Numerical Integration in Meshfree Methods

Pravin Madhavan

New College

University of Oxford

A thesis submitted for the degree of

Master of Science in

Mathematical Modelling and Scientific Computing

Trinity 2010

Dedicated to Henry Lo

Acknowledgements

I would like to thank my supervisor Professor Holger Wendland for all his help and guidance during the writing of this dissertation. I am also very grateful to Dr. Kathryn Gillow for the useful advice she gave me to improve my dissertation. Last but not least I would like to thank all the students of the MSc Mathematical Modelling and Scientific Computing course who have made this past year such a great experience.

Abstract

The issue of finding efficient numerical integration schemes for integrals arising in the Galerkin approximation of partial differential equations using compactly supported radial basis functions is studied. In this dissertation, the model problem of the Helmholtz equation with natural boundary conditions will be considered. Different quadrature rules will be discussed and studied as well as their influence on the convergence and stability of the approximation process.

Contents

List of Figures

Chapter 1

Introduction

1.1 Meshfree Methods

During the past few years the idea of using meshfree methods as a modern tool for the numerical solution of partial differential equations has received a lot of attention within the scientific and engineering communities. As described in [12], the growing interest in these methods is in part due to the fact that they are very flexible, particularly when it comes to problems with moving boundaries or that are high dimensional. Conventional grid-based methods such as finite difference (FDM) and finite element (FEM) methods suffer from some inherent difficulties, namely the requirement of generating a mesh. This is not an easy task, especially for domains with irregular and complex geometries which require additional mathematical transformations that can be as expensive as solving the problem itself. Furthermore, using such methods to model time-dependent physical systems involving large deformations requires a costly remeshing process at each time step, making them impractical to solve such problems. Meshfree methods overcome such issues by not requiring a meshing process in the traditional sense of the term. Namely, no knowledge of the connectivity between the nodes is required when implementing the meshfree method: this means that the nodes can be arbitrarily distributed within the domain without any constraints, making them ideal for understanding the behaviour of complex solids, structures and fluid flows. For a survey of various meshfree methods and underlying results we refer the reader to [1] and [2].

1.2 Model Problem

In this dissertation we will focus our attention on the Helmholtz equation with natural boundary conditions:

$$-\nabla^2 u + u = f \text{ in } \Omega, \tag{1.1}$$

$$\frac{\partial}{\partial \mathbf{n}} u = 0 \text{ on } \partial\Omega \tag{1.2}$$

where $\Omega \subset \mathbb{R}^2$ is a bounded domain with a sufficiently smooth boundary $\partial\Omega$ and $f \in L_2(\Omega)$ is a given Lebesgue integrable function. The outer unit normal vector of Ω is denoted by $\mathbf{n}$. For simplicity we will choose our domain to be $\Omega = [-1, 1]^2$ but much of the work done in this dissertation can be applied to more general 2D domains. Due to its connection with the wave equation, the Helmholtz equation arises in many physical problems involving electromagnetism, seismology or accoustics.

1.2.1 Weak Formulation

The weak formulation of the boundary value problem (BVP) (1.1)—(1.2) can be stated as follows: find $u \in H^1(\Omega) = \left\{g : \Omega \to \mathbb{R} \mid g, \frac{\partial g}{\partial x}, \frac{\partial g}{\partial y} \in L_2(\Omega)\right\}$ such that

$$a(u, v) = l(v) \ \forall v \in H^1(\Omega), \tag{1.3}$$

where the bilinear form $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ is given by

$$a(u, v) = \int_\Omega (\nabla u \cdot \nabla v + uv) \, d\mathbf{x}$$

and the linear form $l : H^1(\Omega) \to \mathbb{R}$ by

$$l(v) = \int_\Omega fv \, d\mathbf{x}.$$

The following theorem taken from [4] guarantees uniqueness of the BVP.

Theorem 1.2.1 (Lax-Milgram) *If $a(\cdot, \cdot)$ is coercive and continuous and $l(\cdot)$ is continuous on a real Hilbert space $\mathbf{V}$ then there is a unique $u \in \boldsymbol{V}$ for which*

$$a(u, v) = l(v) \quad \forall v \in \boldsymbol{V}.$$

It can be easily shown that the bilinear form of (1.3) is coercive and continuous on the Hilbert space $\mathbf{V} = H^1(\Omega)$ (in fact it is an inner product), and that the linear form is continuous on this space. Hence the weak formulation (1.3) has a unique solution.

Before we attempt to solve the BVP numerically by considering a finite-dimensional subspace of $H^1(\Omega)$, we need some background on the basis functions we will be using to span the subspace.

1.3 Radial Basis Functions

Amongst meshfree methods, one of the most popular techniques is given by radial basis functions (RBF). An excellent reference for much of the theory and proofs described in this section can be found in [20] as well as [19], whilst implementation aspects are found extensively in [10]. The following is a formal definition of a radial function.

Definition A function $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called *radial* provided there exists a *univariate* function $\varphi : [0, \infty] \to \mathbb{R}$ such that

$$\Phi(\mathbf{x} - \mathbf{y}) = \varphi(r), \text{ where } r = \|\mathbf{x} - \mathbf{y}\|$$

where we choose $\|\cdot\|$ to be the Euclidean norm on $\mathbb{R}^d$.

Furthermore, we require these functions to be **compactly supported** and to have the following property:

Definition A real-valued continuous function $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is said to be *positive semi-definite* on $\mathbb{R}^d$ if for all $N \geq 1$ and for all pairwise distinct centres $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, the matrix $(\Phi(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ is positive semi-definite i.e.

$$\sum_{j=1}^{N} \sum_{k=1}^{N} c_j c_k \Phi(\mathbf{x}_j, \mathbf{x}_k) \geq 0 \tag{1.4}$$

for all $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{R}^N$.

The function Φ is called *positive definite* on $\mathbb{R}^d$ if the matrix in (1.4) is zero if and only if $\mathbf{c} \equiv \mathbf{0}$.

An important property of dealing with positive definite functions on $\mathbb{R}^d$ is their invariance under *translation*, in which case we have $\Phi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x} - \mathbf{y}, 0) := \tilde{\Phi}(\mathbf{x} - \mathbf{y})$ i.e. it can be written as a function with only one variable component. Furthermore, if the function is radial then it is also invariant under *rotation* and *reflection*.

Now that we have defined the sort of functions we will be using, it would be interesting to find the natural function spaces that arise when using them. Consider the following:

Definition Let $\mathbf{H}$ be a real Hilbert space of functions $f : \Omega(\subset \mathbb{R}^d) \to \mathbb{R}$ with inner product $< \cdot, \cdot >_{\mathbf{H}}$. Then $\mathbf{H}$ is called a *reproducing kernel Hilbert space* (RKHS) with reproducing kernel $K : \Omega \times \Omega \to \mathbb{R}$ if

- $K(\cdot, \mathbf{x}) \in \mathbf{H}$ for all $\mathbf{x} \in \Omega$.
- $f(\mathbf{x}) =< f, K(\cdot, \mathbf{x}) >_{\mathbf{H}}$ for all $f \in \mathbf{H}$ and all $\mathbf{x} \in \Omega$.

One important property of a RKHS is that its associated reproducing kernel K is known to be positive semi-definite, as stated in the following theorem.

Theorem 1.3.1 *Suppose* $\mathbf{H}$ *is a RKHS with reproducing kernel* $K : \Omega \times \Omega \to \mathbb{R}$. *Then* K *is positive semi-definite. Moreover,* K *is positive definite if and only if point evaluations are linearly independent.*

This theorem provides one direction of the connection between positive definite functions and reproducing kernels. However, we are also interested in the other direction: we would like to know how to construct a RKHS when given positive definite functions. This is of obvious interest to us because we would like to construct a RKHS using, in particular, the compactly supported positive definite radial basis functions mentioned before.

We will now show that every positive definite function is associated with a RKHS, which is called its *native space*. Firstly, we define the space spanned by the positive definite functions $\Phi(\cdot, \mathbf{x})$ given by

$$F_\Phi = \text{span}\{\Phi(\cdot, \mathbf{x}) : \mathbf{x} \in \mathbf{R}^d\}.$$

Note that by choosing such a space we have already satisfied the first property of the definition of a RKHS. Furthermore let

$$f = \sum_{j=1}^{N} \alpha_j \Phi(\cdot, \mathbf{x}_j) \ , \ g = \sum_{k=1}^{M} \beta_k \Phi(\cdot, \mathbf{y}_k)$$

and define the inner product

$$< f, g >_\Phi := \sum_{j=1}^{N} \sum_{k=1}^{M} \alpha_j \beta_k \Phi(\mathbf{x}_j, \mathbf{y}_k).$$

Then

$$< f, \Phi(\cdot, \mathbf{x}) >_K = \sum_{j=1}^{N} \alpha_j \Phi(\mathbf{x}_j, \mathbf{x}) = f(\mathbf{x})$$

and so Φ is in fact a reproducing kernel. To construct a RKHS all we need now is for the space F_Φ to be complete.

Definition The completion of F_Φ with respect to $|| \cdot ||_\Phi$ is the associated RKHS (native space) to Φ, call it H_Φ.

The following theorem states that every RKHS is the completion of the space F_Φ.

Theorem 1.3.2 *If G is a RKHS with kernel Φ, then $G = H_\Phi$ i.e. the native space.*

Using this theorem it can be shown what space can be "generated" by a given positive definite reproducing kernel Φ.

Theorem 1.3.3 *Let Φ be given as before. Consider the space*

$$G = \{f \in L_2(\mathbb{R}^d) \cap C(\mathbb{R}^d) : ||f||_G < \infty\}$$

where the norm $|| \cdot ||_G$ is given by

$$||f||_G^2 := (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \frac{|\hat{f}(w)|^2}{\hat{\Phi}(w)} \, dw$$

where $\hat{f}$ denotes the Fourier transform of f. Then G is a RKHS with reproducing kernel Φ.

By Theorem 1.3.2, it follows that we have $G = H_\Phi$.

One of the most popular classes of compactly supported RBFs is the one introduced Wendland in [17]. The Wendland functions $\Phi_{l,k}(r)$ where $l = \lfloor \frac{d}{2} \rfloor + k + 1$ are strictly positive definite in $\mathbb{R}^d$ and have the property of being constructed to have any desired amount of smoothness $2k$. For example, the functions

$$\begin{aligned}
\Phi_{2,0}(r) &= (1-r)_+^2, \\
\Phi_{3,1}(r) &= (1-r)_+^4(4r+1), \qquad (1.5)\\
\Phi_{4,2}(r) &= (1-r)_+^6(35r^2+18r+3), \\
\Phi_{5,3}(r) &= (1-r)_+^8(32r^3+25r^2+8r+1)
\end{aligned}$$

are respectively in C^0, C^2, C^4 and C^6, and strictly positive definite in $\mathbb{R}^3$. Using the compactly supported Wendland functions $\Phi = \Phi_{l,k}$ as the spanning set for the space F_Φ, it can be shown by Theorem 1.3.3 that the space "generated" by these functions is the Sobolev space i.e.

$$\Phi_{l,k} \text{ "generates" } H^\tau \text{where } \tau = \frac{d}{2} + k + \frac{1}{2}.$$

As discussed in the previous section, this is exactly the space in which we seek to find a weak solution to the Helmholtz equation. The compactly supported Wendland functions thus provide a natural choice of basis functions to use for its numerical solution.

In this dissertation we will focus our attention on the C^2 compactly supported Wendland function (1.5) which has been plotted in Figure 1.1. As shown in this figure,

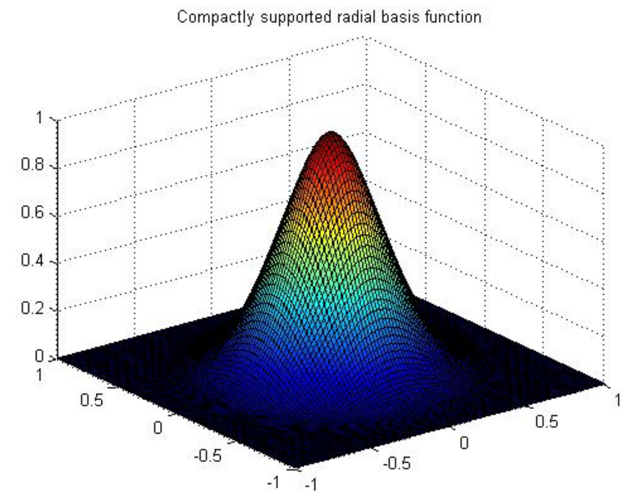


Figure 1.1: C^2 Wendland basis function

the support of these functions are circles with radius $R = 1$ i.e. supp $\Phi \subseteq B(0, 1)$. The support can be scaled appropriately by defining $\Phi_\delta(\cdot) := \Phi(\frac{\cdot}{\delta})$, in which case we have supp $\Phi_\delta = B(0, \delta)$.

1.4 Galerkin Formulation

We can now refer back to the weak formulation (1.3) of the Helmholtz PDE with natural boundary conditions. To solve this problem numerically we replace the infinite-dimensional space $H^1(\Omega)$ by some finite-dimensional subspace $V_N \subseteq H^1(\Omega)$ and solve the Galerkin problem,

$$\text{find } u_N \in V_N \text{ such that } a(u_N, v) = l(v)\ \forall v \in V_N. \tag{1.6}$$

So far this process has not been any different to how we would compute the finite element solution of the Helmholtz PDE, for which we would now choose a tessellation (say, a triangulation) of the domain Ω and take V_N as the space of piecewise polynomials induced by the tessellation, the details of which can be found in [8]. We move away from this usual approach by requiring the space V_N to now be spanned by the compactly supported radial basis functions described in the previous section i.e.

$$V_N = \text{span}\{\Phi(\cdot - \mathbf{x}_j) : 1 \leq j \leq N\}$$

where as before $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subseteq \Omega$ is a set of pairwise distinct centres in Ω. Note that from now on, Φ will always denote a scaled version of (1.5) chosen so that the support radius is R (unless otherwise stated). The Galerkin approximation u_N can thus be written as

$$u_N = \sum_{j=1}^{N} c_j \Phi(\cdot - \mathbf{x}_j) \tag{1.7}$$

and the Galerkin problem (1.6) is given by

$$\int_\Omega [\nabla u_N(\mathbf{x}) \cdot \nabla\Phi(\mathbf{x} - \mathbf{x}_i) + u_N \Phi(\mathbf{x} - \mathbf{x}_i)] d\mathbf{x} = \int_\Omega f(\mathbf{x})\Phi(\mathbf{x} - \mathbf{x}_i) d\mathbf{x},\ i = 1, ..., N.$$

Using linearity and expression (1.7) for u_N we obtain

$$\sum_{j=1}^{N} c_j \left(\int_\Omega [\nabla\Phi(\mathbf{x} - \mathbf{x}_j) \cdot \nabla\Phi(\mathbf{x} - \mathbf{x}_i) + \Phi(\mathbf{x} - \mathbf{x}_j)\Phi(\mathbf{x} - \mathbf{x}_i)] d\mathbf{x} \right)$$
$$= \int_\Omega f(\mathbf{x})\Phi(\mathbf{x} - \mathbf{x}_i) d\mathbf{x},\ i = 1, ..., N. \tag{1.8}$$

Expression (1.8) is a system of equations which can be written in the more compact and suggestive form

$$\mathbf{Ac} = \mathbf{f}, \tag{1.9}$$

with the entries of the N-by-N *stiffness matrix* $\mathbf{A} = \{a_{ij}\}$ given by

$$a_{ij} = \int_\Omega [\nabla\Phi(\mathbf{x} - \mathbf{x}_j) \cdot \nabla\Phi(\mathbf{x} - \mathbf{x}_i) + \Phi(\mathbf{x} - \mathbf{x}_j)\Phi(\mathbf{x} - \mathbf{x}_i)] d\mathbf{x},$$

the entries of the N-by-1 *load vector* $\mathbf{f}$ by

$$f_i = \int_\Omega f(\mathbf{x})\Phi(\mathbf{x} - \mathbf{x}_i) d\mathbf{x}$$

and the solution vector $\mathbf{c}$ by

$$\mathbf{c} = (c_1, c_2, ..., c_N)^T.$$

Theorem 1.4.1 (Céa) *Suppose that* $\mathbf{V}$ *is a real Hilbert space and that the bilinear form* $a(\cdot,\cdot)$ *and linear functional* $l(\cdot)$ *satisfy*

$$a(u,u) \geq \gamma ||u||_V^2 \textit{ for all } u \in \mathbf{V},$$

$$a(u,v) \leq \Gamma ||u||_V ||v||_V \textit{ for all } u, v \in \mathbf{V},$$

$$l(v) \leq \Lambda ||v||_V \textit{ for all } v \in \mathbf{V}.$$

If $V_N \subset \mathbf{V}$ *is a vector subspace with* $u_N \in V_N$ *satisfying*

$$a(u_N, v) = l(v) \textit{ for all } v \in V_N$$

then

$$||u - u_N||_V \leq \frac{\Gamma}{\gamma} ||u - s||_V \textit{ for all } s \in V_N.$$

By Céa's lemma and the fact that $\mathbf{V} = H^1(\Omega)$ is a Hilbert space, solving system (1.9) is equivalent to solving the following Sobolev minimisation problem: find $s \in V_N$ that satisfies

$$\min_{s \in V_N} ||u - s||_{H^1(\Omega)}^2.$$

Proposition 1.4.2 *The stiffness matrix* $\mathbf{A}$ *is symmetric and positive definite.*

Proof The matrix is clearly symmetric since

$$\begin{aligned} a_{ij} = \int_\Omega & [\nabla\Phi(\mathbf{x}-\mathbf{x}_j)\cdot\nabla\Phi(\mathbf{x}-\mathbf{x}_i) + \Phi(\mathbf{x}-\mathbf{x}_j)\Phi(\mathbf{x}-\mathbf{x}_i)]d\mathbf{x} \\ &= \int_\Omega [\nabla\Phi(\mathbf{x}-\mathbf{x}_i)\cdot\nabla\Phi(\mathbf{x}-\mathbf{x}_j) + \Phi(\mathbf{x}-\mathbf{x}_i)\Phi(\mathbf{x}-\mathbf{x}_j)]d\mathbf{x} = a_{ji}. \end{aligned}$$

Furthermore, for all $\mathbf{w} \in \mathbb{R}^N$ we have

$$\begin{aligned} \mathbf{w}^T\mathbf{A}\mathbf{w} &= \sum_i w_i \sum_j a_{ij} w_j = \sum_i \sum_j w_i w_j a_{ij} \\ &= \sum_i \sum_j w_i w_j \int_\Omega [\nabla\Phi(\mathbf{x}-\mathbf{x}_j)\cdot\nabla\Phi(\mathbf{x}-\mathbf{x}_i) + \Phi(\mathbf{x}-\mathbf{x}_j)\Phi(\mathbf{x}-\mathbf{x}_i)]d\mathbf{x}. \quad (1.10) \end{aligned}$$

Now letting $w = \sum_{j=1}^N w_j \Phi(\cdot - \mathbf{x}_j)$ and substituting it into (1.10) we get

$$\mathbf{w}^T\mathbf{A}\mathbf{w} = \int_\Omega |\nabla w|^2 + w^2 d\mathbf{x} \geq 0$$

with

$$\int_\Omega |\nabla w|^2 + w^2 d\mathbf{x} = 0 \Leftrightarrow w \equiv 0$$

since $a(w,v) = \int_\Omega \nabla w \cdot \nabla v + wv \, d\mathbf{x}$ is an inner product over $H^1(\Omega)$ for all $w, v \in H^1(\Omega)$. But if $w \equiv 0$ then $\mathbf{w} = \mathbf{0}$ where $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^N$. ▮

Hence the stiffness matrix $\mathbf{A}$ is certainly invertible and the matrix problem (1.9) has a solution which is unique.

Now that we have described in detail the setting for the problem, the remainder of this dissertation will discuss various issues related to the numerical solution of the Galerkin problem (1.6). This will be structured as follows: in Chapter 2, numerical integration schemes required to approximate the entries of the stiffness matrix will be analysed and compared to one another. Part of Chapter 2 is based on work done in a relevant Special Topic but substantial improvements have been made to consider other numerical integration schemes and discuss problems arising when using some of them. In Chapter 3, these schemes will be used to verify numerically known theoretical convergence bounds. We hope that this will be a good indicator of how efficient they are. In the process, we will solve the Helmholtz equation numerically for a particular right-hand side function f which yields a known, closed-form solution u. Finally in Chapter 4, a multilevel Galerkin algorithm required to resolve certain convergence issues will be briefly described.

Chapter 2

Numerical Integration

2.1 Motivation

The evaluation of the integrals in the previous chapter is the most time-consuming process in the RBF Galerkin approach to solving PDEs. Finding efficient quadrature rules for accurately approximating these integrals is thus one of the major problems with this approach. The classical way of approximating these integrals is to apply a quadrature rule over the entire square domain $\Omega = [-1,1]^2$ as shown in Figure 2.1. However, integrating over the entire domain does not exploit the fact that the

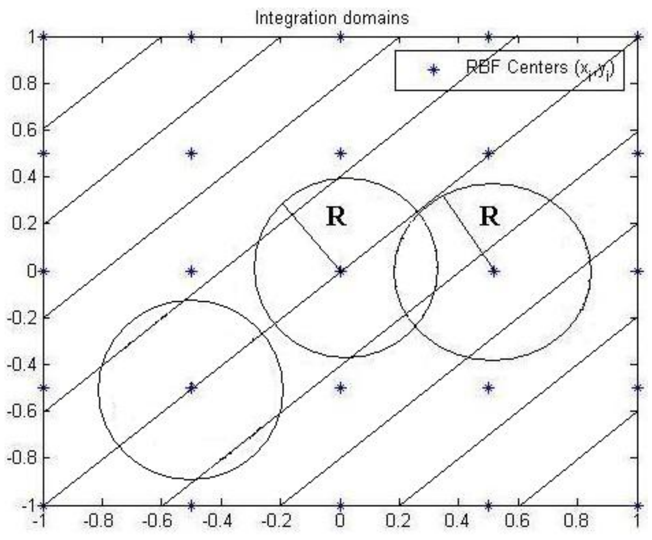


Figure 2.1: Integration on whole domain.

RBFs only have their compact support over a cicular domain: it would be pointless to integrate over regions on which we know that the RBF is zero, moreso for the product of RBFs (and their gradients) whose support is even "smaller" and corresponds to the intersection of two circles i.e. a lens-shaped region. This is shown in Figure 2.2. Furthermore, integrating over the whole domain is generally one of the main sources

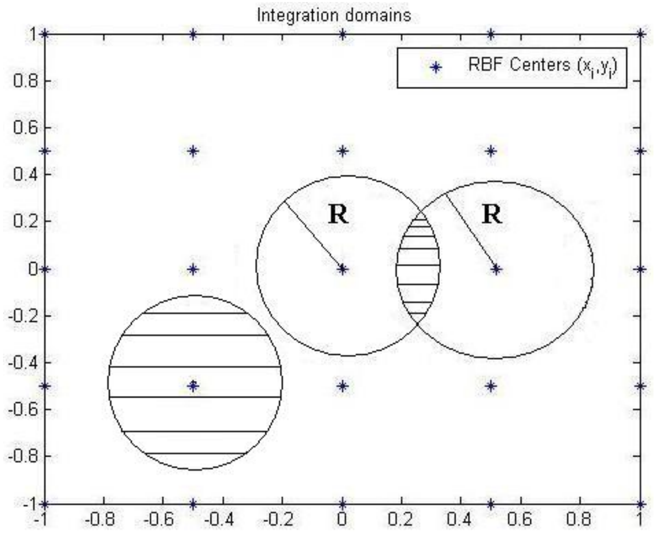


Figure 2.2: Integration on subdomains.

of error created by the corresponding quadrature rule. This chapter seeks to discuss alternative quadrature rules that exploit the properties of the RBF mentioned above in order to compute the entries of the stiffness matrix and load vector more accurately. We will analyse and compare different numerical integration schemes for the circle and the lens-shaped domain. Once this is done these quadrature rules can then be used in the calculation of the entries of the stiffness matrix and load vector. For details of numerical integration schemes associated with the use of compactly supported basis functions which have square/rectangular supports we refer the reader to [7].

2.2 Background

We will now outline some of the basic tools and properties required for what will follow. These will then be used to develop the required numerical integration schemes.

2.2.1 Some Basic Geometry

In this subsection we will mainly be concerned with some geometric properties of the lens, as those of the circle are well understood. Consider the lens-shaped intersection of two circles of equal radii R in the (x, y)-plane as shown in Figure 2.3. Without loss of generality we assume that one of the circles is centred at the origin $(0, 0)$ and

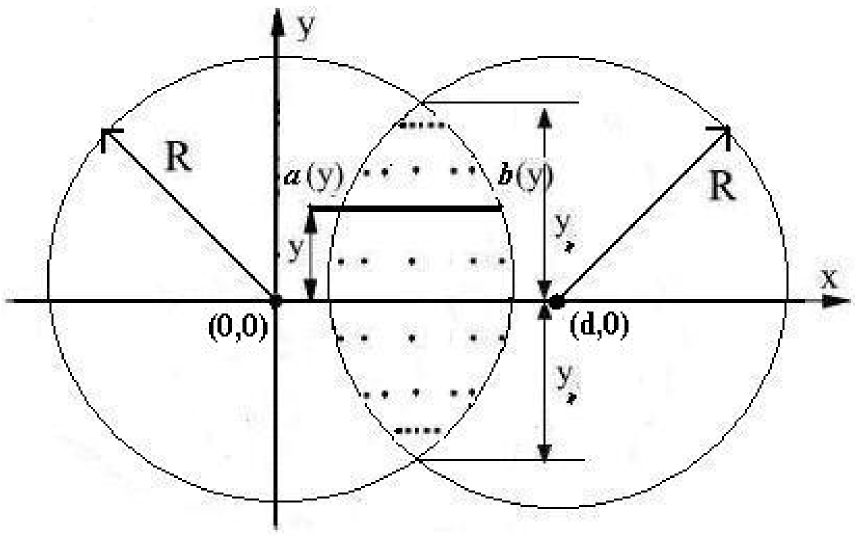


Figure 2.3: Lens-shaped domain of integration.

the other at $(d, 0)$ where $d > 0$, assuming so will significantly simplify our analysis. The equations of these circles are thus respectively

$$x^2 + y^2 = R^2, \tag{2.1}$$

$$(x - d)^2 + y^2 = R^2. \tag{2.2}$$

We are interested in finding the intersection points $(x_*, \pm y_*)$ between these two circles, which exist if $d \leq 2R$. To do so, we substitute (2.1) into (2.2) to get

$$(x - d)^2 + (R - x^2) = R^2$$

$$\Rightarrow -2dx + d^2 = 0.$$

And so the x-component of the intersection points is given by

$$x_* = \frac{d}{2}. \tag{2.3}$$

Substituting (2.3) back into (2.1) we obtain the y-component of the intersection points as follows:

$$y_*^2 = R^2 - x_*^2 = R^2 - \left(\frac{d}{2}\right)^2.$$

Simplifying this expression we get the y-components of the intersections given by

$$y_* = \pm\frac{\sqrt{4R^2 - d^2}}{2}. \tag{2.4}$$

Now if we had to integrate a function $f(x, y)$ over this lens i.e. $\int\int_\Omega f(x, y)\ dxdy$ where Ω is the lens-shaped domain, then the above working shows that the bounds of the double integral will be given by:

$$\int_{-y_*}^{y_*} \int_{d-\sqrt{R^2-y^2}}^{\sqrt{R^2-y^2}} f(x, y) dxdy. \tag{2.5}$$

This will be one of the double integrals we will seek to find quadratures for in this chapter. With such bounds it can now be understood why it is rarely possible to integrate (2.5) exactly. However, in order to compare the numerical integration schemes on this region we need a test function that yields an exact result. Choosing

$$\chi_{\text{lens}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \text{lens} \\ 0 & \text{if } \mathbf{x} \notin \text{lens} \end{cases}$$

(where $\mathbf{x}$=(x, y)) will yield the area of the lens. It can be shown geometrically that the area of the lens is given by the formula

$$\text{Area(lens)} = 2R^2 \cos^{-1}\left(\frac{d}{2R}\right) - \frac{d}{2}\sqrt{4R^2 - d^2}.$$

So we can use this simple function as one of the test functions used to compare the different numerical integration schemes.

2.2.2 Numerical Integration in 1-D

In this section we will outline some of the important numerical integration rules and properties in 1-D, much of which has been taken from and extensively discussed in [14]. Namely, we will briefly describe some of the important results related to two well-known quadrature rules: the Newton-Cotes formula and the more accurate Gauss quadrature. Though some of these properties cannot be easily generalised to higher dimensions it will nevertheless be our starting point for developing quadratures of the 2-D integrals we are interested in approximating.

2.2.2.1 Newton-Cotes Quadrature

For this simple quadrature rule we approximate a real-valued, continuous function $f(x)$ on the closed interval $[a, b]$ by its Lagrange interpolant $p_n(x)$, defined as follows:

Definition For a positive integer n, let x_i, $i = 0, 1, ..., n$, denote the equally spaced interpolation points in the interval $[a, b]$, that is,

$$x_i = a + ih\ ,\ \ i = 0, 1, ..., n$$

where

$$h = (b - a)/n.$$

Then the **Lagrange interpolation polynomial** of degree n for the function f in the interval $[a, b]$, with these interpolation points, is of the form

$$p_n(x) = \sum_{k=0}^{n} L_k(x) f(x_k) \text{ where } L_k(x) = \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}.$$

Assuming that the function f is sufficiently smooth, we can derive a bound for the size of the interpolation error between f and its Lagrange interpolant $p_n(x)$ as follows:

Theorem 2.2.1 *Suppose that $n \geq 0$ and that f is a real-valued function, defined and continuous on the closed real interval $[a, b]$, such that the derivative of order $n + 1$ exists and is continuous on $[a, b]$. Then given $x \in [a, b]$, there exists $\xi \in [a, b]$ such that*

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\pi_{n+1}(x)|$$

where $M_{n+1} := \max_{\xi \in [a,b]} |f^{n+1}(\xi)|$ and $\pi_{n+1}(x) = (x - x_0)(x - x_1)...(x - x_n)$.

Given such a property, it is reasonable to believe that

$$\int_a^b f(x)dx \approx \int_a^b p_n(x)dx. \tag{2.6}$$

Inserting the expression above for $p_n(x)$ into (2.6) we get,

$$\int_a^b f(x)dx \approx \sum_{k=0}^{n} w_k f(x_k) \tag{2.7}$$

where

$$w_k = \int_a^b L_k(x)dx \ , \ k = 0, 1, ..., n. \tag{2.8}$$

The values w_k, $k = 0, 1, ..., n$ are known as the **quadrature weights** of the interpolation, and the interpolation points x_k, $k = 0, 1, ..., n$, are called the **quadrature points**. The numerical integration formula (2.7) with quadrature weights (2.8) and equally spaced quadrature points is known as the **Newton-Cotes formula** of order n.

A special case of this formula arises when we take $n = 1$, so that $x_0 = a$ and $x_1 = b$. This yields the **trapezium rule** which gives the quadrature

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a) + f(b)].$$

Let $E_n(f) = \int_a^b f(x)dx - \sum_{k=0}^n w_k f(x_k)$ be the quadrature error when using n points. By the previous theorem, for $n = 1$ (trapezium rule), we have

$$|E_1(f)| \leq \frac{(b-a)^3}{12} M_2.$$

In order to improve this bound, we can divide the interval $[a, b]$ into an increasing number of subintervals of decreasing size. We can then apply the trapezium rule on each of these subintervals. This yields the **composite trapezium rule**:

$$\int_a^b f(x)dx \approx h\Big[\frac{1}{2}f(x_0) + f(x_1) + ... + \frac{1}{2}f(x_m)\Big] \tag{2.9}$$

where

$$x_i = a + ih = a + \frac{i}{m}(b-a), \quad i = 0, 1, ..., m.$$

The error $E_1^c(f)$ using the composite trapezium rule can easily be found by applying the previously derived error bound on each of the subintervals. Doing so we obtain,

$$|E_1^c(f)| \leq \frac{(b-a)^3}{12m^2} M_2$$

which is a significant improvement from the bound for the classical trapezium rule.

2.2.2.2 Gauss Quadrature

We now consider approximating f by its Hermite interpolant defined as follows.

Definition Suppose f is a real-valued, continuous function defined on the closed interval $[a, b]$ and f is continuous and differentiable on this interval. Suppose furthermore that $x_i, \quad i = 0, ..., n$ are distinct points in $[a, b]$. Then the polynomial p_{2n+1} defined by

$$p_{2n+1}(x) = \sum_{k=0}^{n} [H_k(x)f(x_k) + K_k(x)f'(x_k)]$$

where

$$H_k(x) = [L_k(x)]^2(1 - 2L_k'(x_k)(x - x_k)) \ , \ K_k(x) = [L_k(x)]^2(x - x_k)$$

is the **Hermite interpolation polynomial** of degree $2n+1$ for f with interpolation points $x_i, \quad i = 0, ..., n$.

It can be shown that p_{2n+1} is the unique polynomial in P_{2n+1} (space of polynomials of degree $2n + 1$ or less) such that $p_{2n+1}(x_i) = f(x_i)$, $p_{2n+1}'(x_i) = f'(x_i)$. As before, given f smooth enough, we can derive bounds for the interpolation error using the Hermite interpolant:

Theorem 2.2.2 *Suppose that $n \geq 0$ and that f is a real-valued function, defined, continuous and $2n+2$ times differentiable on $[a,b]$, such that the derivative of order $2n+2$ exists and is continuous on this interval. Then for each $x \in [a,b]$ there exists $\xi \in [a,b]$ such that*

$$|f(x) - p_{2n+1}(x)| \leq \frac{M_{2n+2}}{(2n+2)!}[\pi_{n+1}(x)]^2$$

where $M_{2n+2} := \max_{\xi\in[a,b]} |f^{2n+2}(\xi)|$ and $\pi_{n+1}(x) = (x-x_0)(x-x_1)...(x-x_n)$.

Approximating the weighted integral of f with positive weight $\omega(x)$ by that of its Hermite interpolant p_{2n+1}, we get:

$$\int_a^b \omega(x)f(x)dx \approx \int_a^b \omega(x)p_{2n+1}(x)dx = \sum_{k=0}^{n} W_k f(x_k) + \sum_{k=0}^{n} V_k f'(x_k)$$

where $W_k = \int_a^b \omega(x)H_k(x)$ and $V_k = \int_a^b \omega(x)K_k(x)dx$. Now we would like to choose the quadrature points so that $V_k = 0$ for all k and hence we would not need to calculate $f'(x_k)$. After some algebraic manipulation we obtain,

$$\begin{aligned} V_k &= \int_a^b \omega(x)[L_k(x)]^2(x-x_k)dx \\ &= C_n \int_a^b \omega(x)\pi_{n+1}(x)L_k(x)dx. \end{aligned}$$

for some constant C_n. Given that π_{n+1} is a polynomial of degree $n+1$ and $L_k(x)$ has degree n for each k, we can choose the quadrature points $x_k,\ k = 0,...,n$ to be the zeros of a polynomial of degree $n+1$ from a system of (weighted) orthogonal polynomials over (a,b) and this would get rid of the V_k coefficients. Additionally, by choosing the quadrature points in such a way, we also have

$$\begin{aligned} W_k &= \int_a^b \omega(x)H_k(x)dx \\ &= \int_a^b \omega(x)[L_k(x)]^2dx - 2L_k'(x_k)V_k \\ &= \int_a^b \omega(x)[L_k(x)]^2dx \end{aligned}$$

Hence we obtain the **Gauss quadrature rule** given by

$$\int_a^b \omega(x)f(x)dx \approx \sum_{k=0}^{n} W_k f(x_k)$$

where $W_k = \int_a^b \omega(x)[L_k(x)]^2 dx$. For the case when $\omega(x) \equiv 1$ we obtain the Gauss-Legendre quadrature rule with the quadrature points $x_k,\ k = 0, ..., n$ being the zeros of the Legendre polynomial of degree $n+1$ over the interval $[-1, 1]$.

As for the Newton-Cotes formula it can be shown using Theorem 2.2.2 that for sufficiently smooth f we have

$$\int_a^b \omega(x) f(x) dx - \sum_{k=0}^{n} W_k f(x_k) = K_n f^{(2n+2)}(\xi)$$

where $\xi \in (a, b)$ and $K_n = \frac{1}{(2n+2)!} \int_a^b \omega(x)[\pi_{n+1}(x)]^2 dx$. This is clearly a significant improvement from Newton-Cotes since this rule is now exact for polynomial of degree $2n+1$ with n quadrature points.

2.3 Numerical Integration Schemes on the Lens

We can now apply some of the theory and results discussed in the previous section to find an adequate quadrature rule for integrals over the lens. As before, we will assume that both circles have radius R, and without loss of generality that the lens is formed by the intersection of one circle centred at the origin $(0, 0)$ and of another circle centred at $(d, 0)$ where $d > 0$. Indeed, an adequate mapping (translation and rotation) can always be found to reduce our analysis to this simpler case.

2.3.1 First Approach

We now wish to find a suitable way of choosing the quadrature points and weights for our 2-dimensional domain. To do so, we apply the quadrature rules described above to each one of the integrals of

$$\int \int_{\Omega} f(x, y) dx dy$$

where Ω is the lens-shaped domain i.e. apply the quadrature on a line integral dependent only on y and then apply a quadrature rule along each line integral dependent only on x, passing through each of the quadrature points y_j as shown in Figure 2.3. Mathemtically, this can be done as follows:

$$\int_{-y_*}^{y_*} \int_{d-\sqrt{R^2-y^2}}^{\sqrt{R^2-y^2}} f(x, y) dx dy = \int_{-y_*}^{y_*} \int_{a(y)}^{b(y)} f(x, y) dx dy \tag{2.10}$$

where $a(y) := d - \sqrt{R^2 - y^2}$ and $b(y) := \sqrt{R^2 - y^2}$. We now consider the following substitution of variables,

$$y = y_* \tilde{y}$$

and

$$x = \frac{b(y) - a(y)}{2} \tilde{x} + \frac{a(y) + b(y)}{2}.$$

Substituting these expressions into (2.10) maps the lens-shaped domain Ω to the square $[-1, 1]^2$. Such a mapping is required in order to apply the Gauss-Legendre quadrature rule described previously (having chosen $\omega(x) \equiv 1$). We obtain

$$\int_{-y_*}^{y_*} \int_{-1}^{1} f\Big(\frac{b(y) - a(y)}{2} \tilde{x} + \frac{a(y) + b(y)}{2}, y\Big) \frac{b(y) - a(y)}{2} d\tilde{x} dy$$

$$= \int_{-1}^{1} \int_{-1}^{1} f\Big(\frac{b(y_*\tilde{y}) - a(y_*\tilde{y})}{2} \tilde{x} + \frac{a(y_*\tilde{y}) + b(y_*\tilde{y})}{2}, y_*\tilde{y}\Big) \times \frac{y^*(b(y_*\tilde{y}) - a(y_*\tilde{y}))}{2} d\tilde{x} d\tilde{y}. \quad (2.11)$$

Now we can approximate the first of the integrals in (2.11) by applying a quadrature rule described in the previous section:

$$\int\int_{\Omega} f \ dxdy \approx \sum_{j=1}^{N} w_j \int_{-1}^{1} f\Big(\frac{b(y_* y_j) - a(y_* y_j)}{2} \tilde{x} + \frac{a(y_* y_j) + b(y_* y_j)}{2}, y_* y_j\Big) \times \frac{y^*(b(y_* y_j) - a(y_* y_j))}{2} d\tilde{x}$$

where w_j denotes the quadrature weights in the y-direction. We can now do the same for the other integral to obtain a quadrature rule for the double integral, and hence obtain

$$\int\int_{\Omega} \ dxdy \approx \sum_{j=1}^{N} \sum_{k=1}^{M} w_j \tilde{w}_k f\Big(\frac{b(y_* y_j) - a(y_* y_j)}{2} x_k + \frac{a(y_* y_j) + b(y_* y_j)}{2}, y_* y_j\Big) \times \frac{y_*(b(y_* y_j) - a(y_* y_j))}{2} \quad (2.12)$$

where $\tilde{w_k}$ denotes the quadrature weights in the x-direction. In Figure 2.4 we plot the points $a(y_* y_j)$ and $b(y_* y_j)$ when using a Newton-Cotes quadrature in the y-direction. Expression (2.12) will be the starting point for our study of simple quadrature rules for the lens, given by the tensor product of 1-dimensional quadrature rules such as Newton-Cotes and Gauss. We can now compare these numerical schemes and deduce which one would be the best to use by balancing accuracy with the number of quadrature points required to attain such accuracy.

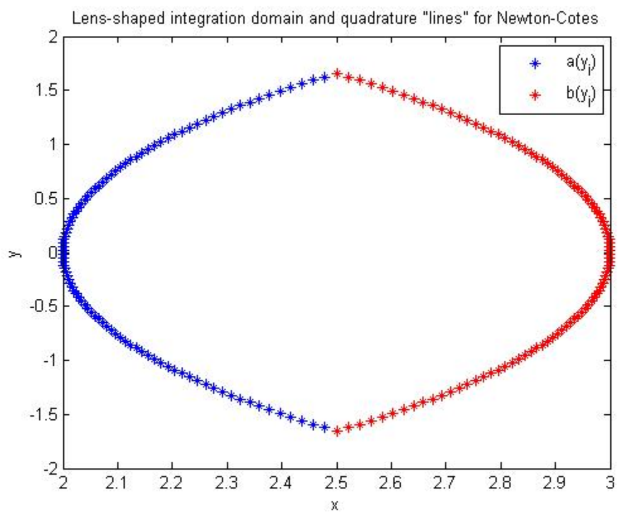


Figure 2.4: Plot of points $a(y_* y_j)$ and $b(y_* y_j)$.

2.3.2 Comparison of Quadrature Rules

As described earlier, we can now compare the numerical integration schemes by choosing to integrate the test function

$$f = \chi_{\text{lens}}(x) = \begin{cases} 1 & \text{if } x \in \text{lens} \\ 0 & \text{if } x \notin \text{lens} \end{cases}$$

which we know integrates exactly to the area of the lens-shaped domain, whose formula was given in Section 2.2.1. Substituting this test function into (2.12) yields the simpler expression,

$$\int\int_{\Omega} 1 \; dxdy \approx \sum_{j=1}^{N} \sum_{k=1}^{M} w_j \tilde{w_k} \frac{y_*(b(y_* y_j) - a(y_* y_j))}{2}. \tag{2.13}$$

Notice how this expression only depends on the y-direction quadrature points $y_j \;\; j = 1, ..., N$. To analyse different quadrature schemes it would be wise to use a function for which there will be both a x- and y-direction dependence, but this is a good starting point.

2.3.2.1 Newton-Cotes for y-direction and Gauss for x-direction

This scheme will involve taking equally spaced points in the y-direction with weights corresponding to those defined by the Newton-Cotes quadrature rule, and Gauss points and weights in the x-direction. In the previous section we saw how Gauss quadrature is more accurate than its Newton-Cotes equivalent so it may seem futile to even consider such a scheme for approximating integrals over more complex domains, but quadrature properties in 1-dimension do not easily extend to higher dimensions so it is worth considering this numerical integration scheme.

In particular we will choose to use the composite trapezium rule as our Newton-Cotes quadrature and increase the number of subintervals N that divide up the interval $[a, b]$. The reason for not dealing with Newton-Cotes formulas of increasing order is to avoid finding the corresponding weights, which leads to an ill-conditioned problem involving the solution of a Vandermonde system whose matrix becomes close to singular as we increase the order of the Newton-Cotes quadrature.

Figure 2.5 shows the relative error in the l_2 norm between the numerical approximation and the exact solution versus the number of quadrature points taken over the lens-shaped domain; for which we chose $R = 3$ and $d = 5$. It is only worth considering

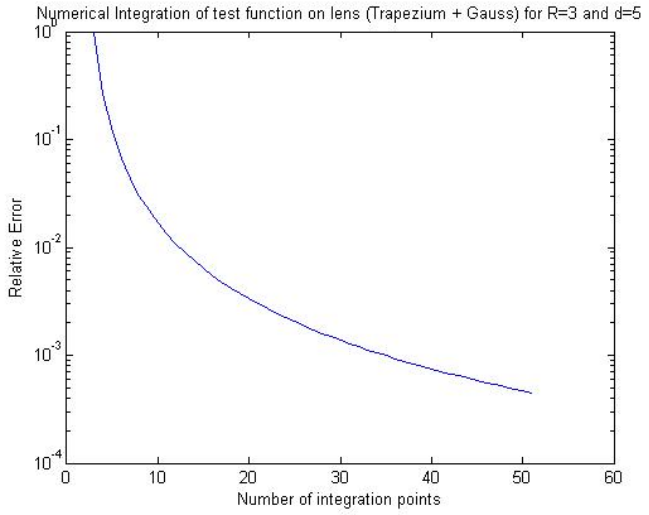


Figure 2.5: Numerical integration of test function on lens using composite trapezium + Gauss.

and increasing the number of y-direction quadrature points since in the x-direction (for which we are using Gauss quadrature) the test function satisfies

$$\sum_{k=1}^{M} w_k \approx \int_{-1}^{1} 1\ dx = 2.$$

But Gauss-Legendre weights always satisfy

$$\sum_{k=1}^{M} w_k = \text{length of interval [-1,1]} = 2. \tag{2.14}$$

So all that is required is that we take one Gauss point and weight in the x-direction, and so $w_{k=1} = 2$ in (2.13) which can thus be reduced to a single sum approximating a 1-D integral. Notice how it is required to take over 40 quadrature points to attain a relative error of approximately 1×10^{-3}. Clearly such a numerical integration scheme is far from ideal, especially considering that we are integrating the simplest non-trivial function. If we were to apply such a scheme to integrate much more complicated functions such as the product of radial basis functions, which occur when approximating the entries of the stiffness matrix and load vector to solve the Galerkin problem, then we should not expect to obtain an accurate approximation without having to take an unreasonably large number of quadrature points. This motivates our need to find better schemes.

2.3.2.2 Gauss for Both x- and y-directions

We will now try a quadrature rule which may have been the obvious choice to consider in the first place: performing Gauss quadrature on both the x-direction line integral as well as in the y-direction. Indeed as discussed for the 1-D case, Gauss quadrature provides higher order accuracy than Newton-Cotes for the same number of quadrature points. It is thus reasonable to believe that this scheme will yield higher order accuracy for the 2-D case, namely for the lens-shaped domain considered so far. As before, taking only one quadrature point and weight in the x-direction is sufficient to always satisfy (2.14) so we focus on increasing the number of quadrature points in the y-direction.

Figure 2.6 now shows the relative error between the numerical approximation and the exact solution versus the number of quadrature points taken for this new quadrature rule, with the same parameters as before. Figure 2.6 should be compared with Figure 2.5: in the former we attain close to machine error precision with only 16 or so quadrature points, a dramatic improvement from the previous scheme which required double the number of points for a significantly larger relative error. It is worth noticing that we never actually attain Matlab's machine error precision of 1×10^{-16} due to the dominance of rounding errors, but for most purposes and applications this will not be a problem.

2.3.2.3 Quadrature using Fekete Points

Fekete points are known to have near-optimal interpolation properties, and for polynomials of degree $N > 10$ they are the best interpolation points known for the triangle. Furthermore it has been shown in [4] that tensor product Gauss-Lobatto points (in

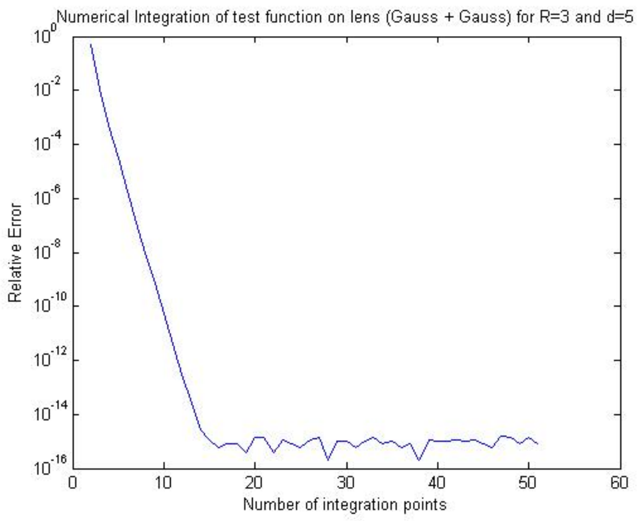


Figure 2.6: Numerical integration of test function on lens using Gauss + Gauss.

1-D these correspond to Gauss-type points which include the end-points of the interval) are Fekete points for the square. Given that we map the lens to a tensor product domain it would be interesting to know how a quadrature based on Fekete points on the square would fare against those analysed previously.

Figure 2.7 plots the relative error in the l_2-norm between the numerical approximation and the exact solution for the test function versus the number of quadrature points taken over the lens-shaped domain for which we have taken the same values as previously, namely $R = 3$ and $d = 5$, in order to compare it with the other quadrature rules. It appears that such a scheme is very inefficient, requiring roughly 108 quadrature points [1] to attain a relative error of 10^{-1}. Attempting to find Fekete points directly for the lens-shaped domain would certainly yield better results than those shown in Figure 2.7, but finding such points for arbitrary 2-D domains is a complicated problem. Approaches for doing so have been suggested in [13] and [3]. Such approaches are hard to implement when one is faced with domains that have different dimensions and areas, which is the case for our circular and lens-shaped domains, so we will not pursue this approach any further.

[1]In Figure 2.7 and all other figures where this notation is present, "(x9)" denotes the fact that we have fixed the number of "y"-quadrature points to $N = 9$, so the number of quadrature points is $9M$ where M denotes, as usual, the number of quadrature points in the x-direction.

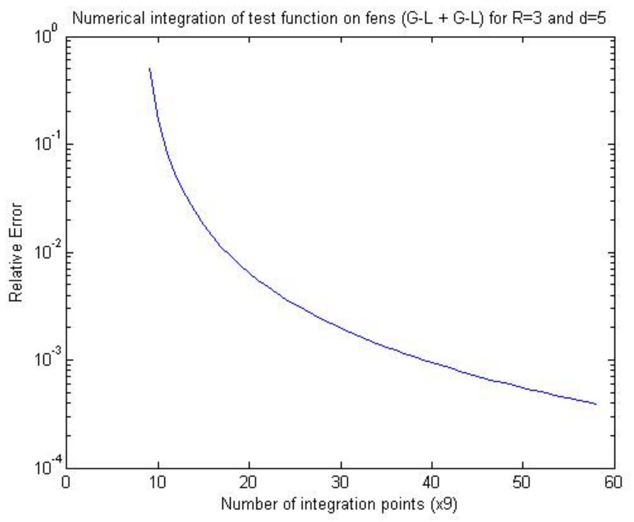


Figure 2.7: Numerical integration of test function on lens using Gauss-Lobatto + Gauss-Lobatto (Fekete points on square).

2.3.2.4 Remarks on Quadrature in Polar Coordinates

Given that the lens-shaped domain is the intersection of two circular supports, it would be interesting to apply quadrature schemes once we converted the variables x and y into polar coordinates. One possible set-up would be to map the lens-shaped domain onto the unit disc and perform a quadrature rule on this new domain (we will be analysing quadrature rules on circles in the next section). Such a mapping is given by the substitution,

$$y = y_* \sin(\theta)$$

and

$$x = \frac{b\big(y_* \sin(\theta)\big) - a\big(y_* \sin(\theta)\big)}{2b\big(y_* \sin(\theta)\big)} R\cos(\theta) + \frac{a\big(y_* \sin(\theta)\big) + a\big(y_* \sin(\theta)\big)}{2}.$$

However, as reported in [5], the Jacobian resulting from such a transformation yields a complicated function which makes this approach inefficient.

2.3.2.5 Problems with Quadrature on Lens

One major problem with the above quadrature rule is that convergence becomes dramatically slower as we let $d \to 0$ for fixed radii i.e. for circles getting progressively closer to one another, or if we increase the radii of the circles whilst keeping d fixed. One simple reason for this behaviour is that the area of the lens increases as d is decreased or as both radii are increased (all else remaining fixed), and we thus require an increasing number of quadrature points to attain the sort of accuracy mentioned before. In Figure 2.8 we show how this slow convergence arises for $R = 3$ and $d = 0.25$, having still taken Gauss points and weights in both the x- and y-directions as before. Another reason for the occurence of such a problem is the inherent discontinuity of the

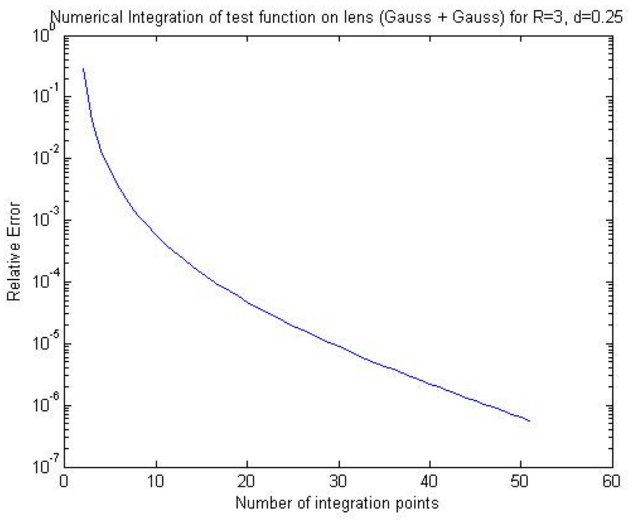


Figure 2.8: Numerical integration of test function on lens using Gauss + Gauss for small $d > 0$.

derivative of the integrand once the lens has been mapped to a square. Consider, as before, the integration of the test function for the lens-shaped domain. As mentioned previously this reduces to the computation of a 1-D integral as follows:

$$\int_{-y_*}^{y_*} \int_{d-\sqrt{R^2-y^2}}^{\sqrt{R^2-y^2}} 1 \; dxdy = \int_{-y_*}^{y_*} \left(2\sqrt{R^2-y^2} - d\right) \, dy$$

$$= y_* \int_{-1}^{1} \left(2\sqrt{R^2-(y_*\tilde{y})^2} - d\right) \, d\tilde{y}.$$

Letting $f(\tilde{y}) = y_*\left(2\sqrt{R^2-(y_*\tilde{y})^2} - d\right)$, we get the 1-D quadrature

$$\int_{-1}^{1} f(\tilde{y}) \, d\tilde{y} \approx \sum_{j=1}^{N} w_j f(y_j).$$

We now consider the following theorem, taken from [16].

Theorem 2.3.1 *Let Gauss quadrature be applied to a function* $f \in C[-1,1]$. *If* $f, f', ..., f^{(k-1)}$ *are absolutely continous on* $[-1,1]$ *and* $||f^{(k)}||_T = V < \infty$ *for some*

$k \geq 1$, where $||\cdot||_T$ denotes the Chebyshev weighted 1-norm given by

$$||u||_T = \left|\left|\frac{u'(x)}{\sqrt{1-x^2}}\right|\right|_1.$$

Then for each $n \geq \frac{k}{2}$,

$$|E_n(f)| \leq \frac{32V}{15\pi k(2n+1-k)^k}$$

with $E_n(f)$ as before. If f is analytic with $|f(z)| \leq M$ in the region bounded by the ellipse with foci ± 1 and major and minor semiaxis lengths summing to $\rho > 1$, then for each $n \geq 0$,

$$|E_n(f)| \leq \frac{64M}{15(1-\rho^{-2})\rho^{2n+2}}.$$

This theorem gives an upper bound for the difference between the intergral over the lens-shaped domain and its approximation. The first inequality of Theorem 2.3.1 suggests that integrals whose integrands have higher derivatives that are discontinuous in $[-1,1]$ would converge more slowly. The parameter ρ in the second inequality is a good indicator of how fast the approximation will converge to the exact integral. To find its value, we need to find the semiaxis lengths a and b such that $f(z)$ is analytic in the ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. Differentiating f, we get

$$f'(\tilde{y}) = -y_*^3\tilde{y}\left(\frac{1}{\sqrt{R^2 - y_*^2\tilde{y}^2}} + \frac{1}{\sqrt{R^2 - y_*^2\tilde{y}^2}}\right). \tag{2.15}$$

Expression (2.15) (and in fact all the higher order derivatives of f) is discontinuous when

$$R^2 - y_*^2\tilde{y}^2 = 0 \Rightarrow \tilde{y}^2 = \frac{R^2}{R^2 - x_*^2}$$

since $y_*^2 = R^2 - x_*^2$. So we choose

$$a = \frac{R}{\sqrt{R^2 - x_*^2}} = \frac{R}{\sqrt{R^2 - \frac{d^2}{4}}} = \frac{2}{\sqrt{4 - (\frac{d}{R})^2}}$$

as $x_* = \frac{d}{2}$. Given that the foci are centred at ± 1 we require that $b^2 = a^2 - 1$ and so we must have

$$b = \sqrt{a^2 - 1} = \frac{\frac{d}{R}}{\sqrt{4 - (\frac{d}{R})^2}}.$$

Hence we obtain

$$\rho = a + b = \frac{2 + \frac{d}{R}}{\sqrt{4 - (\frac{d}{R})^2}}.$$

This expression for ρ gives us two important results, namely:

- As $\frac{d}{R} \to 2$ (i.e. the area of the lens shrinks to zero as the circles cease to intersect), $\rho \to \infty \Rightarrow$ faster convergence.
- As $\frac{d}{R} \to 0$ (i.e. the circles get closer together and the lens tends to the circle), $\rho \to 1 \Rightarrow$ slower convergence.

This explains the sort of behaviour we observed in Figure 2.8 for the lens quadrature applied to the test function. We back-up these theoretical results numerically by plotting the quadrature error for $R = 3$ and $d = 5, 7, 7.9995$. which are shown in Figures 2.9(a)–(c) respectively. Notice that $a, b \to \infty$ as $\frac{d}{R} \to 2$, in which case the

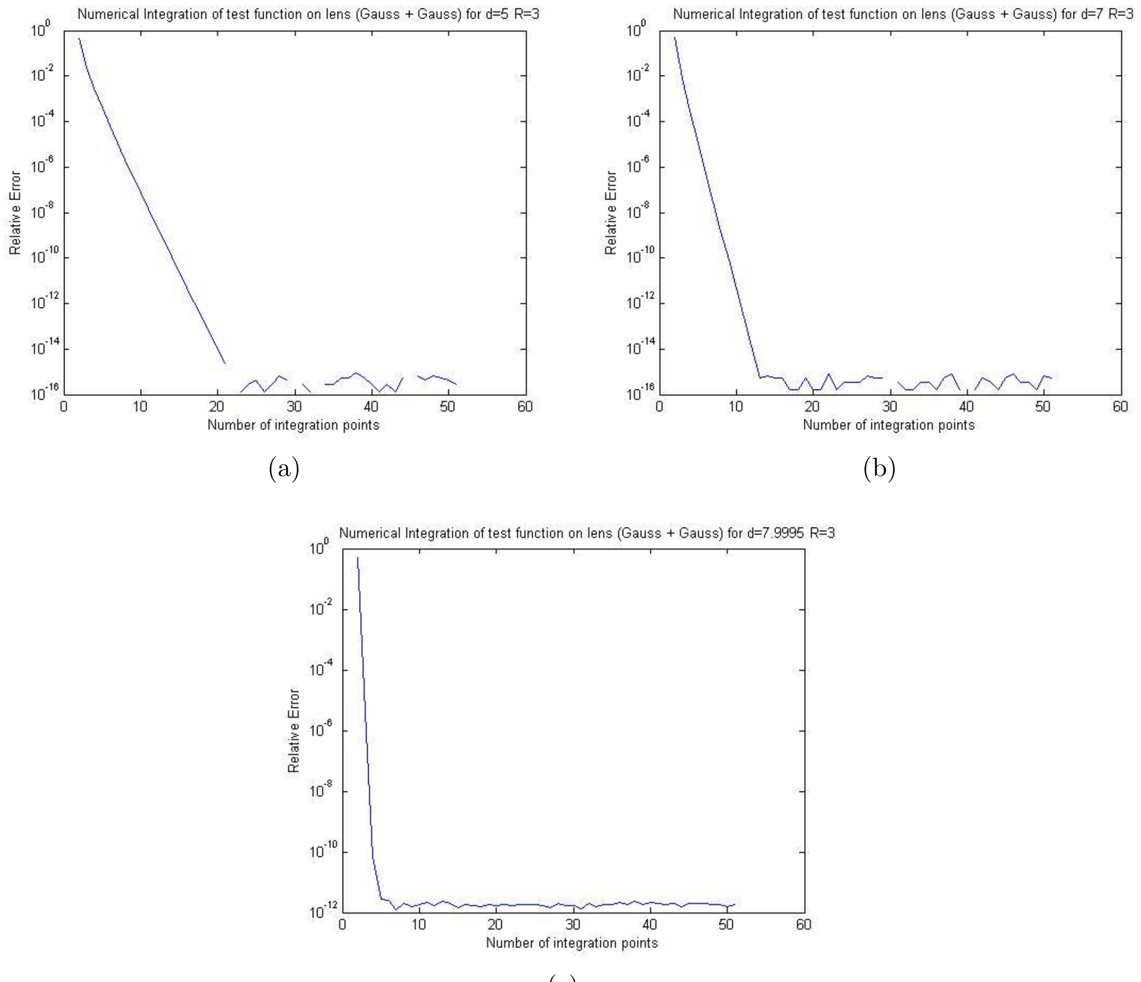


Figure 2.9: Numerical integration of test function on lens using Gauss + Gauss for respectively $d = 5, 7$ and 7.9995.

integrand is analytic over the infinitely large ellipse. Whereas when $\frac{d}{R} \to 0$, we have $b \to 1$ and $a \to 0$ so discontinuities of the higher order derivatives of the integrand "approach" the integration domain $[-1,1]$ and will be a cause for slower convergence by Theorem 2.3.1.

Though these results only apply for the simple case of a constant integrand it does give us some insight as to why this quadrature rule may not be so efficient for 2-D integrals of more complex integrands, namely the integration of (products of) radial basis functions which is our main application of interest.

2.4 Numerical Integration Schemes on the Circle

We will now focus our attention on developing efficient quadrature rules for the circle. These will be required to approximate the entries of the load vector $\mathbf{b}$ as well as the diagonal entries of the stiffness matrix $\mathbf{A}$. As before we will assume without loss of generality that the circle is centred at the origin, as a straightforward translation would always map it to this simpler case.

2.4.1 First Approach

As was done in the previous section for the lens, a first approach at approximating the double integral could be done by contructing quadrature rules in the (x,y)-plane. We first apply a 1-dimensional quadrature rule on a line integral dependent only on y and then apply a quadrature rule along the line integrals dependent only on x, making the overall quadrature rule a tensor product of the simple 1-d rules as before. This scheme is illustrated in Figure 2.10. In more mathematical terms we have

$$\int_{-R}^{R}\int_{-\sqrt{R^2-y^2}}^{\sqrt{R^2-y^2}} f(x,y)dxdy = \int_{-R}^{R}\int_{-a(y)}^{a(y)} f(x,y)dxdy \qquad (2.16)$$

where R is the radius of the circle and $a(y) := \sqrt{R^2-y^2}$. We now map the circular domain onto the square $[-1,1]^2$ by making the substitutions:

$$y = R\tilde{y} \text{ and } x = a(y)\tilde{x}$$

which enables us to apply the Gauss-Legendre quadrature rule as before. Substituting these into (2.16) we get

$$\int_{-R}^{R}\int_{-a(y)}^{a(y)} f(x,y)\ dxdy = \int_{-1}^{1}\int_{-1}^{1} f(a(R\tilde{y})\tilde{x}, R\tilde{y}) \cdot a(R\tilde{y})R\ d\tilde{x}d\tilde{y}$$

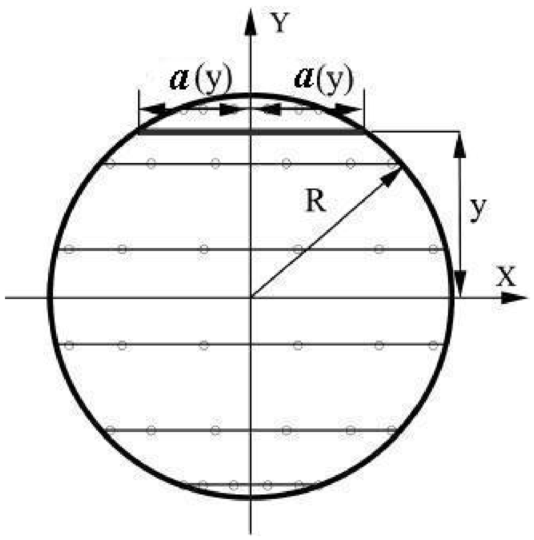


Figure 2.10: Circular-shaped domain of integration.

with its corresponding tensor product quadrature rule,

$$\int_{-R}^{R}\int_{-a(y)}^{a(y)} f(x,y)\ dxdy \approx \sum_{j=1}^{N}\sum_{k=1}^{M} w_j\tilde{w}_k f(a(Ry_j)x_k, Ry_j)\cdot a(Ry_j)R. \tag{2.17}$$

2.4.2 Comparison of Quadrature Rules

For the lens we considered the simplest possible function for which we can find an exact solution i.e.

$$f = \chi_{\text{lens}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega. \\ 0 & \text{if } \mathbf{x} \notin \Omega. \end{cases}$$

As described previously, using such a function does not make full use of quadrature points in both x- and y-directions. But the complicated integration bounds arising when integrating over the lens-shaped domain made it hard to consider even slightly more complex functions as an exact solution would then have been hard to find. For the simpler circular domain however, exact solutions for such integrands can easily be found. We will consider the function $f(x,y) = x^2y^2$ which can be integrated exactly over a circle of radius 1 for which we get

$$\int_{-1}^{1}\int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} x^2y^2\ dxdy = \frac{\pi}{24}.$$

2.4.2.1 Gauss for Both x- and y-directions

We will now consider a numerical integration scheme taking Gauss points and weights in both the x- and y-directions, in the same way as we did for the lens. We would expect such a scheme to be good given that we no longer have the constraint of having to keep the centres of two circles far enough from each other in order to have an efficient quadrature rule. Figure 2.11 plots the relative error in the l_2-norm between the exact solution and the approximation against the number of quadrature points taken over the circular domain with radius $R = 1$. Notice how this scheme

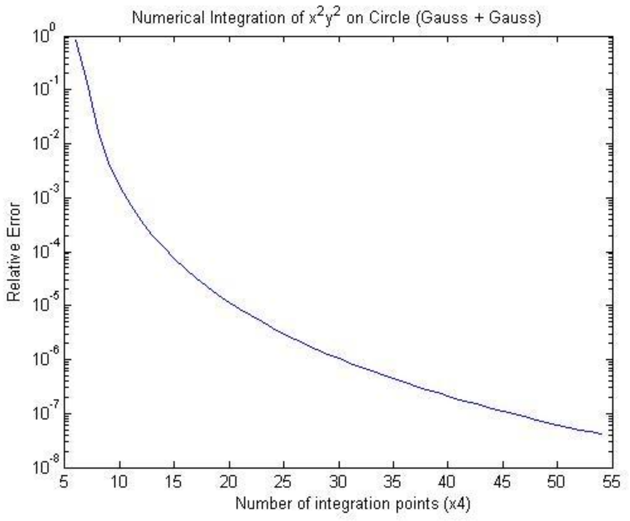


Figure 2.11: Numerical integration of x^2y^2 on circle using Gauss + Gauss.

requires about 200 quadrature points to achieve a relative error of 10^{-7}. Though such accuracy is satisfactory for most purposes, the large number of quadrature points that are needed to attain such accuracy suggests the computational cost of this quadrature rule is expensive and will be moreso for more complex functions.

2.4.2.2 Quadrature in Polar Coordinates

Switching to polar coordinates appears to be the natural way of numerically integrating a function over a circular domain, especially considering that the Jacobian of such a transformation is very simple. We make the usual change of variables $x = r\cos(\theta)$ and $y = r\sin(\theta)$ with the integral over the circle Ω now becoming

$$\int\int_{\Omega} f(x,y)dxdy = \int_0^{2\pi}\int_0^R f(r\cos(\theta), r\sin(\theta))rdrd\theta. \tag{2.18}$$

We now consider taking equally spaced angles $\theta_j \ \ j = 1, ..., M$ in the interval $[0, 2\pi]$ using a composite trapezium rule to generate the corresponding quadrature weights. And for each such angle θ_j kept fixed we consider the line integral on the interval $0 \le r \le R$, independent of θ, and approximate it using a 1-D Gauss quadrature rule with quadrature points r_k $k = 1, ..., N$. Approximating integral (2.18) using such a quadrature rule yields

$$\int\int_\Omega f(x,y)dxdy \approx \sum_{j=0}^{M}\sum_{k=0}^{N} w_j \tilde{w_k} f(r_k\cos(\theta_j), r_k\sin(\theta_j))r_k$$

where w_j are the quadrature weights for the θ-dependent integral approximation, and $\tilde{w_k}$ those for the r-dependent approximation. In Figure 2.12 we plot the relative error between the exact solution and the approximation using the quadrature above for the same function and parameters as before. Clearly such a scheme is a dramatic

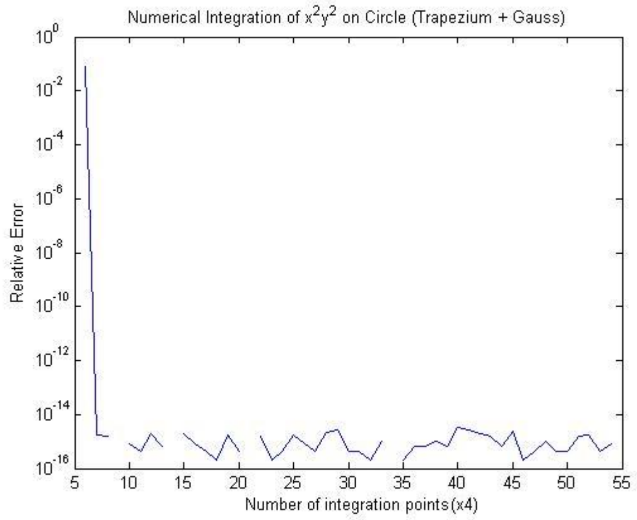


Figure 2.12: Numerical integration in polar coordinates of x^2y^2 on circle using composite trapezium rule for θ and Gauss for r.

improvement over the previous one: indeed, we achieve close to machine-error precision with only about 24 quadrature points, hence obtaining much greater accuracy with only a fraction of the number of quadrature points required for the previous numerical scheme. It is worth mentioning here that our choice of quadrature for the circle wasn't completely arbitrary: Reference [15] suggests that the composite trapezium rule often converges very quickly for periodic functions. Given that all of the integrands considered in this dissertation are inherently 2π-periodic over the circular domain with respect to θ, it was a natural choice to consider.

For this reason, if we were to instead apply Gauss quadrature for θ we would actually need significantly more quadrature points to attain the same accuracy. Recall that for the lens we showed that performing Gauss quadrature for both x and y variables provides greater accuracy than using a composite trapezium rule for one and Gauss for the other. This suggests that it is not straightforward to generalise which quadrature rule would be good and which would not be for arbitrary 2-D domains and coordinate systems.

2.5 Quadrature for Boundary Supports

Until now we have implicitely only discussed numerical integration schemes on lens-shaped domains and circles that are fully contained inside the solution domain [2] $\Omega = [-1, 1]^2$ (call them *interior supports*). However, many of the integrands whose integrals we seek to approximate will not have their supports fully contained inside the solution domain (call them *boundary supports*) as shown in Figure 2.13. One

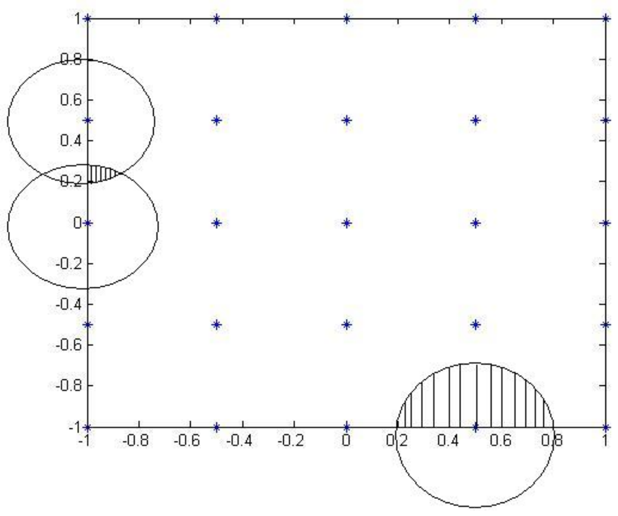


Figure 2.13: Example of supports not fully contained in boundary (i.e. boundary supports).

question worth asking is: how troublesome will these boundary supports be when approximating the corresponding integral and how efficienct would the quadrature rules described in the previous section be for these? To answer this question, we consider the problem of integrating $f(x, y) \equiv 1$ over a circular domain whose radius is large enough that it contains the entire solution domain $\Omega = [-1, 1]^2$. The problem set-up is shown in Figure 2.14. We accommodate for integrals with boundary supports by multiplying the integrand by the characteristic function for the solution domain,

$$\chi_\Omega(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega \\ 0 & \text{if } \mathbf{x} \notin \Omega. \end{cases}$$

[2]Recall that the solution domain is the one in which we seek to numerically solve the Helmholtz equation

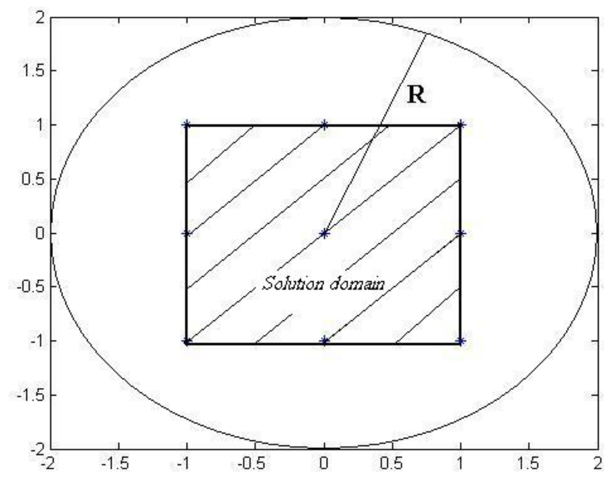


Figure 2.14: Problem set-up to test numerical integration of integrals with boundary supports. Area of square is $2 * 2 = 4$.

Integrating $f \cdot \chi_\Omega$ over the circular domain of the set-up above should yield an approximation of the area of the square solution domain Ω that it contains. Figure 2.15 shows the relative error between the exact area and its approximation, plotted against the number of quadrature points, when choosing the radius of the circle containing the domain to be $R = 2$. As before, we use the circle quadrature in polar coordinates described in the previous section to compute the approximation. Notice how the convergence is now extremely slow, requiring more than 4900 quadrature points to attain a relative error of 1×10^{-2}. It appears that the efficiency of the circle quadrature in polar coordinates (and in fact the lens quadrature) discussed in the previous section for supports fully contained inside the solution domain vanishes completely for supports that are not fully contained in it. The discontinuous nature of an integrand having been multiplied by the characteristic function may well play a part in explaining such poor results. But the non-optimality of the choice of quadrature points for supports that are not fully contained in the domain would be the main reason for the reduced accuracy of the quadrature rules. They were in some way “designed” for a particular 2-D domain, so changing the latter would clearly have a huge effect.

Given that a potentially significant number of entries of the stiffness matrix and load vector (1.9) will involve the approximation of integrals over boundary supports, we should expect these to create a significant source of error in the numerical solution of the Helmholtz equation.

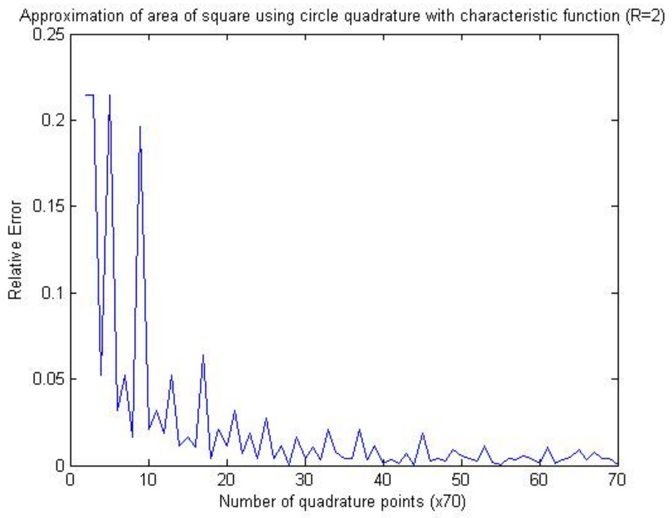


Figure 2.15: Numerical integration in polar coordinates of $f(x, y) \cdot \chi_\Omega$ where $f(x, y) \equiv 1$ on circle using composite trapezium rule for θ and Gauss for r. Exact value of integral is 4.

2.6 Comparison with Quadrature on Whole Domain

Now that we have discussed numerical integration schemes on the circle and lens-shaped domains as well as the multiple problems associated with their use, it would be interesting to now compare them with corresponding numerical integration schemes over the entire solution domain $\Omega = [-1, 1]^2$. The integrands that we will use to perform these comparisons are the ones required to evaluate the entries of the stiffness matrix and load vector in (1.9), these are given by

$$\int_{[-1,1]^2} [\nabla\Phi(\mathbf{x} - \mathbf{x}_j) \cdot \nabla\Phi(\mathbf{x} - \mathbf{x}_i) + \Phi(\mathbf{x} - \mathbf{x}_j)\Phi(\mathbf{x} - \mathbf{x}_i)]d\mathbf{x} \tag{2.19}$$

and

$$\int_{\text{lens}} [\nabla\Phi(\mathbf{x} - \mathbf{x}_j) \cdot \nabla\Phi(\mathbf{x} - \mathbf{x}_i) + \Phi(\mathbf{x} - \mathbf{x}_j)\Phi(\mathbf{x} - \mathbf{x}_i)]d\mathbf{x} \tag{2.20}$$

which are equal for interior supports since products and gradients of RBFs have their support in lens-shaped domains, but different for supports that intersect with the boundary since the integrand of (2.20) will have to be multiplied by the characteristic function χ_Ω. Similarly, we will compare the schemes associated with the integrals

$$\int_{[-1,1]^2} f(\mathbf{x})\Phi(\mathbf{x} - \mathbf{x}_i)d\mathbf{x} \tag{2.21}$$

and

$$\int_{\text{circle}} f(\mathbf{x})\Phi(\mathbf{x}-\mathbf{x}_i)d\mathbf{x} \tag{2.22}$$

which, again, are equal for interior supports but different for boundary supports as the integrand of (2.22) will be multiplied by χ_Ω. We choose $f(x,y) = \cos(\pi x)\cos(\pi y)$ when approximating these integrals.

2.6.1 Interior Supports

In Figures 2.16(a)–(d) we plot the approximations of (2.19) and (2.20) versus the number of quadrature points using respectively the numerical integration scheme over the whole solution domain (in red) and the one over the lens-shaped domain (in blue). We have chosen $R = 1$ and respectively $d = 0.1, 0.25, 1.5$ and 1.75. Notice that the radii of the circles making up the lens are such that the lens-shaped domain is fully contained in Ω. As expected, when we increase the number of quadrature points the approximations generated by the different quadratures appear to get arbitrarily close to one another. In Figures 2.16(a) and 2.16(b), notice how for small values of d the numerical integration over the lens does not provide any significant advantage over the one for the whole domain, both requiring a similar number of quadrature points to attain a certain accuracy (we will say that it "converges" [3]). Part of the reason why approximating (2.20) for small d requires so many quadrature points is that the area of integration is increasing in size and thus requires more quadrature points. But there is also another reason: recall from Theorem 2.3.1 that we were able to determine the convergence rate of a 1-D quadrature rule by considering the ellipse in which the integrand was analytic and bounded. We were able to do so for a constant integrand, for which the 2-D integral could be reduced to the 1-D case. We must now consider the 2-D function $g(x,y)$ given by the integrand of (2.19), we then have

$$\int_{\text{lens}} g(x,y)\ dxdy = \frac{y_*}{2}\int_{-1}^{1}\int_{-1}^{1}\left(2\sqrt{R^2 - y_*^2\tilde{y}^2} - d\right)\hat{g}(\tilde{x},\tilde{y})\ d\tilde{x}d\tilde{y}$$

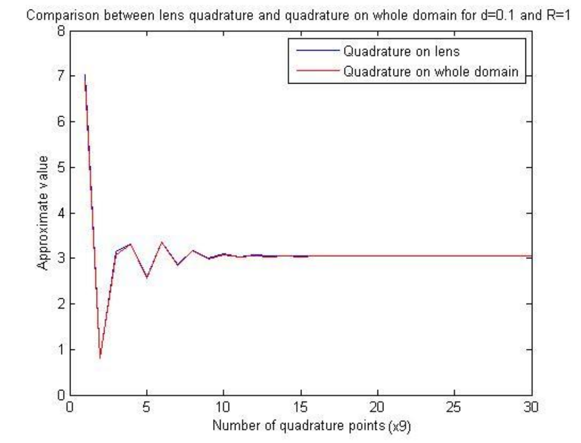
where

$$\hat{g}(\tilde{x},\tilde{y}) = g\left(d - \sqrt{R^2 - y_*^2\tilde{y}^2}, y_*\tilde{y}\right).$$

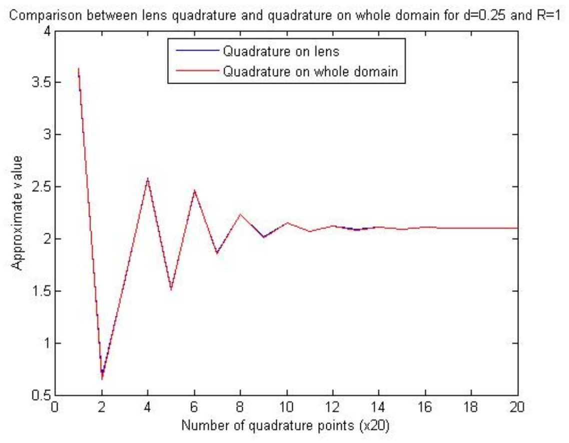
If we now define

$$F(\tilde{y}) = \left(2\sqrt{R^2 - y_*^2\tilde{y}^2} - d\right)\int_{-1}^{1}\hat{g}(\tilde{x},\tilde{y})\ d\tilde{x}$$

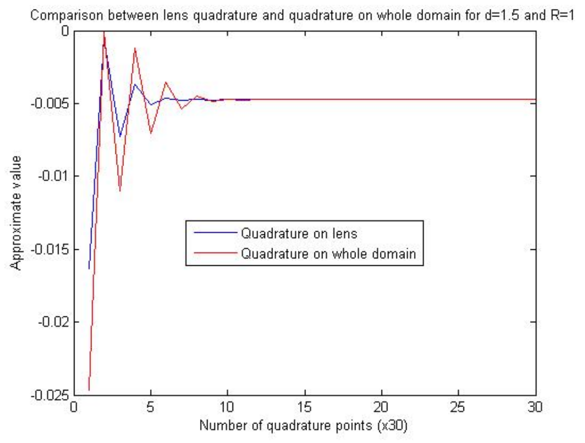
[3]The exact values of the integrals aren't known but we assume that as we increase the number of quadrature points the approximation get arbitrarily close to the exact value



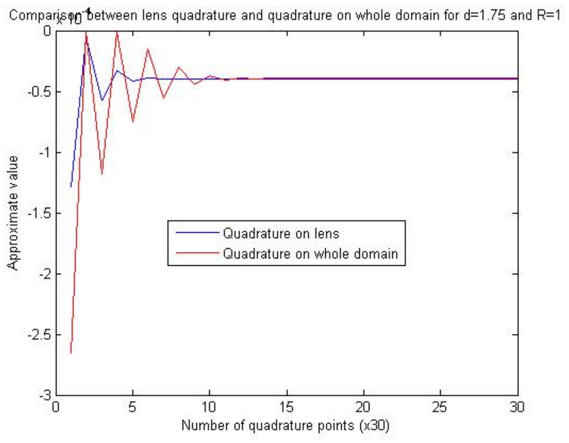
(a)



(b)



(c)



(d)

Figure 2.16: Comparison between quadrature on interior lens (blue) and quadrature on whole domain (red) for $R = 1$ and respectively $d = 0.1, 0.25, 1.5$ and 1.75.

and note that this is the same integrand as before with the addition of the integral $\int_{-1}^{1} \hat{g}(\tilde{x}, \tilde{y})\, d\tilde{x}$ which makes the analysis much more tedious. However, given that the (higher order) derivative of this integrand will have the same discontinuities as before in addition to those of the RBF (which occur at the centre of their support as well as the boundary of their support), it is reasonable to assume this would be another cause for slower convergence. Namely when the discontinuities are inside the domain of integration.

For larger values of d (when $d > R$ and the discontinuity is outside of the integration domain) the lens quadrature requires significantly less points to converge as shown in Figures 2.16(c) and 2.16(d).

In Figure 2.17 we plot the approximations of (2.21) and (2.22) versus the number of quadrature points using respectively the numerical integration scheme over the whole solution domain and the one over a circular domain centered at the origin. The support has been chosen so that it is fully contained in the solution domain. Again, notice how the circle quadrature requires significantly less points than the

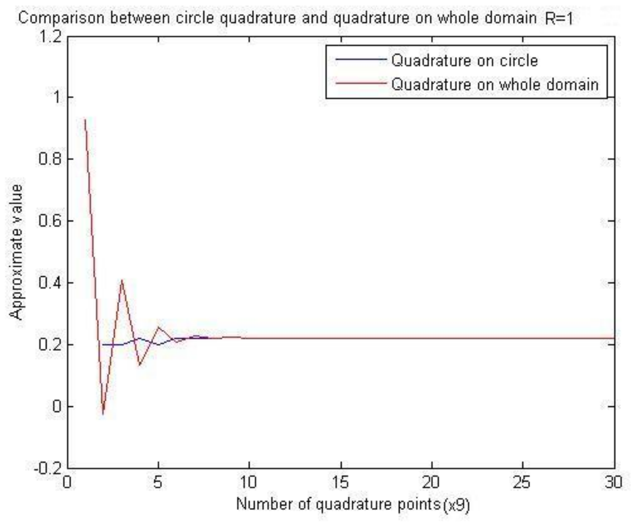


Figure 2.17: Comparison between quadrature on interior circle (blue) and quadrature on whole domain (red) for $R = 1$.

scheme over the entire solution domain to converge.

2.6.2 Boundary Supports

We can now compare the numerical integration scheme over the whole domain with that of the lens and circle in the case when the integrands (2.20) and (2.22) have been multiplied by the characteristic function χ_Ω to accommodate for boundary supports. In Figure 2.18 we plot the approximations of (2.19) and (2.20) as before, but we now choose $R = 2$ and $d = 1$ so that the lens is not fully contained in the solution domain. As it was observed for simpler integrands, the efficiency of the quadrature rule is significantly deteriorated and convergence is thus very slow due to the discontinuity of the integrand, and more importantly the non-optimality of the quadrature points. Clearly, the quadrature rule over the whole domain is much more efficient and accurate. A similar phenomenon occurs for the approximations of (2.21) and (2.22) when we choose $R = 2$ as shown in Figure 2.19.

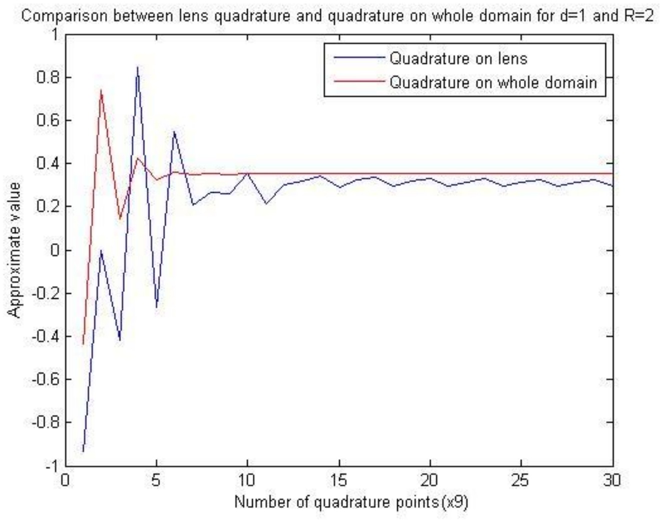


Figure 2.18: Comparison between quadrature on boundary lens (blue) and quadrature on whole domain (red) for $R = 2$ and $d = 1$.

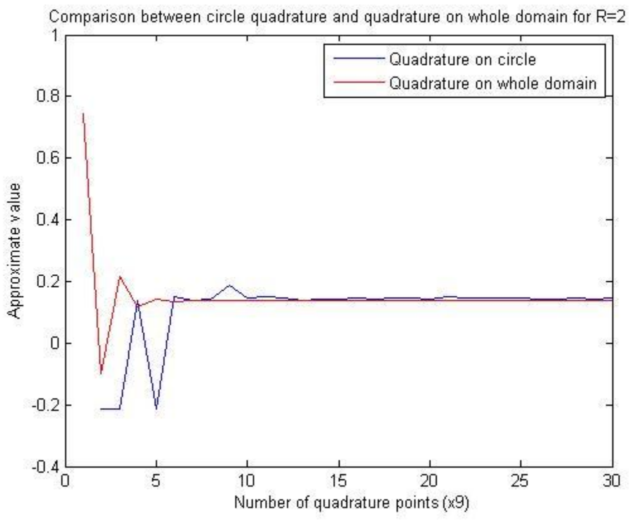


Figure 2.19: Comparison between quadrature on boundary circle (blue) and quadrature on whole domain (red) for $R = 2$.

Our analysis so far suggests that many problems can arise when using the lens and circle quadratures. For the lens, small values of d can lead to slower convergence of the associated quadrature rule. The main issue however is the approximation of integrals over boundary supports, which affects both the lens and circle quadratures. In such cases, the numerical integration scheme over the whole domain would in fact produce more accurate results with less quadrature points. Substituting the (discontinuous) characteristic function by a continuous approximation of the form

$$c(x, y) = \frac{1}{(1 + x^{2k})(1 + y^{2k})}$$

(where the parameter k is such that the larger it is, the better it approximates the characteristic function) only marginally improves accuracy and does not really solve the problem. A plot of this approximation for $k = 7$ is shown in Figure 2.20. It was

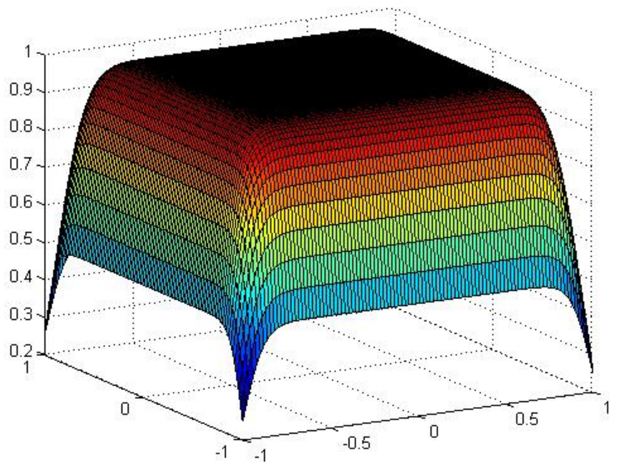


Figure 2.20: Continuous approximation of the characteristic function with $k = 7$.

suggested in [5] and [6] to consider boundary elements case by case and apply specific quadrature rules tailored for each one of these but this seems like a tedious way of going about solving this problem and hard to implement in practice.

Chapter 3

Convergence

Now that we have described the different numerical integrations schemes, compared them and mentioned problems arising when using them in particular cases we can now discuss the Helmholtz equation and convergence issues linked to its solution. Theoretical and practical issues related to convergence of the Galerkin problem for the Helmholtz equation are discussed in more detail in [10], [18], [19] and [20]. All of the theorems in this chapter have been taken from and are proved in [19] and [20].

3.1 Least-Squares Problem

3.1.1 Theoretical Aspects

Before moving on to solving the Galerkin problem arising from the Helmholtz equation and analysing some convergence aspects of its approximate solution, we will solve the simpler least-squares minimisation problem given by

$$\min_{s \in V_N} ||f - s||_{L_2(\Omega)}$$

for a given L_2-integrable target function f (though less stringent requirements can be imposed on this function as we will see in the next theorem) and as before

$$V_N = \text{span}\{\Phi(\cdot - \mathbf{x}_j) : 1 \leq j \leq N\}^4$$

where $\Phi(\cdot - \mathbf{x}_j)$ are the Wendland C^2 basis functions (1.5). Convergence estimates in the L_2-norm are given by the following theorem and corollary.

[4]This is obviously not the same "N" as the one used in Chapter 2, which represented the number of y-direction quadrature points for the numerical integration schemes.

Theorem 3.1.1 *Suppose Φ is positive definite with RKHS $H_\Phi \subseteq H^\tau(\mathbb{R}^d)$, $\tau > \frac{d}{2}$, $\Omega \subseteq \mathbb{R}^d$ bounded, satisfying a cone condition* [5] *and $f \in H_\Phi$. Then for sufficiently small fill distance $h = \sup_{x\in\Omega} \min_{1\leq j\leq N} ||x - x_j||_2$ we have*

$$||f - s||_{L_2(\Omega)} \leq Ch^\tau ||f||_{H_\Phi}$$

where $s \in V_N$ is the function that minimises $||f - s||_{L_2(\Omega)}$.

Corollary 3.1.2 *If Φ generates $H_\Phi = H^\tau(\mathbb{R}^d)$ and if Ω has a Lipschitz boundary then*

$$||f - s||_{L_2(\Omega)} \leq Ch^\tau ||f||_{H^\tau(\Omega)} \text{ for all } f \in H^\tau(\Omega).$$

As we have informally noted in the introductory chapter, the compactly supported Wendland basis functions do in fact "generate" the Sobolev space as its native space i.e. $H_\Phi = H^\tau(\Omega)$ with $\tau = \frac{d}{2} + k + \frac{1}{2}$. Hence the convergence estimate given above applies for our setting. In particular, given that we are working in a 2-dimensional domain (d=2) and using C^2 Wendland basis functions (k=1), we must have a theoretical convergence rate of order $O(h^{2.5})$ for the least-squares problem.

However, hoping to achieve such a convergence rate comes with a number of constraints. Firstly, we will have to work in the so-called non-stationary setting: we need to fix a basis function, namely the size R of its support, and let the fill distance tend to zero i.e. $h \to 0$. The reasons for doing so come from interpolation theory, which is relevant to the least-squares problem because the interpolation error provides an upper-bound for the latter. Indeed, when using basis functions that "generate" the Sobolev space H^τ we have the following bound in the L_2-norm:

$$||f - s_{f,X}||_{L_2(\Omega)} \leq C\Big(\frac{h}{R}\Big)^\tau ||f||_{H^\tau(\Omega)}$$

where f is the target function as before and $s_{f,X}$ its interpolant. This estimate suggests that if we choose to work in the non-stationary setting, we will have convergence as $h \to 0$. But doing so annihilates the advantages of using basis functions with compact support, namely the associated matrix loses its sparsity and thus we cannot take advantage of it when solving the system of equations. Furthermore, results from interpolation theory suggest that the behaviour of the condition number of the interpolation matrix is given by

$$\text{cond}(\mathbf{A}) = \Big(\frac{q}{R}\Big)^{-2\tau}$$

[5]The boundary must not have any cusp-like features.

where $q = \min_{j \neq k} ||x_j - x_k||$ is the separation distance. So it is likely that the matrix associated with the least-squares problem will behave in a similar way. Hence if we work in the non-stationary setting and thus keep R fixed, the matrix will have a rapidly increasing condition number as $h \to 0$.

If on the other hand we choose to scale the radius to be proportional to the fill distance i.e. $R = ch$ (this is known as the stationary setting) where c is some constant, then since $R = \tilde{c}q$ [5] where $\tilde{c}$ is some other constant, the condition number will remain fixed. But from the L_2 convergence estimate above, it is clear that convergence no longer occurs when working in the stationary setting. This is a well-known problem in scattered data approximation by RBFs and is solved by considering a multilevel algorithm which we shall briefly describe in Chapter 4.

Secondly, still in the context of interpolation theory, numerics show that it is necessary to choose a radius R such that a good portion of the underlying domain is covered in order to obtain accurate results. Otherwise, one would need substantially smaller h to compensate. This can somewhat be explained by the above inequality which suggests that if R were small, one would need to take significantly smaller h to get accurate results.

3.1.2 Numerical Aspects

We now wish to consider whether the theoretical convergence rate for the least-squares problem is observed in practice when solving the problem numerically. To do so we define

$$e_j = ||f - s_j||_{L_2(\Omega)}. \tag{3.1}$$

Then by Corollary 3.1.2 we have

$$e_j \leq C h_j^{\tau} ||f||_{H^{\tau}}.$$

Here $\{h_j\}_{j=1}^{\infty}$ is a discrete set of monotonically decreasing fill distance values and s_j the function in the finite dimensional subspace spanned by Wendland RBFs and corresponding to the particular fill distance h_j. In other words, by decreasing h_j we increase the dimension of the subspace. We now have

$$\frac{e_{j+1}}{e_j} = \left(\frac{h_{j+1}}{h_j}\right)^{\tau}.$$

[5] This is the case since h and q are clearly interlinked in this problem

Taking logs on both sides of the expression above we obtain the discrete convergence rate

$$\tau_j = \frac{\log(e_{j+1}/e_j)}{\log(h_{j+1}/h_j)}. \tag{3.2}$$

Expression (3.2) gives us a way of testing how efficient our numerical integration schemes are. If they were, what we hope would be achieved numerically is

$$\tau_j \longrightarrow 2.5 \text{ as } j \longrightarrow \infty.$$

Indeed if we did not have at least this sort of convergence rate, it could suggest that the evaluation of the integrals associated with this problem are not done accurately enough and the numerical integration schemes would have to be reviewed. From now on we will refer to the discrete fill distance values $\{h_j\}_{j=1}^{\infty}$ as simply "h".

When verifying this convergence numerically we obviously need to consider the discrete L_2-norm instead of its continuous counterpart, which we denote by l_2. Hence when calculating the least-squares error (3.1) on a computer, we will compute its l_2-norm instead given by

$$e_j = \left[\frac{1}{L}\sum_{k=1}^{L} |f(t_k) - s_j(t_k)|^2\right]^{\frac{1}{2}}$$

along a fine mesh consisting of L points inside the domain Ω. We will choose

$$f(x,y) = \frac{\cos(\pi x)\cos(\pi y)}{2\pi^2 + 1}$$

as our target function for the least-squares problem, which is clearly L_2-integrable. We also choose the set of centres $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subseteq \Omega$ to make up a finite uniform grid of N distinct points in $[-1,1]^2$ as was shown in Figure 2.2. We keep on using the C^2 Wendland basis functions and choose their support radii to all be equal to the fixed value R. This is then implemented using the scientific computing software Matlab and we use the software's matrix inversion operator $\backslash$ to solve the resulting matrix problem. Note that for large values of N the system of equations can be solved iteratively using the conjugate gradient method since the matrix is symmetric positive-definite.

Table 3.1 shows the l_2-error as well as the condition number of the associated matrix as we decrease the fill distance h (i.e. the non-stationary setting) when using a quadrature rule over the whole solution domain Ω to approximate the integrals. We use a tensor product of Gauss quadrature rules over the whole domain, for which we use 20 quadrature points in both the x- and y-directions, so a total of 400 quadrature points are used for the approximation of each integral of the matrix. The radii of the supports has been set to $R = 1.4$, which in some sense covers the whole domain, as required to obtain accurate results.

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	1.138067e-003		8.197211e+000
25	1/2	9.527643e-004	0.2564	3.335351e+003
49	1/3	1.048943e-004	5.4412	1.794808e+005
81	1/4	2.156888e-005	5.4977	2.965538e+006
121	1/5	8.247713e-006	4.3088	2.845482e+007
169	1/6	3.935580e-006	4.0587	3.191473e+008
225	1/7	3.646614e-006	0.4950	5.580711e+011
289	1/8	5.685706e-004	-37.8227	6.688194e+014

Table 3.1: Errors and condition numbers in the non-stationary setting with $R = 1.4$ for least-squares problem using quadrature on whole domain with 400 quadrature points.

Notice how the convergence is rather erratic at first but thereafter remains relatively stable as h is decreased. The convergence rate averages out to be surprisingly high for $h > \frac{1}{7}$, moreso than the theoretical convergence rate that we expected. This is probably due to the smoothness of the target function $f(x, y)$, which is a C^∞ function in both arguments.

As expected, the condition number increases rapidly as h decreases. The higher the condition number, the more the errors due to the quadrature rules will have an impact on the least-squares error. In fact, as shown in Table 3.1 for $h \leq \frac{1}{8}$, convergence ceases to occur because of this despite working in the non-stationary setting.

Table 3.2 shows the l_2-error and the condition number of the associated matrix as h is decreased for the same parameters as before, but now we use the lens quadrature rule (2.12) for the integrals of the associated matrix based on taking a Gauss tensor product in both the x- and y-directions, and the circle quadrature (2.18) for the integrals of the right-hand side vector based on applying Gauss quadrature for the radius r and a composite trapezium rule for the angles θ. Again, each integral is approximated using $20 * 20 = 400$ quadrature points.

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	2.134717e-003		8.497649e+000
25	1/2	2.794296e-002	-3.7103	1.446217e+003
49	1/3	3.421768e-002	-0.4997	3.017600e+004
81	1/4	1.065335e+000	-11.9517	1.604743e+006
121	1/5	1.930566e-002	17.9732	5.875189e+004
169	1/6	2.615597e-002	-1.6657	2.359948e+005
225	1/7	1.856221e-002	2.2244	2.363553e+006
289	1/8	4.063846e-002	-5.8696	3.922929e+006

Table 3.2: Errors and condition numbers in the non-stationary setting with $R = 1.4$ for least-squares problem using circle and lens quadratures with 400 quadrature points.

It appears that no convergence takes place when using these quadrature rules for this choice of R, despite the large number of quadrature points taken. The l_2-error fluctuates randomly and appears to stagnate at an error of order $O(10^{-2})$ without any improvements as h is decreased. This can be explained by our choice of a large radii for the supports of the RBFs: with such a choice, the vast majority of the entries of the associated matrix and right-hand side vector require approximating integrals with boundary supports. In Chapter 3, we have discussed how approximating these would create a significant source of error unless we chose a disproportionately large number of quadrature points, so such poor convergence results were not unexpected.

In Table 3.3 and 3.4 we consider exactly the same problem but using the smaller radius $R = 0.33$. Figure 3.1 shows the sparsity of the matrix for this particular value of R. Table 3.3 shows the l_2-errors and condition numbers for the least-squares problem when using the quadrature over the whole domain. Table 3.4 on the other hand shows the same but using the circle and lens quadrature rules instead. Both now use 10 quadrature points in both the x- and y-directions, making it a total of 100 quadrature points per integral approximation.

Notice how we now obtain more accurate results with the circle and lens quadratures than we did with the quadrature on the whole domain for this value of R, in complete contrast to what we observed before for the much larger radius. Indeed, a much larger proportion of the entries of the associated matrix are integrals with interior supports. In Chapter 3 we have seen how in this case, the circle and lens

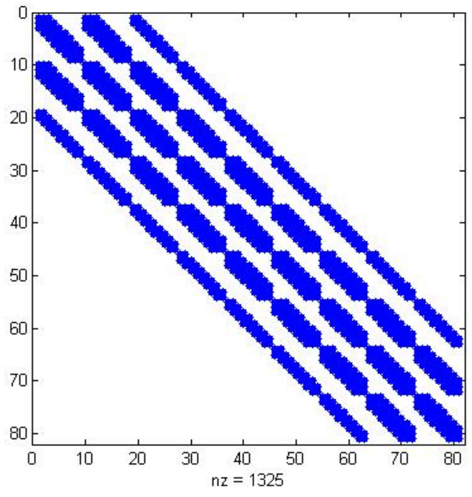


Figure 3.1: Sparsity of matrix associated with least-squares problem for $R = 0.33$ and $N = 81$ equally spaced nodes in $[-1, 1]^2$.

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	4.668942e-002		3.990771e+000
25	1/2	4.510292e-002	0.0499	1.662029e+001
49	1/3	4.154178e-002	0.2027	3.596113e+001
81	1/4	8.694786e+001	-26.5793	6.068565e+010
121	1/5	2.353071e-001	26.4951	1.793855e+018
169	1/6	2.039406e+000	-11.8445	1.391427e+018
225	1/7	3.044862e-001	12.3373	2.271383e+019

Table 3.3: Errors and condition numbers in the non-stationary setting with $R = 0.33$ for least-squares problem using quadrature on whole domain with 100 quadrature points.

quadratures generally work well and produce more accurate approximations than the quadrature on the whole domain.

Furthermore, using the circle/lens quadratures also appears to dramatically reduce the condition number. This would enable us to decrease h even further and not expect the former to interfere with convergence. That said, Table 3.4 shows a clear sign that the convergence rate will deteriorate for $h \leq \frac{1}{7}$ leading to non-convergence. This is due to the increasing number of integrals with boundary supports that need to be approximated, which create a significant source of error.

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	2.003157e-002		4.495205e+000
25	1/2	1.974932e-002	0.0205	4.496236e+000
49	1/3	1.253948e-002	1.1202	4.976892e+000
81	1/4	4.845443e-003	3.3050	8.724346e+000
121	1/5	1.479043e-003	5.3177	2.858306e+001
169	1/6	1.371853e-003	0.4125	2.461422e+002
225	1/7	3.260164e-003	-5.6153	3.977739e+003

Table 3.4: Errors and condition numbers in the non-stationary setting with $R = 0.33$ for least-squares problem using circle and lens quadratures with 100 quadrature points.

Table 3.5 and 3.6 show the results from exactly the same computations as those done in respectively Table 3.3 and 3.4, but the quadrature rules now use 20 quadrature points in both x- and y-directions i.e. 400 quadrature points to approximate each of the integrals.

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	2.011313e-002		3.474298e+000
25	1/2	1.981171e-002	0.0218	4.241641e+000
49	1/3	1.265591e-002	1.1052	4.260343e+000
81	1/4	4.991905e-003	3.2338	7.696114e+000
121	1/5	1.447959e-003	5.5467	2.218149e+001
169	1/6	6.027346e-004	4.8069	1.725686e+002
225	1/7	7.353432e-004	-1.2903	1.446568e+005

Table 3.5: Errors and condition numbers in the non-stationary setting with $R = 0.33$ for least-squares problem using quadrature on whole domain with 400 quadrature points.

Notice first of all how both schemes produce more accurate results than they did with lesser quadrature points, as expected. Table 3.5 shows how using more quadrature points dramatically reduces the condition number of the associated matrix, which again allows us to further decrease h and not expect any errors caused by a high condition number to interfere with convergence. Comparing Table 3.5 and 3.6, we observe that for $h > \frac{1}{6}$ the circle and lens quadratures produce more accurate results than the quadrature on the whole domain, but for $h < \frac{1}{6}$ the opposite occurs. Such results suggest that the interplay between the condition number, the proportion of integrals with interior/boundary supports in the matrix (dependent on R and h)

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	2.003158e-002		4.215894e+000
25	1/2	1.974932e-002	0.0205	4.216861e+000
49	1/3	1.253967e-002	1.1202	4.665419e+000
81	1/4	4.828023e-003	3.3179	8.064376e+000
121	1/5	1.225484e-003	6.1445	2.462371e+001
169	1/6	7.658254e-004	2.5784	1.164224e+002
225	1/7	1.437655e-003	-4.0856	8.088218e+004

Table 3.6: Errors and condition numbers in the non-stationary setting with $R = 0.33$ for least-squares problem using circle and lens quadratures with 400 quadrature points.

and the number of quadrature points chosen makes valid conclusions hard to infer. But generally speaking, approximations of the least-squares problem using the circle and lens quadrature would be increasingly more accurate than the approximation using the quadrature on the whole domain the more we decrease the support radii R.

Unfortunately there is a problem: as discussed previously, decreasing the size of the support radii yields lesser accuracy. This can be observed numerically by comparing Table 3.6 with Table 3.1. So by requiring the radii of the RBFs to be kept relatively large in order to achieve greater accuracy, it becomes evident that we will have to use the quadrature rule on the whole domain to attain this. However, the stationary setting multilevel algorithm may provide a framework in which the lens and circle quadratures can lead to accurate approximations. This will be discussed in Chapter 4.

3.2 Sobolev Problem

3.2.1 Theoretical Aspects

Having rigorously described the solution of the simpler least-squares problem, we are now ready to tackle the main issue of this dissertation: the solution of the Helmholtz equation which translates to a minimisation problem in the Sobolev norm as noted in Chapter 1. It was shown in [18] that if the radial basis functions $\Phi(\mathbf{x} - \mathbf{x}_i),\ i = 1, ..., N$ spanning the finite dimensional subspace V_N are positive definite with Fourier transforms decaying like $(1 + ||\cdot||_2)^{-2\beta}$ then we have the following.

Theorem 3.2.1 *Let the bounded domain Ω possess a C^1 boundary and let the solution u of (1.3) satisfy $u \in H^\tau(\Omega)$. As before, we let h denote the fill distance given by the expression $h = \sup_{x\in\Omega} \min_{1\leq j\leq N} ||x - x_j||_2$. If $\beta \geq \tau > \frac{d}{2} + 1$, the error between u and the discrete solution u_N of (1.6) can be bounded for sufficiently small h by*

$$||u - u_N||_{H^1(\Omega)} \leq Ch^{\tau-1}||u||_{H^k(\Omega)}.$$

The inequality in the above theorem reflects the theoretical order of convergence when using an approximation space spanned by RBFs, and is comparable to that achieved in classical finite element methods. It can be shown that the compactly supported Wendland basis functions in $\mathbb{R}^d$ (with smoothness $2k$) have Fourier transforms that decay like $(1 + ||\cdot||_2)^{-d-2k-1}$. Hence the above estimate suggests that functions which are C^{2k}, satisfying $k \geq \tau - \frac{d+1}{2}$ and strictly positive definite on $\mathbb{R}^d$ will have $O(h^{k+(d-1)/2})$ convergence order. In particular, the C^2 Wendland basis function (1.5) should yield $O(h^{1.5})$ convergence in $\mathbb{R}^2$. Note that

$$||u - u_N||_{L^2(\Omega)} \leq ||u - u_N||_{H^1(\Omega)}$$

so this theoretical convergence rate also applies for the least-squares error between the exact solution and its approximation, which we will be considering later on. Given this relationship between the least-squares norm and the Sobolev norm, it is clear that all the interpolation bounds we derived for the former will affect the latter. As before we will have to work in the non-stationary setting to hope for convergence, and keep the radii of the RBFs relatively large to get accurate results. It is also likely that the stiffness matrix will have a rapidly increasing condition number as $h \to 0$.

3.2.2 Numerical Aspects

Having described some of the more theoretical aspects associated with the solution of the Sobolev minimization problem, we will now consider an example for which a closed-form solution is known in order to find out whether the theoretical convergence applies in practice. Consider the following Helmholtz test problem which was considered in [10] and [18]:

$$-\nabla^2 u(x, y) + u(x, y) = \cos(\pi x)\cos(\pi y) \text{ in } \Omega, \tag{3.3}$$

$$\frac{\partial}{\partial \mathbf{n}} u(x, y) = 0 \text{ on } \partial\Omega \tag{3.4}$$

where as before $\mathbf{x} = (x, y)$, $\mathbf{n}$ denotes the outward unit normal vector and $\Omega = [-1, 1]^2$. Notice that $f(x, y) = \cos(\pi x)\cos(\pi y)$ is an L_2-integrable function as required. It can be shown that for the closed-form solution for this problem is given by

$$u(x, y) = \frac{\cos(\pi x)\cos(\pi y)}{2\pi^2 + 1}. \tag{3.5}$$

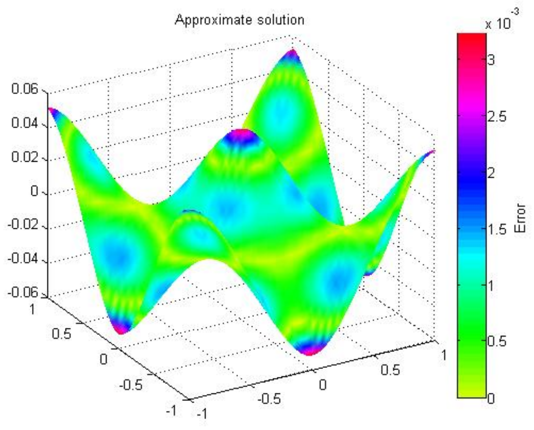
We can now solve the Sobolev minimization problem via the solution of the matrix system (1.9) arising from the Galerkin formulation of (3.3)—(3.4). We continue to use the C^2 Wendland functions (1.5) as our basis functions for the finite-dimensional subspace. In addition, we choose the support radii of the basis functions to all be equal to R. As before, we choose the set of centres $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subseteq \Omega$ to make up a finite uniform grid of N distinct points in $[-1, 1]^2$. Given the poor convergence results observed when using the lens and circle quadrature rules for big R in the simpler least-squares problem, the entries of the stiffness matrix $\mathbf{A}$ and load vector $\mathbf{f}$ associated with (3.3)—(3.4) will be approximated using the quadrature on the entire solution domain Ω. For this scheme, we use 20 quadrature points in both the x- and y-directions i.e. a total of 400 quadrature points per integral approximation. Table 3.7 shows the l_2-error between the exact solution (3.5) and the approximation, as well as the condition number of the stiffness matrix for decreasing fill distance h (as usual we are working in the non-stationary setting).

N	h	l_2-error	rate	cond($\mathbf{A}$)
9	1	1.108119e-003		8.159248e+000
25	1/2	9.565087e-004	0.2122	1.408073e+002
49	1/3	1.091387e-004	5.3536	3.302420e+003
81	1/4	7.780432e-005	1.1763	3.295054e+004
121	1/5	8.422028e-005	-0.3549	1.869342e+005
169	1/6	4.030209e-005	4.0423	8.511945e+005
225	1/7	2.403545e-005	3.3532	5.523046e+007
289	1/8	6.470658e-003	-41.9139	1.375148e+010

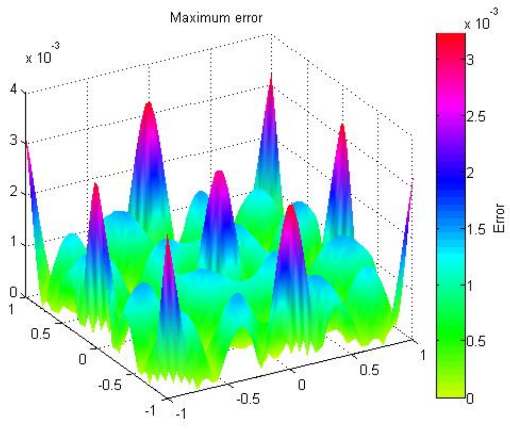
Table 3.7: Errors and condition numbers in the non-stationary setting with $R = 1.4$ for the Sobolev problem using quadrature on whole domain with 400 quadrature points.

As for the least-squares problem, convergence is very erratic. The condition number also increases rapidly as h is decreased. This causes significant errors that ultimately ceases convergence occurring, namely when $h < \frac{1}{7}$. In Figure 3.2(a) we show the plot of the approximate solution to the Galerkin problem of (3.3)—(3.4) for

$R = 1.4$ and $N = 25$. Figure 3.2(b) shows the graph of the maximum error between the exact solution (3.3) and the approximate solution along each point of a fine grid contained in $[-1, 1]^2$.



(a)



(b)

Figure 3.2: Approximate solution (left) and maximum error (right) for Galerkin solution of Helmholtz equation with C^2 Wendland functions using 25 equally spaced points in $[-1, 1]^2$.

Chapter 4

Multilevel Algorithms

As mentioned in Chapter 3 working in the stationary setting (scaling RBF supports proportional to the data density) keeps the matrix sparse and the condition number low, but this process does not always converge. In Reference [11], a multilevel algorithm was suggested as a way of resolving this issue in the context of interpolation by radial basis functions. In References [9] and [18], it was applied to the Galerkin approximation process.

We will now describe the Galerkin multilevel algorithm and how it works. Consider a sequence of finite dimensional subspaces $V_1, V_2, ... \subseteq H^1(\Omega)$ given by

$$V_m = \text{span}\{\Phi_{R_m}(\cdot - \mathbf{x}_i),\ i = 1, ..., N_m\}$$

where N_m is the number of nodes at level m of the algorithm and R_m the support radius of the RBFs at level m. As in Chapter 1, increasing the index m of V_m corresponds to increasing the dimension of the subspace. But additionally the RBFs that span each subspace V_m has their own fixed support radius R_m, which is proportional to the fill distance h_m (i.e. the stationary setting). Because of this, it is no longer the case that $V_1 \subseteq V_2 \subseteq ... \subseteq V_m$.

Furthermore, instead of solving the Galerkin problem (1.6) for increasingly larger subspace V_m, we solve the problem with different right-hand sides. At step m of the algorithm, we now seek $u_m \in V_m$ such that

$$a(u_m, v) = l(v) - a(u_{m-1}, v) \text{ for all } v \in V_m. \tag{4.1}$$

If we set

$$u_m = \sum_{j=1}^{N_m} c_j^m \Phi_{R_m}(\cdot - \mathbf{x}_j) \tag{4.2}$$

then substituting (4.2) into (4.1) yields

$$\sum_{j=1}^{N_m} c_j^m a(\Phi_{R_m}(\cdot - \mathbf{x}_j), \Phi_{R_m}(\cdot - \mathbf{x}_i))$$
$$= l(\Phi_{R_m}(\cdot - \mathbf{x}_i)) - \sum_{j=1}^{N_{m-1}} c_j^{m-1} a(\Phi_{R_{m-1}}(\cdot - \mathbf{x}_j), \Phi_{R_m}(\cdot - \mathbf{x}_i)). \quad (4.3)$$

Thus we obtain the matrix problem

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

where $\mathbf{c}^m = (c_1^m, ..., c_{N_m}^m)^T \in \mathbb{R}^{N_m}$, $\mathbf{f} = (f_1, ..., f_{N_m})^T \in \mathbb{R}^{N_m}$ with $f_i = l(\Phi_{R_m}(\cdot - \mathbf{x}_i))$. The matrix $\mathbf{A} \in \mathbb{R}^{N_m \times N_m}$ is given by

$$a_{ij} = a(\Phi_{R_m}(\cdot - \mathbf{x}_j), \Phi_{R_m}(\cdot - \mathbf{x}_i)),\ 1 \le i, j \le N_m$$

and $\mathbf{B} \in \mathbb{R}^{N_m \times N_{m-1}}$ by

$$b_{ij} = a(\Phi_{R_{m-1}}(\cdot - \mathbf{x}_j), \Phi_{R_m}(\cdot - \mathbf{x}_i)),\ 1 \le i \le N_m \text{ and } 1 \le j \le N_{m-1}.$$

Notice that the first and second arguments of the bilinear form for the matrix **B** have different support radii. An algorithm for this process is shown in Algorithm 4.0.1.

```
Algorithm 4.0.1: MULTILEVEL GALERKIN( )

 (1) Set $u_0 = 0$,
 (2) for $k = 1, 2, ..., m$
       (a) Find $u_k \in V_k$ such that $a(u_k, v) = l(v) - a(u_{k-1}, v)\ \forall v \in V_k$
       (b) Update $u_k \leftarrow u_{k-1} + u_k$
```

Intuitively, what Algorithm 4.0.1 does is update the approximation at each step by a fit to the most recent residual. The final solution u_N of the algorithm still satisfies

$$a(u_m, v) = l(v) \text{ for all } v \in V_m \quad (4.4)$$

but instead of being an element of V_m as it was the case in (1.6), the approximation is now an element of $V_1 + ... + V_m$. As far as I am aware no convergence results have been proved for this algorithm, but a useful stability result can be derived from (4.4):

$$||u - u_m||_{H^1(\Omega)} \le ||u - u_{m-1}||_{H^1(\Omega)} \le ... \le ||u||_{H^1(\Omega)}.$$

Given that the radius R_m will decrease at each step of the algorithm, the circle and lens quadrature rules required to evaluate the entries of the matrix $\mathbf{A}$ are likely to produce more accurate approximations than the quadrature rule of the whole domain as we increase the dimension of the subspace. Furthermore, $\mathbf{A}$ will become increasingly more sparse as R is decreased, leaving way for efficient iterative methods (such as Conjugate Gradient or GMRES) to solve the corresponding system of equations.

A major feature of this algorithm is the evaluation of the right-hand side term $l(v) - a(u_{k-1}, v)$. Indeed, the entries of the matrix $\mathbf{B}$ associated with $a(u_{k-1}, v)$ would need to be approximated using numerical integration schemes over the lens-shaped intersection of circular supports that may not necessarily have the same radius. Figure 4.1 shows an example of such an asymmetric lens.

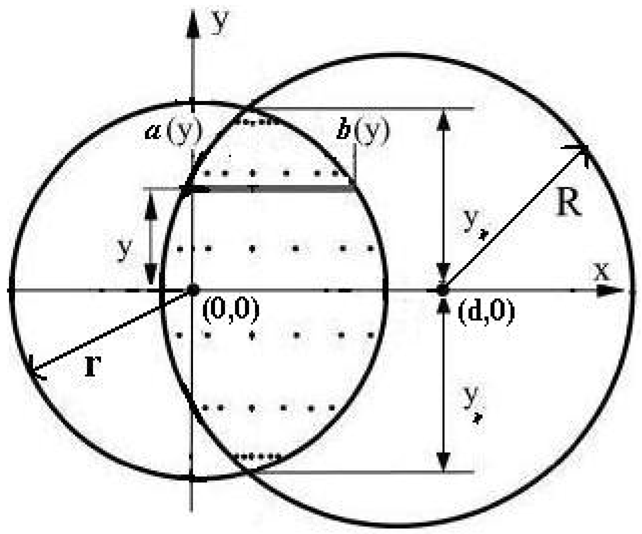


Figure 4.1: Asymmetric lens-shaped domain of integration.

To integrate over functions over this new domain, we need to find the intersection points between the circles of radius R and r. Without loss of generality we will again assume that one circle of radius R is centred at the origin $(0, 0)$ while the other circle of radius r is centred at $(d, 0)$, where $d > 0$. The equations of these circles are given by

$$x^2 + y^2 = R^2,$$

$$(x - d)^2 + y^2 = r^2.$$

The intersection points $(x_*, \pm y_*)$ exist if $d \leq r + R$. After some algebraic manipulation, the x-component of the intersection points is given by

$$x_* = \frac{d^2 - r^2 + R^2}{2d}$$

and the y-component by

$$y_* = \pm\frac{1}{2d}\sqrt{(-d + r - R)(-d - r + R)(-d + r + R)(d + r + R)}.$$

Using these bounds to integrate a function over the asymmetric lens is less obvious than before. Consider the following scenario for which we choose $R = 3$, $r = 2$ and $d = 1.5$ as shown in Figure 4.2. There are points in this lens-shaped domain that have

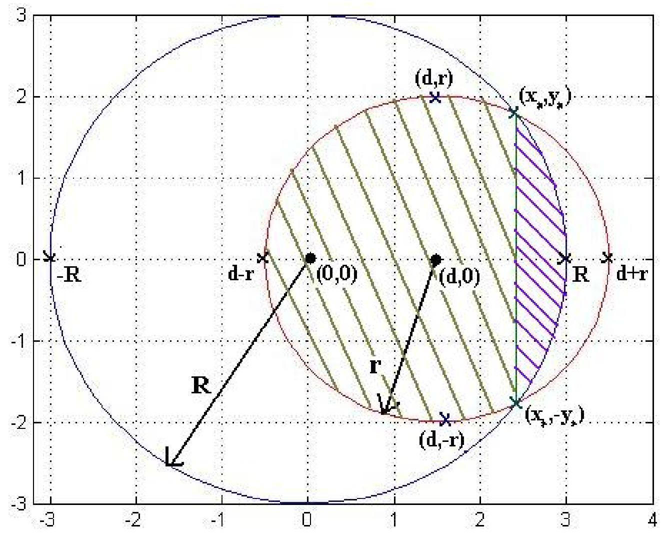


Figure 4.2: Asymmetric lens with $R = 3$, $r = 2$ and $d = 1.5$.

a larger y-component than that of the intersection point (x_*, y_*), so $[-y_*, y_*]$ can no longer be the "y" bounds for the double integral (2.5) in the case of the asymmetric lens. As shown in Figure 4.2 this problem is solved by integrating separately over the grey region which yields the bounds

$$\int_{d-r}^{x_*}\int_{-\sqrt{r^2-(x-d)^2}}^{\sqrt{r^2-(x-d)^2}} f(x,y)\ dydx,$$

and the pink region which yields the bounds

$$\int_{x_*}^{R}\int_{-\sqrt{R^2-x^2}}^{\sqrt{R^2-x^2}} f(x,y)\ dydx.$$

Adding these two expressions will give us the required double integral. However, specific numerical integration schemes can be applied on each of these double integrals separately for optimal approximations.

We will not pursue this analysis any further. Instead, for the sake of completeness, we briefly discuss an improvement of Algorithm 4.0.1. Numerical tests have shown that this algorithm does not always converge. A modified multilevel algorithm based on the interpretation of weak formulations as Hilbert space projection methods was proposed to solve this problem, details of which can be found in [18]. The new algorithm is given as follows:

```
Algorithm 4.0.2: MODIFIED MULTILEVEL GALERKIN( )

 (1) Fix $N$ and $M \in \mathbb{N}$ and set $v_0 = 0$
     for $k = 0, 1, ...$
         $(a)$ Set $u_0 = v_k$
         $(b)$ Apply Algorithm 4.1.1. Denote result by $u_N(v_k)$
         $(c)$ Set $v_{k+1} = u_N(v_k)$
```

Wendland proved that Algorithm 4.0.2 converges, with a convergence rate that was shown numerically to be at least linear based on the data in Table 2 of [18], whose values have been copied to Table 4.1 for convenience.

N	l_2-error
25	3.5258e-002
81	4.4971e-003
289	9.7278e-004
1089	5.3662e-004
4225	2.8713e-004
16641	1.9330e-004

Table 4.1: l_2-error of multilevel algorithm 4.0.2 taken from [18]

Chapter 5

Conclusions

We have studied several different numerical integration schemes for domains arising in the Galerkin approximation of the Helmholtz equation using compactly supported radial basis function known as Wendland functions. For the lens-shaped domain, applying Gauss quadrature rules in both the x- and y-directions appeared to give good results. Moving into polar coordinates was a better option for the circular domain, for which we used a composite trapezium rule in r and Gauss quadrature rule in θ. These two schemes were then compared with the conventional method of using a numerical integration scheme over the entire domain Ω. Problems arising when using them were discussed extensively and two particular scenarios stood out: the case when the two centres of the circular supports making up the lens are too close to one another, and the case when supports are not fully contained inside the domain Ω. In both cases the integrand or its derivatives had discontinuities in or close to the domain of integration, and was a cause for slower convergence. The main issue however was the non-optimality of quadrature points for integrals over boundary supports, which created the greatest source of error.

Having developed the numerical integration schemes, we then analysed how they (and their drawbacks) affect convergence for the simpler least-squares problem. Once this was understood, convergence tests were performed for the original Galerkin problem. Given the restrictions required to obtain theoretical convergence (namely the size of the support R) and the problems caused by the circle and lens quadratures for boundary supports, it became apparent that we had to use the quadrature on the whole domain to obtain accurate results. If one can improve the circle and lens quadrature rules so that they give rise to accurate approximation regardless of the choice of R, then this would lead to a highly efficient meshfree PDE solver. However, this would require dealing with approximating boundary integrals for arbitrary domains which is not an easy task.

Given more time, it would have been interesting to implement the multilevel algorithm described in Chapter 4 and study the influence of the numerical integration schemes on its convergence. Considering that the support radius R_m is decreased after each step of the algorithm, the lens and circle quadrature may well yield more accurate results when approximating the integrals associated with the corresponding matrices. This would involve analysing and comparing numerical integration schemes for asymmetric lenses. Another issue that hasn't been discussed in this dissertation is an adaptive Galerkin solver in which the nodes and support radii of the radial basis functions can be respectively moved and changed across the domain (in practice we could use such an adaptive solver for problems containing some sort of moving discontinuity, say the propagation of a fracture). In particular, such a solver can be defined so that all supports are fully contained within the domain, hence no longer requiring the approximation of integrals over boundary supports. To date, convergence estimates aren't known for such solvers but numerical results can be derived and analysed.

Bibliography

[1] Babuska, U., Banerjee, U., Osborn, J. (2002) Meshless and Generalised Finite Element Methods: a Survey of some Major Results. *Lecture Notes in Computational Science and Engineering* Springer.

[2] Babuska, U., Banerjee, U., Osborn, J. (2003) Survey of Meshless and Generalised Finite Element Methods: a Unified Approach. *Acta Numerica, pp. 1-125* Cambridge University Press, Cambridge, England.

[3] Bos, L., De Marchi, S. Sommariva, A., Vianello, M. (2010). Weakly Admissible Meshes and Discrete Extremal Sets. (submitted and available online at: http://www.math.unipd.it/ marcov/pdf/survey.pdf).

[4] Bos, L., Taylor, M.A., Wingate, B.A. (2000). Tensor Product Gauss-Lobatto Points are Fekete Points for the Cube. *Mathematics of Computation Vol. 70, Num. 236, pp. 1543-1547.*

[5] De, S., Bathe, K-J. (2001). Towards an Efficient Meshless Computational Technique: the Method of Finite Spheres. *Engineering Computations, Vol. 18, No. 1/2, pp. 170-192* MCB University Press, USA.

[6] De, S., Bathe, K-J. (2001). The Method of Finite Spheres with Improved Numerical Integration. *Computers and Structures 79 pp. 2183-2196* Pergamon, USA.

[7] Dolbow, J., Belytschko, T. (1999). Numerical Integration of the Galerkin Weak Form in Meshfree Methods. *Computational Mathematics* Springer.

[8] Elman, H., Silvester, D., Wathen, A. (2005). Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics. *Numerical Mathematics and Scientific Computation* Oxford University Press, Oxford, England.

[9] Fasshauer, G.E. (1999). Solving Differential Equations with Radial Basis Functions: Multilevel Methods and Smoothing. *Advances in Computational Mathematics Vol. 11 p 11* USA.

[10] Fasshauer, G.E. (2007). Meshfree Approximation Methods with Matlab. *Interdisciplinary Mathematical Sciences - Vol. 6* World Scientific Publishers, Singapore.

[11] Floater, M.S., Iske, A. (1996). Multistep Scattered Data Interpolation using Compactly Supported Radial Basis Functions. *Journal of Computational Science and Applied Mathematics 73 pp. 65-78* Elsevier Science Publishers.

[12] Liu, G.R, Liu, M.B. (2003). *Smoothed Particle Hydrodynamics: a Meshfree Particle Method.* World Scientific Publishers, Singapore.

[13] Santin, G., Sommariva, A., Vianello, M. (2010). An Algebraic Cubature Formula on Curvilinear Polygons. (submitted and available online at: http://www.math.unipd.it/ marcov/pdf/chebcub.pdf).

[14] Suli, E., Mayers, D. (2006). *An Introduction to Numerical Analysis.* Cambridge University Press, Cambridge, England.

[15] Weideman, J.A.C (2002). Numerical Integration of Periodic Functions: a Few Examples. *The American Mathematical Monthly, Vol. 109, No. 1, pp. 21-36* Mathematical Association of America, USA.

[16] Trefethen, L.N. (2008). Is Gauss Quadrature Better than Clenshaw-Curtis? *SIAM Review Vol. 50, No. 1, pp. 67-87* SIAM.

[17] Wendland, H. (1995). Piecewise Polynomial, Positive Definite and Compactly Supported Radial Functions of Minimal Degree. *Advances in Computational Mathematics 4, pp. 389-396* Springer.

[18] Wendland, H. (1998). Numerical Solution of Variational Problems by Radial Basis Functions. *Approximation Theory IX, Vol. 2: Computational Aspects* Vanderbilt University Press, USA.

[19] Wendland, H. (1999). Meshless Galerkin Methods using Radial Basis Functions. *Mathematics of Computation Vol. 68, Num. 228, pp. 1521-1531* American Mathematical Society, USA.

[20] Wendland, H. (2005). Scattered Data Approximation. *Cambridge Monographs on Applied and Computational Mathematics* Cambridge University Press, Cambridge, England.