UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXVI CICLO DEL DOTTORATO DI RICERCA IN

INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

STABLE MESHLESS METHODS FOR 3D CFD SIMULATION ON COMPLEX GEOMETRIES BASED ON RADIAL BASIS FUNCTION

Settore scientifico-disciplinare: **ING-IND/10**

DOTTORANDO / A
MIOTTI DAVIDE

COORDINATORE
PROF. FULVIO BABICH

SUPERVISORE DI TESI

PROF. ENRICO NOBILE

ANNO ACCADEMICO 2022/2023

Summary

List of Figures

List of Tables

Chapter 1

Introduction

This work deals with the development and applications of Radial Basis Functions (RBF) to the solution of Partial Differential Equations (PDEs) involving fluid flow and heat transfer. More specifically, the main focus is the study of those stability and accuracy issues arising in presence of Neumann boundary conditions and the consequent development of stabilization techniques.

The algorithms presented fall in the broad category of meshless solvers, which are aimed at solving PDEs without relying on the mesh data structure. In order to approximate the solution to a given boundary value problem, these algorithms usually rely on a set of nodes which are scattered on the computational domain with no connectivity information. The two approaches investigated in the present thesis are called Radial Basis Function-Finite Difference (RBF-FD) and Radial Basis Function-Hermite Finite Difference (RBF-HFD). The lack of connectivity information and the reliance on radial functions allow them to provide better geometrical flexibility than traditional mesh-based methods, at the same time, they also enable finite difference-like discretization of differential operators. The interest in meshless methods is motivated by the fact that, as new Computer Aided Engineering (CAE) techniques are developed, it seems that a major obstacle to their greater diffusion is constituted by the intrinsic limitations of the mesh generation. This happens, for instance, in applications requiring automatic shape optimization, where the domain of calculus is subject to extensive deformations requiring frequent remeshing and quality assessments. Similar problems also arise in presence of moving boundaries or multi-phase simulations, furthermore, especially in the field of CFD, highly valuable and experienced operators are often kept busy by activities related to mesh generation for extended periods of time.

The starting point of the research activity presented below was the work done at the University of Trieste by Riccardo Zamolo and Enrico Nobile [117]: a clear vision of how to implement the first solver for generic 3D geometries was already established, along with ideas on possible further improvements. However, after some initial success in the solution of heat conduction problems on complex geometries [73], it soon became clear that some stability issues

had to be addressed when dealing with more complex physics. The main topic of research has then become the stability and accuracy of the Radial Basis Function-Finite Difference method in presence of Neumann boundary conditions (BC). This is of critical importance in many cases, for example in the solution of incompressible flows where the projection scheme for pressure correction is adopted and Neumann BC are enforced in the associated elliptic equation.

1.1 Outline of the work

The remainder of chapter 1 provides a motivation for the development of a new numerical meshless solver based on Radial Basis Functions (RBF) and also outlines the numerical procedure in order to provide some context for the subsequent discussion.
Chapter 2 reviews the theory of RBF-based scattered data interpolation, this is the theoretical foundation for the following chapters. The theory of interpolation can be adapted with minimal changes to the solution of boundary value problems involving partial differential equations (PDEs) where many theoretical results remain true.
Chapter 3 discusses the process of node generation, which is adopted in meshless methods instead of the usual mesh generation. This chapter also provides an overview of the main features of the algorithm adopted by the author.
Chapter 4 presents the Radial Basis Function-Finite Difference (RBF-FD) algorithm for the solution of PDEs and presents some results in order to justify the analysis of ill-conditioning issues related to the presence of Neumann boundary conditions.
Chapter 5 discusses in detail the topic of ill-conditioning induced by boundary conditions and proposes two approaches for their solution within the RBF-FD method. This is a review of the unpublished work [122] on the same topic. In order to split an otherwise very long discussion, Appendices A.1, A.2 and A.3 are used to complement the discussion on the stabilization procedures. Stabilization approaches discussed in chapter 5 are put at the test in chapter 6, where the resulting stabilized RBF-FD method is applied to the solution of two benchmark problems involving natural convection in 3D. The unpublished work [120] is reported almost unchanged in this chapter, appendices B.1 and B.2 contain further visualizations of the results.
Chapter 7 provides an overall presentation of the theory of Generalized Hermite Interpolation, which can be understood as a generalization of the Scattered Data Interpolation discussed in chapter 2 and presents two algorithms resulting from the application of this theory to the RBF-FD method. The first, called Minimal RBF-HFD is an alternative for the stabilization approaches discussed in chapter 5, while the second, called Compact RBF-HFD is ment to improve the spatial resolution of Minimal RBF-FD for specific applications. The assessment of the accuracy of Compact RBF-FD is a review of the unpublished work [1].
Finally, conclusions are drawn in chapter 8.

1.2 Radial Basis Functions

Over time a growing activity of research has been focused on the development of new numerical methods based on RBFs and has produced remarkable results and a voluminous literature. Since the '90s a good uniformity in notation and terminology was adopted by many authors and few monographs published in recent times provide a good overall picture of the theoretical background and report some major results. [35, 12, 66, 27]

A good starting point might be to give an idea of what a radial basis function is: given a point $\boldsymbol{x}_1$, which is known, the RBF associated to $\boldsymbol{x}_1$ can be written as

$$\varphi(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2)$$

where $\|\cdot\|_2$ is the euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}_1$. If $\boldsymbol{x} \in \mathbb{R}^d$, then $\varphi(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2)$ is to be intended as a scalar field defined on $\mathbb{R}^d$, which take the same value at any point $\boldsymbol{x}$ lying at given distance from $\boldsymbol{x}_1$, thus satisfying a radial symmetry in $\mathbb{R}^d$, hence the term *radial*.

Once a set of (distinct) nodes $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ contained in a domain $\Omega \subset \mathbb{R}^d$ is given, the set of radial functions $\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ associated to each node of $\mathcal{X}$ form a basis for the following space of functions, hence the term *basis*.

$$F_\Phi := \left\{ \sum_{i=1}^{N} \alpha_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2), \quad \alpha_i \in \mathbb{R},\ \boldsymbol{x}_i \in \mathcal{X} \right\}$$

1.3 Mesh-based methods

The research in the field of Computational Fluid Dynamics (CFD) dates back to the early '50s but the investigation of new numerical methods for the solution of Navier-Stokes equations still motivates a prolific community. Indeed, many approaches have proven successful over time, the most established ones being the Finite Volume Method (FVM) [107, 21] and the Finite Element Method (FEM) [124, 5], such methods have been adopted throughout the academy and industry. While there are many differences between the two, a common feature is the presence of a mesh, i.e. a data structure which describes how the domain of calculus is divided into simpler interconnected cells. In practice the mesh is always generated with the help of a computer program once the shape of the computational domain has been defined and it is fed to the program in charge of performing the actual solution of the equations.

1.3.1 Integration with CAE software

As the availability of computational resources increases, numerical methods responsible for the solution of the partial differential equations, the solvers, are integrated in larger software packages along with other design tools such as 3D CAD (Computer Aided Design) and optimization algorithms. Such packages are usually generically labeled as CAE (Computer Aided Engineering) software and

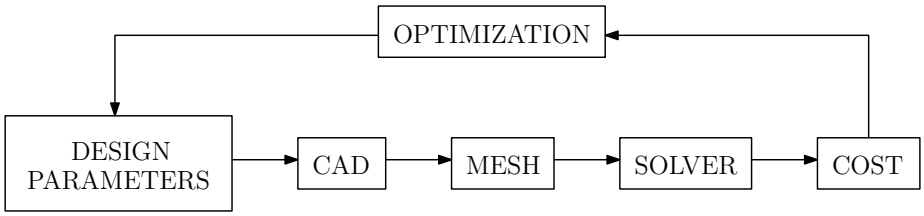


Figure 1.1: Block diagram describing an optimization process: an initial set of design parameters is modified in order to minimize a cost function calculated from the output of a numerical solver.

their goal is to aggregate every software tool required for a complete assessment of the performance of a given design and for its optimization. In other words, CAE software help the user in managing complex design processes, such as the one described by the block diagram of Figure 1.1: a set of design parameters is fed to a parametric CAD which is used to generate a geometry, this in turn is discretized by a mesh generator, a solver is then used to perform some simulation and a cost function is finally calculated from the solver's output. The loop is then closed with an optimization algorithm that modifies the design parameters in order to minimize the cost function.

In this process the physical simulation performed by the solver is probably the most important step, not only it needs to provide reliable results, but its level of flexibility and integration with the other software packages determines the efficiency of the overall design process and to which extent it can be automated. Therefore, in developing modern solvers the objective can no longer be limited to the maximization of computational efficiency or accuracy: its capability of integration within a CAE environment should be regarded as equally important. Furthermore, the design of CAE software is increasingly focusing on the so called democratization. That is the simplification of the user experience and the minimization of compulsory input parameters, in order to make advanced engineering tools accessible even to less experienced practitioners. In other words, ideal simulation tools should require as little human supervision as possible and still provide accurate and consistent results even as the shape of the computational domain is modified by an optimization algorithm within a CAE environment.

1.3.2 Mesh-related issues

Most CAE software packages are developed around preexisting solvers, typically based on FEMs and FVMs. While both methods provide a satisfactory level of accuracy and computational efficiency for most applications, their flexibility and the resulting real-life performance is limited by the reliance on a mesh. The mesh generation process could theoretically be carried out automatically by some specific software, but this is not always recommended and a lot of human intervention is usually required in order to achieve satisfactory results

[66], especially when higher accuracy is sought. Furthermore, the adoption of a mesh present some important weaknesses [66], two of them are:

1. the cost of the mesh creation,
2. the inability to allow large (geometric) deformations of the domain, since this may lead to an unacceptable decrease of the quality of the mesh.

The first one refers to both computational and economical costs, indeed operator costs now outweigh the cost of CPU time for the computer [66]. A well trained operator with some knowledge on the physics of the problem will produce a high quality mesh by manually controlling the level of refinement in different parts of the domain and the subsequent simulation will achieve high computational efficiency since no unnecessary data are stored or processed by the solver. Such a workflow often requires lower computational resources and provides the best end results but is often very expensive in economical terms. An automatic mesh generator, on the other side, does not require much human intervention but its employment will most likely result in either lower computational efficiency, in the case of a mesh that is highly refined everywhere, or in lower accuracy, in the case of a mesh that is not adequately refined at some specific location. This can be compensated only partially with the adoption of Automatic Mesh Refinement (AMR) tools, which are capable of performing refinement or coarsening where needed. The second weakness becomes of interest in many practical cases, for example when the solver is paired with some optimization algorithm, like in Figure 1.1, and the geometry is changed many times in a closed loop. At each step the mesh is degraded in quality and even when morphing algorithms are employed, the solution usually becomes less reliable at every iteration. In most cases a periodic complete re-meshing of the domain becomes necessary [66].

In order to address the issues enumerated above, a large research community is increasingly focused on the development of mesh-free or meshless methods, in this category fall the ones discussed in this thesis. While the underlying theory has reached a certain maturity and most inherent shortcomings of the main meshless algorithms have been addressed, their widespread adoption by the industry is still to come.

1.4 Why the Radial Basis Functions

1.4.1 Node clouds instead of a mesh

In meshless solvers the mesh data structure is usually replaced by a set of nodes (or node cloud) generated within the domain of calculus and over its boundary. The generation of a node cloud in the domain is especially advantageous with respect to a mesh generation when it allows to avoid any storage of connectivity information, i.e. information on the mutual relation between nodes. In other words, the nodes scattered must satisfy only minimal requirements, such as variable spatial resolution and adequate filling of the boundary, but do not need to form any local polygons or polyhedrons, this in turns allows greater geometrical

flexibility, memory efficiency and parallel execution. The only information stored once the node generation process is finished is a set of nodes' coordinates, in order to avoid the introduction of further structure in the data, it is therefore advantageous to adopt a collocation approach towards the solution of PDEs. This means that the the governing equations are enforced locally at the nodes and, for efficiency requirements (i.e. in order to ensure some sparsity), also the differential operators are discretized using only local information available in the neigborhood of each node. Theoretically, this approach is justified since any derivative is a local property of a function and therefore it should be an optimal choice to rely on spatially localized approximations [36]. The resulting freedom in the node placement is balanced by a much more constrained choice of the discretization scheme: it must take into account the lack of any rigid structure in the node placement. Since no strict requirement was made on the relative position of the nodes it is not possible to assume that they are placed on a grid of any kind and therefore no out-of-the-box Finite Difference (FD) scheme can be used.

The selection of a suitable discretization scheme therefore requires some extra care, luckily such a scheme can be derived from the well developed theory of scattered data interpolation. Here follows a brief presentation on how this theory can be applied to the discretization of linear differential operators in a simple case.

1.4.2 Solution from scattered data interpolation

Suppose that we have generated N nodes $\boldsymbol{x}_j$, $j = 1, \dots, N$ inside the domain $\Omega \in \mathbb{R}^d$, and we look for a solution to the following Dirichlet boundary value problem in strong form:

$$\begin{cases} \mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}) & in\ \Omega \\ u(\boldsymbol{x}) = g(\boldsymbol{x}) & on\ \partial\Omega \end{cases} \tag{1.1}$$

$\mathcal{L}$ being a linear differential operator, $b(\boldsymbol{x})$ and $g(\boldsymbol{x})$ two known functions.

The basic idea is to assume that the real valued solution $u(\boldsymbol{x})$ of the problem can be approximated by a function $u^h(\boldsymbol{x})$ defined as a linear combination of suitable basis functions $B_k(\boldsymbol{x})$ [27]:

$$u^h(\boldsymbol{x}) = \sum_{k=1}^{N} \alpha_k B_k(\boldsymbol{x}) \tag{1.2}$$

Because of the linearity of (1.2), we can therefore write:

$$\mathcal{L}u^h(\boldsymbol{x}) = \sum_{k=1}^{N} \alpha_k \mathcal{L}B_k(\boldsymbol{x}) \tag{1.3}$$

The theory on scattered data interpolation tells us that coefficients α_k can be uniquely determined by enforcing an equal number of conditions:

$$u^h(\boldsymbol{x}_j) = u(\boldsymbol{x}_j) \quad j = 1 \dots N \tag{1.4}$$

or equivalently, in matrix form and substituting (1.2) into (1.4):

$$\underbrace{\begin{pmatrix} B_1(\boldsymbol{x}_1) & \dots & B_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ B_1(\boldsymbol{x}_N) & \dots & B_N(\boldsymbol{x}_N) \end{pmatrix}}_{\boldsymbol{B}} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} u(\boldsymbol{x}_1) \\ \vdots \\ u(\boldsymbol{x}_N) \end{pmatrix} \tag{1.5}$$

The solution to problem (1.1) is obtained instead by enforcing the following collocation conditions:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_j) &= f(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \Omega \\ u^h(\boldsymbol{x}_j) &= g(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \partial\Omega \end{aligned} \tag{1.6}$$

Suppose that the nodes were numbered so that the first N_I are internal and the last N_B lie on the boundary, with $N = N_I + N_B$, we can find the approximate solution $u^h(\boldsymbol{x}_j)$ by solving the following linear system:

$$\begin{pmatrix} c_{1,1} & \dots & c_{1,N_I} \\ \vdots & \ddots & \vdots \\ c_{N_I,1} & \dots & c_{N_I,N_I} \end{pmatrix} \begin{pmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{pmatrix} = \boldsymbol{f} - \begin{pmatrix} c_{1,N_I+1} & \dots & c_{1,N_B} \\ \vdots & \ddots & \vdots \\ c_{N_I,N_I+1} & \dots & c_{N_I,N_B} \end{pmatrix} \boldsymbol{g} \tag{1.7}$$

where $\boldsymbol{f} = \{f(\boldsymbol{x}_1) \dots f(\boldsymbol{x}_{N_I})\}^T$ and $\boldsymbol{g} = \{g(\boldsymbol{x}_{N_I+1}) \dots g(\boldsymbol{x}_{N_B})\}^T$, and the coefficient matrix $\boldsymbol{C}$ is found as the solution of:

$$\begin{pmatrix} c_{1,1} & \dots & c_{N_I,1} \\ \vdots & \ddots & \vdots \\ c_{1,N} & \dots & c_{N_I,N} \end{pmatrix} = \boldsymbol{B^{-T}} \begin{pmatrix} LB_1(\boldsymbol{x}_1) & \dots & \mathcal{L}B_1(\boldsymbol{x}_{N_I}) \\ \vdots & \ddots & \vdots \\ \mathcal{L}B_N(\boldsymbol{x}_1) & \dots & \mathcal{L}B_N(\boldsymbol{x}_{N_I}) \end{pmatrix} \tag{1.8}$$

Thus, the coefficients $\boldsymbol{c}_i = \{c_{i,1} \dots c_{i,N}\}$ of each line of equation (1.7) are calculated by solving the following linear system, which is closely related to that of scattered data interpolation of equation (1.5):

$$\boldsymbol{B^T c}_i = \begin{pmatrix} \mathcal{L}B_1(\boldsymbol{x}_i) \\ \vdots \\ \mathcal{L}B_N(\boldsymbol{x}_i) \end{pmatrix} \tag{1.9}$$

Indeed, by comparing equation (1.9) and equation (1.5) it is clear that both scattered data interpolation and the solution of partial differential equations are associated to the same matrix $\boldsymbol{B}$. This is why most of the literature on the solution of PDEs using radial basis functions is focused on the properties of matrix $\boldsymbol{B}$ and of equation (1.5). An important remark to be made here is that equation (1.8), with matrix $\boldsymbol{B}$ defined as in equation (1.5) is only correct if Dirichlet boundary conditions are enforced, if this is not the case different boundary conditions appear inside matrix $\boldsymbol{B}$.

Chapter 2

RBF Interpolation

As stated in the Introduction, the theory of scattered data interpolation with RBFs really is at the foundation of their use in the solution of Partial Differential Equations. The aim of this chapter is therefore to summarize some results from the theory of interpolation which I find essential for an intuitive understanding of the mechanisms underlying the solvers based on RBFs and are especially important for CFD applications.

2.1 The Mairhuber-Curtis Theorem

In equation (1.8) at page 15 we see how matrix $\boldsymbol{B}$ needs to be non-singular in order to allow for the solution of the boundary value problem, moreover, this must hold regardless of the node placement as long as the nodes are distinct: the solver must work for any shape or discretization of the domain Ω. The choice of a suitable set of basis functions, however, is not trivial: if for example we assume that $B_k(\boldsymbol{x}) \in \Pi_P^d$ is a polynomial basis of the space Π_P^d of polynomials of degree at most P in $\mathbb{R}^d$, then we are guaranteed that $\boldsymbol{B}$ is non-singular only for $d = 1$, i.e. for one-dimensional problems, but we cannot be sure it is invertible for $d > 1$. This negative result is formalized in the **Mairhuber-Curtis theorem** [20, 69, 36]:

Theorem. *Given any set of basis functions $B_k(\boldsymbol{x})$, $k = 1, \ldots, N$ with $\boldsymbol{x} \in \mathbb{R}^d$, $d \geq 2$, the problem of determining an interpolant defined as in* (1.2) *and satisfying conditions* (1.4)*, is singular for infinitely many configurations of distinct nodes $\boldsymbol{x}_k$, $k = 1, \ldots, N$.*

The proof in [36] is given by pointing out that in more than one dimension it is possible to move the nodes continuously so that two nodes end up interchanged following two non intersecting paths. If all the other nodes remain at their initial positions, the determinant of matrix $\boldsymbol{B}$ after this exchange has changed sign because $\boldsymbol{B}$ now appears with two rows exchanged. By continuity, there must be a position at which matrix $\boldsymbol{B}$ is singular. In other words, if the basis

$\{B_1(\boldsymbol{x}), \ldots, B_N(\boldsymbol{x})\}$ is independent from the nodes position, it is bound to produce a singular matrix $\boldsymbol{B}$ sooner or later. However, by choosing a basis that is also a function of the nodes positions, it is in general possible to ensure the non singularity of the interpolation problem and hence proceed to solve differential equations. This is because if we move the collocation nodes the basis also changes and if two nodes switch places not only the rows are switched but also the columns.

2.2 Interpolation with Radial Basis Functions

At first we remark that in equation (1.2) the basis functions $B_k(\boldsymbol{x})$ need to be multivariate, since $\boldsymbol{x} \in \mathbb{R}^d$. Following the treatment of the topic proposed in [27], in order to introduce the (multivariate) *basis* functions it is convenient to illustrate how these are derived from the (univariate) *basic* functions. Consider the univariate Gaussian function:

$$\varphi(r) = e^{-(\varepsilon r)^2}, \quad r \in \mathbb{R} \tag{2.1}$$

This can be made multivariate by composition with the Euclidean distance function $||\boldsymbol{x} - \boldsymbol{x}_k||_2$ from a given center $\boldsymbol{x}_k \in \mathbb{R}^d$:

$$\Phi_k(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_k\|_2) = e^{-\varepsilon^2 \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2}, \quad \boldsymbol{x} \in \mathbb{R}^d \tag{2.2}$$

Sometimes the notations $\Phi(\boldsymbol{x} - \boldsymbol{x}_k)$ or $\Phi(\cdot, \boldsymbol{x}_k)$ are also used instead of $\Phi_k(\boldsymbol{x})$. Following the naming introduced in [27], φ is called a *basic* function and Φ_k is called a *basis* function, one single basic function generates all of the basis functions that are used in expansion (1.2). We also remark that the centers used to define the basis functions can be different from the collocation nodes, as highlighted in [12]. More often, and it will also be the case here, the centers of the basis functions and the collocation points (or nodes) coincide, this leads to a square and symmetric matrix $\boldsymbol{B}$:

$$\boldsymbol{B} = \begin{pmatrix} \varphi(||\boldsymbol{x}_1 - \boldsymbol{x}_1||) & \ldots & \varphi(||\boldsymbol{x}_1 - \boldsymbol{x}_N||) \\ \vdots & \ddots & \vdots \\ \varphi(||\boldsymbol{x}_N - \boldsymbol{x}_1||) & \ldots & \varphi(||\boldsymbol{x}_N - \boldsymbol{x}_N||) \end{pmatrix} \tag{2.3}$$

Along with the Gaussian basic function, many others have been proposed in the literature, some of them are listed in Table 2.1. The smoothness of each function, indicated in the leftmost column, is linked to the theoretical order of convergence allowed by the associated interpolation scheme, more details will be provided in the following pages. As for the rightmost column, the status of *strictly* and *conditionally* positive definiteness really requires further explanation, a very minimal and by no means exhaustive one is provided in the next section. The interested reader is encouraged to see [110, 27, 35].

Smoothness	Name	Definition $\varphi((\varepsilon)r)$	Positive definite, order
Infinitely S.	Gaussian (GA)	$e^{-(\varepsilon\, r)^2}$	strictly
	Multiquadric (MQ)	$\sqrt{1+(\varepsilon\, r)^2}$	s. conditionally, 1
	Inverse Multiquadric (IMQ)	$1/\sqrt{1+(\varepsilon\, r)^2}$	strictly
	Inverse Quadratic (IQ)	$1/(1+(\varepsilon\, r)^2)$	strictly
Piecewise S.	Monomial PHS	r^{2k+1}, $k \in \mathbb{N}$	s. conditionally, $k+1$
	Thin Plate Spline PHS	$r^{2k}\log r$, $k \in \mathbb{N}$	s. conditionally, $k+1$

Table 2.1: most common basic functions

2.3 Positive Definiteness

In Table 2.1 basic functions are denoted either as strictly positive definite or as strictly conditionally positive definite, the same attributes are inherited by associated RBFs Φ. In agreement with the most traditional terminology [27], we will call *strictly* positive definite those RBFs which are associated with a positive definite matrix (2.3). The associated interpolation system $\boldsymbol{B}\boldsymbol{\alpha} = \boldsymbol{u}$ of equation (1.5) is always well posed and therefore yields a unique solution. *Strictly conditionally* positive definite functions of order P, instead, are those basic functions which require polynomial augmentation with a basis for all polynomials of order at least $P-1$ for the interpolation problem to be non singular. Furthermore, in case of polynomial augmentation, nodes must also be positioned in order to form a polynomially unisolvent set [110].

2.3.1 Definitions and terminology

The definition of positive definite functions provided in [27] is for generic complex valued continuous functions $\Phi : \mathbb{R}^d \to \mathbb{C}$. Since we are only interested in real valued functions $\Phi : \mathbb{R}^d \to \mathbb{R}$, we report a theorem which provides a unique characterization [27]:

Theorem. *A real-valued continuous function Φ is positive definite on $\mathbb{R}^d$ if and only if it is even and*

$$\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_j\alpha_k\Phi(\boldsymbol{x}_j-\boldsymbol{x}_k)\geq 0$$

for any pairwise different points picked from $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in \mathbb{R}^d$, and $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_N]^T \in \mathbb{R}^N$. The function Φ is strictly positive definite on $\mathbb{R}^d$ if the quadratic form above is zero only for $\boldsymbol{\alpha} = \boldsymbol{0}$.

Analogously to the case of positive definite functions, also conditionally positive definite ones are associated to specific interpolation matrices, we follow once again the terminology proposed in [27]. Many radial basis functions are not strictly positive definite and thus the associated matrix $\boldsymbol{B}$ in (2.3) is not guaranteed to be non-singular for any possible node arrangement. In the case of

the generic strictly conditionally positive definite basic function of order 1, the uniqueness of the solution to the interpolation problem can be guaranteed by requiring that the coefficients α_i in (1.5) satisfy the additional condition:

$$\sum_{i=1}^{N} \alpha_i = 0 \tag{2.4}$$

This makes the associated matrix $\boldsymbol{B}$ conditionally positive definite of order one.

Definition. A real symmetric matrix $\boldsymbol{B}$ is called conditionally positive semi-definite of order one if the associated quadratic form is non singular, i.e:

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k \boldsymbol{B}_{jk} \geq 0 \tag{2.5}$$

for all $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_N\}^T \in \mathbb{R}^N$ that satisfy equation (2.4). If $\boldsymbol{\alpha} \neq \boldsymbol{0}$ implies strict inequality in (2.5) then $\boldsymbol{B}$ is called conditionally positive definite of order one. [27]

We see now that equation (1.5) is no longer uniquely solvable in general for conditionally positive definite matrices of order 1 or higher, fortunately it can be modified according to the following theorem [27], which will be proven later.

Theorem. *Let $\boldsymbol{B}$ be a real symmetric $N \times N$ matrix that is conditionally positive definite of order* 1*, and let $\boldsymbol{P_0} = \{1, \ldots, 1\}^T \in \mathbb{R}^N$. Then the system of linear equations*

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{P_0} \\ \boldsymbol{P_0}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ d \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ 0 \end{pmatrix} \tag{2.6}$$

is uniquely solvable.

We will call strictly conditionally positive definite (again, as in [27]) of order 1 those functions which yield conditionally positive definite matrices. This concept is formalized in the following theorem (modified from [27]).

Theorem. *A real valued continuous function Φ is called strictly conditionally positive definite of order one on $\mathbb{R}^d$ if it is even and*

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 0 \tag{2.7}$$

for any pairwise different points picked from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$, and $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}^T \in \mathbb{R}^N$ satisfying (2.4)*. And the quadratic form above is zero only for $\boldsymbol{\alpha} = \boldsymbol{0}$.*

If a strictly conditionally positive definite function Φ is chosen as basis function, the associated matrix in the interpolation system becomes the one

at the left-hand-side of Equation (2.6). We remark that such a matrix is still symmetric and the whole process of solution of PDEs explained in subsection 1.4.2 can still be followed with minimal changes, i.e. by using the augmented matrix instead of matrix $\boldsymbol{B}$.

In the case of Monomial Poly Harmonic Splines (PHS) and Thin Plate spline PHS in Table 2.1 and other conditionally positive definite functions of order $P > 1$, in order to ensure that the interpolation problem is well posed, further conditions are required. We characterize conditionally positive definite functions of order P as follows [27].

Theorem. *A real valued continuous function Φ is called strictly conditionally positive definite of order P on $\mathbb{R}^d$ if it is even and*

$$\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_j\alpha_k\Phi(\boldsymbol{x}_j-\boldsymbol{x}_k)\geq 0 \tag{2.8}$$

for any N pairwise different points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\in\mathbb{R}^d$, and $\boldsymbol{\alpha}=\{\alpha_1,\ldots,\alpha_N\}^T\in\mathbb{R}^N$ satisfying

$$\sum_{i=1}^{N}\alpha_i p(\boldsymbol{x}_i)=0 \tag{2.9}$$

for any real-valued polynomial p of degree at most $P-1$. The quadratic form above is zero only for $\boldsymbol{\alpha}=\boldsymbol{0}$.

2.3.2 Derivation from a quadratic programming problem

We remark that condition (2.6) can be viewed as a particular case of (2.9), where $p(\boldsymbol{x}_i)$ is the constant polynomial of degree 0. In order to derive a general approach for the treatment of interpolation problems with conditionally positive definite functions of order P we propose the following procedure, based on the theory of constrained optimization [77].

Suppose once again that a function $u(\boldsymbol{x})$ is given at a set of points $\boldsymbol{x}_i$, $i=1,\ldots,N$, with $\boldsymbol{u}=\{u(\boldsymbol{x}_1),\ldots,u(\boldsymbol{x}_N)\}^T$. We are now interested in finding the solution to the following equality constrained quadratic programming problem.

$$\begin{aligned}\min_{\boldsymbol{\alpha}\in\mathbb{R}^N}\ & q(\boldsymbol{\alpha})=\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{B}\boldsymbol{\alpha}-\boldsymbol{\alpha}^T\boldsymbol{u}\\ \textit{subject to}\quad & \boldsymbol{P}^T\boldsymbol{\alpha}=\boldsymbol{0}\end{aligned} \tag{2.10}$$

Where:

$$\boldsymbol{B}=\begin{pmatrix}\Phi(\boldsymbol{x}_1-\boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_1-\boldsymbol{x}_N)\\ \vdots & \ddots & \vdots\\ \Phi(\boldsymbol{x}_N-\boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_N-\boldsymbol{x}_N)\end{pmatrix}$$

is the same matrix as in (2.3), reported here again for convenience, and

$$\boldsymbol{P}^T = \begin{pmatrix} p_0(\boldsymbol{x}_1) & \dots & p_0(\boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ p_M(\boldsymbol{x}_1) & \dots & p_M(\boldsymbol{x}_N) \end{pmatrix} \tag{2.11}$$

is a matrix associated to a polynomial base of the space Π^d_{P-1} of multivariate polynomials of degree up to $P-1$ defined on $\mathbb{R}^d$. Each row corresponds to an element of the basis evaluated at the points $\boldsymbol{x}_i$, for instance, the first row is made of ones: $\{1,\dots,1\}^T \in R^N$, there are $M = \binom{d+(P-1)}{P-1}$ elements in the basis and therefore M rows. For the moment we suppose that the M rows are linearly independent and therefore that $\boldsymbol{P}$ has full rank. We remark that the quadratic form in problem (2.10) is the same as the one in equation (2.8) and that, by definition of strictly conditionally positive definite functions it must be $\boldsymbol{\alpha}^T\boldsymbol{B}\boldsymbol{\alpha} \geq 0$ for any $\boldsymbol{\alpha}$ in the null space of Π^d_{P-1} and is $= 0$ only if $\boldsymbol{\alpha}$ is the zero vector.

In order to solve the problem (2.10), the following Lagrangian function can be defined:

$$J(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{B}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{u} + \boldsymbol{\beta}^T(\boldsymbol{P}^T\boldsymbol{\alpha}) \tag{2.12}$$

Where the sign of the multiplier $\boldsymbol{\beta}$ is not important because of the presence of null equality constraints $\boldsymbol{P}^T\boldsymbol{\alpha} = \boldsymbol{0}$, we proceed to require the first order necessary conditions for a minimum $\boldsymbol{\alpha}^*$ and thus write $\nabla_{\boldsymbol{\alpha},\boldsymbol{\beta}} J(\boldsymbol{\alpha},\boldsymbol{\beta}) = \boldsymbol{0}$ in matrix form as follows.

$$\underbrace{\begin{pmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{pmatrix} \tag{2.13}$$

Since $\boldsymbol{P}$ has full rank and Φ is strictly conditionally positive definite, matrix $\boldsymbol{M}$ is non singular and therefore equation (2.13) has a unique solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, which is also the global minimum for the problem (2.10) (Lemma 16.1 and Lemma 16.2 in [77]). Non singularity of $\boldsymbol{M}$ is proven as follows, suppose there are vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ such that:

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix} = \boldsymbol{0} \tag{2.14}$$

From the second row of (2.14) we have $\boldsymbol{P}^T\boldsymbol{w} = 0$, it follows that:

$$0 = \begin{pmatrix} \boldsymbol{w}^T & \boldsymbol{v}^T \end{pmatrix} \boldsymbol{M} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix} = \boldsymbol{w}^T\boldsymbol{B}\boldsymbol{w}$$

Furthermore, since $\boldsymbol{w}$ lies in the null space of $\boldsymbol{P}^T$ we know that $\boldsymbol{w}^T\boldsymbol{B}\boldsymbol{w} = 0$ can only happen if $\boldsymbol{w} = 0$ because of the definition of strictly conditionally positive definite function. From the first row of equation (2.14) we then have $\boldsymbol{P}\boldsymbol{v} = 0$, but this can only happen if $\boldsymbol{v} = 0$ because $\boldsymbol{P}$ is full rank by hypothesis. We conclude that equation (2.14) can only be true if $\boldsymbol{w} = 0$ and $\boldsymbol{v} = 0$, and therefore $\boldsymbol{M}$ is non singular.

Matrix $\boldsymbol{P}^T$ is a Vandermonde-like matrix associated to a multivariate polynomial basis of degree up to $P-1$ and equation (2.13) can be interpreted as the matrix form of the following interpolation problem.

$$\begin{aligned} u^h(\boldsymbol{x}_i) = \sum_{j=1}^{N} \alpha_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}_i) = u(\boldsymbol{x}_i) \qquad & i = 1, \dots, N \\ \sum_{i=1}^{N} \alpha_i p_k(\boldsymbol{x}_i) = 0 \qquad & k = 1, \dots, M \end{aligned} \tag{2.15}$$

We conclude that, given a set of collocation points $\boldsymbol{x}_i$, $i = 1, \dots, N$ where a function u is given, it can be approximated by an interpolant u^h defined as in (2.15) and that such approximation problem is well defined for any strictly conditionally positive definite function Φ of order $\leq P$. When Φ is obtained from an RBF basic function chosen from Table 2.1, then (2.15) defines the associated interpolation problem with polynomial augmentation, i.e. where a polynomial basis is added to the usual basis made of RBFs.

2.4 Polynomial Augmentation

As explained in the previous subsection, given a function u, this can be approximated by another function u^h which is defined by equation (2.15). Because of the non singularity of the associated matrix $\boldsymbol{B}$, u^h is unique and satisfies the collocation conditions $u^h(\boldsymbol{x}_i) = u(\boldsymbol{x}_i)$, for each $\boldsymbol{x}_i$ in a given set of collocation nodes. The proof of the non-singularity of M, however, assumed that $\boldsymbol{P}$ in equation (2.13) was full-rank, i.e. had linearly independent columns. This requirement is seemingly in contrast with what stated in the Mairhuber-Curtis Theorem. Indeed, while the theoretical limit for the size of matrix $\boldsymbol{P}$ is $M = N$, where the matrix becomes square and $\boldsymbol{P}$ is the polynomial interpolation matrix, ill-conditioning and singularity issues might already appear for polynomials of lower degree (and hence with $M < N$).

For the matrix $\boldsymbol{P}$ to be full rank, the set of points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ must be (P-1)-unisolvent, where P-1 is the degree of the polynomials. Here is a formal definition [27]:

Definition. We call a set of points $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ *P-unisolvent* if the only polynomial of total degree at most P interpolating zero data on $\mathcal{X}$ is the zero polynomial.

This property is also called nondegeneracy of the set $\mathcal{X}$ for Π_P^d[114]. In [27] this topic is discussed in greater detail, we remark that with the node generation

techniques discussed in the present work, a safe rule for a stable implementation of the polynomial augmentation is to use $2M \leq N$.

Aside from the fact that polynomial augmentation of a suitable degree unlocks the possibility to use strictly conditionally positive definite RBFs, it also has other advantages. The most important is that, regardless of the nodes distribution, as long as it is P-unisolvent, a polynomial basis of degree P enables polynomial precision of the same degree. This means that, if the data $u(\boldsymbol{x}_i)$ come from a polynomial of total degree less than or equal to P, then they are fitted exactly by $u^h(\boldsymbol{x}_i)$ and we have $u^h(\boldsymbol{x}) = u(\boldsymbol{x})$ not only at the collocation points but everywhere. Furthermore, granted that the set of points is P-unisolvent, P can be as high as we want and the invertibility of matrix (M) in equation (2.13) is still granted, indeed, a function that is strictly conditionally positive definite of order P_0 is also strictly conditionally positive definite of any higher order [27].

2.5 Cardinal Functions

Up to this point the most complete formulation for u^h is written in equation (2.15) and for a generic $\boldsymbol{x}$ takes the form:

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \Phi(\boldsymbol{x} - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \tag{2.16}$$

It is however possible to find a different formulation in terms of the so-called cardinal functions ψ_j, $j = 1, \ldots, N$, which takes the form:

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \psi_j(\boldsymbol{x}) u(\boldsymbol{x}_j) \tag{2.17}$$

where the cardinal functions ψ_j for a given set of nodes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ satisfy the following property:

$$\psi_j(\boldsymbol{x}_i) = \delta_{ij} \ , \quad i, j = 1, \ldots, N \tag{2.18}$$

We can obtain the vector $\boldsymbol{\psi} = \{\psi_1(\boldsymbol{x}), \ldots \psi_N(\boldsymbol{x})\}^T$ by solving the following linear system.

$$\boldsymbol{M}^T \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{x}) \\ \boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} \tag{2.19}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}) = \{\Phi(\boldsymbol{x} - \boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x} - \boldsymbol{x}_N)\}^T$, and $\boldsymbol{p}(\boldsymbol{x}) = \{p_1(\boldsymbol{x}), \ldots, p_M(\boldsymbol{x})\}^T$ is the usual polynomial base evaluated at $\boldsymbol{x}$. Equation (2.19) is obtained by substituting u^h as defined in (2.17) into (2.16) and substituting coefficients $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}^T$ and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_N\}^T$ from equation (2.13), as follows.

$$\begin{pmatrix} \boldsymbol{u} \\ 0 \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\psi}(\boldsymbol{x}) \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{x}) \\ \boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ 0 \end{pmatrix}^T \boldsymbol{M}^{-T} \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{x}) \\ \boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} \tag{2.20}$$

The uniqueness of the cardinal functions is guaranteed if Φ is strictly conditionally positive definite and therefore matrix $\boldsymbol{M}$ is non-singular. By evaluating the right-hand-side of equation (2.19) at the nodes $\boldsymbol{x}_j$ it can easily be seen that ψ_j satisfy conditions (2.18).

2.6 Reproducing Kernels

In order to discuss the main error estimates for the interpolation methods based on radial basis functions, it is necessary to introduce the notions of reproducing kernel and native space. The goal of this section is only to introduce these concepts at an intuitive level along with the related notation, therefore no proofs will be given and the discussion will mostly be limited to the case of strictly positive definite functions. Extensions to the case of conditionally positive definite functions require more calculations but follow naturally from the definitions given in the previous sections.

As explained above, a function u can be approximated by u^h defined as in equation (2.16) or (2.17). The simplest case is when Φ is strictly positive definite and therefore the approximation u^h for each u is unique and written as:

$$u(\boldsymbol{x}) \approx u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \Phi(\boldsymbol{x} - \boldsymbol{x}_j) \tag{2.21}$$

In order to adapt our notation to the one usually adopted in this field, we denote variables as "$\cdot$" and write for example $\Phi(\cdot, \boldsymbol{x})$ to indicate that the first argument varies freely in the domain Ω and the second is fixed. With this notation it is implied that:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x} - \boldsymbol{y}) \tag{2.22}$$

$\Phi(\cdot, \cdot)$ then becomes a real valued function defined on the Cartesian product $\Omega \times \Omega$. In summary, all of radial basis functions associated to basic functions in Table 2.1 can also be associated with functions $\Phi : \Omega \times \Omega \to \mathbb{R}$, which are also called *kernels*, according to the definition of kernel given in [88].

Consider now the set of functions $u^h = \sum_{i=1}^{N} \alpha_i \Phi(\cdot, \boldsymbol{x}_i)$, u^h can be evaluated at any given point $\boldsymbol{x}$ by substituting $\boldsymbol{x}$ in the first argument of all $\Phi(\cdot, \boldsymbol{x}_i)$: $u^h(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \Phi(\boldsymbol{x}, \boldsymbol{x}_i)$. The action of evaluating u^h at $\boldsymbol{x}$ can be described as that of a linear point evaluation functional. Indicating point evaluation functionals with $\delta_{\boldsymbol{x}}$ we write $\delta_{\boldsymbol{x}} \Phi(\cdot, \boldsymbol{x}_i) = \Phi(\boldsymbol{x}, \boldsymbol{x}_i)$ and also $\delta_{\boldsymbol{x}} u^h = u^h(\boldsymbol{x})$, where $\delta_{\boldsymbol{x}}$ is defined to be linear and therefore distributes on the expansion of u^h.

Two notable properties of strictly positive RBFs are:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{y}, \boldsymbol{x}) \tag{2.23}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k \Phi(\boldsymbol{x}_j, \boldsymbol{x}_k) \geq 0, \text{ and } = 0 \text{ only if } \boldsymbol{\alpha} = 0 \tag{2.24}$$

If we define $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ to be the bilinear product which satisfies:

$$\langle \Phi(\cdot, \boldsymbol{x}), \Phi(\cdot, \boldsymbol{y}) \rangle_{\mathcal{H}} = \Phi(\boldsymbol{x}, \boldsymbol{y}) \tag{2.25}$$

and the symmetry of $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a consequence of (2.23).

Furthermore, $\langle u^h, \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} = \delta_{\boldsymbol{x}} u^h$ is also satisfied since:

$$\begin{aligned} \langle u^h, \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} &= \langle \sum_{i=1}^{N} \alpha_i \Phi(\cdot, \boldsymbol{x}_i), \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \alpha_i \langle \Phi(\cdot, \boldsymbol{x}_i), \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \alpha_i \Phi(\boldsymbol{x}, \boldsymbol{x}_i) = u^h(\boldsymbol{x}) \end{aligned} \tag{2.26}$$

Finally, $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces a norm because of (2.24): $\langle u^h, u^h \rangle_{\mathcal{H}} > 0$ and $= 0$ only if $u^h = 0$, as shown below:

$$\begin{aligned} \langle u^h, u^h \rangle_{\mathcal{H}} &= \langle \sum_{i=1}^{N} \alpha_i \Phi(\cdot, \boldsymbol{x}_i), \sum_{j=1}^{N} \alpha_j \Phi(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \langle \Phi(\cdot, \boldsymbol{x}_i), \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{2.27}$$

We can now have an intuitive understanding of how the radial basis function Φ, considered as a function of two arguments can be used to generate a Hilbert Space $\mathcal{H}$ with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that (2.25) is satisfied, $\mathcal{H}$ is often called the native space associated with the function Φ.

In the case of strictly conditionally positive definite functions, $\mathcal{H}$ contains linear combinations of the basis $\Phi(\cdot, \boldsymbol{x})$ that vanish on Π_P. For conditionally positive definite functions the inner product induces a seminorm instead of a norm. In the case of strictly conditionally positive definite functions, the interpolant u^h can be shown to be the solution of (2.10).

All Radial Basis Functions discussed in this thesis are reproducing kernels for the associated native spaces, the concept of reproducing kernel is formally defined as [27]:

Definition. Let $\mathcal{H}$ be a real Hilbert space of functions $f : \Omega (\subseteq \mathbb{R}^d) \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. A function $\Phi : \Omega \times \Omega \to \mathbb{R}$ is called *reproducing kernel for* $\mathcal{H}$ if

$$\begin{aligned} &\Phi(\cdot, \boldsymbol{x}) \in \mathcal{H} \text{ for all } \boldsymbol{x} \in \Omega \\ &f(\boldsymbol{x}) = \langle f, \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} \text{ for all } f \in \mathcal{H} \text{ and all } \boldsymbol{x} \in \Omega \end{aligned} \tag{2.28}$$

Where property (2.28) is called reproducing property. The existence of a reproducing kernel for a given Hilbert space is equivalent to the fact that the point evaluation functionals $\delta_{\boldsymbol{x}}$ are bounded linear functionals on Ω, this fact is a consequence of the Riesz-Frechet representation theorem [27, 8].

We also have the following two uniqueness results [87]:

Theorem. *The native space for a given strictly positive definite function Φ is unique if it exists, and it coincides with the closure of the space of finite linear combinations of functions $\Phi(\cdot, \boldsymbol{x})$, $\boldsymbol{x} \in \Omega$ under the inner product defined via equation* (2.25)

And vice versa:

Theorem. *Any strictly positive definite function Φ on some domain Ω has a unique native space which is the closure of the space*

$$F_\Phi(\Omega) = \left\{ \sum_{i=1}^{N} \alpha_i \Phi(\cdot, \boldsymbol{x}_i), \quad \alpha_i \in \mathbb{R}, N \in \mathbb{N}, \ \boldsymbol{x}_i \in \Omega \right\}$$

and the elements of the native space can be interpreted as functions on Ω.

The results obtained for strictly positive definite functions can be extended to strictly conditionally positive definite ones with some revisions, see also [87]. In order to avoid further theoretical details we conclude here the discussion on the reproducing kernel Hilbert spaces, interested readers are strongly advised to research original material on [110, 88, 28].

2.7 Error estimates and orders of convergence

In the case of meshless methods based on Radial Basis Functions, as hinted in section 1.4.2, the same matrices appear in the linear systems associated to the discretization of linear operators and the interpolation of scattered data. It is therefore very reasonable to expect that reconstruction errors affecting scattered data interpolation also appear in the discretization of linear operators. Indeed, while many different sources of errors might affect the discretization of differential operators over a given domain, the study of the interpolation errors is somewhat simpler and has been conducted with success by multiple authors [110, 35]. In this section we discuss the main error estimates for the scattered data interpolation using different types of RBFs.

2.7.1 The Power Function

When the function to be interpolated u lies in the Native Hilbert Space $\mathcal{H}$ associated with the basis function Φ, i.e. it can be written as $u(\cdot) = \sum_i \alpha_i \Phi(\cdot, \boldsymbol{x}_i)$. Then given an interpolant u^h obtained using the nodes in $\mathcal{X}$, the interpolation

error can be written as $E_{\Phi,\mathcal{X}}(u) = |u(\boldsymbol{x}) - u^h(\boldsymbol{x})|$, where both u and u^h lie in $\mathcal{H}$.

$$\begin{aligned} E_{\Phi,\mathcal{X}}(u) = |u(\boldsymbol{x}) - u^h(\boldsymbol{x})| &= |\langle u, \Phi(\cdot,\boldsymbol{x})\rangle_{\mathcal{H}} - \sum_{j=1}^{N} \psi_j(\boldsymbol{x})\langle u, \Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{H}}| \\ &= |\langle u, \Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} \psi_j(\boldsymbol{x})\Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{H}}| \\ &\leq \|u\|_{\mathcal{H}} \, \|\Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} \psi_j(\boldsymbol{x})\Phi(\cdot,\boldsymbol{x}_j)\|_{\mathcal{H}} \end{aligned} \tag{2.29}$$

where $\psi_j(\boldsymbol{x})$ with $j = 1 \dots N$ are the cardinal functions evaluated at $\boldsymbol{x}$ and the expansion (2.17) was used. Following [27, 86] we define the power function $P_{\Phi,\mathcal{X}}$ as follows.

$$P_{\Phi,\mathcal{X}}(\boldsymbol{x}) = \|\Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} \psi_j(\boldsymbol{x})\Phi(\cdot,\boldsymbol{x}_j)\|_{\mathcal{H}} \tag{2.30}$$

We see that for $\boldsymbol{x}_i \in \mathcal{X}$, $P_{\Phi,\mathcal{X}}(\boldsymbol{x}_i) = 0$, furthermore $P_{\Phi,\mathcal{X}}$ does not depend on u and a first estimate of the interpolation norm can be written as:

$$|u(\boldsymbol{x}) - u^h(\boldsymbol{x})| \leq P_{\Phi,\mathcal{X}}(\boldsymbol{x})\|u\|_{\mathcal{H}} \tag{2.31}$$

Before concluding we state a few more remarks on the power function, if we evaluate $P^2_{\Phi,\mathcal{X}}$ we have:

$$\begin{aligned} P^2_{\Phi,\mathcal{X}} &= \langle P_{\Phi,\mathcal{X}}, P_{\Phi,\mathcal{X}}\rangle_{\mathcal{H}} \\ &= \langle \Phi(\cdot,\boldsymbol{x}), \Phi(\cdot,x)\rangle_{\mathcal{H}} - 2\sum_{j=1}^{N} \psi_j(\boldsymbol{x})\langle \Phi(\cdot,\boldsymbol{x}), \Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{H}} \\ &\quad + \sum_{j=1}^{N}\sum_{k=1}^{N} \psi_j(\boldsymbol{x})\psi_k(\boldsymbol{x})\langle \Phi(\cdot,\boldsymbol{x}_j), \Phi(\cdot,\boldsymbol{x}_k)\rangle_{\mathcal{H}} \\ &= \Phi(\boldsymbol{x},\boldsymbol{x}) - 2\sum_{j=1}^{N} \psi_j(\boldsymbol{x})\Phi(\boldsymbol{x},\boldsymbol{x}_j) + \sum_{j=1}^{N}\sum_{k=1}^{N} \psi_j(\boldsymbol{x})\psi_k(\boldsymbol{x})\Phi(\boldsymbol{x}_j,\boldsymbol{x}_k) \end{aligned} \tag{2.32}$$

which can also be written in matrix form as follows:

$$P^2_{\Phi,\mathcal{X}} = \Phi(\boldsymbol{x},\boldsymbol{x}) - 2\boldsymbol{\psi}^T\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\psi}^T\boldsymbol{B}\boldsymbol{\psi} \tag{2.33}$$

where the notation is the same used in section 2.5 where the cardinal functions are defined. Furthermore, suppose we had calculated (2.33) using another vector $\boldsymbol{\psi}$, if we seek to find the minimum for $P_{\Phi,\mathcal{X}}$ with respect to $\boldsymbol{\psi}$, we obtain the following necessary condition:

$$\boldsymbol{B}\boldsymbol{\psi} = \boldsymbol{\Phi}(\boldsymbol{x}) \tag{2.34}$$

which is precisely the definiton of the cardinal functions $\psi_i(\boldsymbol{x})$ given in equation (2.19). Therefore, $P_{\Phi,\mathcal{X}}$ is the smallest function satisfying (2.31) and can therefore be considered the operator norm of the error operator inside the native space generated by $\Phi(\cdot,\cdot)$ [86].

2.7.2 Fill Distance

In general, in order to improve the accuracy of the available solution to a given Partial Differential Equation, the most immediate thing to do is to increase the spatial resolution by increasing the density of nodes or cells in a given domain. The single most important requirement for a numerical method is therefore the ability to improve its estimates of the exact solution as the density of nodes or cells is increased. This is even more true for problems of interpolation over scattered data, where increasing the density of available data is always expected to reduce the error. It is clear, however, that the mere increase in the number of nodes is not enough to guarantee an improvement and that some regularity in the node distribution must be required for it to be effective. A measure of the effective node density that is used in approximation theory is the so-called *fill distance*. Given a set of nodes $\mathcal{X}$ scattered on a domain Ω, the fill distance h is defined as follows:

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_i\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \tag{2.35}$$

The fill distance denotes the radius of the largest empty (or data-less) ball that can be placed among the nodes $\boldsymbol{x}_i \in \mathcal{X}$, i.e. it measures the largest gap in the data [12] or the maximum distance any point in Ω can be from $\mathcal{X}$ [76]. Given an exact function u, in this section we provide a few notable estimates for how fast a given norm of the error $\|u - u^h\|$ tends to zero as $h \to 0$.

2.7.3 Approximation Order

This term is used to indicate the speed of convergence to zero of the approximation error with respect to the fill distance. In [27, 110] this concept is formalized as follows. We say that the approximation operator $\mathcal{A}^{(h)}$ has L_p-*approximation order* k if:

$$\|u - \mathcal{A}_u^{(h)}\|_p = \mathcal{O}(h^k) \quad for\ h \to 0 \tag{2.36}$$

where $\mathcal{A}_u^{(h)}$ is the interpolant provided by the approximation operator on a node distribution $\mathcal{X}$ with a fill distance h.

Usually, the interpolation error converges to zero as $h \to 0$ at a rate dictated by the minimum smoothness of u and basic function φ used to derive Φ. [12].

2.7.4 Error bound as a function of the fill distance

A lot of research activity around Radial Basis Functions has been focused on the derivation of error bounds that take h into account, both for functions belonging to the native space and outside of it. An interested reader is encouraged to consult [110, 27, 76, 114].

The notation $D^{\boldsymbol{\alpha}}$, with multi-index $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_d\}$ and $|\boldsymbol{\alpha}| = \sum_i \alpha_i$ will be used, it indicates the differential operator:

$$D^{\boldsymbol{\alpha}} = \frac{\partial^{|\boldsymbol{\alpha}|}}{(\partial x_1)^{\alpha_1} \ldots (\partial x_d)^{\alpha_d}} \tag{2.37}$$

and $D_2^{\boldsymbol{\alpha}}\Phi(\cdot,\cdot)$ indicates that such operator is applied to Φ considered as a function of the second argument.

Non Stationary Interpolation

The most basic results (cfr. Theorem 14.5 and Theorem 14.6 in [27]) can be summarized as follows: suppose Ω is bounded and satisfies the interior cone condition, and that $\Phi \in C^{2k}(\Omega \times \Omega)$ is symmetric and strictly positive definite. If u is in the native space $\mathcal{H}$ of Φ, then from the upper bound of equation (2.31) it is possible to derive the following limits:

$$\begin{aligned} |u(\boldsymbol{x}) - u^h(\boldsymbol{x})| &\leq C_1(\Phi, \boldsymbol{x}) h_{\mathcal{X},\Omega}^k |u|_{\mathcal{H}} \\ |D^{\boldsymbol{\alpha}} u(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} u^h(\boldsymbol{x})| &\leq C_2(\Phi, \boldsymbol{x}) h_{\mathcal{X},\Omega}^{k-|\boldsymbol{\alpha}|} |u|_{\mathcal{H}} \end{aligned} \tag{2.38}$$

For functions which are infinitely smooth (cfr. Table 2.1) it is possible to improve the limit above [110], in the case of multiquadric, under the same hypotheses we obtain that there is a real constant c such that:

$$\|u - u^h\|_{L_\infty(\Omega)} \leq |u|_{\mathcal{H}} \exp\left(\frac{-c}{h_{\mathcal{X},\Omega}}\right) \tag{2.39}$$

whereas for the Gaussian:

$$\|u - u^h\|_{L_\infty(\Omega)} \leq |u|_{\mathcal{H}} \exp\left(\frac{-c|\log h_{\mathcal{X},\Omega}|}{h_{\mathcal{X},\Omega}}\right) \tag{2.40}$$

Therefore infinitely smooth radial basis functions provide a so-called spectral accuracy, i.e. they potentially allow an approximation order that is higher than any polynomial order.

Similar error bounds have also been proven for functions u lying outside the native space and in Sobolev Spaces, i.e. those containing the solutions to some partial differential equations [27].

Stationary Interpolation

At this point an important clarification must be given, in Table 2.1 some basic functions (the infinitely smooth ones) are defined with a factor ε multiplying the radius r, usually called *scaling* or *shape factor*. All of the error bounds above are derived under the assumption that ε remains fixed as the node density is increased. This is the so-called *non stationary* approximation and implicitly assume that, as the node density increase, the product $\varepsilon\, r$ can tend to zero and the interpolation problem remains well posed. While this can be true analytically, all authors highlight that in practice some ill-conditioning issues arise for the interpolation when $(\varepsilon\, r) \to 0$. If $r \to 0$ while ε remains constant this can intuitively be easily understood by looking at matrix $\boldsymbol{B}$ in equation (2.3): the closer any two nodes come together and the smaller their distance becomes, at the limit two or more rows (and columns) become identical.

The simplest solution to this problem is to define the scaling factor ε to be related to $h_{\mathcal{X},\Omega}$ with inverse proportionality:

$$\varepsilon h_{\mathcal{X},\Omega} = \varepsilon_0 \tag{2.41}$$

for some real number ε_0 which can also be optimized, in this case we have the so-called *stationary* interpolation scheme. However, when this is done, the approximation orders decrease drastically. The worst case is that of the Gaussian (GA) basic function, which does not provide any positive approximation order [27]. In the case of the multiquadrics (MQ), [115] provides error bounds even for the stationary interpolation. As defined in our table 2.1, MQ still guarantees an order $\mathcal{O}(h^2_{\mathcal{X},\Omega})$ for functions in the Sobolev Space W_2^2 [115, 27].

Fortunately, high accuracy can still be achieved even in the case of stationary interpolation thanks to polynomial augmentation: if the RBF is augmented with a polynomial term, the exact polynomial reconstruction is maintained even with stationary interpolation. Therefore, if a polynomial of degree P is added to the RBF, a polynomial convergence order $\mathcal{O}(h^P_{\mathcal{X},\Omega})$ can still be achieved. In practice even higher approximation orders are attained [27] and this is why it is often convenient to employ polynomials of higher order than that required for ensuring the non-singularity of the interpolation problem. The combination of stationary interpolation and polynomial augmentation has many advantages: it is stable even when node density is increased and allows high accuracy. The penalty in terms of computational cost is in most cases lower than that of stable non-stationary RBF schemes [35].

2.8 The Trade-off Principle

A severely ill conditioned linear system (2.13) associated to the interpolation problem leads to high reconstruction errors. The mechanism is purely numerical: when matrix $\boldsymbol{M}$ has a very high condition number $cond(\boldsymbol{M})$, the sensitivity with respect to minimal perturbations on the right hand side or even in the matrix entries becomes so high that the linear system can no longer be reliably solved in double precision floating-point arithmetic. This fact led many to believe that system (2.13) is either well conditioned or highly accurate but not both, this idea was formalized in a so-called trade-off principle (also called uncertainty principle by some authors [86]).

2.8.1 Separation Distance

At the end of the previous section the discussion on the conditioning of $\boldsymbol{B}$ matrix was taken into account in dealing with stationary and non stationary interpolation. We stated that, in the non stationary framework, the more we increase the node density, the worse the conditioning of the $\boldsymbol{B}$ matrix becomes and that this happens because the distance between any two collocation nodes $\boldsymbol{x}_i$ decreases. The parameter used for monitoring the distance between collocation

nodes is the *separation distance*, defined as follows [86, 12]:

$$s(\mathcal{X}) = \frac{1}{2} \min_{1 \leq j < k \leq N} \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2 \tag{2.42}$$

where $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ as usual. A node distribution is called *quasi uniform* if there is a positive constant $\gamma \leq 1$ such that:

$$\gamma h_{\mathcal{X},\Omega} \leq s(\mathcal{X}) \leq h_{\mathcal{X},\Omega} \tag{2.43}$$

It follows that if the node density is increased while maintaining a quasi uniform node distribution, $s(\mathcal{X})$ and $h_{\mathcal{X},\Omega}$ are not unrelated. The node distribution will be assumed to be quasi uniform from now on.

2.8.2 Bounds for Condition Number

In the solution of system (2.13) the conditioning of matrix M is important since it determines the sensitivity of the solution vector with respect to the known data vector. Suppose that no polynomial augmentation is used and that Φ is strictly positive definite, the interpolation problem becomes:

$$\boldsymbol{B}\boldsymbol{\alpha} = \boldsymbol{u} \tag{2.44}$$

where now $\boldsymbol{\alpha}$ is the solution vector and $\boldsymbol{u}$ is the data vector and the sensitivity of the former with respect to the latter is described by the conditioning $cond(\boldsymbol{B})$ of matrix $\boldsymbol{B}$.

$$cond(\boldsymbol{B}) = \|\boldsymbol{B}\|_2 \|\boldsymbol{B}\|_2^{-1} = \frac{\sigma_{max}}{\sigma_{min}} \tag{2.45}$$

where σ_{max} and σ_{min} are the largest and smallest singular values respectively. Since $\boldsymbol{B}$ is positive definite, $cond(\boldsymbol{B})$ can also be computed as the ratio of the largest λ_{max} and lowest λ_{min} eigenvalues [27]:

$$\frac{\lambda_{max}}{\lambda_{min}} \tag{2.46}$$

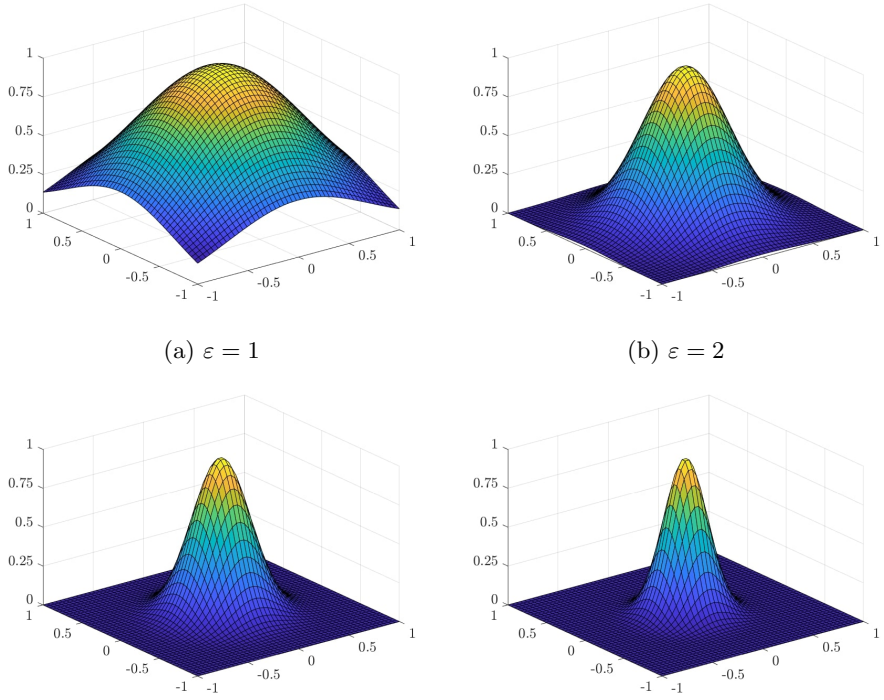
λ_{max} grows slowly as nodes are refined, therefore ill-conditioning issues must be related to a rapid growth of λ_{min}^{-1} [27]. In [74, 86] lower bounds for λ_{min} are provided as functions G_Φ of the separation distance $s(\mathcal{X})$, which gives smaller values as $s(\mathcal{X})$ decreases:

$$\lambda_{min} \geq G_\Phi(s(\mathcal{X})) \tag{2.47}$$

and, under the additional hypothesis of quasi uniform node distribution they are put in relation with a function F_Φ of the fill distance which satisfies $P^2_{\Phi,\mathcal{X}}(h_{\mathcal{X},\Omega}) \leq F_\Phi(h_{\mathcal{X},\Omega})$ ($P^2_{\Phi,\mathcal{X}}$ being the power function) [27]:

$$G_\Phi(s(\mathcal{X})) \leq \lambda_{min} \leq P^2_{\Phi,\mathcal{X}}(h_{\mathcal{X},\Omega}) \leq F_\Phi(h_{\mathcal{X},\Omega}) \tag{2.48}$$

If the interpolation problem is solved directly, it is not possible to arbitrarily increase accuracy by reducing $h_{\mathcal{X},\Omega}$ and have good conditioning at the same



(a) $\varepsilon = 1$ (b) $\varepsilon = 2$

(c) $\varepsilon = 3$ (d) $\varepsilon = 4$

Figure 2.1: Gaussian (GA) Radial Basis Function $\Phi = e^{-\varepsilon^2 \|\boldsymbol{x}\|^2}$ at different values of the shape parameter ε, as $\varepsilon \to 0$ the graph becomes more flat.

time: $s(\mathcal{X})$ will also be reduced and cause a proportional reduction of λ_{min}, eventually leading to instability. This fact is often called trade-off principle [27] or uncertainty principle [86, 12, 35]. In [12] it is stated as follows: it is impossible to construct radial basis functions which guarantee good stability and small errors at the same time.

In reality, while the error estimates on the approximation order, i.e. those derived from the power function, are unrelated from the conditioning of the interpolation problem and are unavoidable, numerical errors introduced by ill-conditioning can be mitigated [12] or even eliminated altogether [38, 37, 44, 35].

2.9 Flat limit for $\varepsilon \to 0$

By looking at the definitions of infinitely smooth RBFs given in Table 2.1, it is clear that $\varepsilon \to 0$ has similar effect as $r \to 0$ in practice (i.e. if neither is allowed to grow unbounded).

In Figure 2.1 the Gaussian (GA) radial basis function is visualized at various

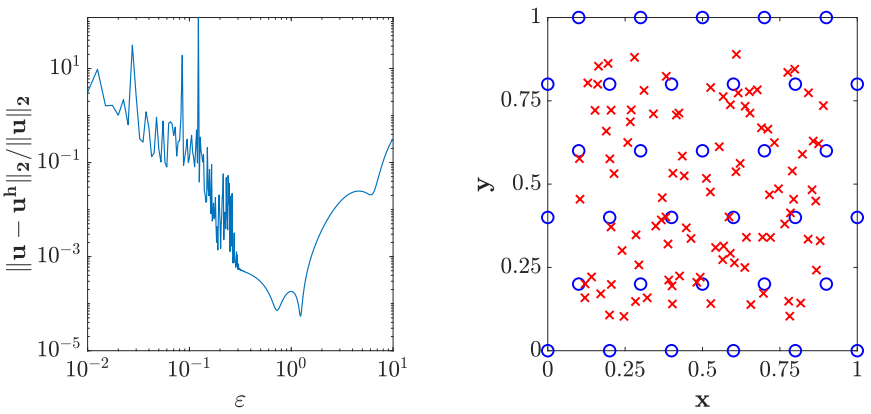


Figure 2.2: Error behavior at the flat limit on a 2D domain with Gaussian RBF. On the left the relative error is reported against the value of ε, $\boldsymbol{u}$ and $\boldsymbol{u^h}$ are to be interpreted as vectors, where each entry is associated to the value of the analytic function u or the interpolant u^h at a specific evaluation point. On the right the collocation points are depicted as blue circles and the evaluation points are represented by red crosses. The analytic function is $u(x, y) = sin(\pi x)\, sin(\pi y)$.

numbers of the shape parameter ε, other basis functions have similar behavior. It is possible to see that as $\varepsilon \to 0$, the graphs associated with the basic function $\phi(\varepsilon r)$ becomes more flat, intuitively it is as if the nodes were perceived as being closer. The first analysis of the error as $\varepsilon \to 0$ seems to be that in [99]: the approximation error rapidly decreases with ε until the associated linear system becomes too ill-conditioned and therefore unstable. This phenomenon is visible in Figure 2.2, where the approximation error is visualized against the value of ε in the case of the Gaussian RBF and is closely related to the trade-off principle from [86], discussed above.

As pointed out by Fornberg and others in many works [25, 41], ill-conditioning issues are particular to the RBF-Direct procedure: where the linear system (2.13) is solved using a square matrix M. Over time multiple alternative procedures have been developed by Fornberg *et al.* in order allow stable interpolation problems at very low values of the scaling parameter ε. It was observed that for node sets with some irregularity, the interpolant in the flat limit $\varepsilon \to 0$ will take the form of a multivariate polynomial [25, 41, 36], which is often the most accurate interpolation scheme in absence of the Runge phenomenon [40].

2.9.1 Eigenvalue patterns

Further understanding on the ill-conditioning issues of matrix $\boldsymbol{B}$ can be gained by looking at the pattern formed by its eigenvalues if an infinitely smooth basic function is employed. In the case of a 2D non periodic geometry the pattern is

[36]:

$$\begin{aligned} &\{\mathcal{O}(\varepsilon^0)\} \\ &\{\mathcal{O}(\varepsilon^2), \mathcal{O}(\varepsilon^2)\} \\ &\{\mathcal{O}(\varepsilon^4), \mathcal{O}(\varepsilon^4), \mathcal{O}(\varepsilon^4)\} \\ &\{\mathcal{O}(\varepsilon^6), \mathcal{O}(\varepsilon^6), \mathcal{O}(\varepsilon^6), \mathcal{O}(\varepsilon^6)\} \\ &\dots \end{aligned} \tag{2.49}$$

Which is intended to be read as follows: there is one eigenvalue which has order $\mathcal{O}(1)$, 2 with order $\mathcal{O}(\varepsilon^2)$, 3 with order $\mathcal{O}(\varepsilon^4)$ and so on. Different patterns are reported in [36] also for other geometries, in all of them the lower eigenvalues of $\boldsymbol{B}$ are proportional to high powers of ε. The worst case is also the most interesting, i.e. the one for non periodic 3D geometries: $1 \times \mathcal{O}(\varepsilon^0), 3 \times \mathcal{O}(\varepsilon^2), 6 \times \mathcal{O}(\varepsilon^4), 10 \times \mathcal{O}(\varepsilon^6), \dots$.

These patterns can be used to obtain an immediate estimate of $cond(\boldsymbol{B})$ and $det(\boldsymbol{B})$ and thus gaining a broad assessment of the stability of the associated interpolation process.

2.9.2 Stable Algorithms in the flat limit

Since the condition number is proportional to the sensitivities with respect to the numerical entries in the linear system (2.13), if no perturbation or uncertainty affects the right-hand-side or the matrix $\boldsymbol{M}$, then the only source of errors in the solution of the linear system is given by the round-off associated to the double precision arithmetic. A straightforward approach to address numerical ill-conditioning affecting the RBF-Direct solution is to use extended precision arithmetic [35, 14, 52, 85]. The increase in computational cost, however, is considerable even if higher precision hardware is used, if extended precision is available only by means of emulation software, the cost penalty already for stepping past the hardware-provided double precision (64 bit) is more likely to be in the 40 to 200 range or higher [35]. It is therefore clear that extended precision arithmetic, while possible, is not a practicable way for general purpose implementations aimed at solving Computational Fluid Dynamic problems of engineering interest. The goal of stable algorithms discussed in this subsection is therefore to find computational paths that are genuinely numerically well-conditioned all the way into the $\varepsilon \to 0$ limit and therefore requiring only standard double precision arithmetic no matter how small ε is [35].

In [12] a stable approach based on regularization is proposed and it works because of the special form of the degeneration affecting RBF linear systems discussed in 2.8.2: large eigenvalues usually remain moderate but in the flat limit smaller ones go to zero leading to bad conditioning [12]. The authors propose to circumvent this issue by opting for an approximate solution, achieved by calculating a singular value decomposition [47] first and then using only the subsystem corresponding to large singular values, an interested reader might wish to consult the referenced paper for implementation details. While the techniques is promising and has strong theoretical foundations, in my opinion it

is unpractical when paired with local RBF-FD or RBF-HFD approaches (cfr. chapters 4 and 7) because it requires the calculation of a certain number of singular values for each node lying in the interior of the domain. For this reason it will not be discussed in greater detail.

Other algorithms achieving stability at the flat limit are listed below.

- *Contour-Padé* [39]: this algorithm was developed for the Gaussian (GA) basic function. The starting point, shared with other stable algorithms, is to consider the interpolant $u^h(\boldsymbol{x}, \varepsilon)$ as a function of $\varepsilon \in \mathbb{C}$, i.e. ε is a complex number. Furthermore, $u^h(\boldsymbol{x}, 0)$ is known to be finite, and the authors conclude that $\varepsilon = 0$ is a removable singularity. To obtain $u^h(\boldsymbol{x}, \varepsilon)_{|\varepsilon=0}$, a set of values $u^h(\boldsymbol{x}, \varepsilon_i)$ are computed with ε_i lying on the periphery of a circle centered at the origin, then, if no poles lie inside the circle, the sought value can be taken as the average of the values around the periphery [35]. This algorithm is still affected by some limitations and is no longer advised by the author but it is nonetheless historically relevant since it was the first to achieve the stability in the flat limit and therefore paved the way for subsequent developments.

- *RBF-QR* [38]: was developed from the concept that the reason behind ill-conditioning of the interpolation process in the flat limit is due to the fact that the standard RBF basis is no longer a good basis because its element become almost indistinguishable from each other. In the RBF-QR the RBF basis is converted into a different one and then ε is sent to zero. In [38] this was done in the case of a spherical geometry by adopting a spherical harmonics expansion of radial functions like the ones in Table 2.1. Further developments of this method and applications to different geometries, but limited to the case of the Gaussian (GA) RBF, are also discussed in [29, 43].

- *RBF-GA* [44]: was developed as an alternative for the RBF-QR in the case of the GA radial function (hence the name) and it is still based on the idea of finding a better conditioned basis for its native space. We skip the explanation of this method, any discursive description which does not rely on equations would indeed end up being longer and probably more convoluted than the original or the one in [35]. To this day, this algorithm remains the fastest stable algorithm in the flat limit and in the 3D case it is about 10 times slower than the standard RBF-Direct approach.

- *RBF-RA* [113]: intended as an improvement over the initial Contour-Padé algorithm in terms of robustness, accuracy and ease of implementation, it is reportedly the most flexible in this list. It is based on the vector valued approximation of functions $f(\varepsilon) : \mathbb{C} \to \mathbb{C}^M$; in the case of RBF interpolation, f becomes the interpolant u^h, which is considered as a function of ε evaluated at a given set of M nodes. This algorithm applies to any type of smooth RBF, to any dimension and can be immediately integrated in the RBF-FD or RBF-HFD mehods (discussed in chapters 4 and 7).

Chapter 3

Node Generation

Traditional numerical methods for the solution of partial differential equations often rely on a mesh or a grid. In order to achieve the geometrical flexibility required for solving engineering problems involving complex geometries unstructured mesh are usually sought [79]. As mentioned above, however, even the most modern mesh generators require some expertise and the search for alternatives is open. In the case of meshless methods a set of nodes distributed over a computational domain is all that is required in order to perform a numerical simulation. Such a node distribution, while being more flexible than an unstructured mesh, is nonetheless required to satisfy some quality requirements as hinted by the results on approximation order. Furthermore, spatially variable nodes distribution must be allowed since it permits localized refinement where the solution is expected to vary rapidly. Therefore, while the distance between any two neighbouring nodes can vary in different portions of the domain, it is important that a local uniformity, as defined in Equation (2.43), is maintained.

For clarification, the terms *point* and *node* will be used as synonyms throughout the discussion and the same holds for *node generation*, *node distribution*, *point clouds*, *node initialization* or any other term indicating the placement of nodes within a domain and/or over its boundary.

3.1 Classification

Most of the algorithms for node generations can be categorized into either *mesh-based*, *iterative*, *advancing front* or *sphere-packing* [93]. In [97] other methods are also proposed, like random or quasi-random points or methods based on oversampling followed by thinning or, working in reverse, those based on undersampling followed by filling; these do not fall in one of the categories proposed above but are somewhat limited to lower dimensions or simpler geometries.

Mesh-based ones are usually employed because already available from traditional FEM preprocessors [65], they are generally not recommended because they apparently defeat one of the main reasons behind the use of meshless methods

and also for stability issues [90]. We remark, however, that when this solution is adopted there is no need for storing connectivity information and hence a point cloud can be saved simply in the form of a set of unrelated coordinates. This allows great freedom for modifications of node positions at a later stage and therefore an improvement in geometrical flexibility over standard mesh-based method is still achieved. As pointed out in [97], the use of a mesh generator in the context of meshless methods also makes sense in specific circumstances. For instance it makes easier to perform comparisons and validation against mesh-based solutions and it allows to recycle mesh quality metrics in the assessment of the quality of node distributions. Indeed, the idea of applying some metrics derived from those used in the context of mesh-based methods might have some advantages. At the moment, the assessment of the quality of node distributions and its impact over the stability and accuracy of RBF-based methods are still subject of research and usually different authors chose to adopt different metrics. A final reason for adopting a mesh-based approach for generating a node distribution, according to [97], is that the meshless algorithm might rely on some mesh-based calculations at some stage, like for some Lagrangian methods.

Examples of iterative approaches are those simulating the movement of charged particles minimizing some energy function defined via an inter-particle repulsion force like [49], other examples are processes based on Voronoi relaxation [2]. Iterative nodes generation processes are also those called *pre-simulations*, which start from an initial node placement and then use time integration procedures, instead of optimization, to obtain the desired point cloud [97], an example is the bubble simulation procedure [67]. Iterative node generation algorithms are affected by different flaws: are considered to be more computationally expensive and require an initial node distribution that satisfies certain quality constraints. On the other side, they are regarded as being a useful tool for improving the quality of an available distribution [34]. Therefore, while they are not advised as stand-alone solutions, they can be paired with an efficient algorithm that provides an almost optimal initial node placement.

Advancing front techniques seem to be regarded as the most promising methods and working implementations have been recently discussed in [93, 105]. Even though the implementation detail vary among different authors, the key idea is that, given the coordinates of a node, new neighboring nodes can be generated in its proximity so that an optimal packing is obtained locally. A node which is taken as reference to generate its neighbors is usually labelled as *active*, once the space surrounding an active node has been saturated, that node is removed from the set of active nodes. The reason behind the name is that while the algorithm is proceeding, the set of active nodes constitute the frontier separating the space which has already been filled with nodes from the empty portion of the domain. The definition of an initial active front can be done in multiple ways, in [93] it is stated that the active front can be initialized as the set of boundary nodes, a set of internal seed nodes or a union of both. Major advantages of this method derive from the fact the function responsible for the placement of new nodes operates locally, therefore very steep variation in the spacing between nodes can be easily obtained, for example by defining a spacing

function $s(\boldsymbol{x})$ which prescribes how far from the active node $\boldsymbol{x}$ new nodes must be placed. Furthermore, advancing front implementations proposed in [93, 105] are remarkably efficient and allow a good quality of the node placement, making further refinement based on iterative node repel only optional in the case of simple geometries. An important remark must be made on the boundary node generation: while a good packing is guaranteed in the interior of the domain, node generation at the boundary must be performed separately. This is important not only because boundary nodes can be used to initialize the active front [93], but also because the relative position of boundary nodes and interior nodes in their immediate proximity is of paramount importance for the correct enforcement of boundary conditions. In the author's opinion, the limitations of advancing front methods seem to be related to the initialization phase, in [93], where the initial set of active nodes is defined to be the boundary distribution, irregularities appear at the encounter of advancing fronts originating from surfaces with different orientations and seem to propagates from the edges of the boundary. In [105], where the distribution of internal nodes is entirely separate from that of boundary nodes, irregularities might appear in the proximity of the boundary. It seems reasonable to conclude that if we assume the domain to be arbitrarily complex, we need to give for granted that a certain number of node refinement iterations will be needed after an initial advancing front phase.

The last category of algorithms is that of the circle or sphere packing methods [62], a recent example is that in [90], where Poisson disk sampling [19] is used. This algorithm is also analysed in [93] and to this day is considered one of the best node generators to be paired with the RBF-FD method. Its major drawback is the lack of support for variable nodal spacing [93], which prevents its use in problems where sharp variations in the solutions are expected.

3.2 Performance Assessment

In order to make comparisons between different algorithms and advance the development of those already existing some metrics must be introduced that take into account not only the quality of node distributions but also geometrical flexibility and ease of use.

3.2.1 Node quality metrics

As highlighted above, assessing the quality of a node distributions is not an easy task due to the absence, at the best of author's knowledge, of a definitive mathematical model estimating the errors in the solution of PDEs as a function of the node placement. Error estimates from the theory of scattered data interpolations suggest that a first idea might be to minimize the fill distance for a given amount of nodes and this is achieved by a uniform distribution. On the other side it is often advantageous to reduce the node density (thus increasing the fill distance) and therefore the maximum theoretical accuracy, in areas where the solution varies slowly and increase it where sharp variations are expected.

The optimal node placement therefore seems to be the one that minimizes the fill distance, i.e. empty spaces in the domain, at a local level while allowing variations in the node density where needed. In 2D such a node placement can be achieved by making sure that at each point the nodes are scattered according to the hexagonal configuration and that variations in node density do not alter such a lattice configuration, Figure 3.1. In 3D the highest density arrangements are the face-centered cubic (FCC) and the hexagonal close-pack (HCP) lattice. Each of these 3D arrangements can be obtained by packing planes filled with 2D hexagonal node arrangements [123]. In order to assess how close to the hexagonal node placement a given distribution is, the most sensible metric in the author's opinion is that of the so-called *distance ratio* R_H. In the 2D case, at any given node $\boldsymbol{x}_i$, R_H is given by the ratio of maximum and minimum distances between the node $\boldsymbol{x}_i$ and each one of its 6 nearest neighbours $\boldsymbol{x}_j$, $j = 1, \ldots, 6$. In the 3D case, the same holds with the difference that the number of equally distanced neighbors from any point in expected to be 12, the 1D case is also included for completeness.

$$R_H(\boldsymbol{x}_i) = \begin{cases} \dfrac{\max\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^2} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\min\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^2} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2} & \text{in 1D} \\ \dfrac{\max\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^6} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\min\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^6} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2} & \text{in 2D} \\ \dfrac{\max\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^{12}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\min\limits_{\boldsymbol{x}_j \in \mathcal{X}_i^{12}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2} & \text{in 3D} \end{cases} \tag{3.1}$$

where $\mathcal{X}_i^6$ is the set made of the 6 closest neighbors to $\boldsymbol{x}_i$, $\mathcal{X}_i^{12}$ is the set of the 12 closest neighbors and $\mathcal{X}_i^2$ is the set of the 2 closest neighbors in the 1D case. Of course $R_H(\boldsymbol{x}_i) \geq 1$ and $R_H = 1$ holds in the ideal case of hexagonal (2D) or FCC/HCC (3D) lattices. A visualization of the 2D hexagonal lattice is provided in Figure 3.1 from [123], where circumference at different radii are drawn around a central node encompassing an increasing number of neighbors.

Another quality indicator is the *mean distance indicator* d_H, which can be defined similarly to R_H:

$$d_H(\boldsymbol{x}_i) = \begin{cases} \frac{1}{6} \sum_{j \in \mathcal{X}_i^6} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 & \text{in 2D} \\ \frac{1}{12} \sum_{j \in \mathcal{X}_i^{12}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 & \text{in 3D} \end{cases} \tag{3.2}$$

In the case of perfectly isotropic distributions achieving the prescribed spacing function $s(\boldsymbol{x})$ at each node the relation $d_H(\boldsymbol{x}_i) = s(\boldsymbol{x}_i)$ must hold. The ratio $d_H(\boldsymbol{x}_i)/s(\boldsymbol{x}_i)$ can therefore be used to assess how much the effective node distribution approximates the desired one. Even smaller numbers of neighbours can be used to define d_H, for example in [93] only the closest 3 neighbors are used.

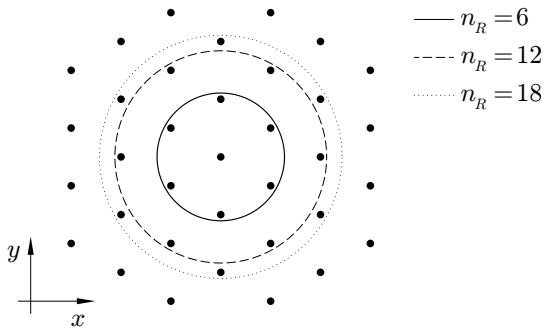


Figure 3.1: Hexagonal node two-dimensional lattice, from [123]

We conclude by remarking that node distributions satisfying the hexagonal lattice or the HCP which are locally isotropic (i.e. node distances do not vary as a function of direction) are stable with respect to node-repel iterative algorithms. Given an initial distribution, the application of one of such methods will eventually converge to a lattice which satisfies these node arrangements, provided that some containment strategy is implemented in order to prevent the nodes from escaping the domain.

3.2.2 Requirement list

An assessment of the performance of a node generation algorithm can be achieved by measuring how much it satisfies a certain number of requirements. In [93] the following list of properties is reported, with the remark that an ideal node-positioning algorithm should possess them all. Consequently, a qualitative but comprehensive performance assessment might be done by giving a certain algorithm a score in each one of the voices of the following list, loosely ordered by decreasing importance:

1. *Local regularity* Indicates the quality of the resulting node distribution, we suggest the use of the distance ratio R_H to derive a quantitative indicator.

2. *Minimal spacing guarantees* This is again a property of the node distribution, in case an RBF-based interpolation procedure is attempted, nodes that are too close might lead to ill-conditioning issues of the associated matrix, as explained in the previous chapter. While the ill-conditioning issues can be overcome adopting stable algorithms, it is nonetheless preferable to ensure that the minimum distance between any two nodes is never less than a certain threshold. The satisfaction of this requirement can be easily checked.

3. *Spatially variable densities* The algorithm should be able to generate distributions with spatially variable nodal spacing, which is usually defined by means of a scalar field $s(\boldsymbol{x}) : \Omega \to (0, \infty)$ [93]. The ratio $d_H(\boldsymbol{x}_i)/s(\boldsymbol{x}_i)$

then indicates the deviation from the prescribed spacing function at each node.

4. *Computational efficiency and scalability* In the ideal algorithm time complexity should scale linearly with respect to the number of generated nodes. The level of satisfaction for this requirement is checked by measuring how execution time varies as the number of generated nodes is increased.

5. *Compatibility with boundary discretization* The generated discretization of the whole domain should seamlessly join with the boundary discretization [93]. This can be quantified by measuring R_H for boundary nodes and comparing it with that of interior nodes, we remark that the difference in dimension must be taken into account: in case of 3D domain distributions R_H at the boundary must be calculated using the 2D formula, in case of 2D domain the 1D formula must be used, cfr. equation (3.1).

6. *Compatibility with irregular domains* This capability is tested by monitoring how much quality indicators like R_H or $d_H/s(\boldsymbol{x})$ degrade in such cases.

7. *Dimension independence* The perfect algorithm should be able to work in 2D and 3D providing equally good results.

8. *Direction independence* The orientation of the domain should not affect the node generation.

9. *No free parameters* In order to maximize the possibility of automation the algorithm should not require any tuning by the user and work *out of the box*. In case this is not possible default values for free parameters should work satisfactorily and if they are slightly changed this should not disrupt the correct execution.

10. *Simplicity* While this is a vague statement, we might interpret it in the sense that the algorithm should work correctly without requiring multiple modifications and additions for the treatment of special geometrical features or for addressing particular issues. In this sense simplicity is somewhat related to *Compatibility with irregular domains*.

3.3 Proposed node generation algorithm

The node generation algorithm adopted for this work is almost identical to the one proposed by Zamolo & Nobile in [116, 123, 118] based on the dithered Octree node generation followed by a node-repel refinement process. Here follows an introductory presentation of the method and a description of its performance according to the list proposed by Slak & Kosek in [93].

3.3.1 Underlying Principles

The underlying idea behind the design choices is that the node generation has to be used for the subsequent solution of Navier-Stokes partial differential equations over generic and arbitrarily complex 2D and 3D domains. In other words, the most importance must be given to the requirement number 6. Furthermore, because of the high cost of any CFD simulation, computational efficiency of the node generation process at point 4 can be considered less impacting on the overall cost, and thus much less important than points 5 and 6, since such requirements are those enabling the application of the CFD solver to engineering problems in the first place. Once such a rearrangement of priorities is done, it emerges that node repel iterative processes, which are the ones allowing higher quality of the node generation and greater geometrical flexibility should be considered as an integral part of the node generation algorithm rather than an optional addition.

An important remark on iterative node refinement techniques regards the effectiveness in the treatment of deviations from the prescribed spacing that are present in the initial node distribution. When such deviations appear at a high frequency, they are eliminated very efficiently and few iterations are required with a properly tuned implementation, on the contrary, deviations with low frequency are smoothed out very slowly. This fact was pointed out by Fornberg in [34] and in the 1D case is illustrated in Figure 3.2, taken from [123]. In this figure the target spacing function is constant and is satisfied when N nodes are equally spaced on the closed interval $[0, 1]$: $s(x) = 1/(N-1)$. In the picture above the initial node distribution is affected by a low frequency perturbation, in the one at the bottom a low frequency error is present. The deviation $\sigma[\Delta x]$ from the target spacing $s(x)$ is calculated as:

$$\sigma[\Delta x] = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}\left(\frac{\Delta x}{s(x)} - 1\right)^2} \tag{3.3}$$

As can be easily seen, the number of iterations required in order to smooth lower frequency errors is two to three order of magnitudes higher.

3.3.2 Quadtree\Octree

The fist stage of the node generation algorithm is constituted by the initial node placement, for this purpose a dithered Octree (called Quadtree in the 2D case) algorithm was chosen. This algorithm was originally developed for space partitioning [83, 84] and has already been applied to the node generation problem in [106].

General Description

For the implementation details we direct the reader to [116, 123, 118]. The operation of the 2D variant, i.e. quadtree, can be broadly understood by looking at Figure 3.3: the domain enclosed within Γ is recursively split into smaller

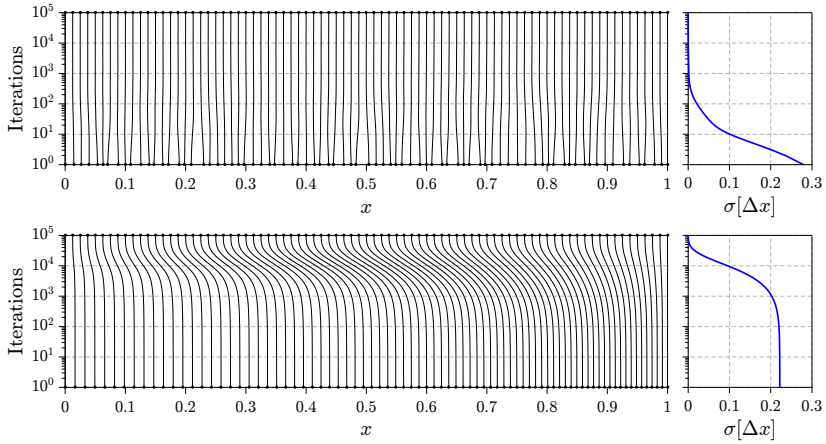


Figure 3.2: Evolution of the node-repel phase for 1D node distributions with an initially high-frequency (top) and low-frequency (bottom) error in space in the case of a constant spacing function and $N = 80$ nodes [123].

boxes until some nodes are placed in the smaller ones, called *leaf boxes* so that a spacing function $s(\boldsymbol{x})$ is locally approximated. In the case of Figure 3.3 the spacing function is prescribed to be bigger at the center, hence smaller boxes appear in that area producing a higher density of nodes. The algorithm proceeds as follows, at first a root box $\mathcal{B}$ containing the whole domain boundary Γ is constructed and then it is split into smaller ones, the split operation generate K smaller boxes every time, with $K = 4$ for the Quadtree and $K = 8$ for the Octree, as shown in Figure 3.4, thus creating a tree-like structure where the *leaf boxes* are the lowest level. The number of nodes calculated for each leaf box $n_b^e \in \mathbb{R}$ is such that a prescribed spacing function $s(\boldsymbol{x})$ is satisfied overall and is given by equation (3.4), where V_{box} is the volume (in 3D) or the surface (in 2D) of the box [123].

$$n_b^e = \int_{V_{box}} \frac{2}{\zeta s(\boldsymbol{x})^d} d\mu \tag{3.4}$$

where $\zeta = \sqrt{3}$ if $d = 3$, i.e. 3D case, $\zeta = \sqrt{2}$ if $d = 2$ and $\zeta = 2$ if $d = 1$. The splitting continues until the condition on the maximum numbers of nodes is met:

$$n_b^e \leq n_{max} \tag{3.5}$$

where $n_{max} \in \mathbb{N}$ is the maximum numbers of nodes allowed in a leaf box, in Figure 3.3 $n_{max} = 3$. When condition (3.5) is satisfied, an integer number of nodes $\lfloor n_b \rceil \in \mathbb{N}$ is placed inside the box. Finally, the difference $E_b = \lfloor n_b \rceil - n_b^e$ is called the quantization error and is diffused towards the neighboring cells according to the Floyd-Steinberg dithering algorithm [31].

The pseudocode for the Quadtree \ Octree algorithm is reported below in the form of a recursive function named GenerateNodesDitheredTree. In

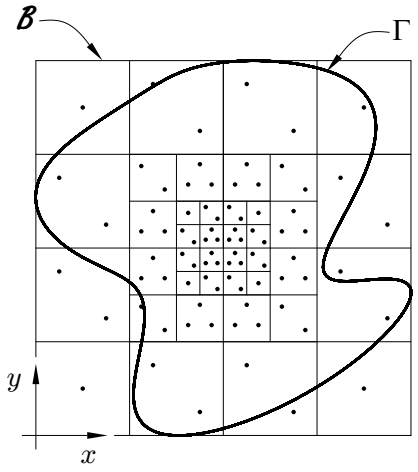


Figure 3.3: Quadtree node generation with $n_{max} = 3$, from [123]

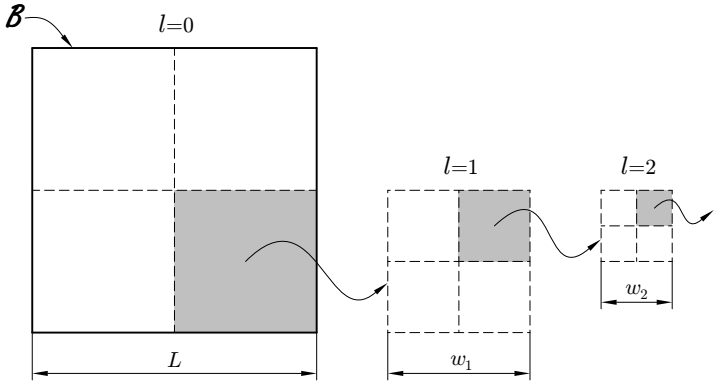


Figure 3.4: Recursive Quadtree splitting, from [123]

order to make it understandable we clarify some notation: M_s is the *numerosity variable* and for any box b it is defined as follows:

$$M_s(b) = \begin{cases} n_b & \text{if } b \text{ is a leaf box} \\ \sum_{k=1}^{K} M_s(b_k) & \text{if } b \text{ has child boxes } b_k \text{ with } k = 1, \dots, K \end{cases} \tag{3.6}$$

We have that $M_s(\mathcal{B}) = N$ for the root box $\mathcal{B}$, N being the total number of nodes. The dithering correction function DitheringTree$(N, b, \lfloor n_b \rceil)$ diffuses the nodal quantization error E_b by modifying the values of M_s for the unvisited neighbouring boxes according to the Floyd-Steinberg algorithm [31]. In order to ensure that the definition of M_s retains the summation property in equation (3.6), any modification in M_s is then propagated by recursively updating child boxes until the leaf level and parent boxes until the root level (this correction feature is assumed to be contained within DitheringTree for simplicity).

Algorithm 1 Pseudocode for the Quadtree \ Octree recursive function

```
1:  function GenerateNodesDitheredTree(N, b, M_s, n_b^e)
2:      n_b ← M_s(b)
3:      if n_b < n_b^e + 0.5 then
4:          InsertNodes(N, b, ⌊n_b⌉)
5:          M_s ← ⌊n_b⌉ − n_b^e
6:          DitheringTree(N, b, ⌊n_b⌉)
7:      else
8:          for all b_k child box of b do
9:              GenerateNodesDitheredTree(N, b_k, M_s, n_b^e)
10:         end for
11:     end if
12: end function
```

where InsertNodes$(N, b, \lfloor n_b \rceil)$ appropriately insert $\lfloor n_b \rceil$ in the box b when b is a leaf box.

Performance Description

If we were to apply this algorithm on its own, its performance assessment following the list proposed in [93] would be:

- we begin with the most important for our purposes: requirement number 6 is satisfied, not only it is possible to effectively generate nodes on complex domains defined analytically by means of a level set function, but it is also possible to obtain an initial node distribution for any 3D geometry contained within an `.stl` surface of whatever complexity like the one in Figure 3.5.
- requirements number 1, 3 and 2, which regard the quality of node distribution, are poorly fulfilled, this is particularly true for requirement number

1, since the attained lattice is far from the ideal isotropic distribution with $R_H = 1$, as can be seen in Figure 3.6 and Figure 3.8. Fulfillment of requirement number 3 is only partial, as can be seen in Figure 3.6, this is because while low frequency error is at an acceptable level, high frequency errors are present. This means that the spacing is correct at a larger scale but considerable errors are present at the scale of the hexagonal lattice. Furthermore, as the nodes are positioned in some pre-defined cartesian arrangement within each leaf box the R_H histogram shows some characteristic clustering and empty areas are present in some areas of the domain, see Figure 3.8. While this phenomenon can be mitigated by applying random perturbations to the node locations within each leaf box, it will never be eliminated completely.

- requirement number 4 is easily satisfied, indeed, computational complexity of the case of dithered Quadtree/Octree algorithm is $O(N \log N)$, N being the total number of nodes generated. If dithering is skipped a marginal speed-up can be achieved and the extra computational time required is well worth it since the dithering reduces the number of refinement iterations required at a later stage.

- requirement number 5 is not satisfied at all since no boundary distribution is provided by the algorithm, only nodes belonging to the interior are kept and the ones falling outside of the domain are deleted.

- requirements number 7 and 8 are largely fulfilled since the same algorithm, with some obvious modifications manageable within the code at no additional computational cost, can work in 2 and 3 dimensions and no difficulty is introduced if the domain is rotated although the node placement may vary slightly.

- requirement 9 is largely satisfied since the only free parameter is the number of nodes in each leaf box n_{max}, however this has no major impact on the correct execution and the value $n_{max} = 2^d - 1$ can be used leading to optimal results, with $d = 2$ in 2D and $d = 3$ in 3D. The same can be said about requirement 10, while the implementation is somewhat delicate and not straightfarward in the case of geometries defined by means of `.stl` files, there is no need to implement modifications for dealing with special cases.

3.3.3 Node Repel Refinement

Once an initial node distribution has been attained by means of the Quadtree\Octree algorithm discussed above, a node-repel refinement is carried out in order to improve the quality of the distribution and project nodes on the boundary. This belongs to the class of iterative node generation algorithms, and more specifically to the ones aimed at minimizing an energy function.

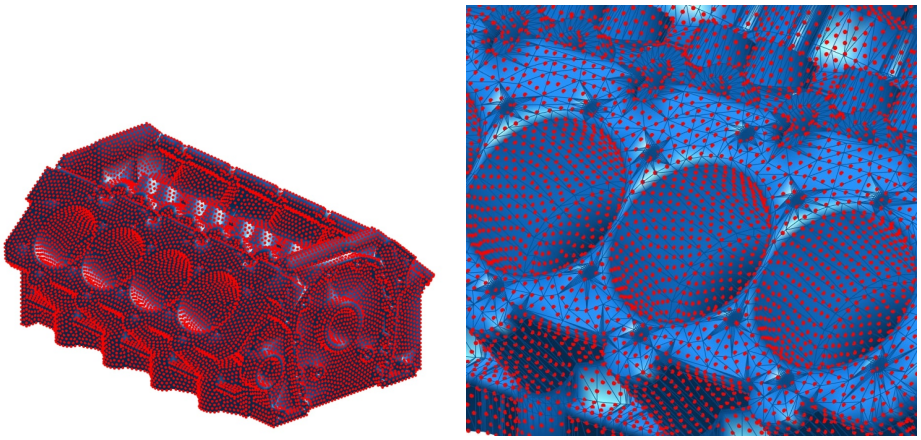


Figure 3.5: example of a complex geometry discretized by means of the dithered Octree algorithm followed by iterative node-repel refinement, the surface nodes are visible on the `.stl` surface, from [73]

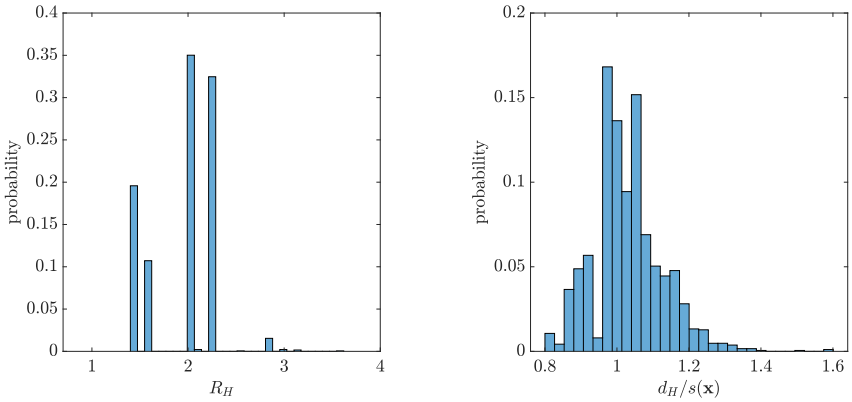


Figure 3.6: Histograms used for quality assessment of the node distribution provided by the Quadtree algorithm on the 2D square geometry $[0,1]^2$ with uniform spacing $s(\boldsymbol{x}) = 0.025$. Probability of bin value v_i is calculated as c_i/N, where c_i is the number of nodes in the bin i, N is total number of nodes, 30 bins are used.

General Description

The iterative process scan every node $\boldsymbol{x}_i$ and for each one finds the m nearest neighbors $\mathcal{X}_i^m = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, some of them might belong to the boundary $\partial\Omega$ and some to the interior $\Omega \setminus \partial\Omega$, then proceed to displace $\boldsymbol{x}_i$ according to a repulsive force. At iteration k the displacement of node $\boldsymbol{x}_i^{(k)}$ normalized by the spacing function is given by:

$$\frac{\boldsymbol{x}_i^{(k+1)} - \boldsymbol{x}_i^{(k)}}{s(\boldsymbol{x}_i^{(k)})} = -\frac{1}{\alpha}\left(\sum_{\boldsymbol{x}_j \in \mathcal{X}_i^m} F(r_{ij}^{(k)}) \frac{\left(\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\right)}{\|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\|_2}\right) \tag{3.7}$$

where $r_{ij}^{(k)} = \|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\|_2 / s(\boldsymbol{x}_i^{(k)})$ is the normalized radius and $F(r)$ is a radial basic function defined as:

$$F(r) = \frac{1}{(r^2 + \beta)^2} \tag{3.8}$$

Since the distances between nodes r_{ij} of equation (3.7) are normalized with respect to the prescribed spacing function $s(\boldsymbol{x})$, any localized concentration of nodes prescribed by $s(\boldsymbol{x})$ will be preserved and satisfied throughout the node repel refinement.

The parameters α in equation (3.7) and β in equation (3.8) are free parameters: $1/\alpha$ acts as a gain factor which has to be tuned by trial and error in order to maximize the convergence speed while avoiding instabilities [123] and β prevents an infinite force when two nodes are superposed. The two are not completely independent: large values of α make the refinement process slow but robust and there is little to no need to define β, smaller values of α increase the speed of convergence but reduce the stability and hence larger values of β are required. We remark that since $r_{ij}^{(k)}$ is normalized with respect to the spacing functions the two parameters can be chosen once and for all, suggested values are: $\alpha = 10$ and $\beta = 0.2$ and can be safely tuned. Another free parameter is m, i.e. the number of neighbors to be chosen in the nearest neighbors search, too little numbers reduce the computational burden but slow down the convergence while larger numbers make the algorithm inefficient. Safe choices are $m = 20$ for the 2D case and $m = 30$ in the 3D case, this parameter can also be tuned safely in the intervals $m \in [15, 25]$ for 2D and $m \in [20, 37]$ for 3D.

In order to prevent the nodes from leaving the domain under the action of unbalanced repulsion forces a containment strategy is required. This is attained by checking whether any new position $\boldsymbol{x}_i^{(k+1)}$ lies outside the domain and, if so, by projecting it on the boundary $\partial\Omega$. Once $\boldsymbol{x}_i$ is projected on the boundary, it will systematically be projected at every iteration without performing further checks and will continue to exert repulsive forces on its neighbors. This ensures an efficient containment as long as α is small enough, furthermore, mutual repulsion between boundary nodes ensures a good cover of very complicated domains and a perfect compatibility between boundary and interior discretization.

Performance Description

At last, here is the evaluation of the resulting node generation process following the list from [93]:

- The adoption of an iterative algorithm consistently ensures very high quality in the node arrangement when this is evaluated using the proposed indices R_H and d_H. Thus points 1, 2 and 3 are all fulfilled, as is also evident from Figure 3.7 and Figure 3.8.
- The fulfillment of requirement number 4 is penalized by the iterative nature of the node-repel refinement, which makes it less computationally efficient. Nonetheless, this can be contextualized by considering the following remarks:
  - only high frequency deviations from the optimal HCP positioning of the nodes are left after the execution of the Octree algorithm, thus the number of iteration needed is minimized by the initial node placement,
  - the execution of the node-repel refinement gives a boundary distribution which also satisfies the hexagonal lattice as a byproduct, therefore there is no need for a separate algorithm that initializes boundary nodes,
  - some node-repel refinement is also presumably needed in most practical cases, even when advancing front or sphere packing algorithms are employed, indeed in [93] some iterations of it are given for granted and excluded from the comparison between algorithms. Probably other methods require a smaller number of such iterations but they often need a separate generation of a boundary distribution,
  - the amount of time required for the node generation process including the node-repel refinement will ultimately be almost negligible relative to that consumed by the solution of Navier-Stokes equations.

  A measurement of times required for a complete node generations have been included in the following subsection 3.3.3.
- Compatibility requirements 5 and 6 are fulfilled, as well as direction independence 8.
- Dimension independence, i.e. requirement number 7 remains the same as that of the Quadtree\Octree algorithm.
- Requirement 9 prescribes the limitation of the number of free parameters needed for the correct execution of the algorithm. While the node-repel refinement process requires 3 parameters, experiments done so far suggest that fixed values can be used safely regardless of the shape of the domain or the spacing function and that small changes in their values influence the speed of convergence but not the final result in any appreciable way.

- Finally, the proposed algorithm fulfill the requirement number 10 on simplicity because it does not require any separate treatment of boundary nodes and relies on the combination of well established techniques.

Any additional detail concerning implementation details or performance measure can be found in [116, 123, 118].

Computational Times

In order to assess the computational performances of the proposed algorithm here follows a brief prsentation of the actual computational times required for an execution using the Julia programming language [6] and an optimized C implementation written by Zamolo for [123]. This was also done in order to highlight the suitability of the presented algorithm to be executed in parallel on multiple threads.

The following results have been obtained in the case of a unit ball domain, i.e., $f(\boldsymbol{x}) = 1 - \|\boldsymbol{x}\|$, and a spacing function $s(\boldsymbol{x}) \propto (\|\boldsymbol{x}\|^2 + 1)^{-1}$. The chosen number of neighboring nodes for the refinement phase is $m = 13$ in 2D and $m = 36$ in 3D, whith 100 refinement iterations. Two examples of Julia-generated node distributions in 2D and 3D are depicted in Figure 3.9.

An important remark is that the times in seconds (s) reported in Table 3.1 are referred to the sum of the times taken by the Quadtree\Octree algorithm and those taken by the node-repel refinement phase. However, the initial node placement achieved by means of the Quadtree\Octree algorithm is usually completed in less than a second. More specifically, in the case of $N = 10^6$ and single core, the time required is $0.33s$ for Quadtree and $1.45s$ for Octree.

Multi-core parallelism for the `for` loops over the N nodes in the refinement phase is achieved by using `Distributed` and `Threads` modules with `@threads` macro in Julia, while OpenMP API with `#pragma omp` directives have been employed for the C code. Both codes have been run on a modern laptop equipped with an Intel® i7 2.6GHz processor with 4 cores (8 threads).

A performance comparison between Julia and C implementations is also reported in Table 3.1 in terms of computing times and speedup when using more than 1 thread, for both 2D and 3D cases, for different number of nodes N and for different number of threads. Single thread computing times for the Julia implementation are $30\% - 50\%$ higher than those with C, while this performance gap increases when using multiple threads, encountering the worst performance for the Julia implementation for the 2D case with the smallest number of nodes $N = 50,000$ and with the maximum number of threads, i.e., 8. This fact can be better appreciated by comparing the speedup values: the C (OpenMP) speedup is always larger than the Julia one (Threads module), with larger differences for small N and large number of threads. Furthermore, the C speedup is almost independent upon the number of nodes N and upon the dimensionality d, while the Julia speedup shows an asymptotical behaviour, tending to the C speedup for large values of N.

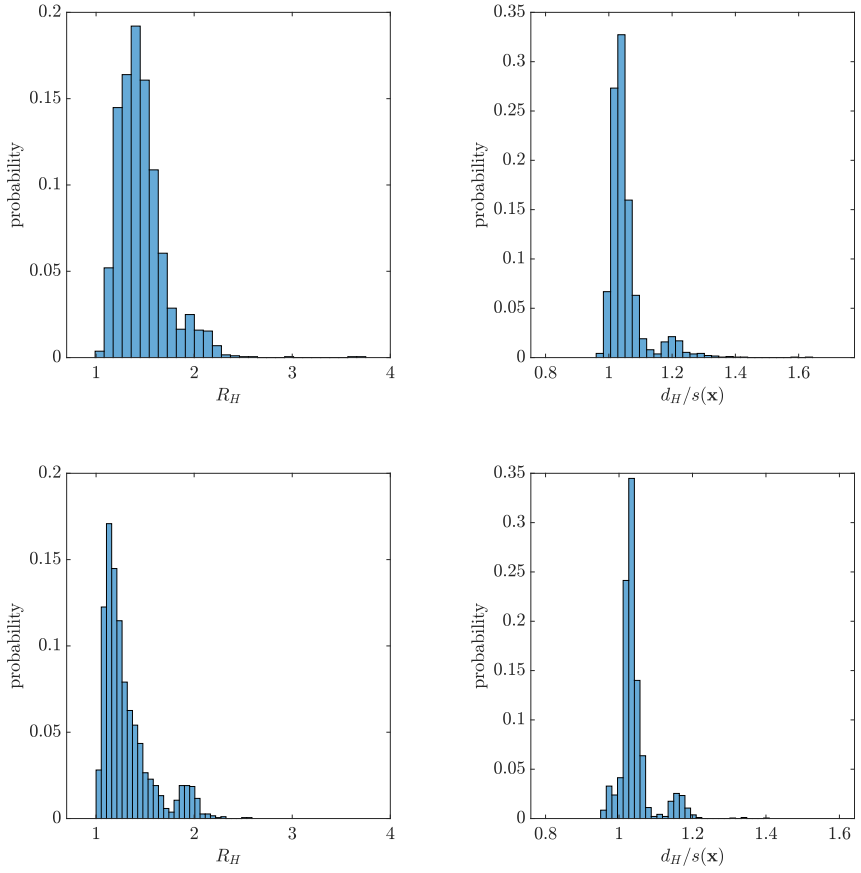


Figure 3.7: Histograms used for quality assessment of the node distribution provided by the node-repel refinement algorithm on the 2D square geometry $[0,1]^2$ with uniform spacing $s(\boldsymbol{x}) = 0.025$. Upper row refers to the results attained after 20 iterations, the bottom one after 200 iterations. Probability of bin value v_i is calculated as c_i/N, where c_i is the number of nodes in the bin i, N is total number of nodes, 30 bins are used. The columns in the bottom row appear narrower because the same number of bins is used to cluster a narrower distribution.

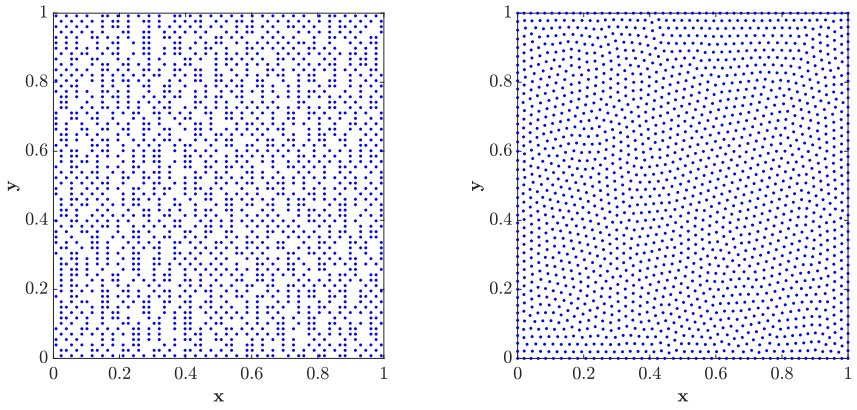
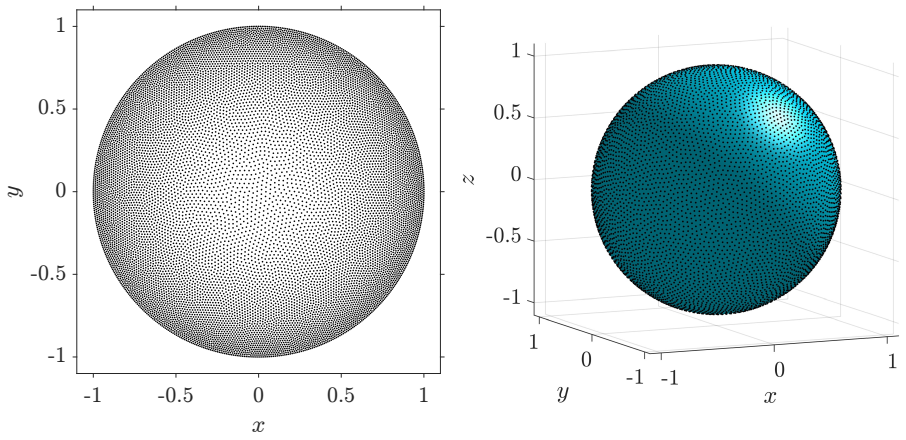


Figure 3.8: Distributions of nodes in the square domain $[0,1]^2$ with uniform spacing $s(\boldsymbol{x}) = 0.025$. On the left the nodes generated by the Quadtree algorithm, on the right those obtained after 200 iterations of the node-repel refinement algorithm.



(a) (b)

Figure 3.9: Examples of node distributions achieved after node-refinement: $N = 10,000$ nodes inside a 2D circle (a), $N = 40,000$ nodes inside a 3D sphere (b).

3.3.4 Conclusions

The proposed node generation algorithm was selected primarily for its geometrical flexibility and robustness when adopted for very complex geometries. The penalty in terms of computational efficiency due to the reliance on an iterative node-repel

refinement technique is justified by the quality of the final result. Ultimately, overall computational times of few seconds for a multithreaded execution can be considered more than satisfactory for practical applications where the solution of simple natural convection problems can take multiple minutes.

The implementation written in the Julia has been adopted as the starting point for the future development of 3D solvers based on the RBF-FD and RBF-HFD methods.

Table 3.1: computing times (s) and speedup, Julia code vs. (C code).

	N	Number of threads			
		1	2	4	8
2D	50,000	3.07 (2.45)	1.95 (1.25)	1.41 (0.81)	1.12 (0.49)
			1.6 (2.0)	2.2 (3.0)	2.7 (5.0)
	100,000	6.17 (4.63)	3.58 (2.53)	2.60 (1.61)	1.85 (1.00)
			1.7 (1.8)	2.4 (2.9)	3.3 (4.6)
	300,000	19.1 (14.2)	10.9 (7.70)	6.84 (4.92)	5.02 (2.89)
			1.8 (1.8)	2.8 (2.9)	3.8 (4.9)
3D	50,000	12.0 (7.90)	6.62 (4.26)	4.26 (2.78)	2.97 (1.62)
			1.8 (1.9)	2.8 (3.0)	4.0 (4.9)
	100,000	21.1 (16.0)	11.6 (8.37)	7.23 (5.54)	5.05 (3.18)
			1.8 (1.9)	2.9 (2.8)	4.2 (5.0)

Chapter 4

RBF-FD method

As anticipated, the theory of interpolation of scattered data can be applied to the solution of partial differential equations. In this chapter the Radial Basis Function-Finite Difference method is presented. In the following discussion it is supposed that a point cloud consisting of N collocation nodes has been generated in the domain of calculus Ω and on its boundary $\partial\Omega$ using a suitable node generation technique.

4.1 Global RBF-based methods

The first to propose the use of analytic derivatives of radial basis functions was Kansa in [54, 55], the resulting method became known as *Kansa's* or *unsymmetric* formulation.

We recall that the theory of scattered data interpolation allows to approximate a scalar field $u(\boldsymbol{x})$, which is known at a set of points $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \Omega$, with a function $u^h(\boldsymbol{x})$ defined as in equation (2.16), that is repeated here for convenience:

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \Phi(\boldsymbol{x} - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \tag{4.1}$$

We also recall that the vector of coefficients $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_N\}$ and $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_M\}$ are found by solving equation (2.13):

$$\underbrace{\begin{pmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{pmatrix} \tag{4.2}$$

Suppose now that the following linear PDE is given:

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega \\ \mathcal{B}u = g & \text{on } \partial\Omega \end{cases} \tag{4.3}$$

where $\mathcal{L}$ and $\mathcal{B}$ are linear operators, $\mathcal{L}$ is the partial differential operator acting on function u in the interior of Ω, whereas $\mathcal{B}$ is the operator enforcing some boundary condition (BC) and does not necessarily involve partial derivatives. f and g are known functions.

The most commonly encountered BC in numerical simulations involving fluid motions or heat exchange are:

$$\text{Dirichlet BC:} \qquad u = g \tag{4.4}$$

$$\text{Neumann BC:} \qquad \frac{\partial u}{\partial \boldsymbol{n}} = g \tag{4.5}$$

$$\text{Robin BC:} \qquad au + b\frac{\partial u}{\partial \boldsymbol{n}} = g \tag{4.6}$$

where ${}^{\partial u}/_{\partial \boldsymbol{n}}$ is the normal derivative of u and a and b in (4.6) are known functions. The Robin BC at a point $\boldsymbol{x}$ becomes Dirichlet if $b(\boldsymbol{x}) = 0$ and Neumann if $a(\boldsymbol{x}) = 0$.

From the theory of interpolation we know that u^h given by equation (2.16) is the best approximation of u in the native space $\mathcal{H}$ associated with Φ which satisfies interpolation conditions [27]. Since $\mathcal{L}u$ and $\mathcal{B}u$ are the known functions at the set of collocation nodes $\mathcal{X}$, it is natural to use the theory of interpolation on $\mathcal{L}u(\boldsymbol{x})$ itself. The idea behind Kansa's formulation is therefore to look for a function $u^h(\boldsymbol{x})$ such that $\mathcal{L}u^h(\boldsymbol{x}) \approx \mathcal{L}u(\boldsymbol{x})$ in Ω and to also require that: $u^h(\boldsymbol{x}) = u(\boldsymbol{x})$ if $\boldsymbol{x}$ is a collocation node inside Ω and $\mathcal{B}u^h(\boldsymbol{x}) = \mathcal{B}u(\boldsymbol{x})$ if $\boldsymbol{x}$ is a node generated on $\partial\Omega$.

When solving an interpolation problem there is usually no reason for dealing with BC, but they are unavoidable in boundary value problems and sometimes more complicated than the Robin BC of equation (4.6). In order to enforce them it is recommended to split the set of collocation nodes $\mathcal{X}$ into the set $\mathcal{X}_I$ of the N_I nodes lying inside Ω and the set $\mathcal{X}_B$ of the N_B nodes lying on the boundary. From now on we will number the nodes so that the first N_I are internal and the last N_B lie on the boundary, with $N = N_I + N_B$

If we apply the linear operator $\mathcal{L}$ to the definition of u^h given in equation (2.16) and evaluate the resulting $\mathcal{L}u^h$ at a generic point $\boldsymbol{x}$, we have that by linearity $\mathcal{L}$ distributes over the basis functions (Radial and polynomial) and gives:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}) &= \sum_{j=1}^{N} \alpha_j \mathcal{L}\Phi(\boldsymbol{x} - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k \mathcal{L}p_k(\boldsymbol{x}) \\ &= \begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{\beta} \end{pmatrix} \begin{pmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} \end{aligned} \tag{4.7}$$

where $\mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}) = \{\mathcal{L}\Phi(\boldsymbol{x}-\boldsymbol{x}_1), \ldots, \mathcal{L}\Phi(\boldsymbol{x}-\boldsymbol{x}_N)\}$ and $\mathcal{L}\boldsymbol{p} = \{\mathcal{L}p_1(\boldsymbol{x}), \ldots, \mathcal{L}p_M(\boldsymbol{x})\}$, and the same hold for linear operator $\mathcal{B}$ associated with boundary conditions.

We remark that $\mathcal{L}\Phi(\cdot, \boldsymbol{x}_i)$ still lies in the native space $\mathcal{H}$ of Φ for all linear differential operators and infinitely differentiable Radial Basis functions of Table

2.1. Therefore $\mathcal{L}u^h \in \mathcal{H}$ is determined as soon as we calculate the coefficient vectors $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}$ and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_M\}$.

Conditions (4.3) could be readily enforced by collocation, however, the real strength in the unsymmetric formulation lies in the possibility of attaining a Finite Difference-like discretization of $\mathcal{L}$. In order to do so, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ should be found by enforcing the following collocation conditions on u^h:

$$\begin{aligned} u^h(\boldsymbol{x}_i) &= u(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \\ \mathcal{B}u^h(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_B \end{aligned} \tag{4.8}$$

Which take the following matrix form:

$$\underbrace{\begin{pmatrix} \boldsymbol{\Phi_I} & \boldsymbol{P_I} \\ \boldsymbol{\mathcal{B}\Phi_B} & \boldsymbol{\mathcal{B}P_B} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M_{BC}}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_I} \\ \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \tag{4.9}$$

where $\boldsymbol{u_I} = \{u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_{N_I})\}$, $\boldsymbol{g} = \{g(\boldsymbol{x}_{N_{I+1}}), \ldots, g(\boldsymbol{x}_N)\}$ and new terms in matrix $\boldsymbol{M_{BC}}$ are defined as follows:

$$\begin{aligned} \boldsymbol{\Phi_I} &= \begin{pmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{N_I}, \boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_{N_I}, \boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N_I, N} \\ \boldsymbol{P_I} &= \begin{pmatrix} p_1(\boldsymbol{x}_1) & \ldots & p_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_{N_I}) & \ldots & p_M(\boldsymbol{x}_{N_I}) \end{pmatrix} \in \mathbb{R}^{N_I, M} \\ \boldsymbol{\mathcal{B}\Phi_B} &= \begin{pmatrix} \mathcal{B}\Phi(\boldsymbol{x}_{N_I+1}, \boldsymbol{x}_1) & \ldots & \mathcal{B}\Phi(\boldsymbol{x}_{N_I+1}, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ \mathcal{B}\Phi(\boldsymbol{x}_N, \boldsymbol{x}_1) & \ldots & \mathcal{B}\Phi(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N_B, N} \\ \boldsymbol{\mathcal{B}P_B} &= \begin{pmatrix} \mathcal{B}p_1(\boldsymbol{x}_{N_I+1}) & \ldots & \mathcal{B}p_M(\boldsymbol{x}_{N_I+1}) \\ \vdots & \ddots & \vdots \\ \mathcal{B}p_1(\boldsymbol{x}_N) & \ldots & \mathcal{B}p_M(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N_B, M} \end{aligned} \tag{4.10}$$

When Dirichlet BC are enforced, operator $\mathcal{B}$ is simply the identity operator and the linear system (4.9) coincides with that associated with the interpolation problem of equation (2.13). In that case the problem is ensured to be uniquely solvable regardless of the shape of domain Ω, this however does not hold in case of Neumann BC, as will be discussed in the following chapter.

It is now possible to substitute $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into equation (4.7) and thus obtain:

$$\mathcal{L}u^h(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{u_I} & \boldsymbol{g} & 0 \end{pmatrix} {\boldsymbol{M_{BC}}}^{-T} \begin{pmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} \tag{4.11}$$

It follows that the discrete form of the linear partial differential operator $\mathcal{L}$ is given by the following Finite Difference formulation:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}) &= \boldsymbol{c_I}(\boldsymbol{x})^T \boldsymbol{u_I} + \boldsymbol{c_B}(\boldsymbol{x})^T \boldsymbol{g} \\ &= \sum_{j=1}^{N_I} c_j(\boldsymbol{x}) u(\boldsymbol{x}_j) + \sum_{k=N_{I+1}}^{N} c_k(\boldsymbol{x}) g(\boldsymbol{x}_k) \end{aligned} \tag{4.12}$$

where coefficient vectors $\boldsymbol{c_I}(\boldsymbol{x})$ and $\boldsymbol{c_B}(\boldsymbol{x})$ are found by solving the dual problem:

$${\boldsymbol{M_{BC}}}^T \begin{pmatrix} \boldsymbol{c_I}(\boldsymbol{x}) \\ \boldsymbol{c_B}(\boldsymbol{x}) \\ \boldsymbol{c_p}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}) \end{pmatrix} \tag{4.13}$$

Proceeding with the solution of boundary value problem (4.3), the unknown values associated with the N_I interior points $\mathcal{X}_I$ can now be found by enforcing the following N_I collocation conditions, i.e. requiring u^h to approximate the exact solution at each point $x_i \in \mathcal{X}_I$:

$$\mathcal{L}u^h(\boldsymbol{x}_i) = \mathcal{L}u(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \tag{4.14}$$

Which takes the matrix form:

$$\boldsymbol{C_I} \begin{pmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{pmatrix} = \begin{pmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_{N_I}) \end{pmatrix} - \boldsymbol{C_B} \begin{pmatrix} g(\boldsymbol{x}_{N_{I+1}}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{pmatrix} \tag{4.15}$$

where matrices $\boldsymbol{C_I}$ and $\boldsymbol{C_B}$ are formed by row vectors $\boldsymbol{c_I}(\boldsymbol{x}_i)^T$ and $\boldsymbol{c_B}(\boldsymbol{x}_i)^T$ respectively, found by solving equation (4.13) N_I times:

$$\begin{aligned} \boldsymbol{C_I} &= \begin{pmatrix} c_1(\boldsymbol{x}_1) & \dots & c_{N_I}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_1(\boldsymbol{x}_{N_I}) & \dots & c_{N_I}(\boldsymbol{x}_{N_I}) \end{pmatrix} \in \mathbb{R}^{N_I, N_I} \\ \boldsymbol{C_B} &= \begin{pmatrix} c_{N_I+1}(\boldsymbol{x}_1) & \dots & c_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_{N_I+1}(\boldsymbol{x}_{N_I}) & \dots & c_N(\boldsymbol{x}_{N_I}) \end{pmatrix} \in \mathbb{R}^{N_I, N_B} \end{aligned} \tag{4.16}$$

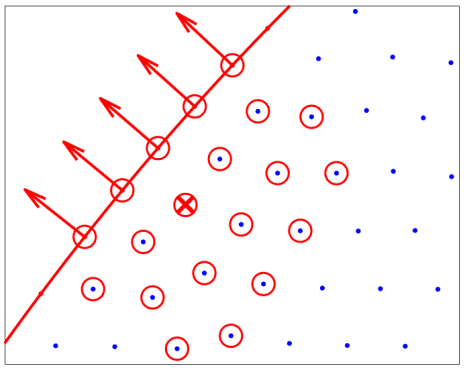


Figure 4.1: Example of a 2D stencil

Equation (4.15) can finally be solved using an iterative program to find the unknown values $\{u^h(\boldsymbol{x}_1), \dots, u^h(\boldsymbol{x}_{N_I})\}$.

When the number of nodes N is very high, however, this method becomes highly inefficient because the solution of system (4.13) has to be repeated for each node $\boldsymbol{x}_i \in \mathcal{X}_I$, every time with a cost of at least $\mathcal{O}(N^3)$. Furthermore, the resulting matrix $\boldsymbol{C_I}$ is full even if derivatives are local properties of functions. Computational efficiency issues make global approaches like the one discussed so far extremely impractical and are the reason why local methods, like the RBF-FD were introduced [35]. Computational efficiency could be improved if conditions (4.3) were enforced directly using collocation techniques, however in that case there is no warranty on the solvability of the resulting linear system.

4.2 RBF-FD formulation

The first to introduce the Radial Basis Function-Finite Difference (RBF-FD) method seems to have been Tolstykh in [100], and the method was also treated in early works [101, 91, 11, 111]. In recent years the RBF-FD approach is being developed and applied with success [9, 10, 57, 58].

The main difference from the global formulation lies in the introduction of a *stencil*. Given a node $\boldsymbol{x}_i \in \mathcal{X}_I$, the stencil associated to $\boldsymbol{x}_i$ is the set $\mathcal{X}_i \subseteq \mathcal{X}$ formed by its neighbors, which may lie in the interior Ω or on the boundary $\partial\Omega$ of the domain. An example of a stencil in 2D is depicted in Figure 4.1, where the central node is marked with a red cross while its neighbors included in the stencil are marked with a red circle. The stencil nodes which belong to $\partial\Omega$ are associated to boundary normals $\boldsymbol{n}$ depicted as red arrows.

The solution of a boundary value problem by means of the RBF-FD method proceeds as explained for global RBF-based methods in the previous section, with the important difference that the interpolation scheme is now local, i.e. $u^h(\boldsymbol{x})$ is expanded locally using a basis that changes depending on the position $\boldsymbol{x}$. Given a point $\boldsymbol{x}_i$, a stencil of m nodes $\mathcal{X}_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ around $\boldsymbol{x}_i$ must be

found. Stencil nodes $\mathcal{X}_i$ are split in two groups: the first m_I lie in the interior and form the set $\mathcal{X}_{i,I} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m_I}\}$, whereas the following m_B lie on the boundary and form the set $\mathcal{X}_{i,B} = \{\boldsymbol{x}_{m_I+1}, \dots, \boldsymbol{x}_m\}$, with $m = m_I + m_B$. $u^h(\boldsymbol{x})$ can then be expressed as:

$$u^h(\boldsymbol{x}_i) = \sum_{j=1}^{m} \alpha_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}_i) \tag{4.17}$$

With this initial remark, Kansa's formulation can be adopted almost unchanged, with the difference that equation (4.7) now holds locally at each interior point $\boldsymbol{x}_i \in \mathcal{X}_I$:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \sum_{j=1}^{m} \alpha_j \mathcal{L}\Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k \mathcal{L}p_k(\boldsymbol{x}_i) \\ &= \begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{\beta} \end{pmatrix} \begin{pmatrix} \mathcal{L}\Phi(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \end{aligned} \tag{4.18}$$

As a consequence, different vector of coefficients $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_m\} \in \mathbb{R}^m$ and $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_M\} \in \mathbb{R}^M$ are now to be found for each $\boldsymbol{x}_i$. This also means that internal and boundary conditions (4.8) are enforced locally. Equation (4.9) can still be written with the same notation, paying attention that now $\boldsymbol{u_I}$ is the unknown field at the stencil nodes $\boldsymbol{u_I} = \{u(\boldsymbol{x}_1), \dots, u(\boldsymbol{x}_{m_I})\}$ and $\boldsymbol{g}$ is the boundary condition enforced on neighbors belonging to $\mathcal{X}_{i,B}$: $\boldsymbol{g} = \{g(\boldsymbol{x}_{m_I+1}), \dots, g(\boldsymbol{x}_m)\}$. Equation (4.12) becomes:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \boldsymbol{c_I}(\boldsymbol{x}_i)^T \boldsymbol{u_I} + \boldsymbol{c_B}(\boldsymbol{x}_i)^T \boldsymbol{g} \\ &= \sum_{j=1}^{m_I} c_j(\boldsymbol{x}_i) u(\boldsymbol{x}_j) + \sum_{k=m_{I+1}}^{m} c_k(\boldsymbol{x}_i) g(\boldsymbol{x}_k) \end{aligned} \tag{4.19}$$

with coefficient vectors $\boldsymbol{c_I} \in \mathbb{R}^{m_I}$ and $\boldsymbol{c_B} \in \mathbb{R}^{m_B}$ defined as the solution of the local problem:

$$\boldsymbol{M_{BC}}^T \begin{pmatrix} \boldsymbol{c_I}(\boldsymbol{x}_i) \\ \boldsymbol{c_B}(\boldsymbol{x}_i) \\ \boldsymbol{c_p}(\boldsymbol{x}_i) \end{pmatrix} = \begin{pmatrix} \mathcal{L}\Phi(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \tag{4.20}$$

where the size of matrix $\boldsymbol{M_{BC}} \in \mathbb{R}^{(m+M),(m+M)}$ is now determined by the size of the local stencil.

As a consequence of the local formulation, the resulting coefficient matrices $\boldsymbol{C_I} \in \mathbb{R}^{N_I,N_I}$ and $\boldsymbol{C_B} \in \mathbb{R}^{N_I,N_B}$ are now sparse. The improvement in computational efficiency is therefore given by two factors:

- linear systems (4.20) must still be solved at any point $\boldsymbol{x}_i \in \mathcal{X}_I$ but are now associated to matrices $\boldsymbol{M_{BC}}$ of a much smaller and fixed size

- matrix $\boldsymbol{C_I} \in \mathbb{R}^{N_I,N_I}$ is sparse with a number of nonzero entries in each row i equal to the number of internal nodes belonging to the stencil $\mathcal{X}_i$.

An important remark on practical implementations of the RBF-FD method concerns the nearest neighbors research algorithm. The brute force approach requires finding all pairwise distances between nodes and then sorting them, such algorithm with a cost of $\mathcal{O}(N^2)$ operations would almost defeat the advantage of a local approach, more efficient algorithms, like a *k-d tree* require $\mathcal{O}(N \log N)$ operations for rearranging the nodes and other $\mathcal{O}(N \log N)$ operations for finding some fixed number of nearest neighbors to all nodes [35]. In case a node-repel refinement algorithm is used for generating the nodes, the same nearest-neighbors research method can also be employed for the implementation of the RBF-FD method. Another important aspect in the solution of system (4.20), or system (4.13) in case of a global approach, is the conditioning of matrix $\boldsymbol{M_{BC}}$ enforcing boundary conditions. Matrix $\boldsymbol{M_{BC}}$ is closely related to the original interpolation matrix $\boldsymbol{M}$ in (2.13) and the two are identical in absence of boundary nodes or when all boundary nodes have Dirichlet conditions. In the case of infinitely smooth RBFs, when the node density is increased or the scaling factor ε in Φ is changed, the same considerations on the stability of the interpolation problem continue to hold. For instance, if equation (4.20) is solved explicitly, stability issues arise unless the value of ε is changed accordingly because we still fall in the case of *non stationary* interpolation. The remedy is to adopt *stationary* interpolation also for the RBF-FD method as explained in equation (2.41). In practice, since $h_{\mathcal{X},\Omega}$ is not available, this is achieved by defining ε to be a function of the value $s(\boldsymbol{x}_i)$ taken by a prescribed spacing function at the central node of stencil $\mathcal{X}_i$. More specifically, every time a stencil $\mathcal{X}_i$ is built around point $\boldsymbol{x}_i$, the following value $\varepsilon(\boldsymbol{x}_i)$ is used in all terms of blocks $\boldsymbol{\Phi_I}$ and $\boldsymbol{B\Phi_B}$ within matrix $\boldsymbol{M_{BC}}$:

$$\varepsilon(\boldsymbol{x}_i) = \frac{\varepsilon_0}{s(\boldsymbol{x}_i)} \tag{4.21}$$

where ε_0 is a fixed constant. As pointed out for the case of stationary interpolation, the approximation order of the resulting scheme is determined by the degree of the polynomial augmentation [53]: a polynomial base of degree P will allow an approximation order $\mathcal{O}(s^P)$ at best.

Finally, when polynomial augmentation in combination with stationary interpolation is adopted, the number of nodes included in the stencil must guarantee that the stencil is *q-unisolvent*, cfr. definition at page 23. Such a condition can be easily met by setting the number of nodes included in the stencil m to be at least twice the number of polynomial terms M [3]:

$$m \geq 2M = 2\binom{P+d}{P} \tag{4.22}$$

Further investigation on the influence of the stencil size has been recently discussed in [56], where changing the stencil size was found to induce oscillations in both the solution and discretization errors for the Poisson equation. This phenomenon is currently under research and can be considered a promising research topic.

4.3 RBF-FD errors

In this section some results from [73] are reported, showing the convergence curves with the discretization error of the Laplacian operator $\Delta = \nabla^2$ and the solution of the Poisson equation (4.23) in the case of two different 3D geometries.

$$\begin{cases} -\Delta u(\boldsymbol{x}) & = f(\boldsymbol{x}) \quad in\ \Omega \\ au(\boldsymbol{x}) + b\,\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) & = g(\boldsymbol{x}) \quad on\ \partial\Omega \end{cases} \tag{4.23}$$

The Multiquadric Radial Basis Function is used (cfr. Table 2.1):

$$\Phi(\boldsymbol{x}, \boldsymbol{x}_i) := \sqrt{1 + \varepsilon^2 \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2} \tag{4.24}$$

Solution Error

The accuracy in the solution of the Poisson equation (4.23) is assessed by comparing it with the exact solution:

$$u(\boldsymbol{x}) = \exp(x_1 + x_2 + x_3) \tag{4.25}$$

with $\boldsymbol{x} = \{x_1, x_2, x_3\} \in \mathbb{R}^3$.
Values $(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_{N_I}))$ and $(g(\boldsymbol{x}_{N_I+1}), \ldots, g(\boldsymbol{x}_N))$ are computed analytically from $u(\boldsymbol{x})$ and substituted in equation (4.15).

Discretization Error

The discretization error on the Laplacian operator, instead, is computed by comparing the value $\Delta u(\boldsymbol{x}_i)$ obtained analytically from (4.25) at $\boldsymbol{x}_i \in \mathcal{X}_I$ with $\Delta^h u(\boldsymbol{x}_i)$ defined as:

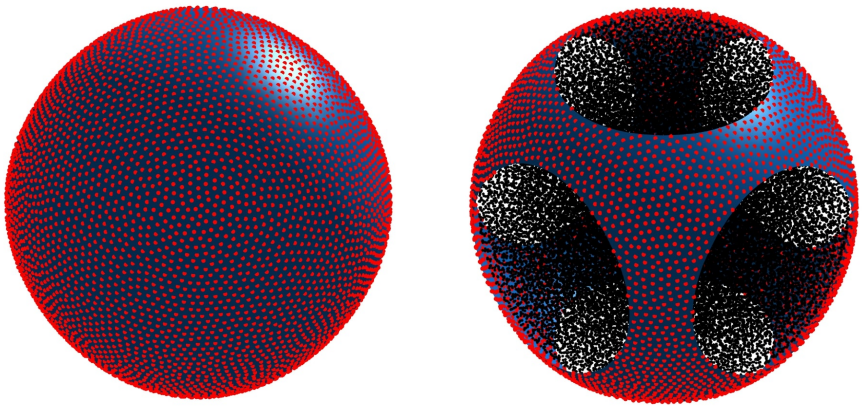
$$\Delta^h u(\boldsymbol{x}_i) = \boldsymbol{c_I}(\boldsymbol{x}_i)^T \boldsymbol{u_I} + \boldsymbol{c_B}(\boldsymbol{x}_i)^T \boldsymbol{g} \tag{4.26}$$

where $\boldsymbol{u_I}$ is the exact solution at the inner nodes, $\boldsymbol{g}$ is the boundary condition at boundary nodes and vectors $\boldsymbol{c_I}(\boldsymbol{x}_i)$ and $\boldsymbol{c_B}(\boldsymbol{x}_i)$ satisfy:

$$\boldsymbol{M_{BC}}^T \begin{pmatrix} \boldsymbol{c_I}(\boldsymbol{x}_i) \\ \boldsymbol{c_B}(\boldsymbol{x}_i) \\ \boldsymbol{c_p}(\boldsymbol{x}_i) \end{pmatrix} = \begin{pmatrix} \Delta\Phi(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \Delta \boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \tag{4.27}$$

4.3.1 3D Sphere

The domain of calculus Ω is delimited by a spherical boundary $\partial\Omega$ depicted in figure 4.2 with collocation nodes. This geometry is characterized by the absence of any edge which may induce some instability when Robin or Neumann boundary conditions are enforced.



(a) Boundary nodes (b) Inner nodes

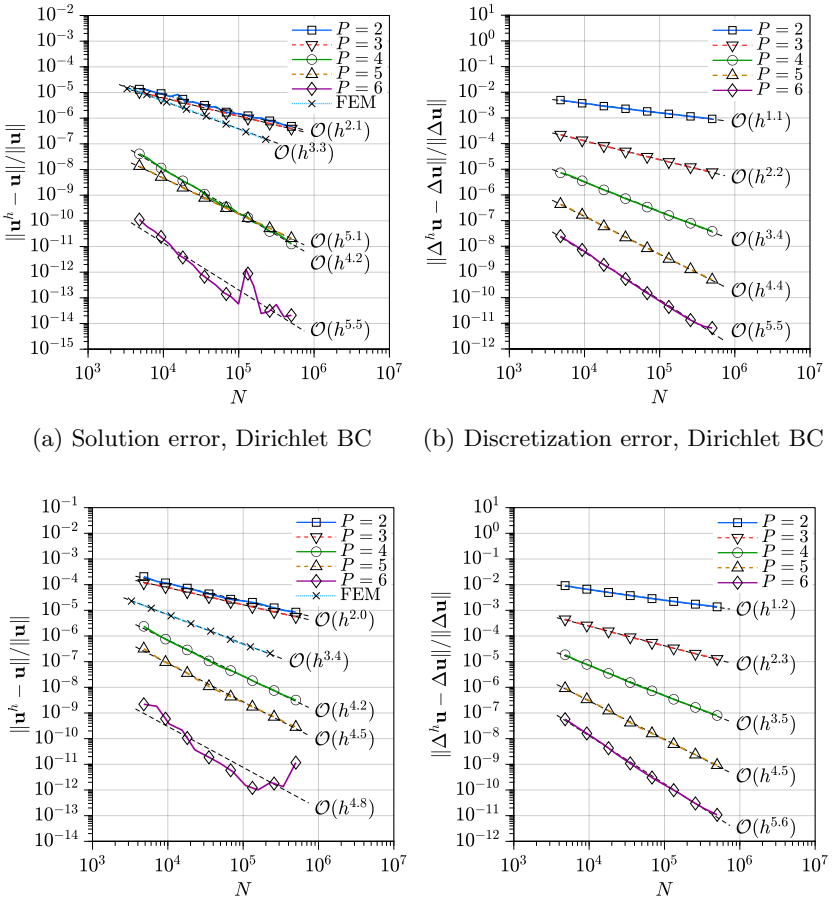
Figure 4.2: Example of node distribution with $N \approx 50$k nodes in the sphere: boundary nodes are represented in red, inner nodes in black.

In Figure 4.3 the convergence curves for the solution error (Figures 4.3a,4.3c) and discretization error (Figures 4.3b,4.3d) are shown with polynomial augmentation of degrees up to $P = 6$. The number of nodes generated ranges from $N = 5k$ to $N = 500k$, boundary conditions considered are Dirichlet ($a = 1, b = 0$ in Equation (4.23)) and Robin ($a = 1, b = 1$ in Equation (4.23)). The spacing function is set to be uniform $s(\boldsymbol{x}) = h$ with $h \propto N^{-1/3}$, as can be seen the discretization errors depicted in Figures 4.3b and 4.3d follow the expected pattern:

$$\frac{\|\Delta^h u - \Delta^u\|}{\|\Delta^u\|} \propto h^P \tag{4.28}$$

When considering solution errors, the expected pattern is satisfied by polynomial of even degrees when solving Poisson equation, which is a second order PDE. In the case of $P = 6$ and $N > 200k$ the normalized error becomes close to machine epsilon and further increase do not translate into better performance and the graph displays some oscillations. FEM convergence curves are also displayed in Figure 4.3a,c for comparison; it can be observed that the FEM curve is very similar to the ones for the cases $P = 2, 3$ for Dirichlet b.c., while for Robin b.c. the FEM curve lies between the cases $P = 3$ and $P = 4$, this indicates some loss of accuracy for the RBF-FD method.

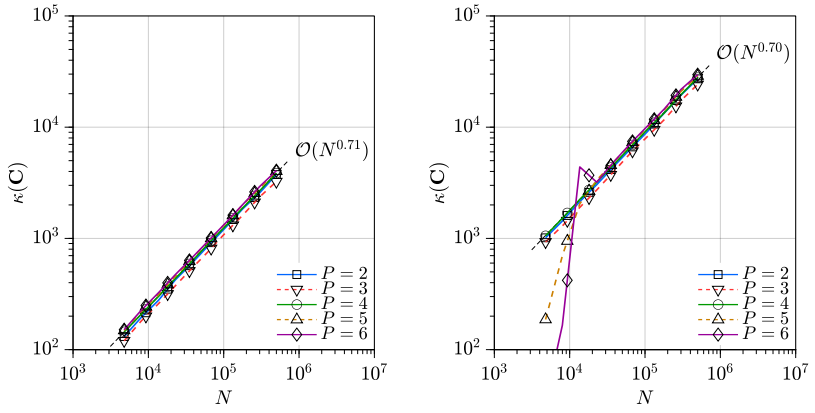
Finally, the estimation of the condition number $\kappa(\boldsymbol{C_I})$ of the final sparse matrix $\boldsymbol{C_I}$ is shown in Figure 4.4. $\kappa(\boldsymbol{C_I})$ being defined as $\kappa(\boldsymbol{C}) = \|\boldsymbol{C}\|_2\|\boldsymbol{C}^{-1}\|_2$. It is possible to see how in this case the condition nomber grows with order $\mathcal{O}(N^{0.7})$ in both cases regardless of P, thus very much comparable with the order $\mathcal{O}(N^{2/3})$ of classic finite difference schemes for the 3D Laplacian operator on uniform grids [48]. The condition number for Robin BC reported in Figure 4.4b is approximately 7 times larger than that for Dirichlet BC of Figure 4.4a,



(a) Solution error, Dirichlet BC (b) Discretization error, Dirichlet BC

(c) Solution error, Robin BC (d) Discretization error, Robin BC

Figure 4.3: Convergence curves for solution and Laplace operator (or Discretization Error) for Dirichlet and Robin b.c., Ω: Sphere.



(a) Dirichlet BC (b) Robin BC

Figure 4.4: Condition number of the sparse matrix $\boldsymbol{C_I}$.

which is again comparable to the value 5.78 of classic finite difference schemes with the same BC [73].

4.3.2 3D Engine Crankcase `.stl`

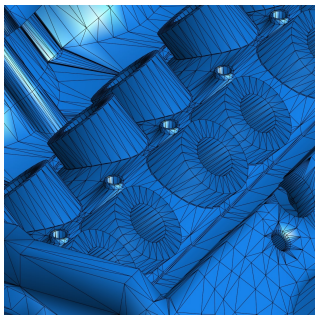
In this case the domain of calculus is delimited by a 3D model of the crankcase of a V8 ICE (Internal Combustion Engine) defined as a set of triangles in the `.stl` format. This domain is depicted in Figure 4.5. A variety of features is present on the `.stl` surface and these can trigger some ill-conditioning in presence of boundary conditions involving normal derivatives, leading to a degradation of accuracy.

In Figure 4.6 convergence curves for the solution and discretization errors are displayed. Polynomial degrees are $P = 2, 3, 4$, and total number of nodes ranges from $N = 5$k to $N = 750$k nodes. In the case of Dirichlet BC the results in Figure 4.6a and 4.6b are somewhat similar to those attained on the sphere, some oscillations arise in the case of polynomial of degree $P = 4$. In the case of Robin BC, instead, ill-conditioning issues affecting matrix $\boldsymbol{M_{BC}}$ disrupt the stability of the overall solution process. In order to obtain the data displayed in Figures 4.6c and 4.6d, some modifications to the usual nearest neighbor research were applied. Every time a boundary node $\boldsymbol{x}_j$ had to be included in a given stencil centered at $\boldsymbol{x}_i$, the following requirements had to be satisfied [73]:

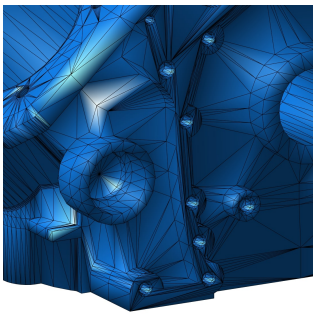
- Angle requirement:

$$\frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2} \cdot \boldsymbol{n_j} > 0.6 \tag{4.29}$$

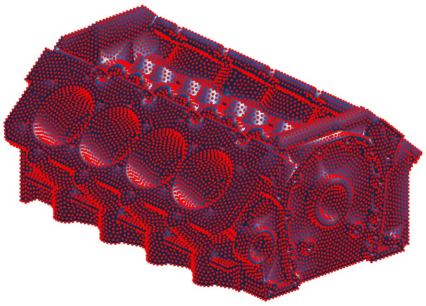
  where $\boldsymbol{n_j}$ is the normal direction at point $\boldsymbol{x}_j$. Equation (4.29) states that a boundary node x_b can be included in the stencil only if the angle γ



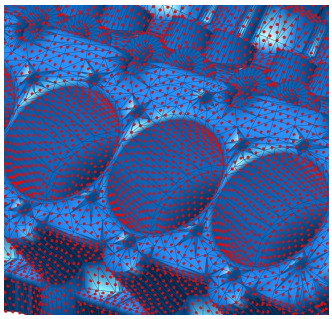
(a) Enlarged view of a particular



(b) Enlarged view of a particular



(c) Boundary nodes



(d) Enlarged view of a particular

Figure 4.5: Enlarged views of some details of the `.stl` crankcase model and an example of node distribution with $N \approx 75$k nodes.

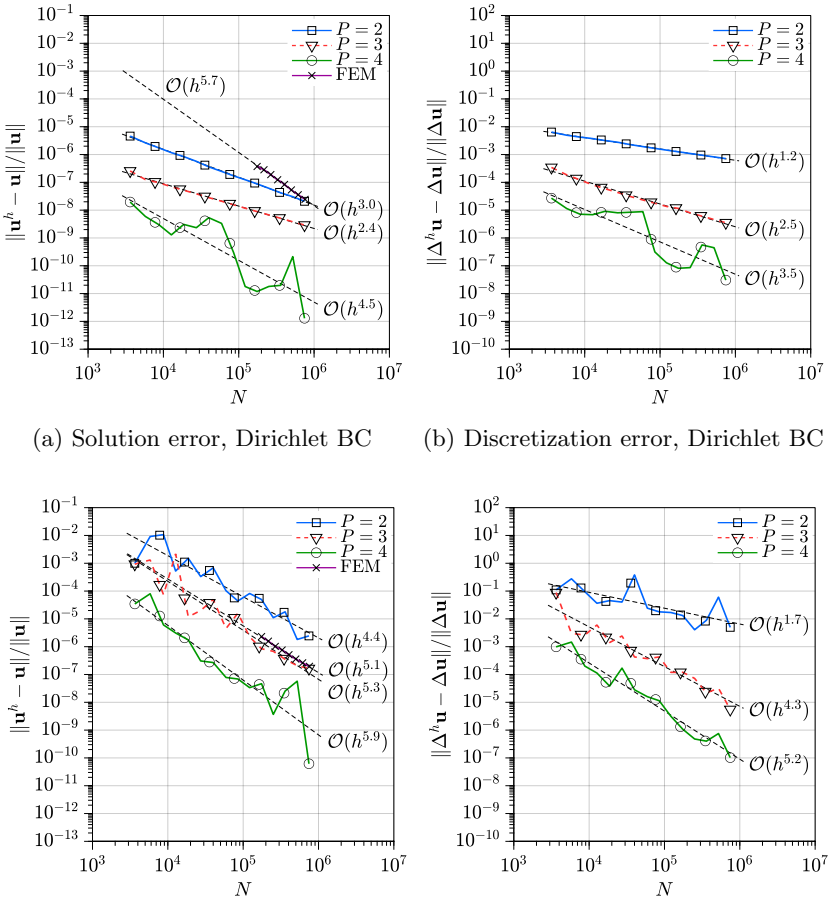
between the outer normal $\boldsymbol{n}_{\boldsymbol{x}_j}$ associated to the boundary node $\boldsymbol{x}_j$ and the line connecting $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$ satisfies $\gamma < \arccos(0.6) \approx 53°$.

- Distance requirement: $\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2 < 2s$, i.e., only boundary nodes closer to the center $\boldsymbol{x}_i$ than twice the spacing function can be included in the stencil.

- Number requirement: no more than 10 boundary nodes can be included in the stencil. If more then 10 are found, only the 10 closest are included.

Such requirements were motivated by the assumption that instability issues related to Robin or Neumann boundary conditions are only triggered by the normal directions associated to boundary nodes. The insurgence of such instabilities is a completely different phenomenon than the Runge phenomenon, the instability due to flat RBFs or singularity of the polynomial augmentation related to unisolvency issues. Even when the solution process was stabilized and ill-conditioning issues affecting matrix $\boldsymbol{M_{BC}}$ were solved achieving high orders of convergence, evident oscillations appeared in the convergence curves, a symptom that the underlying issue was solved only partially.

The awareness of the problem of ill-conditioning due to the presence of Neumann BC motivated the deeper investigation which is presented in chapter 5.



(a) Solution error, Dirichlet BC (b) Discretization error, Dirichlet BC

(c) Solution error, Robin BC (d) Discretization error, Robin BC

Figure 4.6: Convergence curves for solution and Laplace operator (or discretization error) for Dirichlet and Robin BC, Ω: Engine.

Chapter 5

Neumann Stability

5.1 Introduction

In the chapter 4 it was conjectured that the RBF-FD method, based on an unsymmetric formulation, is affected by ill-conditioning of the local interpolation matrix in presence of Neumann or Robin boundary conditions. Ill-conditioning issues of this kind translate into instability and sometimes the whole solution process is disrupted. Because Dirichlet BC do not induce any such problem, this chapter will focus on the insurgence of such instabilities linked to Neumann BC only and some improvements for the traditional unsymmetric interpolation scheme will also be proposed.

In order not to defeat the major advantages offered by meshless methods, the proposed solutions must work with any possible shape of the boundary $\partial\Omega$. Furthermore, any modification of the node generation algorithm should not compromise the fulfillment of requirements listed in subsection 3.2.2 at page 41, for example by forcing the user to input specific parameters based on the specific shape of $\partial\Omega$. As a consequence, it is not possible to make any a priory assumption on the direction of the boundary normals.

Neumann BC can be enforced both at the stencil level, i.e. when the local interpolants exactly satisfy the BC at the boundary nodes [70, 72, 71, 73, 119], or at the assembly level, i.e. when the equations for the BCs at the boundary nodes appear explicitly in the final sparse matrix [92, 53]. This kind of boundary condition is essential in most boundary value problems of engineering interest and related instability issues constitute a major obstacle for any practical application of the unsymmetric RBF-FD method. Some effective solutions for these problems have been suggested, however no one has been universally adopted yet.

While the stability problem due to the presence of Neumann BCs is unrelated to that due to the flat limit with non stationary interpolation (cfr. section 2.9 page 33), both are caused by ill-conditioning of the interpolation matrix and therefore may be solved at once. This might be the case when the stabilization is achieved by means of least squares procedures like the ones presented in

[64, 103, 102], where the interpolation matrix is not solved directly. The one in [102] seems especially capable of providing both stable and accurate results by using two different node sets.

An approach to the stabilization which is instead specific for the management of boundary conditions is the one based on the so called ghost nodes. In [32] one layer of ghost nodes is placed outside the domain at a distance roughly continuing the pattern of the interior nodes in order to address the issues related to one-sided boundary stencils. In [63] the authors propose instead the use of fictitious nodes that extend far outside the domain, resembling the approach from the method of fundamental solutions [26]. In another implementation [30] the authors add a set of nodes, which can lie inside or outside of the domain, adjacent to the boundary and, correspondingly, add an additional set of equations obtained via collocation of the PDE on the boundary. The strategy of ghost nodes is successfully employed also in [24], for the stable RBF-FD solution of elliptic PDEs problems on complex 3D geometries and even in higher dimension in [53].

Alternatively to the ghost nodes, in [89] additional nodes are placed close to the boundary, but within the domain, in order to reduce the error in those areas in the context of the overlapped RBF-FD method. The overlapped variant differs from the traditional RBF-FD in the selection of the interpolation stencils and can be intended as a generalization of the latter.

Yet another approach specific for boundary stabilization is adopted in [13, 70], where every time a Neumann boundary node has to be included in the stencil, its normal is checked according to different geometric criteria, similar to that adopted in [73] and explained in the previous chapter.

The main result contained in this chapter is the observation that, given a certain stencil, it is always possible to find normal directions that make the local discretization matrix $\boldsymbol{M_{BC}}$ singular. Such singular directions can simply be calculated as the ones that send the determinant of the said matrix to zero.

The most straightforward solution and the first to be presented, is the implementation of a control which discards from a given stencil those nodes associated with dangerous normal directions. This approach, somewhat similar to the one adopted in [70], is more expensive but totally robust and it is tested with good results. At the best of the authors' knowledge it represents the only possible solution that carries on the traditional RBF-FD method totally unchanged and does not require any intervention on the node generation.

Alternatively, it is also possible to properly move the nodes on the boundary in order to rule out the possibility of singular interpolation matrices. This operation appears to be the most computationally efficient and the simplest to implement on smooth and regular domains but requires non-trivial geometrical operations on generic domains. A robust implementation of this approach, capable of working on arbitrarily complex domains, can theoretically be developed at the price of introducing more free parameters within the node-repel algorithm.

We remark that the proposed strategies for the prevention of ill-conditioning problems due to Neumann BCs were developed specifically for the unsymmetric RBF-FD. Other RBF-based schemes which are inherently stable regardless of

normal directions, like the symmetric RBF-HFD method discussed in chapter 7, do not require the adoption of any of those stabilization strategies.

5.2 RBF-FD with Neumann BC

This section mainly serves as a clarification for the notation that will be adopted in the following pages, where the interest will be solely focused on stencils with Neumann boundary nodes. When there is no risk of confusion, the normal derivative will be indicated as:

$$\partial F(\boldsymbol{x}_i) := \frac{\partial F}{\partial \boldsymbol{n}}(\boldsymbol{x}_i) \tag{5.1}$$

where $\boldsymbol{n}$ is the unit normal at $\boldsymbol{x}_i$.

Suppose therefore that we are interested in solving the following boundary value problem:

$$\begin{cases} \mathcal{L}u &= f \quad \text{in } \Omega \\ \partial u &= g \quad \text{on } \partial\Omega \end{cases} \tag{5.2}$$

As done above, we assume that the nodes are listed in the following order within the interpolation stencil: the first m_I are internal nodes, the following m_B are boundary nodes and $m_I + m_B = m$ holds. Index $m_I + 1$, i.e. that associated to the first boundary node, will be written as $m_I + 1 = m_J$, the subset of internal nodes within a stencil will be written as: $\mathcal{X}_I = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m_I}\}$, whereas that of boundary nodes will be: $\mathcal{X}_I = \{\boldsymbol{x}_{m_J}, \dots, \boldsymbol{x}_m\}$. In the unsymmetric formulation, boundary conditions are enforced at a local level by requiring that $u^h(\boldsymbol{x})$ satisfies conditions in equation (4.8), which in the present case becomes:

$$\begin{aligned} u^h(\boldsymbol{x}_i) &= u(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \\ \partial u^h(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_B \end{aligned} \tag{5.3}$$

With the following matrix form where blocks $\boldsymbol{\Phi_I}$ and $\boldsymbol{\mathcal{B}\Phi_B}$ of equation (4.9) have been united as well as $\boldsymbol{P_I}$ and $\boldsymbol{\mathcal{B}P_B}$:

$$\underbrace{\begin{pmatrix} \boldsymbol{\Phi_N} & \boldsymbol{P_N} \\ \boldsymbol{P}^T & \mathbf{0} \end{pmatrix}}_{\boldsymbol{M}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_I} \\ \boldsymbol{g} \\ \mathbf{0} \end{pmatrix} \tag{5.4}$$

Matrix $\mathbf{M}$ in equation (5.4) corresponds to matrix $\boldsymbol{M_{BC}}$ of the previous chapter. The subscript $\boldsymbol{_{BC}}$ was dropped throughout this chapter since there is no confusion: Neumann BC, being the main topic, are always present. Furthermore, $\boldsymbol{M}$ will sometimes be called *interpolation matrix* since it is obtained by enforcing conditions (5.3) within the standard framework of scattered data interpolation

theory:

$$\boldsymbol{\Phi_N} = \begin{pmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \\ \partial\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{pmatrix} \tag{5.5}$$

$$\boldsymbol{P_N} = \begin{pmatrix} p_1(\boldsymbol{x}_1) & \dots & p_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_{m_I} & \dots & p_M(\boldsymbol{x}_{m_I}) \\ \partial p_1(\boldsymbol{x}_{m_J}) & \dots & \partial p_M(\boldsymbol{x}_{m_J}) \\ \vdots & \ddots & \vdots \\ \partial p_1(\boldsymbol{x}_m & \dots & \partial p_M(\boldsymbol{x}_m) \end{pmatrix} \tag{5.6}$$

$$\boldsymbol{P}^T = \begin{pmatrix} p_1(\boldsymbol{x}_1) & \dots & p_1(\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ p_M(\boldsymbol{x}_1) & \dots & p_M(\boldsymbol{x}_m) \end{pmatrix} \tag{5.7}$$

The normal derivatives of RBFs in equation (5.5) are calculated by implying that $\Phi(\cdot, \boldsymbol{x})$ is in a function with respect to the first argument:

$$\partial\Phi(\boldsymbol{x}_k, \boldsymbol{x}_i) = \frac{\partial\Phi}{\partial\boldsymbol{n}}(\boldsymbol{x}_k, \boldsymbol{x}_k) = \varphi'(r_{i,k})\, \boldsymbol{e}_{i,k} \cdot \boldsymbol{n} \tag{5.8}$$

where $\varphi'(r_{i,k})$ is the derivative of the associated basic function, $r_{i,k} = \|\boldsymbol{x}_k - \boldsymbol{x}_i\|_2$, $\boldsymbol{e}_{i,k} = (\boldsymbol{x}_k - \boldsymbol{x}_i)/r_{i,k}$ is the unit direction from node $\boldsymbol{x}_i$ towards node $\boldsymbol{x}_k$ and $\boldsymbol{n}$ is always the normal at point $\boldsymbol{x}_k$ taken as first argument.

We also require the basic functions to satisfy the following condition, which is necessary for the associated RBF $\Phi(\cdot, \boldsymbol{x}_i)$ to be differentiable also at the nodes:

$$\varphi'(0) = 0 \tag{5.9}$$

Equation (5.9) implies that the diagonal entries $\partial\Phi(\boldsymbol{x}_i, \boldsymbol{x}_i)$ of the last m_B rows of $\boldsymbol{\Phi_N}$ are all 0.

As for the rest, the solution procedure remains unchanged, as explained in

section 4.2, leading to the final sparse system of equation (4.15):

$$\boldsymbol{C_I}\begin{pmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{pmatrix} = \begin{pmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_{N_I}) \end{pmatrix} - \boldsymbol{C_B}\begin{pmatrix} g(\boldsymbol{x}_{N_{I+1}}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{pmatrix} \tag{5.10}$$

which can also be written more compactly as:

$$\boldsymbol{C_I}\,\boldsymbol{u^h} = \boldsymbol{f} - \boldsymbol{C_B}\,\boldsymbol{g} \tag{5.11}$$

Equation (5.11), which is the discrete version of the boundary value problem (5.2) can now be solved for $\boldsymbol{u}^h \in \mathbb{R}^{N_I}$. Furthermore, because of the lack of connectivity information, local interpolation systems (4.27) can be solved independently from one another.

We will show that the onset of instabilities at the boundary is due to ill-conditioning of the interpolation matrix $\boldsymbol{M}$, and that such ill-conditioning can also be induced by the direction of the boundary normals appearing in the corresponding rows in blocks (5.5) and (5.6). It is important to remark that high condition numbers for matrix $\boldsymbol{M}$ do not necessarily imply any loss of accuracy in the method. Indeed, in presence of a regular domain, like the spherical one adopted in section 4.3.1, ill-conditioning of matrix $\boldsymbol{M}$, if present, was not serious enough to impact the final solution calculated in double precision arithmetic. The present study was indeed motivated by those cases where, in proximity of some strangely shaped domain, matrix $\boldsymbol{M}$ becomes so badly conditioned that final accuracy is compromised.

5.2.1 Cardinal functions with Neumann BC

As pointed out in section 2.5 at page 24, an expression with cardinal functions of u^h is obtained by considering the identity operator $\mathcal{L} = \mathcal{I}$ in the left side of equation (4.13):

$$\mathcal{I}u^h(\boldsymbol{x}_i) = u^h(\boldsymbol{x}) = \sum_{j=1}^{m_I} u(\boldsymbol{x}_j)\psi_j(\boldsymbol{x}_i) + \sum_{k=m_J}^{m} \partial u(\boldsymbol{x}_k)\psi_k(\boldsymbol{x}_i) \tag{5.12}$$

where the cardinal functions $\{\psi_1(\boldsymbol{x}_i), \ldots, \psi_m(\boldsymbol{x}_i)\}$ are given by the first m elements $\{c_1(\boldsymbol{x}_i), \ldots, c_m(\boldsymbol{x}_i)\}$ in equation (4.27) in the case $\mathcal{L} = \mathcal{I}$, i.e., when the right side of the same equation is the column vector of the bare basis functions:

$$\mathbf{M}^T\begin{pmatrix} \boldsymbol{\psi}(\boldsymbol{x}_i) \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \Phi(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \tag{5.13}$$

with $\boldsymbol{\psi}(\boldsymbol{x}_i) = \{\psi_1(\boldsymbol{x}_i), \ldots, \psi_m(\boldsymbol{x}_i)\}$, $\Phi(\boldsymbol{x}_i, \mathcal{X}_{i,I}) = \{\Phi(\boldsymbol{x}_i - \boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_i - \boldsymbol{x}_m)\}$ and $\boldsymbol{p}(\boldsymbol{x}_i) = \{p_1(\boldsymbol{x}_i), \ldots, p_M(\boldsymbol{x}_i)\}$. These cardinal functions extend the definition

given in [23, 40] to the case with Neumann BCs. The cardinal functions associated to the internal nodes, i.e., $\{\psi_1(\boldsymbol{x}_i), \ldots, \psi_{m_I}(\boldsymbol{x}_i)\}$, satisfy:

$$\begin{cases} \psi_j(\boldsymbol{x}_i) = \delta_{ij} & \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \\ \partial\psi_j(\boldsymbol{x}_i) = 0 & \text{otherwise} \end{cases} \tag{5.14}$$

while the ones associated to the boundary nodes, i.e., $\{\psi_{m_J}(\boldsymbol{x}_i), \ldots, \psi_m(\boldsymbol{x}_i)\}$, satisfy:

$$\begin{cases} \psi_k(\boldsymbol{x}_i) = 0 & \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \\ \partial\psi_k(\boldsymbol{x}_i) = \delta_{kj} & \text{otherwise} \end{cases} \tag{5.15}$$

where δ_{ij} is the Kronecker delta.

In order to separate the contributions of internal and boundary nodes, we define two different Lebesgue functions:

$$\begin{aligned} \lambda_I(\boldsymbol{x}_i) &= \sum_{j=1}^{m_I} |\psi_j(\boldsymbol{x}_i)| \\ \lambda_B(\boldsymbol{x}_i) &= \sum_{k=m_J}^{m} |\psi_k(\boldsymbol{x}_i)| \end{aligned} \tag{5.16}$$

where λ_I is associated to the internal nodes and λ_B to the boundary nodes.

When the interpolation stencil is restricted to lie in a compact set $K \subset \mathbb{R}^d$, then the Lebesgue constant is defined as the maximum of the Lebesgue function on K [23, 40]. In our case we will consider K to be the minimal convex set containing all of the stencil nodes (including those lying on the boundary). In order to keep the distinction between boundary nodes and inner nodes, we define separately the following two Lebesgue constants, Λ_I for the interior and Λ_B for the boundary:

$$\begin{aligned} \Lambda_I &= \max_{\boldsymbol{x} \in K} \lambda_I(\boldsymbol{x}) \\ \Lambda_B &= \max_{\boldsymbol{x} \in K} \lambda_B(\boldsymbol{x}) \end{aligned} \tag{5.17}$$

5.3 Neumann Ill-Conditioning

5.3.1 Preliminary considerations

In order to study the causal relationship between Neumann BCs and ill-conditioning issues affecting matrix $\boldsymbol{M}$, it is necessary to reproduce them in an environment isolated from other factors influencing the solvability of system (4.27). For this purpose the reference stencil of Figure 5.1 was introduced, it represents the typical situation encountered when system (4.27) is solved for a stencil including several boundary nodes. In the reference stencil of Figure 5.1 nodes are arranged in a perfectly hexagonal layout and normal vectors are aligned with the segments connecting each boundary node with point G. In this way the direction of the

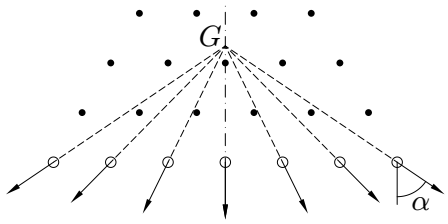


Figure 5.1: reference stencil with $m_I = 15$ internal nodes (filled dots) and $m_B = 7$ boundary nodes (empty circles).

normals can be controlled by moving the point G along the axis of symmetry and an angle $\alpha \in [-\pi/2\,,\,\pi/2]$ remains defined by the normal located further to the right. It also follows that $\alpha > 0$ if G is above the line of the boundary nodes and $\alpha < 0$ otherwise. The whole analysis is performed for the 2D case but the results continue to hold also in the 3D case.

Figure 5.2 depicts the condition number $\kappa(\boldsymbol{M})$ and the constant Λ_I, defined in equation (5.17), against the angle α in the case of the reference stencil of Figure 5.1. Whenever one of these quantities grows unbounded, we have a very ill-conditioned system in equation (4.27), which translates into large errors and most likely stability issues. All plots of Figures 5.2 and 5.3 are attained using MQ RBF (cfr. 2.1) with shape parameter ε satisfying $\varepsilon_0 = 0.5$ in equation (4.21).

Negative values of α are the most dangerous, they are associated with locations of the point G placed below the boundary in Figure 5.1 and for small values approximate the normal directions in case of a concave boundary profile. From Figure 5.2 we see that multiple singularities appear for certain negative values of α, near which both $\kappa(\boldsymbol{M})$ and Λ_I assume very large values. We also remark that the very same values of α are detected as problematic in both charts. Furthermore such singular configurations exist for modest values of α regardless of the degree of the polynomial augmentation and this remains true even with larger stencils. These remarks seem to tell us that no matter the node density or the polynomial degree, even a slight local concavity of the boundary might cause severe ill-conditioning issues for the considered RBF interpolations. It is true that the actual position of boundary nodes in the reference stencil of Figure 5.1 does not change, but it is also true that it is not possible to forecast any possible node arrangement attainable in reality. The purpose of this construction is therefore not so much to mimic the behavior of matrix $\boldsymbol{M}$ in any realistic situation, but rather to reproduce the insurgency of ill-conditioning issues due to Neumann BC in a controlled environment.

Values taken by $\lambda_I(\boldsymbol{x})$ and $\lambda_B(\boldsymbol{x})$ for two particular values of the angle α are displayed in Figure 5.3. These plots are again attained using the same MQ RBF scheme with no polynomial augmentation, polynomial augmentation would have increased even more the values of λ_I and λ_B near the edges of the stencil. Once again it emerges that, for certain negative values of α, large interpolation

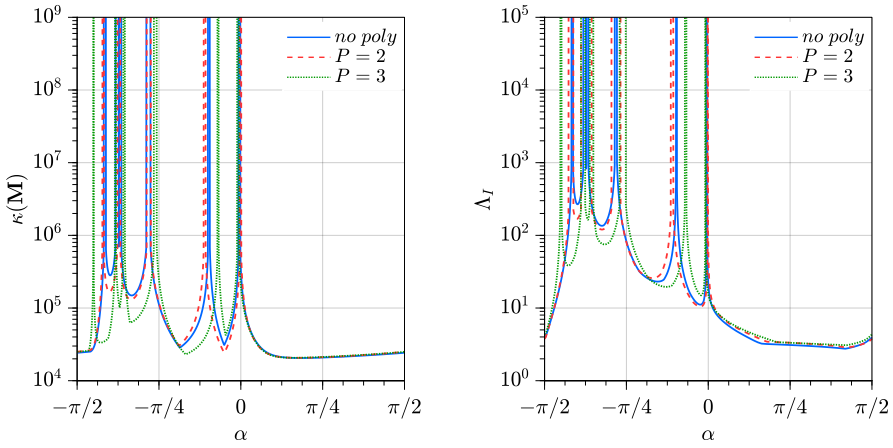


Figure 5.2: condition number of the interpolation matrix (left), and Lebesgue constant Λ_I (right), for the RBF interpolation on the reference stencil of Figure 5.1.

errors can occur. This is yet another hint at the fact that Neumann BC can compromise the accuracy and the well-posedness of the system (4.27) and thus introduce large errors in the final solution of system (5.11). In the following sections this phenomenon is analysed quantitatively in the case of one or multiple boundary neighbors.

5.3.2 Bare RBF with one boundary node

Singular direction

Consider for simplicity a stencil with $m_B = 1$ boundary node $\boldsymbol{x}_m$ and an interpolant with no polynomial augmentation. The $m \times m$ interpolation matrix is therefore:

$$\boldsymbol{M} = \boldsymbol{\Phi_N} = \begin{pmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_1) & \cdots & \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \\ \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_1) & \cdots & \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{pmatrix} \tag{5.18}$$

The determinant of $\boldsymbol{M}$ can be expressed through a cofactor expansion along the last row:

$$\det(\boldsymbol{M}) = \sum_{j=1}^{m_I} C_{m,j} \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_j) \tag{5.19}$$

In equation (5.19), $C_{m,j}$ is the cofactor of entry $(m, j) = \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_j)$ in $\boldsymbol{M}$:

$$C_{m,j} = (-1)^{m+j} D_{m,j} \tag{5.20}$$

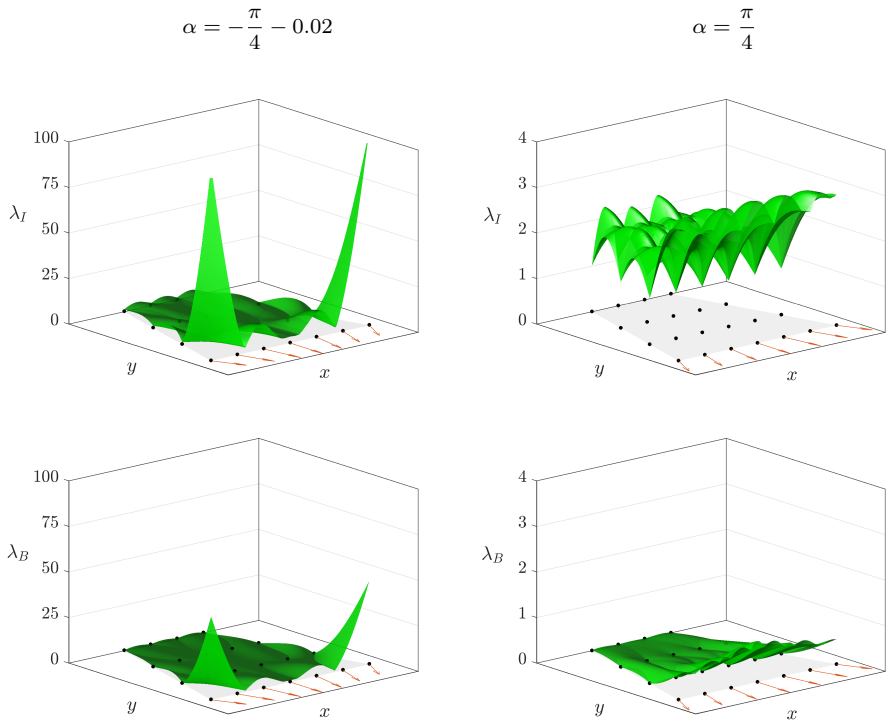


Figure 5.3: 3D visualization of the Lebesgue functions $\lambda_I(\boldsymbol{x})$ and $\lambda_B(\boldsymbol{x})$ for the MQ RBF interpolation ($\varepsilon_0 = 0.5$ and no polynomial augmentation) on the stencil of Figure 5.1. Note the large difference in the z-axis scaling for the two values of α.

where $D_{m,j}$ is the minor of the corresponding entry, i.e., the determinant of the submatrix obtained by deleting row m and column j from $\boldsymbol{M}$. The expansion in equation (5.19) has $m_I = m - 1$ terms because the diagonal entry of the last row is $\partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) = 0$, from equation (5.9).

Recalling the form of the normal derivative and the notation of equation (5.8), the determinant can be expressed as:

$$\det(\boldsymbol{M}) = \sum_{j=1}^{m_I} C_{m,j}\varphi'(r_{j,m})\, \boldsymbol{e}_{j,m} \cdot \boldsymbol{n} = \hat{\boldsymbol{n}} \cdot \boldsymbol{n} \tag{5.21}$$

where $\boldsymbol{n}$ is the unit normal at the boundary node $\boldsymbol{x}_m$ and $\hat{\boldsymbol{n}}$ is a linear combination of $\boldsymbol{e}_{j,m}$ terms, which are the unit directions pointing towards the boundary node $\boldsymbol{x}_m$:

$$\hat{\boldsymbol{n}} = \sum_{j=1}^{m_I} w_{j,m}\boldsymbol{e}_{j,m} \tag{5.22}$$

where coefficients $w_{j,m}$ are:

$$w_{j,m} = C_{m,j}\varphi'(r_{j,m}) \tag{5.23}$$

When the location of the boundary node $\boldsymbol{x}_m$ is fixed and the unit normal $\boldsymbol{n}$ can vary, the direction of $\hat{\boldsymbol{n}}$ can be interpreted as the optimal direction for $\boldsymbol{n}$ since it maximizes $\det(\boldsymbol{M})$. On the other hand, equation (5.21) states that if $\boldsymbol{n}$ is orthogonal to $\hat{\boldsymbol{n}}$, then $\boldsymbol{M}$ becomes singular.

The existence of such singular directions can also be directly deduced from the fact that $\det(\boldsymbol{M}(\boldsymbol{n}))$ in (5.18) is a continuous function of $\boldsymbol{n}$ and $\det(\boldsymbol{M}(-\boldsymbol{n})) = -\det(\boldsymbol{M}(\boldsymbol{n}))$ for any $\boldsymbol{n}$, since the reversal of $\boldsymbol{n}$ results in the change of sign of the last row of $\boldsymbol{M}(\boldsymbol{n})$. Since $\boldsymbol{n}$ can be changed continuously to reach $-\boldsymbol{n}$, there exist at least one normal direction for which the determinant vanishes. This conclusion is similar to the proof of the Mairhuber-Curtis theorem [27] stated at page 17 and this is to be expected since the unsymmetric RBF basis $\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_m)\}$ is independent from the direction of the normal vector $\boldsymbol{n}$.

In turn this imply that it is not possible to rule out the possibility of singular $\boldsymbol{M}$ on a generic domain and for this reason some countermeasures must be taken in order to make the standard RBF-FD formulation applicable to engineering problems.

Possible remedies

In this chapter two approaches will be proposed as stabilization techniques against ill-conditioning induced by Neumann BC, they are:

Approach 1. Discard a boundary node if the corresponding dot product $|\hat{\boldsymbol{n}} \cdot \boldsymbol{n}|/\|\hat{\boldsymbol{n}}\|_2$ is too small.

Approach 2. Change the boundary node position $\boldsymbol{x}_m$ according to the corresponding normal $\boldsymbol{n}$ in such a way that $\hat{\boldsymbol{n}}$ and $\boldsymbol{n}$ become more aligned.

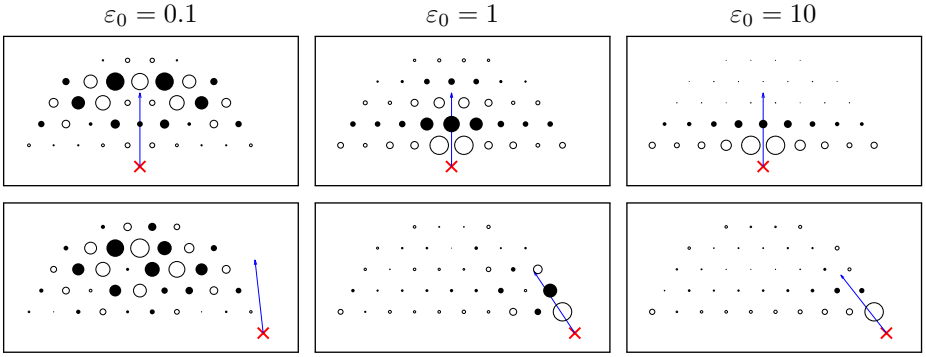


Figure 5.4: coefficients $w_{j,m}$ in equation (5.21) for the optimal direction, shown as a blue arrow, associated to the boundary node, shown as a red cross. Positive and negative coefficients are respectively represented as filled dots and empty circles. The size of the circles is proportional to the magnitude of the coefficients. Top row: symmetric position of the boundary node; bottom row: asymmetric position of the boundary node.

Both approaches are inspired by the parallel with the Mairhuber-Curtis theorem and involve some modifications to the discretization problem aimed at embedding some awareness of the actual direction of the normals without modifying the RBF basis.

Heuristic approaches for estimating $\hat{\boldsymbol{n}}$

At first it is clear that a calculation of $\hat{\boldsymbol{n}}$ or at least a suitable approximation is required. Given the definition of $\hat{\boldsymbol{n}}$ in equation (5.22), the easiest approximation is to consider the direction pointing from the boundary node towards some reference point, e.g., the stencil centroid. If the coefficients $w_{j,m}$ in equation (5.22) are very similar to each other this approximation becomes a good one. Unfortunately, this condition never occurs, as shown in Figure 5.4. In this figure the magnitude and the sign of the coefficients $w_{j,m}$ are shown in the case of the MQ RBF for a boundary stencil, i.e., one-sided stencil close to a straight boundary, with hexagonal node arrangement and for two different positions of the boundary node, shown as a red cross. Three different values for the parameter ε_0 are considered, the case $\varepsilon_0 = 0.1$ has been computed with MATLAB variable precision arithmetic in order to compensate for to the flat-limit effect.

Figure 5.4 shows that moving away from the boundary node, the coefficients $w_{j,m}$ have alternating signs and a decay rate growing with ε_0, which is in perfect accordance with theoretical observations for RBF interpolation on equispaced infinite lattices [42]. More specifically, by qualitative arguments based on the adaptation to a finite stencil, the cardinal expansion coefficients λ_k defined in [42] correspond to the cofactors $C_{m,j}$ in equation (5.19) up to a multiplicative constant, where the subscript k is the nondimensional distance from the boundary

node.

The qualitative behaviour of the magnitude of the coefficients $w_{j,m}$ is therefore given by the following equation, where the second proportionality can be found in [42] for the exponential regime:

$$|w_{j,m}| \propto |\lambda_k \varphi'(ks)| \propto e^{-\mu k} |\varphi'(ks)| \tag{5.24}$$

s being the spacing between nodes.

In the case of MQ RBF, equation (5.24) can be expressed in the form:

$$|w_{j,m}| \propto \frac{k e^{-\mu k}}{\sqrt{1 + (\varepsilon_0 k)^2}} = \tilde{w}(k) \tag{5.25}$$

When $\varepsilon_0 = 0.1$, then $\mu = 0.15$ [42] and $\tilde{w}(k)$ has a maximum for $k \approx 5$, which in Figure 5.4 corresponds to the large magnitude coefficients $w_{j,m}$ far from the boundary node. When $\varepsilon_0 = 1$, then $\mu = 1.04$ [42] and $\tilde{w}(k)$ has a maximum for $k \approx 1$, which in Figure 5.4 corresponds to the large magnitude coefficients $w_{j,m}$ immediately close to the boundary node, followed by a rapid decay. When $\varepsilon_0 = 10$ the coefficients with largest magnitude are associated to the nodes immediately close to the boundary node, followed by a faster decay.

Figure 5.4 also shows the optimal direction, i.e., the direction of $\hat{\boldsymbol{n}}$ defined in equation (5.22), as a blue vector. In the case of the symmetric position of the boundary node (top row in Figure 5.4), the optimal direction is vertical, regardless of ε_0, as expected for symmetry. In the case of the lateral placement of the boundary node (bottom row in Figure 5.4), the optimal direction changes from almost vertical for $\varepsilon_0 = 0.1$ to a certain inclination for $\varepsilon_0 = 10$, in accordance to the previous observations. Indeed, the larger the shape parameter ε, the larger the coefficients $w_{j,m}$ associated to the closest nodes and therefore the more the optimal direction points towards these closest node.

Figure 5.5 shows the direction and the magnitude of the optimal vectors $\hat{\boldsymbol{n}}$, depicted with solid black lines at different location of the boundary, indicated with a blue dashed curve. On the left the case of a stencil with $m_I = 3$ internal nodes and on the right one with $m_I = 15$ hexagonal internal nodes. The blue dashed curve indicating the boundary is formed by those points that have constant distance s from the nearest internal node. MQ RBF with $\varepsilon_0 = 0.5$ has been employed. The envelopes of the optimal directions of $\hat{\boldsymbol{n}}$ are shown as solid red curves, these were displayed in order to show that the strategy of comparing the actual boundary normal with some direction connecting the boundary node with some inner point is not viable. Indeed, if optimal directions were found to converge in some *accumulation point* contained within the stencil, it would be possible to avoid further calculations and approximate them as the line joining the boundary node with that accumulation point.

We conclude that an estimate for the optimal direction $\hat{\boldsymbol{n}}$ can not be obtained through simple geometric heuristics and a more robust calculation is therefore mandatory in the general case. Such calculation follows below.

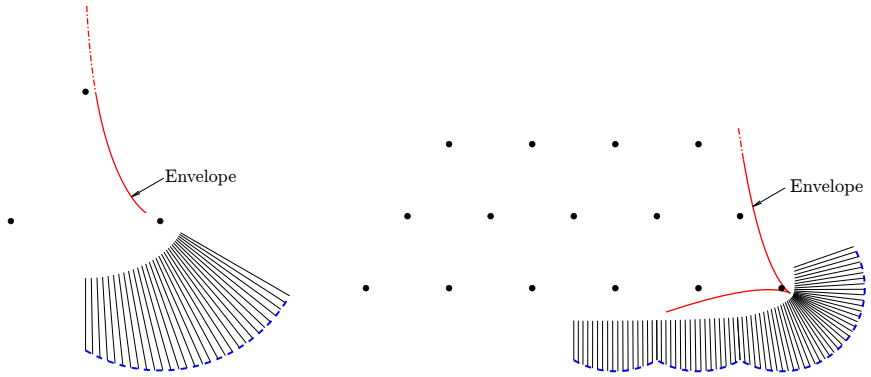


Figure 5.5: optimal vectors $\hat{\boldsymbol{n}}$ (solid black lines) and their envelopes (solid red curves) for two stencils with $m_I = 3$ (left) and $m_I = 15$ (right) internal nodes as the boundary node moves along the blue dashed curve. This last curve represents the points having constant distance s, i.e., the nodal spacing, from the nearest internal node.

Analytic calculation of $\hat{\boldsymbol{n}}$

The formulation given in equation (5.22) for $\hat{\boldsymbol{n}}$, is not practical due to the high computational cost involved in the calculation of the cofactors $C_{m,j}$. In order to develop a more convenient approach we split matrix $\boldsymbol{M}$ into smaller boxes in order to separate the rows associated with internal nodes from those associated with the boundary node:

$$\begin{aligned}\boldsymbol{M} &= \left(\begin{array}{ccc|c} \Phi(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \Phi(\boldsymbol{x}_1,\boldsymbol{x}_{m_I}) & \Phi(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \vdots \\ \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_1) & \cdots & \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_{m_I}) & \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_m) \\ \hline \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_1) & \cdots & \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_{m_I}) & 0 \end{array}\right) \\ &= \left(\begin{array}{c|c} \boldsymbol{\Phi}_{II} & \boldsymbol{\Phi}_{IB} \\ \hline \partial\boldsymbol{\Phi}_{BI} & 0 \end{array}\right)\end{aligned} \tag{5.26}$$

The Schur complement S_{BB} of the block $\boldsymbol{\Phi}_{II}$ is:

$$S_{BB} = -\partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB} \tag{5.27}$$

and for a property of the Schur complement one more formulation for the determinant of matrix $\boldsymbol{M}$ is possible:

$$\det(\boldsymbol{M}) = \det(\boldsymbol{\Phi}_{II})S_{BB} \tag{5.28}$$

We give for granted the nonsingularity of $\boldsymbol{\Phi}_{II}$, it is also important to notice that the Schur complement S_{BB} is efficient to compute since does not require the

evaluation of any determinant. In a sense the formulation with S_{BB} allowed to factor many smaller determinants hidden within the $C_{m,j}$ terms into one larger $\det(\boldsymbol{\Phi}_{II})$.

Because of the symmetry of the RBFs, i.e., $\Phi(\boldsymbol{x}_j, \boldsymbol{x}_m) = \Phi(\boldsymbol{x}_m, \boldsymbol{x}_j)$, the column vector $\boldsymbol{\Phi}_{IB}$ can also be interpreted as the values of the RBFs centered at the m_I internal nodes and evaluated at the boundary node $\boldsymbol{x}_m$. Therefore, the column vector $\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}$ in (5.27) can be interpreted as the vector of cardinal functions $\bar{\psi}_j$ associated to the internal nodes only and evaluated at the boundary node $\boldsymbol{x}_m$ (cfr. definition of cardinal functions in equation (2.19) at page 24):

$$\bar{\boldsymbol{\psi}} = \boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB} \tag{5.29}$$

where $\bar{\boldsymbol{\psi}} = \{\bar{\psi}_1(\boldsymbol{x}_m), \dots, \bar{\psi}_{m_I}(\boldsymbol{x}_m)\}^T$.

It is now possible to perform further substitutions in the formulation of $\det(\boldsymbol{M})$ involving cofactors. This is because, up to the multiplicative constant $-\det(\boldsymbol{\Phi}_{II})$, the cofactors $C_{m,j}$ correspond to the aforementioned cardinal functions $\bar{\boldsymbol{\psi}}$.

$$C_{m,j} = -\det(\boldsymbol{\Phi}_{II})\bar{\psi}_j(\boldsymbol{x}_m)\,, \quad j = 1, \dots, m_I \tag{5.30}$$

which lead to the following formulation of $\hat{\boldsymbol{n}}$ where cofactors do not appear:

$$\hat{\boldsymbol{n}} = -\det(\boldsymbol{\Phi}_{II}) \sum_{j=1}^{m_I} \bar{\psi}_j(\boldsymbol{x}_m)\varphi'(r_{j,m})\boldsymbol{e}_{j,m} \tag{5.31}$$

We conclude by remarking that in (5.31) there is no need to evaluate $\det(\boldsymbol{\Phi}_{II})$ if we are only interested in calculating the optimal direction $\hat{\boldsymbol{n}}$ since it is a scalar and that values $\bar{\psi}_j(\boldsymbol{x}_m)$ can be computed all at once from equation (5.29).

Qualitative interpretation of $\boldsymbol{S_{BB}}$

By transposing equation (5.27), by the symmetry of $\boldsymbol{\Phi}_{II}$ we obtain:

$$S_{BB} = -\boldsymbol{\Phi}_{IB}^T\boldsymbol{\Phi}_{II}^{-1}\partial\boldsymbol{\Phi}_{BI}^T \tag{5.32}$$

which is still a valid expression for the Schur complement.

We start by remarking that entries of $\partial\boldsymbol{\Phi}_{BI}$ are antisymmetric (cfr equation (5.8)):

$$\partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_j) = -\partial\Phi(\boldsymbol{x}_j, \boldsymbol{x}_m) \tag{5.33}$$

contrary to $\Phi(\cdot, \cdot)$, which instead is symmetric, this means that S_{BB} can also be written as:

$$S_{BB} = \Big(\Phi(\boldsymbol{x}_m, \boldsymbol{x}_1) \dots \Phi(\boldsymbol{x}_m, \boldsymbol{x}_{m_I})\Big)\, \boldsymbol{\Phi}_{II}^{-1} \begin{pmatrix} \partial\Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots \\ \partial\Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \end{pmatrix} \tag{5.34}$$

Suppose we only had the internal nodes $\mathcal{X}_I$ and we wanted to approximate the function $u(\cdot) = \partial\Phi(\cdot, \boldsymbol{x}_m)$ at $\boldsymbol{x}_m$, then we would solve the interpolation system arising from the usual conditions $u^h(\boldsymbol{x}_i) = u(\boldsymbol{x}_i)$ with $i \in \mathcal{X}_I$ and then write:

$$u^h(\boldsymbol{x}_m) = \Big(\Phi(\boldsymbol{x}_m, \boldsymbol{x}_1) \dots \Phi(\boldsymbol{x}_m, \boldsymbol{x}_{m_I})\Big) \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{m_I} \end{pmatrix} \tag{5.35}$$

We now proceed by comparing the two equations: we see that the rightmost term in (5.34) are the values taken by the function $\partial\Phi(\cdot, \boldsymbol{x}_m)$ at $\mathcal{X}_I$ and its product with $\boldsymbol{\Phi}_{II}^{-1}$ coincides with the vector of interpolation coefficients $\{\alpha_1, \dots, \alpha_{m_I}\}^T$ of equation 5.35 because $\boldsymbol{\Phi}_{II}$ is exactly the matrix arising from the associated interpolation conditions.

Therefore S_{BB} is $u^h(\boldsymbol{x}_m)$ where u^h is the RBF interpolation of $\partial\Phi(\cdot, \boldsymbol{x}_m)$ attained using the internal nodes. If such interpolation was exact, then matrix $\boldsymbol{M}$ would have been singular since $u(\boldsymbol{x}_m) = \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) = 0$. We conclude that S_{BB} measures the error made when the normal derivative of the RBFs is approximated using the internal nodes only.

Additional remarks

A final remark before concluding the discussion about the case with a single boundary node concerns the following questions. Given a node arrangement for the m_I internal nodes, are there particular locations of the boundary node for which $\det(\boldsymbol{M}) = 0$ regardless of the direction of the normal $\boldsymbol{n}$, i.e., for which $\hat{\boldsymbol{n}} = \boldsymbol{0}$ in equation (5.21) or, equivalently, $S_{BB} = 0$ in equation (5.28), provided that $\det(\boldsymbol{\Phi}_{II}) \neq 0$? If so, is it possible that a Neumann boundary condition ends up being defined in one of such singular locations?

Fortunately, this seems not to be the case: Figure 5.6 shows these singular locations as red dots for different node arrangements in the case of MQ RBF with $\varepsilon_0 = 0.5$, together with a representation of $\|\hat{\boldsymbol{n}}\|$. From this figure it can be observed that all singular locations fall inside the stencil and dangerous locations, i.e., locations of the boundary node for which $\|\hat{\boldsymbol{n}}\|$ is close to 0, are also in the neighbourhood of the stencil. In other words, the further the boundary node is from the internal nodes, the further $\boldsymbol{M}$ is from being singular, provided that the normal is not orthogonal to the optimal direction $\hat{\boldsymbol{n}}$. Whenever we have a boundary node lying outside the region delimited by the internal nodes, we do not need therefore to worry about the singular locations. This principle has been verified numerically for each type of RBF presented in Table 2.1 which is at least conditionally positive definite of order 1, i.e., RBFs that can be used for RBF interpolation without the need of any polynomial augmentation.
This result support the conjecture that Neumann BC-induced ill-conditioning issues are exclusively caused by orthogonality of $\boldsymbol{n}$ and $\hat{\boldsymbol{n}}$.

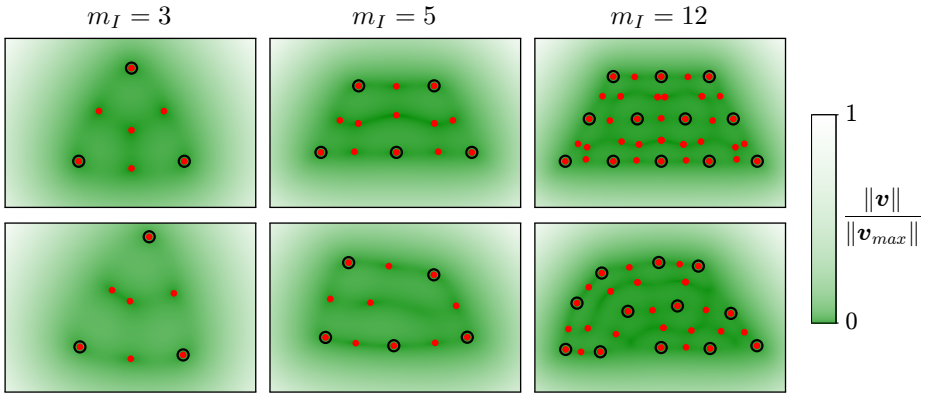


Figure 5.6: norm of $\hat{\boldsymbol{n}}$ for different node arrangements as the position of the boundary node changes. The m_I internal nodes are shown as black empty circles and the red dots represent the positions of the boundary node for which $\hat{\boldsymbol{n}} = \boldsymbol{0}$. Red dots represent therefore forbidden positions for the boundary node. Top row: hexagonal arrangements; bottom row: hexagonal arrangements perturbed by random displacements. $\|\hat{\boldsymbol{n}}_{max}\|$ represents the maximum norm of $\hat{\boldsymbol{n}}$ for each subfigure.

5.3.3 Bare RBF with multiple boundary nodes

Preliminary results

Consider now a stencil with $m_B > 1$ boundary nodes and, again, an interpolant with no polynomial augmentation. The $m \times m$ interpolation matrix is therefore:

$$\boldsymbol{M} = \boldsymbol{\Phi}_{BC} = \begin{pmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \\ \partial\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{pmatrix} \tag{5.36}$$

where $m_J = m_I + 1$.

Similarly to what was done in section 5.3.2, consider the following splitting

of $\boldsymbol{M}$:

$$\boldsymbol{M} = \left(\begin{array}{ccc|ccc} \Phi(\boldsymbol{x}_1,\boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_1,\boldsymbol{x}_{m_I}) & \Phi(\boldsymbol{x}_1,\boldsymbol{x}_{m_J}) & \dots & \Phi(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_{m_I}) & \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_{m_J}) & \dots & \Phi(\boldsymbol{x}_{m_I},\boldsymbol{x}_m) \\ \hline \partial\Phi(\boldsymbol{x}_{m_J},\boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_{m_J},\boldsymbol{x}_{m_I}) & \partial\Phi(\boldsymbol{x}_{m_J},\boldsymbol{x}_{m_J}) & \dots & \partial\Phi(\boldsymbol{x}_{m_J},\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_1) & \dots & \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_{m_I}) & \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_{m_J}) & \dots & \partial\Phi(\boldsymbol{x}_m,\boldsymbol{x}_m) \end{array}\right) = \left(\begin{array}{c|c} \boldsymbol{\Phi}_{II} & \boldsymbol{\Phi}_{IB} \\ \hline \partial\boldsymbol{\Phi}_{BI} & \partial\boldsymbol{\Phi}_{BB} \end{array}\right) \tag{5.37}$$

where the m_B diagonal entries of $\partial\boldsymbol{\Phi}_{BB}$ are zeros.

By taking the Schur complement $\boldsymbol{S_{BB}}$ of $\boldsymbol{\Phi}_{II}$, we obtain the following $m_B \times m_B$ Schur complement matrix:

$$\boldsymbol{S}_{BB} = \partial\boldsymbol{\Phi}_{BB} - \partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB} \tag{5.38}$$

The following equality also holds:

$$\det(\boldsymbol{M}) = \det(\boldsymbol{\Phi}_{II})\det(\boldsymbol{S}_{BB}) \tag{5.39}$$

where, again, $\det(\boldsymbol{\Phi}_{II}) \neq 0$ when using strictly positive definite or conditionally positive definite functions $\Phi(r)$ or order 1.

Qualitative interpretation of $\boldsymbol{S_{BB}}$

In order to repeat the procedure used for the case with $m_B = 1$, the following notation must be introduced, if $\boldsymbol{x}_k$ is one of the boundary nodes in $\mathcal{X}_B$, then we write:

$$\partial_2\Phi(\boldsymbol{x}_i,\boldsymbol{x}_k) := \frac{\partial\Phi}{\partial\boldsymbol{n_k}}(\boldsymbol{x}_i,\boldsymbol{x}_k) \tag{5.40}$$

where $\boldsymbol{n_k}$ is the normal at $\boldsymbol{x}_k$ and the symbol ∂_2 indicates that the derivative is done by considering with respect to the second argument.

Because of the symmetry of the RBFs, from equation (5.8) we have:

$$\partial\Phi(\boldsymbol{x}_k,\boldsymbol{x}_i) = \frac{\partial\Phi}{\partial\boldsymbol{n}}(\boldsymbol{x}_k,\boldsymbol{x}_i) = -\partial_2\Phi(\boldsymbol{x}_i,\boldsymbol{x}_k) \tag{5.41}$$

and then from equation (5.38) we have:

$$\boldsymbol{S}_{BB} = -\big(\partial_2\boldsymbol{\Phi}_{BB} - \partial_2\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}\big) \tag{5.42}$$

where $\partial_2\boldsymbol{\Phi}_{BB}$ and $\partial_2\boldsymbol{\Phi}_{BI}$ are obtained from $\partial\boldsymbol{\Phi}_{BB}$ and $\partial\boldsymbol{\Phi}_{BI}$ respectively, by

replacing ∂ with ∂_2 and exchanging the arguments within each RBF function:

$$\partial_2 \boldsymbol{\Phi}_{BB} = \begin{pmatrix} \partial_2\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_{m_J}) & \cdots & \partial_2\Phi(\boldsymbol{x}_m, \boldsymbol{x}_{m_J}) \\ \vdots & \ddots & \vdots \\ \partial_2\Phi(\boldsymbol{x}_{m_J}, \boldsymbol{x}_m) & \cdots & \partial_2\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{pmatrix} \tag{5.43}$$

$$\partial_2 \boldsymbol{\Phi}_{BI} = \begin{pmatrix} \partial_2\Phi(\boldsymbol{x}_1, \boldsymbol{x}_{m_J}) & \cdots & \partial_2\Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_{m_J}) \\ \vdots & \ddots & \vdots \\ \partial_2\Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) & \cdots & \partial_2\Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \end{pmatrix} \tag{5.44}$$

By following the same arguments presented in the previous section, we have that column j of $\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}$ in equation (5.42) contains the values of the m_I cardinal functions associated with the internal nodes only and evaluated at the j^{th} boundary node.
Row i of $\partial_2\boldsymbol{\Phi}_{BI}$ in equation (5.44), on the other hand, contains the values of the function $\partial_2\Phi(\cdot, \boldsymbol{x}_{m_I+i})$, evaluated at the m_I internal nodes.
Therefore, entry (i, j) of matrix $\partial_2\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}$ in equation (5.42) is the approximation at the j^{th} boundary node $\boldsymbol{x}_{m_I+j}$ of the function $\partial_2\Phi(\cdot, \boldsymbol{x}_{m_I+i})$ achieved using the internal nodes only. On the other hand, entry (i, j) of matrix $\partial_2\boldsymbol{\Phi}_{BB}$ is the actual value of $\partial_2\Phi(\cdot, \boldsymbol{x}_{m_I+i})$ at that same node $\boldsymbol{x}_{m_I+j}$.

Once again, given the definition of ∂_2, we conclude that entry (i, j) of $\boldsymbol{S}_{BB}$ can be interpreted as the error made when the normal derivative $\partial_2\Phi(\cdot, \boldsymbol{x}_{m_I+i})$ is approximated at $\boldsymbol{x}_{M_I+j}$ using the standard interpolation procedure with collocation conditions on the internal nodes only. Similarly to the case with one boundary node, this tells us that the interpolation matrix $\boldsymbol{M}$ in equation (5.36) will be well-conditioned if the additional information provided by the Neumann BCs, i.e., the last m_B rows in $\boldsymbol{M}$, are not redundant with the RBF interpolation based on the internal nodes only. This can therefore be interpreted as a form of linear independence of the rows corresponding with boundary conditions.

Dependance upon the normals

Equation (5.38) can be written as follows:

$$\boldsymbol{S}_{BB} = \partial\boldsymbol{\Phi}_{BB} - \partial\boldsymbol{\Phi}_{BI}\bar{\boldsymbol{\psi}} \tag{5.45}$$

where $\bar{\boldsymbol{\psi}} = \boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}$, already defined in equation (5.29), is now a $m_I \times m_B$ matrix in the case of m_B boundary nodes.

Despite the matrices in equation (5.45) could be handled directly, it is convenient to use the notation introduced in section A.1. By denoting with $\boldsymbol{d}_{i,k}$ the following term appearing in equation (5.8):

$$\boldsymbol{d}_{i,k} = \varphi'(r_{i,k})\,\boldsymbol{e}_{i,k} \tag{5.46}$$

the normal derivatives in $\partial\boldsymbol{\Phi}_{BB}$ and in $\partial\boldsymbol{\Phi}_{BI}$ of equation (5.37) can be written as simple dot products:

$$\partial\Phi(\boldsymbol{x}_k, \boldsymbol{x}_i) = \boldsymbol{d}_{i,k} \cdot \boldsymbol{n_k} \tag{5.47}$$

and therefore $\partial\boldsymbol{\Phi}_{BB}$ and $\partial\boldsymbol{\Phi}_{BI}$ can be written using the operator H, as defined in equation (A.5), as follows:

$$\begin{aligned} \partial\boldsymbol{\Phi}_{BB} &= H(\mathcal{D}_{BB}, \mathcal{N}) \\ \partial\boldsymbol{\Phi}_{BI} &= H(\mathcal{D}_{BI}, \mathcal{N}) \end{aligned} \tag{5.48}$$

In the previous equations $\mathcal{D}_{BB} \in \mathbb{R}^{m_B, m_B}$ and $\mathcal{D}_{BI} \in \mathbb{R}^{m_B, m_I}$ are d-matrices storing purely geometric information:

$$\begin{aligned} \mathcal{D}_{BB} &= \begin{pmatrix} \boldsymbol{d}_{m_J, m_J} & \cdots & \boldsymbol{d}_{m, m_J} \\ \vdots & \ddots & \vdots \\ \boldsymbol{d}_{m_J, m} & \cdots & \boldsymbol{d}_{m, m} \end{pmatrix} \\ \mathcal{D}_{BI} &= \begin{pmatrix} \boldsymbol{d}_{1, m_J} & \cdots & \boldsymbol{d}_{m_I, m_J} \\ \vdots & \ddots & \vdots \\ \boldsymbol{d}_{1, m} & \cdots & \boldsymbol{d}_{m_I, m} \end{pmatrix} \end{aligned} \tag{5.49}$$

and $\mathcal{N} = \{\boldsymbol{n}_{m_J}, \ldots, \boldsymbol{n}_m\} \in \mathbb{R}_d^{m_B}$ is the d-matrix storing the unit normals at the m_B boundary nodes. By considering the vector entries of $\mathcal{D}_{BB}$, we see that $\boldsymbol{d}_{i,k} = -\boldsymbol{d}_{k,i}$ for $k \neq i$ by definition (5.46), while the diagonal entries are zeros because of equation (5.9), i.e., $\mathcal{D}_{BB}$ is an antisymmetric d-matrix.

By using the previous notations and properties (A.6)-(A.7), equation (5.45) becomes:

$$\boldsymbol{S}_{BB} = H(\mathcal{D}_{BB}, \mathcal{N}) - H(\mathcal{D}_{BI}, \mathcal{N})\bar{\boldsymbol{\psi}} = H(\underbrace{\mathcal{D}_{BB} - \mathcal{D}_{BI}\bar{\boldsymbol{\psi}}}_{\mathcal{G}_{BB}}, \mathcal{N}) \tag{5.50}$$

where $\mathcal{G}_{BB} = (\boldsymbol{g}_{ij}) \in \mathbb{R}^{m_B, m_B}$.

By comparing equation (5.50) with interpretation of $\boldsymbol{S}_{BB}$ given in section 5.3.3 we conclude that vector $\boldsymbol{g}_{ij}$ of $\mathcal{G}_{BB}$ represents the difference between the gradient of the RBF $\Phi(\cdot, \boldsymbol{x}_{m_I+i})$ and its approximation obtained through a RBF interpolation based on the m_I internal nodes only, both evaluated at the j^{th} boundary node $\boldsymbol{x}_{m_I+j}$.

The main reason for deriving equation (5.50) is that the determinant of $\boldsymbol{S}_{BB}$ can now be explicitly written as follows:

$$\det(\boldsymbol{S}_{BB}) = \det\big(H(\mathcal{G}_{BB}, \mathcal{N})\big) = \begin{vmatrix} \boldsymbol{g}_{1,1} \cdot \bar{\boldsymbol{n}}_1 & \cdots & \boldsymbol{g}_{1, m_B} \cdot \bar{\boldsymbol{n}}_1 \\ \vdots & \ddots & \vdots \\ \boldsymbol{g}_{m_B, 1} \cdot \bar{\boldsymbol{n}}_{m_B} & \cdots & \boldsymbol{g}_{m_B, m_B} \cdot \bar{\boldsymbol{n}}_{m_B} \end{vmatrix} \tag{5.51}$$

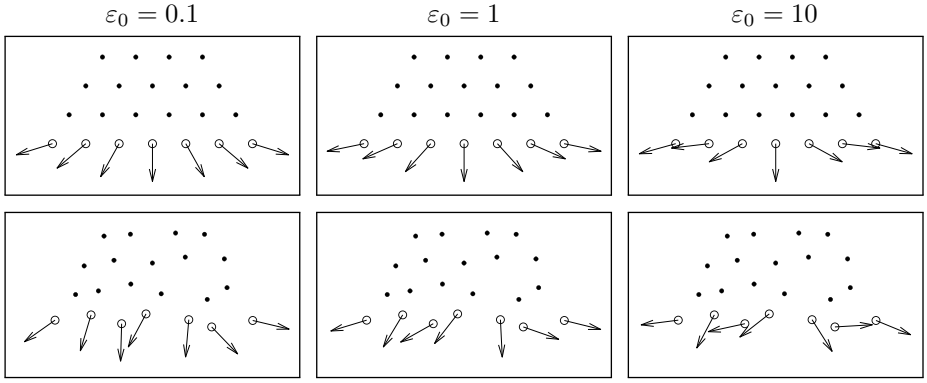


Figure 5.7: optimal directions for the reference stencil of Figure 5.1 (top row) and for the same stencil with random nodal displacements (bottom row).

where the notation $\bar{\boldsymbol{n}}_i$ (with a bar) indicates the unit normal at the i^{th} boundary node and is introduced for reducing the number of indices: $\bar{\boldsymbol{n}}_i = \boldsymbol{n}_{m_I+i}$. In the case with $m_B = 1$ boundary nodes, equation (5.51) becomes:

$$\det(\boldsymbol{S}_{BB}) = S_{BB} = \boldsymbol{g}_{1,1} \cdot \bar{\boldsymbol{n}}_1 \tag{5.52}$$

which corresponds to equation (5.21) with $\hat{\boldsymbol{n}} = \det(\boldsymbol{\Phi}_{II})\boldsymbol{g}_{1,1}$.

Optimal directions

Given the formulation of equation (5.51), one may ask which is the set $\hat{\mathcal{N}}$ of unit vectors $\hat{\mathcal{N}} = \{\hat{\boldsymbol{n}}_1, \ldots, \hat{\boldsymbol{n}}_{m_B}\}$ that maximize $\det(\boldsymbol{S}_{BB})$, i.e., the optimal directions of the m_B normals leading to a well-conditioned interpolation matrix. Such optimal directions can be obtained by solving the following constrained maximization problem:

$$\begin{aligned} &\arg\max_{\hat{\mathcal{N}}} && \det\big(H(\mathcal{G}_{BB}, \mathcal{N})\big) \\ &\text{subject to} && \|\hat{\boldsymbol{n}}_i\|_2^2 = 1, \quad i = 1, \ldots, m_B \end{aligned} \tag{5.53}$$

Problem (5.53) can be solved by the Lagrange multipliers method as presented in A.2. Such optimal directions are shown in Figure 5.7 for the reference stencil of Figure 5.1 when using MQ RBF with different values of the shape factor ε_0. In order to highlight the effect of the relative position of the nodes, optimal directions are also shown when the reference stencil is perturbed with random nodal displacements.

In the case of the reference stencil, the optimal directions are approximately pointing towards some reference mid point of the stencil, which however depends upon the shape factor ε_0, analogously to the case with one boundary node depicted in Figure 5.4. When $\varepsilon_0 = 0.1$, the long range influence of the internal

nodes on the optimal normals seems to dominate: as a result, the optimal directions have a radial pattern originating from the geometrical center of the internal nodes. For larger shape factors, conversely, the short range mutual influence of the boundary nodes seems to dominate, leading to a more flat pattern. In the case of the perturbed stencil, instead, the optimal directions are quite unpredictable, as there are boundary nodes close to each other with quite different optimal directions. This last remark suggests that a simple geometric approximation of the optimal directions is not safe when nodes are not scattered according to some regular pattern.

Figure 5.7 suggest that the likelihood of a singularity of matrix M caused by the actual normals being orthogonal to the optimal ones is larger for higher values of ε_0. This idea is suggested by the fact that optimal normals at high ε_0 are almost parallel to the boundary for the reference stencil whereas those at lower ε_0 are somewhat pointed outwards. Unfortunately this phenomenon is compensated by the ill-conditioning of matrix M induced by flattness of RBFs at low values of ε_0, for instance, stationary interpolation schemes with $\varepsilon_0 = 0.1$ are already almost unsolvable using double precision arithmetic in the standard procedure.

5.3.4 Influence of polynomial augmentation

In order to also take into account the effects of polynomial augmentation, let us consider again the interpolation matrix $\boldsymbol{M}$ in equation (5.4). By using the same matrix splitting employed in equation (5.37), submatrix $\boldsymbol{\Phi}_{BC}$ initially defined in equation (5.5) can be expressed as:

$$\boldsymbol{\Phi}_{BC} = \left(\begin{array}{c|c} \boldsymbol{\Phi}_{II} & \boldsymbol{\Phi}_{IB} \\ \hline \partial\boldsymbol{\Phi}_{BI} & \partial\boldsymbol{\Phi}_{BB} \end{array} \right) \tag{5.54}$$

while matrices $\boldsymbol{P}_{BC}$ and $\boldsymbol{P}^T$, defined in equations (5.6) and (5.7) are:

$$\boldsymbol{P}_{BC} = \left(\begin{array}{ccc} p_1(\boldsymbol{x}_1) & \cdots & p_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_{m_I}) & \cdots & p_M(\boldsymbol{x}_{m_I}) \\ \hline \partial p_1(\boldsymbol{x}_{m_J}) & \cdots & \partial p_M(\boldsymbol{x}_{m_J}) \\ \vdots & \ddots & \vdots \\ \partial p_1(\boldsymbol{x}_m) & \cdots & \partial p_M(\boldsymbol{x}_m) \end{array} \right) = \left(\begin{array}{c} \boldsymbol{P}_I \\ \hline \partial\boldsymbol{P}_B \end{array} \right) \tag{5.55}$$

$$\boldsymbol{P}^T = \left(\begin{array}{ccc|ccc} p_1(\boldsymbol{x}_1) & \cdots & p_1(\boldsymbol{x}_{m_I}) & p_1(\boldsymbol{x}_{m_J}) & \cdots & p_1(\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_M(\boldsymbol{x}_1) & \cdots & p_M(\boldsymbol{x}_{m_I}) & p_M(\boldsymbol{x}_{m_J}) & \cdots & p_M(\boldsymbol{x}_m) \end{array} \right) = \left(\begin{array}{c|c} \boldsymbol{P}_I^T & \boldsymbol{P}_B^T \end{array} \right) \tag{5.56}$$

By rearranging rows and columns of $\boldsymbol{M}$ in order to separate internal nodes, polynomial terms and boundary nodes, the interpolation matrix in block form becomes:

$$\boldsymbol{M} = \left(\begin{array}{cc|c} \boldsymbol{\Phi}_{II} & \boldsymbol{P}_I & \boldsymbol{\Phi}_{IB} \\ \boldsymbol{P}_I^T & \boldsymbol{0} & \boldsymbol{P}_B^T \\ \hline \partial\boldsymbol{\Phi}_{BI} & \partial\boldsymbol{P}_B & \partial\boldsymbol{\Phi}_{BB} \end{array}\right) = \left(\begin{array}{c|c} \boldsymbol{M}_{II} & \boldsymbol{Q}_{IB} \\ \hline \partial\boldsymbol{Q}_{BI} & \partial\boldsymbol{\Phi}_{BB} \end{array}\right) \tag{5.57}$$

Similarly to the approach employed in section 5.3.3, we consider the Schur complement $\boldsymbol{S}_{BB}$ of the $\boldsymbol{M}_{II}$ block of $\boldsymbol{M}$:

$$\boldsymbol{S}_{BB} = \partial\boldsymbol{\Phi}_{BB} - \underbrace{\partial\boldsymbol{Q}_{BI}\boldsymbol{M}_{II}^{-1}\boldsymbol{Q}_{IB}}_{\boldsymbol{W}_{BB}} \tag{5.58}$$

where $\boldsymbol{M}_{II}$ is not singular as long as the internal stencil is unisolvent for the chosen degree of the polynomial augmentation.

By comparing equation (5.58) with the corresponding equation (5.38), obtained without polynomial augmentation, we see that the only difference is in the double matrix-product term $\boldsymbol{W}_{BB}$, that in the case of polynomial augmentation can be written as follows:

$$\boldsymbol{W}_{BB} = \begin{pmatrix} \partial\boldsymbol{\Phi}_{BI} & \partial\boldsymbol{P}_B \end{pmatrix} \begin{pmatrix} \boldsymbol{\Phi}_{II} & \boldsymbol{P}_I \\ \boldsymbol{P}_I^T & \boldsymbol{0} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{\Phi}_{IB} \\ \boldsymbol{P}_B^T \end{pmatrix} = \partial\boldsymbol{\Phi}_{BI}\boldsymbol{K}_1 + \partial\boldsymbol{P}_B\boldsymbol{K}_2 \tag{5.59}$$

where $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are:

$$\begin{aligned} \boldsymbol{K}_1 &= \boldsymbol{\Phi}_{II}^{-1}\big(\boldsymbol{\Phi}_{IB} - \boldsymbol{P}_I\boldsymbol{K}_2\big) \\ \boldsymbol{K}_2 &= \underbrace{\big(\boldsymbol{P}_I^T\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{P}_I\big)^{-1}}_{\boldsymbol{S}} \big(\boldsymbol{\Phi}_{IB}^T\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{P}_I - \boldsymbol{P}_B\big)^T \end{aligned} \tag{5.60}$$

Further algebraic manipulations lead to the following form:

$$\boldsymbol{W}_{BB} = \partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB} - \underbrace{\big(\partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{P}_I - \partial\boldsymbol{P}_B\big)\boldsymbol{K}_2}_{\Delta\boldsymbol{W}} \tag{5.61}$$

which, compared to the corresponding term $\partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{\Phi}_{IB}$ in equation (5.38), reveals that the difference in the Schur complement of the interpolation matrix, with and without polynomial augmentation, is solely due to the term $\Delta\boldsymbol{W}$. Substituting $\boldsymbol{K}$ into $\boldsymbol{\Delta W}$ we write:

$$\Delta\boldsymbol{W} = \underbrace{\big(\partial\boldsymbol{\Phi}_{BI}\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{P}_I - \partial\boldsymbol{P}_B\big)}_{\partial\boldsymbol{E}}\boldsymbol{S}\big(\underbrace{\boldsymbol{\Phi}_{IB}^T\boldsymbol{\Phi}_{II}^{-1}\boldsymbol{P}_I - \boldsymbol{P}_B}_{\boldsymbol{E}}\big)^T \tag{5.62}$$

Similarly to the qualitative interpretations given to the Schur complement in the previous sections, we can also interpret terms $\boldsymbol{E}$ and $\boldsymbol{\partial E}$. Entry (i, j) of

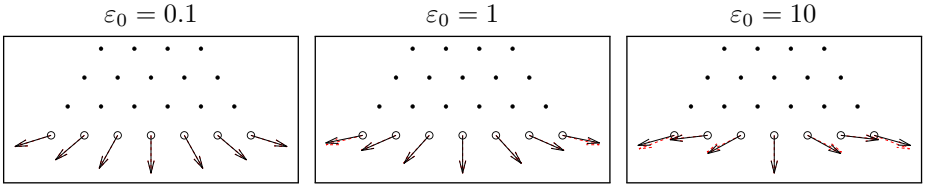


Figure 5.8: optimal directions for the reference stencil of Figure 5.1 without polynomial augmentation (black solid vectors) and with polynomial augmentation with degree $P = 2$ (red dashed vectors).

matrix $\partial \boldsymbol{E}$ can be understood as the error in the approximation of the value $\partial p_j(\boldsymbol{x}_{m_I+i})$ at the i^{th} boundary node obtained through a RBF interpolation based on the m_I internal nodes only. On the other hand, entry (i, j) of matrix $\boldsymbol{E}$ can be interpreted as the error in the approximation of $p_j(\boldsymbol{x}_{m_I+i})$ obtained through an RBF interpolation based on the m_I internal nodes only.

If the internal stencil is unisolvent for the chosen degree of the polynomial augmentation, each entry of matrices $\partial \boldsymbol{E}$ and $\boldsymbol{E}$ vanishes as $\varepsilon \to 0$ and the same holds for the entries of matrix $\boldsymbol{S}$ except for the diagonal entry corresponding to the constant monomial of the polynomial basis, which tends to a constant value. We argue that in practice matrix $\Delta \boldsymbol{W}$ vanishes rapidly as $\varepsilon \to 0$, and in the usable range $\varepsilon_0 \in [0.2, 0.6]$ its value is small enough that the role of polynomial augmentation in the insurgence of ill-conditioning due to Neumann BC can be rightfully neglected.

The influence of the polynomial augmentation, with degree $P = 2$, on the optimal directions for the reference stencil of Figure 5.1 is shown in Figure 5.8 in the case of MQ RBF for different values of the shape parameter ε_0. In the case $\varepsilon_0 = 0.1$ there is no noticeable difference in the optimal directions with and without polynomial augmentation, confirming the previous theoretical observations that the influence of the polynomial augmentation is negligible for sufficiently small values of ε. For larger values of ε_0, i.e., in the cases $\varepsilon_0 = 1$ and $\varepsilon_0 = 10$, the differences in the optimal direction start to be visually noticeable, although very small and almost negligible from a practical point of view.

In conclusion we remark that, since in practical cases $\varepsilon_0 \in [0.2, 0.6]$ when MQ RBF-FD method is employed and $m > 2M$ in equation (4.17), the calculation of the optimal directions can be reasonably carried out without taking into account the polynomial terms.

5.4 Techniques for an improved interpolation

For reference, we report here the proposed stabilization approaches introduced in the case of stencils with a single boundary node:

Approach 1. Discard a boundary node if the corresponding dot product $|\hat{\boldsymbol{n}} \cdot \boldsymbol{n}|/\|\hat{\boldsymbol{n}}\|_2$ is too small, where $\hat{\boldsymbol{n}}$ is the optimal boundary normal.

Approach 2. Change the node locations in order to avoid configurations where, for some boundary node $|\hat{\boldsymbol{n}} \cdot \boldsymbol{n}|/\|\hat{\boldsymbol{n}}\|_2$ is too small.

5.4.1 Approach 1: boundary node selection based on optimal directions

Once the optimal directions $\hat{\boldsymbol{n}}_i$ have been calculated according to equation (5.53), it is possible to extend Approach 1 in section 5.3.2 to the case with multiple boundary nodes, i.e., discarding from the stencil those boundary nodes whose normal vectors are too different from the corresponding optimal normals. This strategy allows to rule out the possibility of a singular interpolation matrix $\mathbf{M}$ by dropping both the rows and columns associated with dangerous nodes. We remark that discarded nodes are not eliminated from the node cloud, they simply are not included in the present stencil, very often such nodes are quite far from the center node and will be included in the neighboring stencils. It is very unlikely that some boundary node is discarded from all stencils built around neighboring inner nodes, therefore boundary information should be preserved. Furthermore, whenever some boundary node is discarded from a given stencil centered around node $\boldsymbol{x}_i$, it can be replaced by looking for the nearest node to $\boldsymbol{x}_i$ not yet included in that same stencil.

The boundary node selection can be implemented as an iterative process (see A.2.3) that allows to discard the i^{th} boundary node whenever the dot product between the associated normal $\bar{\boldsymbol{n}}_i$ and the corresponding optimal direction $\hat{\boldsymbol{n}}_i$ is less than a certain threshold value d_{min}:

$$\text{discard boundary node } i \text{ if } |\bar{\boldsymbol{n}}_i \cdot \hat{\boldsymbol{n}}_i| < d_{min} \tag{5.63}$$

where the absolute value comes from the fact that the reversal of any normal has no effect on the interpolation.

The condition number $\kappa(\mathbf{M})$, the Lebesgue constant Λ_I and the number of removed nodes N_{rem} for the reference stencil of Figure 5.1 are shown in Figure 5.9 as functions of the angle α. From a comparison with Figure 5.2 it emerges that all singularities for $\alpha < 0$ have been deleted. Both $\kappa(\mathbf{M})$ and Λ_I become bounded and a more stable interpolation is attained. Figure 5.9 is obtained with MQ RBF ($\varepsilon_0 = 0.5$) and with a threshold value $d_{min} = 0.6$ in equation (5.63). Lower values are not able to bound the Lebesgue constant Λ_I when α approaches the limit values of $\pi/2$ and $-\pi/2$. When 6 nodes are removed, only the central boundary node associated to a vertical normal is included in the stencil. The algorithm removes a number of boundary nodes N_{rem} which is always even because of the symmetry of the reference stencil.

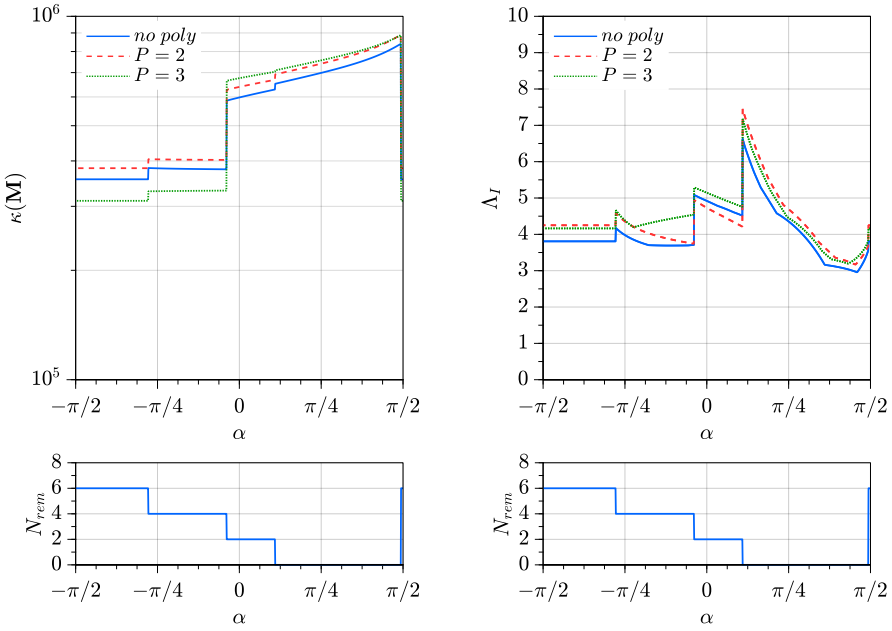


Figure 5.9: condition number $\kappa(\mathbf{M})$ of the interpolation matrix (top left), and Lebesgue constant Λ_I (top right) for the RBF interpolation on the reference stencil of Figure 5.1 when the selection of boundary nodes is based on the optimal directions. Bottom row: number of removed nodes N_{rem} (the same plot is reported twice in order to simplify comparisons).

5.4.2 Approach 2: optimal placement for boundary nodes

As hinted in section 5.3.2, another strategy for avoiding a singular interpolation matrix consists in moving the stencil nodes in an appropriate way. It follows that any implementation of this strategy involves a modification of the node placement which also depends on the direction of the normals.

In section A.3 the optimization of the boundary node position is discussed, it emerges that the interpolation can be improved by placing boundary nodes in a very specific manner. For each boundary node there should be a paired inner node in its immediate vicinity and the two should be perfectly alingned along the normal direction. In in Figure A.1 the projected nodes are depicted as asterisks, when the curvature of the boundary is limited, in general they provide a very good approximation for an optimal locations for boundary nodes. In practice, the adoption of the projected boundary nodes can be implemented with minimal computational cost, here we discuss the attained improvement on the interpolation properties when this is done for the reference stencil of Figure 5.1.

In Figure 5.10 the condition number $\kappa(\mathbf{M})$ and the Lebesgue constant Λ_I are plotted against the angle α when the boundary nodes of the reference stencil are replaced by the projected ones. The angle α is bounded within a smaller interval

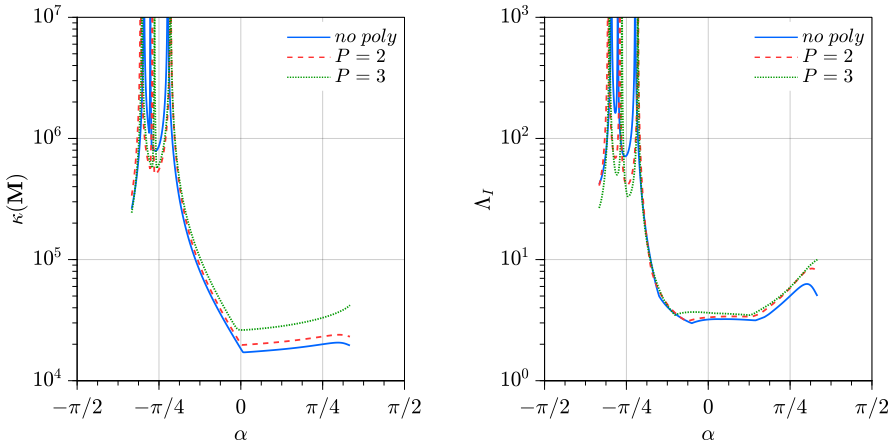


Figure 5.10: condition number of the interpolation matrix (left) and Lebesgue constant Λ_I (right) for the RBF interpolation on the reference stencil of Figure 5.1 with projected boundary nodes.

$\alpha \in [-\pi/3\,,\,\pi/3]$, for values outside this interval the projected nodes end up being either too close together or too far from one another. From Figure 5.10 emerges that, for moderate angles, i.e. approx $\alpha \geq -\pi/8$, the node projection alone prevents the appearance of ill-conditioning problems.

Finally, since the use of the projected boundary nodes does not impact the solution process, it can be applied in conjunction with the selection based on the optimal normals explained above without any undesirable effect.

In Figure 5.11 the interpolation error obtained with the different techniques described so far is displayed against the angle α. The interpolation error $|u(\boldsymbol{x}_{eval}) - u^h(\boldsymbol{x}_{eval})|$ is evaluated at the point $\boldsymbol{x}_{eval} = (s/2, s/2)$, willingly chosen not to be on the symmetry axis, where s is the distance between any two inner nodes of the reference stencil of Figure 5.1 and the analytic function is defined as $u = \exp(x + 2y)$. For the selection of boundary nodes based on the optimal directions (right column) the same settings were used as in Figure 5.9.

We can see once again that both stabilization techniques succeed in improving the interpolation for small variations of the angle α. For the projection of the boundary nodes, α varies within the interval $\alpha \in [-\pi/4\,,\,\pi/4]$ for the reasons explained above. We can see that the spikes in the errors on the left column, corresponding to singular configurations, are located as in Figures 5.2 and 5.10, this confirms the reliability of the Lebesgue constant Λ_I as a measure for accuracy and stability of the interpolation scheme.

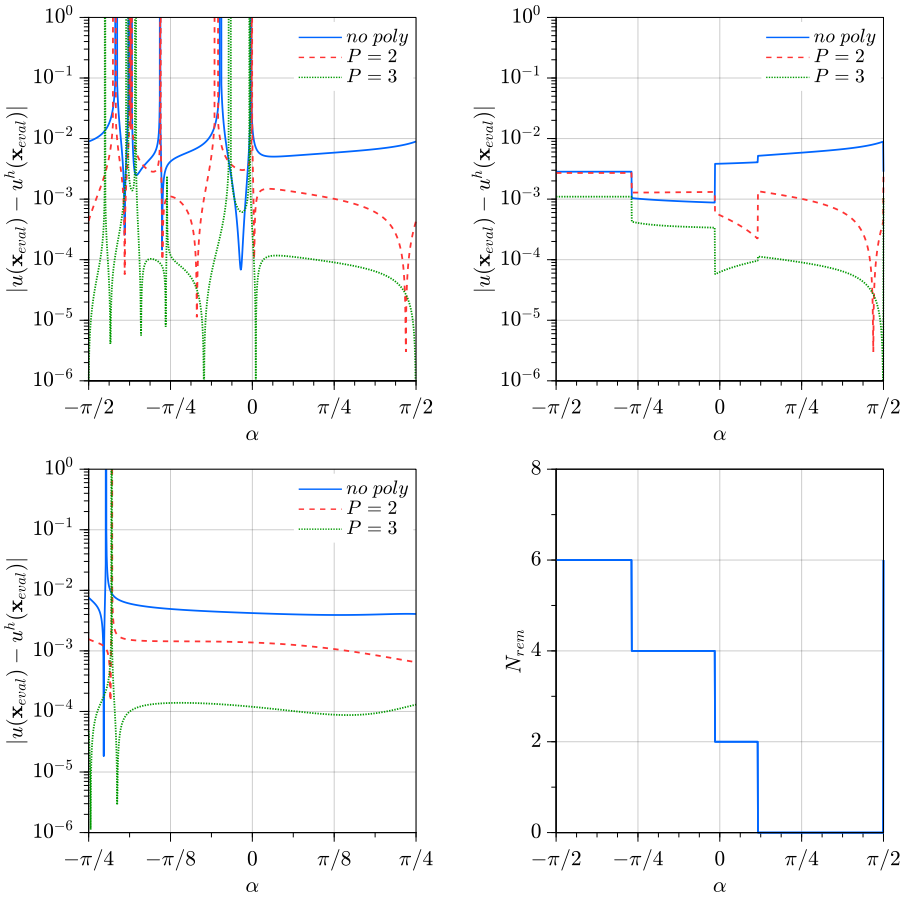


Figure 5.11: interpolation error $|u(\boldsymbol{x}_{eval}) - u^h(\boldsymbol{x}_{eval})|$ at a given location $\boldsymbol{x}_{eval} = (s/2, s/2)$ for the reference stencil of Figure 5.1. No improvement strategy (top left), node selection based on the optimal directions (top right) with the corresponding number N_{rem} of removed boundary nodes (bottom right), boundary node projection (bottom left). The interpolated function is $u = \exp(x + 2y)$.

5.5 Applications

5.5.1 Stability of the Helmholtz-Hodge Decomposition

In this section the boundary node selection and the node projection strategies, discussed above, are applied to the stabilization of the Helmholtz-Hodge decomposition (HHD) [51]. The HHD plays a pivotal role in many theoretical and practical applications [7], among others, it is at the base of the projection methods for the numerical solution of the incompressible Navier-Stokes equations [15, 17]. According to the HHD, under certain hypothesis, any vector field $\boldsymbol{u}^*$ can be expressed as a sum of the gradient of a scalar potential ϕ and a divergence-free vector field $\boldsymbol{u}$:

$$\boldsymbol{u}^* = \nabla\phi + \boldsymbol{u} \tag{5.64}$$

When solving incompressible Navier-Stokes equations through projection methods, the computation of the unknown velocity field $\boldsymbol{u}$ is carried out iteratively or by marching in time. In any case, each iteration or time step is subdivided into two substeps: first an intermediate velocity $\boldsymbol{u}^*$ is obtained from the momentum equations, then $\boldsymbol{u}$ is obtained from equation (5.64) as follows:

$$\boldsymbol{u} = \boldsymbol{u}^* - \nabla\phi \tag{5.65}$$

where the unknown scalar potential ϕ is calculated from equation (5.66), that results from taking the divergence of equation (5.64), recalling $\nabla \cdot \boldsymbol{u} = 0$:

$$\nabla^2\phi = \nabla \cdot \boldsymbol{u}^* \tag{5.66}$$

In the case of known velocity at the boundary, e.g., in a cavity, homogeneous Neumann BC must be enforced in order to solve Poisson equation (5.66):

$$\partial\phi/\partial\boldsymbol{n} = 0 \tag{5.67}$$

Since the HHD expressed by equations (5.65)-(5.66) is repeated many times, once for every iteration (or time step), it is critical that the discretization of such equations is stable, i.e., it does not introduce any spurious mode which grows indefinitely, thus compromising the whole simulation. In order to assess the stability properties of the strategies presented in this work, it was decided to apply multiple times the projection scheme described in equations (5.65)-(5.66) starting from a given vector field $\boldsymbol{w}$, defined on the complex-shaped domain depicted in Figure 5.12. The boundary is characterized by locally convex and concave features with varying curvature, designed to trigger ill-conditioning issues described in section 5.3. A Lagrange multiplier is employed to solve the linear systems arising from the RBF-FD discretization of equation (5.66) since Neumann BCs are prescribed on the whole boundary [121].

The pseudocode for the stability test is presented in Algorithm 2. It is worth mentioning that any conservative discretization scheme, e.g., FVM, implicitly satisfies the previous stability test since the value of the discretized divergence operator vanishes after the first iteration. On the other hand, the basic formulation of the RBF-FD discretization, as presented in chapter 4, is not conservative,

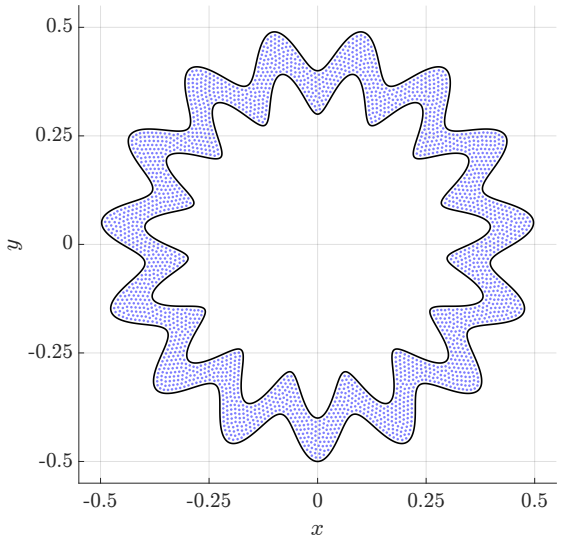


Figure 5.12: test domain with $N_I \approx 3,300$ internal nodes.

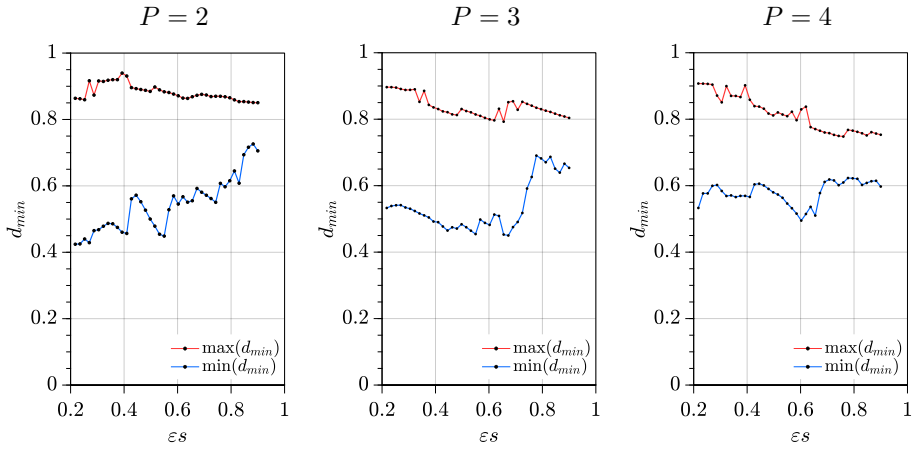


Figure 5.13: stability range for the d_{min} parameter in the case of approach 1, i.e., boundary node selection based on optimal directions. Upper and lower bounds for d_{min} values are displayed with red and blue curves, respectively.

therefore the value of the discretized divergence operator never vanishes at the nodes. Algorithm 2 might therefore diverge if the Poisson equation is solved with the basic RBF-FD method without a proper treatment of the Neumann BC.

Since the equations involved in Algorithm 2 are linear, the choice of the initial vector field $\boldsymbol{w}$ does not affect the stability of the iterative process. The employed initial vector field $\boldsymbol{w}$ is chosen to be an irrotational vortex for simplicity: the

tangential velocity is $w_\theta = r^{-1}$ where r is the distance from the origin.

The results for approach 1 presented in section 5.4.1, i.e., boundary node selection, are shown in Figure 5.13 in the case of MQ RBF with polynomial degrees $P = 2, 3, 4$ and for different values of the shape parameter ε, such that $\varepsilon s \in [0.2, 0.9]$. The total number of nodes inside the domain is $N_I \approx 15,000$. The number of internal nodes for each stencil is chosen to follow the rule $m_I = 2q$ [3]: $m_I = 20$ for $P = 3$ and $m_I = 30$ for $P = 4$. In the case $P = 2$ it was decided to use slightly larger stencils with $m_I = 15$ internal nodes, since this enhanced stability. In Figure 5.13 the values of d_{min} allowing a stable computation were identified as those lying between the two curves. As expected, too small values of d_{min}, e.g., $d_{min} < 0.2$, lead to instability due to stencils with boundary nodes whose actual normal is too different from the corresponding optimal direction, resulting in ill-conditioned local interpolants. On the other hand, too large values of d_{min}, e.g., $d_{min} > 0.95$, lead to instability due to stencils with too few boundary nodes enforcing the prescribed Neumann BCs (5.67). From the same figure it is possible to observe that the stability range is slightly reduced when P is increased from $P = 2$ to $P = 4$, while a general and reasonably good choice is $d_{min} = 0.7$.

In the case of approach 2 presented in section 5.4.2, i.e., projected boundary nodes, stability is always attained for $P = 2, 3, 4$ in the range $\varepsilon_0 \in [0.2, 0.9]$ previously considered, highlighting the remarkable stabilization effect of this approach. Stability issues are still encountered for shape factors outside this range.

Figure 5.14 shows the effect of the proposed stabilization approaches on the condition number of the local interpolation matrix, in the case of MQ RBF with $\varepsilon s = 0.5$ and $P = 3$ (results with different shape parameters ε and polynomial degrees P are qualitatively similar). As expected from the theoretical observations presented in section 5.3, stencils close to locally concave boundaries can be very ill-conditioned if no stabilization technique is employed (red nodes in Figure 5.14, left), leading to large errors and stability issues.

By using the stabilization approach 1 with $d_{min} = 0.7$, this issue is properly addressed and ill-conditioned stencil are no longer present (Figure 5.14, center). The same result is attained by employing the stabilization approach 2 (Figure 5.14, right). In the latter case we observe an increase in the condition number for the stencils close to locally convex boundaries: this is due to the employed projection technique that is performed from the boundary inward. Therefore, internal nodes projected from locally convex boundaries are very close to each other, leading

Algorithm 2 Pseudocode for the stability test of repeated HHD

1: Initialize: $\boldsymbol{u}^{(1)} \leftarrow \boldsymbol{w}$
2: **for** $i = 1, 2, \ldots$ **do**
3: Solve $\nabla^2 \phi = \nabla \cdot \boldsymbol{u}^{(i)}$ for ϕ with homogeneous Neumann BCs
4: Update: $\boldsymbol{u}^{(i+1)} \leftarrow \boldsymbol{u}^{(i)} - \nabla \phi$
5: **end for**

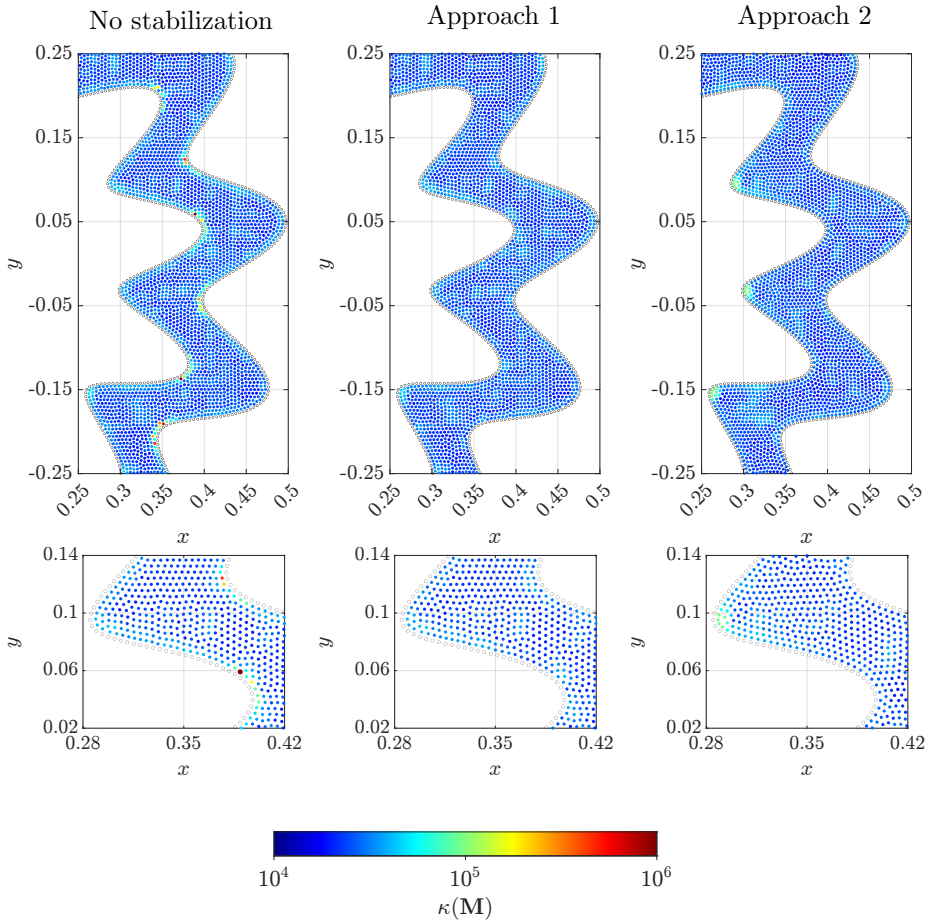


Figure 5.14: condition number of the local interpolation matrix. Top row, from left to right: no stabilization, boundary selection based on optimal directions, boundary projection. Bottom row: enlarged visualization of a portion of the plots of the top row.

to an increased condition number. This geometrical issue is common to any simple geometrical projection technique, regardless of the direction and starting locations employed for the projection. With reference to arbitrary geometries, such problem can only be addressed by taking into account the need for projected boundary nodes already at the node generation phase, thus requiring a proper modification of the node generation algorithm.

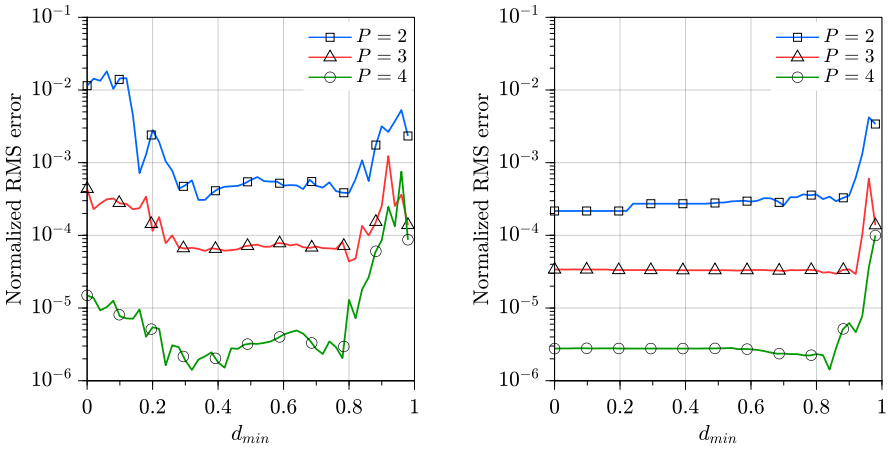


Figure 5.15: Solution error for the Poisson problem (5.68) when employing stabilization approach 1: initial node distribution (left), projected boundary nodes (right).

5.5.2 Accuracy

Besides the stability considerations presented in the previous section, we also provide some insight about the influence of the proposed stabilization approaches on the accuracy of the RBF-FD discretization. In order to do so, we consider the following Poisson equation:

$$\nabla^2 u = f \tag{5.68}$$

defined over the previously employed domain depicted in Figure 5.12. Neumann BCs are prescribed on the whole boundary:

$$\frac{\partial u}{\partial \boldsymbol{n}} = g \tag{5.69}$$

and the analytic solution is chosen to be $u = r^{-1}$, where r is the distance from the origin. The node distributions and the parameters for the RBF-FD discretization are the same as those used in the previous section 5.5.1, and a Lagrange multiplier is again employed to solve the corresponding discretized linear system.

The normalized RMS errors are shown in Figure 5.15 when approach 1 is applied to both node distributions without and with projected boundary nodes. The latter case is therefore a combination of both the proposed stabilization techniques. In the former case, and for all polynomial degrees $P = 2, 3, 4$, the error is lower when d_{min} ranges from 0.4 to 0.8. As expected, the error grows when d_{min} is decreased below 0.4 because of the increasing number of ill-conditioned stencils near the boundary. The limit case $d_{min} = 0$ corresponds indeed to no stabilization action at all. On the other hand, the error also grows

when d_{min} is increased beyond 0.8 because of the increasing number of boundary stencils with an insufficient number of boundary nodes where BCs are enforced.

When both the stabilization approaches are employed, Figure 5.15, the increase in the error for $d_{min} < 0.4$ is no longer present since the ill-conditioning issues are already addressed by the projected nodes on the boundary. The increased error for $d_{min} > 0.8$, instead, is still present and is caused by the fact that too few boundary nodes are included in the boundary stencils. In this case we also observe that the boundary node selection based on the optimal directions does not affect the accuracy of the discretization at all if $d_{min} < 0.8$, as expected.

5.6 Conclusions

In conclusion, the main pros and cons of the two stabilization strategies are the following:

- Approach 1 is the most robust and does not require any control over the node placement. However, the price to pay for its application is a higher computational effort, required to estimate the optimal directions for all the boundary stencils. We remark that this cost can be significantly reduced if the optimal directions are approximated with lower accuracy.

- Approach 2 does not involve any additional computation and is even more effective at ensuring the stabilization of the RBF-FD method. Unfortunately, it requires the implementation of a more sophisticated node generation algorithm.

At last we remark that both approaches can be applied without any modification to the 3D cases, where all of the considerations made in this chapter remain valid. The application of the stabilization techniques described above does not require the user to intervene during the simulation process. Even the d_{min} parameter can be set once for all to be $d_{min} = 0.7$ without incurring in any undesired effect. This fact is especially important since it preserves the main advantage of the RBF-FD meshless method, namely high geometrical flexibility and ease of automation.

Furthermore, the presented approaches can also be adopted simultaneously in order to further increase the robustness of the unsymmetric RBF-FD method.

Chapter 6

Verification Test

6.1 Introduction

In order to assess the performance achieved by the RBF-FD scheme stabilized against ill-conditioning induced by Neumann BC, the resulting implementation has been subjected to some tests. In [78], the theory of verification and validation (V&V) of CFD codes is discussed, according to that terminology the present chapter can be regarded as the report of a verification test focused on the algorithm only, where any details concerning the software quality testing are omitted. The reader will be able to assess the level of accuracy of the present implementation of the RBF-FD meshless method, when applied to two different laminar natural convection problems in 3D.

The first verification test to be presented is a natural convection problem in a differentially heated cubic cavity. While an abundant literature is available for the 2D case of the square cavity, the 3D version, is much less common. Indeed, we were able to select five cases in which the problem is solved for Rayleigh number $10^3 \leq Ra \leq 10^5$, here follows a brief review of the literature. In [104] the very same problem is solved using a pseudo-spectral Chebyshev algorithm based on the projection-diffusion method for $10^3 \leq Ra \leq 10^7$ on a fine discretization grid. The authors provide very clear definitions for a number of characteristic values in order to simplify any future comparison with other solutions. Because of the well-known accuracy of the adopted numerical method, the amount of available results and their clear presentation, we will consider [104] as the main benchmark reference for this test case. In [46] the discretization is achieved via a finite difference procedure, the iterative SIMPLE algorithm [80] is adopted along with the Strongly Implicit Scheme (SIP) [96], the convection terms are treated by the QUICK methodology [61, 45]. This paper was published much earlier and reports the results for $10^3 \leq Ra \leq 10^6$ for the 3D case and, compared to [104], a coarser grid along with a less accurate numerical method were employed. In [81] the Lattice Boltzmann method is used with $D3Q15$ and $D3Q19$ particle velocity models. Full 3D results are reported for $10^3 \leq Ra \leq 10^4$ and partial

(representative values on the symmetry plane) for $Ra = 10^5$. In the present paper the results attained using the $D3Q19$ model are reported for comparison since it was indicated by the author as the best fit for the investigated case. In [68] a differential quadrature (DQ) method is proposed to solve the problem with the cubic cavity inclined at different angles by using the velocity-vorticity formulation. Complete results are provided by the authors for $10^3 \leq Ra \leq 10^5$. The results corresponding to angle 0, i.e., the horizontal cube, are considered here. In [109] the problem is solved for $10^3 \leq Ra \leq 10^6$ with a finite volume method called DUGKS and the attained results are compared with those of [46]. In [108] the problem is solved for $10^4 \leq Ra \leq 10^6$ with a fourth-order finite difference method for space discretization and a third-order backward finite difference scheme for time discretization. The authors in [108] also compare their results to the ones presented by [46].

The comparison between the results reported by the different authors needed some manipulation due either to different reference systems or to different definition of the boundary conditions. In some cases, however, the agreement between different literature results was lower then expected. In order to be able to conclude which of the available data were more trustworthy, and to offer the reader further reliable reference results, additional solutions were calculated with the commercial Finite Volume solver ANSYS Fluent.

The second verification test consists in a natural convection problem in a spherical shell enclosed between two concentric spheres with radii ratio equal to 5. The temperature is fixed on both the spheres, the inner being the hotter. Even though the problem is axisymmetric, all meshless computations are performed on the whole 3D domain, in order to provide further insights into the accuracy of the RBF-FD approach for the solution of natural convection problems in 3D. The axial symmetry of the problem is only exploited in the generation of reference solutions, obtained with a commercial solver: a simple axisymmetric 2D model was employed and solved with very high accuracy.

The whole RBF-FD code employed to obtain the presented results is developed using Julia programming language [6] and can be executed in parallel on multiple cores, although only the discretization phase has been parallelized in the present implementation. The adoption of Julia allowed extensive code reuse and excellent computational performances already at the development stage.

The chapter is laid out as follows. In Section 6.2 the governing equations are presented and the practical solution procedure is discussed in Section 6.3. In Section 6.4 the adopted stabilization technique, with reference to those discussed in the previous chapter, is presented. In Section 6.5 the differentially heated cubic cavity benchmark is discussed. In Section 6.6 the spherical shell benchmark is discussed.

6.2 Governing Equations

Both the thermo-fluid problems of interest are described by the following coupled conservation equations of mass, momentum and energy with Boussinesq

approximation:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{6.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + Pr\nabla^2 \boldsymbol{u} + Ra\, Pr\, T\hat{\boldsymbol{z}}, \tag{6.2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \nabla^2 T, \tag{6.3}$$

where $\hat{\boldsymbol{z}}$ is the unit vector along the vertical direction z. In the above equations, length l, velocity $\boldsymbol{u}$, time t, pressure p and temperature T are made dimensionless by taking respectively L, $u_0 = \alpha/L$, L/u_0, ρu_0^2 and ΔT as reference quantities, where L is a reference length, α is the thermal diffusivity and ρ is the reference density. This formulation of the governing equations is the same as in [104]. The following dimensionless numbers are defined:

- $Pr = \nu/\alpha$ is the Prandtl Number, which is chosen to be $Pr = 0.71$ for all configurations,
- $Ra = g\beta L^3 \Delta T/\nu\alpha$ is the Rayleigh number, which varies in the interval $10^2 \leq Ra \leq 10^5$,

where ν is the kinematic viscosity, g is the gravitational acceleration and β is the thermal expansion coefficient. Such values lead to steady state solutions for both problems.

6.3 Solution procedure

At each time step, the computation of velocity, pressure and temperature through Equations (6.1)-(6.3) is decoupled using a projection scheme [16] with a three-level Gear scheme for the time discretization. Since steady-state solutions are sought, no subiterations within each time step are employed. First, the following linearized momentum equation is solved for the tentative velocity $\boldsymbol{u}^*$:

$$\frac{3\boldsymbol{u}^* - 4\boldsymbol{u}^l + \boldsymbol{u}^{l-1}}{2\Delta t} + (\boldsymbol{u}^l \cdot \nabla)\boldsymbol{u}^* = -\nabla p^l + \nabla^2 \boldsymbol{u}^* + Ra\, Pr\, T^l \hat{\boldsymbol{z}} \tag{6.4}$$

where l is the time level and $\Delta t = 0.2$ is the chosen time step size.

The tentative velocity $\boldsymbol{u}^*$ is then forced to satisfy the continuity Eq. (6.1) by means of an irrotational correction $\boldsymbol{u}^{l+1} = \boldsymbol{u}^* - \nabla\Theta$, leading to the following Poisson Eq. (6.5) in the auxiliary variable Θ:

$$\nabla^2 \Theta = \nabla \cdot \boldsymbol{u}^* \tag{6.5}$$

with zero normal correction $\nabla\Theta \cdot \boldsymbol{n} = 0$ at the boundary, i.e. Neumann BC. Such BCs are appropriate since both geometries are cavities and therefore have no inlets or outlets. When no Dirichlet BC are given, equation (6.5) is undetermined up to a constant, in order to guarantee the uniqueness of the solution field Θ,

the problem can be augmented with a Lagrange multiplier λ which constraints the result to have null integral mean:

$$\begin{aligned} \nabla^2\Theta + \lambda &= \nabla \cdot \boldsymbol{u}^* \\ \int_\Omega \Theta &= 0 \end{aligned} \tag{6.6}$$

This corresponds to adding a constant row and a constant column to the resulting coefficient matrix, with a zero on the bottom-right corner of the main diagonal.

The pressure is then updated as $p^{l+1} = p^l + \Theta/\Delta t$ and the temperature is computed from the energy equation:

$$\frac{3T^{l+1} - 4T^l + T^{l-1}}{2\Delta t} + \boldsymbol{u}^{l+1} \cdot \nabla T^{l+1} = \nabla^2 T^{l+1} \tag{6.7}$$

We remark that this is the most commonly adopted approach for pressure-velocity deocupling in incompressible flows, however, alternative formulations aimed at impriving stability have been proposed and succesfully tested in recent years for the RBF-FD method [9].

6.3.1 Collocation

Each partial derivative in the considered system of PDEs, Eqs. (6.4)-(6.7), is approximated at each internal node through the RBF-FD discretization method with stationary interpolation. By composing the linear terms associated to each differential operator, the following sparse linear systems are obtained:

$$\mathbf{A}_u \boldsymbol{u}^*_{i,I} = \boldsymbol{q}_{u_i}, \quad i = 1,2,3 \tag{6.8}$$

$$\mathbf{A}_\Theta \boldsymbol{\Theta}_I = \boldsymbol{q}_\Theta, \tag{6.9}$$

$$\mathbf{A}_T \boldsymbol{T}^{l+1}_I = \boldsymbol{q}_T, \tag{6.10}$$

which represent the discrete versions of Eqs. (6.4)-(6.7): Eq. (6.8) is the discrete momentum equation for each of the three cartesian components of the tentative velocity $\boldsymbol{u}^* = (u^*_1, u^*_2, u^*_3)$, Eq. (6.9) is the discrete Poisson equation in the auxiliary variable Θ and Eq. (6.10) is the discrete energy equation for the temperature field T^{l+1}. Each field is therefore computed and stored at the internal nodes only, including the vectors $\boldsymbol{u}^{l+1}_{i,I}$, $i = 1,2,3$, and p^{l+1}_I of the cartesian components of the velocity and pressure, respectively, at the end of each time step.

6.3.2 Final solution

The resulting sparse linear systems, Eqs. (6.8)-(6.10), are solved using an iterative method. Such linear systems are preconditioned with an Incomplete LU factorization (ILU) (package `IncompleteLU` in `Julia`) with absolute dropping tolerance $\tau = 100$ for the Poisson equation and $\tau = 0.5$ for the momentum and energy equations. In the case of the Poisson Eq. (6.9), ILU factorization is

Table 6.1: CPU times per time step and memory usage, normalized to $N_I = 100,000$ nodes.

		Memory [MB]		
P	$t\,[s]$	C_I	ILU($\mathbf{A}_u$)	ILU($\mathbf{A}_\Theta$)
2	1.7	27	52	47
3	2.0	46	56	52
4	2.9	80	85	73

performed only once at the beginning of the simulation since the related matrix $\mathbf{A}_\Theta$ is constant, in contrast to the matrices $\mathbf{A}_u$ and $\mathbf{A}_T$ which change over time due to the implicit formulation of the advective term in the momentum and energy equations, respectively. Furthermore, because of the Neumann BC, a Lagrange multiplier is added to the Poisson equation in order to obtain a solvable linear system. The Biconjugate Gradient Stabilized Method (BiCGSTAB) [95] (package `IterativeSolvers` in `Julia`) is employed as iterative solver with a relative tolerance of 10^{-8}. In each case the typical number of BiCGSTAB iterations is 45 for the Poisson equation and 3 for both the momentum and energy equations.

The typical CPU time t per time step and the typical memory usage, both normalized to $N_I = 100,000$ nodes, are reported in Table 6.1. The memory usage refers to the memory required to store the following sparse matrices in double precision (`Float64` in Julia): the internal matrix C_I for a generic operator (cfr. equation (4.15) page 58), the ILU factorization of $\mathbf{A}_u$ and the ILU factorization of $\mathbf{A}_\Theta$. In the current implementation, neither the iterative solver (BiCGSTAB) nor the ILU factorization is parallelized.

The steady-state is considered to be reached when the following conditions are both met:

$$\max_{i=1,2,3} \left(\|\boldsymbol{u}_{i,I}^{l+1} - \boldsymbol{u}_{i,I}^{l}\|_1 \right) < 10^{-8} \Delta t \sqrt{RaPr} \tag{6.11}$$

$$\|\boldsymbol{T}_I^{l+1} - \boldsymbol{T}_I^{l}\|_1 < 10^{-8} \Delta t \tag{6.12}$$

6.4 Stabilization Technique

A stabilization technique is required in order to ensure the accurate resolution of the partial differential equations described in sections 6.2 in presence of Neumann BC, and especially the Poisson Equation (6.5) of section 6.3.

For this purpose the technique of projected nodes has been adopted, cfr. Chapter 5 Section 5.4.2: each boundary node has a corresponding internal node at fixed distance along the direction of the corresponding boundary normal.

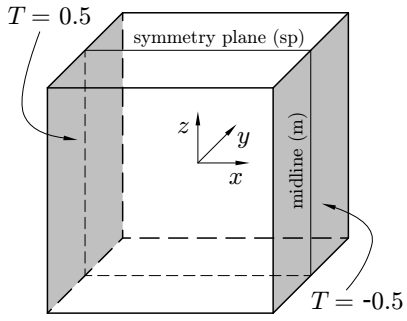


Figure 6.1: differentially heated cubic cavity domain.

Table 6.2: node distributions and Fluent grids

RBF-FD (3D)		Fluent (3D)	
ID	Nodes	ID	$N_x \times N_y \times N_z$
$N1$	161217	$G1$	$80 \times 40 \times 80$
$N2$	249560	$G2$	$96 \times 48 \times 96$
$N3$	386421	$G3$	$114 \times 57 \times 114$
$N4$	593836	$G4$	$134 \times 67 \times 134$
$N5$	919146	$G5$	$160 \times 80 \times 160$
		$G6$	$190 \times 95 \times 190$

In addition, the following control on the inclusion of boundary nodes within each stencil was implemented as an approximation of the selection based on optimal normals (cfr. 5.4.1). A boundary node $\boldsymbol{x}_j$ associated to Neumann or Robin BC is added to the stencil $\mathcal{X}_i$ only if:

$$\left| \frac{\boldsymbol{x}_j - \bar{\boldsymbol{x}}}{\|\boldsymbol{x}_j - \bar{\boldsymbol{x}}\|} \cdot \boldsymbol{n} \right| \geq d_{min} \tag{6.13}$$

where $\bar{\boldsymbol{x}}$ is the mean coordinate of the internal nodes of the stencil $\mathcal{X}_i$, $\boldsymbol{n}$ is the normal at the boundary node $\boldsymbol{x}_j$ and $d_{min} = 0.65$.

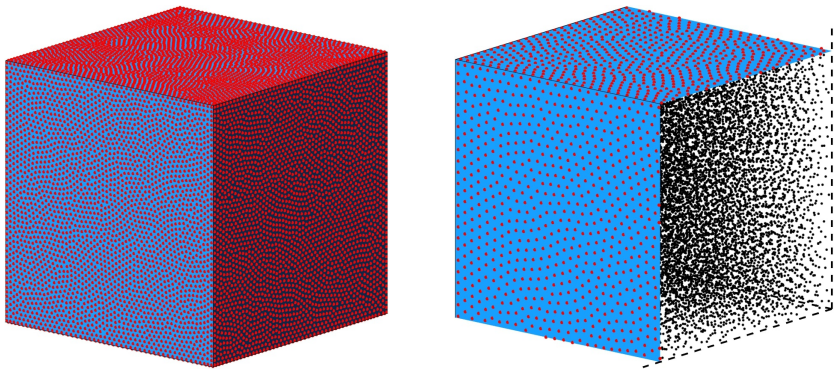
6.5 Differentially heated cubic cavity

6.5.1 Domain, BCs and node distributions

The domain is a cubic cavity with side length L, chosen as the reference length. Therefore the actual computational domain is a cube with unitary side length. A schematic representation of the domain is shown in Figure 6.1, where the Cartesian coordinates system, centered at the center of the cube, and the positions of the hot and cold walls are also shown. The difference between the temperatures T_H and T_C of the hot and cold walls, respectively, is chosen as the reference temperature scale $\Delta T = T_H - T_C$, while their mean value $(T_H + T_C)/2$ is taken as the reference temperature.

The following BCs have been considered:

- velocity: no-slip walls (Dirichlet);
- fixed temperature at the hot and cold vertical walls, respectively: $T = 0.5$ at $x = -0.5$ and $T = -0.5$ at $x = 0.5$ (Dirichlet); adiabatic $\partial T/\partial \boldsymbol{n} = 0$ at the other walls (Neumann).



(a) (b)

Figure 6.2: example of a meshless node distribution for the differentially heated cubic cavity: boundary distribution (a), enlarged view of a corner (b). Node distribution $N1$ (144846 internal nodes, 16371 boundary nodes).

Five node distributions $N1, \ldots, N5$ have been generated with a number of nodes N ranging from about 150,000 nodes to about 900,000 nodes, as reported in Table 6.2. In order to simplify the comparisons for each of the simulation parameters, uniform node distributions have been employed, i.e., obtained with constant spacing function $s(\boldsymbol{x})$. In Figure 6.2 it is possible to qualitatively see how nodes are generated: in Figure 6.2a the overall geometry is visible along with boundary nodes, shown in red. Figure 6.2b shows an enlarged view of the upper-left corner of the cubic cavity where internal nodes, coloured in black, are distinguishable.

6.5.2 Fluent reference solution

In order to obtain very accurate and reliable reference results, uniform Cartesian grids are employed. Because of the symmetry of the problem with respect to the symmetry plane, see Figure 6.1, the chosen computational domain is half of the cubic cavity. Six grids $G1, \ldots, G6$ are employed, for each one the number of cells along the three dimensions is reported in Table 6.2. In order to minimize numerical diffusion, central difference schemes are used for both the momentum and energy equations. Computations are performed in double precision and the equations are solved to machine precision, i.e., calculation is stopped when relative residuals reach the machine lower bound.

6.5.3 Verification of the RBF-FD discretization

In order to assess the correctness and the accuracy of the current implementation of the RBF-FD method, the discretization error for the differential operators ∇ and ∇^2, required by the governing PDEs, is computed for the five node distributions $N1, \ldots, N5$ and for polynomial degrees $P = 2, 3, 4$. The employed

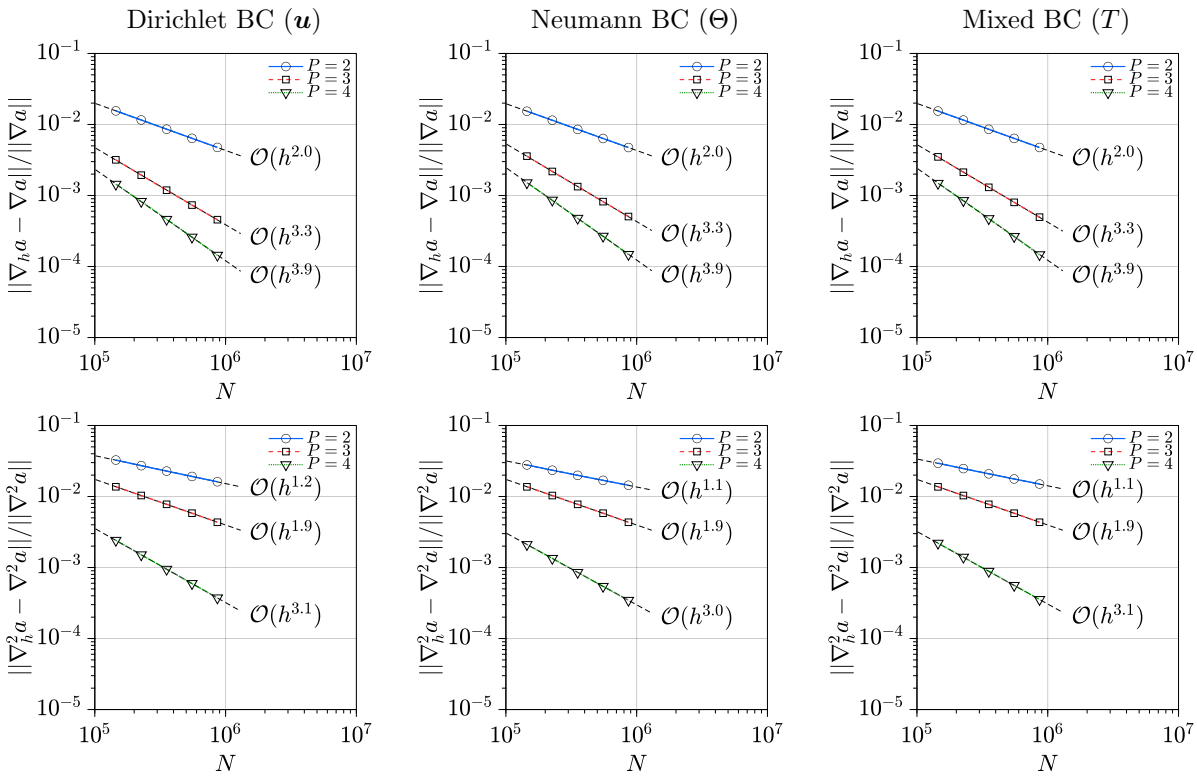


Figure 6.3: discretization error on the cubic cavity geometry with different boundary conditions, analytic function defined by $a(x, y, z) = \sin(\omega x)\sin(\omega y)\sin(\omega z)$.

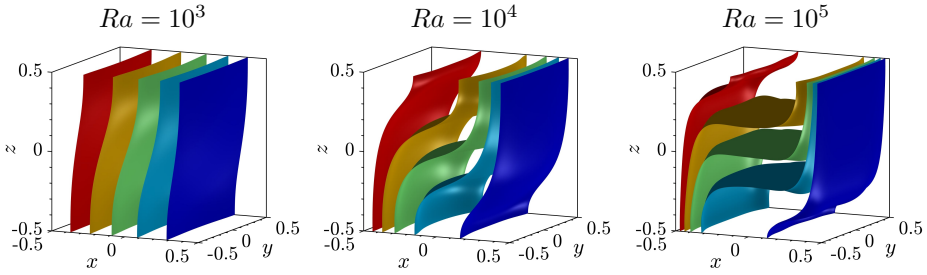


Figure 6.4: isothermal surfaces $T = 0, \pm 1/6, \pm 2/6$ for different values of the Rayleigh (Ra) number.

analytical solution is $a(x, y, z) = \sin(\omega x)\sin(\omega y)\sin(\omega z)$ with $\omega = 6\pi$. Three types of BCs are employed with reference to the same type of BC used for the computation of the three fields $\boldsymbol{u}$, Θ and T on this geometry. The results are shown in Figure 6.3, where the notation ∇_h and ∇_h^2 indicate the approximate operators calculated using the RBF-FD method described above, while ∇ and ∇^2 indicate instead the exact ones. The employed norm $\|f\|$ is the 2-norm of the vector of values $f(\boldsymbol{x}_i)$ at the internal nodes if f is a scalar, i.e., in the case of the ∇^2 operator, otherwise it is intended as the the 2-norm of the vector of values $\|f(\boldsymbol{x}_i)\|_2$ at the internal nodes if f is a vector, i.e., in the case of the ∇ operator.

From Figure 6.3 it is possible to see how, for the ∇ operator, the convergence rate q is approximately equal to the degree of the employed polynomial, $q \approx P$. For the ∇^2 operator, the rule $q \approx P - 1$ holds instead. Furthermore, the relative errors varies very little when Neumann or Mixed BCs are enforced, thus confirming the effectiveness of the projected nodes strategy in combination with the boundary nodes selection based on the dot product described in chapter 5. The reader is encouraged to compare these results with the ones reported at the end of chapter 4, where no stabilization technique was employed.

6.5.4 Results

The differentially heated cubic cavity benchmark is solved for three different values of the Rayleigh number: $Ra = 10^3$, 10^4 and 10^5. In Figure 6.4 the isothermal surfaces are shown for five values of the temperature: $T = 0$, $T = \pm 1/6$ and $T = \pm 2/6$. This figure allows to qualitatively identify the flow regimes and the areas of the domain which are interested the most by the appearance of sharp variations in the temperature field, i.e., the boundary layers, especially for the larger Ra numbers.

The RBF-FD simulations are computed for each combination of the Ra number, node distributions $N1, \ldots, N5$ and polynomial degrees $P = 2, 3, 4$, giving a total of 45 simulations. In order to assess the level of independence of the computed steady-state solutions upon the meshless discretizations, some local and global meaningful quantities obtained from each simulation are reported

in Tables 6.3, 6.4 and 6.5 for $Ra = 10^3$, $Ra = 10^4$ and $Ra = 10^5$, respectively. The reported values are the overall Nusselt number Nu, the mean Nusselt number Nu_m along the midline m, i.e, the line $(0.5, 0, z)$ (see Figure 6.1), the maximum values of the Cartesian components of the velocity $(u, v, w)_{max}$ and the corresponding locations. The reported values of Nu and Nu_m for the RBF-FD simulations are the mean of the two corresponding values computed at the hot and cold walls. Fluent results for grids $G1, \ldots, G6$ are also reported in the same table for comparison.

For the lines marked by the symbol ∞ the values are obtained by using the extrapolation defined by the following equation:

$$a_{\mathrm{ID}} \approx a_\infty + \xi_a M_{\mathrm{ID}}^{-\frac{q}{d}} \tag{6.14}$$

where a_{ID} is a generic quantity a computed with a node distribution/mesh identified by ID, M_{ID} is the corresponding number of nodes/cells, q is the convergence rate and d is the dimension ($d = 3$, except for the spherical shell problem presented in Section 6.6 solved in Fluent, where $d = 2$; $q = P$ is used for the RBF-FD method, while $q = 2$ is used for the Fluent results). Values a_∞ and ξ_a are fitted using least squares and the extrapolated value is therefore a_∞.

The comparisons reveal an excellent agreement with Fluent results, with a very satisfying level of convergence for each quantity in all cases. Within the considered range of the Ra number it is observed that the higher the polynomial degree P, the faster the values converge towards the reference ones, the case with $P = 4$ being particularly reliable even for very coarse node distributions. We remark that, even if very fine meshes are adopted for the Fluent solution, it is not possible to assess whether the Fluent $G6$ solutions, i.e., on the finest grid, are more accurate than the RBF-FD solution obtained for $P = 4$ on the finest node distributions.

For $Ra = 10^5$, Table 6.5, a slower convergence of the flow quantities emerges, especially for v_{max} with $P = 2$: this is due to the thin boundary layers, occurring at the isothermal walls, that have not been fully resolved properly by the employed meshless discretizations. Indeed, the node distributions employed in this work are uniform, which are not optimal when dealing with thin boundary layers. Accurate solutions at higher Ra numbers therefore require locally refined node distributions.

Tables 6.6, 6.7 and 6.8 show some comparisons with literature results. Additional quantities are presented in these tables: the maximum values of the Cartesian components (u, w) of the velocity on the symmetry plane sp, $(u, w)_{max,sp}$, and along the z and x center-lines, i.e., $u_{max,z}$ along the line $(0, 0, z)$ and $w_{max,x}$ along the line $(x, 0, 0)$ (cfr. Figure 6.1), together with the corresponding locations, defined as in [104]. The comparisons with literature results are not complete since most authors presented only partial results. The reported quantities are calculated by keeping into account the differences in the definition of the problem, in the size of the domain or in the coordinate system employed by the different authors. For example, velocity values from [46, 81, 68, 109, 108] were obtained by scaling the reported values by a factor $\sqrt{RaPr}$. This was done in order to allow

an immediate comparison with the results obtained here and those published by [104], which is assumed to be the main reference. As outlined before, the largest differences between the RBF-FD and the main reference, [104], occur for $Ra = 10^5$, Table 6.8, due to the inability of the uniform node distributions to fully resolve the thin boundary layers. Nonetheless, there is a very strong agreement between the RBF-FD and the reference results in all cases. Favorable agreement is also found with the other references, although incomplete. Among the the reported literature references, the presented RBF-FD implementation seems to be the closest both to the results provided by Fluent and to those found in [104].

Additional results can be found in Appendix B.1, where relative errors on some meaningful flow quantities are displayed if Figures B.1, B.2, B.3 and B.4.

Table 6.3: results for the differentially heated cavity, $Ra = 10^3$.

	ID	Nu	Nu_m	u_{max}	(x, y, z)	v_{max}	$(x, -y, z)$	w_{max}	$(-x, y, -z)$
RBF-FD, $P = 2$	$N1$	1.0711	1.0885	3.5390	(0.0166, 8E-4, 0.3169)	0.1729	(6E-4, 0.2515, 1E-4)	3.543	(0.3234, 5E-4, 0.0033)
	$N2$	1.0711	1.0884	3.5392	(0.0168, 4E-4, 0.3170)	0.1728	(6E-4, 0.2525, 4E-4)	3.542	(0.3236, 1E-3, 0.0031)
	$N3$	1.0711	1.0885	3.5390	(0.0167, 6E-5, 0.3170)	0.1727	(4E-4, 0.2520, 7E-5)	3.542	(0.3234, 6E-4, 0.0034)
	$N4$	1.0711	1.0885	3.5390	(0.0167, 1E-3, 0.3173)	0.1727	(2E-4, 0.2520, 9E-5)	3.542	(0.3233, 7E-4, 0.0033)
	$N5$	1.0711	1.0885	3.5390	(0.0169, 5E-4, 0.3170)	0.1727	(1E-5, 0.2524, 8E-5)	3.541	(0.3233, 1E-3, 0.0033)
	∞	1.0711	1.0885	3.5389	–	0.1726	–	3.541	–
RBF-FD, $P = 3$	$N1$	1.0710	1.0883	3.5391	(0.0167, 6E-4, 0.3170)	0.1725	(9E-5, 0.2521, 3E-5)	3.541	(0.3231, 1E-3, 0.0033)
	$N2$	1.0710	1.0884	3.5395	(0.0167, 2E-4, 0.3170)	0.1725	(4E-5, 0.2521, 7E-5)	3.541	(0.3233, 1E-3, 0.0033)
	$N3$	1.0710	1.0884	3.5394	(0.0167, 1E-5, 0.3170)	0.1726	(8E-5, 0.2521, 7E-5)	3.541	(0.3234, 6E-6, 0.0033)
	$N4$	1.0710	1.0884	3.5393	(0.0167, 2E-4, 0.3170)	0.1726	(4E-5, 0.2521, 5E-5)	3.541	(0.3233, 1E-4, 0.0033)
	$N5$	1.0711	1.0884	3.5393	(0.0167, 1E-5, 0.3170)	0.1726	(3E-6, 0.2521, 1E-5)	3.541	(0.3233, 6E-5, 0.0033)
	∞	1.0711	1.0884	3.5392	–	0.1726	–	3.541	–
RBF-FD, $P = 4$	$N1$	1.0711	1.0885	3.5396	(0.0167, 3E-4, 0.3170)	0.1727	(9E-5, 0.2522, 2E-5)	3.541	(0.3234, 2E-4, 0.0033)
	$N2$	1.0711	1.0885	3.5393	(0.0167, 3E-4, 0.3170)	0.1726	(5E-5, 0.2521, 2E-5)	3.541	(0.3234, 8E-4, 0.0033)
	$N3$	1.0711	1.0885	3.5392	(0.0167, 7E-5, 0.3170)	0.1726	(1E-5, 0.2521, 2E-5)	3.541	(0.3234, 5E-6, 0.0033)
	$N4$	1.0711	1.0885	3.5393	(0.0167, 2E-4, 0.3170)	0.1726	(1E-5, 0.2521, 1E-5)	3.541	(0.3234, 2E-4, 0.0033)
	$N5$	1.0711	1.0885	3.5392	(0.0167, 1E-5, 0.3170)	0.1726	(5E-6, 0.2521, 1E-5)	3.541	(0.3234, 2E-4, 0.0033)
	∞	1.0711	1.0885	3.5392	–	0.1726	–	3.541	–
Fluent	$G1$	1.0714	1.0889	3.5423	(0.0167, 0, 0.3169)	0.1731	(0, 0.2521, 0)	3.544	(0.3234, 0, 0.0033)
	$G2$	1.0713	1.0887	3.5413	(0.0168, 0, 0.3169)	0.1730	(0, 0.2521, 0)	3.543	(0.3234, 0, 0.0033)
	$G3$	1.0713	1.0887	3.5408	(0.0167, 0, 0.3170)	0.1729	(0, 0.2523, 0)	3.542	(0.3234, 0, 0.0033)
	$G4$	1.0712	1.0886	3.5403	(0.0167, 0, 0.3169)	0.1728	(0, 0.2523, 0)	3.542	(0.3233, 0, 0.0033)
	$G5$	1.0712	1.0886	3.5400	(0.0167, 0, 0.3170)	0.1727	(0, 0.2521, 0)	3.542	(0.3234, 0, 0.0033)
	$G6$	1.0712	1.0885	3.5398	(0.0167, 0, 0.3170)	0.1727	(0, 0.2522, 0)	3.541	(0.3234, 0, 0.0034)
	∞	1.0711	1.0885	3.5393	–	0.1726	–	3.541	–

Table 6.4: results for the differentially heated cavity, $Ra = 10^4$.

	ID	Nu	Nu_m	u_{max}	(x, y, z)	v_{max}	(x, y, z)	w_{max}	$(-x, y, -z)$
RBF-FD, $P = 2$	$N1$	2.0555	2.2518	16.734	(0.0198, 1E-2, 0.3254)	2.155	(0.3849, 0.2806, 0.3479)	19.000	(0.3840, 0.2277, 0.0189)
	$N2$	2.0554	2.2521	16.726	(0.0197, 1E-2, 0.3251)	2.164	(0.3833, 0.2815, 0.3459)	18.997	(0.3834, 0.2297, 0.0192)
	$N3$	2.0553	2.2518	16.724	(0.0198, 2E-3, 0.3248)	2.161	(0.3820, 0.2824, 0.3446)	18.994	(0.3835, 0.2325, 0.0207)
	$N4$	2.0551	2.2516	16.721	(0.0196, 7E-3, 0.3250)	2.160	(0.3826, 0.2822, 0.3446)	18.989	(0.3834, 0.2313, 0.0203)
	$N5$	2.0551	2.2516	16.720	(0.0195, 2E-3, 0.3249)	2.158	(0.3821, 0.2826, 0.3448)	18.986	(0.3835, 0.2268, 0.0212)
	∞	2.0549	2.2515	16.719	–	2.157	–	18.982	–
RBF-FD, $P = 3$	$N1$	2.0544	2.2510	16.730	(0.0200, 2E-2, 0.3250)	2.157	(0.3828, 0.2822, 0.3451)	19.001	(0.3834, 0.2313, 0.0208)
	$N2$	2.0546	2.2512	16.727	(0.0198, 1E-2, 0.3249)	2.160	(0.3823, 0.2826, 0.3436)	18.995	(0.3834, 0.2297, 0.0206)
	$N3$	2.0548	2.2514	16.726	(0.0197, 1E-4, 0.3249)	2.159	(0.3821, 0.2823, 0.3449)	18.992	(0.3834, 0.2311, 0.0208)
	$N4$	2.0548	2.2514	16.723	(0.0197, 9E-3, 0.3250)	2.158	(0.3823, 0.2826, 0.3449)	18.990	(0.3834, 0.2312, 0.0207)
	$N5$	2.0549	2.2514	16.721	(0.0197, 4E-3, 0.3249)	2.158	(0.3823, 0.2825, 0.3448)	18.989	(0.3834, 0.2309, 0.0207)
	∞	2.0550	2.2515	16.720	–	2.157	–	18.987	–
RBF-FD, $P = 4$	$N1$	2.0553	2.2519	16.717	(0.0197, 2E-2, 0.3250)	2.158	(0.3826, 0.2822, 0.3445)	18.986	(0.3835, 0.2294, 0.0205)
	$N2$	2.0552	2.2518	16.716	(0.0196, 1E-2, 0.3250)	2.158	(0.3822, 0.2826, 0.3451)	18.984	(0.3835, 0.2285, 0.0205)
	$N3$	2.0552	2.2517	16.717	(0.0198, 2E-4, 0.3249)	2.158	(0.3824, 0.2825, 0.3450)	18.984	(0.3834, 0.2312, 0.0208)
	$N4$	2.0552	2.2517	16.717	(0.0197, 1E-2, 0.3250)	2.157	(0.3824, 0.2825, 0.3447)	18.985	(0.3834, 0.2314, 0.0207)
	$N5$	2.0551	2.2516	16.717	(0.0197, 5E-3, 0.3250)	2.157	(0.3823, 0.2826, 0.3448)	18.984	(0.3834, 0.2304, 0.0206)
	∞	2.0551	2.2516	16.717	–	2.157	–	18.984	–
Fluent	$G1$	2.0592	2.2563	16.727	(0.0197, 0, 0.3251)	2.161	(0.3824, 0.2827, 0.3447)	18.986	(0.3834, 0.2304, 0.0203)
	$G2$	2.0580	2.2549	16.724	(0.0199, 0, 0.3249)	2.160	(0.3824, 0.2827, 0.3448)	18.985	(0.3835, 0.2309, 0.0207)
	$G3$	2.0572	2.2539	16.722	(0.0198, 0, 0.3250)	2.159	(0.3824, 0.2827, 0.3448)	18.985	(0.3834, 0.2306, 0.0205)
	$G4$	2.0566	2.2533	16.720	(0.0198, 0, 0.3249)	2.159	(0.3824, 0.2827, 0.3448)	18.985	(0.3834, 0.2307, 0.0205)
	$G5$	2.0562	2.2528	16.720	(0.0198, 0, 0.3250)	2.158	(0.3823, 0.2826, 0.3448)	18.985	(0.3834, 0.2306, 0.0205)
	$G6$	2.0559	2.2524	16.719	(0.0197, 0, 0.3249)	2.158	(0.3823, 0.2826, 0.3448)	18.984	(0.3835, 0.2306, 0.0207)
	∞	2.0551	2.2516	16.717	–	2.158	–	18.984	–

Table 6.5: results for the differentially heated cavity, $Ra = 10^5$.

	ID	Nu	Nu_m	u_{max}	$(-x, y, z)$	v_{max}	(x, y, z)	w_{max}	$(-x, y, -z)$
RBF-FD, $P = 2$	$N1$	4.340	4.611	44.52	$(0.1781, 0.216, 0.3858)$	9.905	$(0.4218, 0.3479, 0.3834)$	71.387	$(0.4303, 0.3722, 0.0036)$
	$N2$	4.339	4.616	44.25	$(0.1839, 0.241, 0.3885)$	9.770	$(0.4155, 0.3334, 0.3837)$	71.258	$(0.4303, 0.3749, 0.0083)$
	$N3$	4.337	4.613	44.22	$(0.1838, 0.216, 0.3885)$	9.716	$(0.4170, 0.3439, 0.3825)$	71.241	$(0.4302, 0.3731, 0.0017)$
	$N4$	4.337	4.612	44.13	$(0.1830, 0.221, 0.3887)$	9.781	$(0.4173, 0.3384, 0.3816)$	71.140	$(0.4295, 0.3709, 0.0052)$
	$N5$	4.336	4.611	44.07	$(0.1841, 0.224, 0.3880)$	9.724	$(0.4175, 0.3391, 0.3807)$	71.150	$(0.4302, 0.3734, 0.0046)$
	∞	4.335	4.607	–	–	–	–	–	–
RBF-FD, $P = 3$	$N1$	4.333	4.609	44.07	$(0.1831, 0.215, 0.3861)$	9.755	$(0.4173, 0.3412, 0.3863)$	71.246	$(0.4302, 0.3725, 0.0021)$
	$N2$	4.334	4.610	44.05	$(0.1836, 0.222, 0.3868)$	9.732	$(0.4166, 0.3385, 0.3834)$	71.225	$(0.4303, 0.3737, 0.0071)$
	$N3$	4.335	4.611	44.01	$(0.1840, 0.218, 0.3873)$	9.715	$(0.4169, 0.3366, 0.3800)$	71.173	$(0.4306, 0.3738, 0.0083)$
	$N4$	4.335	4.611	44.00	$(0.1838, 0.222, 0.3875)$	9.725	$(0.4177, 0.3386, 0.3800)$	71.146	$(0.4299, 0.3725, 0.0004)$
	$N5$	4.336	4.612	43.98	$(0.1840, 0.221, 0.3875)$	9.719	$(0.4177, 0.3392, 0.3800)$	71.128	$(0.4302, 0.3732, 0.0041)$
	∞	4.336	4.612	–	–	–	–	–	–
RBF-FD, $P = 4$	$N1$	4.338	4.613	43.93	$(0.1848, 0.217, 0.3871)$	9.717	$(0.4183, 0.3392, 0.3792)$	71.076	$(0.4300, 0.3729, 0.0020)$
	$N2$	4.337	4.613	43.93	$(0.1835, 0.216, 0.3870)$	9.702	$(0.4178, 0.3395, 0.3823)$	71.071	$(0.4305, 0.3737, 0.0078)$
	$N3$	4.337	4.613	43.93	$(0.1843, 0.219, 0.3874)$	9.718	$(0.4177, 0.3391, 0.3809)$	71.065	$(0.4300, 0.3724, 0.0006)$
	$N4$	4.337	4.613	43.93	$(0.1841, 0.219, 0.3877)$	9.700	$(0.4176, 0.3391, 0.3805)$	71.066	$(0.4300, 0.3725, 0.0004)$
	$N5$	4.337	4.613	43.92	$(0.1842, 0.222, 0.3874)$	9.700	$(0.4178, 0.3388, 0.3796)$	71.070	$(0.4303, 0.3734, 0.0046)$
	∞	4.337	4.613	–	–	–	–	–	–
Fluent	$G1$	4.363	4.644	44.02	$(0.1840, 0.219, 0.3875)$	9.719	$(0.4176, 0.3391, 0.3797)$	71.091	$(0.4302, 0.3732, 0.0034)$
	$G2$	4.355	4.634	44.00	$(0.1839, 0.219, 0.3873)$	9.715	$(0.4176, 0.3390, 0.3797)$	71.096	$(0.4304, 0.3732, 0.0043)$
	$G3$	4.350	4.628	43.97	$(0.1840, 0.220, 0.3873)$	9.708	$(0.4176, 0.3390, 0.3799)$	71.095	$(0.4306, 0.3732, 0.0056)$
	$G4$	4.346	4.623	43.95	$(0.1839, 0.220, 0.3874)$	9.706	$(0.4177, 0.3390, 0.3801)$	71.072	$(0.4304, 0.3735, 0.0052)$
	$G5$	4.344	4.620	43.94	$(0.1840, 0.220, 0.3874)$	9.704	$(0.4176, 0.3390, 0.3801)$	71.067	$(0.4304, 0.3734, 0.0055)$
	$G6$	4.342	4.618	43.94	$(0.1841, 0.220, 0.3873)$	9.703	$(0.4175, 0.3390, 0.3801)$	71.068	$(0.4304, 0.3735, 0.0055)$
	∞	4.337	4.612	43.92	–	9.700	–	–	–

Table 6.6: comparison with literature results for the differentially heated cavity at $Ra = 10^3$.

	Present results			Others				
	$P = 2$	$P = 3$	$P = 4$	[104]	[46]	[81]	[68]	[109]
ID/Grid	$N5$	$N5$	$N5$	81^3	32^3	91^3	$> 41^3$	50^3
Nu	1.0711	1.0711	1.0711	1.0700	1.085	1.076	1.0710	1.0700
Nu_m	1.0885	1.0884	1.0885	1.0873	1.105	1.098	1.0884	1.0880
u_{max}	3.5390	3.5393	3.5392	3.54356	-	-	3.5227	-
x	0.0169	0.0167	0.0167	0.0166	-	-	0.0000	-
y	5E-4	1E-5	1E-5	5E-12	-	-	0.0000	-
z	0.3170	0.3170	0.3170	0.3169	-	-	0.3044	-
v_{max}	0.1727	0.1726	0.1726	0.17331	-	-	0.1726	-
x	1E-5	3E-6	5E-6	1E-11	-	-	0.0000	-
$-y$	0.2524	0.2521	0.2521	0.2521	-	-	0.2500	-
z	8E-5	1E-5	1E-5	4E-11	-	-	0.0000	-
w_{max}	3.541	3.541	3.541	3.54469	-	-	3.5167	-
$-x$	0.3233	0.3233	0.3234	0.3223	-	-	0.3044	-
y	1E-3	6E-5	2E-4	4E-11	-	-	0.0000	-
$-z$	0.0033	0.0033	0.0033	0.0032	-	-	0.0000	-
$u_{max,sp}$	3.5390	3.5393	3.5392	3.54356	-	-	3.5227	-
x	0.0167	0.0167	0.0167	0.0166	-	-	0.0000	-
z	0.3170	0.3170	0.3170	0.3169	-	-	0.3044	-
$w_{max,sp}$	3.5414	3.5410	3.5409	3.54477	-	-	3.5163	-
$-x$	0.3234	0.3233	0.3234	0.3233	-	-	0.3044	-
$-z$	0.0035	0.0033	0.0033	0.0032	-	-	0.0000	-
$u_{max,z}$	3.5327	3.5329	3.5329	3.53875	3.501	3.517	-	3.5039
z	0.3167	0.3167	0.3167	0.3151	0.300	0.312	-	0.3081
$w_{max,x}$	3.5412	3.5407	3.5406	3.54185	3.517	3.544	-	3.5332
$-x$	0.3231	0.3231	0.3231	0.3147	0.3333	0.333	-	0.3301

Table 6.7: comparison with literature results for the differentially heated cavity at $Ra = 10^4$.

	Present results			Others					
	$P = 2$	$P = 3$	$P = 4$	[104]	[46]	[81]	[68]	[109]	[108]
ID/Grid	$N5$	$N5$	$N5$	81^3	62^3	61×45^2	$> 41^3$	50^3	$> 120^3$
Nu	2.0551	2.0549	2.0551	2.0542	2.100	2.085	2.0537	2.0535	2.0624
Nu_m	2.2516	2.2514	2.2516	2.2505	2.302	2.304	2.2509	2.2478	-
u_{max}	16.720	16.721	16.717	16.71986	-	-	16.5312	-	-
x	0.0195	0.0197	0.0197	0.0196	-	-	0.0000	-	-
y	2E-3	4E-3	5E-3	1E-11	-	-	0.0000	-	-
z	0.3249	0.3249	0.3250	0.3250	-	-	0.3044	-	-
v_{max}	2.158	2.158	2.157	2.15657	-	-	2.1092	-	-
x	0.3821	0.3823	0.3823	0.3823	-	-	0.3967	-	-
y	0.2826	0.2825	0.2826	0.2826	-	-	0.3044	-	-
z	0.3448	0.3448	0.3448	0.3447	-	-	0.3536	-	-
w_{max}	18.986	18.989	18.984	18.98359	-	-	18.6971	-	-
$-x$	0.3835	0.3834	0.3834	-	-	-	0.3967	-	-
y	0.2268	0.2309	0.2304	0.2308	-	-	0.1913	-	-
$-z$	0.0212	0.0207	0.0206	0.0206	-	-	0.0000	-	-
$u_{max,sp}$	16.720	16.722	16.718	16.71986	-	-	-	-	-
x	0.0196	0.0197	0.0197	0.0196	-	-	-	-	-
z	0.3250	0.3249	0.3250	0.3250	-	-	-	-	-
$w_{max,sp}$	18.688	18.688	18.684	18.68247	-	-	-	-	-
$-x$	0.3871	0.3870	0.3870	0.3870	-	-	-	-	-
$-z$	0.0226	0.0219	0.0221	0.0219	-	-	-	-	-
$u_{max,z}$	16.698	16.699	16.695	16.72128	16.962	17.36	-	16.583	16.718
z	0.3260	0.3262	0.3262	0.3244	0.3167	0.3370	-	0.3201	0.3250
$w_{max,x}$	18.635	18.634	18.630	18.61615	18.976	18.62	-	18.689	18.672
$-x$	0.3842	0.3841	0.3841	0.3802	0.3833	0.387	-	0.3873	0.3823

Table 6.8: comparison with literature results for the differentially heated cavity at $Ra = 10^5$.

	Present results			Others					
	$P=2$	$P=3$	$P=4$	[104]	[46]	[81]	[68]	[109]	[108]
ID/Grid	$N5$	$N5$	$N5$	81^3	62^3	91×65^2	$> 41^3$	50^3	$> 120^3$
Nu	4.336	4.336	4.337	4.3370	4.361	4.378	4.3329	4.3248	4.3665
Nu_m	4.611	4.612	4.613	4.6127	4.646	4.658	4.6110	4.5995	-
u_{max}	44.07	43.98	43.92	43.9037	-	-	43.6877	-	-
$-x$	0.1841	0.1840	0.1842	0.1841	-	-	0.1913	-	-
y	0.224	0.221	0.222	0.2203	-	-	0.2500	-	-
z	0.3880	0.3875	0.3874	0.3873	-	-	0.3967	-	-
v_{max}	9.724	9.719	9.700	9.6973	-	-	9.3720	-	-
x	0.4175	0.4177	0.4178	0.4175	-	-	0.4330	-	-
y	0.3391	0.3392	0.3388	0.3390	-	-	0.3536	-	-
z	0.3807	0.3800	0.3796	0.3801	-	-	0.3967	-	-
w_{max}	71.150	71.128	71.070	71.0680	-	-	70.6267	-	-
$-x$	0.4302	0.4302	0.4303	0.4304	-	-	0.4330	-	-
y	0.3734	0.3732	0.3734	0.3736	-	-	0.3536	-	-
$-z$	0.0046	0.0041	0.0046	0.00604	-	-	0.0000	-	-
$u_{max,sp}$	43.18	43.12	43.08	43.0610	-	39.70	42.7846	-	-
$-x$	0.1867	0.1864	0.1863	0.1865	-	0	0.1913	-	-
z	0.3851	0.3848	0.3848	0.3848	-	0.364	0.3967	-	-
$w_{max,sp}$	65.50	65.49	65.44	65.4362	-	63.95	65.3083	-	-
$-x$	0.4369	0.4368	0.4369	0.4368	-	0.435	0.4330	-	-
$-z$	0.0095	0.0121	0.0116	0.0100	-	0	0.0000	-	-
$u_{max,z}$	37.75	37.71	37.68	37.5612	39.116	39.70	-	37.997	37.731
z	0.3538	0.3536	0.3536	0.3535	0.3547	0.364	-	0.3507	0.3500
$w_{max,x}$	65.48	65.47	65.42	65.2113	65.842	63.95	-	65.069	65.655
$-x$	0.4361	0.4361	0.4361	0.4330	0.4353	0.435	-	0.4317	0.4323

6.6 Differentially heated spherical shell

6.6.1 Domain, BCs and node distributions

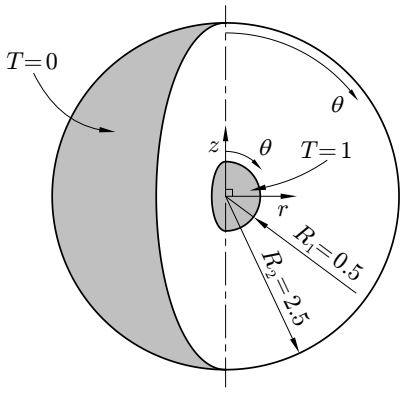


Figure 6.5: spherical shell domain.

Table 6.9: node distributions and Fluent grids.

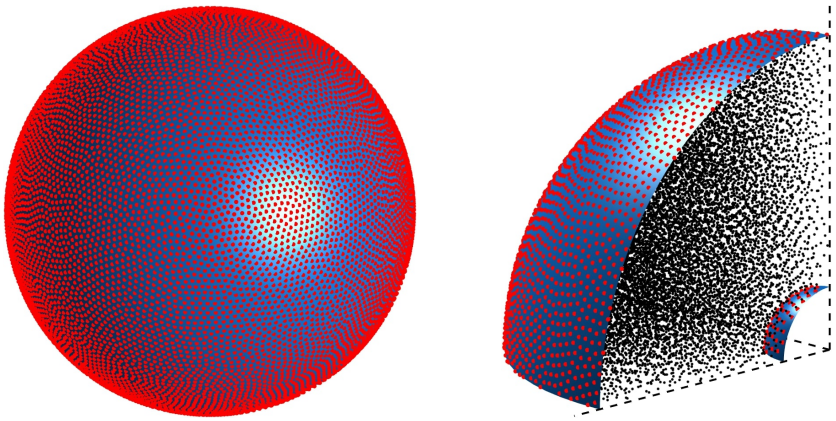
RBF-FD (3D)		Fluent (2D)	
ID	Nodes	ID	$N_r \times N_\theta$
$N1$	151260	$G1$	100×200
$N2$	233800	$G2$	142×284
$N3$	360973	$G3$	200×400
$N4$	557582	$G4$	284×568
$N5$	864261	$G5$	400×800
		$G6$	568×1136
		$G7$	800×1600
		$G8$	1136×2272

The domain is a spherical shell enclosed between two concentric spheres of radius R_1 and R_2, where the chosen ratio is $R_2/R_1 = 5$. The diameter of the inner sphere is chosen as the reference length, therefore the radii of the spheres in the actual computational domain are $R_1 = 0.5$ and $R_2 = 2.5$. A schematic representation of the domain is visible in Figure 6.5, where the cylindrical coordinates system (r, z) is centered at the center of the spheres with z as the symmetry axis. The azimuthal angle in the horizontal plane is neither shown nor employed since the problem is axisymmetric. The polar angle θ is shown instead, since it will be employed to display the (axisymmetric) normal derivatives over the spheres. The difference between the temperatures T_H and T_C of the hot and cold spheres, respectively, is chosen as the reference temperature scale $\Delta T = T_H - T_C$, while T_C is taken as the reference temperature.

The following BCs have been considered:

- velocity: no-slip walls (Dirichlet);
- fixed temperature $T = 1$ at the inner (hot) sphere and $T = 0$ at the outer (cold) sphere.

Five node distributions $N1, \ldots, N5$ have been generated with a number of nodes N ranging from about 150,000 nodes to about 900,000 nodes, as reported in Table 6.9. Uniform nodes distributions have been chosen once again in order to make any comparison simpler. In Figure 6.6 it is possible to qualitatively see how nodes are generated: in Figure 6.6a the overall geometry is visible together



(a) (b)

Figure 6.6: example of a meshless node distribution for the spherical shell: boundary distribution (a), enlarged view of an octant (b). Node distribution $N1$ (138172 internal nodes, 13088 boundary nodes).

with boundary nodes, shown in red. Figure 6.6b shows an enlarged view of an octant of the spherical shell, allowing to also see the distribution of internal nodes, shown in black.

6.6.2 Fluent reference solution

Because of the axial symmetry of the problem, an axisymmetric 2D model is employed and therefore the computational domain consists of half of a 2D annulus. In order to obtain very accurate and reliable reference results, structured grids with quadrilateral cells with aspect ratio close to 1 are employed. Eight grids $G1, \ldots, G8$ are used, whose number of cells along the radial (N_r) and polar (N_θ) directions are reported in Table 6.9. In order to minimize numerical diffusion, central difference schemes are used for both the momentum and energy equations. Computations are performed in double precision and the equations are solved to machine precision, i.e., the relative residuals reached the machine lower bound.

6.6.3 Verification of the RBF-FD discretization

The verification on this geometry has been performed in exactly the same way as presented for the cubic cavity, subsection 6.5.3. The results are quite similar to those shown in Figure 6.3 and therefore they are not reported.

6.6.4 Results

The spherical shell problem is solved for three values of the Ra number: $Ra = 100$, $Ra = 500$ and $Ra = 1000$. In Figure 6.7 the isothermal curves are shown on

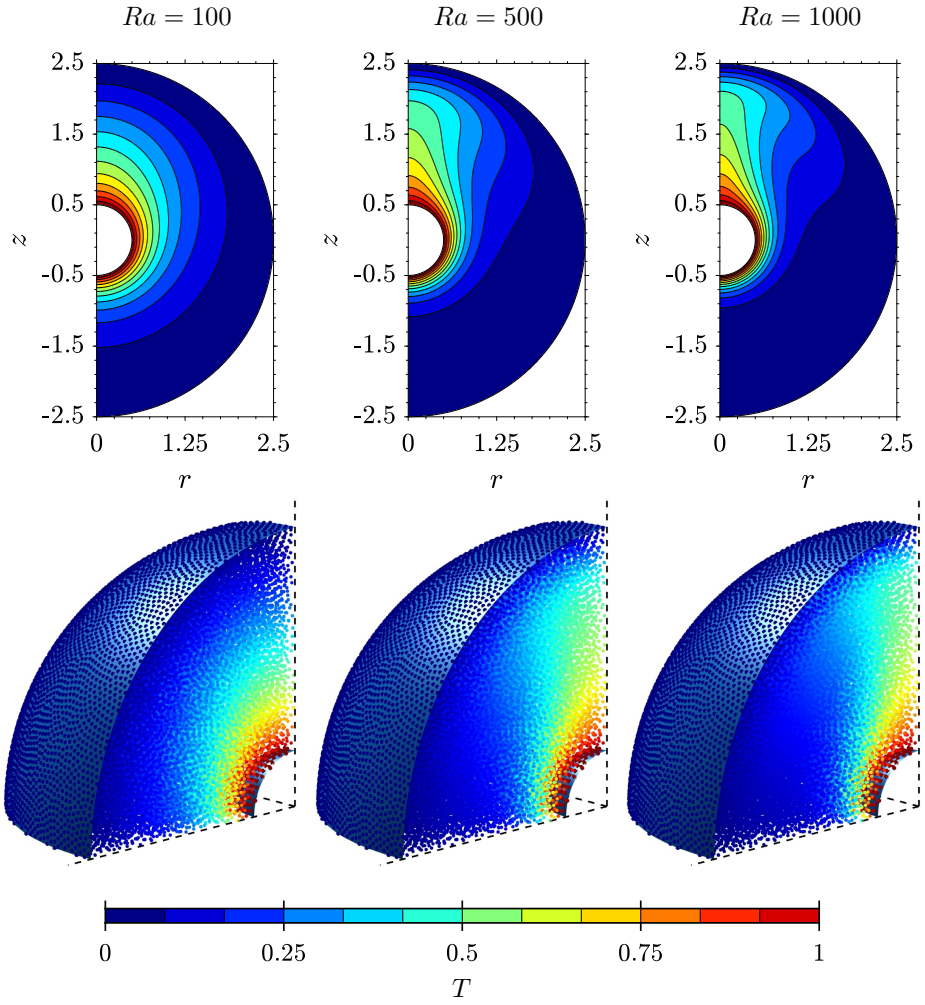


Figure 6.7: isothermal curves (top) and temperature distributions over the node distribution $N1$ (bottom) for the spherical shell problem.

the 2D axisymmetric domain, while the corresponding 3D meshless temperature fields, rendered over one upper octant of the spherical shell, are also shown in the same figure.

Even though the problem is axisymmetric, the equations are solved on the whole 3D domain, using Cartesian components, in order to provide a deeper comprehension of the accuracy of the RBF-FD approach in 3D. Similarly to the case of the cubic cavity, for each value of the Ra number, five node distributions and three different values of the polynomial degree have been used, giving a total of 45 simulations. The steady state solutions for $Ra = 100$, $Ra = 500$ and $Ra = 1000$ are described by using some meaningful quantities reported in

Table 6.10: results for the spherical shell, $Ra = 100$.

		ID	Nu	$u_{r,max}$	(r,z)	$u_{z,max}$	(r,z)	$-u_{z,min}$	(r,z)
RBF-FD (3D)	$P=2$	$N1$	2.6859	1.1301	(1.1139, 1.5991)	2.790	(0.7414, 0.200)	1.1179	(1.936, 0.569)
		$N2$	2.6880	1.1282	(1.1166, 1.5975)	2.785	(0.7498, 0.187)	1.1168	(1.931, 0.571)
		$N3$	2.6881	1.1230	(1.1178, 1.5961)	2.770	(0.7524, 0.170)	1.1137	(1.937, 0.552)
		$N4$	2.6890	1.1228	(1.1172, 1.5969)	2.764	(0.7537, 0.182)	1.1132	(1.937, 0.553)
		$N5$	2.6885	1.1200	(1.1186, 1.5960)	2.758	(0.7551, 0.178)	1.1115	(1.936, 0.558)
	$P=3$	$N1$	2.6874	1.1214	(1.1168, 1.5968)	2.775	(0.7574, 0.172)	1.1126	(1.938, 0.548)
		$N2$	2.6874	1.1210	(1.1181, 1.5955)	2.766	(0.7549, 0.177)	1.1120	(1.943, 0.562)
		$N3$	2.6878	1.1194	(1.1181, 1.5958)	2.761	(0.7558, 0.179)	1.1113	(1.937, 0.555)
		$N4$	2.6880	1.1188	(1.1184, 1.5958)	2.757	(0.7555, 0.176)	1.1110	(1.936, 0.565)
		$N5$	2.6879	1.1183	(1.1185, 1.5957)	2.756	(0.7559, 0.174)	1.1108	(1.938, 0.562)
	$P=4$	$N1$	2.6861	1.1180	(1.1185, 1.5958)	2.763	(0.7551, 0.177)	1.1103	(1.936, 0.564)
		$N2$	2.6873	1.1183	(1.1184, 1.5958)	2.761	(0.7550, 0.177)	1.1107	(1.942, 0.551)
		$N3$	2.6874	1.1180	(1.1184, 1.5959)	2.757	(0.7561, 0.174)	1.1104	(1.935, 0.568)
		$N4$	2.6877	1.1181	(1.1185, 1.5958)	2.755	(0.7560, 0.176)	1.1104	(1.938, 0.565)
		$N5$	2.6879	1.1180	(1.1185, 1.5959)	2.755	(0.7560, 0.175)	1.1103	(1.939, 0.559)
Fluent (2D)		$G1$	2.688101	1.11883	(1.1185, 1.5963)	2.75505	(0.75604, 0.1766)	1.11052	(1.9397, 0.5577)
		$G2$	2.688143	1.11859	(1.1191, 1.5958)	2.75448	(0.75602, 0.1762)	1.11048	(1.9396, 0.5579)
		$G3$	2.688163	1.11841	(1.1186, 1.5960)	2.75416	(0.75601, 0.1764)	1.11044	(1.9395, 0.5582)
		$G4$	2.688173	1.11832	(1.1180, 1.5954)	2.75400	(0.75603, 0.1763)	1.11044	(1.9396, 0.5580)
		$G5$	2.688178	1.11828	(1.1185, 1.5959)	2.75392	(0.75603, 0.1763)	1.11043	(1.9395, 0.5583)
		$G6$	2.688181	1.11826	(1.1185, 1.5959)	2.75388	(0.75600, 0.1763)	1.11043	(1.9395, 0.5583)
		$G7$	2.688182	1.11825	(1.1185, 1.5959)	2.75386	(0.75603, 0.1762)	1.11043	(1.9395, 0.5583)
		$G8$	2.688183	1.11825	(1.1185, 1.5959)	2.75385	(0.75602, 0.1763)	1.11043	(1.9395, 0.5581)
		∞	2.688184	1.11824	–	2.75384	–	1.11043	–

Tables 6.10, 6.11 and 6.12, respectively. These quantities are: the overall Nusselt number Nu, the maximum values of the components of the velocity in cylindrical coordinates, i.e., the radial u_r and the vertical u_z components, the minimum value of the vertical component of the velocity and the corresponding locations. Fluent results, obtained for the 2D axisymmetric case, are also shown in the same tables. The extrapolated results for the Fluent simulations are reported in the lines marked by the ∞ symbol and they are obtained by using the same

Table 6.11: results for the spherical shell, $Ra = 500$.

		ID	Nu	$u_{r,max}$	(r,z)	$u_{z,max}$	(r,z)	$-u_{z,min}$	(r,z)
RBF-FD (3D)	$P=2$	$N1$	3.5708	5.9301	$(0.9779, 1.9344)$	12.353	(2E-2, 1.4019)	4.3047	$(1.684, 1.267)$
		$N2$	3.5665	5.9168	$(0.9778, 1.9301)$	12.342	(8E-3, 1.4014)	4.2951	$(1.700, 1.255)$
		$N3$	3.5627	5.8876	$(0.9801, 1.9326)$	12.270	(4E-3, 1.4033)	4.2836	$(1.700, 1.247)$
		$N4$	3.5626	5.8894	$(0.9805, 1.9318)$	12.264	(1E-3, 1.4029)	4.2829	$(1.694, 1.252)$
		$N5$	3.5593	5.8762	$(0.9810, 1.9314)$	12.228	(2E-3, 1.4037)	4.2752	$(1.700, 1.242)$
	$P=3$	$N1$	3.5574	5.8933	$(0.9779, 1.9322)$	12.271	(4E-3, 1.4008)	4.2814	$(1.686, 1.266)$
		$N2$	3.5544	5.8838	$(0.9805, 1.9305)$	12.251	(1E-3, 1.4021)	4.2769	$(1.703, 1.242)$
		$N3$	3.5549	5.8749	$(0.9798, 1.9312)$	12.227	(9E-4, 1.4026)	4.2740	$(1.692, 1.265)$
		$N4$	3.5554	5.8709	$(0.9810, 1.9306)$	12.218	(2E-3, 1.4031)	4.2727	$(1.694, 1.256)$
		$N5$	3.5544	5.8680	$(0.9807, 1.9309)$	12.207	(9E-4, 1.4034)	4.2720	$(1.694, 1.253)$
	$P=4$	$N1$	3.5479	5.8622	$(0.9815, 1.9308)$	12.210	(1E-3, 1.4033)	4.2680	$(1.697, 1.253)$
		$N2$	3.5521	5.8655	$(0.9811, 1.9308)$	12.196	(1E-3, 1.4039)	4.2702	$(1.703, 1.242)$
		$N3$	3.5521	5.8640	$(0.9810, 1.9309)$	12.196	(3E-3, 1.4037)	4.2692	$(1.700, 1.248)$
		$N4$	3.5531	5.8653	$(0.9815, 1.9307)$	12.202	(3E-4, 1.4037)	4.2694	$(1.697, 1.252)$
		$N5$	3.5537	5.8653	$(0.9809, 1.9311)$	12.201	(7E-4, 1.4037)	4.2696	$(1.690, 1.261)$
Fluent (2D)		$G1$	3.55507	5.8656	$(0.9812, 1.9314)$	12.2148	$(0, 1.4033)$	4.26972	$(1.6946, 1.2533)$
		$G2$	3.55500	5.8649	$(0.9811, 1.9311)$	12.2107	$(0, 1.4034)$	4.27013	$(1.6961, 1.2527)$
		$G3$	3.55496	5.8656	$(0.9812, 1.9311)$	12.2086	$(0, 1.4035)$	4.26996	$(1.6958, 1.2522)$
		$G4$	3.55494	5.8657	$(0.9809, 1.9310)$	12.2076	$(0, 1.4035)$	4.26995	$(1.6959, 1.2517)$
		$G5$	3.55494	5.8658	$(0.9811, 1.9310)$	12.2071	$(0, 1.4036)$	4.26986	$(1.6955, 1.2526)$
		$G6$	3.55493	5.8659	$(0.9810, 1.9310)$	12.2068	$(0, 1.4036)$	4.26987	$(1.6954, 1.2529)$
		$G7$	3.55493	5.8659	$(0.9814, 1.9306)$	12.2067	$(0, 1.4036)$	4.26987	$(1.6957, 1.2520)$
		$G8$	3.55493	5.8660	$(0.9810, 1.9310)$	12.2066	$(0, 1.4036)$	4.26987	$(1.6955, 1.2526)$
		∞	3.55493	5.8660	–	12.2066	–	4.26987	–

technique presented in subsection 6.5.4.

Similarly to the differentially heated cavity, comparisons with Fluent results reveal an excellent agreement. Also in this case, within the considered range of the Ra number, it is observed that the higher the polynomial degree P, the faster the values converge towards the reference ones, the case with $P = 4$ being particularly accurate even for coarse node distributions.

Table 6.12: results for the spherical shell, $Ra = 1000$.

		ID	Nu	$u_{r,max}$	(r,z)	$u_{z,max}$	(r,z)	$-u_{z,min}$	(r,z)
RBF-FD (3D)	$P=2$	$N1$	4.073	10.324	(0.9155, 2.0358)	20.599	(1E-2, 1.4619)	6.8566	(1.618, 1.437)
		$N2$	4.061	10.292	(0.9267, 2.0307)	20.555	(6E-3, 1.4588)	6.8334	(1.599, 1.456)
		$N3$	4.055	10.240	(0.9235, 2.0329)	20.427	(9E-4, 1.4607)	6.8131	(1.607, 1.433)
		$N4$	4.055	10.247	(0.9277, 2.0295)	20.412	(1E-3, 1.4605)	6.8129	(1.608, 1.440)
		$N5$	4.049	10.214	(0.9258, 2.0297)	20.349	(3E-3, 1.4607)	6.7993	(1.612, 1.434)
	$P=3$	$N1$	4.045	10.239	(0.9298, 2.0264)	20.415	(4E-3, 1.4584)	6.8113	(1.600, 1.440)
		$N2$	4.039	10.233	(0.9255, 2.0289)	20.377	(8E-4, 1.4577)	6.7998	(1.605, 1.444)
		$N3$	4.041	10.212	(0.9276, 2.0277)	20.338	(4E-4, 1.4596)	6.7964	(1.608, 1.433)
		$N4$	4.042	10.203	(0.9288, 2.0273)	20.325	(2E-3, 1.4596)	6.7943	(1.613, 1.431)
		$N5$	4.040	10.197	(0.9291, 2.0269)	20.306	(9E-4, 1.4601)	6.7926	(1.608, 1.439)
	$P=4$	$N1$	4.029	10.185	(0.9278, 2.0282)	20.302	(1E-3, 1.4606)	6.7829	(1.606, 1.441)
		$N2$	4.036	10.186	(0.9296, 2.0269)	20.283	(1E-3, 1.4604)	6.7888	(1.621, 1.420)
		$N3$	4.037	10.187	(0.9288, 2.0276)	20.285	(2E-3, 1.4604)	6.7874	(1.618, 1.426)
		$N4$	4.038	10.190	(0.9299, 2.0270)	20.295	(1E-4, 1.4604)	6.7869	(1.615, 1.432)
		$N5$	4.039	10.191	(0.9296, 2.0271)	20.295	(8E-4, 1.4604)	6.7881	(1.613, 1.434)
Fluent (2D)		$G1$	4.04186	10.1832	(0.9331, 2.0212)	20.3223	(0, 1.4603)	6.7890	(1.6026, 1.4360)
		$G2$	4.04167	10.1886	(0.9276, 2.0283)	20.3137	(0, 1.4604)	6.7895	(1.6112, 1.4335)
		$G3$	4.04158	10.1899	(0.9288, 2.0276)	20.3095	(0, 1.4603)	6.7882	(1.6116, 1.4329)
		$G4$	4.04154	10.1913	(0.9285, 2.0276)	20.3074	(0, 1.4603)	6.7886	(1.6119, 1.4322)
		$G5$	4.04151	10.1916	(0.9290, 2.0275)	20.3063	(0, 1.4603)	6.7883	(1.6122, 1.4314)
		$G6$	4.04150	10.1919	(0.9289, 2.0275)	20.3058	(0, 1.4603)	6.7884	(1.6117, 1.4324)
		$G7$	4.04150	10.1920	(0.9290, 2.0275)	20.3056	(0, 1.4603)	6.7883	(1.6116, 1.4324)
		$G8$	4.04149	10.1921	(0.9289, 2.0275)	20.3054	(0, 1.4603)	6.7883	(1.6115, 1.4326)
		∞	4.04149	10.1921	–	20.3053	–	6.7883	–

Again, for the highest Ra value, i.e., $Ra = 1000$, Table 6.12, a slower convergence of the flow quantities emerges, although less evident than the cubic cavity case. This is once again due to the thin boundary layer, occurring at the inner sphere, that has not been fully resolved properly by the employed meshless discretizations, which are based on uniform node distributions for ease of comparison. Locally refined node distributions with smaller spacing around

the inner sphere should be employed for accurate RBF-FD simulations at higher Ra numbers.

Additional results can be found in Appendix B.2, where Figures B.5, B.6, B.7, B.8, B.9 and B.10 display the relative errors on different meaningful flow quantities with respect to a reference solution, for each combination of P and Ra.

6.7 Conclusions

In this chapter some verification tests have been performed in order to assess the accuracy and reliability of the RBF-FD method stabilized against ill-conditioning issues due to Neumann BC. Attained results are compared against reference solutions, which in turn are either gathered from the literature or obtained using a commercial CFD solver.

In order to provide the reader with a clear and quantitative picture of the performances of the numerical method under investigation, the comparison is done in terms of meaningful flow quantities, e.g., Nusselt number, maximum of velocity components and their corresponding locations, and also in terms of relative error of flow variables along chosen locations with respect to the reference ones.

Excellent agreement with reference solutions is consistently achieved for each of the presented cases, thus proving the reliability and accuracy of the RBF-FD method for the considered type of problems. The presented results are also meant to allow further comparisons with other numerical methods and provide support for further developments of similar models.

Chapter 7

RBF-HFD method

7.1 Generalized Hermite Interpolation

The RBF-FD solution method was developed from the standard RBF interpolation theory, where a field u was known at a set of points $\boldsymbol{x}_i$. The availability of data $\boldsymbol{u} = \{u(\boldsymbol{x}_1), \dots, u(\boldsymbol{x}_N)\}$ was given for granted and from the very beginning a basis of Radial Functions $\{\Phi(\cdot, \boldsymbol{x}_1), \dots, \Phi(\cdot, \boldsymbol{x}_N)\}$ was adopted in order to define the standard RBF interpolant:

$$u^h(\cdot) = \sum_{j=1}^{N} \alpha_j \Phi(\cdot, \boldsymbol{x}_j) \tag{7.1}$$

which satisfies collocation conditions for interpolation $u^h(\boldsymbol{x}_i) = u(\boldsymbol{x}_i)$ for all $i = 1, \dots, N$. In (7.1) $(\cdot)$ is the placeholder for the argument of the function: $u^h(\cdot)$ means that u^h takes one argument which remains undefined, i.e. $u^h(\cdot)$ is not evaluated. The notation $u^h(\boldsymbol{x}) = \delta_{\boldsymbol{x}} u^h(\cdot)$ indicates instead that u^h is being evaluated at $\boldsymbol{x}$ by means of the application of a point-evaluation functional and thus takes the corresponding real value.

When the theory of interpolation is applied to the solution of PDEs, point-evaluation functionals are not the only operators acting on u^h. In presence of Neumann BC, for instance, the evaluation of the normal derivative at a certain point $\boldsymbol{x}$ can be interpreted as the successive application of two operators:

$$\frac{\partial}{\partial \boldsymbol{n}} u^h(\boldsymbol{x}) = \left(\delta_{\boldsymbol{x}} \circ \frac{\partial}{\partial \boldsymbol{n}} \right) u^h(\cdot) \tag{7.2}$$

In the standard interpolation scheme the basis made of radial functions is obtained by applying a number of point evaluation functionals to the second argument of the kernel function:

$$\Phi(\cdot, \boldsymbol{x}_j) = \delta_{2,\boldsymbol{x}_j} \Phi(\cdot, \cdot) \tag{7.3}$$

where the subscript 2 in $\delta_{2,\boldsymbol{x}_j}$ indicates that the functional is acting on the second argument. As a further development, a more general basis could theoretically

be constructed by applying a different set of functionals, other than simple point evaluations, to the same kernel function $\Phi(\cdot,\cdot)$. And this is precisely what happens in the so-called *Generalized Hermite Interpolation*.

Suppose that, instead of the set $\{u(\boldsymbol{x}_1),\dots,u(\boldsymbol{x}_N)\}$, the function u which must be interpolated is given along with a linearly independent set of continuous linear functionals $\Lambda=\{\lambda_1,\dots,\lambda_N\}$. For instance, λ_j could denote point evaluation at $\boldsymbol{x}_j$: $\delta_{\boldsymbol{x}_j}$, or it could denote a combination of the evaluation functional with some derivative operator: $\delta_{\boldsymbol{x}_j}\circ D^\alpha$.

An interpolant u^h is now required to satisfy the Hermite interpolation conditions [27]:

$$\lambda_j u=\lambda_j u^h \quad j=1,\dots,N \tag{7.4}$$

If the standard basis of Radial Functions $\{\Phi(\cdot,\boldsymbol{x}_1),\dots,\Phi(\cdot,\boldsymbol{x}_N)\}$ was used, the enforcement of Hermite Interpolation conditions (7.4) would produce a non symmetric matrix:

$$\boldsymbol{B}_{old}=\begin{pmatrix}\lambda_1\Phi(\boldsymbol{x}_1,\boldsymbol{x}_1) & \dots & \lambda_1\Phi(\boldsymbol{x}_1,\boldsymbol{x}_N)\\ \vdots & \ddots & \vdots\\ \lambda_N\Phi(\boldsymbol{x}_N,\boldsymbol{x}_1) & \dots & \lambda_N\Phi(\boldsymbol{x}_N,\boldsymbol{x}_N)\end{pmatrix} \tag{7.5}$$

$\boldsymbol{B}_{old}$ does not satisfy in general the usual conditions for positive definiteness or non-singularity and therefore does not guarantee the well-posedness of the interpolation problem associated with (7.4). A similar case happened when dealing with Neumann BC, where functional $\lambda_j=\delta_{\boldsymbol{x}_j}\circ\partial/\partial\boldsymbol{n}_j$ for some j associated with a boundary node was present in the corresponding row of $\boldsymbol{B}_{old}$.

In order to avoid the singularity of matrix $\boldsymbol{B}_{old}$ the basis must be modified as follows: for each λ_j the usual function $\Phi(\cdot,\boldsymbol{x}_j)$ is replaced with $\lambda_{2,j}\Phi(\cdot,\cdot)$. The resulting basis $\{\lambda_{2,1}\Phi(\cdot,\cdot),\dots,\lambda_{2,N}\Phi(\cdot,\cdot)\}$ leverages the flexibility offered by the fact that each $\Phi(\cdot,\cdot)$ is a kernel which can be considered as a function of both arguments. The functionals $\lambda_{2,i}$ therefore act on $\Phi(\cdot,\cdot)$ considered as a function of the second argument and the resulting basis incorporates a dependency on the particular interpolation conditions (7.4).

The new iterpolant u^h, defined according to the Generalized Hermite Interpolation scheme, is:

$$u^h(\cdot)=\sum_{j=1}^{N}\alpha_j\lambda_{2,j}\Phi(\cdot,\cdot) \tag{7.6}$$

By enforcing once again Hermite conditions (7.4), we obtain the following interpolation matrix:

$$\boldsymbol{B_H}=\begin{pmatrix}\lambda_{1,1}\lambda_{2,1}\Phi(\cdot,\cdot) & \dots & \lambda_{1,1}\lambda_{2,N}\Phi(\cdot,\cdot)\\ \vdots & \ddots & \vdots\\ \lambda_{1,N}\lambda_{2,1}\Phi(\cdot,\cdot) & \dots & \lambda_{1,N}\lambda_{2,N}\Phi(\cdot,\cdot)\end{pmatrix} \tag{7.7}$$

Matrix $\boldsymbol{B_H}$ is symmetric if $\lambda_{1,j} = \lambda_{2,j}$ for $j = 1, \ldots, N$ and the associated matrix form for conditions (7.4) is:

$$\boldsymbol{B_H} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \lambda_1 u(\cdot) \\ \vdots \\ \lambda_N u(\cdot) \end{pmatrix} \tag{7.8}$$

The possibility of using multiquadric basis functions for Hermite interpolations was first mentioned by R. Hardy in 1975 [50, 27]. It is in 1992 with [125], however, that the research on the topic gains the much deserved attention, in this article the authors suppose that only one interpolation condition can be enforced per data point, in the form of some linear combination of function value and derivatives [27]. This limitation is surpassed with further generalizations presented in two researches published in 1994: [98] and [75] and the Generalized Hermite interpolation problem was proven to be well-posed. The adoption of Hermite formulation, also called *symmetric* is now widespread and comparison with the traditional *unsymmetric* formulation are available [59, 82, 27].

From the perspective of a practitioner, the key point is that matrix $\boldsymbol{B_H}$ is symmetric and positive definite when the associated basic function φ is strictly positive definite and operators λ_j are linearly independent. Furthermore, polynomial augmentation is possible and the associated matrix is positive definite for conditionally positive definite functions. In practice, the adoption of the Hermite scheme allows the stable solution of a much larger set of problems.

The adoption of polynomial augmentation requires that operators λ_j are considered in the orthogonality conditions, the resulting interpolant u^h is:

$$u^h(\cdot) = \sum_{j=1}^{N} \alpha_j \lambda_{2,j} \Phi(\cdot, \cdot) + \sum_{k=1}^{M} \beta_k p_k(\cdot) \tag{7.9}$$

with the conditions:

$$\sum_{j=1}^{N} \alpha_j \lambda_j p_k(\cdot) = 0, \qquad k = 1 \ldots, M \tag{7.10}$$

Equation (7.8) then becomes:

$$\underbrace{\begin{pmatrix} \boldsymbol{B_H} & \boldsymbol{P_H} \\ \boldsymbol{P_H}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Lambda u} \\ \boldsymbol{0} \end{pmatrix} \tag{7.11}$$

where $\boldsymbol{\Lambda u} = \{\lambda_1 u, \ldots, \lambda_N u\}$ and block matrix $\boldsymbol{P_H}$ associated with the polyno-

mial augmentation is:

$$\boldsymbol{P_H} = \begin{pmatrix} \lambda_1 p_1(\cdot) & \dots & \lambda_1 p_M(\cdot) \\ \vdots & \ddots & \vdots \\ \lambda_N p_1(\cdot) & \dots & \lambda_N p_M(\cdot) \end{pmatrix} \tag{7.12}$$

We remark that when $\Lambda = \{\delta_{\boldsymbol{x}_1}, \dots, \delta_{\boldsymbol{x}_N}\}$, the Generalized Hermite Interpolation reduces to the traditional RBF Interpolation.

7.2 Global Symmetric formulation

The Hermite Interpolation scheme can be adopted in place of the standard Kansa's approach [27] and its main advantage lies in the fact that it yields symmetric non-singular matrices.

Suppose the method is applied to the solution of problem (4.3) of page 55.

In order to enforce appropriate boundary conditions the set of collocation nodes $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ is split as usual into that of inner nodes $\mathcal{X}_I = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_I}\}$ and that of boundary nodes $\mathcal{X}_B = \{\boldsymbol{x}_{N_J}, \dots, \boldsymbol{x}_N\}$, with $N_J = N_I + 1$.

Collocation conditions are:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \mathcal{L}u(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \\ \mathcal{B}u^h(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \qquad \text{if } \boldsymbol{x}_i \in \mathcal{X}_B \end{aligned} \tag{7.13}$$

where notation $\mathcal{L}u^h(\boldsymbol{x}_i)$ is used for simplicity instead of the more rigorous $(\delta_{\boldsymbol{x}_i} \circ \mathcal{L})u^h(\cdot)$.

While in the non-symmetric framework u^h is independent of the linear operators $\mathcal{L}$ and $\mathcal{B}$, these are now incorporated in the augmented RBF basis:

$$u^h(\cdot) = \sum_{j=1}^{N_I} \alpha_j \mathcal{L}_2 \Phi(\cdot, \boldsymbol{x}_j) + \sum_{j=N_J}^{N} \alpha_j \mathcal{B}_2 \Phi(\cdot, \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\cdot) \tag{7.14}$$

along with orthogonality conditions required by the polynomial augmentation:

$$\sum_{j=1}^{N_I} \alpha_j \mathcal{L} p_k(\boldsymbol{x}_j) + \sum_{j=N_J}^{N} \alpha_j \mathcal{B} p_k(\boldsymbol{x}_j) = 0, \qquad k = 1 \dots, M \tag{7.15}$$

The resulting global linear system is:

$$\begin{pmatrix} \boldsymbol{\mathcal{L}_1 \mathcal{L}_2 \Phi}_{I,I} & \boldsymbol{\mathcal{L}_1 \mathcal{B}_2 \Phi}_{I,B} & \boldsymbol{\mathcal{L} P}_I \\ \boldsymbol{\mathcal{B}_1 \mathcal{L}_2 \Phi}_{B,I} & \boldsymbol{\mathcal{B}_1 \mathcal{B}_2 \Phi}_{B,B} & \boldsymbol{\mathcal{B} P}_B \\ \boldsymbol{\mathcal{L} P}_I^T & \boldsymbol{\mathcal{B} P}_B^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha_I} \\ \boldsymbol{\alpha_B} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mathcal{L} u}_I \\ \boldsymbol{\mathcal{B} u}_B \\ \mathbf{0} \end{pmatrix} \tag{7.16}$$

where subscripts I and B indicate the sets of node coordinates which are used as first or second argument, for instance, $\boldsymbol{\mathcal{L}_1\mathcal{L}_2\Phi}_{I,I}$ is the block matrix obtained when $\mathcal{L}_1\mathcal{L}_2\Phi(\cdot,\cdot)$ is evaluated at all combinations of internal nodes. Contrary to what happens in equation (4.13) there is no need to solve a dual problem in order to estimate the discretization coefficients $\boldsymbol{c_I}$ and $\boldsymbol{c_B}$ because the solution is already uniquely defined by coefficient vectors $\boldsymbol{\alpha_I}$ and $\boldsymbol{\alpha_B}$. While this might be advantageous in a global scheme, where local stencils are not adopted, it does not constitute an advantage when a local discretization scheme is adopted in practice. Available literature resources suggest that, aside from the already mentioned advantage in making the interpolation problem well-posed, there seem to be no clear-cut difference in terms of accuracy between stable solutions provided by the two methods [59, 82, 27].

We also remark that, if additional information were available at the nodes, the Generalized Hermite Interpolation would allow the enforcement of multiple conditions at each node. For instance, $\delta_{\boldsymbol{x}_j} \circ \mathcal{L}$ and $\delta_{\boldsymbol{x}_j} \circ \mathcal{F}$, even if evaluated at the same point $\boldsymbol{x}_j$, might very well be linearly independent and therefore lead to different interpolation conditions. This in turn would translate into having multiples lines and columns of matrix $\boldsymbol{M_H}$ referred to the same node $\boldsymbol{x}_j$.

Finally, the new possibilities offered by the Hermite construction of the RBF basis are at the core of the so-called RBF-HFD method [35], which is the main topic of the followin section.

7.3 RBF-HFD Formulation

As outlined in the case of the RBF-FD scheme in chapter 4, a big improvement in computational efficiency is achieved by adopting a local formulation. The RBF-FD method allowed to achieve accurate results in presence of Dirichlet boundary conditions but lacked stability against ill-conditioning issues due to specific BC, as explained in chapters 4 and 5. The stabilization methods proposed in chapter 5 were found to be successful in the specific case of Neumann BC but at the cost of adding some additional complications to the solution method.

On the other side, the global Hermite Interpolation scheme described in the previous section is not affected by the same ill-conditioning issues but does not lend itself as well to a local formulation because the intermediate finite difference discretization coefficients $\boldsymbol{c}$ are not calculated. This reduces somehow the flexibility of the method when more complex boundary value problems are being solved iteratively and complicates the implementation of a numerical solver for engineering applications.

In the attempt of combining the strength of both approaches, the Hermite scheme is adopted at a local level within the RBF-FD framework, thus leading to a hybrid method which may be called RBF-HFD (Radial Basis Functions-Hermite Finite Difference).

Here follows a step-by-step explanation of the method, in order to make the reading simpler and avoiding excessive referencing, some statements and equations will be a repetition of what is written in chapter 4.

The problem to be solved is once again:

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega \\ \mathcal{B}u = g & \text{on } \partial\Omega \end{cases} \tag{7.17}$$

We introduce local stencils as for the RBF-FD method: a stencil of m nodes $\mathcal{X}_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ can be constructed around each internal node $\boldsymbol{x}_i$ by picking the m nearest neighbors. Suppose that within $\mathcal{X}_i$, the nodes are ordered as for the RBF-FD case: the first m_I are internal giving the set $\mathcal{X}_{i,I} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m_I}\}$ and the following m_B are boundary giving $\mathcal{X}_{i,B} = \{\boldsymbol{x}_{m_J}, \dots, \boldsymbol{x}_m\}$ with $m_J = m_{I+1}$.

7.3.1 Minimal Formulation

In the Generalized Hermite Interpolation scheme multiple linear functionals can be obtained by evaluating at the same point different linear operators. As long as the continuous linear functionals λ_i are linearly independent, additional information can be added to the interpolation without necessarily increasing the number of nodes. For instance, linear combinations of multiple derivatives of u at the surrounding nodes $\boldsymbol{x}_j \neq \boldsymbol{x}_i$ can be employed [112]. In its minimal formulation, however, the properties of the Generalized Hermite Interpolation are only used to ensure the non-singularity of the local interpolation system. In the present section the minimal formulation is presented, whereas a more generic implementation of the Hermite scheme is discussed in the next one.

Within each stencil $\mathcal{X}_i$, the approximate solution u^h is defined in order to satisfy the following Hermite Interpolation conditions at each $\boldsymbol{x}_j \in \mathcal{X}_i$:

$$\begin{aligned} u^h(\boldsymbol{x}_j) &= u(\boldsymbol{x}_j) \quad && \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,I} \\ \mathcal{B}u^h(\boldsymbol{x}_j) &= \mathcal{B}u(\boldsymbol{x}_j) \quad && \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,B} \end{aligned} \tag{7.18}$$

Therefore, according to the theory of Hermite interpolation, when u^h is evaluated at the stencil center $\boldsymbol{x}_i$ it takes the form:

$$u^h(\boldsymbol{x}_i) = \sum_{j=1}^{m_I} \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{j=m_J}^{m} \mathcal{B}_2\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{k=1}^{M} p_k(\boldsymbol{x}_i) \tag{7.19}$$

where $\boldsymbol{\alpha_I} = \{\alpha_1, \dots, \alpha_{m_I}\}$ and $\boldsymbol{\alpha_B} = \{\alpha_{m_J}, \dots, \alpha_m\}$ are restricted to satisfy:

$$\sum_{j=1}^{m_I} \alpha_j p_k(\boldsymbol{x}_j) + \sum_{j=m_J}^{m} \alpha_j \mathcal{B}p_k(\boldsymbol{x}_j) = 0, \quad k = 1, \dots, M \tag{7.20}$$

We remark that conditions 7.18 are exactly the same as the usual RBF-FD method but the local interpolant u^h is defined according to the theory of

Generalized Hermite Interpolation, the resulting local system is:

$$\underbrace{\begin{pmatrix} \boldsymbol{\Phi}_{I,I} & \boldsymbol{\mathcal{B}_2}\boldsymbol{\Phi}_{I,B} & \boldsymbol{P_I} \\ \boldsymbol{\mathcal{B}_1}\boldsymbol{\Phi}_{B,I} & \boldsymbol{\mathcal{B}_1}\boldsymbol{\mathcal{B}_2}\boldsymbol{\Phi}_{B,B} & \boldsymbol{\mathcal{B}}\boldsymbol{P_B} \\ \boldsymbol{P_I}^T & \boldsymbol{\mathcal{B}}\boldsymbol{P_B}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M_{BC}}} \begin{pmatrix} \boldsymbol{\alpha_I} \\ \boldsymbol{\alpha_B} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_I} \\ \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \tag{7.21}$$

Matrix $\boldsymbol{M_{BC}}$ is now symmetric and positive definite with an appropriate RBF $\Phi(\cdot,\cdot)$, as a consequence system (7.21) is well posed regardless of the boundary condition.

Application of linear operator $\delta_{\boldsymbol{x}_i} \circ \mathcal{L}$ to the interpolant u^h, defined in (7.19), yields:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \sum_{j=1}^{m_I} \alpha_j \mathcal{L}_1\Phi(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_{j=m_J}^{m} \alpha_j \mathcal{L}_1\mathcal{B}_2\Phi(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k \mathcal{L}p_k(\boldsymbol{x}_i) \\ &= \begin{pmatrix} \boldsymbol{\alpha_I} & \boldsymbol{\alpha_B} & \boldsymbol{\beta} \end{pmatrix} \begin{pmatrix} \mathcal{L}_1\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,I}) \\ \mathcal{L}_1\mathcal{B}_2\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,B}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \end{aligned} \tag{7.22}$$

Coefficient vectors $\boldsymbol{\alpha_I}$, $\boldsymbol{\alpha_B}$ and $\boldsymbol{\beta}$ can then be substituted from equation (7.21) inside equation (7.22) giving:

$$\mathcal{L}u^h(\boldsymbol{x}_i) = \begin{pmatrix} \boldsymbol{u_I} & \boldsymbol{g} & 0 \end{pmatrix} \boldsymbol{M_{BC}}^{-T} \begin{pmatrix} \mathcal{L}_1\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,I}) \\ \mathcal{L}_1\mathcal{B}_2\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,B}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \tag{7.23}$$

As for the RBF-FD method, it follows that $\mathcal{L}u^h(\boldsymbol{x}_i)$ can be written in a finite difference formulation as follows:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \boldsymbol{c_I}(\boldsymbol{x}_i)^T \boldsymbol{u_I} + \boldsymbol{c_B}(\boldsymbol{x}_i)^T \boldsymbol{g} \\ &= \sum_{j=1}^{m_I} c_j(\boldsymbol{x}_i) u(\boldsymbol{x}_j) + \sum_{j=m_J}^{m} c_j(\boldsymbol{x}_i) g(\boldsymbol{x}_j) \end{aligned} \tag{7.24}$$

where the coefficient vectors $\boldsymbol{c_I}(\boldsymbol{x}_i) \in \mathbb{R}^{m_I}$ and $\boldsymbol{c_B}(\boldsymbol{x}_i) \in \mathbb{R}^{m_B}$ are defined as the solution of the local problem:

$$\boldsymbol{M_{BC}}^T \begin{pmatrix} \boldsymbol{c_I}(\boldsymbol{x}_i) \\ \boldsymbol{c_B}(\boldsymbol{x}_i) \\ \boldsymbol{c_p}(\boldsymbol{x}_i) \end{pmatrix} = \begin{pmatrix} \mathcal{L}_1\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,I}) \\ \mathcal{L}_1\mathcal{B}_2\Phi(\boldsymbol{x}_i,\mathcal{X}_{i,B}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix} \tag{7.25}$$

The discretization proceeds as the local interpolant u^h is required to satisfy the collocation condition at all internal nodes $\boldsymbol{x}_i \in \mathcal{X}_I$, each one considered as the center of an associated stencil $\mathcal{X}_i$:

$$\mathcal{L}u^h(\boldsymbol{x}_i) = \mathcal{L}u(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_I \tag{7.26}$$

yielding the following sparse linear system, which has the same form as that of equation (7.27) obtained for the RBF-FD method:

$$\boldsymbol{C_I} \begin{pmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{pmatrix} = \begin{pmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_{N_I}) \end{pmatrix} - \boldsymbol{C_B} \begin{pmatrix} g(\boldsymbol{x}_{N_J}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{pmatrix} \tag{7.27}$$

where matrices $\boldsymbol{C_I}$ and $\boldsymbol{C_B}$ are formed by row vectors $\boldsymbol{c_I}(\boldsymbol{x}_i)^T$ and $\boldsymbol{c_B}(\boldsymbol{x}_i)^T$ respectively, found by solving equation (7.25) at each of the N_I stencil centers $\boldsymbol{x}_i \in \mathcal{X}_I$.

$$\begin{aligned} \boldsymbol{C_I} &= \begin{pmatrix} c_1(\boldsymbol{x}_1) & \dots & c_{N_I}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_1(\boldsymbol{x}_{N_I}) & \dots & c_{N_I}(\boldsymbol{x}_{N_I}) \end{pmatrix} \in \mathbb{R}^{N_I, N_I} \\ \boldsymbol{C_B} &= \begin{pmatrix} c_{N_J}(\boldsymbol{x}_1) & \dots & c_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_{N_J}(\boldsymbol{x}_{N_I}) & \dots & c_N(\boldsymbol{x}_{N_I}) \end{pmatrix} \in \mathbb{R}^{N_I, N_B} \end{aligned} \tag{7.28}$$

Equation (7.27) can finally be solved using an iterative program to find the unknown values $\{u^h(\boldsymbol{x}_1), \dots, u^h(\boldsymbol{x}_{N_I})\}$.

The major advantage obtained with the RBF-HFD method is the resolution of any ill-conditining due to the presence of boundary conditions associated with differential operators $\mathcal{B}$ different from the identity operator which is used for Dirichlet BC.

The additional computational cost is negligible since none of the linear systems changes in size and the construction of the new local interpolation matrix $\boldsymbol{M_{BC}}$ does not require any additional information. Furthermore, the flexibility of the Finite Difference formulation of equation (7.24) allows the combination of multiple linear operators when the final matrices $\boldsymbol{C_I}$ and $\boldsymbol{C_B}$ are assembled.

We remark that the RBF-HFD method is not immune to ill-conditioning issues related to the flattness of the RBF basis function and therefore the adoption of *stationary* interpolation (4.21) is still recommended when systems (7.25) are solved directly. This in turn implies that the order of accuracy is once again given by the degree of the polynomial augmentation, as for the RBF-FD method.

Figures 7.1 and 7.2 depict the convergence curves achieved when the Poisson Equation is solved on the domain of Figure 5.12 with all Neumann BC. In order to assess the accuracy allowed by the different approaches toward stabilization the analytic function $u(x, y) = \sin(\omega x)\sin(\omega y)$ was used as the exact solution with $\omega = 4\pi$. Discretization and solution error are calculated respectively as:

$$\begin{aligned} \text{Discretization error} &= \frac{\| \triangle^h \boldsymbol{u_I} - \triangle \boldsymbol{u_I} \|_2}{\| \triangle \boldsymbol{u_I} \|_2} \\ \text{Solution error} &= \frac{\|\boldsymbol{u_I^h} - \boldsymbol{u_I}\|_2}{\|\boldsymbol{u}\|_2} \end{aligned} \tag{7.29}$$

where $\triangle^h \boldsymbol{u_I} = \boldsymbol{C_I u_I} + \boldsymbol{C_B}(\partial \boldsymbol{u_B}/\partial \boldsymbol{n})$, $\boldsymbol{u_I}$ being the vector of u at the inner nodes $\{u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_N)\}^T$ and $(\partial \boldsymbol{u_B}/\partial \boldsymbol{n})$ the vector of boundary conditions at all boundary nodes. Results were attained using stationary interpolation with $\varepsilon_0 = 0.4$ in (4.21), the number of nodes included in each stencil is $m = 2M$, M being the length of the polynomial basis. The node placement was achieved with 200 iterations of node-repel refinement with uniform spacing function $s(\boldsymbol{x})$, the slight variation in N in the case of projected nodes is due to the implementation of a node-refinement strategy which generates additional boundary nodes in order to achieve the desired placement for arbitrarily complex 2D geometries. Even if the problem is solved also in the case of standard RBF-FD, because of Neumann BC solution errors are higher than expected and visible oscillations are present in the convergence curves. All stabilization techniques succeed in reducing errors due to ill-conditioning due to Neumann BC, with a slight advantage of the methods examined in Figure 7.2. The Minimal RBF-HFD is however the simplest approach and the most efficient having the exact same computational cost as the standard RBF-FD method.

7.4 Implicit/Compact Scheme formulation

7.4.1 Differences with Minimal RBF-HFD

In the case of the minimal formulation, given a stencil $\mathcal{X}_i$, the interpolant (7.19) was adopted in combination with interpolation conditions (7.18). This allows the stabilization of with respect to boundary conditions but only requires u^h to satisfy $\delta_{\boldsymbol{x}_j} u^h = \delta_{\boldsymbol{x}_j} u$ at nodes $\boldsymbol{x}_j \in \mathcal{X}_{i,I}$. In order to improve accuracy, the polynomial degree in (7.19) can be increased, however, for $\mathcal{X}_i$ to be unisolvent its size must increase rapidly, to the point that computational efficiency and memory requirements soon become unacceptable. To circumvent this problem, in [112] the authors propose a generalization to the compact FD formulas introduced by Collatz [18] and later developed by Lele [60]. The basic idea is to keep the stencil size fixed and to also include in the FD formula a linear combination of derivatives of u at the surrounding nodes, other implementation details on this method within the RBF-HFD framework can be found in [22]. In [112, 22] the resulting method, which involves the inclusion of other differential operators in the interpolant u^h, is simply called RBF-HFD. In accordance with this practice,

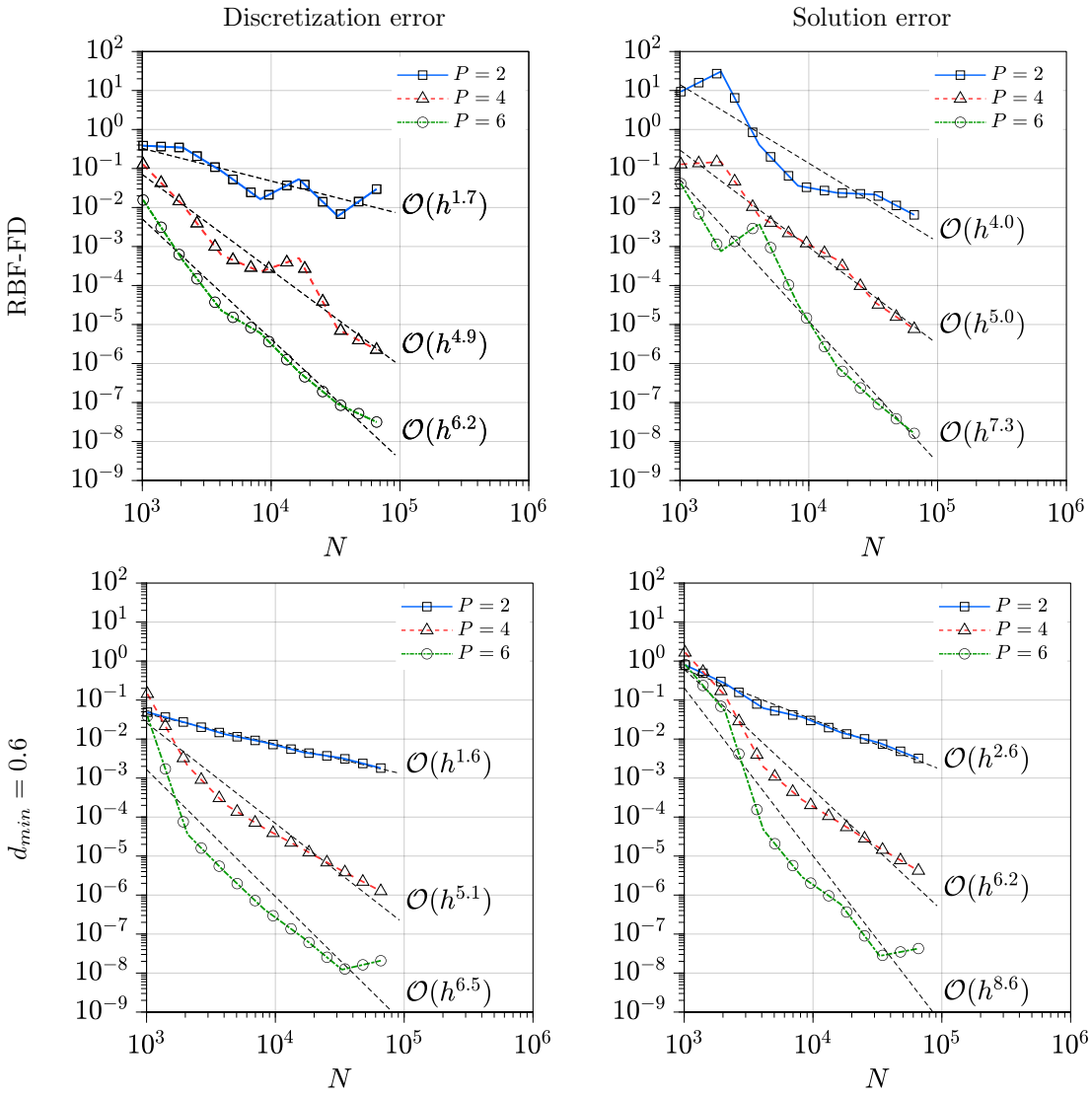


Figure 7.1: convergence curves for discretization error and solution error of the Poisson Equation on the geometry of Figure 5.12 for non stabilized RBF-FD method (top row) and for RBF-FD stabilized with Approach 1 (cfr. 5.4.1)

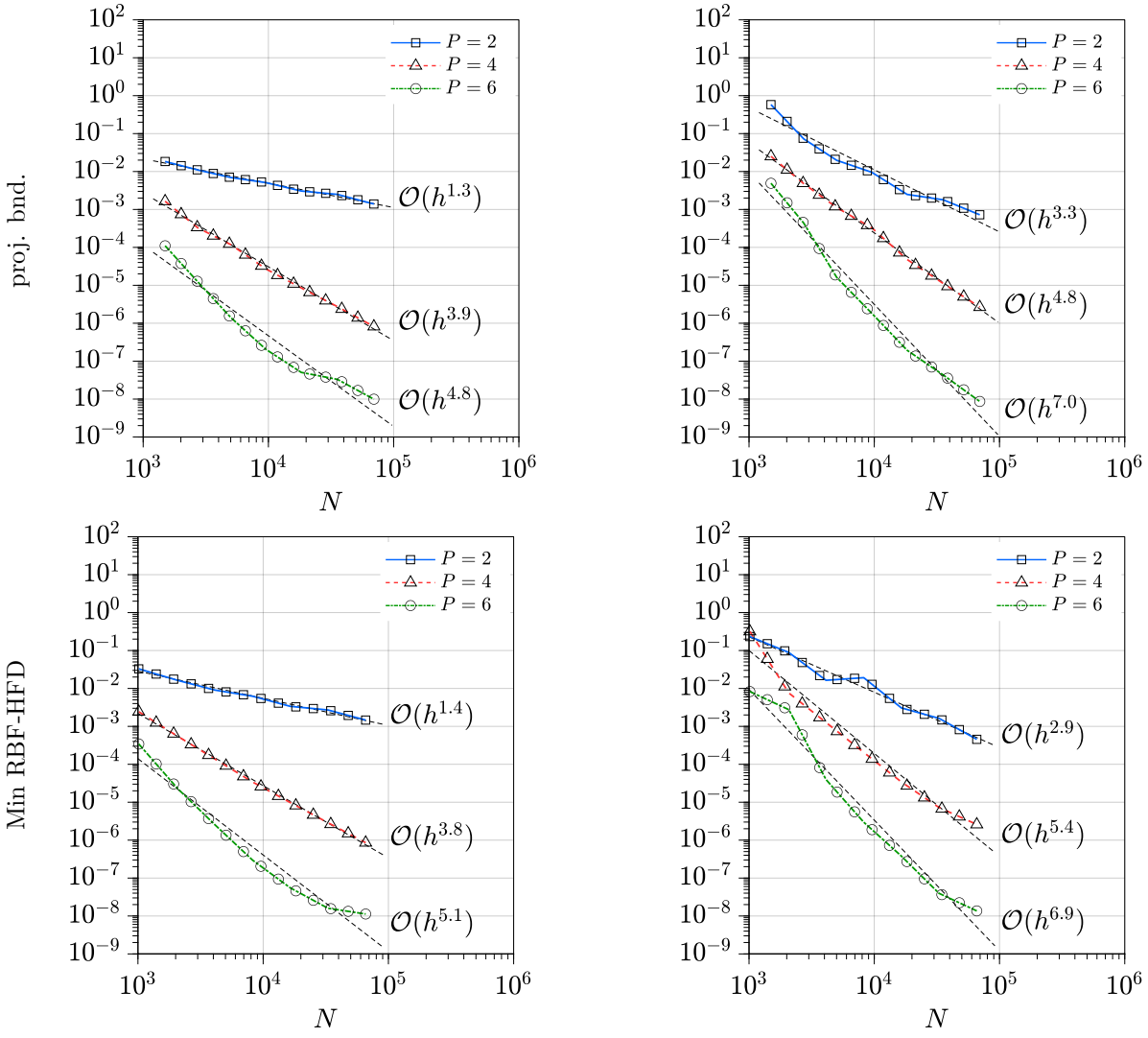


Figure 7.2: convergence curves for discretization error and solution error of the Poisson Equation on the geometry of Figure 5.12 for RBF-FD stabilized with Approach 2 (cfr. 5.4.1) and for Minimal RBF-HFD formulation

we will keep this naming and use the RBF-HFD acronym regardless of the type of functionals included in u^h. When required by the context, however, in order to distinguish it from the minimal formulation stated above, we will refer to this enhanced scheme as *implicit* or *compact scheme* RBF-HFD formulation.

Here follows a presentation of the Compact scheme formulation where operator $\mathcal{F}$ is included in the interpolant u^h.

Suppose that the new u^h is required to satisfy the following conditions:

$$\begin{aligned} u^h(\boldsymbol{x}_j) &= u(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,I} \\ \mathcal{F}u^h(\boldsymbol{x}_j) &= \mathcal{F}u(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,C} \\ \mathcal{B}u^h(\boldsymbol{x}_j) &= \mathcal{B}u(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,B} \end{aligned} \tag{7.30}$$

where $\mathcal{X}_{i,C}$ is a set of nodes placed in the vicinity of $\boldsymbol{x}_i$ with $\boldsymbol{x}_i \notin \mathcal{X}_{i,C}$. $\mathcal{X}_{i,C}$ does not need to be a subset of the stencil $\mathcal{X}_i$, in the sense that it might even include nodes picked from a different node cloud, like in [22]. That said, in order to avoid any additional notation, we will suppose that $\mathcal{X}_{i,C} \subseteq \mathcal{X}_i \setminus \boldsymbol{x}_i$.

The matching Hermite interpolant u^h needs to be:

$$\begin{aligned} u^h(\boldsymbol{x}) = &\sum_{j=1}^{m_I} \alpha_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_{j=m_J}^{m} \alpha_j \mathcal{B}_2 \Phi(\boldsymbol{x}, \boldsymbol{x}_j) \\ &+ \sum_{q=1}^{m_q} \gamma_q \mathcal{F}_2 \Phi(\boldsymbol{x}, \boldsymbol{x}_q) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \end{aligned} \tag{7.31}$$

with the usual orthogonality conditions:

$$\sum_{j=1}^{m_I} \alpha_j p_k(\boldsymbol{x}_j) + \sum_{j=m_J}^{m} \alpha_j \mathcal{B} p_k(\boldsymbol{x}_j) + \sum_{q=1}^{m_q} \gamma_q \mathcal{F} p_k(\boldsymbol{x}_q) = 0, \quad k = 1, \dots, M \tag{7.32}$$

And the resulting linear system is:

$$\underbrace{\begin{pmatrix} \boldsymbol{\Phi}_{I,I} & \boldsymbol{\mathcal{B}_2}\boldsymbol{\Phi}_{I,B} & \boldsymbol{\mathcal{F}_2}\boldsymbol{\Phi}_{I,C} & \boldsymbol{P_I} \\ \boldsymbol{\mathcal{B}_1}\boldsymbol{\Phi}_{B,I} & \boldsymbol{\mathcal{B}_1}\boldsymbol{\mathcal{B}_2}\boldsymbol{\Phi}_{B,B} & \boldsymbol{\mathcal{B}_1}\boldsymbol{\mathcal{F}_2}\boldsymbol{\Phi}_{B,C} & \boldsymbol{\mathcal{B}P_B} \\ \boldsymbol{\mathcal{F}_1}\boldsymbol{\Phi}_{C,I} & \boldsymbol{\mathcal{F}_1}\boldsymbol{\mathcal{B}_2}\boldsymbol{\Phi}_{C,B} & \boldsymbol{\mathcal{F}_1}\boldsymbol{\mathcal{F}_2}\boldsymbol{\Phi}_{C,C} & \boldsymbol{\mathcal{F}P_C} \\ \boldsymbol{P_I}^T & \boldsymbol{\mathcal{B}P_B}^T & \boldsymbol{\mathcal{F}P_C}^T & \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{M_{CS}}} \begin{pmatrix} \boldsymbol{\alpha_I} \\ \boldsymbol{\alpha_B} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_I} \\ \boldsymbol{\mathcal{B}u_B} \\ \boldsymbol{\mathcal{F}u_C} \\ \boldsymbol{0} \end{pmatrix} \tag{7.33}$$

Matrix $\boldsymbol{M_{CS}}$ at the left-hand side is usually symmetric and non singular, when this is the case, the solution can proceed as explained for the minimal formulation. When the final sparse linear system like (7.27) is built, additional equations will be required, thus partially defeating the computational advantage offered by the use of smaller stencils. This issue can be avoided only in specific circumstances, like the one encountered in the following section.

The main advantage offered by the compact scheme approach lies in the increase resolution in the reconstruction of the true partial differential operators. This is the topic of the following section.

7.4.2 Experimental Results

In this section some results attained using the RBF-HFD method are reported, they can also be found in the master's thesis [1].

The problem of interest is the Poisson Equation with Dirichlet boundary conditions:

$$\begin{cases} \triangle u = f & in\ \Omega \\ u = g & on\ \partial\Omega \end{cases} \tag{7.34}$$

where f and g are known functions. The Multiquadric (MQ) basic function was used, cfr. 2.1. Most results were achieved using stationary interpolation, where ε is adjusted as a function of the spacing s between nodes according to equation (4.21).

The Hermite interpolation scheme is required to satisfy the following conditions at the stencil level:

$$\begin{aligned} u^h(\boldsymbol{x}_j) &= u(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \mathcal{X}_i \\ \triangle u^h(\boldsymbol{x}_j) &= \triangle u(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \mathcal{X}_{i,C} \end{aligned} \tag{7.35}$$

where $\mathcal{X}_{i,C}$ is the set of all nodes except the central one $\boldsymbol{x}_i$ and $u(\boldsymbol{x}_j) = g(\boldsymbol{x}_j)$ if $\boldsymbol{x}_j \in \mathcal{X}_{i,B}$. Furthermore, because of the choice $\mathcal{F} = \triangle$ we have that $\triangle u(\boldsymbol{x}_j) = f(\boldsymbol{x}_j)$ if $\boldsymbol{x}_j \in \mathcal{X}_{i,C}$. In order to simplify the notation, we suppose that in the local numbering of the nodes the central node is the first, i.e. $\boldsymbol{x}_i = \boldsymbol{x}_j$ if $j = 1$.

The interpolant u^h takes the form:

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{m} \alpha_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_{q=2}^{m} \gamma_q \triangle_2 \Phi(\boldsymbol{x}, \boldsymbol{x}_q) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \tag{7.36}$$

with the usual orthogonality conditions.

Since $\mathcal{F}$ is the same differential operator operator appearing in the governing equation (7.34), some simplifications are possible at the moment of assembling the global linear system. At the center of each stencil we have:

$$\triangle u^h(\boldsymbol{x}_i) = \begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{\gamma} & \boldsymbol{\beta} \end{pmatrix} \underbrace{\begin{pmatrix} \triangle_1 \Phi(\boldsymbol{x}_i, \mathcal{X}_i) \\ \triangle_1 \triangle_2 \Phi(\boldsymbol{x}_i, \mathcal{X}_{i,C}) \\ \triangle \boldsymbol{p}(\boldsymbol{x}_i) \end{pmatrix}}_{\boldsymbol{B_{CS}}(\boldsymbol{x}_i)} \tag{7.37}$$

Which, by substituting interpolation coefficients $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, becomes:

$$\triangle u^h(\boldsymbol{x}_i) = \boldsymbol{c_I}(\boldsymbol{x}_i)^T \begin{pmatrix} u(\boldsymbol{x}_1) \\ \vdots \\ u(\boldsymbol{x}_{m_I}) \end{pmatrix} + \boldsymbol{c_B}(\boldsymbol{x}_i)^T \underbrace{\begin{pmatrix} g(\boldsymbol{x}_{m_J}) \\ \vdots \\ g(\boldsymbol{x}_m) \end{pmatrix}}_{\boldsymbol{g}} + \boldsymbol{c_{CS}}(\boldsymbol{x}_i)^T \underbrace{\begin{pmatrix} f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_m) \end{pmatrix}}_{\boldsymbol{f}} \tag{7.38}$$

where coefficient vectors $\boldsymbol{c_I}(\boldsymbol{x}_i)$, $\boldsymbol{c_B}(\boldsymbol{x}_i)$ and $\boldsymbol{c_{CS}}(\boldsymbol{x}_i)$ satisfy:

$$\boldsymbol{M_{CS}}^{-T} \begin{pmatrix} \boldsymbol{c_I}(\boldsymbol{x}_i) \\ \boldsymbol{c_B}(\boldsymbol{x}_i) \\ \boldsymbol{c_{CS}}(\boldsymbol{x}_i) \\ \boldsymbol{c_p}(\boldsymbol{x}_i) \end{pmatrix} = \boldsymbol{B_{CS}}(\boldsymbol{x}_i) \tag{7.39}$$

When the final sparse linear system like (7.27) is built, the products $\boldsymbol{c_B}^T \boldsymbol{g}$ and $\boldsymbol{c_{CS}}^T \boldsymbol{f}$, being known, will go at the right-hand side, thus leading to a system of size $N_I \times N_I$, as those attained with standard RBF-FD or minimal formulation. In cases like this, adopting a compact scheme RBF-HFD will also allow significant improvements in terms of computational efficiency.

We remark that, because of the presence of Dirichlet BC, minimal RBF-HFD and RBF-FD coincide and are simply characterized by empty sets $\mathcal{X}_{i,C}$.

7.4.3 1D Case

The 1D case consists in finding the solution to the Poisson equation when the domain is the closed interval $[0, 1]$. The function $q(x) = -\omega^2 \sin(\omega x)$ is being used as source term, yielding the following conditions:

$$\Delta u(x) = -\omega^2 \sin(\omega x) \text{ for } x \in [0, 1]$$

$$u(x) = \sin(\omega x) \text{ for } x \in \{0, 1\}$$

with $\omega = 2\pi$.

The domain is discretized with equally spaced nodes and the distance between two nodes is $s = h = |x_2 - x_1|$. A stencil $\mathcal{X}_i$ is composed of three nodes as in the figure 7.4 and the center node $\boldsymbol{x}_i$ is always the one in the middle.

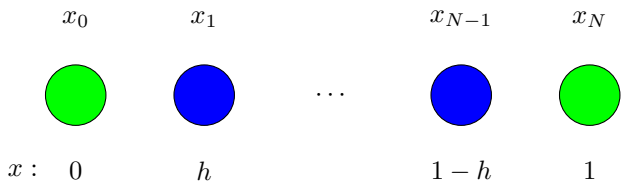


Figure 7.3: Domain in the 1D case.

The maximum degree P for the polynomial basis varies with the chosen method. For the RBF-FD method, where three interpolation conditions are enforced, we can have a polynomial degree up to $P = 2$. For the compact scheme, instead, there are five interpolation conditions (three on the function values and two on the differential operator values) and polynomials of degree $P = 3$ and $P = 4$ are made possible. We remark that in the 1D case there are no

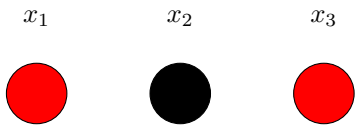
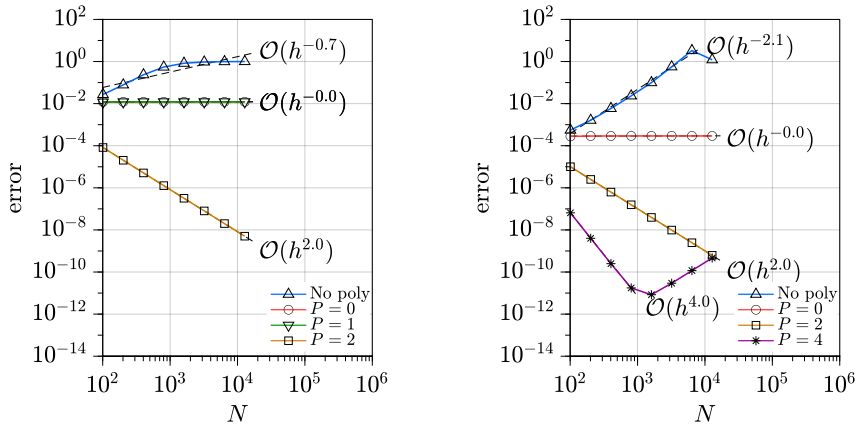


Figure 7.4: Stencil of the 1D case.

unisolvency issues, therefore the maximum polynomial degree can be chosen safely.

In order to improve the conditioning of the interpolation matrix $\boldsymbol{M_{CS}}$, operator $\mathcal{F} = \frac{1}{\varepsilon^2}\Delta$ is enforced instead of the simplest $\triangle$, in order to compensate for the appearance of ε^2 terms in the derivation of the MQ RBF.

With these assumptions, we have studied the differences between the RBF-FD method and the RBF-HFD method by comparing the respective orders of accuracy and by performing a Fourier analysis. The solution error is defined as in equation (7.29).



(a) RBF-FD (b) RBF-HFD

Figure 7.5: convergence curves for the solution error

Figures 7.5a and 7.5b compare the orders of accuracy achieved by the two methods. The main highlight is that the adoption of compact schemes does increase the discretization error but it is nonetheless advantageous. Indeed, the RBF-HFD reduces the solution error by an order of magnitude when P is the same but it also permits the adoption of a polynomial of degree $P = 4$. In the compact scheme with $P = 4$, the error is much smaller than any other of the methods and has an order of convergence $\mathcal{O}(h^4)$.

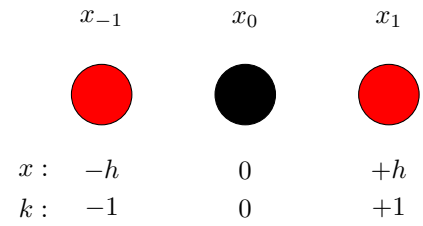


Figure 7.6: stencil considered during the Fourier analysis.

Fourier Analysis

The Fourier analysis, described in [60], has been adopted in order to compare the resolution characteristics of the RBF-FD and RBF-HFD methods. That is, the accuracy with which the discretization allowed by the two methods represents the exact result over the full range of length scales that can be realized on the given set of nodes (cfr. definition in [60]).

Consider the 1D Laplace operator d^2u/dx^2 applied to u. When u is the eigenfunction:

$$u(x) = e^{i\omega x} \tag{7.40}$$

then the second derivative, evaluated at any point $\boldsymbol{x}$ is $u''(\boldsymbol{x}) = -\omega^2 u(\boldsymbol{x})$, $-\omega^2$ being the corresponding eigenvalue.

If such operator is discretized using the RBF-HFD method we have:

$$\mathcal{L}u^h(x_i) = \sum_{j=-1}^{1} c_{I,j}(x_i)u(x_j) + \sum_{k=-1,1} c_{CS,k}(x_i)\frac{d^2u}{dx^2}(x_k) \tag{7.41}$$

In order to compare the approximate discretization coefficient we focus on a single stencil from the 1D domain, as depicted in Figure 7.6, where $\boldsymbol{x}_j = kh$.

By evaluating $\mathcal{L}u^h(x)$ at the central node, we obtain a value $A(\omega, h)$ such that

$$\mathcal{L}u^h(x) = A(\omega, h)e^{i\omega(kh)}$$

where $A(\omega, h)$ is the RBF-HFD approximation of the exact eigenvalue $-\omega^2$ and the same procedure can be followed for the classical RBF-FD method.

It turns out that $|A(\omega, h)|$ deviates from ω^2 more significantly as ω^{-1} reaches the minimum wavelength that can be theoretically resolved by the stencil. Figure 7.7 compares $|A(\omega, h)|$ with ω^2 using RBF-HFD and RBF-FD with different polynomial augmentation. Values of ω are reported on the x-axis while $\sqrt{A(\omega, h)}$ varies along the y-axis, both are scaled by the value h/π, which is the highest resolvable frequency. The diagonal blue line, given by the identity $y = x$, represents the target and make it easier to identify any deviation induced by the discrete approximation. It can be seen how the adoption of compact schemes significantly improves the approximation of the exact second derivative for a wider range of wavenumbers. This fact is also confirmed by the result obtained by Lele [60], where grid-based methods were used. We also remark that the impact

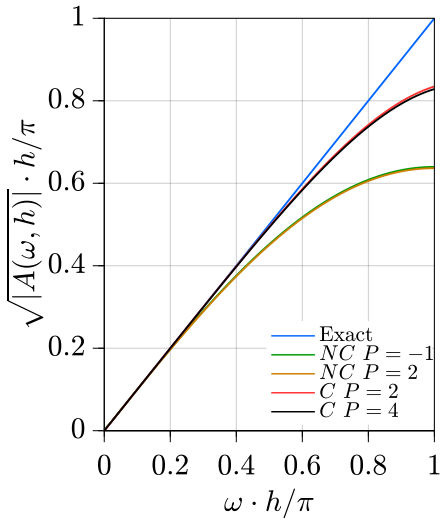


Figure 7.7: comparison of eigenvalues given by RBF-HFD (C) and RBF-FD (NC). "Exact" represents the exact derivative.

on accuracy provided by the adoption of compact schemes is not redundant with polynomial augmentation, indeed, from Figure 7.7 clearly emerges that higher polynomial degrees do not improve the resolution at lower wavelengths.

7.4.4 2D Case

The Poisson equation (7.34) is solved on the 2D domain Ω that can be seen in figure 7.8 with the following known functions f and g:

$$f(x,y) = -2\omega^2 \sin(\omega x)\sin(\omega y) \text{ for } (x,y) \in \Omega$$

$$g(x,y) = \sin(\omega x)\sin(\omega y) \text{ for } (x,y) \in \partial\Omega$$

with $\omega = 4\pi$.

The influence of the following parameters has been investigate:

- the node spacing, $s \propto \sqrt{1/N}$,
- the degree P of the polynomial term,
- the shape factor ϵ, that is the parameter of the RBFs,
- the number m of nodes included in each stencil.

We remind that the number of the nodes included in the stencil m is linked to P since there is a minimum number of required nodes for any order of polynomial term. It has been shown that, in the case of the standard RBF-FD method, a

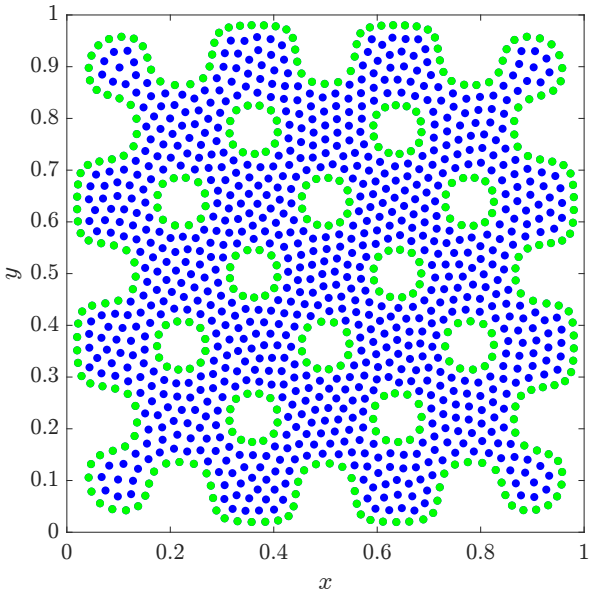


Figure 7.8: domain for the 2D case: green dots are boundary nodes, blue dots are internal nodes.

good choice for the number of nodes in the stencil is given by $m = 2M$ [33, 3, 4]. In order to determine m using a single digit, it was decided to use a factor n_m which satisfies $m = n_m M$, M being the number of terms in the polynomial basis.

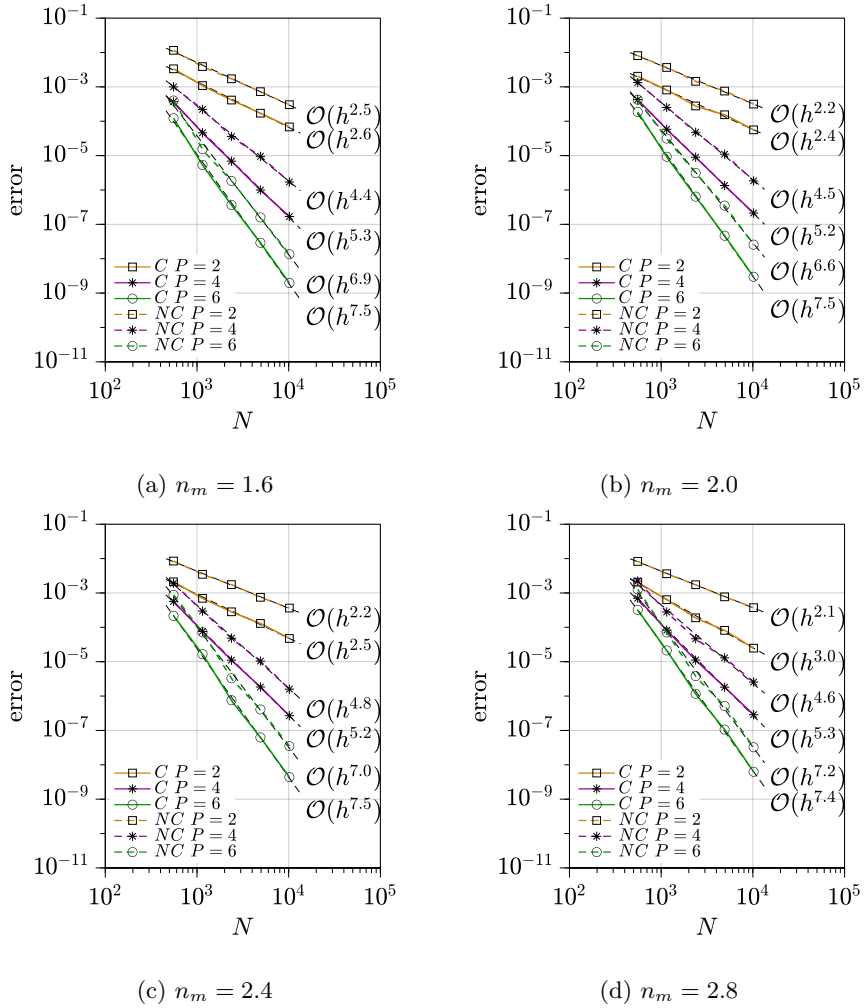
In the results presented below the error is always calculated using equation (7.29) as for the 1D case.

Figure 7.9 highlights how the compact schemes better approximate the solution if compared to the classic RBF-FD method: in all the cases investigated by varying P, n_m and N, the error given by the RBF-HFD is consistently smaller.

Figures 7.10 and 7.11 show the dependence of the error on the value ε when εs varies from 0.05 to 1.0. Results attained with different node densities are reported together, the polynomial degree remains fixed at $P = 4$. In the case of the non compact scheme (figure 7.10) the error trend show significant improvements at lower values of εs. The flattening of the RBF eventually induces a degradation in the performances due to ill-conditioning of the interpolation matrix but this happens at $\varepsilon s < 2.5$. The size of the stencil has a negligible impact in the case of the RBF-FD method.

For compact scheme RBF-HFD (figure 7.11), on the other hand, increasing the number of stencil nodes seems to shift the stability limit toward higher values of εs. Figure 7.12 is a zoom of what happens when ϵs varies from 0.05 to 0.5 with $n_m = 2.0$ and $N = 5022$. It appears that along with improving accuracy, compact schemes also worsen stability for flatter RBFs.

A good choice for the product εs in the case of the RBF-HFD should take



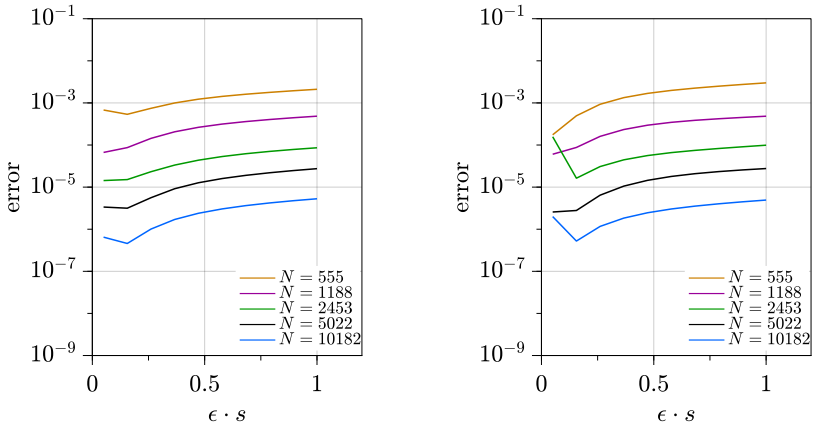
(a) $n_m = 1.6$ (b) $n_m = 2.0$

(c) $n_m = 2.4$ (d) $n_m = 2.8$

Figure 7.9: convergence curves for the solution error, comparison between RBF-HFD (C) and RBF-FD (NC).

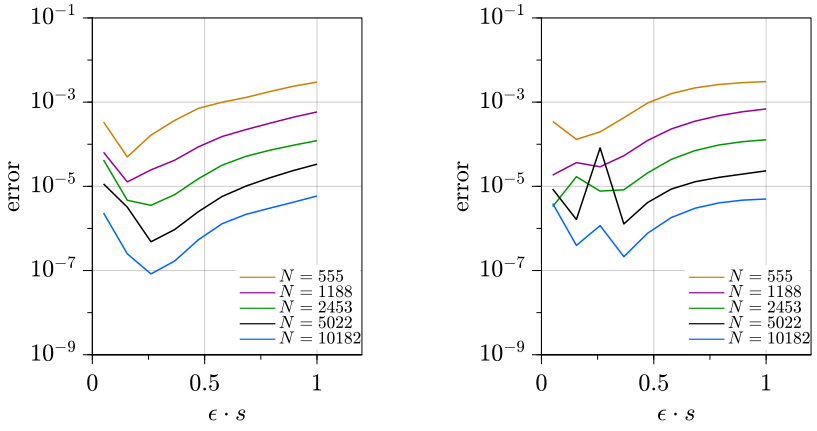
these remarks into account and therefore be higher than $\varepsilon_0 = 0.4$ as a rule of thumb.

Figure 7.13 is the same as Figure 7.5 but for the 2D case and shows the convergence of the solution error when stationary interpolation is used. In 2D, if the value of εs is reduced until $\varepsilon s = 0.5$, the error decrease as expected. For $\varepsilon s < 0.5$, however, the effect of round-off errors becomes more and more pronounced due to the ill conditioned interpolation matrices $\boldsymbol{M_{CS}}$, leading to a significant impact on the error. This can be seen, for instance, in figure 7.13



(a) $n_m = 1.6$ (b) $n_m = 2.0$

Figure 7.10: solution error against ε with the classic RBF-FD method in a range of node densities. Comparisons between stencils of different size n_m.



(a) $n_m = 1.6$ (b) $n_m = 2.0$

Figure 7.11: solution error against ε with the compact RBF-HFD method in a range of node densities. Comparisons between stencils of different size n_m.

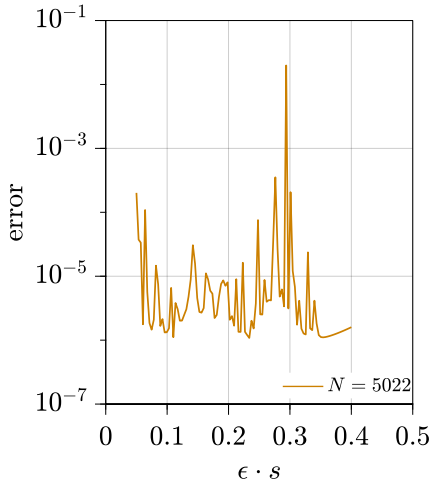
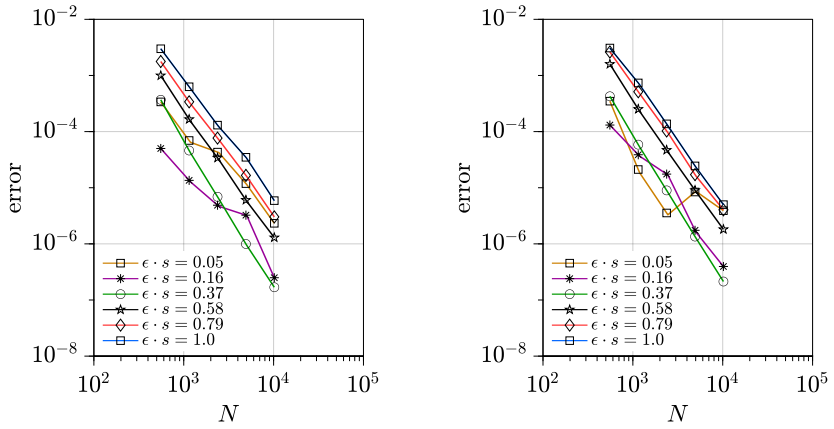


Figure 7.12: closer look at the case $N = 5022$, $n_m = 2.0$, RBF-HFD.

with the compact scheme RBF-HFD and $n_m = 1.6$.

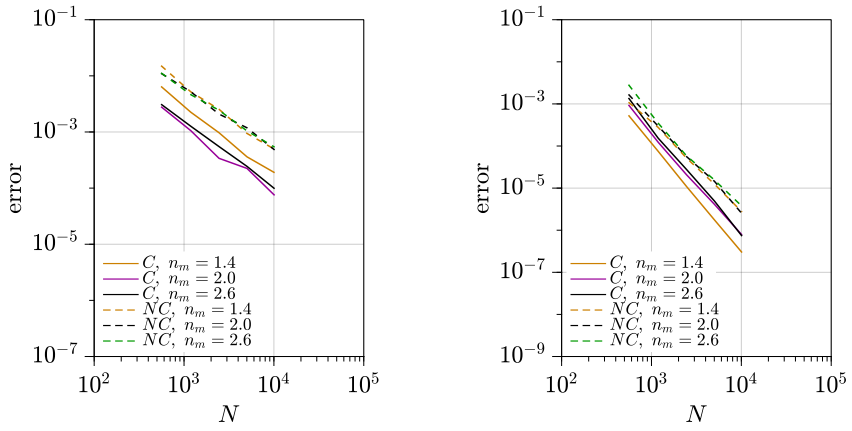


(a) $n_m = 1.6$ (b) $n_m = 2.0$

Figure 7.13: solution error with RBF-HFD method.

For the results reported Figure 7.14, the convergence curves achieved with stencils of different sizes are displayed together while the polynomial degree is kept fixed at $P = 2$ or $P = 4$. In order to avoid numerical instabilities, stationary interpolation with $\varepsilon s = 0.5$ was adopted.

Figure 7.15 is meant to provide a visual appreciation of the advantage in terms of accuracy retained by the compact RBF-HFD approach. It is a zoomed



(a) $P = 2$ (b) $P = 4$

Figure 7.14: convergence curves for the RBF-FD method (NC) and RBF-HFD method (C) when the number of nodes in the stencil (n_m) is varied while keeping the polynomial degree fixed at $P = 2$ or $P = 4$.

in view of the chosen domain and shows by how much stencil size and node density can be reduced using RBF-HFD method without losing accuracy. In this case the error is fixed approximately at $1.5 \cdot 10^{-5}$ with $P = 4$ and $\varepsilon s = 0.5$. $n_m = 2.0$ for the RBF-FD, which is also the most commonly adopted value, and $N = 5073$. In the compact scheme, in order to have the same error, only $N = 2494$ nodes are required and $n_m = 1.4$.

Fourier Analysis

The same exponential test function defined in (7.40) for in the 1D case was used here. In the 2D case the formulation of this analysis is slightly different since the value of ω can vary both along the x direction and the y direction. For this reason equation (7.40) becomes

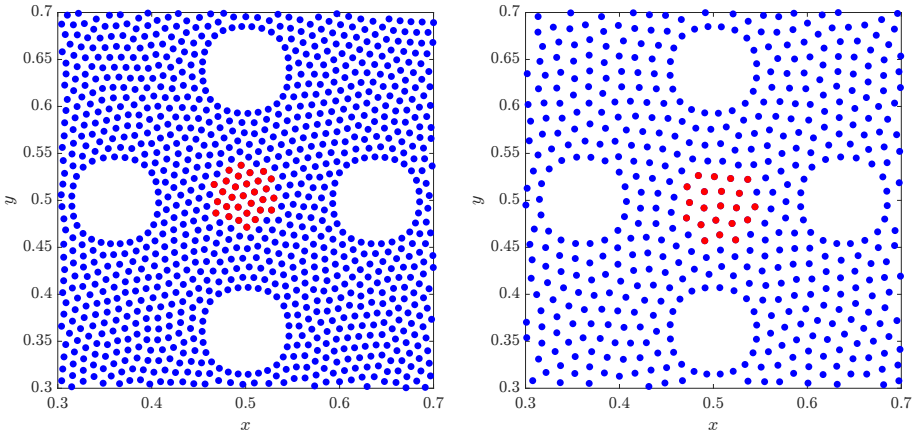
$$u(x, y) = e^{i(\omega_x(x-x_0)+\omega_y(y-y_0))}$$

where (x_0, y_0) is the center of the stencil and ω_x and ω_y are the wave numbers defined along the x-axis and the y-axis respectively. The Laplacian of this function is given by

$$\Delta u(x, y) = -(\omega_x^2 + \omega_y^2)e^{i(\omega_x(x-x_0)+\omega_y(y-y_0))}$$

The approximation of the Laplacian operator is evaluated both with the RBF-FD method and the compact scheme RBF-HFD method and will be written as

$$\mathcal{L}u^h(x, y) = A(\omega_x, \omega_y, d)e^{i(\omega_x(x-x_0)+\omega_y(y-y_0))}$$



(a) One stencil using the non compact scheme, the RBF-FD method

(b) One stencil using the compact scheme, the RBF-HFD method

Figure 7.15: the two figures show the number of nodes and the size of the stencil that is needed to reach a given error. The plotted area in the figure is is a zoom of the domain of Figure 7.8: red dots are those included in the stencil.

The value d is fixed and defined as the mean of the distances between the central node and the six nearest neighbors. This choice is motivated by the assumption of an hexagonal node arrangement achieved after an adequate number of iterations of the node-repel refinement (cfr Figure 3.1 page 41). The chosen stencil is the one in the middle of the domain that does not take into account nodes on the boundary as shown in figure 7.15. Parameters $N = 5022$, $n_m = 2.0$ and $\varepsilon d = 0.4$ were used for both methods.

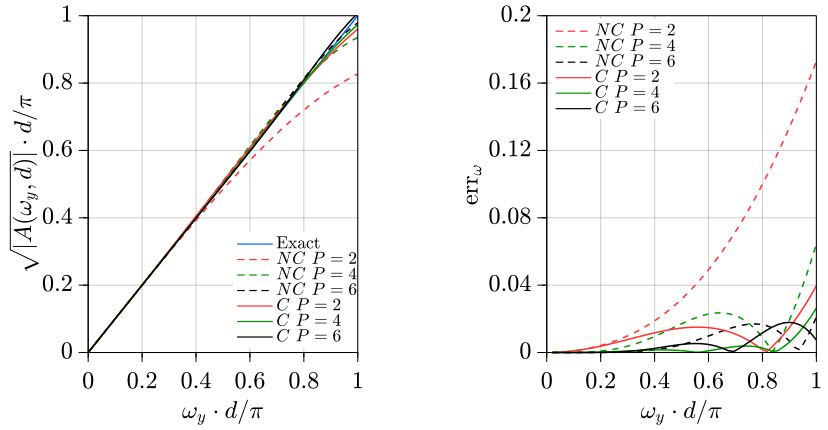
Figure 7.16a shows the dependence of the factor $A(\omega_x, \omega_y, d)$ upon ω_y when $\omega_x = 0$. As in the 1D case, the compact schemes provide a better approximation for the derivatives but the difference from the standard RBF-FD is smaller. In order to highlight the error between the methods we have defined the value err_ω, given by:

$$\mathrm{err}_\omega = \left| \frac{\frac{d}{\pi}\sqrt{|A(\omega_x, \omega_y, d)|} - \frac{d}{\pi}\sqrt{|\Delta u(x_0, y_0)|}}{\frac{d}{\pi}\sqrt{|\Delta u(x_0, y_0)|}} \right| \tag{7.42}$$

In Figure 7.16b err_ω is visualized against ω_y while $\omega_x = 0$ is always kept fixed. Slightly different conclusions can be drawn from the 2D case: while compact RBF-HFD approach remains the most efficient way to improve the resolution characteristics, the polynomial degree P also has a significant impact, suggesting that the adoption of the compact schemes in the 2D case is somewhat less compelling.

The situation is different, however, when both ω_x and ω_y can vary. In such case the Fourier analysis also provides information regarding the isotropy of the discrete approximation.

Figures 7.17a and 7.17b show the contour lines of $\sqrt{|A(\omega_x, \omega_y, d)|}$, i.e., the



(a) Magnitude of the scaled Laplacian eigevalue.

(b) Magnitude of the scaled Laplacian error (err_ω)

Figure 7.16: Fourier analysis with $\omega_x = 0$. C stands for the RBF-HFD method, NC stands for the RBF-FD method.

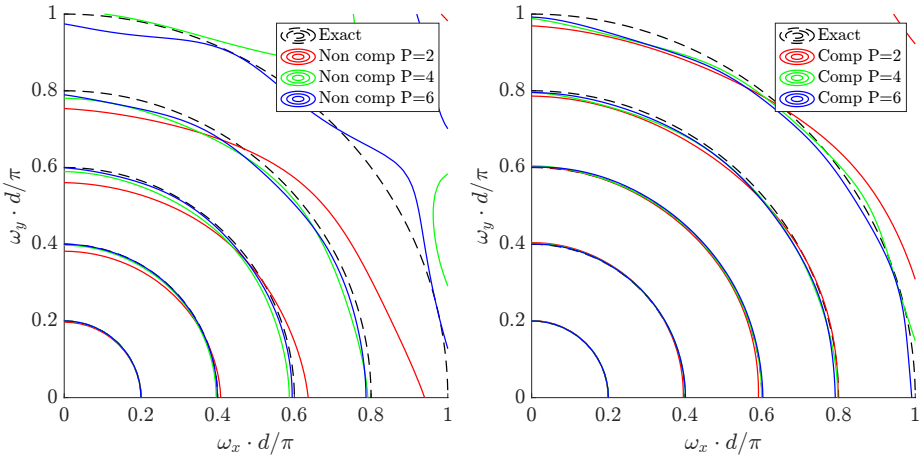
square root of the magnitude of the approximation of the Laplacian eigenvalue obtained with different schemes. The two wavenumbers ω_x and ω_y are scaled and plotted along the two axis. The dotted lines represent the exact solution, evaluated with the equation $\omega_x^2 + \omega_y^2 = c$, where c is a constant value, while the solid ones represent different approximations. The scaled wavenumbers $\omega_x d/\pi$ and $\omega_y d/\pi$ can vary in the interval $[0, 1]$, which is half of the period of the chosen periodic function.

In Figures 7.17a and 7.17b the lines correspond to the function $A(\omega_x, \omega_y, d)$ evaluated with a different number of polynomial terms for the standard RBF-FD and RBF-HFD schemes, respectively. Deviations from the circular dotted lines, especially visible for $P = 2$, are due to the shape of the stencil not being perfectly isotropic. As more nodes are included, in order to increase P, deviations from isotropy are also reduced. Unlike the one-dimensional analysis of Figure 7.16a, the comparison between Figures 7.17a and 7.17b reveal a significant difference in performance between standard RBF-FD and compact RBF-HFD. Indeed, the latter not only allows higher accuracy at smaller wavelength, but is also less sensitive to the shape of the stencil.

7.4.5 Conclusions

The theory of Generalized Hermite Interpolation can be considered as a generalization of that of Scattered Data Interpolation presented in the first chapters and naturally induce some radical improvements for the RBF-FD scheme.

To begin with, the minimal RBF-HFD scheme solves the problem of ill-conditioning due to boundary condition for any possible differential operator



(a) RBF-FD (b) RBF-HFD

Figure 7.17: approximation of the Laplacian operator

enforced at the boundary without introducing additional computational costs and therefore should be recommended over the stabilization methods discussed in chapter 5.

Furthermore, the compact RBF-HFD method seems to provide a clear advantage in terms of resolution at higher wavelengths and was found to also improve accuracy overall. While the computational advantage is probably debatable and case-specific, compact RBF-HFD also make the adoption of polynomial augmentation of higher degree possible without increasing the stencil size.

Results already attained by Lele in [60] for the Finite Difference method were confirmed for the RBF-HFD method and similar improvements are to be expected also for the 3D case.

Chapter 8

Conclusions and Further Developments

In this thesis we have discussed the development of a meshless solver based on the RBF-FD method for CFD and heat transfer simulations on complex geometries.

In order to effectively solve specific problems of engineering interest, the RBF-FD method, which provides a discretization for a given linear partial differential operator, can not be considered in isolation. The overall solution process can be initially subdivided in the successive execution of 4 different steps:

1. the generation of an adequately refined node distribution on a computational domain,

2. the linearization of the governing equation, the definition of the linear operators to be discretized and time integration strategies,

3. the discretization of the aforementioned differential linear operators by means of the RBF-FD (or RBF-HFD) scheme,

4. the numerical solution of the resulting linear system,

5. post processing and visualization of the results

In this work most of the attention is dedicated to point 3 and, more specifically, is focused on certain conditioning issues arising when the method is employed on complex 3D domains. Chapter 3 also deals with point 1 in view of its integration in the overall solution process. Some quality metrics for the node distribution are proposed with references to the literature and some ideas on the relative importance of the different aspects of a node generation algorithms are also discussed. At the moment, the development of node generation procedures for anisotropic node placement is underway in order to improve accuracy where the solution exhibits sharp variations along a known direction. Further developments

in this topic also include the implementation of an adaptive node refinement technique inspired by Slak and Kosec [94].

Many of the topics discussed in this dissertation become especially relevant as soon as the the solution process is applied to some basic engineering benchmark problem. In our case, it became apparent that ill-conditioning issues arising in presence of Neumann BC constituted a major obstacle preventing further developments as soon as the basic RBF-FD solver was applied to problems involving incompressible flow on 3D domains. Such issues were found not to be related to those yielded by flatness of the Radial Basis Function, by the Runge phenomenon or by inadequate quality of the node placement. Indeed, in most cases the reconstruction properties of RBF-based methods remain very good also as the conditioning of the interpolation problem deteriorates and becomes a real issue only as very high accuracy is sought with the stationary interpolation or in the flat limit in the non-stationary case. With arbitrary boundary conditions on arbitrarily complex domains, however, a straightforward adoption of the standard basis generated with translates of a radial kernel is no longer feasible.

The subsequent investigation of possible remedies constitutes most of my contributions to the subject and is largely exposed in Chapters 5 and 7. Multiple alternative improvements to the basic procedure, capable of overcoming the aforementioned ill-conditioning issues, were tested and few were found to enable the simulation of simple convection problems on arbitrarily complex 3D geometries. The three approaches presented here, namely node selection based on optimal directions, boundary node projection and Minimal RBF-HFD scheme, were selected for their robustness and geometrical flexibility. In particular, the investigation of the Generalized Hermite Interpolation and of its possible integration within the preexisting RBF-FD solver, led to the minimal RBF-HFD approach, which I consider to be the most versatile among the different stabilization techniques discussed above.

Finally, Compact RBF-HFD methods were also analyzed in their more traditional form and their impact on the spatial resolution was assessed by means of the Fourier Analysis. As a further development, Compact RBF-HFD might be adopted to improve the accuracy of the solution of the pressure Poisson equation in presence of incompressible fluid flows.

The vast majority of the code used for attaining the reported results was written using Julia programming language [6]. Its high-level syntax along with easily accessible high performance have been instrumental in allowing the rapid development and testing of prototypes based on countless different ideas. The adoption of this language helped in reducing the time required for obtaining initial implementations of the various numerical methods and, once that was done, for developing them to the level of refinement where meaningful tests and comparisons can be made.

In the long term,it is believed that solvers based on the RBF-HFD method will simplify the execution of complex simulations of engineering relevance and allow better integration with CAE software. Furthermore, stable implementations might also be used for highly accurate simulations of complex multi-phase phenomena.

Appendix A

Neumann Stability

A.1 d-matrices

Definition (d-matrix)**.** An $m \times n$ d-matrix is defined as the following $m \times n$ matrix $\mathcal{A} = (\boldsymbol{a}_{ij})$ of d-dimensional vectors $\boldsymbol{a}_{ij} \in \mathbb{R}^d$:

$$\mathcal{A} = \begin{pmatrix} \boldsymbol{a}_{11} & \cdots & \boldsymbol{a}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_{m1} & \cdots & \boldsymbol{a}_{mn} \end{pmatrix} \tag{A.1}$$

and the set of all $m \times n$ d-matrices will be denoted by $\mathbb{R}_d^{m,n}$. When $n = 1$ a d-matrix becomes a d-vector, d-vectors will be denoted as $\mathbb{R}_d^m$.

Operations involving d-matrices are defined as follows:

- product of $\mathcal{A} = (\boldsymbol{a}_{ij}) \in \mathbb{R}_d^{m,n}$ with a scalar $\lambda \in \mathbb{R}$:

$$\lambda\mathcal{A} = (\lambda \boldsymbol{a}_{ij}) \in \mathbb{R}_d^{m,n} \tag{A.2}$$

- sum of $\mathcal{A} = (\boldsymbol{a}_{ij}) \in \mathbb{R}_d^{m,n}$ and $\mathcal{B} = (\boldsymbol{b}_{ij}) \in \mathbb{R}_d^{m,n}$:

$$\mathcal{A} + \mathcal{B} = (\boldsymbol{a}_{ij} + \boldsymbol{b}_{ij}) \in \mathbb{R}_d^{m,n} \tag{A.3}$$

- product of $\mathcal{A} = (\boldsymbol{a}_{ij}) \in \mathbb{R}_d^{m,n}$ with $\mathbf{Q} = (q_{ij}) \in \mathbb{R}^{n \times p}$:

$$\mathcal{A}\mathbf{Q} = \left(\sum_{k=1}^{n} \boldsymbol{a}_{ik} q_{kj} \right) \in \mathbb{R}_d^{m,p} \tag{A.4}$$

For $\mathcal{A} = (\boldsymbol{a}_{ij}) \in \mathbb{R}_d^{m,n}$ and $\mathcal{V} = (\boldsymbol{v}_i) \in \mathbb{R}_d^m$, by using the dot product $\cdot$ in $\mathbb{R}^d$ we can define the following operator H:

$$H(\mathcal{A}, \mathcal{V}) = (\boldsymbol{a}_{ij} \cdot \boldsymbol{v}_i) \in \mathbb{R}^{m \times n} \tag{A.5}$$

Given $\mathcal{A}, \mathcal{B} \in \mathbb{R}_d^{m,n}$, $\mathcal{V} = (\boldsymbol{v}_i) \in \mathbb{R}_d^m$ and $\mathbf{Q} \in \mathbb{R}^{n \times p}$, the following equalities hold:

$$H(\mathcal{A}, \mathcal{V}) + H(\mathcal{B}, \mathcal{V}) = H(\mathcal{A} + \mathcal{B}, \mathcal{V}) \tag{A.6}$$

$$H(\mathcal{A}, \mathcal{V})\mathbf{Q} = H(\mathcal{A}\mathbf{Q}, \mathcal{V}) \tag{A.7}$$

$$\frac{\partial}{\partial v_{i,\eta}} \det\big(H(\mathcal{A}, \mathcal{V})\big) = \det\big(H(\mathcal{A}, \mathcal{V}_{i,\eta})\big) \tag{A.8}$$

for $i = 1, \dots, m$ and $\eta = 1, \dots, d$.

Equations (A.6),(A.7) follow from the simple distributive property of the dot product and the notation introduced above.

Equality (A.8), however, is less immediate and deserves a special attention because it is used for the calculation of optimal normal directions, we begin by explaining the notation used for the partial derivative. Suppose $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_d\}$ is the canonical basis of $\mathbb{R}^d$, then each element $\boldsymbol{v}_i$ of the d-vector $\mathcal{V}$ is an $\mathbb{R}^d$ vector itself. The notation $v_{i,\eta}$ is then used to indicate the projection on $\boldsymbol{e}_\eta$ of the i^{th} component $\boldsymbol{v}_i$ of vector $\mathcal{V}$:

$$v_{i,\eta} := \boldsymbol{v}_i \cdot \boldsymbol{e}_\eta \in \mathbb{R} \tag{A.9}$$

or, in other words, the η^{th} component of the i^{th} vector $\boldsymbol{v}_i$ of $\mathcal{V}$. We know that $\big(H(\mathcal{A}, \mathcal{V})\big)$ is a traditional matrix in $\mathbb{R}^{m,n}$ and we are then interested in understanding the sensitivities of its determinant with respect to each component of the d-vector $\mathcal{V}$ and this is what the partial differential symbol indicates. On the right-hand-side of the same equation, $\mathcal{V}_{i,\eta}$ is obtained from $\mathcal{V}$ by replacing its i^{th} element $\boldsymbol{v}_i$ with $\boldsymbol{e}_\eta$. By applying the chain rule it is possible to see that equality (A.8) follows from the multilinearity of the determinant and from the definition of the operator $H(\cdot, \cdot)$ (A.5).

A.2 Optimal directions

A.2.1 Context

In an attempt to lighten the main discussion on the ill-conditioning due to Neumann BC, the discussion on the algorithm adopted for the calculation of optimal normals was moved in this appendix. For reference, equation (5.53), which states the problem of finding the set of optimal normals is repeated here as problem (A.10). We are interested in finding the set $\hat{\mathcal{N}}$ of optimal directions $\{\hat{\boldsymbol{n}}_1, \dots, \hat{\boldsymbol{n}}_{m_B}\}$ that satisfy:

$$\begin{aligned} &\underset{\hat{\mathcal{N}}}{\arg\max} && \det\big(H(\mathcal{G}_{BB}, \mathcal{N})\big) \\ &\text{subject to} && \|\hat{\boldsymbol{n}}_i\|_2^2 = 1, \quad i = 1, \dots, m_B \end{aligned} \tag{A.10}$$

where a generic set of normals $\mathcal{N}$ is indicated with $\mathcal{N} = \{\bar{\boldsymbol{n}}_1, \dots, \bar{\boldsymbol{n}}_{m_B}\}$.

A.2.2 Generic properties

Equality constraints can be enforced by defining a Lagrangian function $\mathcal{L}$ with multipliers $\boldsymbol{\nu} = \{\nu_1, \dots, \nu_{m_B}\}$ as follows:

$$\mathcal{L}(\mathcal{N}, \boldsymbol{\nu}) = \det\big(H(\mathcal{G}_{BB}, \mathcal{N})\big) - \sum_{i=1}^{m_B} \nu_i \left(\|\bar{\boldsymbol{n}}_i\|_2^2 - 1\right) \tag{A.11}$$

it follows that the solution of problem (A.10) is given by the proper solution of the following system of equations:

$$\begin{cases} \dfrac{\partial}{\partial \hat{n}_{i,\eta}} \mathcal{L}(\hat{\mathcal{N}}, \boldsymbol{\nu}) = \dfrac{\partial}{\partial \hat{n}_{i,\eta}} \det\big(H(\mathcal{G}_{BB}, \hat{\mathcal{N}})\big) - 2\nu_i \hat{n}_{i,\eta} = 0 \\ \|\hat{\boldsymbol{n}}_i\|_2^2 = 1 \end{cases} \tag{A.12}$$

for $i = 1, \dots, m_B$ and $\eta = 1, \dots, d$. It is now possible to apply property (A.8) of appendix A.1:

$$\frac{\partial}{\partial \hat{n}_{i,\eta}} \det\left(H(\mathcal{G}_{BB}, \hat{\mathcal{N}})\right) = \det\left(H(\mathcal{G}_{BB}, \hat{\mathcal{N}}_{i,\eta})\right) \tag{A.13}$$

An equivalent formulation is obtained introducing the m_B auxiliary vectors $\boldsymbol{t}_i = \{t_{i,1}, \dots, t_{i,d}\} \in \mathbb{R}^d$ as follows, with the advantage of an easier enforcement of the constraints on the norms $\|\hat{\boldsymbol{n}}_{\boldsymbol{i}}\|$.

$$t_{i,\eta} = \det\left(H(\mathcal{G}_{BB}, \hat{\mathcal{N}}_{i,\eta})\right) \tag{A.14}$$

$$\hat{\boldsymbol{n}}_{i,\eta} = \pm \frac{\boldsymbol{t}_{i,\eta}}{\|\boldsymbol{t}_i\|_2} \tag{A.15}$$

for $i = 1, \dots, m_B$ and $\eta = 1, \dots, d$. Where the connection between the two formulations is given by:

$$2\nu_i = \pm \|\boldsymbol{t}_i\|_2 \tag{A.16}$$

We remark that the component η of $\hat{\boldsymbol{n}}_i$, $\hat{\boldsymbol{n}}_{i,\eta}$, does not appear in $\hat{\mathcal{N}}_{i,\eta}$ since it has been eliminated by derivation, therefore $\boldsymbol{t}_i$ does not depend on $\hat{\boldsymbol{n}}_i$. In the case of $m_B = 3$ boundary nodes, for instance, $t_{i,\eta}$ with $i = 2$ and $\eta = 1$ becomes:

$$t_{2,1} = \det\left(H(\mathcal{G}_{BB}, \mathcal{N}_{2,1})\right) = \begin{vmatrix} \boldsymbol{g}_{1,1} \cdot \hat{\boldsymbol{n}}_1 & \boldsymbol{g}_{1,2} \cdot \hat{\boldsymbol{n}}_1 & \boldsymbol{g}_{1,3} \cdot \hat{\boldsymbol{n}}_1 \\ \boldsymbol{g}_{2,1} \cdot \boldsymbol{e}_1 & \boldsymbol{g}_{2,2} \cdot \boldsymbol{e}_1 & \boldsymbol{g}_{1,3} \cdot \boldsymbol{e}_1 \\ \boldsymbol{g}_{3,1} \cdot \hat{\boldsymbol{n}}_3 & \boldsymbol{g}_{3,2} \cdot \hat{\boldsymbol{n}}_3 & \boldsymbol{g}_{3,3} \cdot \hat{\boldsymbol{n}}_3 \end{vmatrix} \tag{A.17}$$

From equation (5.51) we see that $\det(\mathbf{S}_{BB})$ is an odd function with respect to each normal $\bar{\boldsymbol{n}}_i$ and reversing any $\bar{\boldsymbol{n}}_i$ results in a change of sign of $\det(\mathbf{S}_{BB})$ because of the properties of the determinant. Multiple combination of changes in signs are possible and as a result the target function in equation (A.10) has at least 2^{m_B} local extrema which are equivalent, hence the symbol $\pm$ in equation

(A.15) for each of the m_B normals. However, once the direction of the optimal normals is known, their sign can easily be determined by picking those that are directed outward from the domain, in accordance with the usual convention. In other words, we are interested in the normals that maximize the magnitude of the target function $\det(\mathbf{S}_{BB})$, regardless of its sign. Furthermore, when $m_B > 1$ the target function also has saddle points solving equations (A.14)-(A.15) and is in general to be expected that multiple extrema can occur with $m_b > 3$ boundary nodes, both in 2D and in 3D.

We remark, however, that $\boldsymbol{g}_{i,j}$ are not arbitrary vectors but rather they are defined in equation (5.50) according to the relative positions of the stencil nodes. In practice, a sufficiently regular distribution of nodes in the stencil is usually enough to guarantee that the algorithm presented below provides the global maximum up to changes in signs, as confirmed by numerical experiments.

A.2.3 Computation of the optimal directions

Equations (A.14)-(A.15) represent a nonlinear system of coupled algebraic equations in the m_B unknown vectors $\hat{\boldsymbol{n}}_i$ and therefore an iterative solution process is required. The iteration step can be directly obtained from equation (A.14) as follows:

$$t_{i,\eta}^{(k+1)} = \det\big(H(\mathcal{G}_{BB}, \hat{\mathcal{N}}_{i,\eta}^{(k)})\big) \tag{A.18}$$

where the superscript (k) refers to iteration k. The values $t_{i,\eta}^{(k+1)}$ for $i = 1, \dots, m_B$ and $\eta = 1, \dots, d$ are therefore computed explicitly by using equation (A.18) where $\hat{\mathcal{N}}_{i,\eta}^{(k)}$ contains the normals at iteration k. Then the actual normals $\hat{\boldsymbol{n}}_i^{(k+1)}$ at iteration $k+1$ are obtained once again through the normalization expressed by equation (A.15).

In order to increase computational efficiency, it is convenient to compute the determinant in equation (A.18) as follows. Let us consider the explicit form of equation (A.14) where both row i and column i of matrix $H(\mathcal{G}_{BB}, \hat{\mathcal{N}}_{i,\eta})$ are moved to the last row and to the last column, respectively:

$$t_{i,\eta} = \det\big(H(\mathcal{G}_{BB}, \hat{\mathcal{N}}_{i,\eta})\big) = \\ = \left|\begin{array}{ccc|c} \boldsymbol{g}_{1,1}\cdot\hat{\boldsymbol{n}}_1 & \cdots & \boldsymbol{g}_{1,m_B}\cdot\hat{\boldsymbol{n}}_1 & \boldsymbol{g}_{1,i}\cdot\hat{\boldsymbol{n}}_1 \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{g}_{m_B,1}\cdot\hat{\boldsymbol{n}}_{m_B} & \cdots & \boldsymbol{g}_{m_B,m_B}\cdot\hat{\boldsymbol{n}}_{m_B} & \boldsymbol{g}_{m_B,i}\cdot\hat{\boldsymbol{n}}_{m_B} \\ \hline \boldsymbol{g}_{i,1}\cdot\boldsymbol{e}_\eta & \cdots & \boldsymbol{g}_{i,m_B}\cdot\boldsymbol{e}_\eta & \boldsymbol{g}_{i,i}\cdot\boldsymbol{e}_\eta \end{array}\right| = \left|\begin{array}{c|c} \mathbf{S}_i & \boldsymbol{c}_i \\ \hline \boldsymbol{r}_{i,\eta} & S_{ii,\eta} \end{array}\right| \tag{A.19}$$

By the properties of the Schur complement of the block $\mathbf{S}_i$ at the r.h.s. of (A.19), we have:

$$t_{i,\eta} = \left|\begin{array}{c|c} \mathbf{S}_i & \boldsymbol{c}_i \\ \hline \boldsymbol{r}_{i,\eta} & S_{ii,\eta} \end{array}\right| = \det(\mathbf{S}_i)\big(S_{ii,\eta} - \boldsymbol{r}_{i,\eta}\mathbf{S}_i^{-1}\boldsymbol{c}_i\big) \tag{A.20}$$

Since equation (A.20) holds for each of the d components of the vector $\boldsymbol{t}_i = \{t_{i,1}, \ldots, t_{i,d}\}$, the whole vector can be written directly as follows:

$$\boldsymbol{t}_i = \det(\mathbf{S}_i) \left(\boldsymbol{g}_{i,i} - \sum_{\substack{j=1 \\ j \neq i}}^{m_B} w_j \boldsymbol{g}_{i,j} \right) \tag{A.21}$$

where w_j are the $m_B - 1$ entries of the following column vector, which is the only term which depends on the normal directions:

$$\boldsymbol{w} = \mathbf{S}_i^{-1} \boldsymbol{c}_i \tag{A.22}$$

Furthermore, there is no need to compute $\det(\mathbf{S}_i)$ in equation (A.21) because it is a scalar and the vectors $\boldsymbol{t}_i$ are to be normalized in order to satisfy constraint (A.15).

With the above remarks, the iteration step (A.18) can be calculated without the need of any determinant as:

$$\boldsymbol{t}_i^{(k+1)} = \left(\boldsymbol{g}_{i,i} - \sum_{\substack{j=1 \\ j \neq i}}^{m_B} w_j^{(k)} \boldsymbol{g}_{i,j} \right) \tag{A.23}$$

where the dependence of the right-hand side upon the iteration index k arises only in the terms $w_j = w_j^{(k)}$. We also remark that vector $\boldsymbol{w} = \{w_1, \ldots, w_{m_B}\}$ does not depend upon dimensional component $\eta_1, \ldots, \eta_d$, and therefore it can be computed only once at each iteration for each of the m_B vectors $\boldsymbol{t}_i$.

Since the system of equations (A.14)-(A.15) is nonlinear, it is important to choose a proper set of initial vectors $\hat{\boldsymbol{n}}_i^{(0)}$ for the iterative scheme (A.18) to converge to the correct solution. Numerical experiments showed that a good starting point can be obtained from the diagonal vectors of d-matrix $\mathcal{G}_{BB}$ defined in equation (5.50):

$$\hat{\boldsymbol{n}}_i^{(0)} = \frac{\boldsymbol{g}_{ii}}{\|\boldsymbol{g}_{ii}\|_2} \tag{A.24}$$

for $i = 1, \ldots, m_B$. This choice corresponds the r.h.s. of equation (A.21) with $w_j = 0$ and the same equation (A.21) can also be employed to provide an improved starting point if any initial guess for w_j is available.

Another good starting point can be obtained from the following geometric heuristic:

$$\hat{\boldsymbol{n}}_i^{(0)} = \frac{\boldsymbol{x}_{m_I+i} - \bar{\boldsymbol{x}}}{\|\boldsymbol{x}_{m_I+i} - \bar{\boldsymbol{x}}\|_2} \tag{A.25}$$

where $\bar{\boldsymbol{x}}$ is some geometrical reference point for the stencil, e.g., the centroid of the internal nodes.

A.2.4 Computational costs

The process of finding optimal directions can be split into two phases:

- the computation of the d-matrix $\mathcal{G}_{BB}$, equation (5.50),
- the iterative solution of the associated system, equation (A.18).

The computation of $\mathcal{G}_{BB}$ requires in turn the computation of the $m_I \times m_B$ matrix $\bar{\psi}$ from equation (5.29), whose cost is therefore $\mathcal{O}(m_I^3) + \mathcal{O}(m_B \cdot m_I^2)$, i.e., factorization of $\boldsymbol{\varphi}_{II}$ + solution of m_B associated linear systems. Then, each iteration updates the m_B optimal directions through equation (A.23), each of which involves the solution of the $(m_B - 1) \times (m_B - 1)$ linear system $\mathbf{S}_i \boldsymbol{w} = \boldsymbol{c}_i$, thus leading to a cost $\mathcal{O}(N_{it} \cdot m_B^4)$, where N_{it} is the number of iterations which depends upon the desired level of convergence. We remark that the optimal directions do not need to be computed with high accuracy since they are only used in equation (5.63).

From a practical point of view, our numerical experiments showed that most of the computational time is due to the iterative phase, even if the number of boundary nodes m_B is usually much smaller than the number of internal nodes m_I of a single stencil. Finally, the computation of the optimal directions is required only for those stencils having boundary nodes and their number is usually much smaller than the total number of internal nodes N_I.

Finally, in case a boundary node is eliminated, this corresponds to the removal of the corresponding rows and columns of $\mathcal{G}_{BB}$ and therefore once these modifications are done there is no need to recompute the same matrix in order to calculate the new optimal normals.

A.3 Optimal position for boundary nodes

A.3.1 Optimization process

In Figure 5.2 it is clearly visible how the singular configurations, that is, those sending the condition number to infinity, are also characterized by the highest values of the Lebesgue constant. Starting from these remarks it seems natural to minimize the Lebesgue functions (as defined in equation (5.16)) by controlling the position of boundary nodes. Here follows a brief discussion on a method proposed for this purpose.

When no polynomial augmentation is used, cardinal functions $\psi_i(\boldsymbol{x})$ can be found by solving linear system (5.13) at page 73, Lebesgue functions and Lebesgue constants can consequently be defined as (5.16) and (5.17) respectively. In some cases λ_I and λ_B exhibit large values at the boundary when Neumann boundary conditions are employed, see for example Figure 5.3.

In order to find the configuration of boundary nodes that minimizes the Lebesgue constants Λ_I and Λ_B, the following cost function can be defined:

$$\mathcal{F}(\boldsymbol{x}_{m_J}, \ldots, \boldsymbol{x}_m) = \sum_{k=m_J}^{m} \lambda_I(\boldsymbol{x}_k) = \sum_{k=m_J}^{m} \sum_{i=1}^{m_I} |\psi_i(\boldsymbol{x}_k)| \tag{A.26}$$

with the usual indices. In equation (A.26) only the internal Lebesgue functions λ_I is considered, however, it was observed that optimizing with respect to the boundary Lebesgue functions λ_B leads to very similar optimal configurations.

By using a more compact vector notation, the sensitivities of the cost function with respect to the position of a boundary node $\boldsymbol{x}_j = \{x_{j,1}\,,\, x_{j,2}\}$ in 2D are:

$$\frac{\partial \mathcal{F}}{\partial x_{j,\eta}} = \sum_{k=m_J}^{m} \operatorname{sgn}\left(\boldsymbol{\psi}_I(\boldsymbol{x}_k)\right)^T \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}_k)}{\partial x_{j,\eta}} \tag{A.27}$$

where $\boldsymbol{\psi}(\boldsymbol{x}_k)$ is the column vector $\{\psi_1(\boldsymbol{x}_k), \ldots, \psi_m(\boldsymbol{x}_k)\}^T$ and $\operatorname{sgn}(\boldsymbol{\psi}_I(\boldsymbol{x}_k))$ is the column vector having the signs of $\boldsymbol{\psi}(\boldsymbol{x}_k)$ in the first m_I entries, and zeros for the last m_B indices corresponding to the boundary nodes.

The term $\partial\boldsymbol{\psi}(\boldsymbol{x}_k)/\partial x_{j,\eta}$ in equation (A.27) can be obtained by deriving equation (5.13) in the case with no polynomial augmentation as follows:

$$\frac{\partial \boldsymbol{\psi}(\boldsymbol{x}_k)}{\partial x_{j,\eta}} = (\mathbf{M}^T)^{-1} \left(\frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}_k)}{\partial x_{j,\eta}} - \frac{\partial \mathbf{M}^T}{\partial x_{j,\eta}} \boldsymbol{\psi}(\boldsymbol{x}_k) \right) \tag{A.28}$$

where $\frac{\partial \mathbf{M}^T}{\partial x_{j,\eta}}$ is obtained by taking the derivative of all the matrix entries.

We then substitute (A.28) into equation (A.27):

$$\operatorname{sgn}\left(\boldsymbol{\psi}_I(\boldsymbol{x}_k)\right)^T (\mathbf{M}^T)^{-1} = \left(\mathbf{M}^{-1} \operatorname{sgn}\left(\boldsymbol{\psi}_I(\boldsymbol{x}_k)\right)\right)^T = \boldsymbol{c}^T \tag{A.29}$$

where $\boldsymbol{c}$ is the column vector given as the solution of the following adjoint equation:

$$\mathbf{M}\,\boldsymbol{c} = \operatorname{sgn}\left(\boldsymbol{\psi}_I(\boldsymbol{x}_k)\right) \tag{A.30}$$

The computational cost for the calculation of the sensitivities of (A.27) then becomes independent of the dimension of the problem and requires the solution of m_B linear systems like equation (A.30), one for each boundary node.

The sensitivities can then be used to perform a gradient based optimization of the boundary nodes positions, this is done by taking the components orthogonal to the normals.

Whenever two nodes ended up overlapping during the optimization, one of them was removed, this is why in Figure A.1 only 6 boundary nodes appear.

A.3.2 Results

The optimization process described above was performed by starting from the reference stencil of Figure 5.1 and using MQ RBF with shape parameter ε satisfying equation (4.21) with $\varepsilon_0 = 0.5$. Once the convergence is reached, we are left with the situation depicted in Figure A.1. We can see that different values of the angle α correspond to different optimal placements of the boundary nodes.

We remark that, even if the procedure described so far gives the best placement for the boundary nodes, this comes at a relatively high computational cost. Indeed, it is required to solve m_B local linear systems in order to calculate

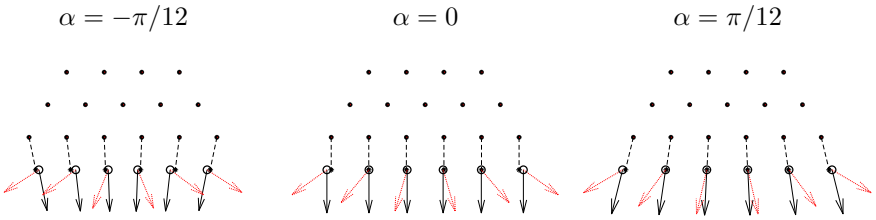


Figure A.1: optimal placement of boundary nodes obtained starting from the reference stencil of Figure 5.1. Black solid lines are the actual normals, red dotted lines are the optimal directions, asterisks are the position of boundary nodes obtained by projecting the first layer of inner nodes along the normals.

the cardinal functions, in addition to m_B linear systems like (A.30) in order to estimate the sensitivities at each step of the optimization. On top of that, a control on the normals, like the one explained in section 5.4.1, is still needed when the boundary has high curvature, see the angle between actual and optimal normals for the case $\alpha = -\pi/12$ in Figure A.1.

A.3.3 Projected nodes

Fortunately, all results obtained so far suggest that the interpolation can also be improved by placing the boundary nodes on the locations identified by projecting along the normals the inner nodes of the first layer. This can be deduced also by looking at Figure A.1, optimal node placement is obtained by projecting internal nodes along the dashed black lines, which are almost parallel to the actual normals. In all cases this was found to be beneficial for the stability in presence of Neumann BC and also enhancing accuracy. Replacing standard boundary nodes with the projected ones can be done within the node generation phase and requires the adoption of a different node generation procedure. In presence of complex geometries, for instance, a modification of the node-repel algorithm might produce the desired result with a marginal increase in complexity.

Appendix B

Additional Benchmark Results

B.1 Differentially Heated Cubic Cavity

Some visualizations of the relative error from the Differentially Heated Cubic Cavity (DHCC) benchmark are reported below. They refer to the results discussed in Chapter 6 and, more specifically, in subsection 6.5.4.

Figures B.1, B.2, B.3 and B.4 display the relative error of different meaningful flow quantities with respect to a reference solution, for each combination of P and Ra. In Figures B.1 and B.2 the relative error of temperature T and vertical velocity w is depicted along the horizontal center-line $(x, 0, 0)$. In Figure B.3 the relative error of the horizontal velocity u is depicted along the vertical center-line $(0, 0, z)$. In Figure B.4 the relative error of the normal derivative of the temperature $\partial T/\partial \boldsymbol{n}$ is depicted along the vertical midline $(0.5, 0, z)$. When the normal derivative is averaged on this line it takes therefore the value Nu_m reported in Tables 6.6, 6.7 and 6.8.

Since grid-independent Fluent solutions would require a very high number of cells for the required level of accuracy, the reference solution is obtained through an extrapolation from the two finest Fluent meshes, i.e., $G5$ and $G6$, once again by applying Eq. (6.14) with $q = 2$, $d = 3$. This yields the following expression for the reference solution $\hat{a}(\boldsymbol{x})$ for a generic field $a(\boldsymbol{x})$:

$$\hat{a}(\boldsymbol{x}) = \frac{a_{G6}(\boldsymbol{x}) M_{G6}^{2/3} - a_{G5}(\boldsymbol{x}) M_{G5}^{2/3}}{M_{G6}^{2/3} - M_{G5}^{2/3}} \tag{B.1}$$

where M_{G5} and M_{G6} are the number of Fluent cells for meshes $G5$ and $G6$, respectively.

The relative error $e_r(\boldsymbol{x})$ is then obtained as follows:

$$e_r(\boldsymbol{x}) = \frac{|a_{\mathrm{ID}}(\boldsymbol{x}) - \hat{a}(\boldsymbol{x})|}{\max\limits_{\boldsymbol{x} \in \Lambda} |\hat{a}(\boldsymbol{x})|} \tag{B.2}$$

where $a_{\mathrm{ID}}(\boldsymbol{x})$ is the RBF-FD solution computed with node distribution ID and Λ is a given subset of the domain, e.g., center-lines or the midline in this case.

Figures B.1, B.2, B.3 and B.4 show that in most cases the decrease in the relative error is monotone with respect to the number of meshless nodes, i.e., going from $N1$ to $N5$, although in some cases it is not, especially for $P = 2$. A clear advantage is visible for the case when the polynomial degree is the highest, i.e., $P = 4$, especially in the case $Ra = 10^3$. For higher Ra values the advantage of a high polynomial degree P is somehow limited by the insufficient node resolution in the boundary layers, as already pointed out. Once again we outline that the node placement was not optimized by taking into account the solution, but rather to make any comparison easier.

The decrease in the relative error with respect to the polynomial degree is also monotone in most cases, with the most evident exception for the relative error of the temperature along the horizontal center-line, Figure B.1, where the intermediate case $P = 3$ seems to be the less accurate. Nonetheless, the relative errors are less than 10^{-2} in most cases, which is generally acceptable in the engineering practice. Extremely small relative errors, in the order of 10^{-5}, are obtained for the highest polynomial degree $P = 4$ and $Ra = 10^3$, while the largest errors emerge in the opposite case, i.e., $P = 2$ and $Ra = 10^5$, as expected.

The noise effect visible in many of the figures, and especially evident for the cases with very low relative error, is due to the fact that, when the RBF-FD solution locally intersect the reference one, the error goes to zero in the point of intersection, sending the logarithmic graph to $-\infty$ very sharply. The noise thus appear to be greater when there are more points of intersection between the two solutions compared.

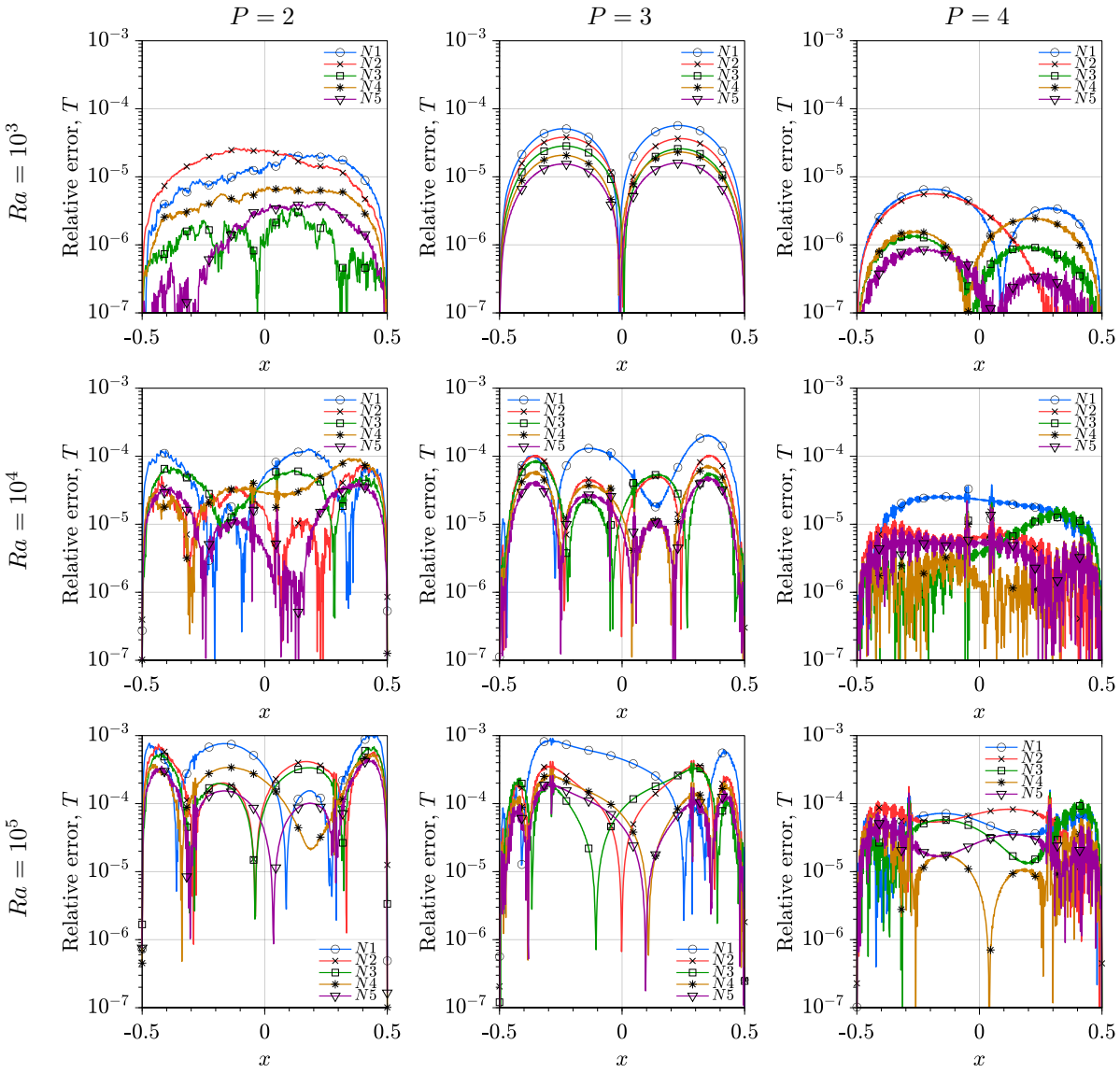


Figure B.1: differentially heated cavity: relative error of temperature T along center-line $(x, 0, 0)$.

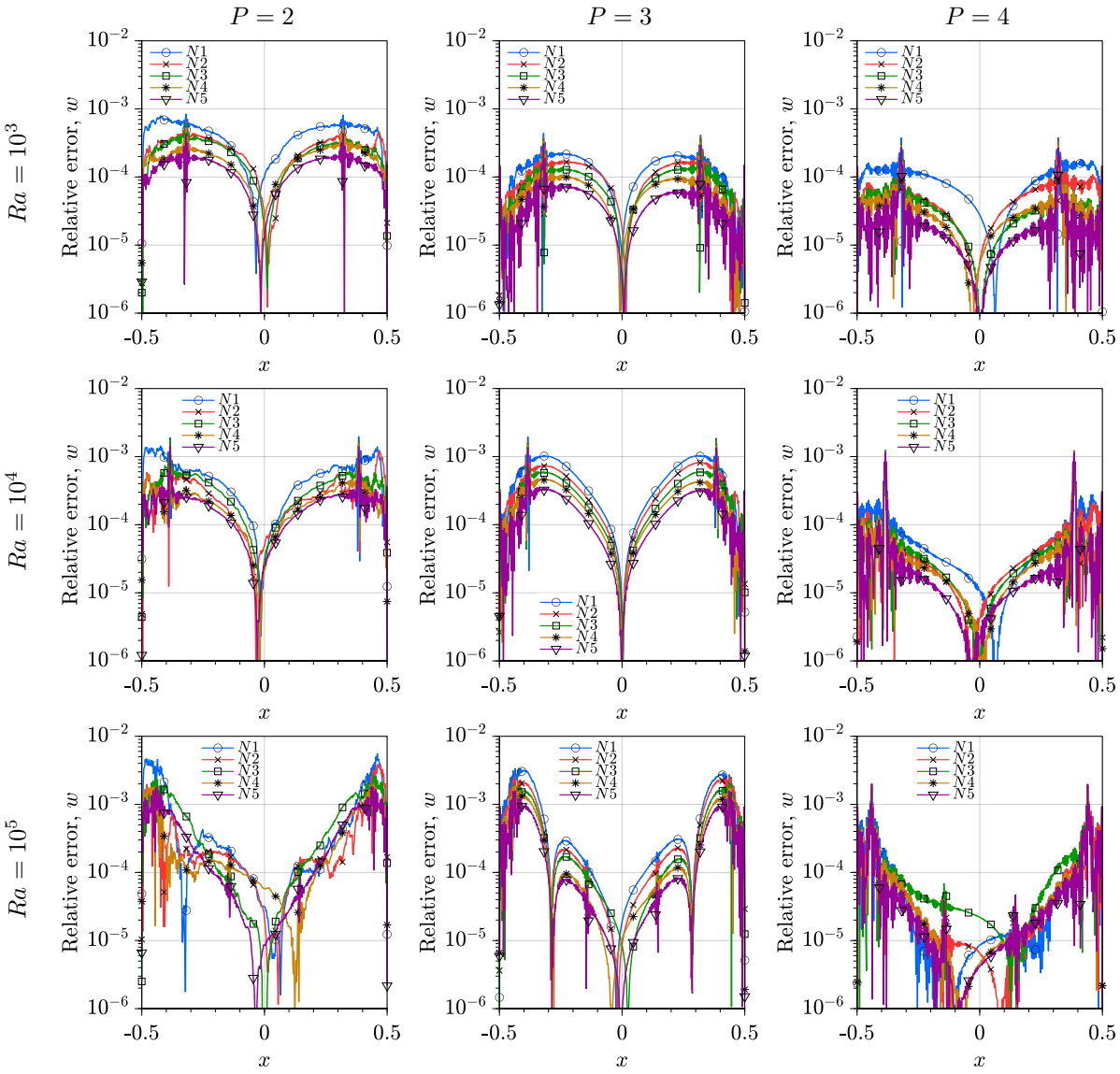


Figure B.2: differentially heated cavity: relative error of velocity component w along center-line $(x, 0, 0)$.

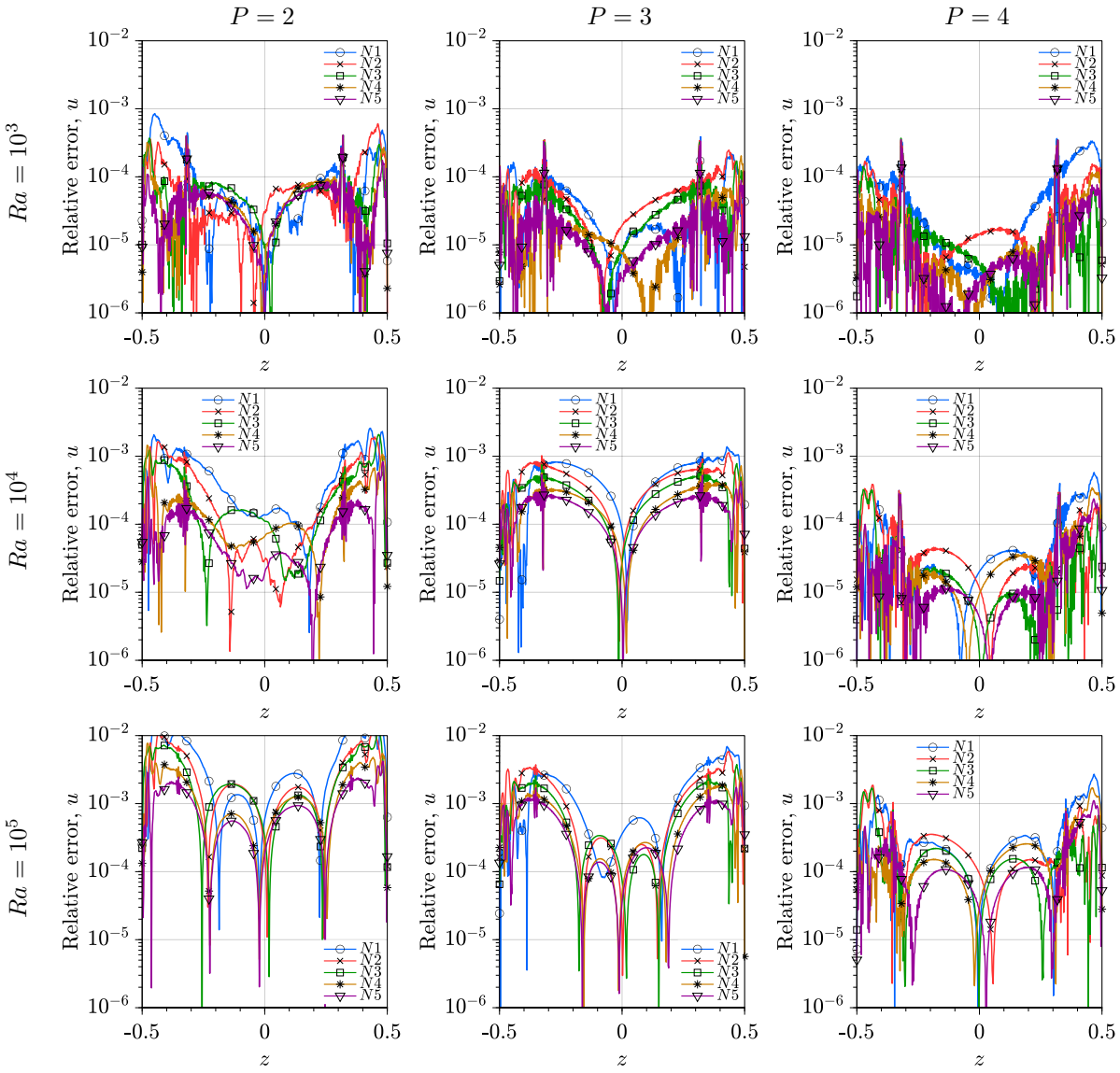


Figure B.3: differentially heated cavity: relative error of velocity component u along center-line $(0, 0, z)$.

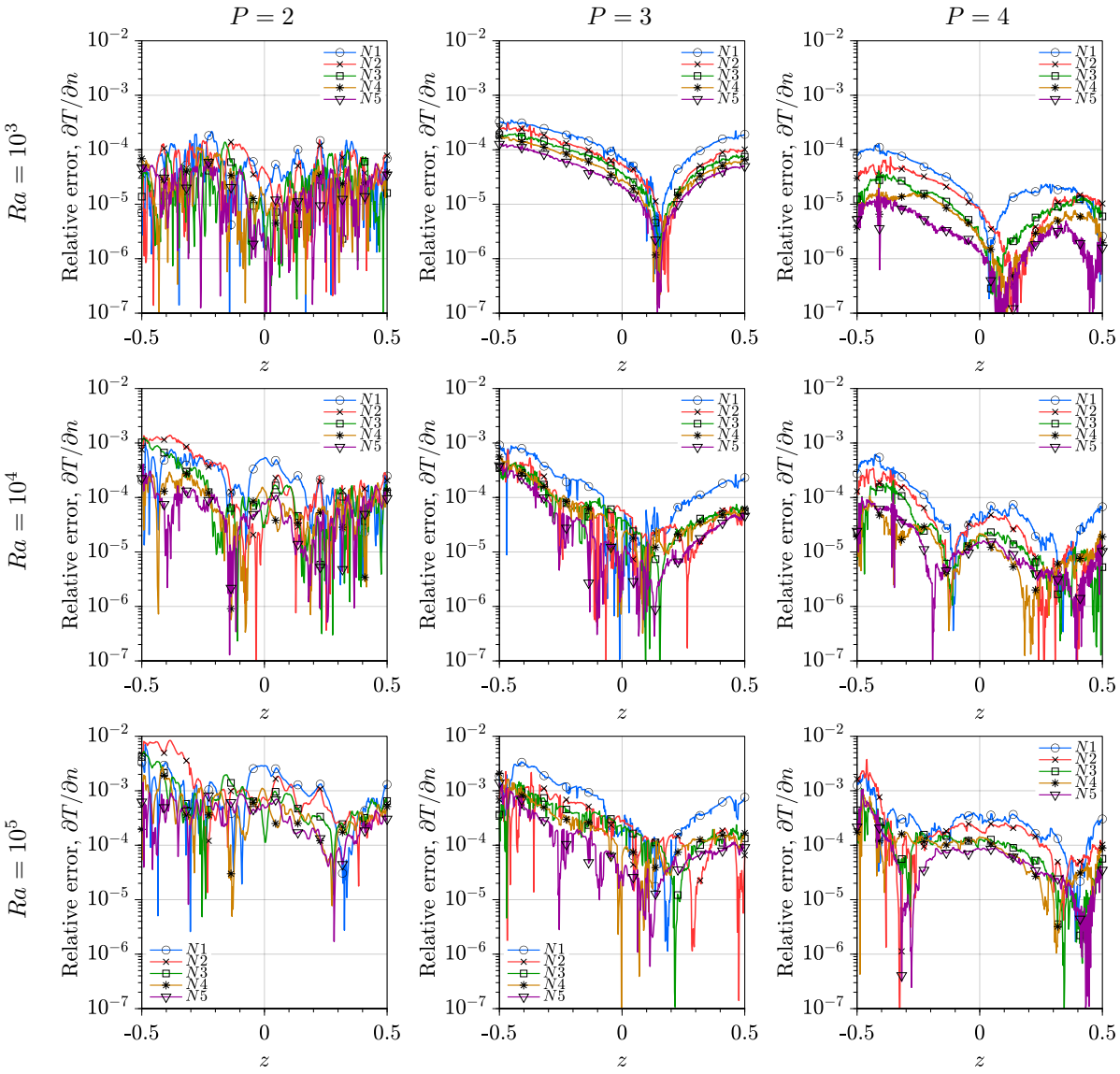


Figure B.4: differentially heated cavity: relative error of the normal derivative along the midline $(0.5, 0, z)$.

B.2 Differentially Heated Spherical Shell

Figures B.5, B.6, B.7, B.8, B.9 and B.10 display the relative error of different meaningful flow quantities with respect to a reference solution, for each combination of P and Ra. The reference solution is the one obtained with the most refined mesh in Fluent, i.e., $G8$, since the corresponding 2D axisymmetric solution is sufficiently accurate and no infinite grid extrapolation is needed.

In Figures B.5 and B.7 the relative error of temperature T and vertical velocity w is depicted along the upper portion of the symmetry axis z, i.e., $0.5 \leq z \leq 2.5$. Figures B.6 and B.8 show the same error along the lower portion of the symmetry axis z, i.e., $-2.5 \leq z \leq -0.5$. The relative error of the normal derivative of the temperature along the generatrices of the inner and outer spheres is shown in Figures B.9 and B.10, respectively, as a function of the polar angle $0 \leq \theta \leq \pi$ (see Figure 6.5; $\theta = 0$ is the north pole, $\theta = \pi$ is the south pole). The RBF-FD values of the normal derivative of the temperature along the generatrices of both spheres are averaged along the azimuthal direction in order to map the results of a fully 3D RBF-FD simulation to a 2D axisymmetric one.

In most cases the decrease in the relative error is monotone with respect to the number of meshless nodes, i.e., going from $N1$ to $N5$, although in some cases it is not, especially for the normal derivatives of the temperature at the inner sphere for $P = 3, 4$, Figure B.9.

In Figure B.6 it is possible to see how the error increases closer to the south pole of the inner sphere, i.e., $z = -0.5$, due to the presence of a thin boundary layer. This effect is accentuated as the Ra number increases and is mitigated both by the increase of the node density and of the polynomial degree P. Very similar remarks can be made to explain the appearance of the high errors of Figures B.7 and B.8 in the proximity of the poles of the inner sphere. The effect on the accuracy of inadequate spatial sampling due to the boundary layer is clearly visible in Figure B.8, where the relative error grows above 10^{-2} for $P = 2, 3$. In the case of the velocity field, however, increasing the polynomial degree turns out to be less effective then increasing the node density. About that, we remark that the adoption of a local refinement strategy would allow for a much greater increase in the local density of nodes than that achieved by going from $N1$ to $N5$. However, even in the present case, the accuracy is always good, i.e., the relative error is less than $\leq 10^{-2}$, the only exception being the cases with $P = 2, 3$ and $Ra = 1000$ in Figure B.8.

In both Figures B.9 and B.10 the errors of the normal derivative of the temperature appear to be effectively reduced both by increasing the node density and polynomial degree P, the latter being more effective, contrary to what holds for the velocity field. In the last figure we can also note that the error is larger for $\theta = 0$ due to the appearance of a thin boundary layer also at the north pole of the outer sphere. This phenomenon is especially relevant for the case $Ra = 1000$ where the convective plume is more pronounced.

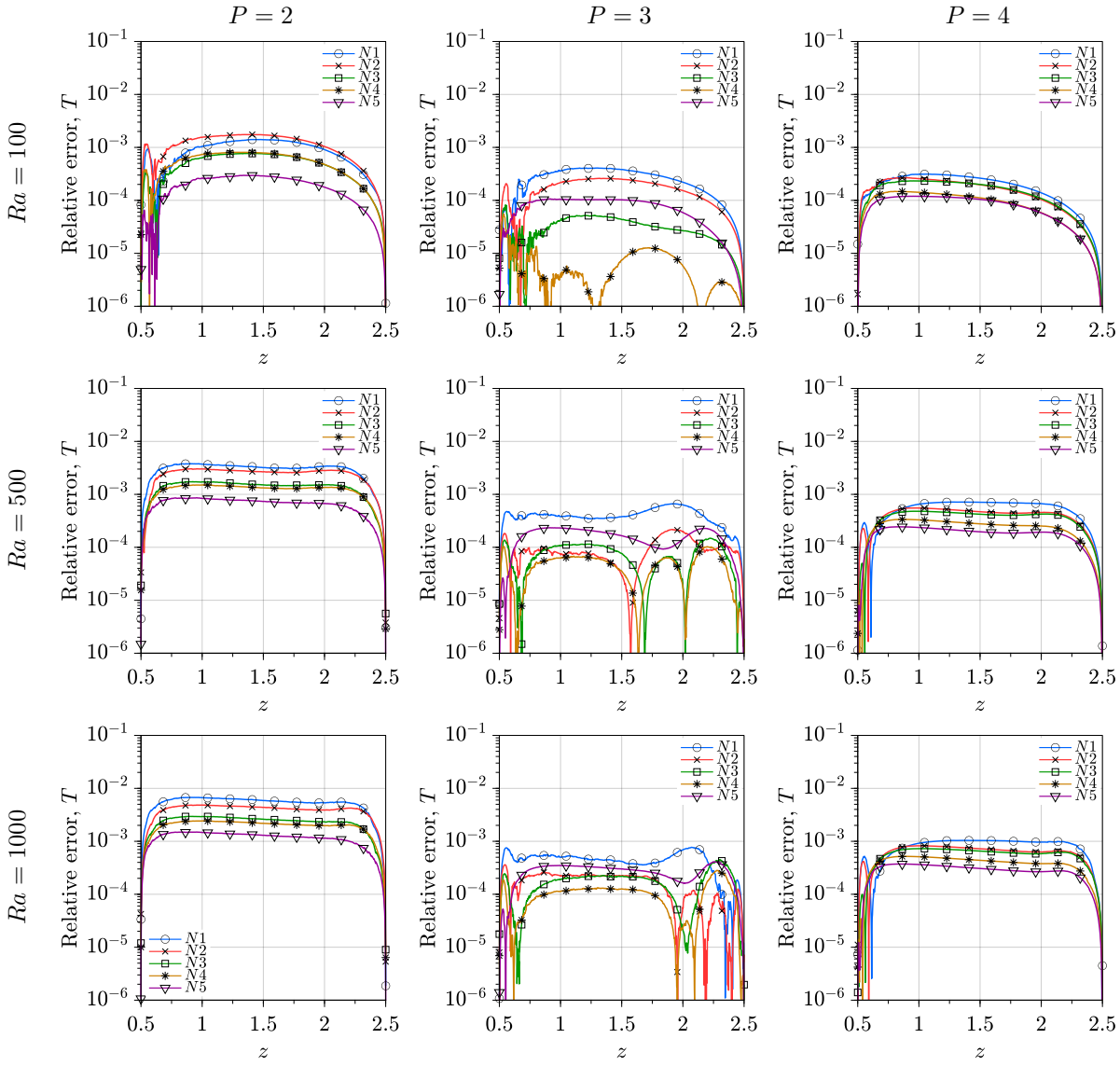


Figure B.5: spherical shell: relative error of temperature T along the upper portion of the symmetry axis z.

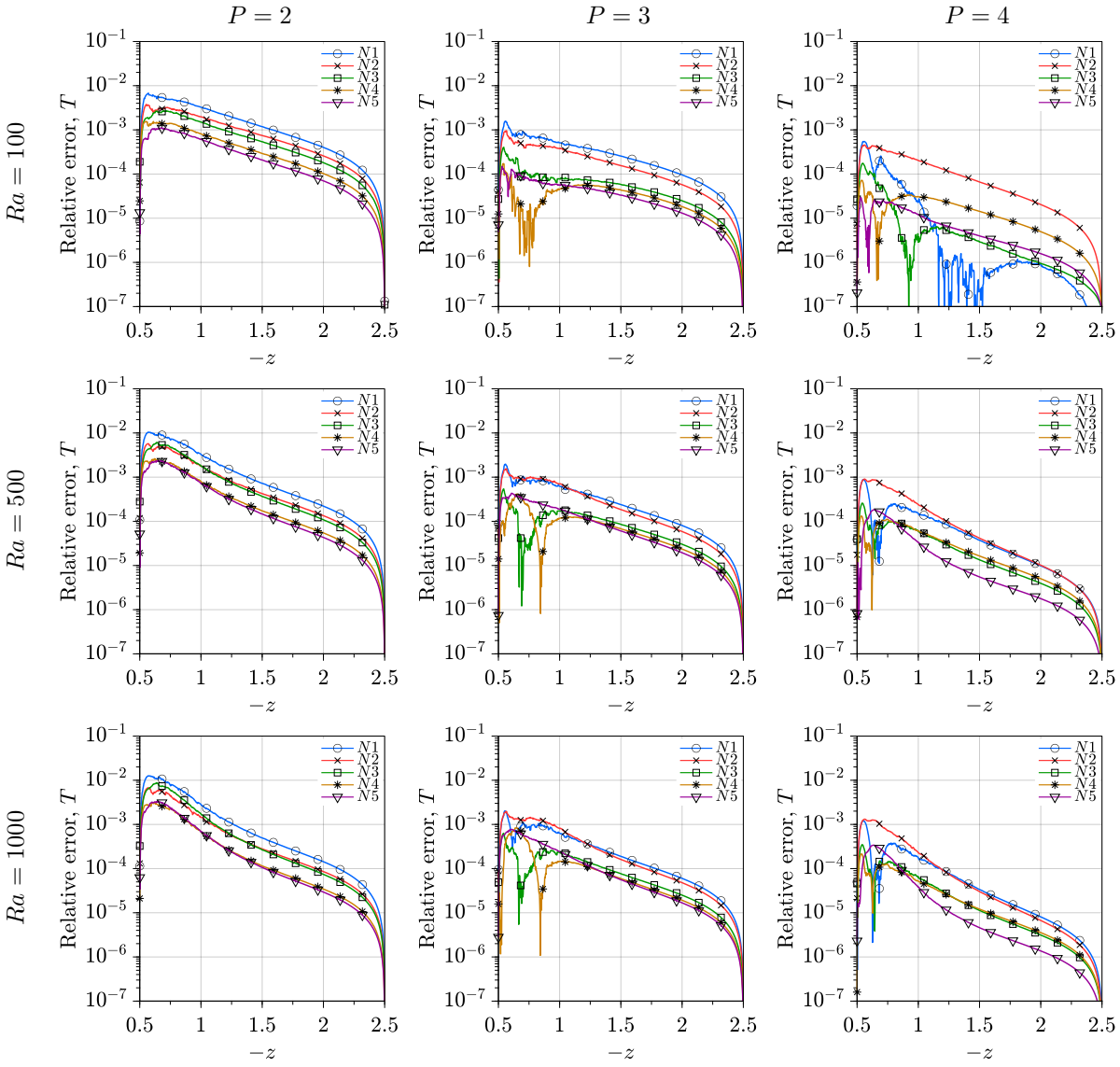


Figure B.6: spherical shell: relative error of temperature T along the lower portion of the symmetry axis z.

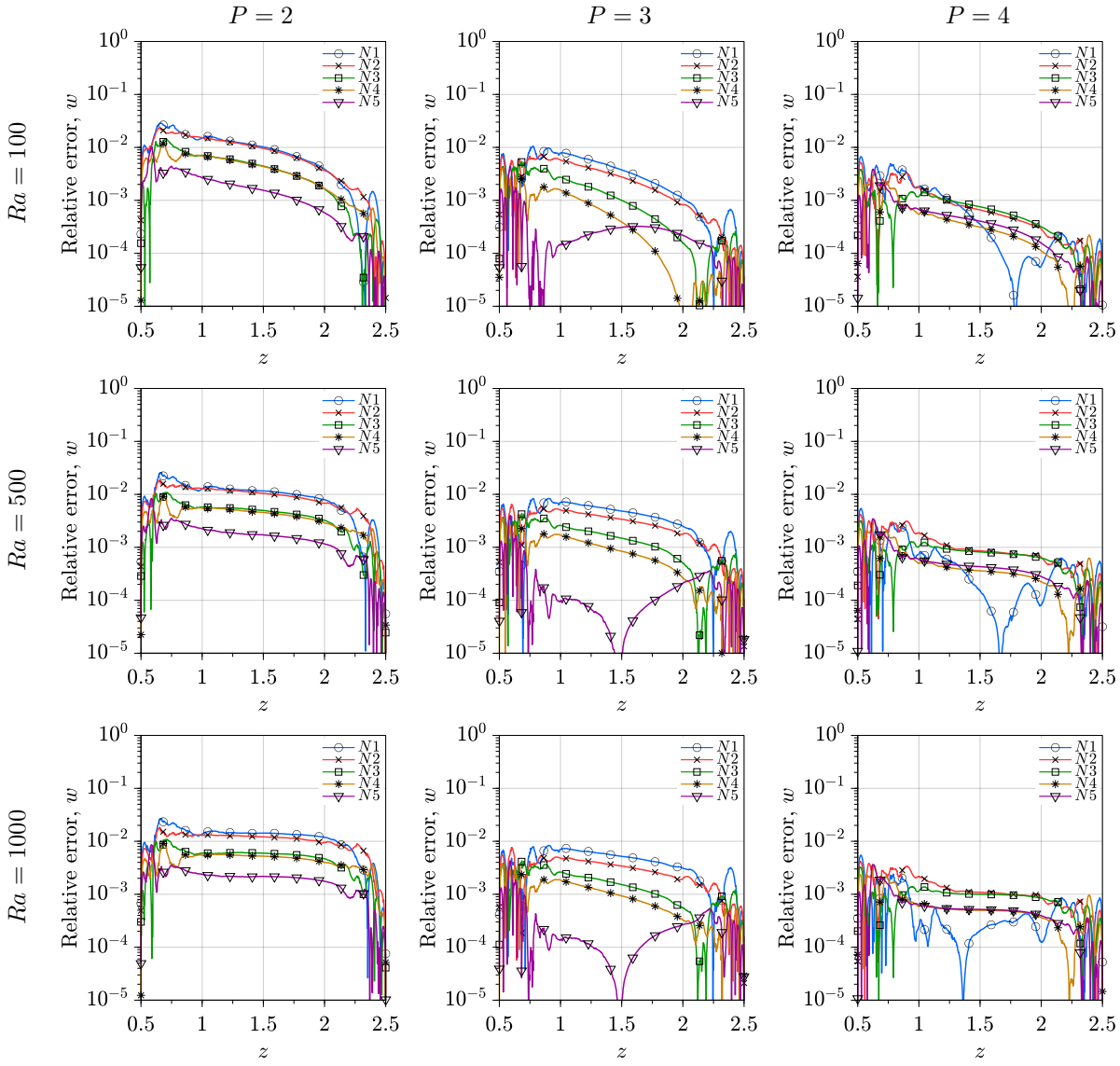


Figure B.7: spherical shell: relative error of vertical velocity w along the upper portion of the symmetry axis z.

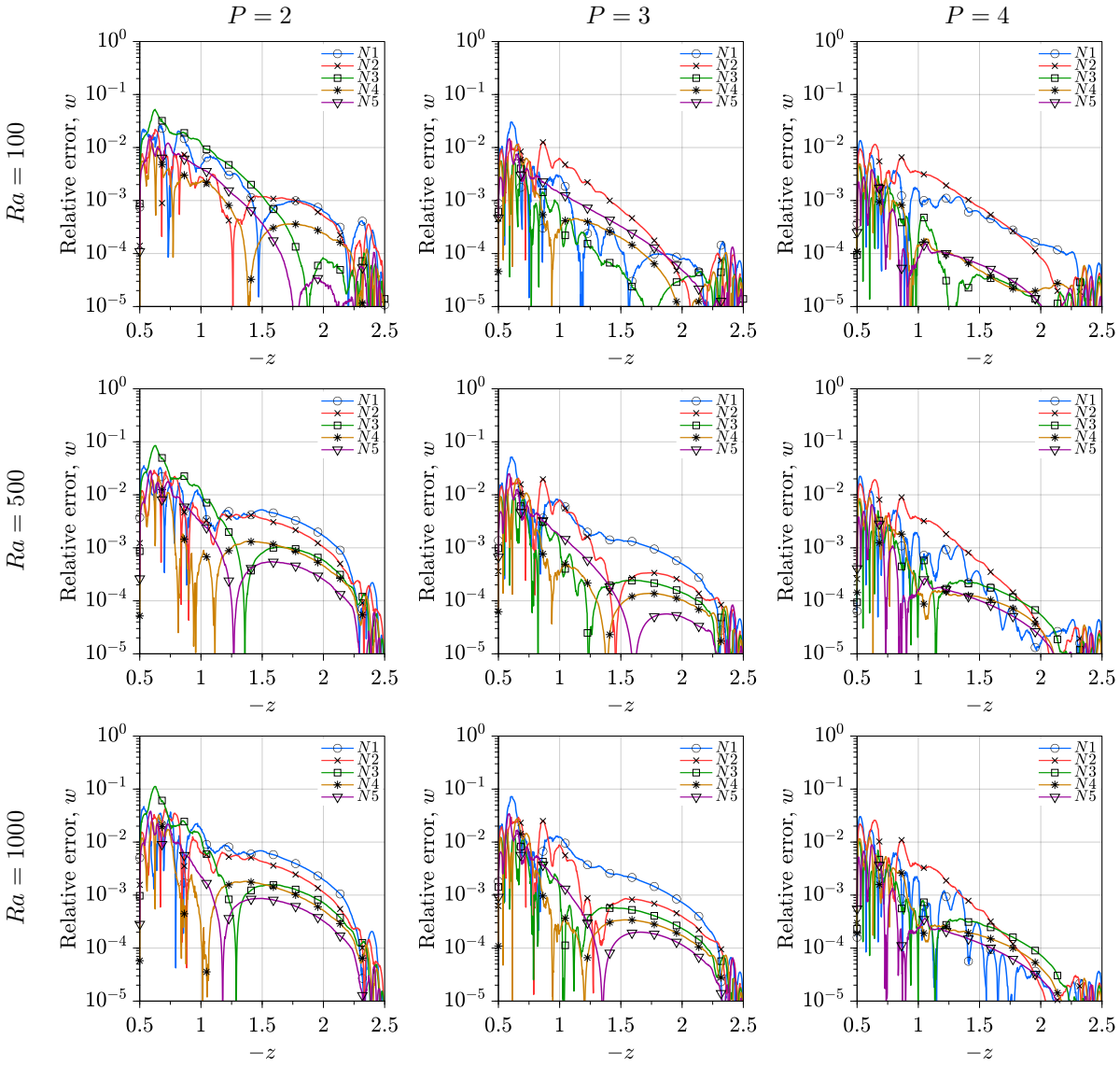


Figure B.8: spherical shell: relative error of vertical velocity w along the lower portion of the symmetry axis z.

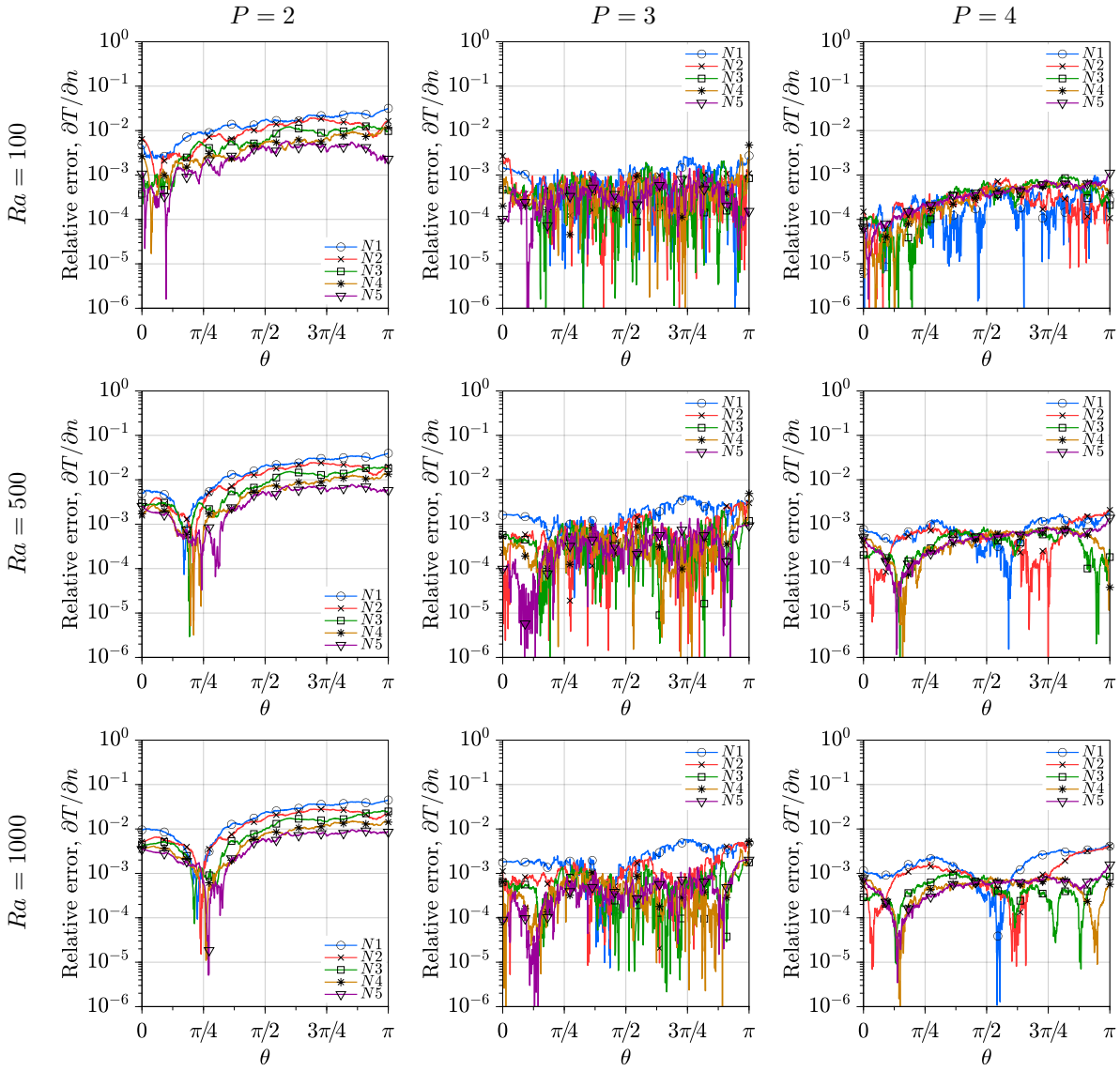


Figure B.9: spherical shell: relative error of the normal derivative along the generatrix of the inner sphere.

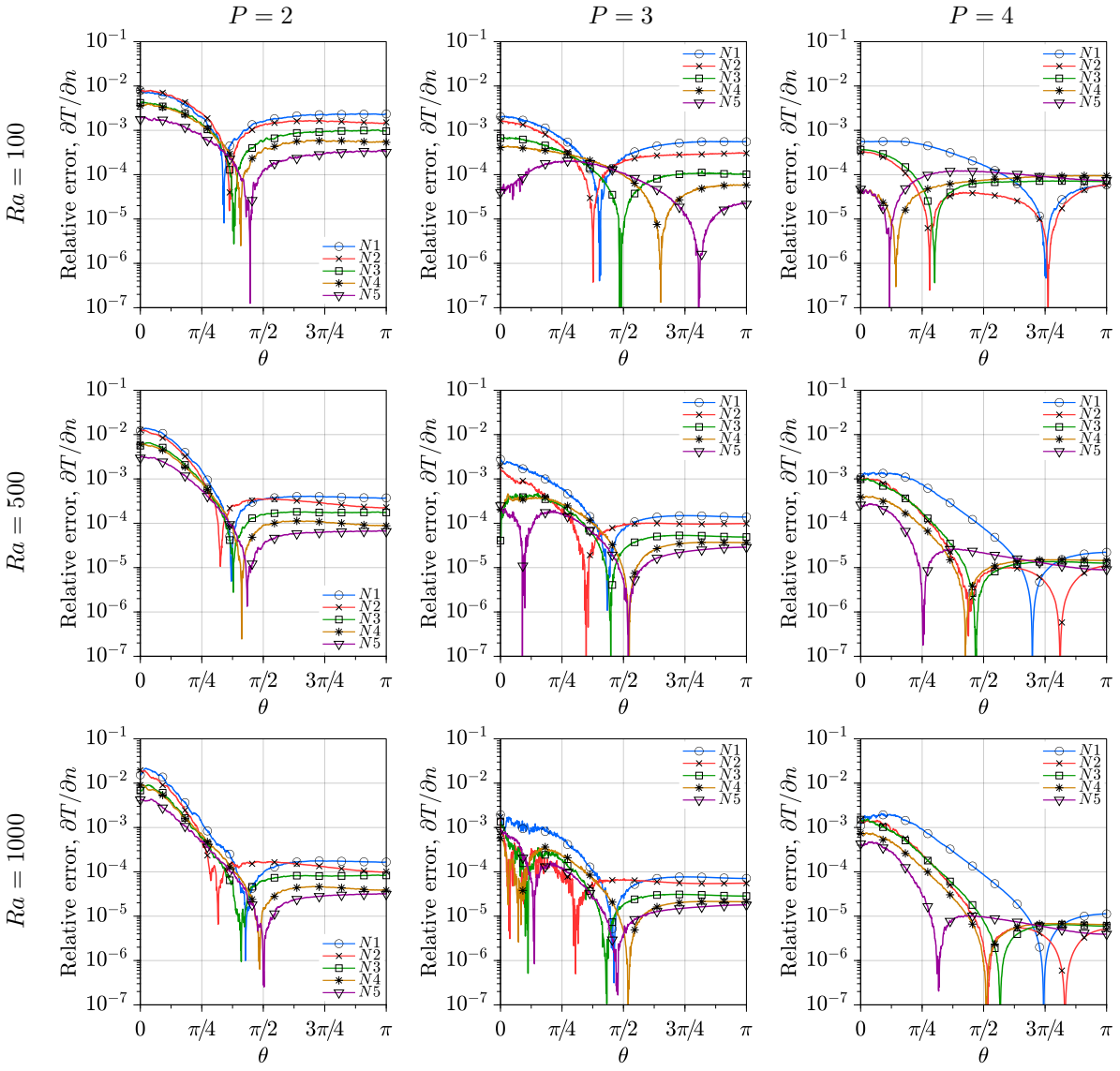


Figure B.10: spherical shell: relative error of the normal derivative along the generatrix of the outer sphere.

Bibliography

[1] L. Bacer. Fully meshless approaches for the numerical solution of partial differential equations: radial basis function finite difference method and physics-informed neural network. Master Thesis, 2023. URL `https://hdl.handle.net/20.500.12072/96637`.

[2] M. Balzer, T. Schlömer, and O. Deussen. Capacity-constrained point distributions: A variant of lloyd's method. *ACM Trans. Graph.*, 28(3), jul 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531392.

[3] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.*, 332:257–273, 2017. doi: 10.1016/j.jcp.2016.12.008.

[4] V. Bayona, N. Flyer, and B. Fornberg. On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries. *J. Comput. Phys.*, 380:378–399, 2019. doi: 10.1016/j.jcp.2018.12.013.

[5] J. F. T. Belytschko. *A first course in finite elements*. 2007. ISBN 978-0470035801.

[6] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017. doi: 10.1137/141000671.

[7] H. Bhatia, G. Norgard, V. Pascucci, and P.-T. Bremer. The Helmholtz-Hodge decomposition — A survey. *IEEE Trans. Vis. Comput. Graph.*, 19 (8):1386–1404, 2012. doi: 10.1109/TVCG.2012.316.

[8] H. Brezis and H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011.

[9] L. A. Bueno, E. A. Divo, and A. J. Kassab. A coupled localized rbf meshless/drbem formulation for accurate modeling of incompressible fluid flows. *International Journal of Computational Methods and Experimental Measurements*, 5(3):359–368, 2017. doi: 10.2495/CMEM-V5-N3-359-368.

[10] L. A. Bueno, E. A. Divo, and A. J. Kassab. Multi-scale cardiovascular flow analysis by an integrated meshless-lumped parameter model. *Boundary Elements and Other Mesh Reduction Methods*, page 181, 2018.

[11] T. Cecil, J. Qian, and S. Osher. Numerical methods for high dimensional hamilton–jacobi equations using radial basis functions. *Journal of Computational Physics*, 196(1):327–347, 2004. ISSN 0021-9991. doi: 10.1016/j.jcp.2003.11.010.

[12] C. Chen, Y. Hon, and R. Schaback. Scientific computing with radial basis functions. *Department of Mathematics, University of Southern Mississippi, Hattiesburg, MS*, 39406, 2005.

[13] Y. Chen, J. D. Lee, and A. Eskandarian. *Meshless methods in solid mechanics*, volume 9. Springer, 2006. doi: 10.1007/0-387-33368-1.

[14] A.-D. Cheng. Multiquadric and its shape parameter—a numerical investigation of error estimate, condition number, and round-off error by arbitrary precision computation. *Engineering Analysis with Boundary Elements*, 36 (2):220–239, 2012. ISSN 0955-7997. doi: 10.1016/j.enganabound.2011.07.008.

[15] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 2(1):12–26, 1967. doi: 10.1016/0021-9991(67)90037-X.

[16] A. J. Chorin. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *B. Am. Math. Soc.*, 73(6):928–931, 1967. doi: 10.1090/S0002-9904-1967-11853-6.

[17] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Math. Comput.*, 22(104):745–762, 1968. doi: 10.2307/2004575.

[18] L. Collatz. *The numerical treatment of differential equations*, volume 60. Springer Science & Business Media, 1960.

[19] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986. doi: 10.1145/7529.8927.

[20] P. C. Curtis Jr. n-parameter families and best approximation. 1959.

[21] M. Darwish and F. Moukalled. *The finite volume method in computational fluid dynamics: an advanced introduction with OpenFOAM® and Matlab®*. Springer, 2016. ISBN 978-3319348643.

[22] O. Davydov and D. T. Oanh. Adaptive meshless centres and rbf stencils for poisson equation. *Journal of Computational Physics*, 230(2):287–304, 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.09.005.

[23] S. De Marchi and R. Schaback. Stability of kernel-based interpolation. *Advances in Computational Mathematics*, 32:155–161, 2010. doi: 10.1007/s10444-008-9093-4.

[24] M. Depolli, J. Slak, and G. Kosec. Parallel domain discretization algorithm for rbf-fd and other meshless numerical methods for solving pdes. *Computers & Structures*, 264:106773, 2022. ISSN 0045-7949. doi: 10.1016/j.compstruc.2022.106773.

[25] T. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3): 413–422, 2002. ISSN 0898-1221. doi: 10.1016/S0898-1221(01)00295-4.

[26] G. Fairweather and A. Karageorghis. The method of fundamental solutions for elliptic boundary value problems. *Adv. Comput. Math.*, 9(1):69–95, 1998. doi: 10.1023/A:1018981221740.

[27] G. E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6. World Scientific, 2007.

[28] G. E. Fasshauer. Positive definite kernels: past, present and future. *Dolomites Research Notes on Approximation*, 4:21–63, 2011.

[29] G. E. Fasshauer and M. J. McCourt. Stable evaluation of gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing*, 34(2): A737–A762, 2012. doi: 10.1137/110824784.

[30] A. Fedoseyev, M. Friedman, and E. Kansa. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Comput. Math. Appl.*, 43(3-5):439–455, 2002. doi: 10.1016/S0898-1221(01)00297-8.

[31] R. Floyd. An adaptive algorithm for spatial grey scale. *SID digest, 1975*, 1975.

[32] N. Flyer, G. A. Barnett, and L. J. Wicker. Enhancing finite differences with radial basis functions: experiments on the Navier-Stokes equations. *J. Comput. Phys.*, 316:39–62, 2016. doi: 10.1016/j.jcp.2016.02.078.

[33] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *J. Comput. Phys.*, 321:21–38, 2016. doi: 10.1016/j.jcp.2016.05.026.

[34] B. Fornberg and N. Flyer. Fast generation of 2-d node distributions for mesh-free pde discretizations. *Computers & Mathematics with Applications*, 69(7):531–544, 2015. ISSN 0898-1221. doi: 10.1016/j.camwa.2015.01.009.

[35] B. Fornberg and N. Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*, volume 87. SIAM, 2015.

[36] B. Fornberg and N. Flyer. Solving pdes with radial basis functions. *Acta Numerica*, 24:215–258, 2015. doi: 10.1017/S0962492914000130.

[37] B. Fornberg and E. Lehto. Stabilization of rbf-generated finite difference methods for convective pdes. *Journal of Computational Physics*, 230(6): 2270–2285, 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.12.014.

[38] B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2008. doi: 10.1137/060671991.

[39] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers & Mathematics with Applications*, 48(5):853–867, 2004. ISSN 0898-1221. doi: 10.1016/j.camwa.2003.08.010.

[40] B. Fornberg and J. Zuev. The runge phenomenon and spatially variable shape parameters in rbf interpolation. *Computers & Mathematics with Applications*, 54(3):379–398, 2007. ISSN 0898-1221. doi: 10.1016/j.camwa.2007.01.028.

[41] B. Fornberg, G. Wright, and E. Larsson. Some observations regarding interpolants in the limit of flat radial basis functions. *Computers & Mathematics with Applications*, 47(1):37–55, 2004. ISSN 0898-1221. doi: 10.1016/S0898-1221(04)90004-1.

[42] B. Fornberg, N. Flyer, S. Hovde, and C. Piret. Locality properties of radial basis function expansion coefficients for equispaced interpolation. *IMA J. Numer. Anal.*, 28(1):121–142, 2008. doi: 10.1093/imanum/drm014.

[43] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2): 869–892, 2011. doi: 10.1137/09076756X.

[44] B. Fornberg, E. Lehto, and C. Powell. Stable calculation of gaussian-based rbf-fd stencils. *Computers & Mathematics with Applications*, 65(4):627–637, 2013. ISSN 0898-1221. doi: 10.1016/j.camwa.2012.11.006.

[45] C. J. Freitas, R. L. Street, A. N. Findikakis, and J. R. Koseff. Numerical simulation of three-dimensional flow in a cavity. *Int. J. Numer. Meth. Fl.*, 5(6):561–575, 1985. doi: 10.1002/fld.1650050606.

[46] T. Fusegi, J. Hyun, K. Kuwahara, and B. Farouk. A numerical study of three-dimensional natural convection in a differentially heated cubical enclosure. *Int. J. Heat Mass Tran.*, 34(6):1543–1557, 1991. doi: 10.1016/0017-9310(91)90295-P.

[47] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU press, 2013.

[48] R. T. Gregory and D. L. Karney. A collection of matrices for testing computational algorithms. *(No Title)*, 1969.

[49] D. P. Hardin, E. B. Saff, et al. Discretizing manifolds via minimum energy points. *Notices of the AMS*, 51(10):1186–1194, 2004.

[50] R. L. Hardy. Research results in the application of multiquadratic equations to surveying and mapping problems. *Surveying and Mapping*, 1975.

[51] H. v. Helmholtz. LXIII. On Integrals of the hydrodynamical equations, which express vortex-motion. *Lond. Edinb. Dublin philos. mag. j. sci.*, 33 (226):485–512, 1867. doi: 10.1080/14786446708639824.

[52] C.-S. Huang, C.-F. Lee, and A.-D. Cheng. Error estimate, optimal shape factor, and high precision computation of multiquadric collocation method. *Engineering Analysis with Boundary Elements*, 31(7):614–623, 2007. ISSN 0955-7997. doi: 10.1016/j.enganabound.2006.11.011.

[53] M. Jančič, J. Slak, and G. Kosec. Monomial augmentation guidelines for RBF-FD from accuracy versus computational time perspective. *J. Sci. Comput.*, 87, 4 2021. doi: 10.1007/s10915-020-01401-y.

[54] E. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—i surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, 19(8):127–145, 1990. ISSN 0898-1221. doi: 10.1016/0898-1221(90)90270-T.

[55] E. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19(8):147–161, 1990. ISSN 0898-1221. doi: 10.1016/0898-1221(90)90271-K.

[56] A. Kolar-Požun, M. Jančič, M. Rot, and G. Kosec. Oscillatory behaviour of the rbf-fd approximation accuracy under increasing stencil size. In *Computational Science – ICCS 2023*, pages 515–522. Springer Nature Switzerland, 2023. ISBN 978-3-031-36027-5. doi: 10.1007/978-3-031-36027-5_40.

[57] G. Kosec and J. Slak. Rbf-fd based dynamic thermal rating of overhead power lines. *Advances in fluid mechanics XII*, 120:255–262, 2018.

[58] G. Kosec and J. Slak. Radial basis function-generated finite differences solution of natural convection problem in 3d. In *AIP Conference Proceedings*, volume 2293. AIP Publishing, 2020. doi: 10.1063/5.0027289.

[59] E. Larsson and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic pdes. *Computers & Mathematics with Applications*, 46(5):891–902, 2003. ISSN 0898-1221. doi: 10.1016/S0898-1221(03)90151-9.

[60] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. ISSN 0021-9991. doi: 10.1016/0021-9991(92)90324-R.

[61] B. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comput. Method. Appl. M.*, 19(1): 59–98, 1979. doi: 10.1016/0045-7825(79)90034-3.

[62] X.-Y. Li, S.-H. Teng, and A. Ungor. Point placement for meshless methods using sphere packing and advancing front methods. *ICCES'00, Los Angeles, USA*, 20:25, 2000.

[63] J. Lin, Y. Zhao, D. Watson, and C. Chen. The radial basis function differential quadrature method with ghost points. *Math. Comput. Simulat.*, 173:105–114, 2020. doi: 10.1016/j.matcom.2020.01.006.

[64] G. Liu, B. B. Kee, and L. Chun. A stabilized least-squares radial point collocation method (LS-RPCM) for adaptive analysis. *Comput. Method. Appl. M.*, 195(37-40):4843–4861, 2006. doi: 10.1016/j.cma.2005.11.015.

[65] G.-R. Liu. *Meshfree methods: moving beyond the finite element method*. Taylor & Francis, 2009.

[66] G.-R. Liu and Y.-T. Gu. *An introduction to meshfree methods and their programming*. Springer Science & Business Media, 2005. doi: 10.1007/1-4020-3468-7.

[67] Y. Liu, Y. Nie, W. Zhang, and L. Wang. Node placement method by bubble simulation and its application. *Computer Modeling in Engineering and Sciences (CMES)*, 55(1):89, 2010.

[68] D. Lo, D. Young, K. Murugesan, C. Tsai, and M. Gou. Velocity-vorticity formulation for 3d natural convection in an inclined cavity by DQ method. *Int. J. Heat Mass Tran.*, 50(3):479–491, 2007. doi: 10.1016/j.ijheatmasstransfer.2006.07.025.

[69] J. C. Mairhuber. On haar's theorem concerning chebychev approximation problems having unique solutions. *Proceedings of the American Mathematical Society*, 7(4):609–615, 1956.

[70] B. Mavrič. *Meshless modeling of thermo-mechanics of low-frequency electromagnetic direct chill casting*. PhD thesis, University of Nova Gorica, Graduate School, 2017. URL `https://repozitorij.ung.si/IzpisGradiva.php?id=3108&lang=eng&prip=rul:10911498:d1`.

[71] B. Mavrič and B. Šarler. Local radial basis function collocation method for linear thermoelasticity in two dimensions. *int. J. Numer. Method. H*, 2015. doi: 10.1108/HFF-11-2014-0359.

[72] B. Mavrič and B. Šarler. Application of the RBF collocation method to transient coupled thermoelasticity. *int. J. Numer. Method. H*, 2017. doi: 10.1108/HFF-03-2016-0110.

[73] D. Miotti, R. Zamolo, and E. Nobile. A fully meshless approach to the numerical simulation of heat conduction problems over arbitrary 3d geometries. *Energies*, 14(5), 2021. ISSN 1996-1073. doi: 10.3390/en14051351.

[74] F. J. Narcowich and J. D. Ward. Norm estimates for the inverses of a general class of scattered-data radial-function interpolation matrices. *Journal of Approximation Theory*, 69(1):84–109, 1992. ISSN 0021-9045. doi: 10.1016/0021-9045(92)90050-X.

[75] F. J. Narcowich and J. D. Ward. Generalized hermite interpolation via matrix-valued conditionally positive definite functions. *Mathematics of Computation*, 63(208):661–687, 1994. doi: 10.1090/S0025-5718-1994-1254147-6.

[76] F. J. Narcowich, J. D. Ward, and H. Wendland. Sobolev error estimates and a bernstein inequality for scattered data interpolation via radial basis functions. *Constructive Approximation*, 24:175–186, 2006. doi: 10.1007/s00365-005-0624-7.

[77] W. S. J. Nocedal J. *Numerical Optimization*. Springer New York, NY, 2006. doi: 10.1007/978-0-387-40065-5.

[78] W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. *Prog. Aerosp. Sci.*, 38(3):209–272, 2002. doi: 10.1016/S0376-0421(02)00005-2.

[79] S. J. Owen. A survey of unstructured mesh generation technology. *IMR*, 239 (267):15, 1998. URL `http://ima.udg.edu/~sellares/ComGeo/OwenSurv.pdf`.

[80] S. V. Patankar. *Numerical heat transfer and fluid flow*. CRC press, 2018. doi: 10.1201/9781482234213.

[81] Y. Peng, C. Shu, and Y. Chew. A 3D incompressible thermal lattice Boltzmann model and its application to simulate natural convection in a cubic cavity. *J. Comput. Phys.*, 193(1):260–274, 2004. doi: 10.1016/j.jcp.2003.08.008.

[82] H. Power and V. Barraco. A comparison analysis between unsymmetric and symmetric radial basis function collocation methods for the numerical solution of partial differential equations. *Computers & Mathematics with Applications*, 43(3):551–583, 2002. ISSN 0898-1221. doi: 10.1016/S0898-1221(01)00305-4.

[83] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, jun 1984. ISSN 0360-0300. doi: 10.1145/356924.356930.

[84] H. Samet. An overview of quadtrees, octrees, and related hierarchical data structures. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, pages 51–68, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-83539-1_2.

[85] S. A. Sarra. Radial basis function approximation methods with extended precision floating point arithmetic. *Engineering Analysis with Boundary Elements*, 35(1):68–76, 2011. ISSN 0955-7997. doi: 10.1016/j.enganabound.2010.05.011.

[86] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995. doi: 10.1007/BF02432002.

[87] R. Schaback. Native hilbert spaces for radial basis functions i. In *New Developments in Approximation Theory: 2nd International Dortmund Meeting (IDoMAT)'98, Germany, February 23–27, 1998*, pages 255–282. Springer, 1999.

[88] R. Schaback and H. Wendland. Kernel techniques: From machine learning to meshless methods. *Acta Numerica*, 15:543–639, 2006. doi: 10.1017/S0962492906270016.

[89] V. Shankar. The overlapped radial basis function-finite difference (RBF-FD) method: a generalization of RBF-FD. *J. Comput. Phys.*, 342:211–228, 2017. doi: 10.1016/j.jcp.2017.04.037.

[90] V. Shankar, R. M. Kirby, and A. L. Fogelson. Robust node generation for mesh-free discretizations on irregular domains and surfaces. *SIAM Journal on Scientific Computing*, 40(4):A2584–A2608, 2018. doi: 10.1137/17M114090X.

[91] C. Shu, H. Ding, and K. Yeo. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 192(7):941–954, 2003. ISSN 0045-7825. doi: 10.1016/S0045-7825(02)00618-7.

[92] J. Slak. *Adaptive RBF-FD Method*. PhD thesis, University of Ljubljana, Faculty of Mathematics and Physics, 2020. URL `https://e6.ijs.si/~jslak/files/phd.pdf`.

[93] J. Slak and G. Kosec. On generation of node distributions for meshless pde discretizations. *SIAM Journal on Scientific Computing*, 41(5):A3202–A3229, 2019. doi: 10.1137/18M1231456.

[94] J. SLAK and G. KOSEC. Adaptive rbf-fd method for poisson's equation. *WIT Transactions on Engineering Sciences*, 126:149–157, 2019. doi: 10.2495/BE420131.

[95] G. L. Sleijpen and D. R. Fokkema. Bicgstab (ell) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numerical Analysis.*, 1:11–32, 1993.

[96] H. L. Stone. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM J. Numer. Anal.*, 5(3):530–558, 1968. doi: 10.1137/0705044.

[97] P. Suchde, T. Jacquemin, and O. Davydov. Point cloud generation for meshfree methods: An overview. *Archives of Computational Methods in Engineering*, 30(2):889–915, 2023. doi: 10.1007/s11831-022-09820-w.

[98] X. Sun. Scattered hermite interpolation using radial basis functions. *Linear Algebra and its Applications*, 207:135–146, 1994. ISSN 0024-3795. doi: 10.1016/0024-3795(94)90007-8.

[99] A. E. Tarwater. Parameter study of hardy's multiquadric method for scattered data interpolation. Technical report, Lawrence Livermore National Lab., CA (USA), 1985.

[100] A. I. Tolstykh. On using rbf-based differencing formulas for unstructured and mixed structured-unstructured grid calculations. In *Proceedings of the 16th IMACS world congress*, volume 228, pages 4606–4624. Lausanne, 2000.

[101] A. I. Tolstykh and D. Shirobokov. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. *Computational Mechanics*, 33(1):68–79, 2003. doi: 10.1007/s00466-003-0501-9.

[102] I. Tominec and E. Breznik. An unfitted RBF-FD method in a least-squares setting for elliptic PDEs on complex geometries. *J. Comput. Phys.*, 436: 110283, 2021. doi: 10.1016/j.jcp.2021.110283.

[103] I. Tominec, E. Larsson, and A. Heryudono. A least squares radial basis function finite difference method with improved stability properties. *SIAM J. Sci. Comput.*, 43(2):A1441–A1471, 2021. doi: 10.1137/20M1320079.

[104] E. Tric, G. Labrosse, and M. Betrouni. A first incursion into the 3d structure of natural convection of air in a differentially heated cubic cavity, from accurate numerical solutions. *Int. J. Heat Mass Tran.*, 43(21):4043–4056, 2000. doi: 10.1016/S0017-9310(00)00037-5.

[105] K. van der Sande and B. Fornberg. Fast variable density 3-d node generation. *SIAM Journal on Scientific Computing*, 43(1):A242–A257, 2021. doi: 10.1137/20M1337016.

[106] U. Varma, R. R. Suswaram, and S. Deshpande. Point distribution generation using hierarchical data structures. *ECCOMAS 2004 - European Congress on Computational Methods in Applied Sciences and Engineering*, 01 2004.

[107] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[108] S. Wakashima and T. S. Saitoh. Benchmark solutions for natural convection in a cubic cavity using the high-order time-space method. *Int. J. Heat Mass Tran.*, 47(4):853–864, 2004. doi: 10.1016/j.ijheatmasstransfer.2003.08.008.

[109] P. Wang, Y. Zhang, and Z. Guo. Numerical study of three-dimensional natural convection in a cubical cavity at high Rayleigh numbers. *Int. J. Heat Mass Tran.*, 113:217–228, 2017. doi: 10.1016/j.ijheatmasstransfer.2017.05.057.

[110] H. Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539.

[111] G. B. Wright. *Radial basis function interpolation: numerical and analytical developments*. University of Colorado at Boulder, 2003.

[112] G. B. Wright and B. Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. *Journal of Computational Physics*, 212(1):99–123, 2006. ISSN 0021-9991. doi: 10.1016/j.jcp.2005.05.030.

[113] G. B. Wright and B. Fornberg. Stable computations with flat radial basis functions using vector-valued rational approximations. *Journal of Computational Physics*, 331:137–156, 2017. ISSN 0021-9991. doi: 10.1016/j.jcp.2016.11.030.

[114] J. Yoon. Spectral approximation orders of radial basis function interpolation on the sobolev space. *SIAM Journal on Mathematical Analysis*, 33 (4):946–958, 2001. doi: 10.1137/S0036141000373811.

[115] J. Yoon. Lp-error estimates for "shifted" surface spline interpolation on sobolev space. *Mathematics of computation*, 72(243):1349–1367, 2003.

[116] R. Zamolo and E. Nobile. Two algorithms for fast 2d node generation: Application to rbf meshless discretization of diffusion problems and image halftoning. *Computers & Mathematics with Applications*, 75(12):4305–4321, 2018. ISSN 0898-1221. doi: 10.1016/j.camwa.2018.03.031.

[117] R. Zamolo and E. Nobile. Solution of incompressible fluid flow problems with heat transfer by means of an efficient rbf-fd meshless approach. *Numerical Heat Transfer, Part B: Fundamentals*, 75(1):19–42, 2019. doi: 10.1080/10407790.2019.1580048.

[118] R. Zamolo and E. Nobile. Node generation in complex 3d domains for heat conduction problems solved by rbf-fd meshless method. In *Journal of Physics: Conference Series*, volume 2116, page 012020. IOP Publishing, 2021. doi: 10.1088/1742-6596/2116/1/012020.

[119] R. Zamolo and L. Parussini. Analysis of geometric uncertainties in CFD problems solved by RBF-FD meshless method. *J. Comput. Phys.*, 421: 109730, 2020. doi: 10.1016/j.jcp.2020.109730.

[120] R. Zamolo, D. Miotti, and E. Nobile. Benchmark meshless rbf-fd solutions for 3d natural convection problems: Differentially heated cubic cavity and spherical shell. URL `https://ssrn.com/abstract=4415284`.

[121] R. Zamolo, D. Miotti, and E. Nobile. Numerical analysis of thermo-fluid problems in 3D domains by means of the RBF-FD meshless method. *J. Phys. Conf. Ser.*, 2177(1):012007, 2022. doi: 10.1088/1742-6596/2177/1/012007.

[122] R. Zamolo, D. Miotti, and E. Nobile. Accurate stabilization techniques for rbf-fd meshless discretizations with neumann boundary conditions. arXiv preprint arXiv:2207.06725, 2022. URL `https://doi.org/10.48550/arXiv.2207.06725`.

[123] R. Zamolo et al. Radial basis function-finite difference meshless methods for cfd problems. 2019. URL `https://hdl.handle.net/11368/2991045`.

[124] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu. *The finite element method for fluid dynamics*. Butterworth-Heinemann, 2013. ISBN 978-0750663229.

[125] W. Zongmin. Hermite-birkhoff interpolation of scattered data by radial basis functions. *Approximation Theory and its Applications*, 8(2):1–10, 1992. doi: 10.1007/BF02836101.