KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computational science

Second cycle, 30 credits

RBF method for solving Navier-Stokes equations

VOLODYMYR YELNYK

Stockholm, Sweden, 2023

RBF method for solving Navier-Stokes equations

VOLODYMYR YELNYK

Date: September 19, 2023

Supervisors: Jennifer Ryan, Davoud Mirzaei
Examiner: Gerald Q. Maguire Jr.
School of Electrical Engineering and Computer Science
Swedish title: Detta är den svenska översättningen av titeln
Swedish subtitle: Detta är den svenska översättningen av undertiteln

© 2023 Volodymyr Yelnyk

Abstract

This thesis explores the application of Radial Basis Functions (RBFs) to fluid dynamical problems. In particular, stationary Stokes and Navier-Stokes equations are solved using RBF collocation method. An existing approach from the literature, is enhanced by an additional polynomial basis and a new preconditioner. A faster method based on the partition of unity is introduced for stationary Stokes equations. Finally, a global method based on Picard linearization is introduced for stationary Navier-Stokes equations.

Keywords

Radial Basis Functions, Navier-Stokes equations, Meshless methods

Sammanfattning

Denna avhandling utforskar tillämpningen av Radial Basis Functions (RBF) på dynamiska problem med vätskor. I synnerhet löses stationära Stokes och Navier-Stokes ekvationer lösas med hjälp av RBF-samlokaliseringsmetoden. En befintlig metod från litteraturen, förbättras genom en ytterligare polynombas och en ny förkonditionering. En snabbare metod baserad på enhetens partition introduceras för stationära Stokes-ekvationer. Slutligen introduceras en global metod baserad på Picard linjärisering för stationära Navier-Stokes ekvationer.

Nyckelord

Radiala Basfunktioner, Navier-Stokes ekvationer, Meshlösa metoder

Acknowledgments

I would like to thank Dr. Davoud Mirzaei. This project was conceived by and would not be possible without him. Large parts of this work are directly based on his own research. I would also like to thank Dr. Jennifer Ryan for excellent literature recommendations and academic guidance.

Stockholm, September 2023
Volodymyr Yelnyk

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rdial Basis functions (RBFs) were first introduced as a method for scattered data interpolation. Such data often comes from real world measurements such as sensor networks or a point cloud coming from LIDAR measurements for example[1]. The general idea is to center some radially symmetric functions at the given data points. Each radial function is multiplied by a scaling parameter, chosen such that the sum of these functions interpolates the data. Figure 1.1 shows Gaussian radial functions centered at 3 data points.

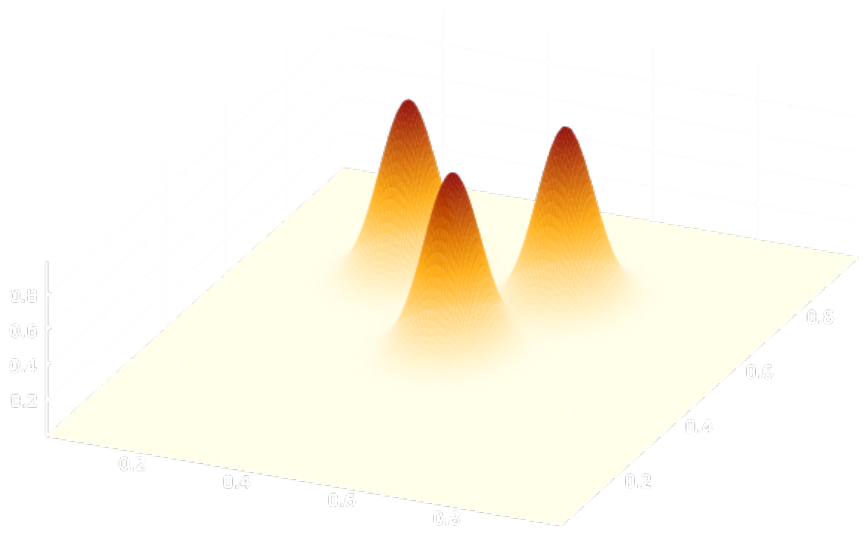

Figure 1.1: 2D shifted radial functions

RBFs have become very popular in the machine learning community in

the context of data classification [2]. The data can be separated using level sets of the RBF interpolant. The latter idea is called the kernel trick [3]

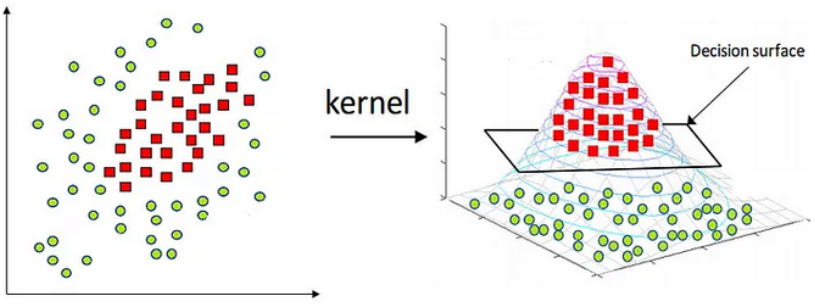

Figure 1.2: Kernel trick, picture by Grace Zhang

It is possible to solve differential equations by interpolating a derivative of the function at some sampled points and then recovering the original function . Such method was first proposed by J. Kansa [4]. Unlike traditional methods for PDEs such as Finite difference (FD), Finite Volume (FM) or Finite Element (FEM), the Kansa method does not require any mesh. The domain and the boundary of the problem are instead purely defined by a set of scattered nodes. This simplifies the implementation of the method and makes it easily extendable to high dimensions.

Since it's introduction in the early 90s, many RBF-based methods have been conceived and applied to various PDEs. Some of these methods are: RBF-finite difference[5], Partition of unity[6] and Local hermitian interpolation[7] to name a few. In recent years there have been an increasing interest in applying RBF based methods for solving Navier-Stokes(NS) equations. Some success have been made by solving the NS equations using vorticity-stream function formulation, or discretizing each component of the velocity field separately. Such methods are only viable in 2 dimensions, or lead to the saddle point problem.

The aim of this thesis is to take advantage of recent developments in the field of RBF interpolation and present a computationally flexible method with desirable numerical properties. In particular, this work is heavily based on work of F. Narcowich and J. Ward [8], who generalized the theory of scalar valued RBFs to matrix-valued RBFs. Using this theory it is possible to construct analytically divergence-free interpolants to a given vector field. When solving Stokes, or NS equations this property allows to ignore the mass-

continuity equation, as it is automatically satisfied. To increase computational efficiency, 2 local algorithms are considered: Local hermitian interpolation [9] and partition of unity algorithm based on [6]. These methods are tested on stationary Stokes equations. Lastly, a global method is developed for nonlinear stationary NS equations.

Chapter 2

Radial Basis Functions

In this chapter Radial Basis Function interpolation is motivated and introduced. Some relevant facts about RBFs are cited from the literature. Then, the applications to PDEs are discussed.

2.1 Why not just use polynomials

Given a set of N data points in 1D such as in figure 2.1, it is easy to find a polynomial of degree N-1 that exactly interpolates the data. It is known that in this case the interpolation matrix is always non-singular, as long as there are no duplicates in the data. The situation is different in 2D. For example, suppose there are 5 randomly positioned data points. It is not obvious which basis to choose: should it be $\{1, x, y, xy, x^2\}$ or $\{1, x, y, xy, y^2\}$? Even if the number of data points matches the dimension of the basis, the solution is not guaranteed to exist. This problem occurs not only for polynomial interpolation, but also for any basis functions that are independent of the data. This is captured in Mairhuber-Curtis theorem. Roughly, it states that the interpolation problem in 2 and more dimensions is not guaranteed to be well-posed if the basis does not depend on the data. precise definition and a short proof can be found in[10].

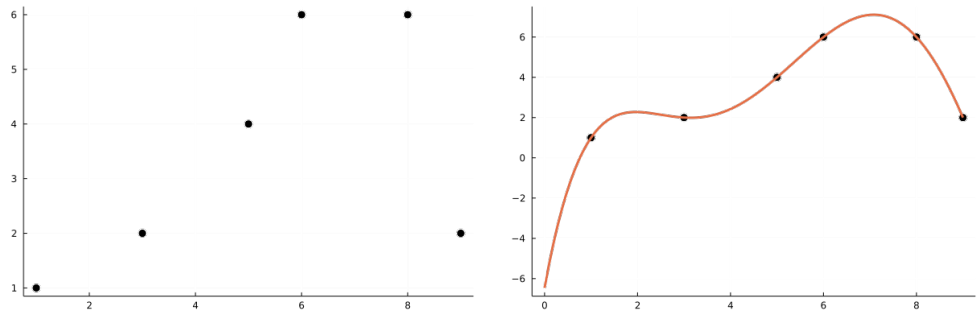

Figure 2.1: Polynomial interpolation in 1D

2.2 Positive definite Radial basis functions

Given a set of distinct datapoints, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2 ... \boldsymbol{x}_N : \boldsymbol{x}_i \in \mathbb{R}^n\}$ in and a set of function values sampled at those points, $f|_X = \{f_1, f_2, ... f_N : f_i \in \mathbb{R}\}$ the original function f is approximated as a weighted sum of radial functions centered at the data points.

$$f(\boldsymbol{x}) \approx s(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi(\boldsymbol{x} - \boldsymbol{x}_i). \tag{2.1}$$

Here, ϕ is a radial function, meaning that it is only a function of the 2-norm of the argument $\phi(\cdot) = \phi(||\cdot||)$, c_i are the weights which are computed such that the following interpolation condition is satisfied:

$$s(\boldsymbol{x}_i) = f_i \quad \text{for } i = 1, 2, ...N. \tag{2.2}$$

Writing out the Equation 2.2 in full for all i yeilds:

$$\begin{bmatrix} \phi(\boldsymbol{x}_1 - \boldsymbol{x}_1) & \phi(\boldsymbol{x}_1 - \boldsymbol{x}_2) & \dots & \phi(\boldsymbol{x}_1 - \boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2 - \boldsymbol{x}_1) & \phi(\boldsymbol{x}_2 - \boldsymbol{x}_2) & \dots & \phi(\boldsymbol{x}_2 - \boldsymbol{x}_N) \\ \vdots & \ddots & \dots & \vdots \\ \phi(\boldsymbol{x}_N - \boldsymbol{x}_1) & \phi(\boldsymbol{x}_N - \boldsymbol{x}_2) & \dots & \phi(\boldsymbol{x}_N - \boldsymbol{x}_N) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}. \tag{2.3}$$

Equation 2.3 is usually abbreviated as $A\boldsymbol{c} = \boldsymbol{f}$. Where A is called the interpolation matrix. Matrix A is symmetric, which follows from the radial property of the basis functions ϕ. For a well-posed interpolation problem, it is necessary for the interpolation matrix to be invertible. This requirement leads to the definition of positive definiteness [11]:

$e^{-\varepsilon^2\|\boldsymbol{x}\|_2^2}$	Gaussian
$(1+\|x\|_2^2)^{-\beta}$	Generalized inverse multiqadratics
$(1-\|\boldsymbol{x}\|_2)_+^{\ell}$	Wendland's compactly supported power functions
$\frac{1}{\sqrt{1+(\epsilon\|\boldsymbol{x}\|_2)^2}}$	Inverse Multicuadratic
$\frac{1}{1+(\epsilon\|\boldsymbol{x}\|_2)^2}$	Inverse Quadratic

Table 2.1: Popular Radial basis functions

Definition 1. *A continuous function $\phi : \mathbb{R}^d \to \mathbb{C}$ is called positive semi-definite if, for all $N \in \mathbb{N}$, all sets of pairwise distinct centers $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^d$, and all $\alpha \in \mathbb{C}^N$, the quadratic form*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} \alpha_j \overline{\alpha_k} \phi\left(\boldsymbol{x}_j - \boldsymbol{x}_k\right)$$

is non-negative. The function Φ is called positive definite if the quadratic form is positive for all $\alpha \in \mathbb{C}^N \backslash \{0\}$.

This definition is for general smooth functions, not only radial ones, which will be important for solving differential equations. Some authors refer to positive semi-definite functions as positive definite and use the term strictly positive definite for positive definite functions. In this thesis, the vocabulary of Definition 1 is used. From the above definition, it is clear that positive definite functions generate positive definite interpolation matrices, which admit an inverse. Over the years mathematicians have discovered numerous positive definite RBFs and techniques to study them. Some of the most popular ones are listed in Table 2.1:

It is worth noting that some RBFs also contain additional argument, ϵ . This argument is referred to as a shape parameter, and it influences the accuracy of the interpolation. Other RBFs, such as Wendland's functions, have integer-valued parameters (for example β or l). These parameters control the smoothness of the RBFs. Many more examples positive definite RBFs can be found in [10] and [11].

2.3 conditionally positive definite kernels and polyharmonics

A powerful feature of RBF interpolants is the ability to exactly reproduce polynomials of given degree. That is: given any polynomial of degree at most m, $f \in \mathbb{P}_m(\mathbb{R}^n)$, and a set of distinct data points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_N : \boldsymbol{x}_i \in \mathbb{R}^n\}$, then the following holds:

$$s(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) \quad \forall \boldsymbol{x}_i \in X \quad => \quad f = s. \tag{2.4}$$

To achieve this property the interpolant in Equation 2.1 needs some modification. More specifically, an additional polynomial basis is added :

$$s(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi(\boldsymbol{x} - \boldsymbol{x}_i) + \sum_{j=1}^{Q} d_j p_j(\boldsymbol{x}), \tag{2.5}$$

where p_j are basis functions of $\mathbb{P}_m(\mathbb{R}^n)$, Q is the total number of such basis polynomials. The easiest basis to implement is monomials. For example, if m = 2, $n = 2$ and $\boldsymbol{x} = [x_1, x_2]$, the interpolant will take the following form:

$$s(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi(\boldsymbol{x} - \boldsymbol{x}_i) + d_1 + d_2 x_1 + d_3 x_2 + d_4 x_1 x_2 + d_5 x_1^2 + d_6 x_2^2 \tag{2.6}$$

In this case, Q= 6. In general $Q = \frac{(m+n)!}{n!m!}$. Now there are more unknowns than equations because there are still N equations coming from the interpolation condition, but $N + Q$ unknowns. To fix this, a set of vanishing moment conditions are added:

$$\sum_{i=1}^{N} p_j(\boldsymbol{x}_i) = 0 \quad \forall\, j = 1, 2..Q. \tag{2.7}$$

to find coefficients c and d the following linear system has to be solved:

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{2.8}$$

Where A is the interpolation matrix as in Equation 2.3, and $P_{ij} = p_j(\boldsymbol{x}_i)$. In order for the solution to exist, the matrix P has to have a full column rank. This introduces some new restrictions for the data sites, which are not present

in the classical interpolation. Suppose the P matrix has full column rank, then the only d such that $Pd = 0$ is 0, therefore the only polynomial of degree at most m that interpolates 0 functions on the dataset is the trivial 0 polynomial. Datasets that satisfy such conditions are called m-unisolvent. An important question is: how likely one encounters degenerate datasets that are not m-unisolvent? First, the number of points has to be sufficiently large. It is easy to see that a 4th-degree polynomial can interpolate 3 points in more than one way. If there are more than Q points, they still can form a degenerate set, but it is very unlikely in practice. Of course, there also exists a sufficient condition for an m-unisolvent dataset, see for example [8].

Theorem 1. *Given m-unisolvent dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_N : \boldsymbol{x}_i \in \mathbb{R}^n\}$ and an interpolant of the form 2.5 with polynomial basis of degree m, Linear system in Equation 2.8 is invertible.*

Proof.

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Rightarrow \tag{2.9}$$

$$c^T Ac + c^T Pd = 0$$
$$P^T c = 0$$

substituting the bottom equation in the top one yields:

$$c^T Ac = 0.$$

Since A is positive definite, $c = 0$. Hence Ac is also zero. From the first equation, it follows that $Pd = 0$. P has full column rank, so $d = 0$, which proves the theorem. □

The key takeaway from the above discussion is that any positive definite RBF interpolant can be appended with polynomials to achieve better approximation. Some RBFs are not positive definite, but still generate invertible systems if the polynomial basis is appended to the interpolant. Such RBFs are called conditionally positive definite. Below is the formal definition from [11]:

Definition 2. *A continuous function $\phi : \mathbb{R}^n \to \mathbb{C}$ is said to be conditionally positive semidefinite of order m if, for all $N \in \mathbb{N}$, all pairwise distinct centers*

$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^n$, and all $\alpha \in \mathbb{C}^N$ satisfying

$$\sum_{j=1}^{N} \alpha_j p\left(\boldsymbol{x}_j\right) = 0$$

for all complex-valued polynomials of degree less than m, the quadratic form

$$\sum_{j,k=1}^{N} \alpha_j \overline{\alpha_k} \phi\left(\boldsymbol{x}_j - \boldsymbol{x}_k\right)$$

is nonnegative. ϕ is said to be conditionally positive definite of order m if the quadratic form is positive, unless α is zero.

Additionally, there is a theorem proving well-poseness of the interpolant of the form 2.5 formed by conditionally positive definite kernel:

Theorem 2. *If the real-valued even function ϕ is conditionally positive definite of order m on $\mathbb{R}^s$ and the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ form an $(m-1)$-unisolvent set, then the system*

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{2.10}$$

is uniquely solvable.

The proof is identical to Theorem 1. From the definition, one can see that a conditionally positive definite function of order m is also a conditionally positive definite of any order larger than m. Here are a few examples of such RBFs:

$\phi(x) = (1 + \|x\|^2)^\beta \;\; m = \max(0, \lceil \beta \rceil)$	Generalized multi quadratics
$\phi(x) = \begin{cases} \|x\|^{2\beta} \log(\|x\|), & m = \beta + 1 \\ \|x\|^\beta, & m = \lceil \beta/2 \rceil \end{cases}$	Polyharmonics

Table 2.2: Examples of conditionally positive definite RBFs

To give an example of why such functions can't be used without a polynomial basis, consider polyharmonic RBF of even degree: $\|x\|^{2\beta} \log(\|x\|)$ and a set of points placed at the nodes of an equal-sided simplex in $\mathbb{R}^n$ with a side length of one. The interpolation matrix in this case is the zero matrix, which is singular.

2.4 Error bounds and stability

Error analysis of RBF interpolation is an active area of research. In this section, only a few relevant results are summarized. As usual, for more detailed discussion refer to [11] and [10]. Intuitively, larger density of points should lead to better RBF interpolation. This density is formaly measured as fill distance.

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \,. \tag{2.11}$$

It can be interpreted as the radius of the largest ball that doesn't touch or contain any datapoints.

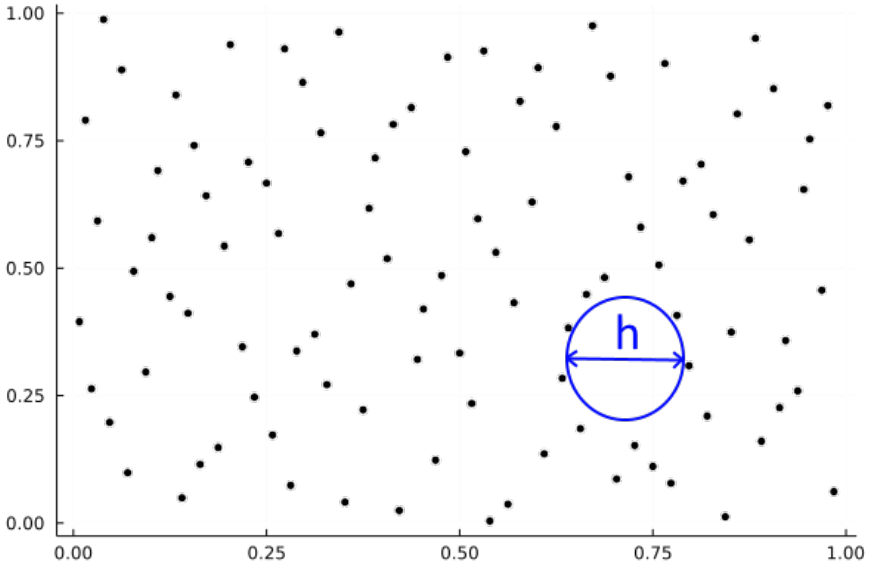

Figure 2.2: Illustration of the fill distance

Let f be the target function and s be an RBF interpolant. for RBFs with smoothness C^{2k}, the error can be expressed as follows:

$$|D^\alpha f(x) - D^\alpha \mathcal{P}_f(x)| \leq C h_{\mathcal{X},\Omega}^{k-|\alpha|}, \tag{2.12}$$

for some multi index α with $k \geq |\alpha|$, some constant C, which depends on the function itself and sufficiently small $h_\mathcal{X}$. On the other hand, if there is a duplicate in the data, the condition matrix is singular. It then follows that decreasing the minimum distance between datapoints increases the condition number of the interpolation matrix. This minimum distance is formally

defined as a separation distance:

$$q = q_{\mathcal{X}} = \frac{1}{2} \min_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 . \tag{2.13}$$

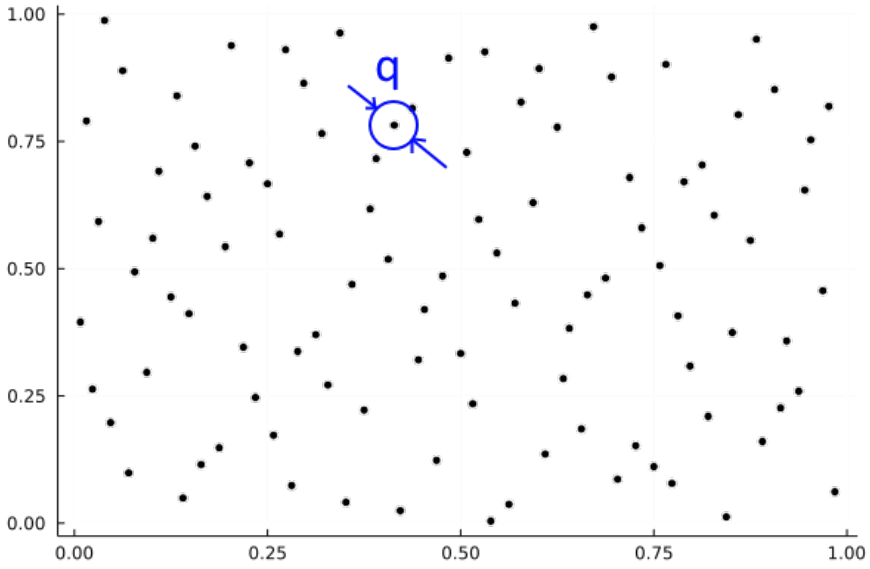

Figure 2.3: Illustration of the separation distance

The condition number and the separation distance are related in the following way: There exists a bound on the smallest eigenvalue of the interpolation matrix, which in turn bounds the condition number.

$$\text{cond}(A) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}. \tag{2.14}$$

For example, the following bound exists for a polyharmonic kernel:

$$\begin{cases} \lambda_{\text{min}} \geq C q_{\mathcal{X}}^{2\beta} & \phi = \|x\|^{2\beta} \log(\|x\|) \\ \lambda_{\text{min}} \geq C q_{\mathcal{X}}^{\beta} & \phi = \|x\|^{\beta} \end{cases} \tag{2.15}$$

The constant C depends on β. The main takeaway is that the overall error depends on the interpolation error related to fill distance, and numerical error related to separation distance. For the lowest overall error, the fill distance has to be as low as possible, while the separation distance has to be as large as possible. These 2 requirements contradict each other, the best one can do

is to distribute the points as uniformly as possible. The problem gets worse with lower floating point precision. Most radial basis functions lead to very ill-conditioned interpolation matrices, even with low density of points. It is then often required to use quadruple precision arithmetic to achive good results. This shortcoming of RBF interpolation is addressed in Chapter 4 with a suitable preconditioner.

2.5 Solving Differential equations

The RBF method was first used to solve partial differential equations in [4] and it is remarkably simple. Consider a differential equation

$$\begin{cases} Lu = f, & \text{on}\Omega \\ Bu = g, & \text{on}\Gamma \end{cases}, \tag{2.16}$$

where L is a linear differential operator, and B is the boundary operator, Ω is the domain and Γ is the boundary of the domain. To illustrate how RBFs can be used,let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_N \ : \boldsymbol{x}_i \in \mathbb{R}^n\}$ be a set of distinct domain points and $Y = \{\boldsymbol{x}_{N+1}, ...\boldsymbol{x}_{N+M} : \boldsymbol{x}_i \in \mathbb{R}^n\}$ be a set of boundary points as illustrated in Figure 2.4

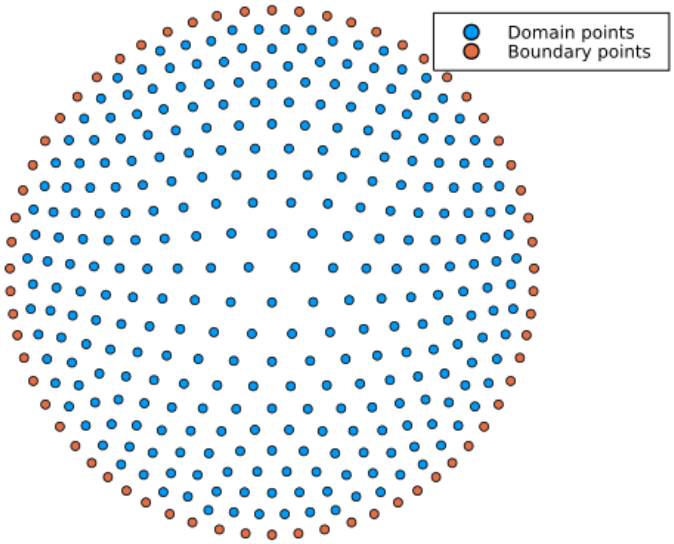

Figure 2.4: 2D domain discretization example

Next, the ansatz solution is formed in the same way as with an ordinary interpolation problem:

$$s(x) = \sum_{i=1}^{N+M} \phi(\boldsymbol{x} - \boldsymbol{x}_i)c_i + \sum_{j=1}^{Q} p_j(\boldsymbol{x}) \tag{2.17}$$

Polynomials are optional in case ϕ is positive definite, but necessary for conditionally positive definite functions. The ansatz function has to satisfy the differential operator at domain points and the boundary operator at the boundary. There are exactly as many equations as unknowns:

$$\begin{aligned} Ls(\boldsymbol{x}_i) &= f(\boldsymbol{x}_i) \quad \forall x_i \in X \\ Bs(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \quad \forall \boldsymbol{x}_i \in Y \\ \sum_{i=1}^{N+M} p_j(\boldsymbol{x}_i)c_i &= 0 \quad \forall j = 1, 2...Q \end{aligned} \tag{2.18}$$

Explicitly written out:

$$\begin{bmatrix} L_1 & L_2 & LP \\ B_1 & B_2 & BP \\ P_1^T & P_2^T & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ d \end{bmatrix} = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix} \tag{2.19}$$

Where the entries of the sub-matrices are as follows:

$$\begin{aligned} [L_1]_{ij} &= L\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad i = 1, \ldots, N; j = 1, \ldots, N \\ [L_2]_{ij} &= L\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad i = 1, \ldots, N; j = N+1, \ldots, N+M \\ [B_1]_{ij} &= B\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad i = N+1, \ldots, N+M; j = 1, \ldots, N \\ [B_2]_{ij} &= B\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad i = N+1, \ldots, N+M; j = N+1, \ldots, N+M \\ [LP]_{ij} &= Lp_j(x_i) \quad i = 1, \ldots, N; j = 1, \ldots, Q; \\ [BP]_{ij} &= Bp_j(x_i) \quad i = N+1, \ldots, N+M; j = 1, \ldots, Q; \\ \left[P_1^T\right]_{ij} &= p_i(x_j), \quad i = 1, \ldots, Q; j = 1, \ldots, N \\ \left[P_2^T\right]_{ij} &= p_i(x_j), \quad i = 1, \ldots, Q; j = N+1, \ldots, N+M \end{aligned}$$

Linear system 2.19 is not guaranteed to be invertible, but in practice singular matrices almost never occur. This method is known as Kansa, or unsymmetrical method for PDEs. There is another method which results in a symmetric positive definite set of equations. This method is based on the

concept of Hermitian (or generalized) RBF interpolation. Let $\{\lambda_1, \lambda_2 ... \lambda_N\}$ be a set of independent linear functionals , meaning $\lambda(af + bg) = a\lambda(f) + b\lambda(g)$ and $a_1\lambda_1 + ... + a_N\lambda_N = 0$ implies $a_1, a_2 ... a_N = 0$. And assume a function of the form 2.5 is required, such that:

$$\lambda_i(s) = f_i \in \mathbb{R} \quad \forall i = 1..N. \tag{2.20}$$

Functionals λ_i can be, for example, defined as follows: $\lambda_i(s) = s(\boldsymbol{x}_i)$. Note that in this case, the generalized interpolation problem is a classical interpolation from before. To solve a differential equation 2.16 the following functionals are required:

$$\begin{aligned} \lambda_i(s) &= Ls(\boldsymbol{x}_i) \quad \text{for } i = 1, \dots, N \\ \lambda_i(s) &= Bs(\boldsymbol{x}_i) \quad \text{for } i = N+1, \dots, N+M \end{aligned} \tag{2.21}$$

Applying the above functionals to 2.5 and equating the result to f and g is then equivalent to solving the differential equation2.16 using Kansa formulation. What makes generalized interpolation different from previous methods, is the way the interpolant is formed. Now, the ansatz function(interpolant) itself contains the functionals2.21:

$$s(x) = \sum_{i=1}^{N+M} \lambda_i^y \phi(\boldsymbol{x} - \boldsymbol{y}) + \sum_{j=1}^{Q} d_j p_j(x), \tag{2.22}$$

where λ^y means that the functional is applied to the second variable. Then the interpolation matrix becomes:

$$\begin{bmatrix} \lambda_1^x\lambda_1^y\phi & \dots & \lambda_1^x\lambda_N^y\phi & \lambda_1(p_1) & \dots & \lambda_1(p_N) \\ \lambda_2^x\lambda_1^y\phi & \dots & \lambda_2^x\lambda_N^y\phi & \lambda_2(p_1) & \dots & \lambda_2(p_Q) \\ \vdots & \dots & \vdots & \vdots & \dots & \vdots \\ \lambda_{N+M}^x\lambda_1^y\phi & \dots & \lambda_{N+M}^x\lambda_{N+M}^y\phi & \lambda_{N+M}(p_1) & \dots & \lambda_{N+M}(p_Q) \\ \lambda_1(p_1) & \dots & \lambda_{N+M}(p_1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda_1(p_Q) & \dots & \lambda_{N+M}(p_Q) & 0 & \dots & 0 \end{bmatrix} \tag{2.23}$$

The advantage of Symmetric method is that matrix 2.23 is always invertible as long as the functionals are linearly independent. The downside is that the functionals must be linear, so the solution of nonlinear PDEs is complicated.

2.6 Sparce Matrices

The computational cost of solving the interpolation problem is dominated by the solution of the dense linear system, which has algorithmic complexity $O(N^3)$.

Fortunately, there are ways to sparsify the linear system. One way is to use positive definite functions with compact support. One drawback of such functions is that they cannot be positive definite on $\mathbb{R}^d$ for all s, as captured by the following theorem [11]:

Theorem 3. *Suppose the continuous and nonvanishing function* $\phi : [0, \infty) \to \mathbb{R}$ *is positive definite on every* $\mathbb{R}^d$. *Then* $\phi(r) \neq 0$ *for all* $r \in [0, \infty)$.

Since RBFs with compact support are 0 for some values of r by definition, they can only be positive definite on $\mathbb{R}^k$ where $k \leq s$, s is some integer. The function is positive definite in all dimensions k less than s because $\mathbb{R}^k$ is a subspace of $\mathbb{R}^s$.

A popular choice of compactly supported RBFs are J. Wendland's piecewise polynomials of minimal degree[12]. These RBFs are constructed starting from functions

$$\varphi_\ell(r) = (1 - r)_+^\ell, \tag{2.24}$$

which are known to be positive definite on $\mathbb{R}^s$ for $\ell \geq \left\lfloor \frac{s}{2} \right\rfloor + 1$.

Then Wendland introduces an special operator which preserves positive definiteness:

$$I(f)(r) := \int_r^\infty s f(s) ds. \tag{2.25}$$

By successively applying I to φ it is possible to generate compactly supported functions which are positive definite and smooth. The general formula is

$$\varphi_{s,k} = \mathcal{I}^k \varphi_{\lfloor s/2 \rfloor + k + 1} \tag{2.26}$$

$\varphi_{s,k}$ is positive definite on $\mathbb{R}^s$ and has a smoothness C^{2k} For example:

$$\varphi_{3,3} = \int_r^1 r \left[\int_r^1 r \left[\int_r^1 r(1 - r)_+^5 dr \right] dr \right] dr = \frac{(1 - r)_+^8 \left(32r^3 + 25r^2 + 8r + 1\right)}{22176} \tag{2.27}$$

The constant in the denominator can be dropped, as it does not affect positive definiteness. To control the support radius of such functions, r can be scaled by some shape parameter ϵ. The support radius is then $\frac{1}{\epsilon}$

$$(1 - \epsilon r)_+^8 \left(32(\epsilon r)^3 + 25(\epsilon r)^2 + 8\epsilon r + 1\right) \tag{2.28}$$

Figure 2.5 shows the effect of the shape parameter on the compactly supported function in equation 2.28.

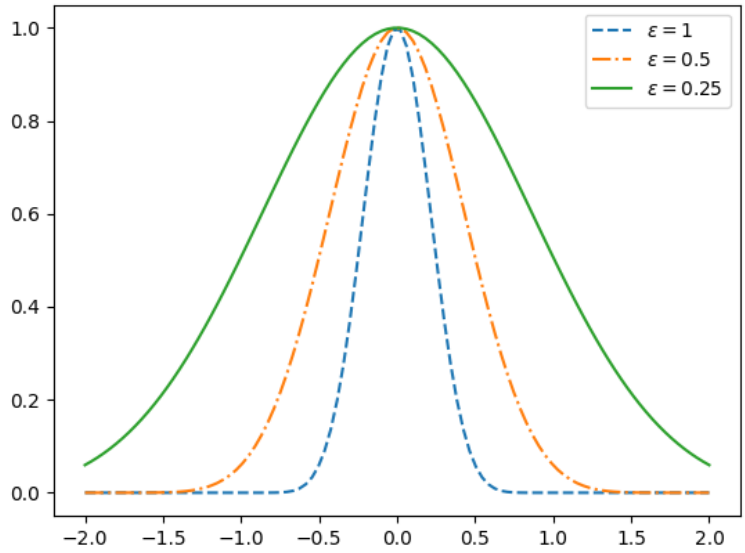

Figure 2.5: $\varphi_{3,3}$ function for different ϵ

Compact RBFs form sparse interpolation matrices, which are always invertible and cheap to solve. the interpolant, however, does not converge to the true solution with the decreasing fill distance if the bandwidth of the interpolation matrix is kept constant. This undesired effect arises from the necessity to decrease the support radius to sustain a constant matrix bandwidth, thereby resulting in basis functions with reduced flatness, ultimately impairing interpolation accuracy. The solution to this problem is to keep the support radius fixed. The downside is increased computational cost of a more dense interpolation matrix. More detailed discussion about RBF flatness and interpolation accuracy can be found in [13] [14] [15].

It is also possible to use RBF interpolation in so-called finite difference mode [5]. Unlike the global method, there is no assumed ansatz function here. The solution is not a function, but rather a restriction of the function to the collocation nodes, just like the classical finite difference. Let point sets X and Y be defined as before. For each point $\boldsymbol{x}_i \in X \cup Y$ assign a subset $X_i \subset X \cup Y$ such that $\boldsymbol{x}_i \in X_i$ and points in X_i are close to $\boldsymbol{x}_i$. X_i is called the stencil of

point $\boldsymbol{x}_i$. Let $\mathcal{I}_i$ be an index set of X_i

$$\mathcal{I}_i = \{k : \boldsymbol{x}_k \in X_i\}. \tag{2.29}$$

Also, define a local interpolant:

$$s_k(\boldsymbol{x}) = \sum_{i \in \mathcal{I}_k} \phi(\boldsymbol{x} - \boldsymbol{x}_i) \tag{2.30}$$

Then, a differential operator can be applied to each local interpolant and evaluated at the domain points

$$Ls_k(\boldsymbol{x}_k) = \sum_{i \in \mathcal{I}_k} L\phi(\boldsymbol{x}_k - \boldsymbol{x}_i) \quad \forall \boldsymbol{x}_k \in X \tag{2.31}$$

The solution is obtained by solving a linear system of equations above. Unlike the classical finite-difference scheme, the position of the collocation points can be arbitrary.

Another option to sparsify the linear system is a partition of unity(PU). The domain of interest, Ω, is partitioned into a set of overlaping sub-domains $\Omega_k \, k = 1, ..., N_\Omega$ such that $\Omega \subset \cup_{k=1}^{N_\Omega} \Omega_k$. Define $X_k = X \cap \Omega_k$ and $Y_k = Y \cap \Omega_k$ to be local domain and boundary points in k^{th} subdomain. Also, define the sets of indices belonging to each subdomain: $\mathcal{I}_k = \{i : x_i \in X_k\} \cup \{i : y_i \in Y_k\}$. Each subdomain is associated with a local interpolation function $s_k(x)$:

$$s_k(x) = \sum_{i \in \mathcal{I}_k} \phi(\boldsymbol{x} - \boldsymbol{x}_i) c_i + \sum_{i=1}^{Q} p_i(\boldsymbol{x}) d_i^k \tag{2.32}$$

$$\sum_{i \in \mathcal{I}_k} c_i p_j(\boldsymbol{x}_i) = 1 \quad j = 1...Q \tag{2.33}$$

Again, the additional polynomial basis is optional. The global solution is constructed as a weighted average of the local solutions:

$$s(\boldsymbol{x}) = \sum_{k=1}^{N_\Omega} s_k(\boldsymbol{x}) w_k(\boldsymbol{x}), \tag{2.34}$$

where weight functions w_k are compactly supported on Ω_k and have a partition of unity property:

$$\sum_{k=1}^{N_\Omega} w(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \Omega. \tag{2.35}$$

The sums in equations 2.34 and 2.35 usually do not run through all indices k, because any given $x \in \Omega$ belongs to only a few subdomains Ω_k. In the literature, weight functions w_k are usually constructed with the Shepard's method [16]. Let ψ_k be compactly supported function on Ω_k, then Shepard's weight functions are defined as follows:

$$w_k(x) = \frac{\psi_k(x)}{\sum_{k=1}^{N_\Omega} \psi_k(x)}. \tag{2.36}$$

To find the coefficients c_i and d_i, the differential and boundary operators are applied to the global interpolant and evaluated at the collocation points:

$$Ls(x_i) = f \forall x_i \in X$$
$$Bs(x_i) = g \forall x_i \in Y.$$

Together with 2.33 there are as many equations as unknowns. The sparsity of the resulting linear system depends on the number of points in each subdomain and the overlap between them.

In classical PU method, the differential operator L is properly applied to the global interpolation function, utilizing the product rule:

$$Ls(x) = \sum_{k=1}^{N_\Omega} Ls_k(x)w_k(x) + s_k(x)Lw_k(x) \tag{2.37}$$

This requires a smooth weight function and complicates the implementation. There is a recent modification of the above method called the Direct Partition of Unity (D-PU) [6]. In the case of D-PU, the differential operator is only applied to the local interpolant:

$$Ls(x) = \sum_{k=1}^{N_\Omega} w_k(x)Ls_k(x). \tag{2.38}$$

For D-PU method, the weight function does not need to be differentiable, or even continuous. In this example, the interpolant did not contain the differential operator. This approach can be classified as unsymmetric. There also exists a symmetric version of this method [17].

These are only some of the RBF methods available in the literature. This work focuses solely on the collocation methods, where the differential operator is satisfied at some discrete set of points. Global and local RBF methods are also available for weak forms of PDEs.

Chapter 3

Navier-Stokes equations

The Navier-Stokes (NS) equations describe the temporal evolution of the velocity field of the fluid. For an incompressible Newtonian fluid these equations are:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.1}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{f} \tag{3.2}$$

Equation 3.1 is a mass conservation equation, and 3.2 is a momentum conservation equation. Also, $\boldsymbol{u}$ is a velocity field, p is the pressure field, μ is the viscosity of the fluid and f is the force acting on the fluid. This study concentrates on the steady Navier-Stokes equations. In this is type of flow the velocity field is not a function of time, therefore all time derivatives are zero. Then, equation 3.2 becomes:

$$\rho(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{f} \tag{3.3}$$

To better understand different flow regimes it is common to work with non-dimensional form of the Navier-Stokes equations. Ignoring the force term and making the following substitutions into equation 3.3:

$$\begin{aligned}\nabla^* &= \nabla L \\ x_i^* &= x_i/L \\ \boldsymbol{u}^* &= \boldsymbol{u}/U, \\ p^* &= p/\rho U^2,\end{aligned}$$

where U is the characteristic velocity and L is characteristic length ,the following is derived:

$$(\mathbf{u}^* \cdot \nabla^*)\,\mathbf{u}^* = -\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\mathbf{u}^* \tag{3.4}$$

where $Re = \frac{UL\rho}{\mu}$ is the Reynolds number. It is the ratio between the inertial and viscous forces in the flow. For a more detailed discussion on dimensionless NS equations please refer to [18]. If Reynolds number is low, the convective term $(\mathbf{u}^* \cdot \nabla^*)\,\mathbf{u}^*$ is much smaller compared to the diffusive term $\nabla^{*2}\mathbf{u}^*$ and can be ignored:

$$-\frac{1}{Re}\nabla^{*2}\mathbf{u}^* + \nabla^* p^* = \mathbf{0}. \tag{3.5}$$

Rewriting the above equation in terms of the original variables with units and adding a force term results in:

$$-\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f}, \tag{3.6}$$

where $\nu = \frac{\mu}{\rho}$ is the dynamic viscosity. Equation 3.6 together with continuity equation 3.1 are known as Stokes equations. These equations are linear and easier to solve than the original NS equations. In the next chapter, a few methods for stationary Stokes equations are derived. In Chapter 5, a method for stationary NS equations is introduced.

Chapter 4

Application of RBFs to Stokes equations

As derived in the previous section, the stationary Stokes problem together with Dirichlet boundary conditions is stated as follows:

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \boldsymbol{u} &= g \quad \text{in } \partial\Omega \end{aligned} \tag{4.1}$$

This chapter describes a few existing approaches to solving the above equations as well as a new method based on the partition of unity. The focus is only on the stationary equations mainly because the time-varying case can be seen as a sequence of linear problems similar to 4.1 [9].

4.1 Straightforward approach

To solve Equation 4.1 Let $X = \boldsymbol{x}_1, \boldsymbol{x}_2, ..\boldsymbol{x}_N$ be a set of scattered points in the domain Ω and $Y = \boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+2}, ..\boldsymbol{x}_{N+M}$ be a set of points located on the boundary of the domain $\partial\Omega$. each component of the velocity and the pressure can then be approximated as sum of radial basis functions. For example,

consider the 2D situation:

$$u(\boldsymbol{x}) = \sum_{i=1}^{N+M} c_i^1 \phi(\boldsymbol{x} - \boldsymbol{x}_i) \tag{4.2}$$

$$v(\boldsymbol{x}) = \sum_{i=1}^{N+M} c_i^2 \phi(\boldsymbol{x} - \boldsymbol{x}_i) \tag{4.3}$$

$$p(\boldsymbol{x}) = \sum_{i=1}^{N} c_i^3 \phi(\boldsymbol{x} - \boldsymbol{x}_i) \tag{4.4}$$

There are $3N + 2M$ unknowns and the same number of equations, so a square linear system can be formed:

$$\begin{bmatrix} -\nu\Delta\phi & 0 & \partial_1\phi \\ 0 & -\nu\Delta\phi & \partial_2\phi \\ \partial_1 & \partial_2 & 0 \\ \phi & 0 & 0 \\ 0 & \phi & 0 \end{bmatrix} \begin{bmatrix} c^1 \\ c^2 \\ c^3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ 0 \\ g_1 \\ g_2 \end{bmatrix} \tag{4.5}$$

Note, that the matrix is square, even if it is not square in the block form. Solving the equations this way would be equivalent to a Kansa method for scalar equations. In this thesis this approach is not discussed further, as this family of methods leads to a saddle point problem. This is only an issue for problems which are far larger then the test examples provided in this work. The goal is, however, to develop a scalable method that could potentially be used for real engineering applications, therefore this Kansa-like approach is not developed further. To use a symmetric form, a generalization of positive definite kernels is useful. Further definitions are taken from [8].

4.2 Divergence free formulation

It is possible to construct an analytically divergence-free solution to 4.1 by using a matrix-valued kernel Φ instead of the scalar ϕ from previous chapters. In this case, the ansatz function automatically satisfies the continuity equation, thus only the momentum equation and boundary conditions are required. This method was first described in [19], using theoretical results from [8]. Before introducing the method, it is useful to introduce a generalization of positive definite radial basis functions.

Definition 3. *A matrix-valued function* $\Phi : \mathbb{R}^d \to \mathbb{R}^{n\times n}$ *is called positive*

definite if $\Phi(-\boldsymbol{x}) = \Phi(\boldsymbol{x})$, $\Phi(x) = \Phi^T(\boldsymbol{x})$ *and*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{\alpha}_i^T \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\boldsymbol{\alpha}_j \ > 0, \tag{4.6}$$

for all N *pairwise distinct points* $\boldsymbol{x}_i \in \mathbb{R}^d$ *and nonzero vectors* $\boldsymbol{\alpha}_i \in \mathbb{R}^n$

One application of matrix-valued RBFs is vector field interpolation: given a set of N points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ in $\mathbb{R}^d$ and a vector field sampled at those points: $\boldsymbol{u}_1, \boldsymbol{u}_2, ..\boldsymbol{u}_N$ in $\mathbb{R}^n$, the RBF interpolant will take the following form:

$$\boldsymbol{s}(\boldsymbol{x}) = \sum_{i=1}^{N} \Phi(\boldsymbol{x} - \boldsymbol{x}_i)\boldsymbol{c}_i. \tag{4.7}$$

Where $\boldsymbol{c}_i \in \mathbb{R}^n$ are now vector-valued coefficients. A trivial example of matrix-RBFs is a tensor product of scalar RBFs :

$$\Phi = \begin{bmatrix} \phi_1 & & \\ & \ddots & \\ & & \phi_n. \end{bmatrix} \tag{4.8}$$

If $\phi_1, \phi_2, ..., \phi_n$ are all positive definite, then Φ is also positive definite by definition. Note, that using such a matrix kernel is equivalent to interpolating each component of the vector field separately.

Another concept from scalar RBF interpolation, that extends naturally to matrix-valued RBFs is generalized interpolation. Let $\mathbf{C}(\mathbb{R}^d, \mathbb{R}^n)$ be a space of continuous functions from $\mathbb{R}^d$ to $\mathbb{R}^n$ Consider a set of independent linear functionals $\{\lambda_1, \lambda_2..\lambda_N\}$ where $\lambda_i : \mathbf{C}(\mathbb{R}^d, \mathbb{R}^n) \to \mathbb{R}$ for $i = 1, ..., n$ As these functionals are linear, they can be represented as a vector of distributions. Applying a functional to a function is then equivalent to the integration of a dot product of a distribution and a function itself. As an example, suppose $d = 2$, $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{f}(x_1, x_2) = [f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2)]^T$ and $\lambda = [-\partial_1\delta(x_1, x_2), 0, -\partial_2\delta(x_1 - a, x_2 - b)]$ where δ is Dirac delta distribution and ∂_1 is the derivative with respect to x_1. Then

$$\begin{aligned}\lambda(f) &= \int_{\infty}^{\infty}\int_{\infty}^{\infty} \left(-\partial_1\delta(x_1, x_2)f_1(x_1, x_2) - \partial_2\delta(x_1 - a, x_2 - b)f_3(x_1, x_2)\right) dx_1 dx_2 \\ &= \partial_1 f_1(0, 0) + \partial_2 f_3(a, b).\end{aligned}$$

As can be seen, this functional returns the sum of partial derivatives of the first and second components of the function f evaluated at the origin and a point (a, b). In fact, any linear differential operator, such as in Stokes equation can be represented as a linear functional. Consider the following ansatz function:

$$\boldsymbol{s}(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i^{\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}) c_i \tag{4.9}$$

where λ_i^{ξ} is applied to the rows of Φ. The top index ξ means that the functional is applied with respect to the second argument. Polynomial basis can also be added, however, they need to form a divergence-free space, otherwise, the ansatz will no longer satisfy the continuity equation. The divergence-free polynomial basis is described in more detail later in this chapter. Below the procedure for applying a functional to the matrix-valued kernel is defined.

$$\lambda_i^{\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}) = \begin{bmatrix} \lambda_i^{\xi}([\Phi_{11}(\boldsymbol{x} - \boldsymbol{\xi}) \ldots \Phi_{1d}(\boldsymbol{x} - \boldsymbol{\xi})]) \\ \lambda_i^{\xi}([\Phi_{21}(\boldsymbol{x} - \boldsymbol{\xi}) \ldots \Phi_{2d}(\boldsymbol{x} - \boldsymbol{\xi})]) \\ \vdots \\ \lambda_i^{\xi}([\Phi_{d1}(\boldsymbol{x} - \boldsymbol{\xi}) \ldots \Phi_{dd}(\boldsymbol{x} - \boldsymbol{\xi})]) \end{bmatrix} \tag{4.10}$$

Weights c_i in Equation 4.9 are determined by solving the following linear system:

$$\boldsymbol{\Phi c} = \boldsymbol{f}, \quad \text{where } \boldsymbol{\Phi}_{ij} = \lambda_i^x \lambda_j^{\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}). \tag{4.11}$$

For a classical interpolation problem 4.7, the following set of functionals is needed:

$$\{\delta(\boldsymbol{x} - \boldsymbol{x_i}) \boldsymbol{e}_j : i = 1, ..N,\ j = 1, ..d\}. \tag{4.12}$$

Where $\boldsymbol{e}_j$ is the j^{th} unit vector. These functionals evaluate each component of the vector field at every collocation node. There is not much utility in matrix-valued RBFs of the form 4.8, as the ansatz function it generates can be produced with scalar valued RBFs as well. For incompressible fluid dynamics applications it is desirable to have anszatz function which is divergence free. If the columns of Φ from 4.7 are divergence free, then their linear combination is also divergence free. Such matrix- valued kernel was first discovered and analyzed by H. Wendland [19]. Here the derivation and proof are omitted, only the final result is introduced. Let ϕ be a positive definite radial basis function, then

$$\Phi_{div} = \left(-\Delta I + \nabla \nabla^T\right) \phi, \tag{4.13}$$

or in tensor form:

$$[\Phi_{div}]_{ij} = \partial_{ij}\phi - \delta_{ij}\partial_{ij}\phi \tag{4.14}$$

is divergence-free and positive definite as in Definition 3. This divergence-free matrix-valued kernel can be used in the ansatz function 4.9 to solve Stokes equations. Consider 2D stationary Stokes problem 4.1. There are $2N$ functionals corresponding to the Stokes operator applied at each collocation point in the domain and $2M$ functionals for the Dirichlet operator applied at each boundary collocation point. These functionals are:

$$\begin{cases} \lambda_i(\boldsymbol{u}) = \delta_{x_i} \circ \left[-\Delta(\boldsymbol{u}^T\boldsymbol{e}_1) + \partial_1(\boldsymbol{u}^T\boldsymbol{e}_3)\right] & i = 1...N \\ \lambda_{i+N}(\boldsymbol{u}) = \delta_{x_i} \circ \left[-\Delta(\boldsymbol{u}^T\boldsymbol{e}_2) + \partial_2(\boldsymbol{u}^T\boldsymbol{e}_3)\right] & i = 1...N \\ \lambda_{i+2N}(\boldsymbol{u}) = \delta_{x_{i+N}} \circ \boldsymbol{u}^T\boldsymbol{e}_1 & i = 1...M \\ \lambda_{i+2N+M}(\boldsymbol{u}) = \delta_{x_{i+N}} \circ \boldsymbol{u}^T\boldsymbol{e}_2 & i = 1...M \end{cases} \tag{4.15}$$

Where $\boldsymbol{u} = [u(\boldsymbol{x}), v(\boldsymbol{x}), p(\boldsymbol{x})]$ is a vector of velocities and pressure, $\boldsymbol{e}_i$ is the i^{th} unit vector and δ_x evaluetes the function at point x. Note that the first 2 functionals correspond to momentum equation and last 2 functionals correspond to boundary conditions in 4.1. The solution now has 3 components: u and v velocities as well as pressure p. This means that the kernel has to be a 3×3 matrix. However 4.13 is not sufficient alone, because only the first 2 components of $\boldsymbol{u}$ have to be divergence-free. This issue is solved by using a tensor product of a divergence-free kernel and a scalar RBF.

$$\Phi = \begin{bmatrix} \Phi_{div} & 0 \\ 0 & \phi_p \end{bmatrix} = \begin{bmatrix} -\partial_{22}\phi & \partial_{12}\phi & 0 \\ \partial_{21}\phi & -\partial_{11}\phi & 0 \\ 0 & 0 & \phi_p \end{bmatrix} \tag{4.16}$$

where ϕ and ϕ_p are positive definite RBFs, which can, but don't have to be, different. As can be seen, kernel in 4.16 has a block structure with a divergence-free sub-matrix for the velocity and a scalar RBF for pressure. The ansatz function then has the form

$$
\begin{aligned}
\boldsymbol{s}(\boldsymbol{x}) &= \sum_{i=1}^{2N+2M} \lambda_i^{\boldsymbol{\xi}} \Phi(\boldsymbol{x}-\boldsymbol{\xi}) c_i \\
&= \sum_{i=1}^{N} c_i \begin{pmatrix} \nu\partial_{22}\Delta\phi\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \\ -\nu\partial_{12}\Delta\phi\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \\ -\partial_1\phi_p\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \end{pmatrix} + \sum_{i=1}^{N} c_{i+N} \begin{pmatrix} -\nu\partial_{12}\Delta\phi\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \\ \nu\partial_{11}\Delta\phi\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \\ -\partial_2\phi_p\left(\boldsymbol{x}-\boldsymbol{x}_i\right) \end{pmatrix} \\
&+ \sum_{i=1}^{M} c_{i+2N} \begin{pmatrix} -\partial_{22}\phi\left(\boldsymbol{x}-\boldsymbol{x}_{i+N}\right) \\ \partial_{12}\phi\left(\boldsymbol{x}-\boldsymbol{x}_{i+N}\right) \\ 0 \end{pmatrix} + \sum_{i=1}^{M} c_{i+2N+M} \begin{pmatrix} \partial_{12}\phi\left(\boldsymbol{x}-\boldsymbol{x}_{i+N}\right) \\ -\partial_{11}\phi\left(\boldsymbol{x}-\boldsymbol{x}_{i+N}\right) \\ 0 \end{pmatrix} \\
&+ \sum_{i=1}^{Q} \boldsymbol{p}_i(\boldsymbol{x}) d_i
\end{aligned} \tag{4.17}
$$

Here the polynomial basis is added as well. To explain the construction of such polynomial basis, first define $\mathbb{P}_m^{d\to n}$ to be a space of vectors with n components, where each component is a d-variate polynomial of degree at most m. Also, define $\mathbb{P}_{div_m}^{d\to n} = \mathbb{P}_m^{d\to n} \cap \mathbf{C}_{div}\left(\mathbb{R}^d, \mathbb{R}^n\right)$ where $\mathbf{C}_{div}\left(\mathbb{R}^d, \mathbb{R}^n\right)$ is the space of n-component ,d-variate divergence-free functions. Then for the 2D Stokes ansatz function the following polynomials are used: span$\{\boldsymbol{p}_1, ..\boldsymbol{p}_Q\} = \mathbb{P}_{div_m}^{2\to 2} \otimes \{p \in \mathbb{P}_{m2}^{2\to 1} : p \neq c\}$. Where c is a constant scalar. To be more precise, only components corresponding to velocity field are divergence-free and the pressure part cannot contain constant functions, as it leads to singular systems. This is caused by the fact that only a derivative of the pressure appears in the Stokes equations with Dirichlet boundary conditions.

Note that there are no functionals corresponding to the conservation equation because the ansatz function is automatically divergence-free, no matter what the coefficients are. Also note that the functionals are applied with respect to the second variable $\boldsymbol{\xi}$. This is why some terms have an otherwise unexpected minus sign. The ansatz function $\boldsymbol{s}$ for Stokes equations has to satisfy the following interpolation conditions:

$$
\begin{cases}
\lambda_i(\boldsymbol{s}) = f_1(\boldsymbol{x}_i) & i = 1..N \\
\lambda_{i+N}(\boldsymbol{s}) = f_2(\boldsymbol{x}_i) & i = 1..N \\
\lambda_{i+2N}(\boldsymbol{s}) = g_1(\boldsymbol{x}_i) & i = 1..M \\
\lambda_{i+2N+M}(\boldsymbol{s}) = g_2(\boldsymbol{x}_i) & i = 1..M
\end{cases} \tag{4.18}
$$

Plugging 4.17 into 4.18 yeilfs the following linear system

$$A\boldsymbol{c} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & P_1 \\ A_{21} & A_{22} & A_{23} & A_{24} & P_1 \\ A_{31} & A_{32} & A_{33} & A_{34} & P_1 \\ A_{41} & A_{42} & A_{43} & A_{44} & P_1 \\ P_1^T & P_2^T & P_3^T & P_4^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{c}_3 \\ \boldsymbol{c}_4 \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ 0 \end{bmatrix} \tag{4.19}$$

Entries of the matrix 4.19 are written out explicitly in Appendix A.

In the original paper [19], Wendland's compactly supported $C^8(\mathbb{R}^2)$ function 4.20 is used with a shape parameter $\epsilon = 0.1$, such that the interpolation matrix is dense.

$$\phi(r) = (1 - r\epsilon)_+^{10} \left(429(r\epsilon)^4 + 450(r\epsilon)^3 + 210(r\epsilon)^2 + 50r\epsilon + 5\right) \tag{4.20}$$

As was discussed in earlier sections, the shape parameter influences the interpolation accuracy and the condition number of the linear system. Choosing the right parameter is tricky, and is usually determined empirically by trial and error. In this work, polyharmonic RBFs are used for the first time in the context of matrix-valued kernels. These functions do not have a shape parameter and possess desirable properties, which are exploited in later sections. For scalar interpolation it is known that polyharmonics are conditionally positive definite, requiring an additional polynomial basis. There are no theoretical results in the literature concerning the behaviour of polyharmonic RBFs in matrix-valued setting, such as 4.13. Nevertheless, numerical experiments in this work hint that Matrix RBFs formed by polyharmonics are indeed positive semi-definite.

In the following numerical experiment, the true solution is assumed to be:

$$\begin{cases} u(x,y) = -y * sin(x^2 + y^2) \\ v(x,y) = x * sin(x^2 + y^2) \\ p(x,y) = sin(x - y), \end{cases} \tag{4.21}$$

with appropriate forcing terms that satisfy the stationary Stokes equations. The matrix kernel is 4.16 with $\phi = r^7$ and $\phi_p = r^3$. These RBFs have the lowest odd degree that satisfy the smoothness requirements on the velocity and pressure. For the numerical experiment, the domain is $[0,1] \times [0,1]$. Collocation points are placed uniformly on a rectangular grid, as displayed in figure 4.1.

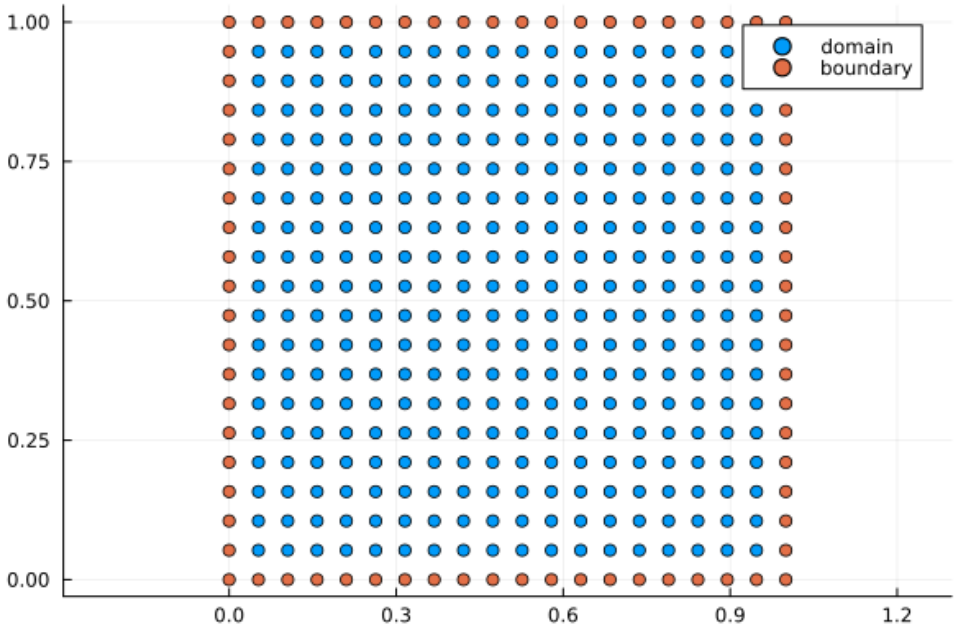

Figure 4.1: Domain and boundary points of a benchmark problem

Figure 4.2 shows the maximum error between the true velocity and recovered velocity for different values of a fill distance h. This experiment is done both for Polyharmonic kernel and a conventional positive definite Wendland kernel form [19]. m is the maximum degree of divergence-free polynomial for velocity field, n is the maximum degree of polynomials for pressure.

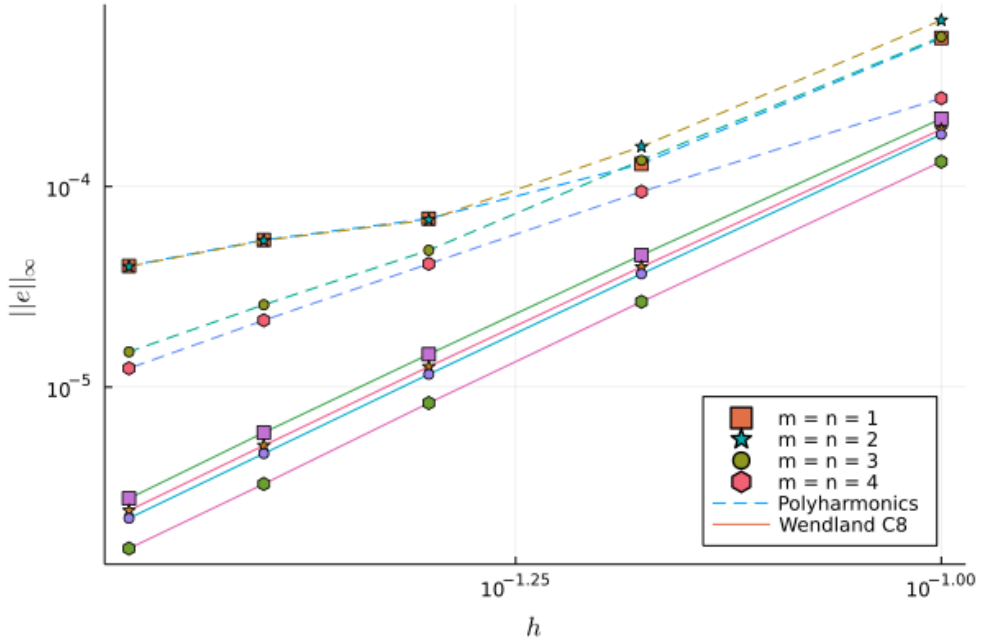

Figure 4.2: convergence of polyharmonic and wendland RBFs

As can be seen in figure 4.2, additional polynomial basis imroves the error, but does not change the convergence rate for Wendland's kernel. For polyharmonic kernel, additional polynomials change the convergence behaviour for small fill distances. This might be related to the fact that polyharmonic kernel is conditionally positive definite.

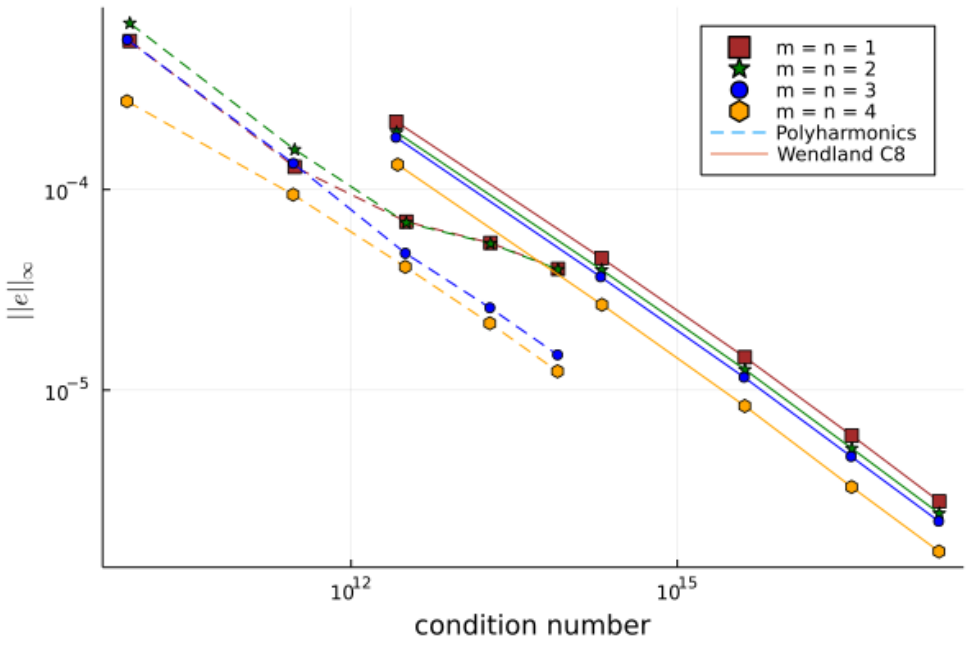

Figure 4.3: condition number of polyharmonic and wendland RBFs

Figure 4.3 shows the maximum error vs the condition number of the interpolation matrix 4.19. Even though polyharmonic kernel has worse convergence rate, it is better conditioned. In latter section the condition number of polyharmonic kernel will be further reduced with the help of a special preconditioner.

4.3 Local Hermitian interpolation

The interpolation matrix 4.19 is full. Therefore, Wendland's method described in the previous section is not suitable for large problems. Local Hermitian interpolation (LHI) is a technique for the efficient solution of linear PDEs using radial basis functions. It is first mentioned in [7] by D. Stevens and then further developed in [20] [21]. LHI was firs applied to Stokes equations in [9]. In this section LHI is introduced and tested for polyharmonic kernel on a benchmark solution. Define $X_I = \{x_I^1, ...x_I^{N_I}\} \subset \Omega$, $X_L = \{x_L^1, ...x_L^{N_L}\} \subset \Omega$ and $X_B = \{x_B^1, ...x_B^{N_B}\} \subset \partial\Omega$ to be sets of solution centers, PDE centers and boundary centers respectively. Each solution center x_I^k is associated with some patch, Ω^k, such that $x_I^k \in \Omega^k$. Now, define the following sets of neighbouring centers: $X_I^k = \Omega^k \bigcap X_I$, $X_L^k = \Omega^k \bigcap X_L$ and $X_B^k = \Omega^k \bigcap X_B$. An example

is provided inFigure 4.4.

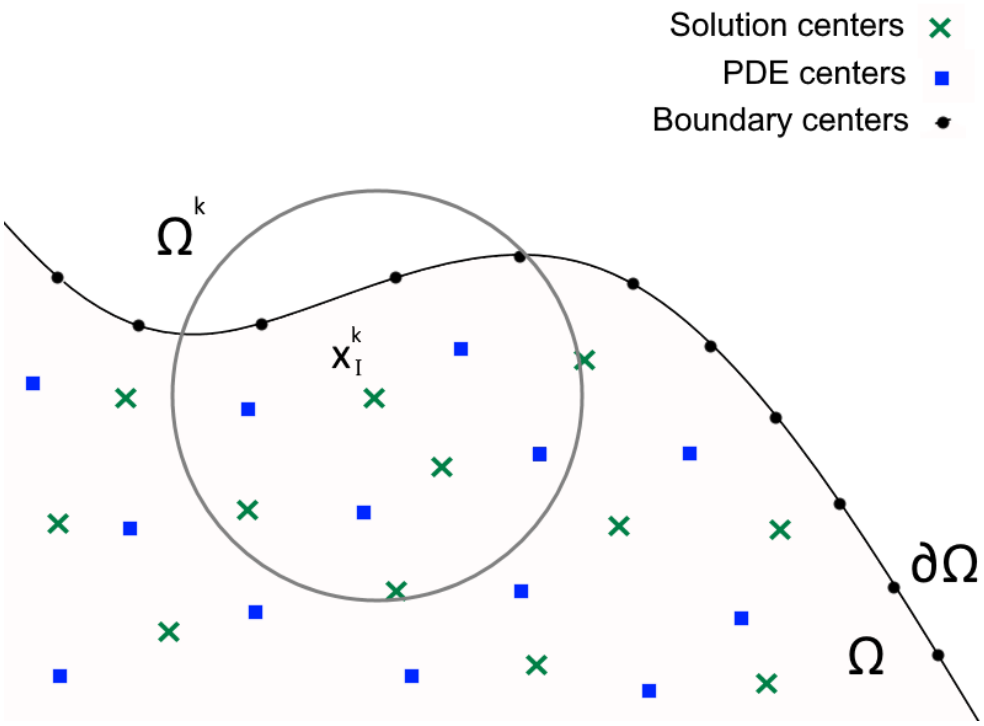

Figure 4.4: the typical collocation center layout in LHI

Next, define the index sets $\mathcal{I}^k = \{i : x_I^i \in X_I^k\}$, $\mathcal{L}^k = \{i : x_L^i \in X_L^k\}$, $\mathcal{B}^k = \{i : x_B^i \in X_B^k\}$ and the following functionals:

$$\begin{cases} \lambda_i^j(\boldsymbol{u}) = \delta_{x_I^i} \circ \boldsymbol{u}^T \boldsymbol{e}_j & i = 1, ..., N_I \; j = 1, ..., d \\ \alpha_i^j(\boldsymbol{u}) = \delta_{x_L^i} \circ \left[-\Delta(\boldsymbol{u}^T \boldsymbol{e}_j) + \partial_j(\boldsymbol{u}^T \boldsymbol{e}_{d+1})\right] & i = 1, ..., N_L \; j = 1, ..., d \\ \beta_i^j(\boldsymbol{u}) = \delta_{x_B^i} \circ \boldsymbol{u}^T \boldsymbol{e}_j & i = 1, ..., N_B \; j = 1, ..., d \end{cases} \tag{4.22}$$

These are the same as in 4.15 but with extra functionals that evaluate the function at the solution centers. Also, functionals are stated not only for 2D flow but also a for general dimension d. As before, however, only 2D benchmarks are provided in this work.

Each solution center ,x_I^k, is associated with a local interpolant $\boldsymbol{s}^k$. If there are no boundary points in Ω^k, then corresponding functionals are missing from

the ansatz function 4.23.

$$\boldsymbol{s}^k(\boldsymbol{x}) = \sum_{j=1}^{d}\left[\sum_{i\in\mathcal{I}^k}\lambda_i^{j\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi})l_i^j + \sum_{i\in\mathcal{L}^k}\alpha_i^{j\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi})a_i^j + \sum_{i\in\mathcal{B}^k}\beta_i^{j\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi})b_i^j\right] + \sum_{i=1}^{Q}\boldsymbol{p}d_i^k \tag{4.23}$$

As before, coefficients l, a and b are found by plugging $\boldsymbol{s}^k$ into the functionals related to Ω^k. For each Ω^k, the following local linear system arises:

$$\boldsymbol{A}^k\begin{bmatrix}l^k\\a^k\\b^k\\d^k\end{bmatrix} = \begin{bmatrix}\lambda\lambda\Phi_k & \lambda\alpha\Phi_k & \lambda\beta\Phi_k & \lambda P_k\\ \alpha\lambda\Phi_k & \alpha\alpha\Phi_k & \alpha\beta\Phi_k & \alpha P_k\\ \beta\lambda\Phi_k & \beta\alpha\Phi_k & \beta\beta\Phi_k & \beta P_k\\ \lambda P_k^T & \alpha P_k^T & \beta P_k^T & 0\end{bmatrix}\begin{bmatrix}l^k\\a^k\\b^k\\d^k\end{bmatrix} = \begin{bmatrix}u^k\\f^k\\g^k\\0\end{bmatrix} \tag{4.24}$$

Where

$$l^k = \begin{bmatrix}l_i^1\\\vdots\\l_i^d\end{bmatrix}\ i\in\mathcal{I}^k \quad a^k = \begin{bmatrix}a_i^1\\\vdots\\a_i^d\end{bmatrix}\ i\in\mathcal{L}^k \quad b^k = \begin{bmatrix}b_i^1\\\vdots\\b_i^d\end{bmatrix}\ i\in\mathcal{B}^k \tag{4.25}$$

$$d^k = \begin{bmatrix}d_i^k\\\vdots\\d_i^k\end{bmatrix}\ i = 1...Q, \tag{4.26}$$

$$u^k = \begin{bmatrix}u_1(\boldsymbol{x_I^i})\\\vdots\\u_d(\boldsymbol{x_I^i})\end{bmatrix}\ i\in\mathcal{I}^k \quad f^k = \begin{bmatrix}f_1(\boldsymbol{x_L^i})\\\vdots\\f_d(\boldsymbol{x_L^i})\end{bmatrix}\ i\in\mathcal{L}^k \quad b^k = \begin{bmatrix}g_1(\boldsymbol{x_B^i})\\\vdots\\g_d(\boldsymbol{x_B^i})\end{bmatrix}\ i\in\mathcal{B}^k. \tag{4.27}$$

The block matrices in 4.24 are the following:

$$\lambda\lambda\Phi_k \in \mathbb{R}^{N_I d\times N_I d} = \begin{bmatrix}\lambda_i^{1x}\lambda_j^{1\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi}) & \dots & \lambda_i^{1x}\lambda_j^{d\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi})\\ \vdots & \ddots & \vdots\\ \lambda_i^{dx}\lambda_j^{1\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi}) & \dots & \lambda_i^{dx}\lambda_j^{d\xi}\Phi(\boldsymbol{x}-\boldsymbol{\xi})\end{bmatrix}\quad i,j\in\mathcal{I}^k \tag{4.28}$$

Superscripts on top of the λ indicate which variable the functional is applied to. Other blocks are constructed similarly. Local linear system 4.24 assumes

that boundary points are present in the stencil. If Ω^k does not contain boundary points, terms corresponding to boundary points are not present:

$$\boldsymbol{A}^k \begin{bmatrix} l^k \\ a^k \\ d^k \end{bmatrix} = \begin{bmatrix} \lambda\lambda\Phi_k & \lambda\alpha\Phi_k & \lambda P_k \\ \alpha\lambda\Phi_k & \alpha\alpha\Phi_k & \alpha P_k \\ \lambda P_k^T & \alpha P_k^T & 0 \end{bmatrix} \begin{bmatrix} l^k \\ a^k \\ d^k \end{bmatrix} = \begin{bmatrix} u^k \\ f^k \\ 0 \end{bmatrix} \tag{4.29}$$

The right-hand side of 4.24 is known except for u^k, which is a solution to be found. Assume for a moment that the velocity, u, is known. Then each interpolant $\boldsymbol{s}^k$ can be found by solving 4.24. Having $\boldsymbol{s}^k$ it is possible to apply the Stokes differential operator to it and evaluate at each x_I^k. In other words, leave u^k as an unknown, apply α_i^j functionals to $\boldsymbol{s}^k$ and obtain a global linear system from which unknown velocity u at each solution center is found. Below stokes functionals at the solution centers are introduced:

$$\alpha_k^j(\boldsymbol{u}) = \delta_{x_I^k} \circ \left[-\Delta(\boldsymbol{u}^T \boldsymbol{e}_j) + \partial_j(\boldsymbol{u}^T \boldsymbol{e}_{d+1})\right] \quad j = 1, ..., d \tag{4.30}$$

Note that they are very similar to those in 4.22, but evaluate the function at the solution centers instead of PDE centers.

Next, the differential operator is applied to 4.23 and evaluated at the solution centers. This is equivalent to applying functionals 4.30 to the ansatz 4.23. For each solution center the following set of equations holds:

$$L_q \boldsymbol{s}^k(\boldsymbol{x_I^k}) = \alpha_k^q(\boldsymbol{s}^k) \tag{4.31}$$

$$= \sum_{j=1}^{d} \left[\sum_{i \in \mathcal{I}^k} \alpha_k^{qx} \lambda_i^{j\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}) l_i^j + \sum_{i \in \mathcal{L}^k} \alpha_k^{qx} \alpha_i^{j\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}) a_i^j \right.$$
$$\left. + \sum_{i \in \mathcal{B}^k} \alpha_k^{qx} \beta_i^{j\xi} \Phi(\boldsymbol{x} - \boldsymbol{\xi}) b_i^j \right] + \sum_{i=1}^{Q} \alpha_k^q \boldsymbol{p}\, d_i^k = \boldsymbol{f}(\boldsymbol{x}_I^k) \quad q = 1, ..., d \tag{4.32}$$

In matrix form this is:

$$\boldsymbol{B}^k \begin{bmatrix} l^k \\ a^k \\ b^k \\ d^k \end{bmatrix} = \begin{bmatrix} \alpha_k^1 \lambda\Phi_k & \alpha_k^1 \alpha\Phi_k & \alpha_k^1 \beta\Phi_k & \alpha_k^1 P_k \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_k^d \lambda\Phi_k & \alpha_k^d \alpha\Phi_k & \alpha_k^d \beta\Phi_k & \alpha_k^d P_k \end{bmatrix} \begin{bmatrix} l^k \\ a^k \\ b^k \\ d^k \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}_I^k) \\ \vdots \\ f_d(\boldsymbol{x}_I^k) \end{bmatrix}. \tag{4.33}$$

Substituting 4.24 into 4.33 results in:

$$\boldsymbol{B}^k \boldsymbol{A}^{k^{-1}} \begin{bmatrix} u^k \\ f^k \\ g^k \\ 0 \end{bmatrix} := \boldsymbol{C}^k \begin{bmatrix} u^k \\ f^k \\ g^k \\ 0 \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}_I^k) \\ \vdots \\ f_d(\boldsymbol{x}_I^k) \end{bmatrix}. \tag{4.34}$$

Each $\boldsymbol{C}^k$ has d rows. Each row represents an equation for the unknown u. In total, there are N_I such matrices, so there are dN_I equations for the velocity field. These equations can be assembled in the global linear system as follows: First partition $\boldsymbol{C}^k$ into blocks with sizes $d \times \#\mathcal{I}^k$, $d \times \#\mathcal{L}^k$, $d \times \#\mathcal{B}^k$, $d \times Q$ such that they match $[u^k f^k g^k 0]^T$

$$\boldsymbol{C}^k := \begin{bmatrix} \boldsymbol{C}_u^k & \boldsymbol{C}_f^k & \boldsymbol{C}_g^k & \boldsymbol{C}_p^k \end{bmatrix}. \tag{4.35}$$

Then, define a global matrix $G \in \mathbb{R}^{dN_I \times dN_I}$ and right hand side vector $b \in \mathbb{R}^{dN_I}$. They are assembled by looping over each k:

$$\boldsymbol{G}_{ij} = \boldsymbol{C}_u^k \quad i = (k, k+N_I, ...k+dN_I);\ j \in (q + nN_I^k : q \in \mathcal{I}^k,\ n = 1, ..., d)$$

The other blocks of $\boldsymbol{C}^k$ are used to construct the global right hand side.

$$\mathbf{RHS}_i = \begin{bmatrix} \boldsymbol{C}_f^k & \boldsymbol{C}_g^k & \boldsymbol{C}_p^k \end{bmatrix} \begin{bmatrix} f^k \\ g^k \\ 0 \end{bmatrix} \quad i = (k, k+N_I ... k+dN_I). \tag{4.36}$$

$$\boldsymbol{G}\boldsymbol{u} = \mathbf{RHS} \tag{4.37}$$

The resulting linear system 4.37 is sparse and has a size of $dN_I \times dN_I$. The sparsity is controlled by the size of the stencil Ω^k. It is important that the number of points in the stencil is larger than the dimension of the polynomial basis Q, otherwise, the local matrix is singular. If there is no polynomial basis in the local interpolant, then there are no restrictions on the stencil size.

LHI is tested for the stationary Stokes problem with $\nu = 1$ on a domain $[0, 1] \times [0, 1]$ with regular point spacing. Solution centers and PDE centers coincide, but when assembling local interpolation matrices, set $\mathcal{L}^k$ excludes the point x_I^k. If this is not done, the differential operator is enforced twice at the same point: once in the local system and once in the global system. This

leads to a singular global matrix. The true solution is given by:

$$u(x,y) = 20xy^3$$
$$v(x,y) = 5x^4 - 5y^4$$
$$p(x,y) = 60x^2y - 20y^3.$$

With the above solution, the forcing terms in 4.1 are zero. The true solution is a polynomial on purpose. It will be recovered exactly if the polynomial basis in 4.23 contains the true solution. Polyharmonic and Wendland[19] kernels with a polynomial basis with $m = 4$ and $m_2 = 2$ are tested. Figure 4.5 shows the maximum velocity error vs separation distance for both kernels. It is interesting to observe that even though LHI is a local method it has a polynomial reproduction property. If the polynomial basis $m = 4$, $m_2 = 3$ is used, the solution is contained in the azats function and the error is close to machine precision (accounted for the condition number of local system). Polynomial reproduction property is achieved when $m = 4$ and $m_2 = 3$. The error is illustrated in figure 4.6.

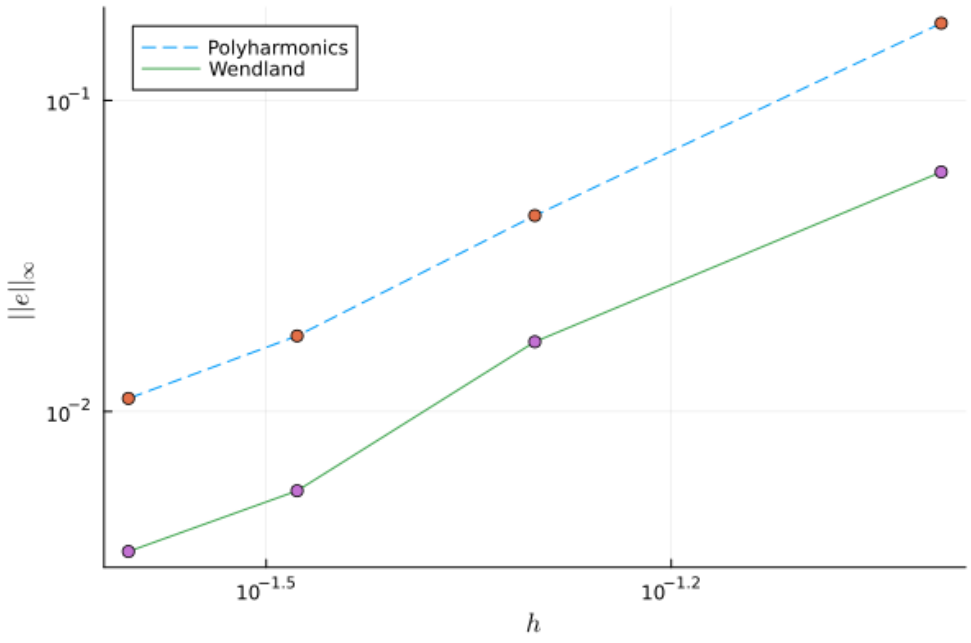

Figure 4.5: Convergence of LHI method for the Polyharmonic and Wendland RBFs with $m = 4$,$m_2 = 2$ polynomial basis

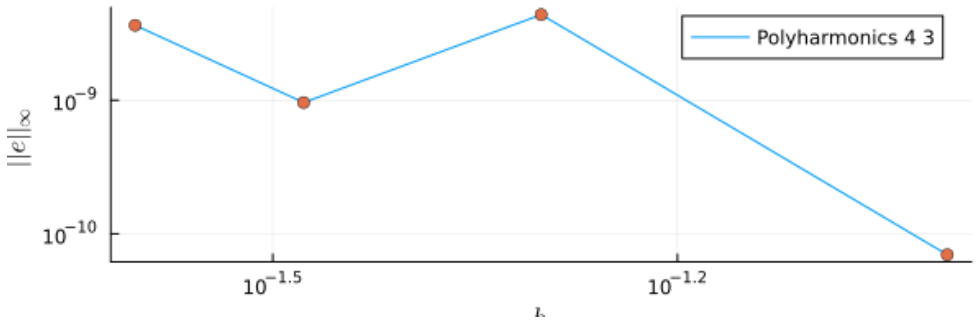

Figure 4.6: error of LHI method when the exact solution is in the space of the ansatz functions

4.4 Stable algorithm for local systems

In this work a polyharmonic kernel is used, even though there are currently no mathematical results concerning properties of divergence-free matrix kernels formed by conditionally positive-definite RBFs. The reason for this choice is the possibility to reduce the ill-conditioning of the local interpolation matrix 4.24. This is achieved by constructing a suitable preconditioner. This procedure can be thought of as a generalization of an algorithm in [22]. It should be noted however that this method only works for Stokes equation, a specific choice of ϕ and ϕ_p in 4.13 and a monomial polynomial basis.

Let

$$\Phi = \begin{bmatrix} -\partial_{22}\phi & \partial_{12}\phi & 0 \\ \partial_{21}\phi & -\partial_{11}\phi & 0 \\ 0 & 0 & \phi_p, \end{bmatrix} \tag{4.38}$$

as before. Also, let

$$\begin{aligned} \phi(r) &= \|r\|^\beta \\ \phi_p(r) &= \|r\|^{\beta-4}, \end{aligned} \tag{4.39}$$

for odd $\beta \in \mathbb{N}$. Since ϕ has to be at least in C^6 and ϕ_p has to be in C^2, the smallest valid β is 7. It can be then verified that each entry in local interpolation matrix A in 4.24 is a homogeneous function. That is, $f(ah) = a^k f(h)$ for some k. Consider a 2D stokes flow for example. For the moment assume that there is no polynomial part in the ansatz function. Suppose all centers in the local system were scaled by some factor h, as illustrated if Figure 4.7

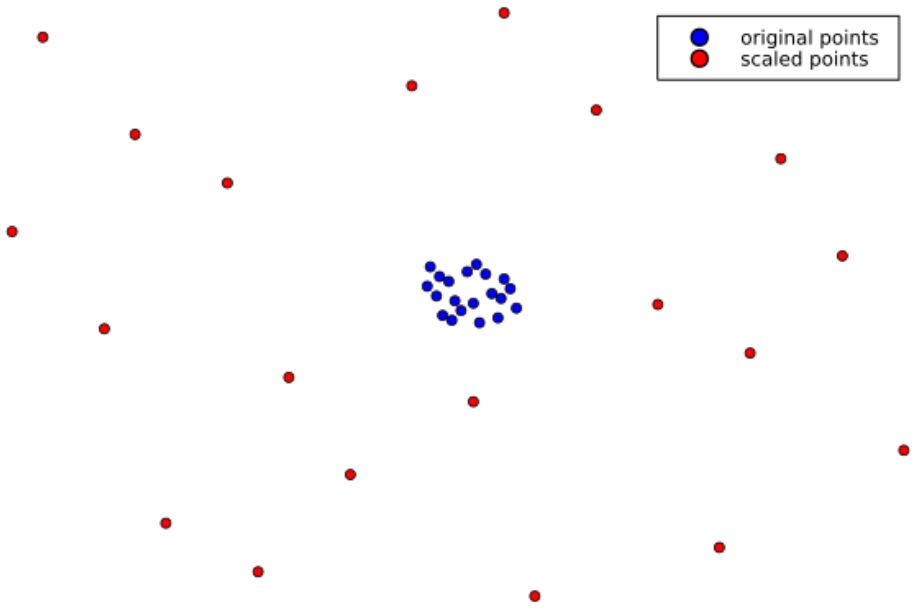

Figure 4.7: Scaled local system

Because of the homogeneity, each entry in the matrix will be multiplied by some power of h. In 4.40 these degrees are written out explicitly. Each block in the interpolation matrix has it's own degree.

$$\begin{bmatrix} h^{\beta-2} & h^{\beta-2} & h^{\beta-4} & h^{\beta-4} & h^{\beta-2} & h^{\beta-2} \\ h^{\beta-2} & h^{\beta-2} & h^{\beta-4} & h^{\beta-4} & h^{\beta-2} & h^{\beta-2} \\ h^{\beta-4} & h^{\beta-4} & h^{\beta-6} & h^{\beta-6} & h^{\beta-4} & h^{\beta-4} \\ h^{\beta-4} & h^{\beta-4} & h^{\beta-6} & h^{\beta-6} & h^{\beta-4} & h^{\beta-4} \\ h^{\beta-2} & h^{\beta-2} & h^{\beta-4} & h^{\beta-4} & h^{\beta-2} & h^{\beta-2} \\ h^{\beta-2} & h^{\beta-2} & h^{\beta-4} & h^{\beta-4} & h^{\beta-2} & h^{\beta-2} \end{bmatrix} \tag{4.40}$$

Let A_h denote an interpolation matrix for scaled (blown up) local points. Looking at 4.40 it can be observed that $A_h = DAD$ where

$$D = \begin{bmatrix} h^{\frac{\beta-2}{2}} & & & & & \\ & h^{\frac{\beta-2}{2}} & & & & \\ & & h^{\frac{\beta-6}{2}} & & & \\ & & & h^{\frac{\beta-6}{2}} & & \\ & & & & h^{\frac{\beta-2}{2}} & \\ & & & & & h^{\frac{\beta-2}{2}} \end{bmatrix}. \tag{4.41}$$

As discussed in Section 2.4, the condition number depends on the separation distance, so the condition number of A_h of the scaled system is lower. The inverse of A is found as follows:

$$A^{-1} = (D^{-1}A_hD^{-1})^{-1} = DA_h^{-1}D. \tag{4.42}$$

Since D is diagonal, its inverse is trivial. The open question is how to optimally choose the scaling factor h. It might seem like the larger the separation distance, the lower the condition number becomes, but the bounds in section 2.4 are derived for the lower eigenvalue, and do not take into account the highest eigenvalue. In practice it has been found that the condition number is the lowest when the system is scaled up such that the maximum distance between centers is on the order of 1.

If a polynomial basis is included, matrix D has to be extended to cover the blocks λP_k, αP_k and βP_k. The extended matrix D_p is also diagonal. This is only the case for the monomial basis. Figure 4.8 illustrates the effect of scaling on a condition number for a local system with equidistant nodes on a rectangular grid. For this numerical experiment $\phi = 7$, $\phi_p = 3$ and a polynomial basis with $m = 4m_2 = 2$ is used.

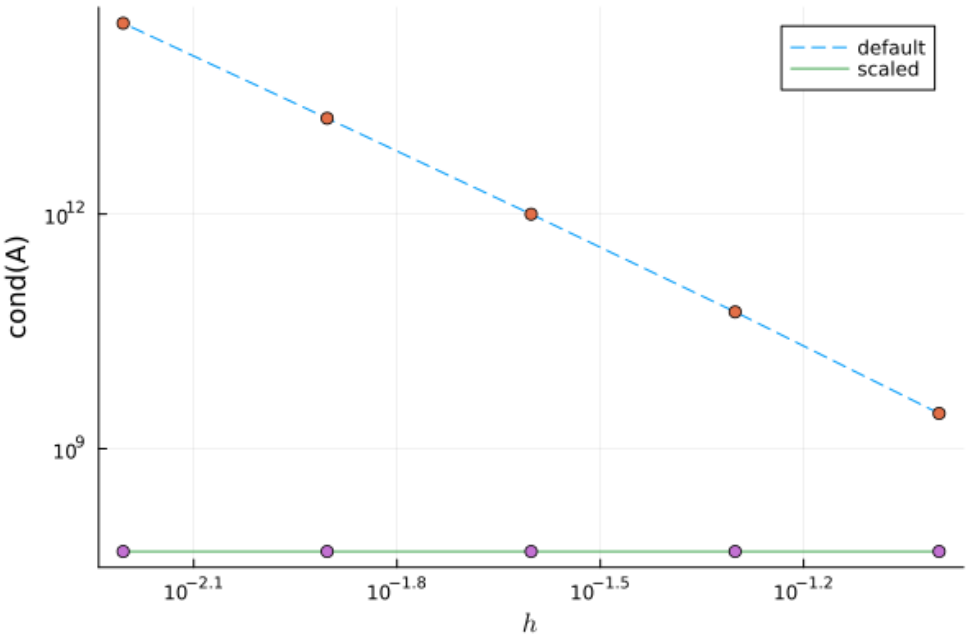

Figure 4.8: Condition number of the default local interpolation matrix vs preconditioned matrix

4.5 LHI-PU

Although the LHI method scales as $O(N)$ with N being the total number of nodes, the algorithmic complexity with respect to the stencil size is $O(M^3)$. Of course, for regular arrangements of points, the stencil weights can be computed once and used throughout the domain, but for arbitrary scattered nodes local matrix construction and inversion dominate in the run time of the algorithm. The idea of this new method is to use the same interpolant, 4.23, as in LHI method, but to evaluate it at multiple solution centers, thereby reducing the total number of local systems considerably. LHI-PU is conceptually identical to [23], the difference being the form of the local system: LHI instead of symmetric RBF finite difference.

Let the PDE domain Ω be partitioned into a set of overlapping subdomains Ω^k $k = 1, ..N_\Omega$ such that $\Omega \subset \bigcup_{k=1}^{N_\Omega} \Omega^k$ Define Solution, PDE and Boundary centers X_I,X_L and X_B the same way as in LHI method. Note that now Ω^k are placed independently from the solution centers. Point sets X_I^k,X_L^k,X_B^k as well as index sets $\mathcal{I}^k$,$\mathcal{L}^k$,$\mathcal{B}^k$ are the same as before. Figure 4.9 illustrates the partition of the domain.

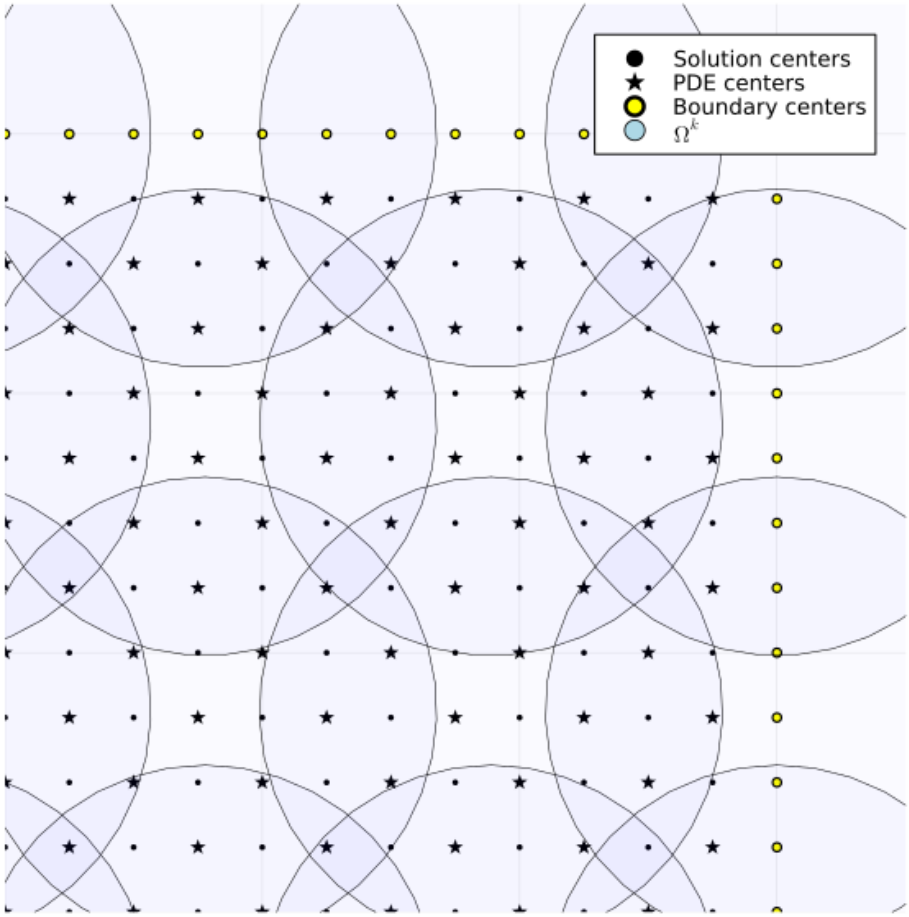

Figure 4.9: The LHI-PU domain partition

Each subdomain Ω^k is associated with a local interpolant s^k, which is exactly the same as in 4.23. The definition of the functionals in $\boldsymbol{s}^k$ are also identical to the LHI method 4.22. Each subdomain Ω^k admits a local interpolation matrix $\boldsymbol{A}^k$, and an evaluation matrix $\boldsymbol{B}^k$. Matrix $\boldsymbol{A}^k$ is the same as in equation 4.24. Matrix $\boldsymbol{B}^k$ is different since the differential operator applied to the interpolant is evaluated at all solution centers belonging to the subdomain Ω^k. New dimensions of $\boldsymbol{B}^k$ are $dN_I^k \times N_{\text{total}}$ instead of $d \times N_{\text{total}}$ for the LHI method. $N_I^k = \#X_I^k$ is the number of solution centers in Ω^k. $N_{\text{total}} = \#X_I^k + \#X_B^k + \#X_L^k$ is the total number of centers in Ω^k.Once matrices $\boldsymbol{A}^k$ and $\boldsymbol{B}^k$, matrix $\boldsymbol{C}^k$ is computed identically to the LHI method. Rows of $\boldsymbol{C}^k$ are the equations used to construct the global matrix $\boldsymbol{G}$. In LHI each

$\boldsymbol{C}^k$ is used to fill d unique rows in $\boldsymbol{G}$, but now there is an ambiguity: If two subdomains Ω^{k_1} and Ω^{k_2} overlap and both of them contain the same solution centres, then $\boldsymbol{C}^{k_1}$ and $\boldsymbol{C}^{k_2}$ will update the same rows in $\boldsymbol{G}$. The question is: which weights for these shared points to take? The answer comes from the direct partition of unity method described in Chapter 2.

Each solution center $\boldsymbol{x}_I^k$ is assigned with N_Ω Shepard's weights. If $\boldsymbol{x}_I^k \notin \Omega^j$ then the corresponding weight is zero. These are stored in a Shepard's matrix $\boldsymbol{S}$

$$\boldsymbol{S}_{ij} = \begin{cases} 0 & \text{if } \boldsymbol{x}_I^i \notin \Omega^j \\ w^j(\boldsymbol{x}_I^i) & \text{if } \boldsymbol{x}_I^i \in \Omega^j \end{cases} \tag{4.43}$$

where the weighting functions w^k is compaclty supported on Ω^k and hve a partition of unity property:

$$\sum_{k=1}^{N_\Omega} w^k(\boldsymbol{x}) = \sum_{\{k:\boldsymbol{x}\in\Omega^k\}} w^k(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \Omega. \tag{4.44}$$

Any given $\boldsymbol{x}$ only belongs to a couple of Ω^k, this makes the matrix $\boldsymbol{S}$ sparse. Below an algorithm for 2D LHI-PU is presented.

Data: X_I, X_L, X_B as well as X_I^k, X_L^k, X_B^k for $k = 1, ...N_\Omega$
Result: velocity at collocation nodes
initialize empty global matrix $\boldsymbol{G}$;
initialize empty global right hand side **RHS**;
construct Shepard matrix $\boldsymbol{S}$;
for $k = 1, ..., N_\Omega$ **do**
  assemble $\boldsymbol{A}^k$;
  assemble $\boldsymbol{B}^k$;
  $\boldsymbol{C}^k = \boldsymbol{B}^k {\boldsymbol{A}^k}^{-1}$ # preconditioner may be applied at this step;
  for $i = 1, ..., N_I^k$ **do**
    $\boldsymbol{C}^k[i,:] = \boldsymbol{C}^k[i,:]\boldsymbol{S}\left[\mathcal{I}^k[i], k\right]$;
    $\boldsymbol{C}^k[i+N_I^k,:] = \boldsymbol{C}^k[i+N_I^k,:]\boldsymbol{S}\left[\mathcal{I}^k[i], k\right]$;
  end
  $\boldsymbol{G}[\text{cat}(\mathcal{I}^k, \mathcal{I}^k. + N_I), \text{cat}(\mathcal{I}^k, \mathcal{I}^k. + N_I)] \mathrel{+}= \boldsymbol{C}^k[:, 1:2N_I^k]$;
  RHS$[\text{cat}(\mathcal{I}^k, \mathcal{I}^k. + N_I)] \mathrel{+}= \boldsymbol{C}[:, 2N_I^k + 1 : \text{end}]$;
  $\boldsymbol{u} = \boldsymbol{G} \setminus$ **RHS** ;
  return $\boldsymbol{u}$;
end

Algorithm 1: pseudo code for LHI-PU in 2D

In the above pseudo code, cat means concatenation.

Two numerical experiments are conducted for both polyharmonic and Wendland RBFs polynomial basis. In the stationary setting, the radius of the subdomains contains the same number of centers, so N_Ω is increasing as the fill distance h decreases. In the other situation, the number of centers is kept fixed. A polynomial basis with $m = 4, m_2 = 2$ is used in this experiment. For the polyharmonic kernel $\phi = \|r\|^7$,$\phi_p = \|r\|^3$, the reference solution is the same as with the LHI method. For the stationary case, the radius of the subdomains is $r_{PU} = \frac{4}{\sqrt{N}-1}$ and the number of subdomains is $N_{PU} = (\frac{0.8}{r_{PU}+1})^2$. For the non-stationary case $N_{PU} = 7$ and $r_{PU} = 0.15$. The edge subdomains are scaled up to contain the same number of centers as the internal subdomains. The weight function w is chosen to be a compactly supported Wendland C^2 function:

$$w(r) = \begin{cases} (1-r)^4(4r+1) & r < 1 \\ 0 & r > 1 \end{cases} \tag{4.45}$$

The domain is as before, $[0,1] \times [0,1]$. Placements of the centers is shown in 4.10

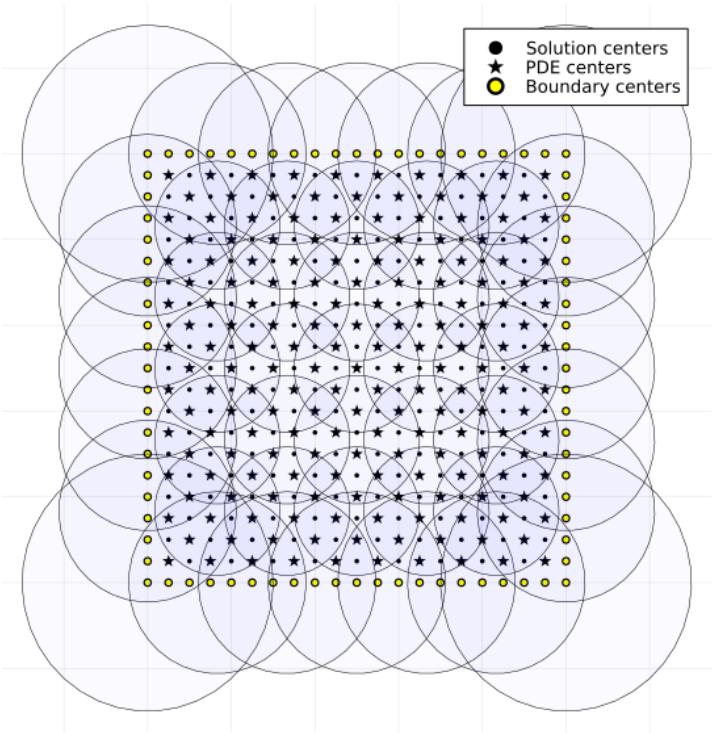

Figure 4.10: Layout of the centers and the subdomains

As can be seen from figure 4.11 the error immediately saturates for stationary interpolation for both kernels. For non-stationary interpolation, Wendland kernel shows signs of convergence. The polyharmonic kernel seems to converge at a slower rate, but then it also saturates. The data is generated using extended precision floats with 85 bits.

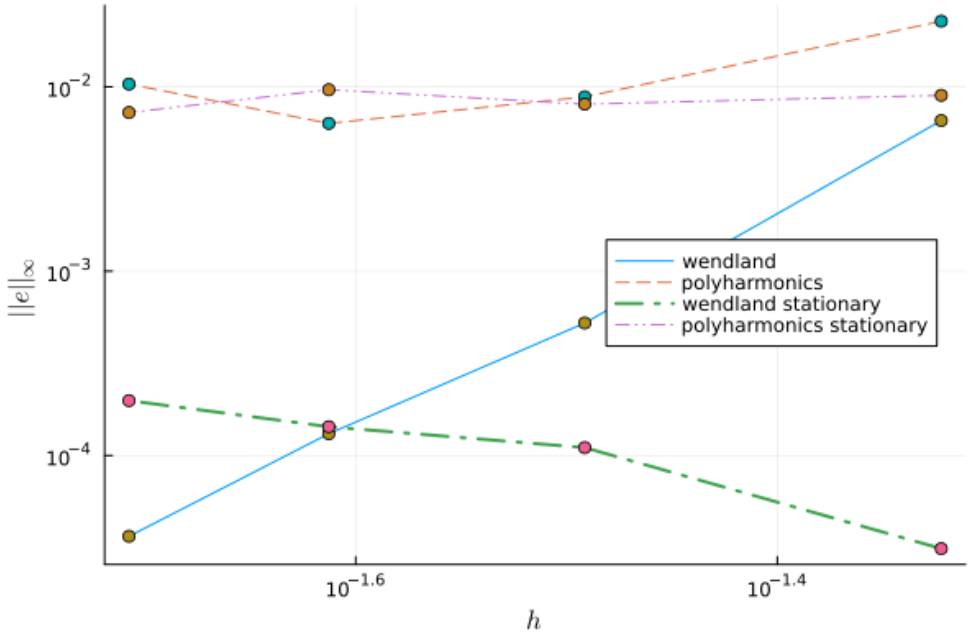

Figure 4.11: LHI-PU error vs total number of centers for stationary and fixed subdomain size

LHI-PU also has a polynomial reproduction property if the polynomial basis contains the true solution. Figure 4.12 demonstrates this property. The data is generated using the non-stationary method and a polyharmonic kernel with the same parameters as in figure 4.11. The only difference is the expanded polynomial basis with $m = 4$ and $m_2 = 3$.

The error in figure 4.12 increases with decreasing fill distance because of the effects of the condition number. The data in the second figure is generated without double precision or preconditioners.

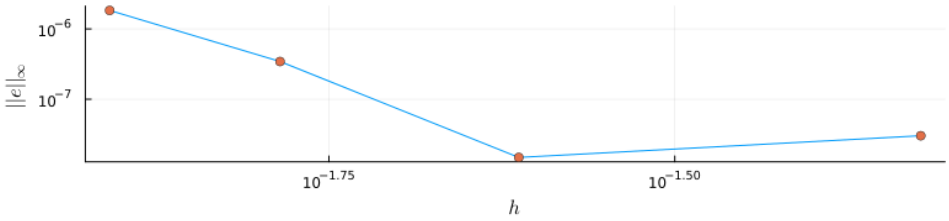

Figure 4.12: LHI-PU error vs fill distance with polynomial basis $m = 4$ and $m_2 = 3$

Chapter 5

Application of RBFs to Navier Stokes equations

In this section stationary Navier-Stokes equation in 2D is solved using matrix-valued divergence free RBfs.

The equations solved are the following:

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - \frac{\mu}{\rho}\Delta u + \frac{1}{\rho}\frac{\partial p}{\partial x} &= f_1 \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} - \frac{\mu}{\rho}\Delta v + \frac{1}{\rho}\frac{\partial p}{\partial y} &= f_2 \quad \text{on } \Omega \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u &= g_1 \\ v &= g_2 \quad \text{on } \partial\Omega \end{aligned}$$

Symmetric methods like Wendland's method described in Section 4.2 are only applicable to linear PDEs because the functionals which generate the interpolation matrix have to be linear, otherwise the divergence-free property of the interpolant is not satisfied. This problem can be tackled by linearizing the nonlinear term in the PDE and solving a sequence of linear problems, updating the functionals each iteration until the solution converges. In the case of NS equations, there are multiple possibilities. The first option to consider is how to linearize the convective term. Either u or v can be fixed for each iteration, or their partial derivatives. Another choice to consider is the construction of the interpolation matrix itself. The matrix can be symmetrical with the anzats function formed by PDE functionals, or it can

be like a Kansa method with diffeent sets of functions for the ansatz function and for the collocation of thereof. Combinatorically there are at least 4 options to consider. In this work The velocities are linearized. This choice is made because it is dominant in other numerical methods for NS equations. The linearized equations at the k^{th} iteration step are the following:

$$\begin{aligned} u_{k-1}\frac{\partial u_k}{\partial x} + v_{k-1}\frac{\partial u_k}{\partial y} - \frac{\mu}{\rho}\Delta u_k + \frac{1}{\rho}\frac{\partial p_k}{\partial x} &= f_1 \\ u_{k-1}\frac{\partial v_k}{\partial x} + v_{k-1}\frac{\partial v_k}{\partial y} - \frac{\mu}{\rho}\Delta v_k + \frac{1}{\rho}\frac{\partial p_k}{\partial y} &= f_2 \quad \text{on } \Omega \\ u_k &= g_1 \\ v_k &= g_2 \quad \text{on } \partial\Omega \end{aligned}$$

The incompressibility condition is dropped as it will be satisfied automatically by the divergence-free ansatz function. The Interpolation matrix is chosen to be non-symmetrical. This choice is made for computational and implementational reasons. Functionals that form the ansatz function are the same as in Wendland's method for Stokes equations 4.15, but the interpolation function itself is collocated using the following linearized-NS functionals

$$\begin{cases} \lambda_i(\boldsymbol{u}) = \delta_{x_i} \circ \left[u_{k-1}\partial_x(\boldsymbol{u}^T\boldsymbol{e}_1) + v_{k-1}\partial_y(\boldsymbol{u}^T\boldsymbol{e}_1) - \frac{\mu}{\rho}\Delta(\boldsymbol{u}^T\boldsymbol{e}_1) + \frac{1}{\rho}\partial_x(\boldsymbol{u}^T\boldsymbol{e}_3)\right] & i = 1, ..., N \\ \lambda_{i+N}(\boldsymbol{u}) = \delta_{x_i} \circ \left[u_{k-1}\partial_x(\boldsymbol{u}^T\boldsymbol{e}_2) + v_{k-1}\partial_y(\boldsymbol{u}^T\boldsymbol{e}_2) - \frac{\mu}{\rho}\Delta(\boldsymbol{u}^T\boldsymbol{e}_2) + \frac{1}{\rho}\partial_x(\boldsymbol{u}^T\boldsymbol{e}_3)\right] & i = 1, ..., N \\ \lambda_{i+2N}(\boldsymbol{u}) = \delta_{x_{i+N}} \circ \boldsymbol{u}^T\boldsymbol{e}_1 & i = 1, ..., M \\ \lambda_{i+2N+M}(\boldsymbol{u}) = \delta_{x_{i+N}} \circ \boldsymbol{u}^T\boldsymbol{e}_2 & i = 1, ..., M \end{cases} \tag{5.1}$$

The benefit of this approach is that the solution space doesn't change between iterations, making it easier to track the errors. Also, there is no need to recompute the interpolation matrix at each iteration. A few matrices are precomputed at the beginning and then reused at each iteration level. To construct these matrices, define the following sets of functionals:

$$\begin{aligned}
\Lambda_u &= \{\delta_{x_i} \circ \left[\boldsymbol{u}^T \boldsymbol{e}_1\right] : i = 1, ..., N\} \\
\Lambda_v &= \{\delta_{x_i} \circ \left[\boldsymbol{u}^T \boldsymbol{e}_2\right] : i = 1, ..., N\} \\
\Lambda_{\partial_x u} &= \{\delta_{x_i} \circ \left[\partial_x(\boldsymbol{u}^T \boldsymbol{e}_1)\right] : i = 1, ..., N\} \\
\Lambda_{\partial_x v} &= \{\delta_{x_i} \circ \left[\partial_x(\boldsymbol{u}^T \boldsymbol{e}_2)\right] : i = 1, ..., N\} \\
\Lambda_{\partial_y u} &= \{\delta_{x_i} \circ \left[\partial_y(\boldsymbol{u}^T \boldsymbol{e}_1)\right] : i = 1, ..., N\} \\
\Lambda_{\partial_y v} &= \{\delta_{x_i} \circ \left[\partial_y(\boldsymbol{u}^T \boldsymbol{e}_2)\right] : i = 1, ..., N\},
\end{aligned} \tag{5.2}$$

Also, define the following matrices: U, V, U_x, V_x, U_y, V_y. They are formed by applying Stokes functionals4.15 to the matrix kernel 4.16 with respect to the second variable and then applying the functionals 5.2 with respect to the first variable. If the polynomial basis is present, define matrices $Pu, Pv, P_{\partial_x u}, P_{\partial_x v}, P_{\partial_y u}, P_{\partial_y v}$. They are formed by applying 5.2 to the polynomial basis. Also let A and right hand side, $b = [\boldsymbol{f}_1 \boldsymbol{f}_2 \boldsymbol{g}_1 \boldsymbol{g}_2 0]^T$, be the same as in 4.19.

At each iteration a linear system,

$$M_k c_k = b, \tag{5.3}$$

is solved. M_k depends on the coefficients in the previous iteration,c_{k-1}. Now, the construction of M_k is discussed. First, use c_{k-1} to calculate the u and v velocities at $k-1$ iteration:

$$\boldsymbol{u}_{k-1} = [UPu]c_{k-1} \tag{5.4}$$
$$\boldsymbol{v}_{k-1} = [VPv]c_{k-1}. \tag{5.5}$$

Also, define

$$D_{u_{k-1}} = \text{diagonal}(\boldsymbol{u}_{k-1}) \tag{5.6}$$
$$D_{v_{k-1}} = \text{diagonal}(\boldsymbol{v}_{k-1}). \tag{5.7}$$

Finally, matrix M_k can be assembled as follows:

$$M_k = A + \begin{bmatrix} D_{u_{k-1}}[U_x P_{\partial_x u}] + D_{v_{k-1}}[U_y P_{\partial_y u}] \\ D_{u_{k-1}}[V_x P_{\partial_x v}] + D_{v_{k-1}}[V_y P_{\partial_y v}] \\ 0 \end{bmatrix}. \tag{5.8}$$

Notice, that the part of Matrix M corresponding to the boundary conditions

doesn't change. This is only the case for the Dirichlet boundary conditions. The method is tested for lid-driven cavity against a reference solution from the literature [24]. Polyharmonic kernel with $\phi_p = \phi = \|r\|^7$ with polynomial basis $m = 4$, $n = 2$ is used. The centers are placed in a regular Cartesian grid, as with the Stokes equations from previous chapters.

Figure 5.1 shows the L^2 error between consecutive solutions c_k and c_{k-1} for each iteration number from 1 to 100 for a fill distance $\frac{1}{20}$ at Re $= 400$. Note that the error in figure 5.1 is not normalized with respect to the number of points. That is why the saturated error is quite large.

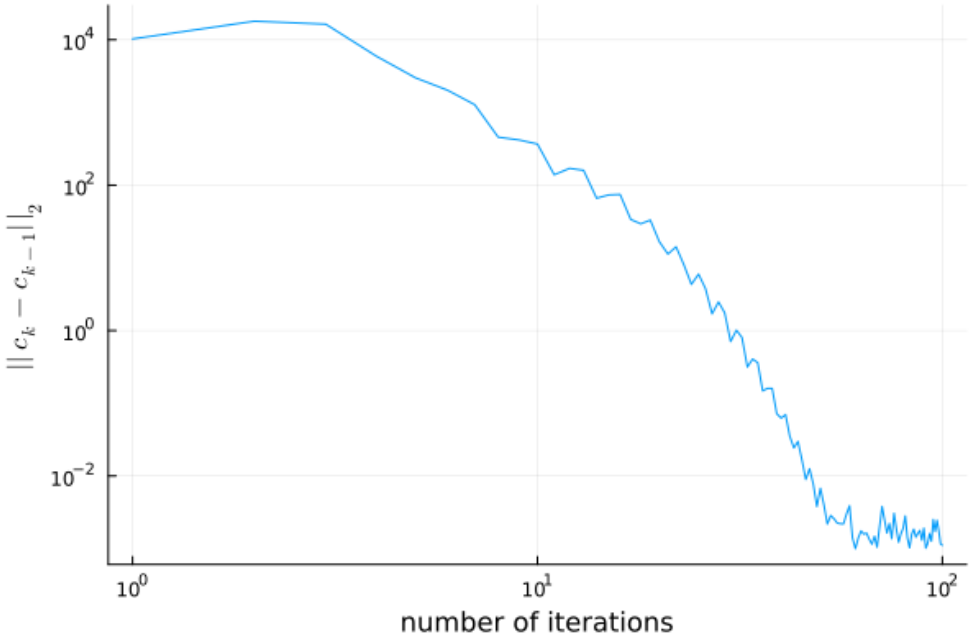

Figure 5.1: Error between consecutive iterations

Figure 5.2 shows a horizontal velocity profile along a vertical center line at $Re = 400$ for different fill distances

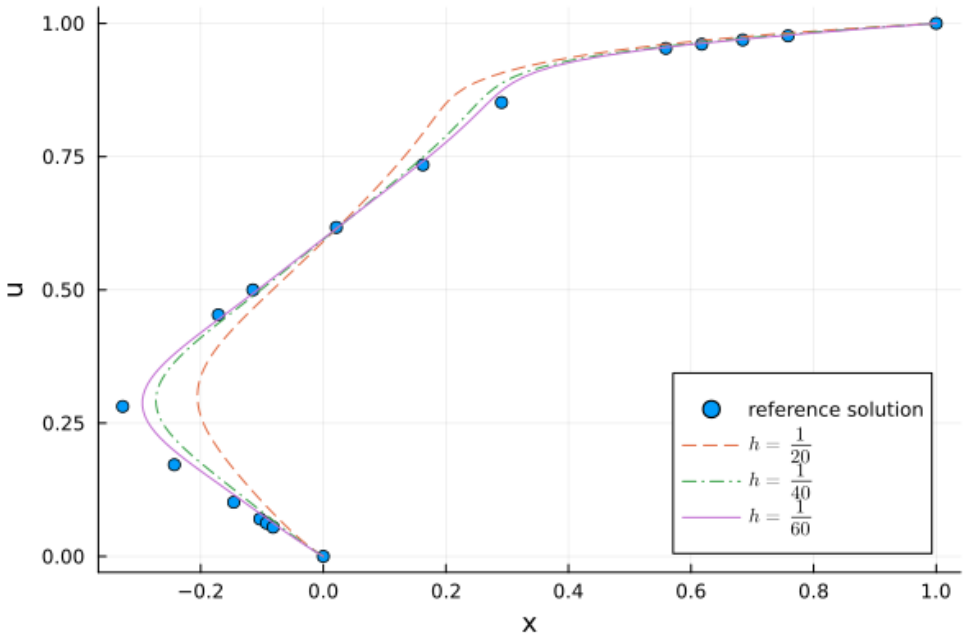

Figure 5.2: The u velocity along the vertical center line for different separation distances at Re = 400.

As can be seen, the RBF solution approaches the reference solution as the fill distance decreases. The same experiment is performed for Re $= 1000$. The velocity profile is plotted in figure 5.3

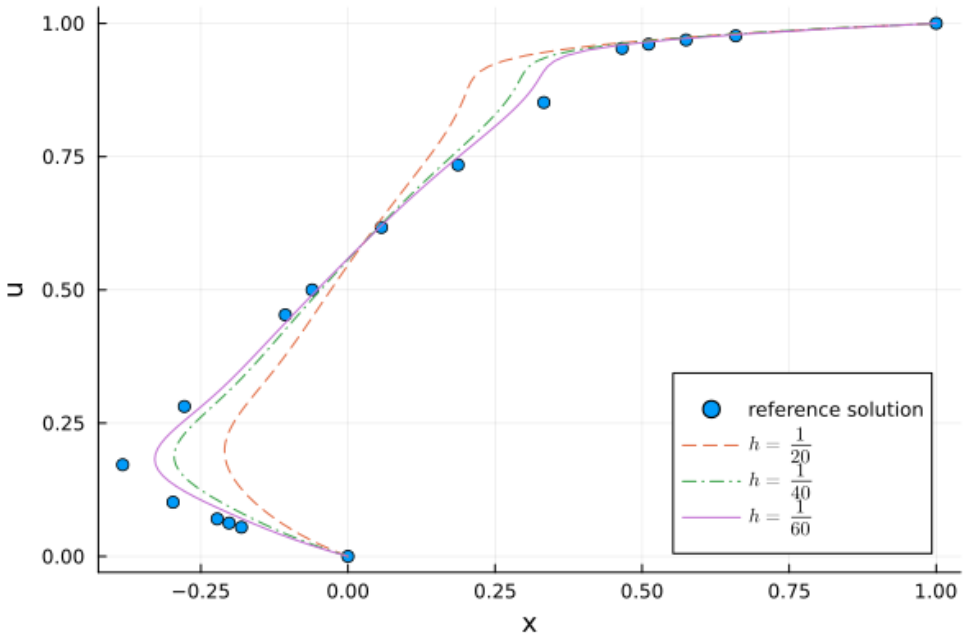

Figure 5.3: The u velocity along the vertical center line for different separation distances at Re = 1000.

The velocity profile in figure 5.3 also converges to the reference solution, but slower. With increasing Reynolds number the spatial resolution has to increase as well to capture all details of the flow field.

Chapter 6

Conclusion and further research

In this thesis, four new contributions to the field of RBF-based methods for PDEs were made. First, the utilization of polyharmonic RBF in matrix-valued divergence-free kernel with divergence-free polynomial basis. Second, a preconditioner for local systems in LHI method. This preconditioner exploits the homogeneity of the polyharmonic kernel and the relation between separation distance and the condition number. Third, a partition of unity approach for assembling a global matrix was demonstrated in the context of stationary Stokes equations. Lastly, a global method for solving stationary stationary NS equations with divergence-free matrix kernel.

There are many more ways to expand the present work. First, there needs to be a mathematical analysis of the interpolation error of matrix-valued kernels formed by conditionally positive definite RBFs. Second, there is a possibility to develop an algorithm for time-dependent problems. Currently, there are time-first discretizations which require recomputation of the local linear systems at each time level [9]. Such schemes are computationally expensive and are not compatible with the scaling preconditioner derived in this work. Alternatively, there are ODE-based methods based on Helmholtz-Hodge decomposition [25],[26], but those methods have a limited range of boundary conditions and are not always stable. Another unsolved problem to consider is a localized method for Navier-Stokes equations. Adapting LHI or LHI-PU methods to work with nonlinear equations is not a trivial task. Another aspect to consider is the solution method of discretized non-linear equations. In this work Picard linearization was used. While it is stable and easy to implement, it requires many iterations to converge. It might be cheaper

to solve the nonlinear system with a Newton method instead. Finally, more numerical experiments have to be performed with polyharmonic kernel and polynomial reproduction for both LHI and LHI-PU methods. More boundary conditions and domain shapes can be tested, as well as convergence for smaller fill distances. Numerical experiments in 3D should also be considered for future research.

Appendix A

Entries of Wendland interpolation matrix

$$
\begin{aligned}
[A_{11}]_{ij} &= -\nu^2 \partial_{22} \Delta\Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) - \partial_{11}\phi_p(\boldsymbol{x}_i - \boldsymbol{x}_j) && \text{for } i = 1...N \; j = 1...N \\
[A_{12}]_{ij} &= \nu^2 \partial_{12} \Delta\Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) - \partial_{12}\phi_p(\boldsymbol{x}_i - \boldsymbol{x}_j) && \text{for } i = 1...N \; j = 1...N \\
[A_{13}]_{ij} &= \nu \partial_{22} \Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_{j+N}) && \text{for } i = 1...N \; j = 1...M \\
[A_{14}]_{ij} &= -\nu \partial_{12} \Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_{j+N}) && \text{for } i = 1...N \; j = 1...M \\
[A_{22}]_{ij} &= -\nu^2 \partial_{11} \Delta\Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_j) - \partial_{22}\phi_p(\boldsymbol{x}_i - \boldsymbol{x}_j) && \text{for } i = 1...N \; j = 1...N \\
[A_{23}]_{ij} &= -\nu \partial_{12} \Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_{j+N}) && \text{for } i = 1...N \; j = 1...M \\
[A_{24}]_{ij} &= \nu \partial_{11} \Delta\phi(\boldsymbol{x}_i - \boldsymbol{x}_{j+N}) && \text{for } i = 1...N \; j = 1...M \\
[A_{33}]_{ij} &= -\partial_{22}\phi(\boldsymbol{x}_{i+N} - \boldsymbol{x}_{j+N}) && \text{for } i = 1...M \; j = 1...M \\
[A_{34}]_{ij} &= \partial_{12}\phi(\boldsymbol{x}_{i+N} - \boldsymbol{x}_{j+N}) && \text{for } i = 1...M \; j = 1...M \\
[A_{44}]_{ij} &= -\partial_{11}\phi(\boldsymbol{x}_{i+N} - \boldsymbol{x}_{j+N}) && \text{for } i = 1...M \; j = 1...M
\end{aligned}
$$

References

[1] C. Chen, Y. Li, N. Zhao, B. Guo, and N. Mou, "Least squares compactly supported radial basis function for digital terrain model interpolation from airborne lidar point clouds," *Remote Sensing*, vol. 10, no. 4, 2018. doi: 10.3390/rs10040587. [Online]. Available: https://www.mdpi.com/2072-4292/10/4/587 [Page 1.]

[2] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer New York, 1998. ISBN 147572442X [Page 2.]

[3] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," pp. 1171–1220, 6 2008. [Page 2.]

[4] E. J. Kansa, "Multiquadrics-a scattered data approximation scheme with applications to computational fluid-dynamics-ii," pp. 147–161, 1990. [Pages 2 and 12.]

[5] A. Tolstykh and D. Shirobokov, "On using radial basis functions in a "finite difference mode" with applications to elasticity problems," *Computational Mechanics*, vol. 33, pp. 68–79, 12 2003. doi: 10.1007/s00466-003-0501-9 [Pages 2 and 16.]

[6] D. Mirzaei, "The direct radial basis function partition of unity (d-rbf-pu) method for solving pdes," 9 2020. [Online]. Available: http://arxiv.org/abs/2009.07175 [Pages 2, 3, and 18.]

[7] D. Stevens, H. Power, and H. Morvan, *An Order-N Complexity Meshless Algorithm Based on Local Hermitian Interpolation*, 11 2008, vol. 33, pp. 99–124. ISBN 978-1-4020-8820-9 [Pages 2 and 30.]

[8] F. J. Narcowich and J. D. Ward, "Generalized hermite interpolation via matrix-valued conditionally positive definite functions," pp. 661–687, 1994. [Pages 2, 8, and 22.]

[9] "Rbf collocation and hybrid-lhi methods for stokes systems and its application to controllability problems," *Computational and Applied Mathematics*, vol. 40, 2 2021. [Pages 3, 21, 30, and 51.]

[10] G. E. Fasshauer, *Meshfree approximation methods with MATLAB*. World Scientific, 2007. ISBN 9789812706331 [Pages 4, 6, and 10.]

[11] H. Wendland, *Scattered Data Approximation*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. [Pages 5, 6, 8, 10, and 15.]

[12] ——, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," pp. 389–396, 1995. [Page 15.]

[13] B. Fornberg, G. Wright, and E. Larsson, "Some observations regarding interpolants in the limit of flat radial basis functions," *Computers Mathematics with Applications*, vol. 47, no. 1, pp. 37–55, 2004. doi: https://doi.org/10.1016/S0898-1221(04)90004-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122104900041 [Page 16.]

[14] E. Larsson and B. Fornberg, "Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions," *Computers Mathematics with Applications*, vol. 49, no. 1, pp. 103–130, 2005. doi: https://doi.org/10.1016/j.camwa.2005.01.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122105000118 [Page 16.]

[15] M. D. Buhmann, S. Dinew, and E. Larsson, "A note on radial basis function interpolant limits," *IMA Journal of Numerical Analysis*, vol. 30, no. 2, pp. 543–554, 02 2009. doi: 10.1093/imanum/drn051. [Online]. Available: https://doi.org/10.1093/imanum/drn051 [Page 16.]

[16] M. A. Schweitzer, *Partition of Unity Method*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–22. ISBN 978-3-642-59325-3. [Online]. Available: https://doi.org/10.1007/978-3-642-59325-3_2 [Page 18.]

[17] S. Arefian and D. Mirzaei, "A compact radial basis function partition of unity method," *Computers and Mathematics with Applications*, vol. 127, pp. 1–11, 12 2022. doi: 10.1016/j.camwa.2022.09.029 [Page 18.]

[18] P. K. Kundu, I. M. Cohen, and D. R. Dowling, "Chapter 4 - conservation laws," in *Fluid Mechanics (Sixth Edition)*, sixth edition ed., P. K. Kundu, I. M. Cohen, and D. R. Dowling, Eds. Boston: Academic Press, 2016, pp. 109–193. ISBN 978-0-12-405935-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124059351000046 [Page 20.]

[19] H. Wendland, "Divergence-free kernel methods for approximating the stokes problem," *SIAM Journal on Numerical Analysis*, vol. 47, pp. 3158–3179, 1 2009. doi: 10.1137/080730299 [Pages 22, 24, 27, 28, and 35.]

[20] D. Stevens, H. Power, M. Lees, and H. Morvan, "The use of pde centres in the local rbf hermitian method for 3d convective-diffusion problems," *Journal of Computational Physics*, vol. 228, pp. 4606–4624, 7 2009. doi: 10.1016/j.jcp.2009.03.025 [Page 30.]

[21] ——, "A local hermitian rbf meshless numerical method for the solution of multi-zone problems," *Numerical Methods for Partial Differential Equations*, vol. 27, pp. 1201–1230, 9 2011. doi: 10.1002/num.20577 [Page 30.]

[22] A. Iske, "On the approximation order and numerical stability of local lagrange interpolation by polyharmonic splines," vol. 145, 01 2002. doi: 10.1007/978-3-0348-8067-1_8 [*Page* 36.]

[23] S. Arefian and D. Mirzaei, "A compact radial basis function partition of unity method," *Computers and Mathematics with Applications*, vol. 127, pp. 1–11, 12 2022. doi: 10.1016/j.camwa.2022.09.029 [Page 39.]

[24] U. Ghia, K. N. Ghia, and C. T. Shin, "High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method*," pp. 387–411, 1982. [Page 48.]

[25] E. J. Fuselier, V. Shankar, and G. B. Wright, "A high-order radial basis function (rbf) leray projection method for the solution of the incompressible unsteady stokes equations," *Computers and Fluids*, vol. 128, pp. 41–52, 4 2016. doi: 10.1016/j.compfluid.2016.01.009 [Page 51.]

[26] C. Keim and H. Wendland, "A high-order, analytically divergence-free approximation method for the time-dependent stokes problem," *SIAM Journal on Numerical Analysis*, vol. 54, pp. 1288–1312, 2016. doi: 10.1137/151006196 [Page 51.]