AMIRKABIR UNIVERSITY OF TECHNOLOGY

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF APPLIED MATHEMATICS

P H D T H E S I S

to obtain the title of

PhD of Applied Mathematics

Defended by

Davoud Mirzaei

Title

Development of Moving Least Squares Based Meshless Methods

Thesis Advisor: Mehdi Dehghan

July 10, 2011

Tehran, Iran

Abstract

This thesis is devoted to the development of meshless methods based on the Moving Least Squares (MLS) approximation for solving partial differential and integral equations. Some basic concepts are discussed, firstly, and the stabilization effect of *shifted and scaled polynomials* as a basis function is investigated. Then, MLS is used for solving integral equations and the error analysis is given for Fredholm-type integral equations of the second kind. Applications are also performed for Volterra-type and integro-differential equations. For numerical solution of differential equations, some local PDE solvers based on MLS are reviewed. We mainly focus on the so called "Meshless Local Petrov-Galerkin (MLPG)" methods. The study then turns to some more important aspects by introducing a *generalized MLS (GMLS)* approximation in next chapters. The relation between this generalization and the so called *diffuse* or uncertain MLS derivatives is extracted, and using the concept of "generalized stable local polynomial reproduction" the error bound of diffuse derivatives is given. We conclude that there is nothing diffuse or uncertain about them and they are *direct* approximations of function derivatives with a rigorous mathematical background. This extension is also done on MLPG methods and new methods *Direct Meshless Local Petrov-Galerkin (DMLPG)* methods, are developed. DMLPG methods are absolutely superior to classical MLPG methods in terms of computational costs and accuracy.

Keywords: Moving least squares approximation; Generalized moving least squares approximation; Meshless methods; Standard derivatives; Diffuse derivatives; GMLS derivative approximation; Meshless Local Petrov-Galerkin (MLPG) methods; Direct Meshless Local Petrov-Galerkin (DMLPG) methods; Partial differential equations (PDEs); Integral equations; Convergence; Stability; Computational costs.

Acknowledgements

I would like to thank Mehdi Dehghan, my supervisor, for his suggestions, guidance and constant support during my PhD (and also Master) research. I am also thankful to Mostafa Shamsi, my advisor, for all his supports and friendly encouragements during my studying in Amirkabir University of Technology (AUT). Here, I would like to express my sincere gratitude to Robert Schaback (Georg-August-Universität Göttingen) for his hospitality, motivation and immense knowledge during my sabbatical leave in Göttingen that helped me in all the time of research and writing of this thesis. I also appreciate the time and details provided by the referees of this dissertation Esmaeil Babolian (Tarbiat Moallem University), Hojjatollah Adibi (AUT), Mahmoud Hadizadeh-Yazdi (K.N. Toosi University) and Shahriar Fariborz (AUT).

I should also mention that my graduate studies were supported in part by the Iran's National Elites Foundation.

Of course, I am so much grateful to my family for their patience and *love*. Without them this work would never have come into existence.

Tehran Davoud Mirzaei
July 10, 2011.

Contents

List of Figures

List of Tables

Preface

There are several approaches for numerical solution of partial differential equations (PDEs). Finite difference method (FDM), finite elements method (FEM), finite volumes method (FVM), boundary elements method (BEM) and spectral methods are some of such technologies. However, these and other mesh-based methods possess some shortcomings due to their reliance on a mesh or a predefined grided point set. Moreover, some of these methods have limitation for increasing the smoothness of the approximate solution. Attempts to overcome these difficulties have been made through the development of *meshless methods* which have attracted considerable interest over the past decades and now are in their own way to be another powerful approach.

The primary objective of meshless methods is to eliminate, or at least alleviate, the difficulty of meshing by writing the unknown solutions entirely in terms of *scattered data* points. This great benefit comes at a price of more complicated interpolation (approximation) functions and subsequently difficulties with implementing efficient integration schemes, among other problems.

The *moving least squares* (MLS) [18] is one of the scattered data approximation methods, that has been used successfully to approximate the trial space in many meshless methods such as the element-free Galerkin (EFG) method [10], the *hp*-clouds [13], the boundary nodes method (BNM) [31], the diffuse element method [33] and the finite points method (FPM) [35]. Many of these methods (unless FPM which is a collocation method) employ basis functions obtained by MLS to approximate the trial solution, and a background mesh to numerically evaluate integrals appearing in the global weak formulation of a problem. They are meshless only in approximation side but have to use background cells to integrate the weak form over the problem domain. The requirement of background cells for integration makes the methods not *truly* meshless. In contrast, the meshless local Petrov-Galerkin (MLPG) method [7] is based on local sub-domains, rather than a global problem domain and no background mesh is required to evaluate integrals appearing in the *local weak forms* of the problem. Since no globally defined integration structure is required, the MLPG method has been referred to as a truly meshless method and has been successfully applied for the solution of a wide range of problems in engineering and science. For more details see [5].

However, MLPG still suffers from the cost of numerical integration. Weak forms of MLPG contain the complicated and non-close form MLS shape functions and their derivatives. To get accurate results, numerical quadrature with many integration points is required and the MLS subroutines must be called very often, leading to high computational costs. In contrast to this, the stiffness matrix in FEM is constructed by integrating over polynomial basis functions which are much cheaper to evaluate. This thesis has a solution for this important problem.

In Chapter 1, the MLS approximation is reviewed and in Chapter 2 an application of MLS for solving integral equations is proposed. Chapter 3 reviews the local weak forms and variations of MLPG methods that will be required in the forthcoming chapters. The main goal of this thesis is to introduce a *generalized* MLS (GMLS) approximation which avoids the complications caused by MLS shape functions in numerical solution of PDEs. Chapter 4 is devoted to this purpose and we prove that this GMLS technique produces *diffuse derivatives* as introduced by Nyroles et. al. in 1992 [33] that turn out to be efficient direct estimates of the true derivatives, without anything "diffuse" about them because we prove optimal rates of convergence towards the true derivatives. Finally, in Chapter 5 an application of GMLS for solving PDEs is given which highly accelerates the original MLPG methods by avoiding integration over shape functions.

For numerical simulations, all routines were written using MATLAB© and run on a Pentium 4 PC with 2.50 GB of Memory and a twin–core 2.00 GHz CPU.

Chapter 1

Basic concepts and MLS approximation

In this chapter, first, we introduce some symbols and notations which will be used throughout the thesis. Then the *Moving Least Squares (MLS) Approximation* –the basic method in the whole parts of this study– will be briefly discussed. These discussions are mostly relied on the book of Wendland [43]. Finally, we establish the stabilization effect of *shifted and scaled polynomial basis functions* in the MLS approximation.

1.1 Notations

Throughout this text we will use the following notations. In whole of this study, $\mathbb{R}$ denotes the real numbers field, $\mathbb{N}_0$ the nonnegative integer numbers, d the dimension and $\mathbb{R}^d$ stands the d-dimensional space with real components. Symbol Ω is used to show a bounded domain in $\mathbb{R}^d$, with boundary $\partial\Omega$ or Γ. Matrices are shown by capital letters such as A, B, P, W, etc., while vectors are denoted by boldface letters such as $\boldsymbol{u}$, $\boldsymbol{b}$, $\boldsymbol{p}$, etc. We have an exception for elements of $\mathbb{R}^d$ which are denoted by letters x, y, z, etc. The component of a point $x \in \mathbb{R}^d$ will be denoted by $x = (\mathsf{x}_1, \dots, \mathsf{x}_d)$, whereas $x_1, \dots, x_N$ will denote N points in $\mathbb{R}^d$. Letters C and c with or without supper/subscripts are mostly used to denote the *constants*, and h to denote the spatial *mesh-size* or *fill distance*. We denote the space of d-variate polynomials of degree at most m by $\mathbb{P}_m^d$ and its dimension by $Q = (m+d)!/(m!d!)$. The function space

$C^k(\Omega)$ is the set of k times continuously differentiable functions on Ω, where we assume $\Omega \subset \mathbb{R}^d$ to be open if $k \geq 1$. The intersection of all these spaces is denoted by $C^\infty(\Omega)$. For a multi-index $\alpha \in \mathbb{N}_0^d$ we denote its components by $\alpha = (\alpha_1, \ldots, \alpha_d)$. The length of α is given by $|\alpha| = \alpha_1 + \cdots + \alpha_d$ and the factorial $\alpha!$ by $\alpha! = \alpha_1! \cdots \alpha_d!$. For two multi-indices α and β, the inequality $\alpha < \beta$ is meant component-wise and

$$\binom{\alpha}{\beta} = \frac{\alpha!}{\beta!(\alpha-\beta)!}.$$

If $|\alpha| \leq k$, $x \in \mathbb{R}^d$, and $f \in C^k(\Omega)$ are given, we denote the α-th derivative of f by

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial \mathsf{x}_1^{\alpha_1} \ldots \partial \mathsf{x}_d^{\alpha_d}}$$

and the α-th power of x by

$$x^\alpha = \mathsf{x}_1^{\alpha_1} \ldots \mathsf{x}_d^{\alpha_d}.$$

Other rare symbols and notations will be introduced through the text at the first place they are appeared.

1.2 Some useful definitions and theorems

According to Belytschko *et.al* [9], a *meshless method* constructs the solution of the problem entirely in terms of *scattered data points.* Let $\Omega \subset \mathbb{R}^d$ and $X = \{x_1, x_2, \ldots, x_N\}$ is a set of distinct scattered points in Ω. We will refer to X as *data sites* or *centers.* The meshless approximation is directly depends on position and quality of these points. To measure the quality of points we should define the quantities *fill distance* and *separation distance.*

Definition 1.1. For a set of points $X = \{x_1, \ldots, x_N\}$ in a bounded domain $\Omega \subset \mathbb{R}^d$ the fill distance is defined to be

$$h_{X,\Omega} = \sup_{x\in\Omega} \min_{1\leq j\leq N} \|x - x_j\|_2, \tag{1.1}$$

and the separation distance is defined by

$$q_X = \frac{1}{2} \min_{i \neq j} \|x_i - x_j\|_2. \tag{1.2}$$

A set X of data sites is said to be *quasi-uniform* with respect to a constant $c_{\mathrm{qu}} > 0$ if

$$q_X \leq h_{X,\Omega} \leq c_{\mathrm{qu}} q_X. \tag{1.3}$$

The fill distance $h_{X,\Omega}$ denotes the radius of the largest ball which is completely contained in Ω and which does not contain a data site. The separation distance gives the largest possible radius for two balls centered at different data sites to be essentially disjoint.

Let $V \in C(\Omega)$ be a N dimensional vector space, i.e. there are basis functions $u_1, u_2, \ldots, u_N$ where $V = \mathrm{span}\{u_1, u_2, \ldots, u_N\}$. As we know from the elementary numerical analysis, every interpolant $s \in V$ based on points $x_1, x_2, \ldots, x_N$ in one dimension $(d = 1)$ is uniquely determined if the points are all disjoint and $\{u_1, \ldots, u_n\}$ forms a *haar* or *Chebysheff* space. In this case the *Vandermonde* matrix $(u_j(x_i))_{i,j=1}^N$ is non-singular. However, this is not the case for $d \geq 2$. It is well-known after Mairhuber-Curtis Theorem that there is no finite dimensional Haar space for dimensions more than one. Hence, the distribution of points should be restricted to a situation that gives a unique interpolant.

Definition 1.2. The points $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ with $N \geq Q = \dim(V)$ are called V-unisolvent if the zero is the only element from V that vanishes on all X.

For instance, if $V = \mathbb{P}_m^d$ then $Q = \binom{m+d}{d}$ and the set $X = \{x_1, \ldots, x_N\}$, $N \geq Q$, is $\mathbb{P}_m^d$-unisolvent if the zero polynomial is the only polynomial from $\mathbb{P}_m^d$ which vanishes on X. As an example take the linear polynomials on $\mathbb{R}^2$. We know that $\dim(\mathbb{P}_1^2) = 3$. Since every bivariate linear polynomial describes a plane in three dimensional space this plane is uniquely determined by three points if and only if these three points are not collinear. Thus three points in $\mathbb{R}^2$ are $\mathbb{P}_1^2$-unisolvent if and only if they are not collinear. The following lemma gives a generalization for Q points in $\mathbb{R}^2$. For a proof and an extension to $\mathbb{R}^d$ see [43, pp 21].

Lemma 1.3. *Suppose that $\{L_0, \dots, L_m\}$ is a set of $m+1$ distinct lines in $\mathbb{R}^2$ and that $X = \{x_1, \dots, x_Q\}$ is a set of $Q = (m+1)(m+2)/2$ distinct points such that the first point lies on L_0, the next two points lie on L_1 but not on L_0, and so on, so that the last $m+1$ points lie on L_m but not on any of the previous lines $L_0, \dots, L_{m-1}$. Then X is $\mathbb{P}^2_m$-unisolvent.*

It is often difficult to theoretically treat the unisolvency for an arbitrary domain Ω. Therefore, we restrict ourselves to some special domains satisfying an interior cone condition.

Definition 1.4. A set $\Omega \subset \mathbb{R}^d$ is said to satisfy an *interior cone condition* if there exists an angle $\theta \in (0, \pi/2)$ and a radius $r > 0$ such that for every $x \in \Omega$ a unit vector $\xi(x)$ exists such that the cone

$$C(x, \xi, \theta, r) := \left\{x + ty : y \in \mathbb{R}^d, \|y\|_2 = 1, y^T\xi \geq \cos\theta, t \in [0, r]\right\}$$

is contained in Ω.

For local polynomial approximations, the cone condition allows to consider the unisolvency on local cones. This uses the fact that the line which connects the cone's center to any other points in cone is completely located inside the cone. Then the multivariate polynomials are restricted to the univariate ones on that line, and the *Markov's inequality* is applied to prove the unisolvency.

The following definition is also essential in local polynomial approximations. A generalization of this definition will be used in the sequel.

Definition 1.5. A process that defines for every set $X = \{x_1, \dots, x_N\} \subset \Omega$ a family of functions $s_j : \Omega \to \mathbb{R}, 1 \leq j \leq N$ provides a *stable local polynomial reproduction* of degree m on Ω if there exist constants $h_0, C_1, C_2 > 0$ such that

1. $\sum_{j=1}^N s_j(x)p(x_j) = p(x), \quad \forall p \in \mathbb{P}^d_m|\Omega, \forall x \in \Omega,$
2. $\sum_{j=1}^N |s_j(x)| \leq C_1, \quad \forall x \in \Omega,$
3. $s_j(x) = 0$ if $\|x - x_j\|_2 > C_2 h_{X,\Omega},$

is satisfied for all X with $h_{X,\Omega} \leq h_0$.

The first condition is a reproducing property that means polynomials of degree at most m are recovered by shape functions s_j, while the second one refers to the L_1 stability of the approximation, and the third one enforces a local support for shape functions. The crucial point in the definition is that the constants involved are independent of the data sites. The shape functions s_j will obviously depend on X. Sometimes we will also say $\{s_j\}$ forms a stable local polynomial reproduction.

The following theorem gives an error bound when a smooth function u is approximated via shape functions $\{s_j\}$ in a nodal form.

Theorem 1.6. *Let $\Omega \in \mathbb{R}^d$ be bounded and Ω^* denotes the closure of $\bigcup_{x\in\Omega} B(x, C_2h_0)$ where C_2 and h_0 are those in Definition 1.5. Assume*

$$\widehat{u}(x) := \sum_{j=1}^{N} s_j(x)u(x_j)$$

where $\{s_j\}$ forms a stable local polynomial reproduction of order m on Ω in sense of Definition 1.5. Then constant C exists such that for all $u \in C^{m+1}(\Omega^*)$ and all X with $h_{X,\Omega} \le h_0$ we have

$$|u(x) - \widehat{u}(x)| \le Ch_{X,\Omega}^{m+1}|u|_{C^{m+1}(\Omega^*)}. \tag{1.4}$$

The semi-norm on the right-hand side is defined as

$$|u|_{C^{m+1}(\Omega^*)} := \max_{|\alpha|=m+1} \|D^\alpha u\|_{L_\infty(\Omega^*)}.$$

The proof of this theorem, which is given in [43, chap. 3], uses the error of the standard multidimensional Taylor expansion of function u around x.

1.3 MLS approximation

The Moving Least Squares (MLS) approximation was introduced by Lancaster and Salkauskas [18] in 1981, inspired by the pioneer work of Shepard [41] in 1968 and McLain [21, 22] in 1974 and 1976. Since the MLS approximation is based on a cluster of scattered nodes instead of interpolation on elements, many MLS-based meshless methods for numerical solution of differential equations have been developed in recent years.

The MLS approximation solves for every point $x \in \Omega$ a locally weighted L_2 minimization problem. For a pure approximation, there is no need to set up and solve a large system, but rather than many small linear system should be solved. For applications to implicit operator equations, such as numerical solution of differential equations, MLS leads to a sparse system of equations allowing to adapt an iterative linear algebra solver.

Let, $\boldsymbol{u} = \{u(x_1), u(x_2), \ldots, u(x_N)\}$ is given at certain data site $X = \{x_1, x_2, \ldots, x_N\} \subseteq \Omega \subset \mathbb{R}^d$. Here, Ω is supposed to satisfy an interior cone condition with radius r and angle θ. For every point $x \in \Omega$ a subset of X in δ-neighborhood of x (i.e. $B(x,\delta) \cap X$) is used to approximate the exact value $u(x)$. We assume the influence of points on approximation of $u(x)$ decrease as their distances to x increase, and the points outside $B(x,\delta)$ have no influence on the approximation. Such behavior can be modeled by a weight function $w : \Omega \times \Omega \to \mathbb{R}$ which becomes smaller the further away its arguments are from each other and vanishes for arguments $x, y \in \Omega$ with $\|x - y\|_2 > \delta$. This means that w is of the form $w(x,y) = \Phi_\delta(x - y)$ where $\Phi_\delta = \Phi(\,\cdot\,/\delta)$ is the scaled version of a compactly supported function $\Phi : \mathbb{R}^d \to \mathbb{R}$.

Definition 1.7. For $x \in \Omega$, the value $\widehat{u}(x)$ of the MLS approximant of $u(x)$ is given by $\widehat{u}(x) = p^*(x)$ where p^* is the solution of

$$\min_{p \in \mathbb{P}_m^d} \left\{ \sum_{j \in J(x)} \big(u(x_j) - p(x_j)\big)^2 \Phi_\delta(x - x_j) \right\}. \tag{1.5}$$

where

$$J(x) = J(x,\delta,X) := \{j \in \{1,2,\ldots,N\} : \|x - x_j\|_2 \le \delta\}.$$

In the simplest case $m = 0$, the minimization problem has the explicit solution

$$\widehat{u}(x) = p^*(x) = \sum_{j \in J(x)} a_j(x) u(x_j)$$

where

$$a_j(x) = \frac{\Phi_\delta(x - x_j)}{\sum_{i \in J(x)} \Phi_\delta(x - x_i)}.$$

This is called the *Shepard approximation* [41], and obviously it reproduces the constants. The basis functions a_j have a support of radius δ where $\delta > C_2 h_{X,\Omega}$

for a suitable constant C_2. Finally, one can immediately see that

$$\sum_{j=1}^{N} |a_j(x)| = \sum_{j\in J(x)} \frac{\Phi_\delta(x-x_j)}{\sum_{i\in J(x)} \Phi_\delta(x-x_i)} = \frac{\sum_{j\in J(x)} \Phi_\delta(x-x_j)}{\sum_{i\in J(x)} \Phi_\delta(x-x_i)} = 1,$$

for all $x \in \Omega$. Hence we have indeed a local polynomial reproduction for polynomials of degree $m = 0$. Moreover, the smoothness of the approximant is ruled by the smoothness of the weight function, provided that the denominator is always nonzero. The latter is always true for a sufficiently large $\delta > 0$.

For $m > 0$ the explicit formula can not be derived simplicity. Indeed more ingredients need to prove the existence and uniqueness problems in general case $m > 0$. To this end we first need a result on quadratic optimization.

Lemma 1.8. *Let, $a \in \mathbb{R}$, $b \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ and $P \in \mathbb{R}^{n\times m}$ are all given. For $v \in \mathbb{R}^n$ define $M_v = \{x \in \mathbb{R}^n : P^T x = v\}$ and suppose $A = A^T$ is positive definite on M_0. If M_v is nonempty, then the quadratic form $f(x) := a + b^T x + x^T A x$ has a unique minimum on M_v.*

Proof. The reader is referred to Lemma 4.2 page 36 of [43]. □

What we need here, is an special case of the above Lemma. Indeed A is positive definite on all $\mathbb{R}^d$.

In 1998 Levin [19] obtained a equivalent constrained minimization problem to (1.5) where the optimization is defined on coefficient vector $\{a_j\}$ instead of on space of polynomials $\mathbb{P}_m^d$. The following theorem states such optimization problem.

Theorem 1.9. *Suppose that for every $x \in \Omega$ the set $\{x_j \in X : j \in J(x)\}$ is $\mathbb{P}_m^d$-unisolvent. Then* (1.5) *is uniquely solvable and solution $\widehat{u} = p^*$ can be written as*

$$\widehat{u}(x) = \sum_{j\in J(x)} a_j^*(x) u(x_j),$$

where the coefficients $a_j^*(x)$ are determined by minimizing the quadratic form

$$\frac{1}{2} \sum_{j\in J(x)} a_j(x)^2 \frac{1}{\Phi_\delta(x-x_j)} \tag{1.6}$$

subject to constraints

$$\sum_{j \in J(x)} a_j^*(x) p(x_j) = p(x), \quad p \in \mathbb{P}_m^d. \tag{1.7}$$

Proof. Although the proof of this Theorem is given in [19] and [43, Theorem 4.3] with slightly different ways, here we rewrite the proof of [43] due to some important details.

Denote a basis of $\mathbb{P}_m^d$ by $\{p_1, p_2, \ldots, p_Q\}$. Suppose our polynomial has the form $p = \sum_{j=1}^{Q} b_j p_j$. This reduces the minimization problem (1.5) to finding the optimal coefficient vector $\boldsymbol{b}^*$. We use the following notation:

$$\begin{aligned}
\boldsymbol{b} &= (b_1, \ldots, b_Q)^T \in \mathbb{R}^Q, \\
\boldsymbol{u} &= \big(u(x_j) \,:\, j \in J(x)\big)^T \in \mathbb{R}^{|J(x)|}, \\
P = P(x) &= \big(p_\ell(x_j)\big)_{j \in J(x),\, 1 \le \ell \le Q} \\
W = W(x) &= \operatorname{diag}\big(\Phi_\delta(x - x_j) \,:\, j \in J(x)\big) \in \mathbb{R}^{|J(x)| \times |J(x)|} \\
\boldsymbol{p} = \boldsymbol{p}(x) &= \big(p_1(x), \ldots, p_Q(x)\big)^T \in \mathbb{R}^Q.
\end{aligned}$$

Then we have to minimize the function

$$\begin{aligned}
C(\boldsymbol{b}) &= \sum_{j \in J(x)} \left[u(x_j) - \sum_{\ell=1}^{Q} b_\ell p_\ell(x_j) \right]^2 \Phi_\delta(x - x_j) \\
&= (\boldsymbol{u} - P\boldsymbol{b})^T W (\boldsymbol{u} - P\boldsymbol{b}) \\
&= \boldsymbol{u}^T W \boldsymbol{u} - 2\boldsymbol{u}^T W P \boldsymbol{b} + \boldsymbol{b}^T P^T W P \boldsymbol{b}
\end{aligned}$$

on $\mathbb{R}^Q$. Since $C(\boldsymbol{b})$ is a quadratic function in $\boldsymbol{b}$ we can apply Lemma 1.8. We get a unique solution if $P^T W P$ is positive definite. From

$$\boldsymbol{b}^T P^T W P \boldsymbol{b} = \boldsymbol{b}^T P^T W^{1/2} W^{1/2} P \boldsymbol{b} = \|W^{1/2} P \boldsymbol{b}\|_2 \ge 0,$$

it follows that $P^T W P$ is positive semi-definite. Moreover, $\boldsymbol{b}^T P^T W P \boldsymbol{b} = 0$ means that $P\boldsymbol{b} = 0$. Thus the polynomial $p = \sum b_\ell p_\ell$ vanishes on every $x_j, j \in J(x)$. Since this set is assumed to be $\mathbb{P}_m^d$-unisolvent, p and hence $\boldsymbol{b}$ must be zero. Now that we know the existence of a unique solution we can use the necessary condition $\nabla C(\boldsymbol{b}^*) = 0$ to compute it. We find that

$$0 = \nabla C(\boldsymbol{b}^*) = -2\boldsymbol{u}^T W P + 2(\boldsymbol{b}^*)^T (P^T W P),$$

which gives $(\boldsymbol{b}^*)^T = \boldsymbol{u}^T WP(P^T WP)^{-1}$, and we obtain the solution

$$p^*(x) = (\boldsymbol{b}^*)^T \boldsymbol{p}(x) = \boldsymbol{u}^T WP(P^T WP)^{-1}\boldsymbol{p}(x).$$

In the final step we treat the problem of minimizing (1.6) under the constraints (1.7). This means that we have to minimize the function

$$C(\boldsymbol{a}) = \frac{1}{2}\sum_{j\in J(x)} a_j^2 \frac{1}{\Phi_\delta(x-x_j)} = \frac{1}{2}\boldsymbol{a}^T W^{-1}\boldsymbol{a}$$

on the set

$$\begin{aligned} M := &\left\{\boldsymbol{a}\in\mathbb{R}^{|J(x)|} : \sum_{j\in J(x)} a_j p_\ell(x_j) = p_\ell(x),\, 1\le \ell \le Q\right\} \\ = &\left\{\boldsymbol{a}\in\mathbb{R}^{|J(x)|} : P^T\boldsymbol{a} = \boldsymbol{p}(x)\right\}. \end{aligned}$$

Since we have supposed $\{x_j : j\in J(x)\}$ to be $\mathbb{P}_m^d$-unisolvent we can always find Q points that allow unique polynomial interpolation. Hence M is not empty. Moreover, W^{-1} is obviously positive definite. Thus Lemma 1.8 gives us a unique solution to this problem. To compute this solution we can use Lagrange multipliers. If $\boldsymbol{a}^*\in M$ is a solution of the modified problem, there has to be $\boldsymbol{z}\in\mathbb{R}^Q$ such that

$$\nabla C(\boldsymbol{a}^*) = \boldsymbol{z}^T \frac{\partial}{\partial \boldsymbol{a}}[P^T\boldsymbol{a} - \boldsymbol{p}(x)]\big|_{\boldsymbol{a}=\boldsymbol{a}^*}.$$

This means that $(\boldsymbol{a}^*)^T W^{-1} = \boldsymbol{z}^T P^T$ or $\boldsymbol{a}^* = WP\boldsymbol{z}$, showing in particular that $\boldsymbol{a}^*$ is the unique solution of the modified problem, which makes it also the solution of the initial problem. From $\boldsymbol{a}^*\in M$ we can conclude that $\boldsymbol{p}(x) = P^T\boldsymbol{a}^* = P^T WP\boldsymbol{z}$, which gives the representation $\boldsymbol{z} = (P^T WP)^{-1}\boldsymbol{p}(x)$. Finally, we find

$$\sum_{j\in J(x)} a_j^*(x) u(x_j) = \boldsymbol{u}^T\boldsymbol{a}^* = \boldsymbol{u}^T WP(P^T WP)^{-1}\boldsymbol{p}(x) = p^*(x).$$

□

The basis functions a_j are called *shape function.* From the proof of the above theorem we can find some interesting properties of the shape functions a_j. The most important of these concerns the explicit form of a_j and the smoothness of the approximant.

Corollary 1.10. *The shape functions a_j are given by*

$$a_j^*(x) = \Phi_\delta(x - x_j) \sum_{k=1}^{Q} z_k p_k(x_j), \tag{1.8}$$

where $\boldsymbol{z}$ is the unique solution of

$$\sum_{k=1}^{Q} z_k \sum_{j \in J(x)} \Phi_\delta(x - x_j) p_k(x_j) p_\ell(x_j) = p_\ell(x), \quad 1 \le \ell \le Q, \tag{1.9}$$

or in abstract form

$$P(x)^T W(x) P(x) \boldsymbol{z} = \boldsymbol{p}(x).$$

Corollary 1.11. *If weight function Φ possesses k continuous derivatives then the approximant $\widehat{u}(x)$ is also in C^k.*

If we set $A(x) = P(x)^T W(x) P(x)$ and $B(x) = P(x)^T W(x)$, then we have $\boldsymbol{a}^*(x) = \boldsymbol{p}(x) A^{-1}(x) B(x)$. Since shape functions a_j are zero outside their "supports" we have

$$\widehat{u}(x) = \sum_{j=1}^{N} a_j(x) u(x_j). \tag{1.10}$$

Smoothness of shape functions allows to calculate the partial derivatives of $\widehat{\boldsymbol{u}}$. For instance, the first order derivatives are given by

$$D^{e_i}(a_j) = \sum_{k=1}^{Q} \big(D^{e_i}(p_k)[A^{-1}B]_{kj} + p_k[A^{-1}D^{e_i}(B) + D^{e_i}(A^{-1})B]_{kj} \big), \tag{1.11}$$

where e_i is the i-th standard vector in $\mathbb{R}^d$ and $D^{e_i}(A^{-1})$ represents the first order derivatives of matrix A and is given by

$$D^{e_i}(A^{-1}) = -A^{-1} D^{e_i}(A) A^{-1}.$$

Higher order derivatives are computationally more involved. This kind of derivatives are called *full derivatives* or *standard derivatives* of MLS shape functions. We will introduce a different type of approximate derivatives with simpler structures in Chapter 4.

Different types of weight functions can be used in MLS approximation. Compactly supported radial basis functions (CSRBF) [43, Chapter 9], for example

$$\Phi_\delta(x - x_j) = (1 - r/\delta)_+^4 (4r/\delta + 1), \quad r = \|x - x_j\|_2, \tag{1.12}$$

can be used for instance. This function has a C^2 smoothness for $d \leqslant 3$. Gaussian weight function

$$\Phi_\delta(x - x_j) = \begin{cases} \frac{\exp[-(r/c)^2] - \exp[-(\delta/c)^2]}{1 - \exp[-(\delta/c)^2]}, & 0 \le r \le \delta, \\ 0, & \text{otherwise} \end{cases}, \tag{1.13}$$

where c is a constant, is another candidate. This is a C^0 function. Spline functions can also be used for this scenario. They can be constructed for any arbitrary order of smoothness. A C^1 example is

$$\Phi_\delta(x - x_j) = \begin{cases} 1 - 6(r/\delta)^2 + 8(r/\delta)^3 - 3(r/\delta)^4, & 0 \le r \le \delta, \\ 0, & \text{otherwise} \end{cases}.$$

For numerical results presented in this dissertation, the Gaussian and CSRBF are used.

Error estimations for MLS approximation are given in various articles and books. Here we refer to some of them. In [19] Levin analyzed the MLS method for a particular weight function and obtained an error estimate in the uniform norm for the approximation of a regular function in multi dimensions. We also refer to [19] for an account of the background connection to Backus–Gilbert optimality and to [20] for the application to numerical integration. In [2], Armentano and Duràn proved error estimates in L^∞ for the function and its derivatives in one dimensional case. In [1], Armentano obtained the error estimates in L^∞ and L^2 norms in higher dimensions but with a restriction on convex domains Ω. In [46], Zuppa proved error estimates for the approximation of the function and the first and second order derivatives in L^∞ norm. Wendland in [42, 43] used the concept of local polynomial reproduction and obtained the error bound in infinity norm. We should also mention the work of Melenk [23] for analysis on derivatives. Here we give the Wendland's result.

Theorem 1.12. *Let Ω be a bounded domain that satisfies an interior cone condition. Consider the approximant* (1.10) *where a_j are MLS shape functions. There exist constant $C > 0$ such that for every $u \in C^{m+1}(\Omega^*)$ and every set of points $X \subset \Omega$ satisfying quasi-uniform condition with $h_{X,\Omega} \leqslant h_0$, we have*

$$\|u - \widehat{u}\|_{L_\infty(\Omega)} \leqslant C h_{X,\Omega}^{m+1} |u|_{C^{m+1}(\Omega^*)}. \tag{1.14}$$

The error bound (1.14) implies that for a sufficiently smooth function u on close domain Ω, the error of the MLS approximation uniformly behaves as $\mathcal{O}(h_{X,\Omega}^{m+1})$, where m is the degree of polynomial basis function.

1.4 Notes on numerical implementation

The method solves a weighted least–squares problem per each test point x. The QR factorization of $\sqrt{W}P$ will usually avoid the instability of the normal Vandermonde-type equations. If $\boldsymbol{a}^* = WP(P^TWP)^{-1}\boldsymbol{p}$ is requested, we decompose $\sqrt{W}P = QR$, where Q is unitary and R is upper triangular to get $P^TWP = R^TR$. In Matlab,

```
[Q R] = qr (sqrt(W)*P);
```

is used. By some simple calculations, $(WP)(P^TWP)^{-1}R^T = \sqrt{W}Q$. Using backward substitution, $(WP)(P^TWP)^{-1}$ is derived from this, and $\boldsymbol{a}^*$ can be calculated directly. For standard derivatives of MLS shape functions, some more but still straightforward calculations are needed. For instance, first derivatives of shape functions are

$$D^{e_i}\boldsymbol{a}^* = \left[\sqrt{W}Q(R^T)^{-1}\right] D^{e_i}(\boldsymbol{p}) + \left[\left(\hat{W}Q - \sqrt{W^{-1}}QQ^T\tilde{W}Q\right)(R^T)^{-1}\right]\boldsymbol{p},$$

where $\hat{W} = D^{e_i}(W)\sqrt{W^{-1}}$ and $\tilde{W} = \sqrt{W^{-1}}\hat{W}$. Both brackets are calculated using backward substitution without taking inverses. Higher order derivatives can be computed similarly, but in a more complicated way.

Sometimes, the normal system P^TWP is directly solved without QR decomposition, and the set

$$\mathcal{B} = \{x^\alpha\}_{0\le|\alpha|\le m} \tag{1.15}$$

is used as a basis for $\mathbb{P}_m^d$ in the MLS approximation. The choice of this basis is important and has a serious influence on conditioning of matrix $A = P^TWP$ and thus on matrix R of the QR factorization. As an example, consider the unit square $[0, 1]^2$ in $\mathbb{R}^2$ with regular node distribution of distance h and fix $m = 2$. Use the Wendland's function (1.12) as weight function. In Figure 1.1, the determinants and condition numbers of A are depicted in terms of h at a sample point $x = (\pi/4, \pi/4) \in [0, 1]^2$ on the left and right sides, respectively. As we see, the results get worse as h decreases. To overcome this drawback it

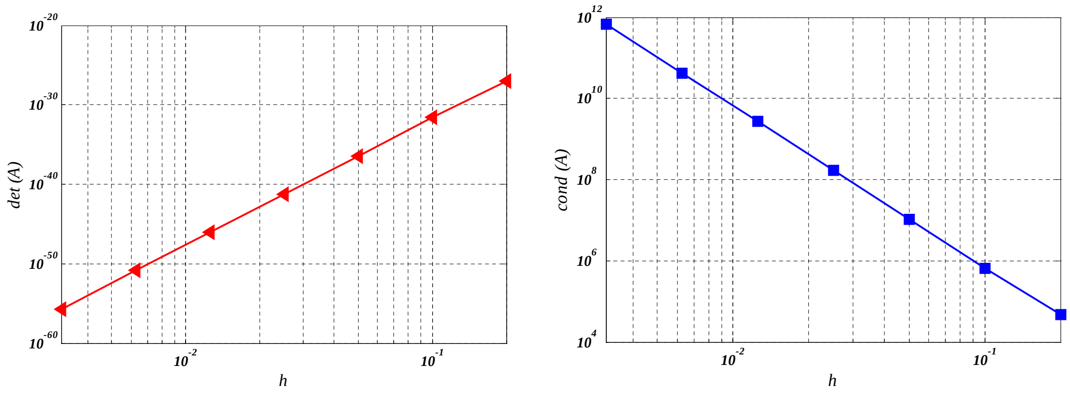


Figure 1.1: Determinants (left) and condition numbers (right) of A at sample point $(\pi/4, \pi/4)$ using basis (1.15)

is better to use the shifted and scaled basis polynomials. The shifted basis, which for example was used by [19] and [42], can be defined for a fixed $x \in \Omega$ by

$$\mathcal{B}^x = \{(\cdot - x)^\alpha\}_{0 \le |\alpha| \le m} \tag{1.16}$$

and the shifted and scaled basis by

$$\mathcal{B}_h^x = \left\{ \frac{(\cdot - x)^\alpha}{h^{|\alpha|}} \right\}_{0 \le |\alpha| \le m}, \tag{1.17}$$

where h can be q_X, $h_{X,\Omega}$ or an average of them for a quasi–uniform set X. In all cases, x is an evaluation point such as a test point or a Gaussian point for integration in weak-form techniques. Figure 1.2 shows the same results as before for the shifted and scaled basis functions. The effect of this variation is shown in Figure 1.3, where we have illustrated the maximum error of

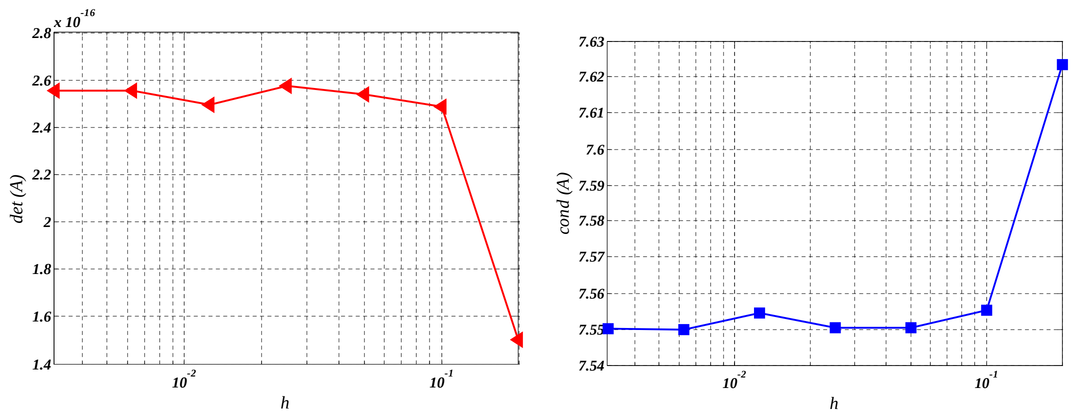


Figure 1.2: Determinants (left) and condition numbers (right) of A at sample point $(\pi/4, \pi/4)$ using basis (1.17)

reconstruction of *Franke's function* and its first and second derivatives with respect to x_1 on $[0,1]^2$ with and without shifted and scaled basis functions. The Franke's function function is defined as

$$\begin{aligned} u(\mathsf{x}_1,\mathsf{x}_2) =& \frac{3}{4}\exp\left(-1/4(9\mathsf{x}_1-2)^2+1/4(9\mathsf{x}_2-2)^2\right) \\ &+\frac{3}{4}\exp\left(-1/49(9\mathsf{x}_1+1)^2-1/10(9\mathsf{x}_2+1)^2\right) \\ &+\frac{1}{2}\exp\left(-1/4(9\mathsf{x}_1-7)^2+1/4(9\mathsf{x}_2-3)^2\right) \\ &-\frac{1}{5}\exp\left(-(9\mathsf{x}_1-4)^2-(9\mathsf{x}_2-7)^2\right) \end{aligned} \tag{1.18}$$

on $[0,1]^2$ which is a standard test function for 2D scattered data fitting since the seminal survey of [14]. Numerical instabilities are evident on the left side, where the basis (1.15) is applied. To analyze this phenomenon, we use the notations $A = P^TWP$, where the basis $\mathcal{B}$ is employed and $A^x = P(\cdot - x)^TWP(\cdot - x)$ and $A_h^x = P(\frac{\cdot - x}{h})^TWP(\frac{\cdot - x}{h})$ where $\mathcal{B}^x$ and $\mathcal{B}_h^x$ are used, respectively. By using

$$(y-x)^\alpha = \sum_{\beta\le\alpha} \underbrace{\binom{\alpha}{\beta}(-1)^{|\alpha-\beta|}x^{\alpha-\beta}}_{C_x(\alpha,\beta)} y^\beta = \sum_{\beta\le\alpha} C_x(\alpha,\beta)y^\beta$$

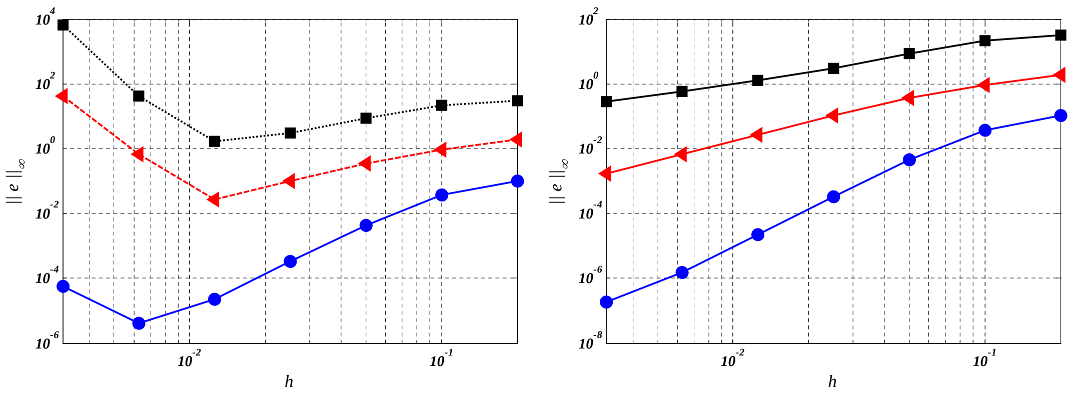


Figure 1.3: Approximation errors of Franke's function (solid lines and circles), its first derivative (dash lines and triangles) and its second derivative (dot lines and squares) using basis (1.15) (left) and basis (1.17) (right).

we have $P(\cdot - x) = PC_x$, where C_x is a Q by Q triangular matrix with diagonal elements 1. It is clear that $A^x = C_x^T A C_x$ and $\det(A) = \det(A^x)$. On the other side, we set

$$H_h = \operatorname{diag}\Big\{1, \underbrace{\frac{1}{h}, \cdots, \frac{1}{h}}_{\binom{d}{d-1} \text{ times}}, \underbrace{\frac{1}{h^2}, \cdots, \frac{1}{h^2}}_{\binom{d+1}{d-1} \text{ times}}, \cdots, \underbrace{\frac{1}{h^m}, \cdots, \frac{1}{h^m}}_{\binom{d-1+m}{d-1} \text{ times}}\Big\}_{Q\times Q}.$$

It is obvious that $P(\frac{\cdot - x}{h}) = PC_x H_h$ and $A_h^x = H_h A^x H_h$, hence $\det(A_h^x) = \det(A^x)\big[\det\big(H_h\big)\big]^2 = \det(A)\big[\det\big(H_h\big)\big]^2$. Using the combinatorial formula

$$\sum_{j=0}^{m} j C_{d-1}^{d-1+j} = d C_{d+1}^{m+d} =: \rho,$$

we have $\det\big(H_h\big) = h^{-\rho}$, therefore

$$\det(A_h^x) = h^{-2\rho} \det(A).$$

This is the reason why the determinant of A_h^x remains constant as h decreases. Consequently, we have

$$\det(R_h^x) = h^{-\rho} \det(R),$$

where R_h^x is upper triangular matrix obtained by QR decomposition of $\sqrt{W}P(\frac{\cdot - x}{h})$.

We can also estimate the condition numbers of both matrices A_h^x and R_h^x. Since $P(\frac{\cdot - x}{h}) = PC_xH_h$ and due to the uniqueness property of QR decomposition of full-rank matrices, we have

$$R_h^x = RC_xH_h.$$

Finally, $\mathrm{cond}(C_x) = 1$ and $\mathrm{cond}(H_h) = h^m$ yield

$$\mathrm{cond}(R) \approx \sqrt{\mathrm{cond}(A)}, \quad \mathrm{cond}(R_h^x) \approx \mathrm{cond}(R)h^m, \quad \mathrm{cond}(A_h^x) \approx \mathrm{cond}(A)h^{2m}. \tag{1.19}$$

Although the QR decomposition gives stable results in many cases, (1.19) implies that the shifted scaled basis is recommendable even when the QR factorization is applied. In all numerical results presented in this thesis, we will follow this strategy.

The quantity h can be replaced by a function which varies in accordance with the node density in Ω, see [15].

The results of this section on shifted and scaled polynomial basis functions are rewritten form [30].

Chapter 2

A MLS based method for solution of integral equations

Integral equations are encountered in various fields of science and engineering with numerous applications. We do not want to concern the applications, but we aim to present a numerical scheme based on MLS approximation for numerical solution of some types of integral equations. There exist several numerical methods for approximating the solution of Fredholm and Volterra integral equations in one, two and three dimensions. For example, see [3, 12] and the references therein. One may ask what the reason of developing such scheme is, while various numerical methods exist in the market? The main reason is the flexibility of MLS (and other meshless methods) with respect to the geometry and scattered point layouts. For solving a multi-variable integral equation on a non-rectangular region using collocation, Galerkin, or Nyström methods the domain must be segmented to small triangles and a discretization for both approximation and numerical integration over the segments is needed. For more details see [3, chap. 5]. Triangulations and mesh refinement are major difficulties in these methods. Replacing meshless methods should overcome these problems.

In this chapter, a *collocation method* with MLS approximation is developed for integral equations with smooth kernels. Error analysis is provided for Fredholm type, and applications are provided for Volterra and integro-differential equations.

2.1 Fredholm integral equations of the second kind

A Fredholm integral equation of the second kind can be written as

$$\lambda u(x) + \int_\Omega \kappa(x,s)u(s)ds = f(x), \quad x \in \Omega \subset \mathbb{R}^d, \tag{2.1}$$

where u is an unknown function, λ is a real parameter, Ω is a compact domain in $\mathbb{R}^d$, f is a given continuous right-hand side function, and κ is a given continuous kernel in $\Omega \times \Omega$. The above integral equation can be written in the abstract form

$$(\lambda - \mathcal{F})u = f \tag{2.2}$$

where

$$\mathcal{F}u = \int_\Omega \kappa(x,s)u(s)ds.$$

The uniform norm of integral operator $\mathcal{F}$ is defined as

$$\|\mathcal{F}\| := \max_{x\in\Omega} \int_\Omega |\kappa(x,s)|ds.$$

If $\mathcal{F}$ is a compact operator and $\|\mathcal{F}\| < \lambda$ then (2.2) has a unique solution for all continuous functions f and $\lambda \neq 0$.

For numerical solution we consider a set of *trial points*

$$X = \{x_1, x_2, \dots, x_N\} \subset \Omega$$

with fill distance $h_{X,\Omega}$. We assume that X is quasi-uniform and admits a well-defined MLS approximation. Suppose that $a_1, \dots, a_N$ are the MLS shape functions constructed by polynomial space $\mathbb{P}_m^d$ and weight function $\Phi \in C^k(\mathbb{R}^d)$, $k \in \mathbb{N}_0$. Define

$$V_N := \text{span}\{a_1, \dots, a_N\},$$

as a finite dimensional subspace of $C(\Omega)$. Recall the MLS approximation $\widehat{u}$

of u:

$$u \approx \widehat{u} = \sum_{j=1}^{N} \phi_j u(x_j) \in V_N.$$

We define a projection operator $\mathcal{P}_N : C(\Omega) \mapsto V_N$ which interpolates any continuous function into V_N on *test points*

$$Y = \{y_1, \dots, y_M\} \subset \Omega.$$

More precisely, for all $u \in C(\Omega)$ we define

$$\mathcal{P}_N u := \sum_{j=1}^{N} a_j c_j, \text{ with } \mathcal{P}_N u(y_k) = u(y_k),\ 1 \leq k \leq M.$$

In what follows, we let $M = N$ and we assume that, the distribution of both sets of test and trial points X and Y are well enough to ensure the non-singularity of $A_N = \big(a_j(y_k)\big)_{k,j=1}^{N}$. If it happens then $\mathcal{P}_N$ is well-defined. Since $\widehat{u} \in V_N$, we simply have $\mathcal{P}_N \widehat{u} = \widehat{u}$. Replacing u by $\widehat{u}$ in (2.1) we get

$$\sum_{j=1}^{N} \left[\lambda a_j(x) + \int_{\Omega} \kappa(x,s) a_j(s) ds \right] u(x_j) = f(x) + r(x),$$

where $r(x)$ is the reminder. In collocation method we assume that the reminder is vanished at test points Y, i.e.

$$\mathcal{P}_N r = 0,$$

which leads to

$$\sum_{j=1}^{N} \left[\lambda a_j(y_k) + \int_{\Omega} \kappa(y_k,s) a_j(s) ds \right] u(x_j) = f(y_k), \quad 1 \leq k \leq N,$$

or in an abstract form

$$\mathcal{P}_N(\lambda - \mathcal{F})\widehat{u} = \mathcal{P}_N f.$$

According to the property $\mathcal{P}_N \widehat{u} = \widehat{u}$, we have

$$(\lambda - \mathcal{P}_N \mathcal{F})\widehat{u} = \mathcal{P}_N f.$$

The involved integral can be treated by a numerical quadrature of the form

$$\int_{\Omega} g(s)\,ds \approx \sum_{\ell=1}^{Q_N} g(\theta_\ell)\omega_\ell, \quad g \in C(\Omega), \tag{2.3}$$

where $\{\theta_\ell\}$ and $\{\omega_\ell\}$, for $1 \le \ell \le Q_N$ are integration points and weights, respectively. We assume that for all $g \in C(\Omega)$ the quadrature converges to the exact value of integral as Q_N increases. Now we define

$$\mathcal{F}_N u(x) := \sum_{\ell=1}^{Q_N} \kappa(x,\theta_\ell)u(\theta_\ell)\omega_\ell, \quad x \in \Omega,\ u \in C(\Omega). \tag{2.4}$$

If we replace $\mathcal{F}\widehat{u}$ by $\mathcal{F}_N\widehat{u}$, we will get

$$\sum_{j=1}^{N}\left[\lambda a_j(y_k) + \sum_{\ell=1}^{Q_N}\kappa(y_k,\theta_\ell)a_j(\theta_\ell)\omega_\ell\right]\widetilde{u}_j = f(y_k), \quad 1 \le k \le N, \tag{2.5}$$

where $\widetilde{u}_j$ are the approximation values of $u(x_j)$. Solving the linear system of equations (2.5) gives the values $\widetilde{u}_j$, $j = 1,\dots,N$, and finally one can approximate

$$u(x) \approx u_N(x) = \sum_{j=1}^{N} a_j(x)\widetilde{u}_j,$$

for any $x \in \Omega$. The abstract form of equation (2.5) is

$$\mathcal{P}_N(\lambda - \mathcal{F}_N)u_N = \mathcal{P}_N f.$$

Since $u_N \in V_N$, we have $\mathcal{P}_N u_N = u_N$ and the above equation can be rewritten as

$$(\lambda - \mathcal{P}_N\mathcal{F}_N)u_N = \mathcal{P}_N f \tag{2.6}$$

which shows that the scheme is a *discrete collocation method* [3]. Consequently an *iterated discrete collocation* solution can be obtained. For this purpose we set

$$v_N(x) = \frac{1}{\lambda}[f(x) + \mathcal{F}_N u_N(x)], \quad \forall x \in \Omega, \tag{2.7}$$

and by applying the operator $\mathcal{P}_N$ on both sides of (2.7), and using the relation

(2.6) we simply have

$$\mathcal{P}_N v_N = u_N.$$

Thus we conclude

$$(\lambda - \mathcal{F}_N \mathcal{P}_N) v_N = f. \tag{2.8}$$

Equations (2.6) and (2.8) will be referred in the next section when we will try to give the error bounds for $u - u_N$ and $u - v_N$.

Usually and in this study the case $M = N$ is assumed which leads to a square final linear system. In addition we may assume $X = Y$. The case $M > N$ is called *oversampling* which may help if there is a problem with solvability.

2.2 Error analysis

As we discussed in the previous section, the method is a discrete collocation, and the solvability of the integral equation (2.1) and some insights on integration operators $\mathcal{F}_N$ and projections $\mathcal{P}_N$ are required to obtain the final error bound. Moreover, an error bound for the MLS approximation should be invoked.

According to (2.4) we define

$$\|\mathcal{F}_N\| := \max_{x \in \Omega} \sum_{\ell=1}^{Q_N} |\omega_\ell \kappa(x, \theta_\ell)|.$$

A direction which makes the analysis possible is to seek for characteristic properties of operators $\mathcal{F}_N$ which imply

$$\|(\mathcal{F} - \mathcal{F}_N)\mathcal{F}\| \to 0, \quad \|(\mathcal{F} - \mathcal{F}_N)\mathcal{F}_N\| \to 0, \ \text{ as } \ N \to \infty. \tag{2.9}$$

For this and following [3] we assume that $\{\mathcal{F}_N,\ N \geq 1\}$ possesses the following properties:

1. $\mathcal{F}$ and $\mathcal{F}_N$, for $N \geq 1$, are linear operators on $\mathcal{X}$ into $\mathcal{X}$ for Banach space $\mathcal{X}$.

2. $\mathcal{F}_N u \to \mathcal{F} u$ as $N \to \infty$, for all $u \in \mathcal{X}$.

3. The set $\{\mathcal{F}_N,\ N \geq 1\}$ is collectively compact which means that $\{\mathcal{F}_N u,\ N \geq 1,\ \|u\| \leq 1\}$ has a compact closure in $\mathcal{X}$.

Then $\{\mathcal{F}_N\}$ is said to be a *collectively compact family of pointwise convergent operators.* According to [3, Lemma 4.1.2], if $\{\mathcal{F}_N,\ N \geq 1\}$ is a collectively compact family of pointwise convergent operators, then (2.9) is satisfied. Finally, [3, Theorem 4.1.1] paves the way for finding the final error bound.

Theorem 2.1. *Let $\mathcal{X}$ be a Banach space, let $\mathcal{S}$ and $\mathcal{T}$ be bounded operators on $\mathcal{X}$ to $\mathcal{X}$ and let $\mathcal{S}$ be compact. For given $\lambda \neq 0$, assume $\lambda - \mathcal{T} :\ \mathcal{X} \xrightarrow[onto]{1-1} \mathcal{X}$, which implies $(\lambda - \mathcal{T})^{-1}$ exists as a bounded operator on $\mathcal{X}$ to $\mathcal{X}$. Finally assume*

$$\|(\mathcal{T} - \mathcal{S})\mathcal{S}\| < \frac{|\lambda|}{\|(\lambda - \mathcal{T})^{-1}\|}, \tag{2.10}$$

then $(\lambda - \mathcal{S})^{-1}$ exists and it is a bounded operator from $\mathcal{X}$ to $\mathcal{X}$. In fact, we have

$$\|(\lambda - \mathcal{S})^{-1}\| \leq \frac{1 + \|(\lambda - \mathcal{T})^{-1}\|\|\mathcal{S}\|}{|\lambda| - \|(\lambda - \mathcal{T})^{-1}\|\|(\mathcal{T} - \mathcal{S})\mathcal{S}\|}. \tag{2.11}$$

If $(\lambda - \mathcal{T})w = f$ and $(\lambda - \mathcal{S})z = f$, then

$$\|w - z\| \leq \|(\lambda - \mathcal{S})^{-1}\|\|\mathcal{T}w - \mathcal{S}w\|. \tag{2.12}$$

Now we go back to equations (2.6) and (2.8). In section 3.4 of [3] it is proved that the existence of the inverse operators $(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}$ and $(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}$ are related to each other. If $(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}$ exists, then so does $(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}$ and

$$(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1} = \frac{1}{\lambda}[I + \mathcal{F}_N(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}\mathcal{P}_N].$$

Conversely, if $(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}$ exists, then so does $(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}$ and

$$(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1} = \frac{1}{\lambda}[I + \mathcal{P}_N(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}\mathcal{F}_N].$$

By combining these, we also have

$$(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}\mathcal{P}_N = \mathcal{P}_N(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}.$$

We can choose to show the existence of either $(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}$ or $(\lambda - \mathcal{P}_N\mathcal{F}_N)^{-1}$ whichever is the more convenient, and the existence of the other inverse will follow immediately.

To use the results of Theorem 2.1 for schemes (2.6) and (2.8), we should first prove that $\{\mathcal{F}_N\mathcal{P}_N,\ N \geq 1\}$ is a "collectively compact family of pointwise convergent operators". To this aim, we need a uniform bound for $\|\mathcal{P}_N\|$. Since $\mathcal{P}_N$ is the interpolation operator to the MLS space, we first prove that the MLS approximation converges uniformly for continuous functions on compact domain Ω.

Theorem 2.2. *Suppose that $\Omega \subset \mathbb{R}^d$ is compact and satisfies an interior cone condition. The MLS approximation $s_{u,X}$ converges uniformly for all continuous function u, as $h_{X,\Omega}$ goes to zero for quasi-uniform sets X.*

Proof. For a fixed $x \in \Omega$, suppose that p_0 is the constant polynomial with $p_0(x) = u(x)$. The conditions of Theorem ensure that the MLS shape functions provide a stable local polynomial reproduction. Thus we can write

$$\begin{aligned}
|u(x) - \widehat{u}(x)| &= \Big|p_0(x) - \sum_{j=1}^{N} a_j(x)u(x_j)\Big| \\
&= \Big|\sum_{j=1}^{N} a_j(x)\big(p_0(x_j) - u(x_j)\big)\Big| \\
&\leq \sum_{j=1}^{N} |a_j(x)|\big|p_0(x_j) - u(x_j)\big| \\
&\leq C_1\|u - p_0\|_{\infty,B(x,\delta)\cap\Omega} \\
&= C_1 \max_{y\in B(x,\delta)\cap\Omega} |u(y) - u(x)| \\
&\leq C_1\omega(u,\delta),
\end{aligned}$$

where $\omega(u,\delta)$ is the modulus of continuity of u. The compactness of Ω and $\delta = ch_{X,\Omega}$ give the uniform convergence. $\square$

Now we can prove the following lemma.

Lemma 2.3. *Assume that $a_1, \ldots, a_N$ are the MLS shape functions on the quasi uniform set $X = \{x_1, \ldots, x_N\}$ with fill distance $h_{X,\Omega}$ on a compact domain Ω which satisfies an interior cone condition. If $\|A_N^{-1}\|_\infty = \mathcal{O}(1)$ independent of N (or $h_{X,\Omega}$), then there exists a constant c_P independent of N such that $\|\mathcal{P}_N\| \leq c_P$, and $\mathcal{P}_N u \to u$ uniformly for all $u \in C(\Omega)$.*

Proof. First

$$\mathcal{P}_N u(x) = \sum_{j=1}^{N} a_j(x) c_j, \quad \text{and} \quad \mathcal{P}_N u(y_k) = u(y_k),\ 1 \le k \le N,$$

give $\boldsymbol{c} = A_N^{-1}\boldsymbol{u}$. On the other hand we have

$$\|\mathcal{P}_N u\|_\infty \le \|\boldsymbol{c}\|_\infty \max_{x\in\Omega} \sum_{j=1}^{N} |a_j(x)| \le C_1 \|A_N^{-1}\|_\infty \|u\|_\infty.$$

The last inequality is satisfied because of the L_1 stability of the MLS shape functions (the second property of a stable local polynomial reproduction). Thus we can write

$$\|\mathcal{P}_N\| = \sup_{u\in C(\Omega)} \frac{\|\mathcal{P}_N u\|_\infty}{\|u\|_\infty} \le C_1 \|A_N^{-1}\|_\infty,$$

which leads to

$$c_P := \sup_N \|\mathcal{P}_N\| < \infty. \tag{2.13}$$

Finally, if $\widehat{u}$ is the MLS approximation of u on X then

$$\begin{aligned}\|\mathcal{P}_N u - u\|_\infty &\le \|\mathcal{P}_N u - \mathcal{P}_N \widehat{u}\|_\infty + \|u - \widehat{u}\|_\infty \\ &\le (1 + c_P)\|u - \widehat{u}\|_\infty \\ &\le C(1 + c_P)\omega(u, h_{X,\Omega})\end{aligned}$$

In the first inequality we have used $\mathcal{P}_N \widehat{u} = \widehat{u}$, and in the last one we have applied Theorem 2.2. Since the points are quasi uniform, $N \to \infty$ implies $h_{X,\Omega} \to 0$ and $\mathcal{P}_N u \to u$, uniformly. □

Remark 2.4. Experiments show that $\|A_N^\dagger\|_\infty$ is of order 1 independent of the fill distance $h_{X,\Omega}$ even if $M = N$. But it remains to prove this assertion, theoretically.

In the following lemma we prove that under some conditions $\{\mathcal{F}_N \mathcal{P}_N,\ N \ge 1\}$ is a collectively compact family of pointwise convergent operators.

Lemma 2.5. *Assume that $\{\mathcal{F}_N,\ N \ge 1\}$ is a collectively compact family of pointwise convergent operators on $\mathcal{X} = C(\Omega)$. Then $\{\mathcal{F}_N \mathcal{P}_N,\ N \ge 1\}$ is a collectively compact family of pointwise convergent operators on $\mathcal{X}$.*

Proof. From (2.13) we have $c_P \equiv \sup \|\mathcal{P}_N\| < \infty$. The pointwise convergence of $\{\mathcal{F}_N\}$ implies that $c_F \equiv \sup \|\mathcal{F}_N\| < \infty$. Together, these imply the uniform boundedness of $\{\mathcal{F}_N\mathcal{P}_N\}$ with a bound of c_Pc_F. For the pointwise convergence on $C(\Omega)$ we have

$$\begin{aligned}\|\mathcal{F}u-\mathcal{F}_N\mathcal{P}_Nu\|_\infty &\le \|\mathcal{F}u-\mathcal{F}_Nu\|_\infty+\|\mathcal{F}_N(u-\mathcal{P}_Nu)\|_\infty\\ &\le \|\mathcal{F}u-\mathcal{F}_Nu\|_\infty+c_F\|u-\mathcal{P}_Nu\|_\infty,\end{aligned}$$

and the convergence now follows from that of $\{\mathcal{F}_Nu\}$ and $\{\mathcal{P}_Nu\}$. To prove the collective compactness of $\{\mathcal{F}_N\mathcal{P}_N\}$ we must show that

$$\mathcal{K}=\{\mathcal{F}_N\mathcal{P}_Nu\,:\,N\ge 1,\ \|u\|_\infty\le 1\}$$

has a compact closure in $C(\Omega)$. From (2.13) we have

$$\mathcal{K}\subset\{\mathcal{F}_Nu\,:\,N\ge 1,\ \|u\|_\infty\le c_P\}$$

which proves the assertion because $\{\mathcal{F}_N\}$ is collectively compact. □

Now (2.9) is satisfied by replacing $\mathcal{F}_N$ by $\mathcal{F}_N\mathcal{P}_N$, and we can apply Theorem 2.1.

Theorem 2.6. *Let $\Omega\subset\mathbb{R}^d$ be a compact set and satisfy an interior cone condition, and the quasi uniform set $X=\{x_1,\dots,x_N\}\subset\Omega$ be a set of trial points with fill distance $h_{X,\Omega}$. Let $\{\mathcal{P}_N\}$ be a family of interpolant operators from $C(\Omega)$ to $V_N=\operatorname{span}\{a_1,\dots,a_N\}$ on test points $Y=\{y_1,\dots,y_N\}\subset\Omega$, where a_j are the MLS shape functions based on X and polynomial space $\mathbb{P}^d_m$. Assume that the distribution of points is well enough to ensure the non-singularity of A_N, and $\|A_N^{-1}\|_\infty=\mathcal{O}(1)$. Further, assume that $\{\mathcal{F}_N\}$ in (2.4) is a collectively compact family of pointwise convergent operators on $C(\Omega)$. Finally, assume that $(\lambda-\mathcal{F})u=f$ is uniquely solvable for all $f\in C(\Omega)$. Then for all sufficiently large N, say $N\ge N_0$, the operator $(\lambda-\mathcal{F}_N\mathcal{P}_N)^{-1}$ exists and it is uniformly bounded. In addition, for the iterative solution v_N for equation $(\lambda-\mathcal{F}_N\mathcal{P}_N)v_N=f$ we have*

$$\|u-v_N\|_\infty\le c_I\big\{\|\mathcal{F}_Nu-\mathcal{F}u\|_\infty+c_F(1+c_P)Ch^{m+1}_{X,\Omega}|u|_{C^{m+1}_\infty(\Omega^*)}\big\},$$

provided that $u\in C^{m+1}_\infty(\Omega^*)$ where Ω^* and $|u|_{C^{m+1}_\infty(\Omega^*)}$ are defined in Theorem 1.6. Moreover, for the discrete collocation solution u_N of equation $(\lambda-\mathcal{P}_N\mathcal{F}_N)u_N=\mathcal{P}_Nf$ we have

$$\|u-u_N\|_\infty\le c_I\big\{c_P\|\mathcal{F}_Nu-\mathcal{F}u\|_\infty+(1+c_Pc_F)c_F(1+c_P)Ch^{m+1}_{X,\Omega}|u|_{C^{m+1}_\infty(\Omega^*)}\big\},$$

where $c_I<\infty$ is a bound for $(\lambda-\mathcal{F}_N\mathcal{P}_N)^{-1}$.

Proof. According to the assumptions and using Lemma 2.5 we conclude that $\{\mathcal{F}_N\mathcal{P}_N\}$ is a collectively compact family of pointwise convergent operators. By [3, Lemma 4.1.2] and the discussions after that, we have

$$\|(\mathcal{F}-\mathcal{F}_N\mathcal{P}_N)\mathcal{F}_N\mathcal{P}_N\| \to 0.$$

Thus, (2.10) is satisfied for $N \geq N_0$ if we insert $\mathcal{T} = \mathcal{F}$ and $\mathcal{S} = \mathcal{F}_N\mathcal{P}_N$ in to Theorem 2.1. Since $(\lambda - \mathcal{F})u = f$ is uniquely solvable we have $(\lambda - \mathcal{F})^{-1} \leq c_0 < \infty$. On the other hand $\|\mathcal{F}_N\mathcal{P}_N\| \leq c_P c_F$. Consequently, (2.11) implies

$$\|(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}\| \leq \sup_{N\geq N_0} \frac{1 + c_0 c_P c_F}{|\lambda| - c_0\|(\mathcal{F}-\mathcal{F}_N\mathcal{P}_N)\mathcal{F}_N\mathcal{P}_N\|} := c_I < \infty,$$

which proves the first assertion. If we set $w = u$ and $z = v_N$ in (2.12) then

$$\begin{aligned}
\|u - v_N\|_\infty &\leq \|(\lambda - \mathcal{F}_N\mathcal{P}_N)^{-1}\|\|\mathcal{F}u - \mathcal{F}_N\mathcal{P}_N u\|_\infty \\
&\leq c_I\big\{\|\mathcal{F}u - \mathcal{F}_N u\|_\infty + \|\mathcal{F}_N(u - \mathcal{P}_N u)\|_\infty\big\} \\
&\leq c_I\big\{\|\mathcal{F}u - \mathcal{F}_N u\|_\infty + c_F\|u - \mathcal{P}_N u\|_\infty\big\} \\
&\leq c_I\big\{\|\mathcal{F}u - \mathcal{F}_N u\|_\infty + c_F(\|u - \widehat{u}\|_\infty + \|\mathcal{P}_N u - \mathcal{P}_N\widehat{u}\|_\infty)\big\} \\
&\leq c_I\big\{\|\mathcal{F}u - \mathcal{F}_N u\|_\infty + c_F(1 + c_P)\|u - \widehat{u}\|_\infty\big\} \\
&\leq c_I\big\{\|\mathcal{F}u - \mathcal{F}_N u\|_\infty + c_F(1 + c_P)Ch_{X,\Omega}^{m+1}|u|_{C_\infty^{m+1}(\Omega^*)}\big\}.
\end{aligned}$$

The last inequality follows from Theorem 1.12. Moreover,

$$\begin{aligned}
u - u_N = u - \mathcal{P}_N v_N &= (u - \mathcal{P}_N u) + \mathcal{P}_N(u - v_N), \\
\|u - u_N\|_\infty &\leq \|u - \mathcal{P}_N u\|_\infty + c_P\|u - v_N\|_\infty
\end{aligned}$$

which finishes the proof. □

Theorem 2.6 shows that, both the quadrature and the MLS approximation error bounds affect the final estimation. If for a sufficiently smooth kernel $\kappa(x, s)$ a high order quadrature is employed then the total error is dominated by the error of the MLS approximation.

2.3 Domain decomposition

For numerical integration, Ω can be partitioned to measurable subdomains Ω_ℓ such that $\Omega = \cup_{\ell=1}^{L}\Omega_\ell$ and $\Omega_\ell \cap \Omega_k = \emptyset$ for $1 \leqslant k, \ell \leqslant L$. For an integrable function g on Ω we then have

$$\int_{\Omega} g(x)dx = \sum_{\ell=1}^{L} \int_{\Omega_\ell} g(x)dx.$$

Usually we can transfer the integration on Ω_ℓ to an integration on the unit cube for applying a tensor product numerical integration rule. For instance in a 2D problem without lose of generality we assume

$$\Omega_\ell = \{(\mathsf{x}_1, \mathsf{x}_2) \in \mathbb{R}^2 : -1 \leqslant \mathsf{x}_2 \leqslant 1,\ g_\ell(\mathsf{x}_2) \leqslant \mathsf{x}_1 \leqslant f_\ell(\mathsf{x}_2)\}$$

for continuous functions g_ℓ and f_ℓ. Now the linear transformation

$$\xi^{(\ell)}(\mathsf{x}_2, \theta) = \frac{f_\ell(\mathsf{x}_2) - g_\ell(\mathsf{x}_2)}{2}\theta + \frac{f_\ell(\mathsf{x}_2) + g_\ell(\mathsf{x}_2)}{2}$$

converts the integration on Ω_ℓ onto an integration on $[-1, 1]^2$ with

$$\int_{\Omega_\ell} g(x)dx = \frac{1}{2}\int_{-1}^{1}\int_{-1}^{1} [f_\ell(\mathsf{x}_2) - g_\ell(\mathsf{x}_2)] g(\xi^{(\ell)}(\mathsf{x}_2, \theta), \mathsf{x}_2) d\theta d\mathsf{x}_2.$$

The same formulation can be adapted for a three dimensional domain.

We note that this is a domain decomposition in the test space (integration) and not in the trial space (approximation) where the MLS shape functions are calculated independent of this domain decomposition.

2.4 Application to Volterra integral equations

The proposed method can be simply applied for solving the Volterra-type integral equations of the second kind. For simplicity we first discuss the one dimensional case.

1-D Volterra integral equation

Consider the following second kind linear Volterra integral equation

$$\lambda u(x) + \int_a^x \kappa(x, s)u(s)ds = f(x), \quad x \in [a, b] \subset \mathbb{R}. \tag{2.14}$$

To apply the method, N nodal points $\{x_1, \ldots, x_N\}$ are selected in interval $[a, b]$ and the MLS approximation

$$\widehat{u}(x) = \sum_{j=1}^{N} a_j(x)u(x_j)$$

is formed based on these points. If $\widehat{u}$ is inserted to (2.14) instead of u we have

$$\lambda\widehat{u}(x) + \int_a^x \kappa(x, s)\widehat{u}(s)ds = f(x) + r(x), \quad x \in [a, b], \tag{2.15}$$

where $r(x)$ is the reminder at x. Now we collocate (2.15) at nodal points x_k to obtain by assuming $\mathcal{P}_N r = r(x_k) = 0$,

$$\sum_{j=1}^{N} \left[\lambda a_j(x_k) + \int_a^{x_k} \kappa(x_k, s)a_j(s)ds\right] u(x_j) = f(x_k), \quad 1 \le k \le N. \tag{2.16}$$

Now all intervals $[a, x_k]$ are transferred to a fix interval $[a, b]$ by the simple linear transformation

$$s(x_k, \theta) = \frac{x_k - a}{b - a}\theta + \frac{b - x_k}{b - a}a.$$

Equation (2.16) now takes the form

$$\sum_{j=1}^{N} \left[\lambda a_j(x_k) + \int_a^b K(x_k, s(x_k, \theta))\ a_j(s(x_k, \theta))d\theta\right] u(x_j) = f(x_k), \quad 1 \le k \le N,$$

where

$$K(x, s(x, \theta)) = \frac{x - a}{b - a}\kappa(x, s(x, \theta)).$$

Using a Q_N-point quadrature formula with points $\{\theta_\ell\}$ and weights $\{\omega_\ell\}$ in interval $[a, b]$ we have

$$\sum_{j=1}^{N} \left[\lambda a_j(x_k) + \sum_{\ell=1}^{Q_N} K(x, s(x_k, \theta_\ell))\ a_j(s(x_k, \theta_\ell))\ \omega_\ell\right] \widetilde{u}_j = f(x_k), \quad 1 \le k \le N, \tag{2.17}$$

where the error of numerical integration forces $u(x_j)$ to be replaced by approximate values $\widetilde{u}_j$. If we define the N by N matrix F by

$$F_{k,j} = \lambda a_j(x_k) + \sum_{\ell=1}^{Q_N} K(x_k, s(x_k, \theta_\ell))\; a_j(s(x_k, \theta_\ell))\; \omega_\ell,$$

and N vectors

$$\widetilde{\boldsymbol{u}} = (\widetilde{u}_1, \widetilde{u}_2, \ldots, \widetilde{u}_N)^T,$$

and

$$\boldsymbol{f} = (f(x_1), \ldots, f(x_N))^T,$$

then we have the following linear system of equations

$$F\widetilde{\boldsymbol{u}} = \boldsymbol{f}. \tag{2.18}$$

If F is nonsingular then $\widetilde{\boldsymbol{u}}$ is uniquely determined and the values of u at any point $x \in [a, b]$ can be approximated by

$$u(x) \approx u_N(x) = \sum_{j=1}^{N} a_j(x)\widetilde{u}_j, \quad x \in [a, b]. \tag{2.19}$$

2-D Volterra integral equation

The two dimensional Volterra integral equation can be expressed as

$$u(x, y) + \int_c^y \int_a^x \kappa(x, y, \xi, \eta)u(\xi, \eta)d\xi d\eta = f(x, y), \quad (x, y) \in [a, b] \times [c, d].$$

where we use the notation (x, y) or (ξ, η) for a point in $\mathbb{R}^2$. This violates our usual notation but simplifies the presentation. A set of N points in the 2-D domain $[a, b] \times [c, d]$ is expressed by $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. Using the linear transformations and techniques employed in 1-D case the final linear system is obtained as

$$F_{k,j} = a_j(x_k, y_k) + \sum_{\ell=1}^{Q_N}\sum_{q=1}^{Q_N} K(x_k, y_k, \xi(x_k, \theta_q), \eta(y_k, \zeta_\ell))a_j(\xi(x_k, \theta_q), \eta(y_k, \zeta_\ell))\omega_q\omega_\ell,$$

where

$$\xi(x,\theta) = \frac{x-a}{b-a}\theta + \frac{b-x}{b-a}a,$$

$$\eta(y,\zeta) = \frac{y-c}{d-c}\zeta + \frac{d-y}{d-c}c,$$

and

$$K(\cdot,\cdot,\cdot,\cdot) = \frac{x_k-a}{b-a}\frac{y_k-c}{d-c}\kappa(\cdot,\cdot,\cdot,\cdot).$$

The error analysis of the above algorithm is not given here but is left for a future work.

Integro-differential equations

Integro-differential equations contain both integral and differential operators where derivative are either inside or outside the integrand. These types of equations were introduced by Volterra in early 1900 to model some population dynamics. Afterward, more applications in heat transfer and diffusion phenomena (for instance in Neutron diffusion theory) where found.

MLS can be simply applied for solving an integro-differential equation. We give details for a one dimensional Fredholm-type equation because the extensions to higher dimensions and Volterra-type equations are straightforward. We consider a Fredholm integro-differential equation of the form

$$u^{(n)} + \int_a^b \kappa(x,s)u^{(k)}(s)ds = f(x), \quad x \in [a,b], \tag{2.20}$$

$$u^{(i)}(a) = b_i, \quad 0 \leqslant i \leqslant n-1, \tag{2.21}$$

where $u^{(i)}$ are i-th derivative of u and b_i are constant initial values. Here $n,k \in \mathbb{N}_0$ are fixed and $n \geqslant k$.

As before, u is replaced by the the MLS approximation $\widehat{u}$ but the initial conditions (2.21) should be imposed, properly. For this we use the following simple collocation

$$\sum_{j=1}^{N} a_j^{(i)}(a)\widetilde{u}_j = b_i, \quad 0 \leqslant i \leqslant n$$

which imposes n linear equations. Collocating (2.20) at M test points

$y_1, \ldots, y_M \in [a,b]$ then leads to $F\widetilde{\boldsymbol{u}} = \boldsymbol{b}$ with $(M+n) \times N$ final matrix

$$F_{i,j} = \begin{cases} a_j^{(i)}(a), & 1 \leqslant i \leqslant n, \\ a_j^{(n)}(y_{i-n}) + \displaystyle\sum_{\ell=1}^{Q_N} \kappa(y_{i-n}, \theta_\ell) a_j^{(k)}(\theta_\ell)\omega_\ell, & n+1 \leqslant i \leqslant M+n \end{cases}$$

for $1 \leqslant j \leqslant N$ and $\boldsymbol{b} = [b_1, \ldots, b_n, f(y_1), \ldots, f(y_M)]^T$. If $M = N - n$ test points are chosen then F is a square matrix. Since the equation contains derivatives up to order n the degree of polynomial basis functions, m, should satisfy $m \geqslant n$. The error analysis is not given here but is left for a future work.

2.5 Numerical results

In this section some numerical examples involving Fredholm and Volterra integral equations in one and two dimensions are considered. Regular or irregular sets points with fill distances h are used. In all cases the Gaussian weight function (1.13) with $c = 0.6h$ and shifted and scaled basis function (1.17) of degree m are employed. The size of MLS weight support is chosen to be $\delta = 2mh$. The same test and trial points (i.e. $X = Y$) are utilized and a 10-point Gauss-Legendre quadrature (and its corresponding tensor product rule in 2D) is applied for numerical integrations.

1D Fredholm equation

Consider the integral equation (2.1) with $\Omega = [0,1]$, $\lambda = 5$ and $\kappa = \exp(xs)$. We use the true solution

$$u(x) = \exp(-x)\cos(x),$$

and define $f(x)$ accordingly. Since $\|\mathcal{F}\| = e - 1 \simeq 1.72$, the integral equation is uniquely solvable for any given $f \in C[0,1]$. Numerical results for $m = 1, 2, 3$ and different numbers of meshless points N (regular points with fill distance $h = 1/(N-1)$) are given in Table 2.1. Errors are measured in the maximum norm on a large evaluation set point. Since h is halved row by row, the orders

Table 2.1: Maximum errors and convergence orders; 1D Fredholm equation

	$m=1$		$m=2$		$m=3$	
h	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders
0.2	2.08×10^{-3}	–	3.68×10^{-4}	–	7.75×10^{-5}	–
$\frac{0.2}{2}$	5.79×10^{-4}	1.84	5.63×10^{-5}	2.71	7.74×10^{-6}	3.36
$\frac{0.2}{4}$	1.53×10^{-4}	1.92	8.53×10^{-6}	2.72	5.75×10^{-7}	3.71
$\frac{0.2}{8}$	3.54×10^{-5}	2.09	1.12×10^{-6}	2.93	3.69×10^{-8}	3.96
$\frac{0.2}{16}$	8.56×10^{-6}	2.07	1.41×10^{-7}	2.99	2.40×10^{-9}	3.94
$\frac{0.2}{32}$	2.19×10^{-6}	1.97	1.78×10^{-8}	2.98	1.54×10^{-10}	3.96
$\frac{0.2}{64}$	5.71×10^{-7}	1.94	2.27×10^{-9}	2.96	9.86×10^{-12}	3.96

are computed via

$$\log_2\left(\frac{\|e(h)\|_\infty}{\|e(h/2)\|_\infty}\right). \tag{2.22}$$

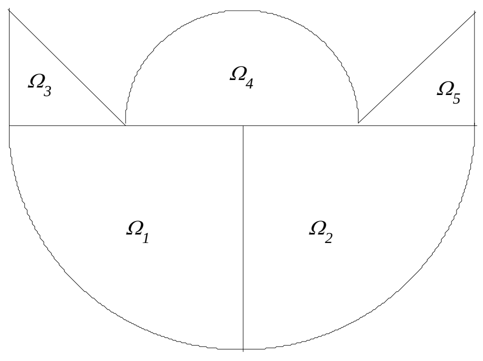
In [26] the convergence rate for $m = 2$ was influenced by an instability that here is overcome by using the shifted and scaled basis functions. As proved in Theorem 2.6, the order of convergence is $\mathcal{O}(h^{m+1})$ provided that an accurate quadrature rule is applied.

2D Fredholm equation in a non-rectangular domain

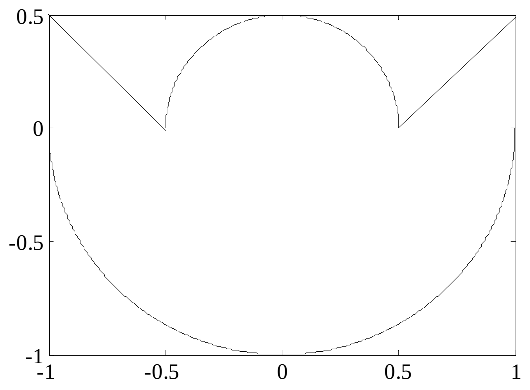
Consider the Fredholm integral equation (2.1) for $\Omega \subset \mathbb{R}^2$ where Ω is a non-rectangular domain shown in Figure 2.1 (a). The kernel $\kappa(\mathsf{x}_1, \mathsf{x}_2, \mathsf{s}_1, \mathsf{s}_2) = \exp(\mathsf{x}_1 + \mathsf{x}_2)\cos(\mathsf{s}_1 + \mathsf{s}_2)$ is assigned and the true solution is assumed to be $u(\mathsf{x}_1, \mathsf{x}_2) = \sin(\mathsf{x}_1 + \mathsf{x}_2)$. The right-hand side then is $f(\mathsf{x}_1, \mathsf{x}_2) = \lambda \sin(\mathsf{x}_1 + \mathsf{x}_2) - \beta \exp(\mathsf{x}_1 + \mathsf{x}_2)$ with $\beta \doteq 0.254801287003867$. As is shown in Figure 2.1 (b), Ω is partitioned to five subdomains. Scattered points of various fill distances are also depicted in Figure 2.1 (c)-(f). Since $\|\mathcal{F}\| = 5\pi/8 + 1/4 \doteq 2.21$, we set $\lambda = 4$. Numerical results are given in Table 2.2.

1-D Volterra equation

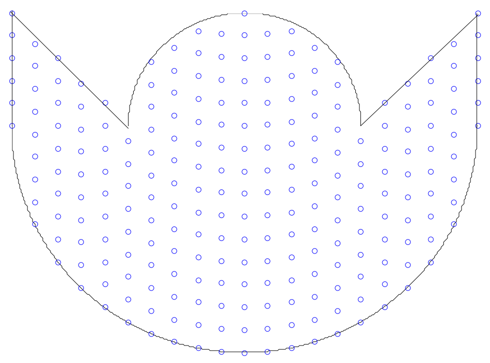
Consider the Volterra integral equation (2.14) with $a = 0$, $\lambda = 3$, $\kappa = (x + s)^2$ and true solution $u(x) = \exp(x)$. The right-hand side function f is calculated accordingly. All MLS parameters are chosen as those for the 1-D Fredholm example. Results are given in Table 2.3. Although we did not prove the error



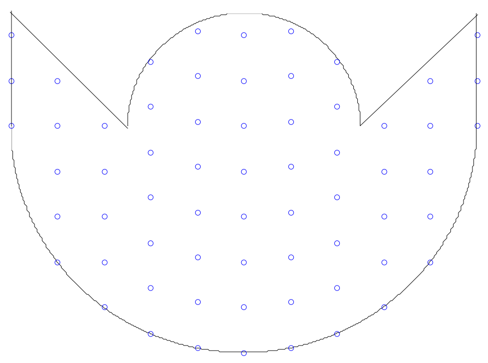
(ب)



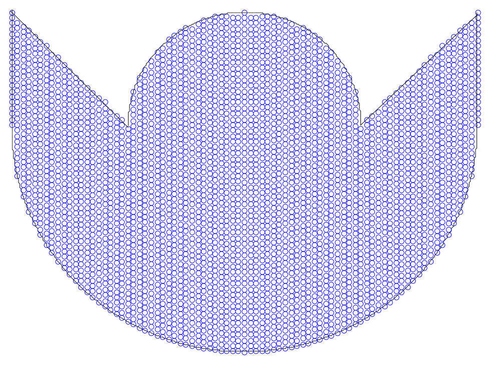
(الف)



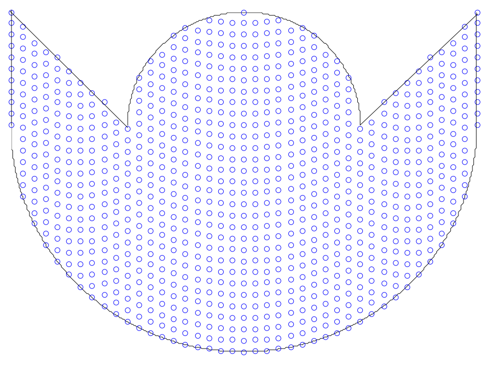
(ت)



(پ)



(ج)



(ث)

Figure 2.1: Domain, subdomains and meshless points for 2D Fredholm equation.

Table 2.2: Maximum errors and convergence orders; 2D Fredholm equation

		$m = 1$		$m = 2$		$m = 3$	
N	h	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders
64	0.2	4.69×10^{-2}	–	1.62×10^{-2}	–	2.98×10^{-4}	–
238	$\frac{0.2}{2}$	1.46×10^{-2}	1.68	3.09×10^{-3}	1.98	3.06×10^{-4}	3.28
917	$\frac{0.2}{4}$	2.62×10^{-3}	2.48	4.43×10^{-4}	2.80	3.41×10^{-5}	3.17
3603	$\frac{0.2}{8}$	2.66×10^{-4}	3.30	1.94×10^{-5}	4.51	1.71×10^{-7}	7.49

Table 2.3: Maximum errors and convergence orders; 1D Volterra equation

	$m = 1$		$m = 2$		$m = 3$	
h	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders
0.2	4.60×10^{-3}	–	6.31×10^{-4}	–	6.07×10^{-5}	–
$\frac{0.2}{2}$	1.36×10^{-3}	1.76	9.09×10^{-5}	2.79	4.38×10^{-6}	3.79
$\frac{0.2}{4}$	3.77×10^{-4}	1.85	1.02×10^{-5}	3.16	2.89×10^{-7}	3.93
$\frac{0.2}{8}$	9.07×10^{-5}	2.05	1.59×10^{-6}	2.68	1.76×10^{-8}	4.03
$\frac{0.2}{16}$	2.46×10^{-5}	1.88	2.05×10^{-7}	2.95	1.17×10^{-9}	3.91
$\frac{0.2}{32}$	6.04×10^{-6}	2.03	2.59×10^{-8}	2.98	7.42×10^{-11}	3.98
$\frac{0.2}{64}$	1.63×10^{-6}	1.89	3.53×10^{-9}	2.88	4.89×10^{-12}	3.92

bound in this case, the experimental results predict the optimal rate h^{m+1}.

2-D Volterra equation

This example covers experimental results of a two-dimensional Volterra integral equation (2.4) for $a = c = 0$, $b = d = 1$, $\lambda = 1$, $\kappa = \cos(2\xi + \eta)$ and true solution $u(x, y) = \sin(2x + y)$. Regular node distributions on $[0, 1] \times [0, 1]$ are used and results are reported in Table 2.4. Again the experimental rate h^{m+1} is observed.

Table 2.4: Maximum errors and convergence orders; 2D Volterra equation

	$m = 1$		$m = 2$		$m = 3$	
h	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders	$\|e\|_\infty$	orders
0.2	1.39×10^{-2}	–	2.47×10^{-3}	–	7.19×10^{-4}	–
$\frac{0.2}{2}$	3.51×10^{-3}	1.98	3.58×10^{-4}	2.78	4.35×10^{-5}	4.04
$\frac{0.2}{4}$	8.86×10^{-4}	1.99	4.63×10^{-5}	2.95	2.72×10^{-6}	4.00
$\frac{0.2}{8}$	2.26×10^{-4}	1.97	5.90×10^{-6}	2.97	1.74×10^{-7}	3.97

Table 2.5: Maximum errors and convergence orders; 1D integro-differential equation

	$m=1$		$m=2$		$m=3$	
h	$\Vert e\Vert_\infty$	orders	$\Vert e\Vert_\infty$	orders	$\Vert e\Vert_\infty$	orders
0.2	9.49×10^{-2}	–	1.94×10^{-2}	–	4.60×10^{-3}	–
$\frac{0.2}{2}$	3.92×10^{-2}	1.28	6.56×10^{-3}	1.56	1.80×10^{-4}	4.68
$\frac{0.2}{4}$	1.17×10^{-2}	1.75	2.46×10^{-4}	4.64	1.50×10^{-5}	3.58
$\frac{0.2}{8}$	4.05×10^{-3}	1.50	3.65×10^{-4}	−0.47	1.36×10^{-6}	3.46
$\frac{0.2}{16}$	1.95×10^{-3}	1.05	6.62×10^{-5}	1.92	1.77×10^{-7}	2.94
$\frac{0.2}{32}$	2.17×10^{-4}	3.17	2.30×10^{-5}	2.06	9.96×10^{-8}	0.83

Integro-differential equation

Finally, we consider equations (2.20) and (2.21) for $n = k = 1$, $a = 0$, $b = 1$ and $\kappa = \exp(xs)$ with true solution $u(x) = x\exp(x)$. Regular node distribution with fill distance $h = 1/(N+1)$ on interval $[0,1]$ are used as trial set X. The test point set is $Y = X \setminus \{0\}$. At 0 the initial condition is imposed. Results are reported in Table 2.5. We guess the convergence rate $h^{m+1-n} = h^m$ in this case.

2.6 Conclusion

For numerical solution of an integral equation, If either the equation is posed on a non-rectangular domain or known information is only available at scattered points, the use of meshless methods such as MLS approximation is recommendable. Although the proposed method works also appropriately in other cases, we suggest more accurate techniques, such as spectral methods, for solving these types of integral equations.

Chapter 3

Local PDE solvers based on MLS

Numerical methods based on MLS for solving PDEs may be classified to *global* and *local* methods. In general, this classification can be used for both trial and test spaces. Here, the trial space is formed via MLS shape functions which construct a "local polynomial reproduction" system as pointed in Chapter 1. Thus by a local method we refer to a method that is local in the test side. This title is assigned for weak-based methods where a *local weak form* is used instead of a global weak formulation on the whole PDE domain. As a global meshless method in this scenario we can mention the Element-Free Galerkin (EFG) method which is meshless is the trial side but still mesh-based in the test side. In this method the numerical integration is based on a background triangulation. Thus EFG is not known as a *truly* meshless method. On the other side, the Meshless Local Petrov-Galerkin (MLPG) methods are based on a local weak formulation that leads to a more simple numerical integration. In these methods integrations are done over independent (non-connected) and well-shaped local subdomains. Thus, triangulation does not require for either approximation or integration. Primer attempts to develop such techniques go back to Meshless Local Boundary Integral Equation (MLBIE) [45] and MLPG [7] methods in 1998.

In this chapter we first give an overview on meshless methods and then the derivation of local weak forms are discussed and variations of MLPG methods are reviewed.

3.1 Meshless Methods

As pointed before, whatever the given problem is, meshless methods construct solutions from a *trial space* U whose functions are parametrized entirely in terms of nodes [9]. We let these nodes form a set $X := \{x_1, \dots, x_N\}$. Then the functions u of the linear trial space U are parametrizable by their values on X iff the linear functionals $\delta_{x_1}, \dots, \delta_{x_N}$ are linearly independent on U. This implies that there must be some basis $u_1, \dots, u_N$ of U such that the $N \times N$ matrix of values $u_j(x_k)$ is invertible, but we are not interested in knowing or constructing this basis. We only assume that the discretized problem is set up with a vector

$$\boldsymbol{u} = (u(x_1), \dots, u(x_N))^T$$

of unknowns in "meshless" style, and all data have to be expressed in terms of these. Furthermore, we assume the discretized problem to consist of equations

$$\lambda_k(u) = \beta_k, \ 1 \le k \le M, \tag{3.1}$$

where we have $M \ge N$ linear functionals $\lambda_1, \dots, \lambda_M$ and M prescribed real values $\beta_1, \dots, \beta_M$. Section 3.2 will describe how this is done for standard linear PDE problems, including the variations of the MLPG. This is a discretization in the *test space*. If u is replaced by a good trial approximation

$$\widehat{u}(x) = \sum_{j=1}^{N} a_j(x) u(x_j)$$

then

$$\lambda_k(\widehat{u}) = \sum_{j=1}^{M} \lambda_k(a_j) u(x_j) \approx \lambda_k(u) \text{ for all } k, \ 1 \le k \le M. \tag{3.2}$$

Putting the $\lambda_k(a_j)$ into an $M \times N$ matrix K, one has to solve the possibly overdetermined linear system

$$K\,\boldsymbol{u} = \boldsymbol{b} \tag{3.3}$$

with $\boldsymbol{b} = (\beta_1, \ldots, \beta_M)^T$. In a second stage, users might want to evaluate u at other places than in the nodes x_j. This is a problem of recovery of functions from discrete data values, and completely independent of PDE solving. There are various possibilities to do so, including the standard MLS with its shape functions.

3.2 Local weak forms

We now write linear PDE problems in the discretized form (3.1), with special emphasis on the MLPG method.

Although the technique proposed in this section can be used for a wide class of PDEs, we illustrate our approach for the Poisson problem

$$\begin{array}{rcll} \Delta u(x) & = & f(x), & x \in \Omega, \\ u(x) & = & u_D(x), & x \in \Gamma_D, \\ \frac{\partial u}{\partial n}(x) & = & u_N(x), & x \in \Gamma_N \end{array} \tag{3.4}$$

where f is a given source function, the bounded domain $\Omega \subset \mathbb{R}^d$ is enclosed by the boundary $\Gamma = \Gamma_D \cup \Gamma_N$, u_D and u_N are the prescribed Dirichlet and Neumann data, respectively, on the Dirichlet boundary Γ_D and on the Neumann boundary Γ_N, while n is the outward normal direction.

The simplest way of discretizing the problem in the form (3.1) is direct and global collocation. In addition to the trial nodes $x_1, \ldots, x_N$ for obtaining nodal solution values, we can choose finite point sets

$$Y_\Omega \subset \Omega,\ Y_D \subset \Gamma_D,\ Y_N \subset \Gamma_N,\ Y := Y_\Omega \cup Y_D \cup Y_N,\ |Y| = M$$

and discretize the problem by M functionals

$$\begin{array}{rcrcll} \lambda_i(u) & = & \Delta u(y_i) & = & f(y_i), & y_i \in Y_\Omega \subset \Omega, \\ \lambda_j(u) & = & u(y_j) & = & u_D(y_j), & y_j \in Y_D \subset \Gamma_D, \\ \lambda_k(u) & = & \frac{\partial u}{\partial n}(y_k) & = & u_N(y_k), & y_k \in Y_N \subset \Gamma_N \end{array} \tag{3.5}$$

using some proper indexing scheme. In MLPG categories, this is MLPG2 [4, 5]. All functionals are local, and *strong* in the sense that they do not involve integration over test functions.

For FEM–style global weak discretization, one can keep the second and third part of (3.5), but the first can be weakened using the Divergence Theorem. With sufficiently smooth test functions v_k, we get

$$\lambda_k(u) := \int_\Gamma (\nabla u \cdot n) v_k \, d\Gamma - \int_\Omega \nabla u \cdot \nabla v_k \, d\Omega = \int_\Omega f v_k \, d\Omega$$

as a replacement of the first functionals in (3.5), leading again to (3.1).

Following the original MLPG method, instead of transforming (3.4) into a global weak form, we construct weak forms over local subdomains Ω_σ^k which are small regions taken around nodes y_k in the global domain Ω. The local subdomains could theoretically be of any geometric shape and size. But for simplicity they are often taken to be balls $B(y_k, \sigma)$ intersected with Ω and centered at y_k with radius σ, or squares in 2D or cubes in 3D centered at y_k with sidelength σ, denoted by $S(y_k, \sigma) \cap \Omega$. The variable σ parametrizes the local subdomain's size, and we denote the boundary within Ω by $\Gamma_\sigma^k := \Omega \cap \partial\Omega_\sigma^k$. We call a node y_k *internal*, if the boundary $\partial\Omega_\sigma^k$ of the local subdomain Ω_σ^k does not intersect Γ.

The derivation of the local weak form starts with the local integral

$$\lambda_k(u) := \int_{\Omega_\sigma^k} \Delta u \, v \, d\Omega = \int_{\Omega_\sigma^k} f \, v \, d\Omega =: \beta_k, \tag{3.6}$$

where v is an appropriate test function on Ω_σ^k. Employing the Divergence Theorem, we get an equation

$$\begin{aligned} \lambda_k(u) &:= \int_{\Gamma_\sigma^k \setminus \Gamma_N} (\nabla u \cdot n) v \, d\Gamma - \int_{\Omega_\sigma^k} \nabla u \cdot \nabla v \, d\Omega \\ &= \int_{\Omega_\sigma^k} f v \, d\Omega - \int_{\Gamma_\sigma^k \cap \Gamma_N} u_N v \, d\Gamma =: \beta_k \end{aligned} \tag{3.7}$$

of the form (3.1). For nodes whose subdomain boundary does not intersect Γ_N, the second term on the left–hand side vanishes.

To impose the Dirichlet boundary conditions we can introduce a penalty term

$$\varpi \int_{\Gamma_\sigma^k \cap \Gamma_D} (u - u_D) v d\Gamma$$

in (3.7) to get

$$\begin{aligned}\lambda_k(u) :=& \int_{\Gamma_\sigma^k \setminus \Gamma_N} (\nabla u \cdot n) v \, d\Gamma - \int_{\Omega_\sigma^k} \nabla u \cdot \nabla v \, d\Omega + \varpi \int_{\Gamma_\sigma^k \cap \Gamma_D} uv \, d\Gamma \\ =& \int_{\Omega_\sigma^k} fv \, d\Omega - \int_{\Gamma_\sigma^k \cap \Gamma_N} u_N v \, d\Gamma + \varpi \int_{\Gamma_\sigma^k \cap \Gamma_D} u_D v \, d\Gamma =: \beta_k.\end{aligned} \tag{3.8}$$

The penalty parameter ϖ must be chosen large enough to impose the boundary condition $u = u_D$ with a reasonable accuracy. Usually, an optimal value for this parameter is not accessible, theoretically. As we will see, the final linear system of MLPG methods is unsymmetric with or without the penalty term. Thus, another possibility for imposing the Dirichlet boundary conditions, that does not violate the final structure of the stiffness matrix, is the use of a direct collocation method. In this case the Dirichlet boundary conditions are imposed directly using the second line of (3.5) for suitable collocation points, usually taking a subset of the trial nodes. Thus λ_k and β_k are defined as (3.7) and neither Lagrange multipliers nor penalty parameters are introduced into the local weak forms.

By applying the Divergence Theorem twice, the second local weak form is obtained as

$$\begin{aligned}&\int_{\Gamma_\sigma^k \setminus \Gamma} \frac{\partial u}{\partial n} v \, d\Gamma - \int_{\Gamma_\sigma^k \setminus \Gamma} u \frac{\partial v}{\partial n} \, d\Gamma + \int_{\Gamma_\sigma^k \cap \Gamma_N} u_N v \, d\Gamma + \int_{\Gamma_\sigma^k \cap \Gamma_D} \frac{\partial u}{\partial n} v \, d\Gamma \\ &\quad - \int_{\Gamma_\sigma^k \cap \Gamma_N} u \frac{\partial v}{\partial n} \, d\Gamma - \int_{\Gamma_\sigma^k \cap \Gamma_D} u_D \frac{\partial v}{\partial n} \, d\Gamma + \int_{\Omega_\sigma^k} u \Delta v \, d\Omega - \int_{\Omega_\sigma^k} fv \, d\Omega = 0.\end{aligned}$$

In this case $\Omega_\sigma^k = B(y_k, \sigma) \cap (\Omega \cup \Gamma)$ and the test function v is shift invariant and compactly supported on Ω_σ^k with center y_k such that $\Delta v(\cdot - y_k) = -\delta(\cdot - y_k)$, where δ is the Dirac delta function (distribution) and $v = 0$ on $\Gamma_\sigma^k \setminus \Gamma$. Such test function is called the *companion solution* and is given as

$$v(x-y) = \begin{cases} -\dfrac{1}{2\pi} \ln \dfrac{r}{\sigma}, & \text{in 2D} \\ \dfrac{1}{4\pi} \left(\dfrac{1}{r} - \dfrac{1}{\sigma} \right), & \text{in 3D} \end{cases} \qquad r = \|x - y\|_2. \tag{3.9}$$

Using the companion solution we arrive at the following test discretization

$$\begin{aligned}\lambda_k(u) :=&c_k u(y_k) - \fint_{\Gamma_\sigma^k \setminus \Gamma} u\frac{\partial v}{\partial n}\, d\Gamma + \int_{\Gamma_\sigma^k \cap \Gamma_N} u\frac{\partial v}{\partial n}\, d\Gamma \int_{\Gamma_\sigma^k \cap \Gamma_D} \frac{\partial u}{\partial n} v\, d\Gamma \\ &+ \int_{\Gamma_\sigma^k \cap \Gamma_N} u_N v\, d\Gamma + \int_{\Gamma_\sigma^k \cap \Gamma_D} u_D \frac{\partial v}{\partial n}\, d\Gamma + \int_{\Omega_\sigma^k} f v\, d\Omega =: \beta_k.\end{aligned} \tag{3.10}$$

where $\fint$ represents the Cauchy Principle Value (CPV) integral (see the BEM monograph [36] for more details). Here $c_k = 1$ if y_k is an internal point and $c_k = 1/2$ for y_k located on smooth parts of the boundary. If y_k is a corner boundary point in a 2D domain then $c_k = \theta/(2\pi)$ where θ is the internal angle of the corner point. Of course, for internal points the CPV integral is replaced by its corresponding regular integral.

Variations of MLPG differ in their choice of functionals. We already discussed about the MLPG2. MLPG1, MLPG5 and MLPG6 are based on locally weak functionals (3.7) (or (3.8) in penalty form). If the test function v is chosen to vanish on $\Gamma_\sigma^y \setminus \Gamma_N$, the first integral in (3.7) is zero, and we have MLPG1. If the local test function v is the constant 1, the second integral vanishes, and we have MLPG5. In MLPG6 the same trial functions are used for test functions. Thus σ is identical to the support of trial weight functions. MLPG3 is based on (3.6) and the trial approximation error function is used as a test function. Since in MLPG3 and MLPG6 trial and test functions are coming from a same approximation space they are better to be called MLG3 and MLG6, respectively. Finally, MLPG4 is based on local test functionals (3.10) with companion solutions (3.9). More details about MLPG classification may be found in [6, 5].

Although, all meshless approximation spaces in the market can be used as trial approximation, usually the MLS approximation

$$\widehat{u}(x) = \sum_{j=1}^{N} a_j(x)u(x_j) \approx u(x), \quad x \in \Omega,$$

where a_j are MLS shape functions, is used to discretize the trial space in MLPG methods. Considering any of the above local weak forms $\lambda_k(u) = \beta_k$, $1 \leqslant k \leqslant M$, and replacing u by $\widehat{u}$, the final linear system (3.3) is resulted where $K_{k,j} = \lambda_k(a_j)$ for $1 \leqslant k \leqslant M$ and $1 \leqslant j \leqslant N$. To get a full-rank and stable system, theoretically M (number of test functionals) must be larger than N (number of trial points). But usually the square case $N = M$ works without any serious problem. The final matrix in all MLPG methods is unsymmetric even if the symmetric weak forms are utilized. For more details around the theoretical background of some MLPG methods we refer the reader to [38, 39].

There exist numerous papers on application of MLPG methods for solving PDE problems in science and engineering. As some contributed work by the present author, in [25] the MLPG4 has been applied for solving the sine-Gordon equation, in [27] an application of MLPG5 to p-Laplace equation has been investigated and in [28] an algorithm that uses the MLS approximation in both time and space domains has been developed.

Chapter 4

Generalized Moving Least Squares

As we observed in Chapter 1, the MLS method provides an approximation $\widehat{u}$ of a function u based solely on values $u(x_j)$ of u on scattered "meshless" nodes $X = \{x_1, \ldots, x_N\}$. Derivatives of u are usually approximated by derivatives of $\widehat{u}$. In contrast to this, in the present chapter we directly estimate derivatives of u from the data, without any detour via derivatives of $\widehat{u}$. This is a *generalized* Moving Least Squares (GMLS) technique, and we prove that it produces *diffuse derivatives* as introduced by Nyroles et. al. in 1992 [33]. Consequently, these turn out to be efficient direct estimates of the true derivatives, without anything "diffuse" about them, and we prove optimal rates of convergence towards the true derivatives. This chapter is completely based on paper [30].

4.1 Introduction

The MLS approximates the value $u(x)$ of an unknown function u from given data $u(x_1), \ldots, u(x_N)$ at nodes $x_1, \ldots, x_N$ near x by a value

$$\widehat{u}(x) = \sum_{j=1}^{N} a_j(x) u(x_j) \approx u(x),$$

There have been many meshless techniques based on the MLS approximation for the numerical solution of differential equations in recent years that we reviewed some of them in Chapter 3. When setting up large linear system for solving PDEs, MLS approximations are used to provide approximations to derivatives $D^\alpha u(x)$. This can be done via $D^\alpha \widehat{u}(x) \approx D^\alpha u(x)$, i.e. taking exact derivatives of the MLS solution, or via a direct estimation of $D^\alpha u(x)$ from the data $u(x_1), \ldots, u(x_N)$ near x. Here we describe the second approach and link it to the concept of *diffuse derivatives* introduced by [34]. It turns out that the second approach calculates diffuse derivatives, and therefore these are a *direct optimal estimation* from the data. We prove optimal convergence rates for the diffuse derivatives and give numerical examples. In next chapter we apply this result to make the MLPG considerably more effective. This is the main motivation behind our approach.

Section 5.2 contains a review of the generalized moving least squares (GMLS) approximation in a form similar to [19]. In section 4.3, classical and diffuse derivatives in the sense of [34] and their connections to the GMLS are described. It is proven that diffuse derivatives are GMLS approximations of true derivatives. The main contribution is in section 4.4 concerning error bounds for GMLS approximations of derivatives. Here, we follow the analysis path introduced by [42, 43] and the concept of norming sets introduced by [16], and adapt it to the approximation of derivatives. Finally, section 4.5 provides some numerical examples.

4.2 The GMLS approximation

In the classical MLS, given a set $\{u(x_1), \ldots, u(x_N)\}$ of values of an unknown function u in a domain $\Omega \subseteq \mathbb{R}^d$ at nodes $x_j \in \Omega \subseteq \mathbb{R}^d$ for $1 \leq j \leq N$, the value $u(x)$ at a fixed point $x \in \mathbb{R}^d$ is approximately recovered by minimizing a certain weighted discrete l_2 norm. But here we start with a generalized version of MLS.

Let $u \in C^m(\Omega)$ for some $m \geq 0$, and let $\{\mu_j(u)\}_{j=1}^N$ be a set of continuous linear functionals μ_j from the dual $C^m(\Omega)^*$ of $C^m(\Omega)$. For a fixed given functional $\lambda \in C^m(\Omega)^*$, our problem is the approximate recovery of the value $\lambda(u)$ from the values $\{\mu_j(u)\}_{j=1}^N$. The functionals λ and μ_j, $1 \leq j \leq N$, can, for instance, describe point evaluations of u and its derivatives up to order m. The approximation $\widehat{\lambda(u)}$ of $\lambda(u)$ should be a linear function of the data

$\mu_j(u)$, i.e. it should have the form

$$\widehat{\lambda(u)} = \sum_{j=1}^{N} a_j(\lambda)\mu_j(u), \tag{4.1}$$

and the coefficients a_j should be linear in λ. As in the classical MLS, we assume the approximation equation (4.1) to be exact for a finite dimensional subspace $\mathbb{P}_m^d = \text{span}\{p_1, p_2, \dots, p_Q\} \subset C^m(\Omega)$, i.e.

$$\sum_{j=1}^{N} a_j(\lambda)\mu_j(p) = \lambda(p) \text{ for all } p \in \mathbb{P}_m^d. \tag{4.2}$$

The GMLS approximation $\widehat{\lambda(u)}$ to $\lambda(u)$ is numerically obtained as $\widehat{\lambda(u)} = \lambda(p^*)$, where $p^* \in \mathbb{P}_m^d$ is minimizing the weighted least-squares error functional

$$\sum_{j=1}^{N} \big(\mu_j(u) - \mu_j(p)\big)^2 w_j, \tag{4.3}$$

among all $p \in \mathbb{P}_m^d$, where we use positive weights $w_1, \dots, w_N$ which later will be chosen in a specific way to localize the approximation. Of course, we then have to prove that (4.1) holds, but we shall get it only for the optimal solution.

Suppose the set point $X = \{x_1, x_2, \dots, x_N\} \subset \Omega$ and $x \in \Omega$. The classical MLS is a special case of GMLS when λ and μ_j, $1 \le j \le N$ are point evaluation functionals at x and x_j, $1 \le j \le N$ and $\mathbb{P}_m^d$ is a finite–dimensional space of polynomials, while the weights are of the form

$$w_j = w(x, x_j),\ 1 \le j \le N \tag{4.4}$$

with a nonnegative weight function w that vanishes when the arguments are at a certain distance. Furthermore, the classical MLS has an equivalent formulation, which in our generalization amounts to minimizing the quadratic form

$$\frac{1}{2}\sum_{j=1}^{N} a_j^2(\lambda)/w_j \tag{4.5}$$

as a function of the coefficients $a_j(\lambda)$ subject to the linear constraints (4.2). By some linear algebra arguments which arise already for the standard MLS and which we repeat in the next section in order to care for the dependence on the weights, the solutions p^* and $\boldsymbol{a}^*(\lambda) = (a_1^*(\lambda), \dots, a_N^*(\lambda))^T$ of the minimization problems (4.3) and (4.5), respectively, are connected by the relation

$$\widehat{\lambda(u)} = \lambda(p^*) = \sum_{j=1}^{N} a_j^*(\lambda)\mu_j(u), \tag{4.6}$$

which also proves (4.1). Formally, the solution p^* of the minimization problem (4.3) does not depend on λ. By calculating p^* from the data $\mu_j(u)$ first, one can obtain estimates of $\lambda(u)$ for all λ by just evaluating $\lambda(p^*)$. This is very useful for approximating derivatives as long as the weights are independent of λ, but needs some care because both problems depend on the weights, and the weights will be connected to the functionals in most cases. This is the main implementation recipe in the general situation. We shall be more precise in the next section.

4.3 Classical and diffuse derivatives

We take a closer look now at estimating derivative values

$$\lambda_{\alpha,x}(u) := u^\alpha(x) = \delta_x D^\alpha u \tag{4.7}$$

for fixed $x \in \Omega$ in standard multi-index notation with $|\alpha| \le m$, and where δ_x denotes the Dirac point–evaluation functional

$$\delta_x \ : \ f \mapsto f(x).$$

This situation was already mentioned as a special case in [19].

We now have to be more careful and take account of the weights. We use the weights (4.4) like in the standard MLS, even when we take more general functionals as in (4.7), but with the same x. By localization at a fixed point

x, the indices $j \in \{1, \dots, N\}$ are restricted to

$$J(x) := \{j \ : \ 1 \le j \le N, \ w(x, x_j) > 0\}$$

and we introduce a basis $p_1, \dots, p_Q$ of $\mathbb{P}^d_m$ and the notation

$$\begin{aligned}
\boldsymbol{u} :=& \big(u(x_j), \ j \in J(x)\big)^T \in \mathbb{R}^{|J(x)|} \\
P :=& \big(p_\ell(x_j)\big)_{j \in J(x), \ 1 \le \ell \le Q} \\
\boldsymbol{b} :=& (b_1, \dots, b_Q)^T \in \mathbb{R}^Q \\
W :=& \operatorname{diag}\big(w(x, x_j) : j \in J(x)\big) \\
p :=& \sum_{k=1}^{Q} b_k p_k \in \mathbb{P}^d_m
\end{aligned}$$

where almost everything depends on x. Then the problem (4.3) is

$$\text{Minimize } \|\sqrt{W}(\boldsymbol{u} - P\boldsymbol{b})\|_2^2 \tag{4.8}$$

over all $\boldsymbol{b} \in \mathbb{R}^Q$, and by classical least–squares argumentation, the solution $\boldsymbol{b}^*$ satisfies the normal equations

$$A\boldsymbol{b}^* = B\boldsymbol{u}, \tag{4.9}$$

where $A = P^T W P$ and $B = P^T W$. The matrix A is of order $Q \times Q$ and plays an important role in the MLS approximation. The solution is unique if the rank of A is Q. We assume this in what follows, i.e. we assume the data point set $X = \{x_1, \dots, x_N\}$ is $\mathbb{P}^d_m$-unisolvent.

The minimization of (4.5) can be rewritten as

$$\begin{gathered}
\text{Minimize } \frac{1}{2}\boldsymbol{a}^T W^{-1} \boldsymbol{a} \\
\text{subject to } P^T \boldsymbol{a} = \lambda(\boldsymbol{p})
\end{gathered}$$

where $\lambda(\boldsymbol{p}) := (\lambda(p_1), \dots, \lambda(p_Q))^T \in \mathbb{R}^Q$. Introducing a Lagrange multiplier $\boldsymbol{z}^*(\lambda)$, we have to construct the global minimizer $\boldsymbol{a}^*(\lambda)$ of

$$\frac{1}{2}\boldsymbol{a}^T W^{-1} \boldsymbol{a} + (\boldsymbol{z}^*(\lambda))^T (P^T \boldsymbol{a} - \lambda(\boldsymbol{p}))$$

with respect to $\boldsymbol{a}$. Then the solution $\boldsymbol{a}^*(\lambda)$ is given by the two systems

$$\begin{aligned} \boldsymbol{a}^*(\lambda) &= WP\boldsymbol{z}^*(\lambda) \\ P^T\boldsymbol{a}^*(\lambda) &= \lambda(\boldsymbol{p}) \end{aligned}$$

which implies

$$P^TWP\boldsymbol{z}^*(\lambda) = \lambda(\boldsymbol{p}),\ \boldsymbol{a}^*(\lambda) = WP(P^TWP)^{-1}\lambda(\boldsymbol{p}).$$

In some more detail,

$$a_j^*(\lambda) = w_j \sum_{k=1}^{Q} z_k^*(\lambda)\mu_j(p_k), \tag{4.10}$$

where, due to our assumption of unisolvency, the $\boldsymbol{z}_k^*(\lambda)$ are the unique solution of

$$\sum_{k=1}^{Q} z_k^*(\lambda) \sum_{j=1}^{N} w_j\mu_j(p_k)\mu_j(p_\ell) = \lambda(p_\ell), \quad 1 \le \ell \le Q. \tag{4.11}$$

If $\mu_j = \delta_{x_j}$, $1 \le j \le N$, the two solutions are connected by

$$\lambda(\boldsymbol{p})^T\boldsymbol{b}^* = \boldsymbol{u}^T\boldsymbol{a}^*(\lambda) = \sum_{j\in J} a_j^*(\lambda)u(x_j)$$

which is (4.1).

The solution $\boldsymbol{b}^*$ of the first problem is dependent on x via the weights and the index set, but, except that, **not on** λ. If λ is independent of x, we can get an approximation to $\lambda(u)$ by

$$\lambda(p^*) = \lambda(\boldsymbol{p})^T\boldsymbol{b}^* = \sum_{k=1}^{Q} b_k^*\lambda(p_k). \tag{4.12}$$

If we keep x fixed and let the multi-index α for $\lambda_{\alpha,x} = \delta_x\, D^\alpha$ vary, we have no problems and can use

$$\widehat{\lambda_{\alpha,x}(u)} = \lambda_{\alpha,x}(\boldsymbol{p})^T\boldsymbol{b}^* = \sum_{k=1}^{Q} b_k^* p_k^{(\alpha)}(x) \tag{4.13}$$

to get estimates of all derivatives of u at x after the calculation of $\boldsymbol{b}^*$, yielding

(4.6). With (4.7), we have

$$\widehat{\lambda_{\alpha,x}(u)} = \widehat{D^\alpha} u(x) = \sum_{k=1}^{Q} b_k^* p_k^{(\alpha)}(x) = \sum_{j\in J(x)} a_{j,\alpha}^*(x) u(x_j), \tag{4.14}$$

where $a_{j,\alpha}^*$ are generalized MLS shape functions that correspond to the above functionals. Note that by (4.14) we have a direct estimation of $D^\alpha u$ from the data.

The implementation of the method solves the weighted least–squares problem (4.8) first, usually by a QR decomposition of $\sqrt{W}P$, avoiding the stability problems induced by solving the normal equations (4.9). Once the solution vector $\boldsymbol{b}^*$ is known, all target functionals λ that use the same input data and weights can be estimated via (4.12). Note that this requires evaluation of λ on polynomials only, not on any shape functions. This can be used to accelerate certain meshless methods for solving PDEs, as will be demonstrated in the next chapter focusing on applications.

If $\boldsymbol{a}^*(\lambda) = WP(P^TWP)^{-1}\lambda(\boldsymbol{p})$ is requested, we decompose $\sqrt{W}P = QR$, where Q is unitary and R is upper triangular to get $P^TWP = R^TR$. By some simple calculations, $(WP)(P^TWP)^{-1}R^T = \sqrt{W}Q$. Using backward substitution, $(WP)(P^TWP)^{-1}$ is derived from this, and $\boldsymbol{a}^*(\lambda)$ can be calculated directly. This is what we need in GMLS derivatives when $\lambda = \delta_x D^\alpha$. As given at the beginning of Section 1.4, for standard derivatives of MLS shape functions more complicated calculations are required. Clearly, the direct estimation of derivatives is computationally much more efficient than calculating the derivatives of the MLS shape functions.

We now connect this to the notion of *diffuse derivatives* (see [11, 17, 34, 44]) that we explain now. If we use the standard MLS with $\lambda = \delta_x$, the vector $\boldsymbol{b}^*$ comes out the same as above, and the resulting approximation is

$$\widehat{u}(x) := \sum_{k=1}^{Q} b_k^* p_k(x),$$

but it should be kept in mind that $\boldsymbol{b}^*$ depends subtly on x via the weights and the index set $J(x)$. The derivatives of $\hat{u}$ at x, if calculations are done for varying x, will thus not be what we did above, since the dependence of $\boldsymbol{b}^*$ on x cannot be ignored. If it is ignored, the value

$$D^{\alpha}_{dif}(\widehat{u})(x) := \sum_{k=1}^{Q} b^*_k p_k^{(\alpha)}(x)$$

is called the *diffuse derivative* of $\hat{u}$ at x.

Theorem 4.1. *The GMLS approximation when applied for $\lambda = \lambda_{\alpha,x} := \delta_x D^\alpha$ and $\mu_j = \delta_{x_j}$, $1 \le j \le N$, calculates the diffuse derivative of the standard MLS. The latter is a good approximation to $D^\alpha u(x)$, but is not the same as the standard derivative $D^\alpha \widehat{u}(x)$.* $\square$

For applications in meshless methods, the estimation of $D^\alpha u(x)$ via (4.7) is all that is needed when setting up linear system of equations, since it is the best weighted moving least squares estimate based on the data $u(x_1,),\dots,u(x_N)$. It is a completely unnecessary detour to go via $\widehat{u}(x)$ and take derivatives thereof. We shall support this theoretically and practically in what follows. As we shall see, the accuracies of both schemes are nearly the same. Comparing the diffuse and full (standard) derivatives of $\widehat{u}$, the computational cost of the diffuse derivatives is considerably less. For the GMLS, we just have to calculate $\boldsymbol{b}^*$, which takes the same amount of work like in the standard MLS, and then we just need the derivatives of the polynomial basis to get (4.13). Since polynomials of degree up to m are exactly reproduced for all choices of weights, the full and diffuse derivatives of these polynomials coincide ([17]).

The use of $\widehat{u}$ and its derivatives is not necessary when setting up the linear system. After solving, we will have approximations for the values $u(x_j)$ at the nodes. Then, for *postprocessing*, it may be necessary to calculate exact derivatives of the approximate solution $\widehat{u}$, e.g. for calculation of stress in elasticity problems. At this time, it is up to the user whether exact or diffuse derivatives are calculated. If users want to have a single solution function $\widehat{u}$ with exact derivatives, they will have to pay a price. If they can admit small errors, they can get away with diffuse derivatives. For postprocessing, the use of diffuse derivatives makes a lot of sense in certain situations, but not for setting up linear systems.

Since the word "diffuse" may mislead readers to assume that these derivatives are not first–choice, we ignore this term from now on and use $\widehat{D^\alpha u}(x)$ or $\widehat{\lambda(u)}$ instead, to let the notation indicate that we have a direct and usually very good numerical approximation to $D^\alpha u(x)$ or $\lambda(u)$, respectively. For future work, we suggest to drop the term *diffuse derivative* in favor of *GMLS derivative approximation.* There is nothing *diffuse* or *uncertain* about it.

4.4 Error bounds

To be more precise with the generation of weights, we choose a continuous function $\Phi : [0, \infty) \to \mathbb{R}$ that is positive on $[0, 1/2]$ and supported in $[0, 1]$, and define

$$w(x, y) = \Phi \left(\frac{\|x - y\|_2}{\delta} \right),$$

for $\delta > 0$ as a weight function. Then we define $J(x) = \{j \in \{1, 2, \ldots, N\} : \|x - x_j\|_2 \leq \delta\}$. At first, the convergence rate of a *generalized local polynomial reproduction* system will be presented and then we will show that the generalized MLS of the first section is a local polynomial reproduction in the following sense.

Definition 4.2. Consider a process that defines for every $\mathbb{P}^d_m$–unisolvent set $X = \{x_1, x_2, \ldots, x_N\} \subset \Omega$ and each multi-index α with $|\alpha| \leq m$ a family of functions $s_{j,\alpha} : \Omega \to \mathbb{R}$, $1 \leq j \leq N$ to approximate

$$D^\alpha u(x) \approx \sum_{j=1}^N s_{j,\alpha}(x) u(x_j)$$

for functions $u \in C^m(\Omega)$. Then we say that the process *provides a local polynomial reproduction of degree m on* Ω if there exist constants $h_0, C_{1,\alpha}, C_{2,\alpha} > 0$ such that

1. $\sum_{j=1}^N s_{j,\alpha}(x) p(x_j) = D^\alpha p(x)$, for all $p \in \mathbb{P}^d_m$, $x \in \Omega$,
2. $\sum_{j=1}^N |s_{j,\alpha}(x)| \leq C_{1,\alpha} h_{X,\Omega}^{-|\alpha|}, \quad \forall x \in \Omega$,
3. $s_{j,\alpha}(x) = 0 \; if \; \|x - x_j\|_2 > C_{2,\alpha} h_{X,\Omega}$,

is satisfied for all $|\alpha| \leq m$ and all X with $h_{X,\Omega} \leq h_0$.

This definition is a generalized form of Definition 1.5 of Chapter 1. (see [43, chap. 3]). We avoided the notation $s_j^{(\alpha)}(x)$ since it is not true that $s_{j,\alpha} = D^\alpha s_{j,0}$, as suggested by item 1 above.

Theorem 4.3. *Suppose that $\Omega \subset \mathbb{R}^d$ is bounded. Define Ω^* to be the closure of $\bigcup_{x\in\Omega} B(x, C_2 h_0)$. Define*

$$\widehat{D^\alpha u}(x) := \sum_{j=1}^{N} s_{j,\alpha}(x) u(x_j),$$

where $\{s_{j,\alpha}\}$ is a local polynomial reproduction of order m on Ω for $|\alpha| \le m$. Then there exists a constant $C > 0$ such that for all $u \in C^{m+1}(\Omega^*)$ and all X with $h_{X,\Omega} \le h_0$ there is an error bound

$$|D^\alpha u(x) - \widehat{D^\alpha u}(x)| \le C h_{X,\Omega}^{m+1-|\alpha|} |u|_{C^{m+1}(\Omega^*)}. \tag{4.15}$$

Proof. Let $p \in \mathbb{P}_m^d$ be an arbitrary polynomial. Using the properties of local polynomial reproduction in Definition 4.2 yields

$$\begin{aligned}
\left|D^\alpha u(x) - \widehat{D^\alpha u}(x)\right| &\le |D^\alpha u(x) - D^\alpha p(x)| + \left|D^\alpha p(x) - \sum_{j=1}^{N} s_{j,\alpha}(x) u(x_j)\right| \\
&\le |D^\alpha u(x) - D^\alpha p(x)| + \sum_{j=1}^{N} |s_{j,\alpha}(x)|\,|p(x_j) - u(x_j)| \\
&\le \|D^\alpha u - D^\alpha p\|_{L_\infty(\mathcal{D})} + \|u - p\|_{L_\infty(\mathcal{D})} \sum_{j=1}^{N} |s_{j,\alpha}(x)| \\
&\le \|D^\alpha u - D^\alpha p\|_{L_\infty(\mathcal{D})} + C_{1,\alpha} h_{X,\Omega}^{-|\alpha|} \|u - p\|_{L_\infty(\mathcal{D})}
\end{aligned} \tag{4.16}$$

where $\mathcal{D} = B(x, C_{2,\alpha} h_{X,\Omega})$. Now choose p to be the Taylor polynomial of u around x. This gives for each $|\beta| = m+1$ and $y \in \Omega$ a $\xi(y,\beta) \in \Omega^*$ such that

$$\begin{aligned}
u(y) &= \sum_{|\beta|\le m} \frac{D^\beta u(x)}{\beta!}(y-x)^\beta + \sum_{|\beta|=m+1} \frac{D^\beta u(\xi(y,\beta))}{\beta!}(y-x)^\beta \\
&= p(y) + \sum_{|\beta|=m+1} \frac{D^\beta u(\xi(y,\beta))}{\beta!}(y-x)^\beta.
\end{aligned} \tag{4.17}$$

where

$$p(y) = \sum_{|\beta|\le m} \frac{D^\beta u(x)}{\beta!}(y-x)^\beta. \tag{4.18}$$

Hence

$$\begin{aligned}\|u-p\|_{L_\infty(\mathcal{D})} \le& (C_{2,\alpha}h_{X,\Omega})^{m+1}\sum_{|\beta|=m+1}\frac{1}{\beta!}\|D^\beta u\|_{L_\infty(\Omega^*)}\\ \le& Ch_{X,\Omega}^{m+1}|u|_{C^{m+1}(\Omega^*)}.\end{aligned} \tag{4.19}$$

Moreover, since $D^\alpha u \in C^{m+1-|\alpha|}(\Omega^*)$ the Taylor expansion of order $m-|\alpha|$ for $D^\alpha u$ around $x\in\Omega$ exists. This gives for each $|\beta| = m+1-|\alpha|$ and every $y\in\Omega$ a $\zeta(y,\beta)\in\Omega^*$ such that

$$D^\alpha u(y) = \sum_{|\beta|\le m-|\alpha|}\frac{D^{\beta+\alpha}u(x)}{\beta!}(y-x)^\beta + \sum_{|\beta|=m+1-|\alpha|}\frac{D^{\beta+\alpha}u(\zeta(y,\beta))}{\beta!}(y-x)^\beta. \tag{4.20}$$

The first part of the right hand side of equation (4.20) is clearly $D^\alpha p(y)$ with $p(y)$ defined in equation (4.18). Therefore

$$\left\|D^\alpha u - D^\alpha p\right\|_{L_\infty(\mathcal{D})} \le Ch_{X,\Omega}^{m+1-|\alpha|}|u|_{C^{m+1}(\Omega^*)}. \tag{4.21}$$

Combining (4.19) and (4.21) with (4.16) leads to (4.15). □

Now it suffices to show that the family of functions $\{a^*_{j,\alpha}\}$ in (4.14) forms a local polynomial reproduction in sense of Definition 4.2. It can be done by the concept of *norming sets*, introduced by Jetter *et.al* [16]. Before that we need some definitions.

Let V be a finite-dimensional vector space with norm $\|\cdot\|_V$ and let $\Lambda\subseteq V^*$ be a finite set consisting of N functionals. Here, V^* denotes the dual space of V consisting of all linear and continuous functionals defined on V.

Definition 4.4. Λ is a *norming set* for V if the mapping $T: V\to T(V)\subseteq\mathbb{R}^N$ defined by $T(v) = (\lambda(v))_{\lambda\in\Lambda}$ is injective. The operator T is called the *sampling operator.*

Theorem 4.5. *([16], [24] and [43]) Suppose V is a finite-dimensional normed linear space and $\Lambda = \{\mu_1,\dots,\mu_N\}$ is a norming set for V, T being the corresponding sampling operator. For every $\lambda\in V^*$ there exists a vector*

$s \in \mathbb{R}^N$ depending only on λ such that, for every $v \in V$,

$$\lambda(v) = \sum_{j=1}^{N} s_j \mu_j(v),$$

and

$$\|s\|_{\mathbb{R}^{N*}} \leq \|\lambda\|_{V^*} \|T^{-1}\|.$$

Theorem 3.8 of [43] proves:

Theorem 4.6. *If $\Omega \subset \mathbb{R}^d$ is compact and satisfies an interior cone condition with radius r and angle $\theta \in (0, \pi/2)$, for fixed number m if the set X satisfies*

$$h_{X,\Omega} \leq \frac{r \sin\theta}{4(1+\sin\theta)m^2}, \tag{4.22}$$

then $\Lambda = \{\delta_{x_1}, \dots, \delta_{x_N}\}$ is a norming set for $\mathbb{P}_m^d(\Omega)$ and $\|T^{-1}\| \leq 2$.

Also it is easy to show that

Proposition 4.7. *$\Lambda = \{\delta_{x_1}, \dots, \delta_{x_N}\}$ forms a norming set for $\mathbb{P}_m^d(\Omega)$ if and only if $X \subset \Omega$ is $\mathbb{P}_m^d$-unisolvent.*

One the other side, from Proposition 2.2 of [32] and 11.6 of [43], we have:

Proposition 4.8. *Suppose that $\Omega \subset \mathbb{R}^d$ is bounded and satisfies an interior cone condition with radius r and angle θ. If $p \in \mathbb{P}_m^d$ and $|\alpha| \leq m$ then*

$$\|D^\alpha p\|_{L_\infty(\Omega)} \leq \left(\frac{2m^2}{r\sin\theta}\right)^{|\alpha|} \|p\|_{L_\infty(\Omega)}. \tag{4.23}$$

If $V = \mathbb{P}_m^d(\Omega)$ and $\lambda = \delta_x D^\alpha$, in the situation of Theorem 4.6, using (4.23) and (4.22) it is easy to show

$$\|\lambda\|_{V^*} \leq \left(\frac{2m^2}{r\sin\theta}\right)^{|\alpha|} \leq \big(2(1+\sin\theta)\big)^{-|\alpha|} h_{X,\Omega}^{-|\alpha|}.$$

Consequently we can state:

Corollary 4.9. *Let $\Omega \subset \mathbb{R}^d$ be bounded and satisfy an interior cone condition with radius r and angle θ. If $X = \{x_1, \dots, x_N\} \subset \Omega$ and (4.22) is satisfied, then there exist for every $x \in \Omega$ real numbers $s_{j,\alpha}(x)$ such that*

$$\sum_{j=1}^{N} |s_{j,\alpha}(x)| \leq 2\big(2(1+\sin\theta)\big)^{-|\alpha|} h_{X,\Omega}^{-|\alpha|},$$

and

$$\sum_{j=1}^{N} s_{j,\alpha}(x)p(x_j) = D^\alpha p(x)$$

for all $p \in \mathbb{P}_m^d$.

Now we should convert this global existence result to the local situation. It can be done using the fact that for every point $x \in \Omega$ we can find a cone $C(x)$ that is completely contained in Ω and since every cone itself satisfies a cone condition, we can apply Corollary 4.9 to the cone $C(x)$ and $Y = X \cap C(x)$. Therefore, as in Theorem 3.14 of [43], we can prove:

Theorem 4.10. *If* $\Omega \subset \mathbb{R}^d$ *is compact and satisfies an interior cone condition with radius* r *and angle* $\theta \in (0, \pi/2)$, *for fixed* $m \in \mathbb{N}$ *there exist constants*

$$C_{1,\alpha} = 2\big(2(1+\sin\theta)\big)^{-|\alpha|}, \quad C_{2,\alpha} = \frac{16(1+\sin\theta)^2 m^2}{3\sin^2\theta}, \quad h_0 = \frac{1}{C_2}$$

such that for every $X \subset \Omega$ *with* $h_{X,\Omega} \le h_0$ *and every* $x \in \Omega$ *we can find real numbers* $s_{j,\alpha}(x)$, $1 \le j \le N$ *such that they form a local polynomial reproduction as in Definition 4.2.*

Now using the minimal property of $a^*_{j,\alpha}$ in (4.14), we can show these functions form a local polynomial reproduction. This comes in the following theorem. The proof is same as the proof of Theorem 4.7 of [43].

Theorem 4.11. *Suppose that* $\Omega \subset \mathbb{R}^d$ *is compact and satisfies an interior cone condition with radius* r *and angle* $\theta \in (0, \pi/2)$. *Fix* $m \in \mathbb{N}$. *Let* h_0, $C_{1,\alpha}$, *and* $C_{2,\alpha}$ *denote the above-mentioned constants. Suppose that* X *satisfies (**??**) and* $h_{X,\Omega} \le h_0$. *Let* $\delta = 2C_{2,\alpha}h_{X,\Omega}$. *Then the basis functions* $a^*_{j,\alpha}$ *from (4.14) provide a local polynomial reproduction as in Definition 4.2 with constant* $\widetilde{C}_{1,\alpha}$ *and* $\widetilde{C}_{2,\alpha}$ *that can be derived explicitly.*

Finally using Theorems 4.3 and 4.11 we conclude the following corollary that includes the order of convergence of the MLS approximation and its diffuse derivatives.

Corollary 4.12. *In the situation of Theorem 4.11, define* Ω^* *to be the closure of* $\bigcup_{x\in\Omega} B(x, C_{2,\alpha}h_0)$. *Define*

$$\widehat{D^\alpha u}(x) := \sum_{j=1}^{N} a^*_{j,\alpha}(x)u(x_j)$$

where $a^*_{j,\alpha}(x)$ are functions derived from the MLS approximation in (4.14). Then there exists a constant $c > 0$ such that for all $u \in C^{m+1}(\Omega^*)$ and all $X \subset \Omega$ with $h_{X,\Omega} \le h_0$ which are quasi–uniform in the sense of (??) with the same constant c_{qu} we have

$$\|D^\alpha u - \widehat{D^\alpha u}\|_{L_\infty(\Omega)} \le c h_{X,\Omega}^{m+1-|\alpha|} |u|_{C^{m+1}(\Omega^*)}. \qquad (4.24)$$

The error estimates of MLS approximation and its full derivatives are given in [2] and [46] using different strategies. They have proved that the error of full derivatives of order $|\alpha|$ is of order $O(h^{m+1-|\alpha|})$ where h plays the same role as $h_{X,\Omega}$. Thus, direct estimation of derivatives from function values is recommendable instead of taking full or diffuse derivatives of the classical MLS solution $\widehat{u}$.

4.5 Numerical examples

To confirm the above theoretical bounds, we look at MLS approximation for *Franke's function* (1.18). First, regular node distributions with distance h along each axis are used. A compactly supported and C^4 RBF

$$\Phi(x - x_j) = (1 - r)^6_+(35r^2 + 18r + 3), \quad r = \|x - x_j\|_2,$$

is used as weight function, and the shifted scaled polynomials (see Section 1.4) are employed as basis functions.

Table 4.1 presents the ratios of errors for the function and its first and second standard and GMLS derivatives in a fixed and sufficiently fine test point mesh of size 31×31 on $[0, 1]^2$. "Order0", "Order1" and "Order2" refer to the computational orders of error of the function, its first derivative with respect to x_1 and its second derivative with respect to x_1, respectively. The distance h is divided by 2 row by row, so the orders are computed by formula (2.22). We consider both standard and GMLS derivatives. The results are presented for $m = 1, 2, 3$ and $\delta = 1.5mh$. According to theoretical bounds, the oeders should be approximately $m + 1 - |\alpha|$. As we can see, the numerical

Table 4.1: The orders of errors of Franke's function and its first and second standard and GMLS derivatives

			Oeder1		Order2	
	h	Order0	standard	GMLS	standard	GMLS
$m = 1$						
	0.1	—	—	—	—	—
	$\frac{0.1}{2}$	1.82	1.39	1.38	—	—
	$\frac{0.1}{4}$	1.94	0.99	0.98	—	—
	$\frac{0.1}{8}$	1.99	0.98	0.98	—	—
	$\frac{0.1}{16}$	2.00	0.95	0.95	—	—
$m = 2$						
	0.1	—	—	—	—	—
	$\frac{0.1}{2}$	3.06	1.37	1.37	1.33	1.34
	$\frac{0.1}{4}$	3.72	1.80	1.79	1.50	1.48
	$\frac{0.1}{8}$	3.93	1.95	1.94	1.22	1.21
	$\frac{0.1}{16}$	3.86	1.98	1.98	1.13	1.12
$m = 3$						
	0.1	—	—	—	—	—
	$\frac{0.1}{2}$	2.23	1.99	2.05	0.88	0.84
	$\frac{0.1}{4}$	3.36	3.30	3.30	1.59	1.55
	$\frac{0.1}{8}$	3.82	3.80	3.81	1.88	1.87
	$\frac{0.1}{16}$	3.95	3.95	3.95	1.97	1.97

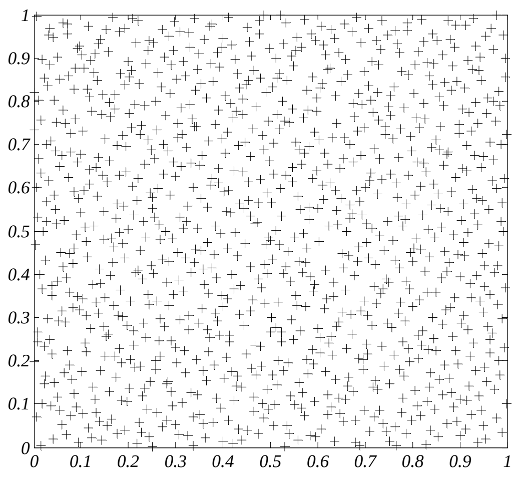


Figure 4.1: First 1000 Halton points

results confirm the analytical bounds. Also it is evident that there is no significant difference between the rates of convergence of standard and GMLS derivatives. Note that with $m = 1$ we can not recover the second derivatives. In this example, for instance, the CPU time needed to compute the second GMLS derivative with $h = 0.1/16$ ($N = 25921$) and $m = 2$ in the above test point mesh is 2.60 sec, while it is 3.35 sec. for the standard derivative.

Now, we choose the set of centers to be *Halton points* in $[0, 1]^2$. We use the following commands in MATLAB to generate such sets:

```
p = haltonset(2,'Skip',1e3,'Leap',1e2);
N = 1000;           % number of selected centers
X = net(p,N);
```

The first 1000 Halton points are depicted in FIG. 4.1. Following [42], it is in general too expensive to compute $h_{X,\Omega}$ exactly. Therefore we used the approximation $h_{X,\Omega} \approx h = 1/\sqrt{N}$ together with $\delta = 1.5mh$. The maximum errors and ratios, which are provided in a regular mesh 31×31 in $[0, 1]^2$, are presented in Tables 4.2 and 4.3 for the first and the second derivatives with respect to first variable, respectively, for $m = 3$. The approximate fill distance h is divided by 2 consecutively and the ratios are computed by (2.22). One can see that the theoretical bounds are obtained and the results are nearly the same for both standard and GMLS derivatives. The CPU time required

Table 4.2: Maximum and ratios of errors of the first standard and GMLS derivatives of Franke's function at Halton points with $m = 3$

		Standard		GMLS	
N	h	$\|e\|_\infty$	ratio	$\|e\|_\infty$	ratio
1000	0.03162	7.89×10^{-2}	–	7.76×10^{-2}	–
4000	0.01581	1.08×10^{-2}	2.87	1.08×10^{-2}	2.84
16000	0.00791	2.00×10^{-3}	2.43	1.99×10^{-3}	2.45
64000	0.00395	1.08×10^{-4}	4.22	1.08×10^{-4}	4.20

Table 4.3: Maximum and orders of errors of the second standard and GMLS derivatives of Franke's function at Halton points with $m = 3$

		Standard		GMLS	
N	h	$\|e\|_\infty$	ratio	$\|e\|_\infty$	ratio
1000	0.03162	8.00×10^{0}	–	7.77×10^{0}	–
4000	0.01581	2.16×10^{0}	1.87	2.17×10^{0}	1.84
16000	0.00791	5.76×10^{-1}	1.93	5.74×10^{-1}	1.92
64000	0.00395	1.40×10^{-1}	2.04	1.41×10^{-1}	2.03

to execute the GMLS subroutine for computing the second derivative with $N = 16000$ in the above-mentioned test point mesh is 2.03 sec, while it is 3.40 sec for the standard MLS derivative subroutine.

4.6 Conclusion

This chapter implies that "diffuse" derivatives used within certain applications of the MLS can be stably implemented and induce no loss in accuracy, because they are identical to direct optimal estimates of derivatives provided by the generalized moving least squares (GMLS). In particular, the orders of convergence of both derivatives to the exact values turn out to be the same, and the computational efficiency of GMLS derivatives is better.

Chapter 5

Direct Meshless Local Petrov-Galerkin (DMLPG) Method

In MLPG and other MLS based methods, the stiffness matrix is provided by integrating over MLS shape functions or their derivatives. These shape functions are complicated and have no closed forms. To get accurate results, numerical quadrature with many integration points is required. Thus the MLS subroutines must be called very often, leading to high computational costs. In contrast to this, the stiffness matrix in finite element methods (FEMs) is constructed by integrating over *polynomial* basis functions which are much cheaper to evaluate. This relaxes the cost of numerical integrations somewhat. For an account of the importance of numerical integration within meshless methods, we refer the reader to [8].

In this chapter, using the GMLS approximation of previous chapter, a new *direct* MLPG technique is presented. As pointed, GMLS recovers local test functionals directly from values at nodes, without any detour via shape functions. This technique avoids integration over MLS shape functions in MLPG and replaces it by the much cheaper integration over polynomials. It ignores shape functions completely. Altogether, the method is simpler, faster and more accurate than the original MLPG method. GMLS directly approximates boundary conditions and local weak forms, shifting the numerical integration into the MLS itself, rather than into an outside loop over calls to MLS routines. We call this approach *Direct Meshless Local Petrov-Galerkin* (DMLPG) method. The convergence rate of MLPG and DMLPG seems to be the same, but thanks to the simplified computation, the results of DMLPG often are more precise than the results of MLPG. Numerical examples illustrate the superiority of the new technique over the classical MLPG.

This chapter is completely based on submitted manuscript [29] that borrows the idea of GMLS approximation of paper [30].

5.1 An overview

If Section 3.1 we presented an overview on meshless methods that parameterize the solution u of a PDE entirely in terms of nodal values

$$\boldsymbol{u} = (u(x_1), \dots, u(x_N))^T$$

if we assume the discretized problem consists of linear equations

$$\lambda_k(u) = \beta_k,\ 1 \leq k \leq M,$$

where $M \geq N$. Also, in Section 3.2 we described a methodology to discretize a PDE problem in the above form via the concept of *local weak forms* for constructing variations of MLPG methods along with equation (3.2). But, looking more generally, the upshot of all meshless methods is to provide good estimates $\widehat{\lambda}_k$ of all λ_k using only values at nodes. This means that one has to only find real numbers $a_j(\lambda_k)$ with

$$\widehat{\lambda_k(u)} = \sum_{j=1}^{M} a_j(\lambda_k) u(x_j) \approx \lambda_k(u) \text{ for all } k,\ 1 \leq k \leq M. \tag{5.1}$$

Putting the $a_j(\lambda_k)$ into an $M \times N$ matrix K, one has to solve the possibly overdetermined linear system

$$K\,\boldsymbol{u} = \boldsymbol{b} \tag{5.2}$$

with $\boldsymbol{b} = (\beta_1, \dots, \beta_M)^T$. Comparing (5.1) with (3.2), here we do not mention *shape functions* at all. They are not needed to set up a linear system in terms of values at nodes. The goal just is to find good estimates for the target functionals λ_k in terms of the values at nodes, e.g. via (5.1), to set up the matrix K. Note that in some cases, e.g. when the functionals λ_k are derivatives at points, this is just a variation of a finite–difference approach.

5.2 GMLS Approximation

As given in Chapter 4, the GMLS approximation recovers a general linear functional $\lambda \in C^{m+1}(\Omega)^*$, $m \geq 0$ from nodal values $\mu_1(u), \dots, \mu_N(u)$ where $u \in C^{m+1}(\Omega)$. Even if a different numerical method is used to minimize (4.5) or (4.3), the optimal solution $\boldsymbol{a}^*(\lambda) \in \mathbb{R}^N$ can be written as

$$\boldsymbol{a}^*(\lambda) = WP^T(P\,W\,P^T)^{-1}\lambda(\boldsymbol{p}) \tag{5.3}$$

where W is the diagonal matrix carrying the weights w_j on its diagonal, P is the $N \times Q$ matrix of values $\mu_j(p_k)$, $1 \leq j \leq N$, $1 \leq k \leq Q$, and $\lambda(\boldsymbol{p}) \in \mathbb{R}^Q$ is the vector with values $\lambda(p_1), \dots, \lambda(p_Q)$ of λ on the basis of $\mathbb{P}^d_m$. Thus it suffices to evaluate λ on the space $\mathbb{P}^d_m$, not on a certain trial space spanned by certain shape functions. This will significantly speed up numerical calculations, if the functional λ is complicated, e.g. a numerical integration against a test function. Standard examples are functionals of the form

$$\lambda(u) = \int_D v(x)\, Lu(x)dx$$

where L is a linear differential operator preserving polynomials or just the identity, v is some weight function and D is a measurable domain. Such functionals will arise for PDE problems in weak form in the next section. Then our GMLS technique will perform integration only over polynomials. Note that this generalizes to any type of functional: we finally just have to evaluate it on a polynomial. No other calls to MLS routines are necessary, because we do not apply the functional to shape functions.

5.3 Implementation of DMLPG

In this section, we describe the implementation of GMLS approximations to solve the Poisson problem (3.4) using the weak form equations (3.7).

At first we fix m, the maximal degree of polynomial basis functions we use. If the problem has enough smoothness, m will determine the convergence rate. Then we choose a set $X = \{x_1, x_2, \dots, x_N\} \subset \Omega$ of scattered trial points which is filling the domain reasonably well, without letting two points come extraordinarily close to each other. In this sense, we require the set X of trial nodes to be quasi–uniform.

We now have to define the functionals $\lambda_1, \dots, \lambda_M$ discretizing our PDE problem. This requires a selection between MLPG1, MLPG2, and MLPG5, and the decision to use oversampling or not, i.e. $M > N$ or $M = N$. Oversampling will often increase stability at increased cost, but we found that in our examples it was not necessary. Since we have to execute the GMLS method for each functional λ_k, approximating it in terms of function values at the trial nodes in $B(y_k, \delta) \cap X$, we have to make sure that the GMLS does not break down. This means that the matrix P of (5.3) must have rank Q, if formed for the nodes in $B(y_k, \delta) \cap X$.

Theorem 5.1. *([40], see also [43]) For any compact domain Ω in $\mathbb{R}^d$ with an interior cone condition, and any $m \geq 0$ there are positive constants h_0 and c_0 such that for all trial node sets X with fill distance $h_{X,\Omega} \leq h_0$, all test points $y \in \Omega$, and all radii $\delta \geq c_0\, h_{X,\Omega}$, the set $B(y, \delta) \cap X$ is $\mathbb{P}_m^d$–unisolvent.*

This means that the placement of test nodes and the choice of weight function supports can be linked to the fill distance of the set of trial nodes. Oversampling never causes problems, if the weight function support radius is kept proportional to the fill distance of the trial nodes.

Some test nodes should be scattered over the Dirichlet boundary Γ_D to impose the Dirichlet boundary conditions. We denote the subset of these points by $Y_D \subset Y \cap \Gamma_D$. For MLPG2, we similarly define Y_N, and then the setup of the functionals simply follows (3.5), with or without oversampling. In principle, the sets Y_Ω, Y_N, Y_D need not be disjoint.

For weak problems in MLPG1 or MLPG5 form, we just implement the functionals λ_k of (3.7) as described in Section 3.2. Altogether, we follow Section 5.1 by implementing (3.1) via (5.1), and ending with the system (5.2).

The order of convergence of the approximated functional to its exact value is important in this case. Applying the same strategy presented in Section 4.4 for $\lambda_k(u) := D^\alpha u(y_k)$, we can prove

Theorem 5.2. *Let*

$$\lambda_k(u) := \int_D v(x)\, Lu(x)dx, \quad D = \Omega_\sigma^k \text{ or } \partial\Omega_\sigma^k,\ y_k \in \Omega_\sigma^k.$$

Define

$$\widehat{\lambda_k(u)} := \sum_{j=1}^{N} a_j^*(\lambda_k)u(x_j),$$

where $a_j^*(\lambda_k)$ are functions derived from the GMLS approximation in (5.3). Then there exists a constant $c > 0$ such that for all $u \in C^{m+1}(\Omega^*)$ and all quasi-uniform $X \subset \Omega$ with $h_{X,\Omega} \leq h_0$ we have

$$\left|\lambda_k(u) - \widehat{\lambda_k(u)}\right| \leq c h_{X,\Omega}^{m+1-n} |u|_{C^{m+1}(\Omega^*)}, \tag{5.4}$$

providing $\int_D |v(x)|dx < \infty$ and if $\lambda(u) \neq 0$, $\int_D v(x)\, Lx^\alpha dx \neq 0$ ($\lambda_k(x^\alpha) \neq 0$) for some α with $|\alpha| = m$. Here n is the maximal order of derivatives involved in linear operator L.

However, we cannot guarantee that the system (5.2) has full rank, since we only made sure that the rows of the system can be calculated via the GMLS if Theorem 5.1 applies. Oversampling will usually help if the system causes problems.

After the solution vector $\boldsymbol{u}$ of (5.2), consisting of values $u(x_j)$ of values at the trial nodes is determined by solving the system, other values of the solution function $u(x)$ and also its derivatives can be calculated in every point $x \in \Omega$ again using the GMLS approximation by taking $\lambda(u) = D^\alpha u(x)$.

Since we have direct approximations for boundary conditions and local weak forms, this method is called *direct meshless local Petrov-Galerkin (DMLPG) method.* It comes in the DMLPG1, DMLPG2, and DMLPG5 variations.

In contrast to MLPG2, if the GMLS derivatives ("diffuse" derivatives) are used instead of the standard derivatives of MLS shape functions, we have DMLPG2. As investigated in Chapter 4, the accuracies for calculating the matrix K of (5.2) are the same, but the computational cost of DMLPG2 is less. When looking into the literature, we found that DMLPG2 coincides with the *Diffuse Approximation Method* (DAM) [37]. But since we avoid using the word *diffuse* because there is nothing "diffuse" about these derivatives, we will call the method DMLPG2 or *direct MLS collocation (DMLSC) method.*

As we saw in Section 5.2, in DMLPG1/5 methods the integrations are done only over polynomials, if the latter are used in the GMLS. For every functional λ_k, $1 \leq k \leq M \geq N$, the GMLS routine is called only once. There are no calls to produce values of shape functions. The standard MLPG/MLS technique at degree m implements numerical integration by calling shape function evaluations, and thus the MLS routine is called approximatively $M\,Q_N$ times where Q_N is the average number of integration points. Moreover, in standard MLPG methods the derivatives of MLS shape function must also be provided, while DMLPG has no shape functions at all. Consequently, DMLPG is considerably faster than MLPG. In addition, due to the error analysis presented in Theorem 5.2 for the new GMLS method, the final accuracies of both MLPG and DMLPG methods are the same. We will see experimentally that DMLPG is even more accurate than MLPG.

As highlighted in [8], numerical integration in FEM is simple because the integrands of the elements of the stiffness matrix are polynomials. In contrast to this, the shape functions used in standard meshless methods are much more costly to evaluate, making numerical integration a much bigger challenge than for the FEM. In MLPG methods, numerical integrations are simpler than for various other meshless methods, since the local weak form breaks everything down to local well–shaped subdomains. However, since the integrands are based on MLS shape functions and their derivatives, a Gauss quadrature with many points is required to get accurate results, especially when the density of nodes increases. Overcoming this drawback is a major advantage of DMLPG methods, because the integrations are done over polynomials, like in FEM.

It is interesting to note that if local sub-domains are chosen in DMLPG5 as $S(x, \sigma)$ (square or cube), a $(d-1)$–times $\lceil \frac{m}{2} \rceil$–points Gauss quadrature gives the exact solution for local boundary integrals around the nodes in the interior of Ω. In DMLPG1, if again $S(x,\sigma)$ is chosen as a local sub-domain and if a polynomial test function is employed, a d-times $\left\lceil\frac{(m-1)(n-1)+1}{2}\right\rceil$–points Gauss quadrature is enough to get exact interior local domain integrals. Here, n is the degree of the polynomial test function. As a polynomial test function on the square or cube for DMLPG1 with $n=2$, we can use

$$v(x;x_k)=\begin{cases}\displaystyle\prod_{i=1}^{d}\left(1-\frac{4}{\sigma^2}(\mathsf{x}_i-\mathsf{x}_{ki})^2\right), & x\in S(x_k,\sigma),\\ 0, & \text{otherwise}\end{cases}$$

where $x=(\mathsf{x}_1,\ldots,\mathsf{x}_d)$ and $x_k=(\mathsf{x}_{k1},\ldots,\mathsf{x}_{kd})$. In DMLPG1 with balls as sub-domains, weight functions of the form function

$$v(x,y)=\Phi\left(\frac{\|x-y\|_2}{\sigma}\right), \tag{5.5}$$

can be used as test functions. Both of these test functions vanish on $\Gamma_\sigma^k\backslash\Gamma_N$ if $x=y_k$, as required in DMLPG1.

Note that, if the second local weak formulation (for example the local weak forms (3.10) for a 2D equation) is used the process gives the DMLPG4 rather than MLPG4 or the meshless LBIE method presented in [45]. In DMLPG4, it is better to use balls as local sub-domains, because in this case the modified fundamental solution, used as a test function, can be derived easily. But the test function is not a polynomial.

Both DMLPG3 and DMLPG6 can be formulated as well using our approach, but they require more ingredients, so we leave them out here.

Instead, we add some remarks about selecting m, the degree of polynomial basis functions in the GMLS. For $m=1$, the variants DMLPG 1, 4, and 5 will necessarily fail. The background is that the GMLS performs an optimal recovery of a functional λ in terms of nodal values, and the recovery is exact on a subspace $\mathbb{P}_m^d$, using minimal coefficients at the nodal values. Thus, in all cases where the functional is zero on $\mathbb{P}_m^d$ by some reason or other, the recovery formula will be zero and will generate a zero row in the stiffness matrix. This happens for all variations based on functionals (3.7) and functionals extracted from the second weak form on interior points, since all those functionals are reformulations of

$$\lambda(u) = \int_{B(y_k,\sigma)} v\, \Delta u \, d\omega$$

and thus vanish on harmonic functions u, in particular on linear functions. Thus, for solving inhomogeneous problems, users should pick spaces $\mathbb{P}_m^d$ of non–harmonic functions, if they employ GMLS with exactness on $\mathbb{P}_m^d$. This rules out polynomials with degree $m \leq 1$.

Another closely related point arises from symmetry of subdomains. Since polynomials in a ball $B(y,\sigma)$ or a cube $S(y,\sigma)$ have symmetry properties, the entries of stiffness matrices in rows corresponding to internal points will often be the same for $m = 2k$ and $m = 2k+1$. Thus convergence rates often do not increase when going from $m = 2$ to $m = 3$, for instance. But this observation affects MLPG and DMLPG in the same way.

5.4 Stability and Convergence

For the classical MLS and the generalized MLS from Chapter 4and Theorem 5.2 it is known that the recovery $\widehat{\lambda(u^*)}$ of values of functionals λ on a true solution u^* has an error of order $\mathcal{O}(h^{m+1-\ell})$, if h is the fill distance of the trial nodes, m is the degree of polynomials used locally, if the exact solution u^* is at least C^{m+1}, ℓ is the maximal order of derivatives of u^* involved in the functional, and if numerical integration has an even smaller error. In particular, the classical MLS and the new GMLS produce roughly the same stiffness matrices, but the GMLS has a considerably smaller computational complexity.

However, the error committed in the approximation of the test functionals in terms of function values at nodes does not always carry over to the convergence rate of the full algorithm, since there is no stability analysis, so far. Even if perfect stability would hold, the best one can expect is to get the convergence rate implied by the local trial approximation, i.e. by local polynomials of degree m. This would again mean a rate of $\mathcal{O}(h^{m+1-\ell})$, but only if the solution is indeed locally approximated by polynomials of that degree. In fact, the next section will show that this rate can often be observed. But our symmetry arguments at the end of the previous section show that sometimes the degree $m = 2k + 1$ cannot improve the behavior for $m = 2k$, because the odd–degree polynomials simply do not show up in most of the calculations for the stiffness matrix.

5.5 Numerical results

In this section some numerical results are presented to demonstrate the efficiency of DMLPG methods and its advantages over MLPG methods. We consider the Poisson equation (3.4) on the square $[0, 1]^2 \subset \mathbb{R}^2$ with Dirichlet boundary conditions. Since we want to study convergence rates without being limited by smnoothness of the solution, we take *Franke's function* (1.18) as analytical solution and calculate the right hand side and boundary conditions accordingly. Regular mesh distributions with mesh-size h are provided in all cases, though the methods would work with scattered data. We do not implement oversampling in the results of this paper. In fact, the trial and test points are chosen to be coincident. Also, the *shifted scaled polynomial* basis (1.17) where x is a fixed evaluation point such as a test point or a Gaussian point for integration is used for both MLS and GMLS approximations. We let $m = 2, 3$ and 4. The Gaussian weight function (1.13) is employed where $c = c_0 h$ is a constant controlling the shape of the weight function and $\delta = \delta_0 h$ is the size of the support domains.

Let $m = 2$ and set $c_0 = 0.6$ and $\delta_0 = 2m$. At first the local sub-domains are taken to be circles. To get the best results in MLPG we have to use an accurate quadrature formula. Here a 20-points regular Gauss-Legendre quadrature is employed for numerical integrations over local sub-domains.

Numerical results, for different mesh-sizes h, are presented in terms of maximum errors, orders and CPU times used for MLPG5 and DMLPG5 in Table 5.1. The mesh-size h is divided by two row by row, therefore the ratios are computed by (2.22). Both methods have nearly the same order 2, which cannot be improved for this trial space, since the expected optimal order is $m + 1 - \ell = 3 - 1 = 2$. But significant differences occur in the

Table 5.1: The maximum errors, orders and CPU times used for MLPG5 and DMLPG5 with $m = 2$

	MLPG5		DMLPG5		CPU time	
h	$\|e\|_\infty$	Orders	$\|e\|_\infty$	Orders	MLPG5	DMLPG5
0.2	0.44×10^{-1}	–	0.23×10^{-1}	–	1.4 sec.	0.2 sec.
0.1	0.15×10^{-1}	1.59	0.72×10^{-2}	1.68	9.0	0.5
0.05	0.73×10^{-2}	0.99	0.20×10^{-2}	1.84	45.0	2.2
0.025	0.24×10^{-2}	1.61	0.58×10^{-3}	1.80	215.2	9.4
0.0125	0.66×10^{-3}	1.85	0.14×10^{-3}	1.98	2456.9	59.6

columns with CPU times. As we stated before, this is due to restricting local integrations to polynomial basis functions in DMLPG rather than to integrate over MLS shape functions in the original MLPG. We could get the same results with fewer integration points for DMLPG, but to be fair in comparison, we use the same quadrature. In addition, to give more insight into the errors, the maximum errors of MLPG5 and DMLPG5 are illustrated in Figure 5.1. Once can see that DMLPG is more accurate, maybe due to avoiding many computations and hence many roundoff errors.

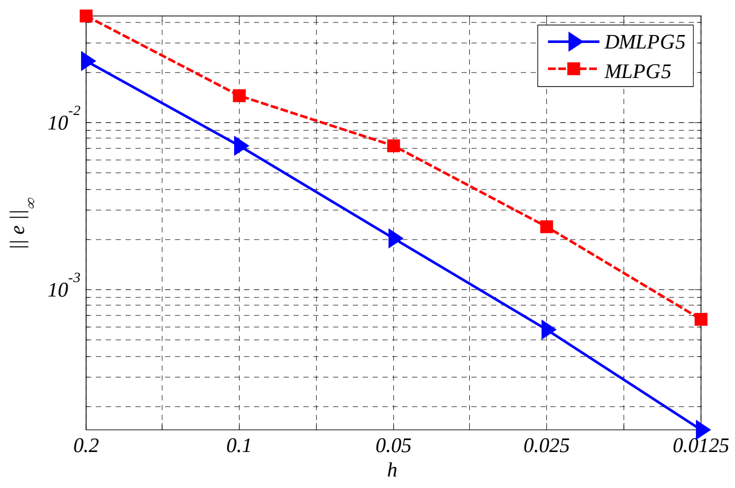


Figure 5.1: Comparison of MLPG5 and DMLPG5 in terms of maximum errors for $m = 2$.

Table 5.2: The maximum errors, orders and CPU times used for MLPG5 and DMLPG5 with $m = 3$

	MLPG5		DMLPG5		CPU time	
h	$\|e\|_\infty$	Orders	$\|e\|_\infty$	Orders	MLPG5	DMLPG5
0.2	0.28×10^{-1}	–	0.23×10^{-1}	–	2.0 sec.	0.2 sec.
0.1	0.13×10^{-1}	1.08	0.74×10^{-2}	1.62	18.2	0.8
0.05	0.33×10^{-2}	1.98	0.20×10^{-2}	1.89	103.2	3.4
0.025	0.78×10^{-3}	2.09	0.58×10^{-3}	1.80	493.9	15.0
0.0125	0.19×10^{-3}	2.06	0.15×10^{-3}	1.98	3830.0	82.5

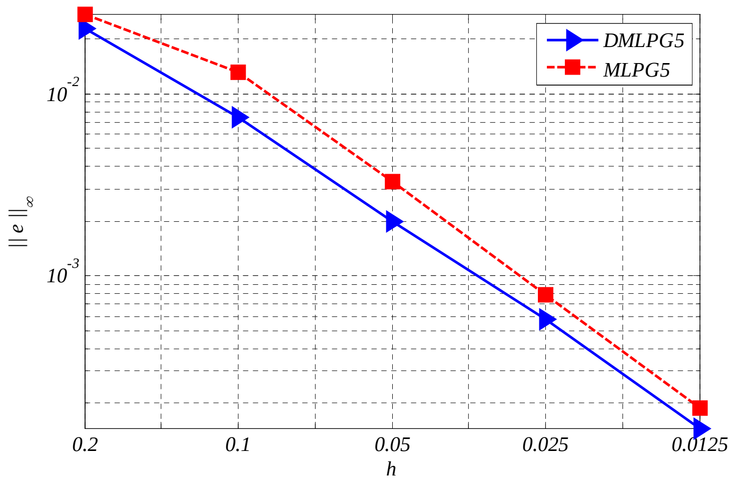


Figure 5.2: Comparison of MLPG5 and DMLPG5 in terms of maximum errors for $m = 3$.

In Table 5.2 and Figure 5.2, we have compared MLPG5 and DMLPG5 in case $m = 3$. The convergence rate stays at 2 for both methods, since the third–degree polynomials cannot contribute to the weak form. The figure shows approximately the same error results. But the CPU times used are indeed different.

In the results presented up to here, we used balls (circles) as local subdomains. Now we turn to use squares for both MLPG5 and DMLPG5. Also, we run the programs with $m = 4$ to see the differences. The parameters $c_0 = 0.8$ and $\delta_0 = 2m$ are selected. In DMLPG5, a 2-points Gaussian

Table 5.3: The maximum errors, ratios and CPU times used for MLPG5 and DMLPG5 with $m = 4$

	MLPG5		DMLPG5		CPU time	
h	$\|e\|_\infty$	Orders	$\|e\|_\infty$	Orders	MLPG5	DMLPG5
0.2	0.10×10^{0}	–	0.12×10^{0}	–	2.2 sec.	0.3 sec.
0.1	0.25×10^{-1}	2.04	0.17×10^{-1}	2.87	28.4	0.9
0.05	0.78×10^{-2}	1.66	0.12×10^{-2}	3.75	189.6	4.2
0.025	0.79×10^{-3}	3.30	0.75×10^{-4}	4.04	1451.0	19.3
0.0125	0.55×10^{-4}	3.86	0.43×10^{-5}	4.12	8021.5	107.6

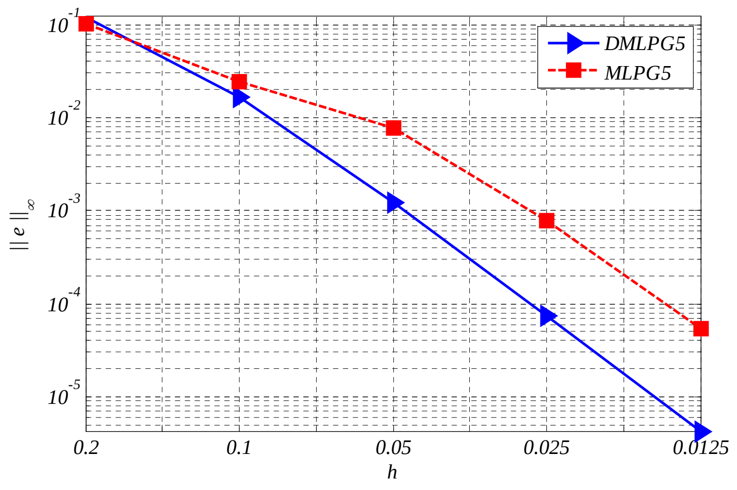


Figure 5.3: Comparison of MLPG5 and DMLPG5 in terms of maximum errors for $m = 4$.

quadrature is enough to get exact numerical integrations. But for MLPG5 and the right hand sides we use a 10-points Gaussian quadrature for every sides of squares. The results are depicted in Table 5.3 and Figure 5.3. DMLPG is more accurate and approximately gives the full order 4 in this case. Beside, as we expected, the computational cost of DMLPG is remarkably less than MLPG.

Results for MLPG1 and DMLPG1 turn out to behave similarly. As we know, MLPG1 is more expensive than MLPG5 [4, 5], but there is no significant difference between computational costs of DMLPG5 and DMLPG1.

Therefore the differences between CPU times used for MLPG1 and DMLPG1 are absolutely larger.

5.6 Conclusion

In the present chapter we describe a new MLPG method, called DMLPG method, based on GMLS approximation for solving boundary value problems. The new method is considerably faster than the classical MLPG variants, because

- direct approximations of data functionals are used for Dirichlet boundary conditions and local weak forms,
- local integrations are done over polynomials rather than over complicated MLS shape functions,
- numerical integrations can sometimes be performed exactly.

The convergence rate of both methods should be the same, but thanks to avoiding many computations and roundoff errors, and of course by treating the numerical integrations in a more elegant and possibly exact way, the results of DMLPG turn often out to be more accurate than the results of MLPG.

On the downside, DMLPG does not work for $m = 1$ since it locally uses (harmonic) linear functions instead of complicated shape functions. But most MLPG users choose higher degrees anyway, in order to obtain better convergence rates.

Altogether, we believe that the DMLPG methods have great potential to replace the original MLPG methods in many situations.

References

[1] M.G. Armentano. Error estimates in sobolev spaces for moving least square approximations. *SIAM Journal on Numerical Analysis*, 39(1):38–51, 2001.

[2] M.G. Armentano and R.G. Durán. Error stimates for moving least square approximations. *Applied Numerical Mathematics*, 37:397–416, 2001.

[3] K. E. Atkinson. *The numerical solution of integral equations of the second kind.* Cambridge University Press, New York, 1997.

[4] S. N. Atluri. *The meshless method (MLPG) for domain and BIE discretizations.* Tech Science Press, Encino, CA, 2005.

[5] S. N. Atluri and S. Shen. *The Meshless Local Petrov-Galerkin (MLPG) Method.* Tech Science Press, Encino, CA, 2002.

[6] S. N. Atluri and S. Shen. The meshless local Petrov-Galerkin (MLPG) method: A simple and less costly alternative to the finite element and boundary element methods. *CMES: Computer Modeling in Engineering and Sciences*, 3 (1):11–52, 2002.

[7] S.N. Atluri and T.-L. Zhu. A new meshless local Petrov-Galerkin (MLPG) approach in Computational mechanics. *Computational Mechanics*, 22:117–127, 1998.

[8] I. Babuska, U. Banerjee, J.E. Osborn, and Q. Zhang. Effect of numerical integration on meshless methods. *Computer Methods in Appleid Mechanics and Engineering*, 198:27–40, 2009.

[9] T. Belytschko, Y. Krongauz, D.J. Organ, M. Fleming, and P. Krysl. Meshless methods: an overview and recent developments. *Computer Methods in Applied Mechanics and Engineering, special issue*, 139:3–47, 1996.

[10] T. Belytschko, Y.Y. Lu, and L. Gu. Element-Free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[11] P. Breitkopf, A. Rassineux, G. Touzot, and P. Villon. Explicit form and efficient computation of MLS shape functions and their derivatives. *International Journal for Numerical Methods in Engineering*, 48:451–466, 2000.

[12] H. Brunner. *Collocation Methods for Volterra Integral and Related Functional Differential Equations.* Cambridge University Press, 2004.

[13] C.A. Duarte and J.T. Oden. H-p clouds-an hp meshless method. *Numerical Methods for Partial Differential Equations*, 12:673–705, 1996.

[14] R. Franke. Scattered data interpolation: tests of some methods. *AMS Mathematics of Computation*, 48:181–200, 1982.

[15] Th. Hangelbroek. On local RBF approximation. to appear in Adv. in Comp. Math., arXiv:0909.5244, 2011.

[16] K. Jetter, J. Stöckler, and J.D. Ward. Error estimates for scattered data interpolation on spheres. *AMS Mathematics of Computation*, 68:733–747, 1999.

[17] D. W. Kim and Y. Kim. Point collocation methods using the fast moving least-square reproducing kernel approximation. *International Journal for Numerical Methods in Engineering*, 56:1445–1464, 2003.

[18] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *AMS Mathematics of Computation*, 37:141–158, 1981.

[19] D. Levin. The approximation power of moving least-squares. *AMS Mathematics of Computation*, 67:1517–1531, 1998.

[20] D. Levin. Stable integration rules with scattered integration points. *Journal of Computational and Applied Mathematics*, 112:181–187, 1999.

[21] D.H. McLain. Drawing contours from arbitrary data points. *Comput. J.*, 17:318–324, 1974.

[22] D.H. McLain. Two dimensional interpolation from random data. *Comput. J.*, 19:178–181, 1976.

[23] J. M. Melenk. On approximation in meshless methods. In A. W. Craig J. F. Blowey, editor, *Frontiers of Numerical Analysis*, pages 65–141, Durham, 2005. Springer.

[24] H.N. Mhaskar, F.G. Narcowich, and J.D. Ward. Spherical Marcinkiewicz-Zygmund inequalities and positive quadrature. *AMS Mathematics of Computation*, 70:1113–1130, 2001.

[25] D. Mirzaei and M. Dehghan. Implementation of meshless LBIE method to the 2D non-linear SG problem. *International Journal for Numerical Methods in Engineering*, 79:1669–1682, 2009.

[26] D. Mirzaei and M. Dehghan. A meshless based method for solution of integral equations. *Applied Numerical Mathematics*, 60:245–262, 2010.

[27] D. Mirzaei and M. Dehghan. MLPG approximation to the p-Laplace problem. *Computational Mechanics*, 46(6):805–812, 2010.

[28] D. Mirzaei and M. Dehghan. MLPG method for transient heat conduction problem with MLS as trial approximation in both time and space domains. *CMES: Computer Modeling in Engineering and Sciences*, 72(3):185–210, 2011.

[29] D. Mirzaei and R. Schaback. Direct meshless local Petrov-Galerkin (DMLPG) method: A generalized MLS approximation. Submitted. 2011.

[30] D. Mirzaei, R. Schaback, and M. Dehghan. On generalized moving least squares and diffuse derivatives. *IMA Journal of Numerical Analysis*, 2011. doi: 10.1098/imanum/drr030.

[31] Y.X. Mukherjee and S. Mukherjee. The boundary node method for potential problems. *International Journal for Numerical Methods in Engineering*, 40:797–815, 1997.

[32] F.J. Narcowich, J.D. Ward, and H. Wendland. Sobolev bounds on functions with scattered zeros, with application to radial basis function surface fitting. *AMS Mathematics of Computation*, 47(250):743–763, 2004.

[33] B. Nyroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992.

[34] B. Nyroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992.

[35] E. Onate, S. Idelsohn, O.C. Zienkiewicz, and R.L. Taylor. A finite point method in computational mechanics. Applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39:3839–3866, 1996.

[36] F. Paris and J. Canas. *Boundary Element Method: Fundamental and Applications.* Oxford University Press, Oxford, 1997.

[37] C. Prax, H. Sadat, and P. Salagnac. Diffuse approximation method for solving natural convection in porous media. *Transport in Porous Media*, 22:215–223, 1996.

[38] R. Schaback. Why does MLPG work? Accepted paper for a keynote lecture in ICCES 07, 2006.

[39] R. Schaback. Unsymmetric meshless methods for operator equations. *Numerische Mathematik*, 114:629–651, 2010.

[40] R. Schaback. Kernel-based meshless methods. Lecture Note, Göttingen, 2011.

[41] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 23th National Conference ACM*, pages 517–523, 1968.

[42] H. Wendland. Local polynomial reproduction and moving least squares approximation. *IMA Journal of Numerical Analysis*, 21:285–300, 2001.

[43] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005.

[44] Y.C. Yoon, S.H. Lee, and T. Belytschko. Enriched meshfree collocation method with diffuse derivatives for elastic fracture. *Computers and Mathematics with Applications*, 51:1349–1366, 2006.

[45] T. Zhu, J.D. Zhang, and S.N. Atluri. A local boundary integral equation (LBIE) method in computational mechanics, and a meshless discretization approach. *Computational Mechanics*, 21:223–235, 1998.

[46] C. Zuppa. Error estimates for moving least square approximations. *Bulletin of the Brazilian Mathematical Society*, 34(2):231–249, 2003.