

[34] G. E. Fasshauer, On the Numerical Solution of Differential Equations with Radial Basis Functions, *Boundary Element Technology XIII*, (C. S. Chen, C. A. Brebbia, and D. W. Pepper, Eds), Wit Press, 291-300, (1999).

[35] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa, Improved Multiquadric Method for Elliptic Partial Differential Equations Via PDE Collocation on the Boundary, *Comput. Math. Appl.*, **43**, (3-5), 439-455, (2002).

[36] G. Fix and G. Strang, A Fourier Analysis of the Finite Element Variational Method, in *Constructive Aspect of Functional Analysis*, pp. 796-830, Edizioni Cremonese, Rome, (1971).

[37] M. S. Floater, and A. Iske, Multistep Scattered Data Interpolation Using Compactly Supported Radial Basis Functions, *Comut. Math. Appl.*, **73**, 65-78, (1996).

[38] B. Fornberg and T. A. Driscoll, Interpolation in the Limit of Increasingly Flat Radial Basis Functions, *Comput. Math. Appl.*, **43**, (3-5), 413-421, (2002).

[39] B. Fornberg, T. A. Driscoll, G. Wright and R. Charles, Observations on the Behavior of Radial Basis Functions Near Boundaries, *Comput. Math. Appl.*, **43**, 473-490, (2002).

[40] B. Fornberg and G. Wright, Stable Computation of Multiquadric Interpolants for All Values of the Shape Parameter, *Adv. Comput. Math.*, (To Appear).

[41] B. Fornberg, G. Wright, and E. Larsson, Some Observations Regarding Interpolants in the Limit of Flat Radial Basis Functions, *Comput. Math. Appl.*, (Submitted, 2002).

[42] R. Franke, Scattered Data Interpolation: Tests of Some Methods, *Math. Comp.*, **38**, (157), 181-200, (1982).

[43] C. Franke, and R. Schaback, Solving Partial Differential Equations by Collocation Using Radial Basis Functions, *Appl. Math. Comput.*, *93*, (1), 73-82, (June 1998).

[44] C. Franke, and R. Schaback, Convergence Order Estimates of Meshless Collocation Methods Using Radial Basis Functions, *Adv. Comput. Math.*, *8*, (4), 381-399, (June 1998).

[45] E. C. Gartland, Graded-Mesh Difference Schemes for Singularly Perturbed Two-Point Boundary Value Problems, *Math. Comp.*, **51**, 631-657, (1995).

[46] M. A. Golberg and C.S. Chen, *Discrete Projection Methods for Integral Equations*, Computational Mechanics Publications, Southampton, Boston, (1997).

[47] M. A. Golberg, C. S. Chen, and A. H.-D. Cheng, Polynomial Particular Solutions for Certain Partial Differential Operators, *Numer. Methods Partial Differential Equations*, **19**, (1), 112-133, (2003).

[48] M. A. Golberg, C. S. Chen and S. R. Karur, Improve multiquadric interpolation for partial differential equations, *Eng. Anal. Bound. Elem.*, **18**, 9-17, (1996).

[49] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, third Ed., Johns Hopkins University Press, Baltimore and London, (1996).

[50] P. P. N. De Groen, The Nature of Resonance in a Singular Perturbation Problem of Turning Point Type, *SIAM J. Math. Anal.*, **11**, 1-22, (1980).

[51] H. M. Gutmann, A Radial Basis Function Method for Global Optimization, *J. Global Optimization*, **19**, 201-207, (2001).

[52] R. L. Hardy, Multiquadric Equations of Topography and Other Irregular Surfaces, *J. Geophys. Res.*, **176**, 1905-1915, (1971).

[53] R. L. Hardy, Theory and Applications of the Multiquadric-Biharmonic Method. 20 Years of Discovery 1968-1988, *Comput. Math. Appl.*, **10**, (8-9), 163-208, (1990).

[54] F. J. Hickernell, and Y. C. Hon, Radial Basis Function Approximation of the Surface Wind Field from Scattered Data, *Internat. J. Appl. Sci. Comput.*, **4**, 221-247, (1998).

[55] Y. C. Hon, and R. Schaback, On Unsymmetric Collocation by Radial Basis Functions, *Internat. J. Appl. Sci. Comput.*, **119**, 177-186, (2001).

[56] Y. C. Hon, Multiquadric Collocation Method with Adaptive Technique for Problems with Boundary Layer, *Internat. J. Appl. Sci. Comput.*, **6**, 3, 173-184, (1999).

[57] Y. C. Hon, A Quasi-Radial Basis Functions Method for American Options Pricing, *Comput. Math. Appl.*, **43**, (3-5), 513-524, (2002).

[58] Y. C. Hon, K. F. Cheung, X. Z. Mao, and E. J. Kansa, Multiquadric Solution for Shallow Water Equations, *ASCE J. Hydraulic Engineering*, **125**, (5), 524-533, (1999).

[59] Y. C. Hon, M. W. Lu, W. M. Xue, and X. Zhou, Multiquadric Method for the Numerical Solution of a Biphasic Model, *Internat. J. Appl. Sci. Comput.*, **88**, 153-175, (1997).

[60] Y. C. Hon, M. W. Lu, W. M. Xue, and X. Zhou, A New Foumulation and Computation of the Triphasic Model for Mechano-Electrochemical Mixtures, *J. Comput. Mechanics*, **24**, 155-165, (1999).

[61] Y. C. Hon, and X. Z. Mao, A Multiquadric Interpolation Method for Solving Initial Value Problems, *J. Sci. Comput.*, **12**, 51-55, (1997).

[62] Y. C. Hon, and X. Z. Mao, An Efficient Numerical Scheme for Burgers' Equation, *Internat. J. Appl. Sci. Comput.*, **95** 37-50 (1998).

[63] Y. C. Hon, and X. Z. Mao, A Radial Basis Function Method for Solving Options Pricing Models, *Financial Engineering*, **8**, (1), 31-49, (1999).

[64] Y. C. Hon, and Z. Wu, Additive Schwarz Domain Decomposition with Radial Basis Approximation, *Int. J. Appl. Math.*, **4**, 81-98, (2002).

[65] T. Ishikawa, and M. Matsunami, An Optimization Method Based on Radial Basis Functions, *IEEE Trans. on Magnetics*, **33**, (2), 1868-1871, (1997).

[66] T. Ishikawa, Y. Tsukui, and M. Matsunami, A Combines Method for the Global Optimization Using Radial Basis Function and Deterministic Approach, *IEEE Trans. on Magnetics*, **35**, (3), 1730-1733, (1999).

[67] Iske, A., and J. Levesley, Multilevel Scattered Data Approximation by Adaptive Domain Decomposition, *Technical Report No. 2002/17*, University of Leicester, April, (2002).

[68] E. J. Kansa, Multiquadrics—A Scattered Data Approximation Scheme with Applications to Computational Fluid-Dynamics. I. Surface Approximations and Partial Derivative Estimates, *Comput. Math. Appl.*, **19**, (8-9), 127-145, (1990).

[69] E. J. Kansa, Multiquadrics—A Scattered Data Approximation Scheme with Applications to Computational Fluid-Dynamics. II. Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations, *Comput. Math. Appl.*, **19**, (8-9), 147-161, (1990).

[70] E. J. Kansa, and Y. C. Hon, Circumventing the Ill-Conditioning Problem with Multiquadric Radial Basis Functions: Applications to Elliptic Partial Differential Equations, *Comput. Math. Appl.*, **39**, (7/8), 123-137, (2000).

[71] E. J. Kansa, H. Power, G. E. Fasshauer, and L. Ling, A Volumetric Integral Radial Basis Function Method for Time Dependent Partial Differential Equations: I. Formulation, *Eng. Anal. Bound. Elem.*, (To Appear).

[72] E. Larsson, and B. Fornberg, A Numerical Study of Radial Basis Functions Based Solution Methods for Elliptic PDEs, *Comput. Math. Appl.*, (To Appear).

[73] C. L. Lawson, and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, (1974).

[74] J. Y. Lee, and L. Greengard, A Fast Adaptive Numerical Method for Stiff Two-Point Boundary Value Problems. *SIAM J. on Scient. Comput.*, **18**, (2), 403-429, (1997).

[75] J. Li, and Y. C. Hon, Domain Decomposition for Radial Basis Meshless Methods, *Numer. Methods for Partial Different Equations*, (In Review).

[76] X. Li, C. H. Ho, C. S. Chen, Computational Test of Approximation of Functions and Their Derivatives by Radial Basis Functions, *Neural Parallel Sci. Comput.*, **10**, (1), 25-46, (2002).

[77] L. Ling, and M. R. Trummer, Multiquadric Collocation Method with Integral Formulation for Boundary Layer Problems, *Comput. Math. Appl.*, (In Review).

[78] W. R. Madych, and S. A. Nelson, Multivariate Interpolation and Conditionally Positive Definite Functions, *Approx. Theory Appl.*, **4**, 77-89, (1988).

[79] W. R. Madych, Miscellaneous Error Bounds for Multiquadric and Related Interpolators, *Comput. Math. Appl.*, **24**, 121-138, (1992).

[80] M. D. Marcozzi, S. Choi, and C. S. Chen, On the Use of Boundary Conditions for Variational Formulations Arising in Financial Mathematics, *Appl. Math. Comput.*, **124**, 197-214, (2001).

[81] N. Mai-Duy, and T. Tran-Cong, Numerical Solution of Differential Equations Using Multiquadric Radial Basis Function Networks, *Neural Networks*, **14**, 185-199, (2001).

[82] N. Mai-Duy, and T. Tran-Cong, Numerical Solution of Navier-Stokes Equations Using Multiquadric Radial Basis Function Networks, *Int. J. Num. Meth. Fluid*, **37**, 65-86, (2001).

[83] C. A. Micchelli, Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions, *Constr. Approx.*, **2**, (1), 11-22, (1986).

[84] J. J. H. Miller, E. O'Riordan, and G. I. Shishkin, On Piecewise-Uniform Meshes for Upwind- and Central-Difference Operators for Solving Singularly Perturbed Problems, *SIAM J. Numer. Anal.*, **15**, 89-99, (1995).

[85] C. T. Mouat, *Fast Algorithms and Preconditioning Techniques for Fitting Radial Basis Functions*, Ph.D. Thesis, Mathematics Department, University of Canterbury, Christchurch, New Zealand, (2001).

[86] J. Moody, and C. J. Darken, Fast Learning in Networks of Locally-Tuned Processing Units, *Neural Comput.*, **1**, 281-294, (1989).

[87] G. J. Moridis, and E. J. Kansa, The Laplace Transform Multiquadrics Method: a Highly Accurate Scheme for the Numerical Solution of Linear Partial Differential Equations, *J. Appl. Sci. Comput.* , **1**, (2), 375-407, (1994).

[88] D. E. Myers, S. De Iaco, D. Posa, and L. De Cesare, Space-Time Radial Basis Functions, *Comput. Math. Appl.*, **43**, 539-549, (2002).

[89] F. J. Narcowich, R. Schaback, and J. D. Ward, Multilevel Interpolation and Approximation, *Appl. Comput. Harmonic Anal.*, **7**, 243-261, (1999)

[90] S. Pereira and T. Pun, Robust Template Matching for Affine Resistant Image Watermarks, *IEEE Trans. on Image Processing*, **9**, (6), 1123-1129, (June 2000).

[91] R. Schaback, Error Estimates and Condition Numbers for Radial Basis Function Interpolation, *Adv. Comput. Math.*, **3**, 251-264, (1995).

[92] R. Schaback, On the Efficiency of Interpolation by Radial Basis Functions, *Proceedings of Chamonix*, (A. Lemehaute, C. Rabut, and L. L. Schumaker, Eds.), 309-318, (1996).

[93] R. Schaback, Multivariate Interpolation and Approximation by Translates of a Basis Function, *Approximation Theory VIII: Invited Lecture*, (C.K. Chui, L. L. Schumaker, Eds), Vanderbilt University Press, (1996).

[94] H. A. Schwarz, *Gesammelte Mathematische Abhandlungen*, **2**, 113-143, Springer, Berlin, (1890). First Published in *Vierteljahrsschrift Naturforsch*, Ges. Zürich, **15**, 272-286, (1870).

[95] R. Sibson, and G. Stone, Computation of Thin-Plate Splines, *SIAM J. Sci. Statist. Comput.*, **12**, 1304-1313, (1991).

[96] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, (1996).

[97] T. Tang, and M. R. Trummer, Boundary Layer Resolving Spectral Methods for Singular Perturbation Problems, *SIAM J. Sci. Comput.*, **17**, (2), 430-438, (1996).

[98] L. N. Trefethen, and D. Bau Iii, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, (1997).

[99] M. Unser, Quasi-Orthogonality and Quasi-Projections, *Appl. Comput. Harmon. Anal.*, **3**, (3), 201-214, (1996).

[100] R. Vulanovic, On a Numerical Solution of a Type of Singularly Perturbed Boundary Value Problem by Using a Special Discretization Mesh, *Univ. U Novom Sadu Zb. Rad. Prirod. Mat. Fak. Ser. Mat.*, **13**, 187-201, (1983).

[101] J. G. Wang, and G. R. Liu, On the Optimal Shape Parameters of Radial Basis Functions Used for 2-D Meshless Methods, *Comput. Methods Appl. Mech. Engrg.*, **191**, 2611-2630, (2002).

[102] Z. M. Wu and R. Schaback, Shape Preserving Properties and Convergence of Univariate Multiquadric Quasi-Interpolation, *ACTA Math. Appl. Sinica (English Ser.)*, **10**, (4), 441-446, (1994).

[103] H. Wendland, Piecewise Polynomial, Positive Definite and Compactly Supported Radial Functions of Minimal Degree, *Adv. Comput. Math.*, **4**, 389-396, (1995).

[104] H. Wendland, Sobolev-Type Error Estimates for Interpolation by Radial Basis Functions, in *Surface Fitting and Multiresolution Methods*, (A. Lemehaute, C. Rabut, and L. L. Schumaker, Eds.), 337-344, Vanderbilt University Press, Nashville, TN, (1997).

[105] H. Wendland, Error Estimates for Interpolation by Compactly Supported Radial Basis Functions of Minimal Degree, *J. Approx. Theory*, **93**, 258-272, (1998).

[106] S. M. Wong, Y. C. Hon, and M. A. Golberg, Compactly Supported Radial Basis Functions for the Shallow Water Equations, *Internat. J. Appl. Sci. Comput.*, **127**, 79-101, (2002).

[107] S. M. Wong, Y. C. Hon, T. S. Li, S. L. Chun, and E. J. Kansa, Multizone Decomposition of Time-Dependent Problems Using the Multiquadric Scheme, *Comput. Math. Appl.*, **37**, (8), 23-43, (1999).

[108] Z. Wu, and R. Schaback, Local Error Estimates for Radial Basis Function Interpolation of Scattered Data, *IMA J. Numer. Anal.*, **13**, 13-27, (1993).

[109] J. Yoon, Spectral Approximation Orders of Radial Basis Function Interpolation on the Sobolov Space, *SIAM J. Math. Anal.*, **33**, 946-958, (2001).

[110] X. Zhou, Y. C. Hon, and K. F. Cheung, A Grid-Free, Nonlinear Shallow-Water Model with Moving Boundary, *Eng. Anal. Bound. Elem.*, (To Appear).

[111] X. Zhou, Y. C. Hon, and J. Li, Overlapping Demain Decomposition Method by Radial Basis Functions, *Appl. Numer. Math.*, **44**, (1-2), 241-255, (Jan 2003).