Università degli Studi di Trieste

XXXI Ciclo del Dottorato di Ricerca in
Ingegneria e Architettura

Radial Basis Function-Finite Difference Meshless Methods for CFD Problems

Settore scientifico-disciplinare: ING-IND/10 (Fisica tecnica industriale)

Dottorando:
Riccardo ZAMOLO

Coordinatore:
Prof. **Diego MICHELI**

Supervisore di tesi:
Prof. **Enrico NOBILE**

Anno Accademico 2017-2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivations

The work presented in this thesis concerns the study and the development of a numerical method belonging to the particular class of meshless methods based on Radial Basis Functions (RBF), with specific reference to the field of Computational Fluid Dynamics (CFD) for the simulation of fluid flow and heat transfer problems of engineering relevance.

Alongside the introduction and wide acceptance of traditional mesh-based methods in the field of numerical computations, meshless methods have also been introduced in order to overcome the intrinsic difficulties connected to the mesh. Such difficulties include different aspects in mesh generation, strong influence of mesh quality on the numerical results and computational issues when dealing with moving or deformable meshes.

The theoretical and practical foundations of classical mesh-based methods such as the Finite Element Method (FEM) [110] and the Finite Volume Method (FVM) [67], which are traditionally employed for the numerical simulation of engineering relevant problems, date back to the 70's, while their basic principles remained practically unchanged up to present. Such lasting history is mainly due to important numerical properties of these methods, e.g., strong mathematical foundations for FEM and conservation laws for FVM, as well as their ability to face practical problems with high robustness.

Meanwhile, the point of view in facing numerical simulations has evolved over time for different reasons, especially in industrial applications which require product simulations to be more and more reliable and, above all, fast. This challenging task can be accomplished thanks also to the continuous increase in the availability of computational resources. With an eye to the future, this evolution process should involve also the numerical methods themselves, looking for more and more effective numerical techniques which can exploit the most recent emerging technologies.

From this point of view, emerging meshless methods have proven to be excellent candidates for this purpose since they rely on a completely different and new approach, for which only a node distribution is needed. Such approach brings great advantages over standard mesh-based approaches since a great number of difficulties connected to the mesh are overcome. For example, mesh generation over complex-shaped domains can be cumbersome because of intrinsic geometrical issues, while fast and automatic node generation over complex-shaped domains has proven to be possible [33, 103, 109]. Another issue which may require important efforts is mesh quality, which is crucial for a correct and accurate solution of the problem equations. Meshless methods does not suffer such a problem since connectivity information is no longer needed. Meshless nodes can easily be moved when dealing with geometry modifications or node refinement, which are complex and onerous operations if a mesh is employed. Furthermore, such geometric flexibility make meshless methods very good candidates for robust simulations within optimization processes.

Last, but not least, meshless methods are typically easier to implement compared to mesh-based method because of the much simpler data structure required. Basically, from a geometrical point of view, only a set of nodes and a fast nearest-neighbour search are required, for which a great variety of algorithms exists.

Because of the presented advantages over mesh-based methods, meshless methods and the related topics, e.g., node generation, are receiving growing attention in the field of engineering numerical simulations, to which the presented work is intended to contribute.

1.2. Meshless methods

1.2.1. Basic principles

A great variety of meshless (or meshfree) methods have been proposed for the solution of a wide range of engineering problems, and many others continue to be proposed [68, 71].

The basic principle of meshless methods is the construction of a trial function $\tilde{u}$ which is intended to approximate the sought solution field u. Given a set of N nodes $\mathbf{x}_i$ which are suitably distributed over the physical domain Ω, the trial function $\tilde{u}$ is formally expressed by the following expansion:

$$\tilde{u}(\mathbf{x}) = \sum_{i=1}^{N} a_i \Pi_i(\mathbf{x}) \tag{1.1}$$

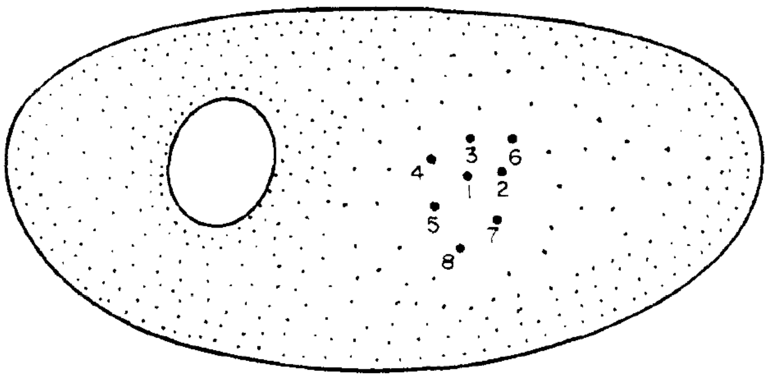
where a_i are the expansion coefficients and Π_i are the basis functions. Then, the values of the coefficients a_i have to be expressed in terms of the nodal values u_i for the sought solution field, taking also account of boundary conditions. Different techniques can be employed to express such relations $a_i(u_1, \dots, u_N)$, $i = 1, \dots, N$, including strict interpolation, least squares approximations [81], or more sophisticated techniques such as reproducing kernel [72] or kriging [61]. In this work we employed only the strict interpolation conditions for the construction of the approximant $\tilde{u}$ since it is the cheapest and straightforward choice. Nonetheless, such technique yields very good results if proper conditions are employed.

Different choices can also be made with respect to the choice of the basis functions Π_i. An important distinction depends upon the choice of global basis functions, which are non-zero over the whole domain, and local basis function, which are non-zero only over a limited part of the domain. In this last case the expansion (1.1) is thus a local expansion which depends upon a small number of nodes which constitute the local support for $\tilde{u}$. The benefits in the use of local (or localized) approaches are well known in the field of meshless simulations of engineering problems [105, 107], and therefore this approach will be used in this work.

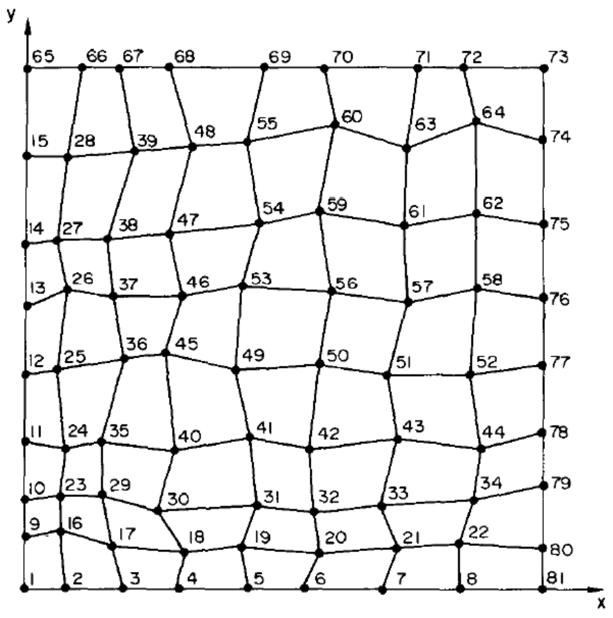
Regardless of the global or local nature of the basis function, different types of basis functions can be employed, among which polynomials and radial functions are the most used. Polynomials are typically coupled with least squares approaches, while radial basis functions are typically employed in the context of strict interpolation approaches. This is not surprising since RBFs have been widely studied and used for interpolation purposes in approximation theory [26].

Once the formal expression (1.1) for the approximation has been completely defined, it can be employed for the actual discretization of the governing equation that is intended to solve. Such discretization process, which aims to approximate a continuous partial differential equation (PDE) with a finite set of equations, can be carried out using both the weak form of the PDE or its strong form. The former case belongs to the class of Galerkin approaches which require to perform some integrals with respect to the basis functions, while the latter case belongs to the class of collocation approaches which require the evaluation of the involved derivatives at the nodes only.

Galerkin formulations are usually more stable than collocation formulations since the governing equations are spatially averaged by integration. Nonetheless, the collocation technique has been chosen for the present work because no additional geometrical construction is required, therefore it represents a "truly" meshless approach. Furthermore, its implementation is simple and straightforward, while complex governing equations can be taken into account with no additional analytic manipulation.



(a)



(b)

Figure 1.1: Meshless node distribution on a complex-shaped domain as predicted by Jensen in 1972 [51], (a). Irregular node distribution employed by Perrone and Kao in 1975 [84], (b).

1.2.2. History of local collocation meshless methods

The finite difference method (FDM) was formally introduced in 1928 by a paper of Courant, Friedrichs and Lewy [18], and it has been the first approach for the numerical simulation of physical problems. Cartesian grids were typically used, while the dealing with curved boundaries and locally refined grids was possible but troublesome. These limitations were already recognized in the early 70's when two extensions of the FDM to irregular node distributions were proposed by Jensen [51] and Perrone and Kao [84], Figure 1.1. Although the node distribution conceived by Jensen in 1972 and reported in Figure 1.1a was not used for actual calculations, it is very interesting to note that this pioneering prediction is incredibly actual after over 40 years.

The problem of the ill-conditioned or even singular differentiation matrix due to singular node arrangements, in the case of polynomial basis functions, was then improved by Liszka and Orkisz in 1980 [70] by the use of a least squares procedure for the calculation of the finite difference coefficients. Another way of avoiding ill-conditioned interpolation matrices due to singular node arrangements is the use of positive definite RBFs [26], which ensures the solvability of the interpolation problem for each set of distinct nodes.

The application of RBFs to the solution of PDEs was introduced by Kansa in 1990 [52, 53], where a global collocation approach using Multiquadrics RBFs was employed. This particular approach is often referred to as Kansa method. The Kansa method was able to successfully solve PDEs on a set of scattered nodes with great accuracy, but suffered from ill-conditioning problems because of the use of global basis functions which requires a dense matrix to be solved. This issue limited the employment of the Kansa method to the solution of small size problems only, and no practical application followed.

In the mid 90's Oñate et al. [81] and Liszka et al. [69] proposed two meshless methods based on least squares approximation with low order polynomials over a small number of local nodes, and succeeded in solving practical engineering problems. The practical success in the application of local approximants was then followed by the introduction of analogous local RBF methods at the beginning of the 2000's, starting from Lee et al. [65] in 2003.

Since then, the local RBF collocation meshless methods, also known as radial basis function-generated finite difference method (RBF-FD), has gained widespread acceptance in the community of numerical engineers and physicists because of its reliability, accuracy, flexibility and ability to face a great variety of practical problems. In the field of CFD, important and pioneering contributions are due the works of Šarler and Vertnik [89], Divo and Kassab [22–24], Kosec and Šarler [58]

1.3. Targets and outline of the work

The main target of this thesis is to develop an efficient RBF-FD meshless scheme for the numerical simulation of 2D/3D fluid flow problems with heat transfer of engineering relevance. This main goal has guided all the work that has been conducted during the PhD, which comprehends many of the different aspects that contribute to the design of a functional numerical method in its entirety. Each of these aspects has been studied and efficient algorithms/procedures have been proposed. Following the philosophy of meshless methods, the implementation of these procedures is kept as simple as possible.

Following the order of the meshless simulation chain, we can subdivide the present work into three main topics: node generation, RBF-FD discretization and solution phase.

The aspect of node generation has been thoroughly studied because it is obviously one of the key points in making meshless methods really competitive with mesh-based methods. Two novel node generation processes based on node-repel refinement have been developed, showing the capability of simple, efficient and automatic node generation on complex 2D/3D domains starting from a defined spacing function.

The analysis of the RBF-FD discretization, which is the main ingredient of the developed meshless scheme, has been conducted in order to give important insights about the accuracy, stability and computational efficiency of the RBF meshless discretization from an engineering point of view. Such properties are studied considering all the different elements which characterize the RBF-FD approach, e.g., number of local nodes, degree of the polynomial augmentation, influence of boundary conditions and RBF shape factor. The Poisson equation and the advection equation have been considered as 2D/3D model problems in this phase. In the case of the incompressible Navier-Stokes equations with a primitive variables formulation, particular attention is given to the development of stable RBF-FD discretizations. This issue is more and more important as the Reynolds or Rayleigh flow numbers increase, which are characteristic features of most fluid flow problems of engineering relevance.

The solution phase is not less important than the previous aspects, since the possibility to deal with large size problems, i.e., large number of meshless nodes, is also crucial in the development of numerical methods intended to solve practical problems. For this purpose, two novel multicloud techniques based on multigrid principles have been developed and successfully employed for the acceleration of the convergence in the iterative solution of the linear systems arising from the RBF-FD discretization in the case of a 2D Poisson equation. In the case of the incompressible 2D/3D Navier-Stokes equations, particular attention is given to the efficient solution of the set of equations in their unsteady formulation in order to provide an effective time integration procedure.

The presented meshless approach is therefore composed by the coupling of all the previous elements which thus represent a complete framework for any kind of meshless simulation.

This meshless framework is ultimately employed for the solution of various isothermal and non-isothermal fluid flow problems over different 2D and 3D domains. The important features of these calculations are the use of a high number of nodes, e.g., $N > 10^5$, the employment of non-trivial node distributions, e.g., very small nodal spacing at the walls, and possible applications to complex-shaped domains. These are all essential features for the accurate solution of typical engineering problems, representing an important challenge posed by academic and industrial applications. The work presented in this thesis is therefore intended to contribute to face this challenge through the use of flexible, robust and efficient RBF-FD meshless approaches.

1.4. Implementation details

The implementation of the presented procedures and algorithms has been done through MATLAB® environment using MATLAB linked MEX functions which are compiled from C source code for the computational expensive tasks, on a modern laptop equipped with an Intel® i7 2.6GHz processor with 4 cores. Multi-core parallelism is achieved by using OpenMP® API for the C source code. Most of the remaining MATLAB operations are natively parallelized on all available cores by MATLAB. Some examples of C code for the nearest neighbour search and node-repel refinement are listed in Appendix F.

Chapter 2

Node generation

Node generation is clearly one of the key elements of meshless methods. The development of efficient and flexible node generation algorithms is therefore crucial for the practical success of meshless methods over mesh-based methods [79] for which a very good degree of maturity is reached in mesh generation [40, 82]. The aim of node generation is to create a suitable distribution of nodes which are properly scattered over the domain in accordance to some requirements. These requirements are typically less strict than the ones imposed for mesh generation, making node generation a potentially easier task than mesh generation.

Different strategies have been employed for node generation within meshless approaches. These strategies include uniform and random distributions [22, 58, 105], transfinite interpolation (TFI) and elliptic node generation (ENG) [44]. Other applications employed standard mesh generators to obtain a high quality polygonization of the domain for which only the vertices were kept as meshless nodes [56, 81].

In recent years the general trend moved to the development and employment of node generation algorithms which are specifically designed for meshless applications only. This change in perspective can be traced back to the works of Löhner and Oñate [73, 74] and Lee [64] where the use of polygonization was reduced of even abandoned in favour of point/node-based techniques.

More recent node generation algorithms employ a node-repel refinement technique which is capable of generating high quality node distributions starting from a prescribed spacing function [33, 103, 109]. These techniques have proven to be flexible and effective tools in practical meshless applications [47, 57, 76]. For this reason the repel-based approach has been chosen as the key element in the developing of novel node generation techniques which will be presented and analyzed in this chapter.

In addition to meshless applications, another interesting use of node generation algorithms is stippling, which is a technique used for the approximation of greyscale

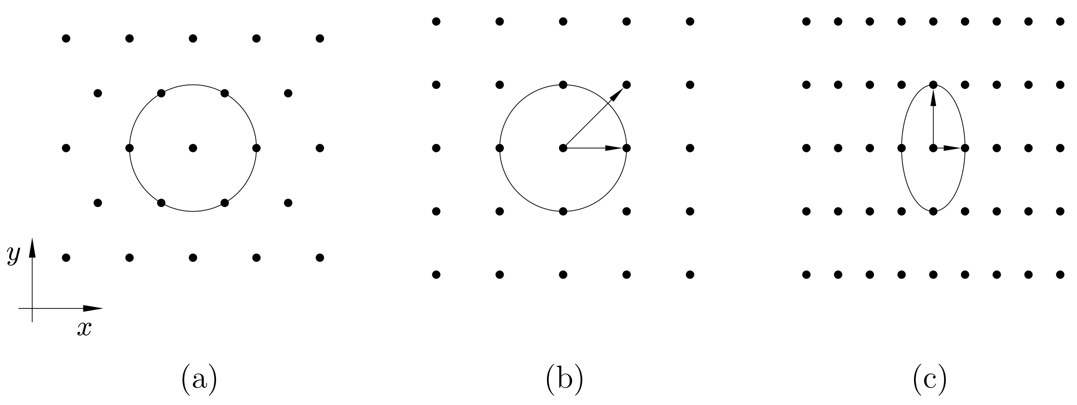


Figure 2.1: 2D examples of node arrangements: (a) hexagonal, (b) uniform cartesian, (c) non-uniform cartesian.

images through the arrangement of scattered points in 2D. At the end of this chapter some visual test cases are presented for this side application in order to show how the proposed node generation algorithms can also be used for an efficient and accurate halftoning approximation of greyscale images.

2.1. Definitions for the node generation problem

In this thesis isotropic node distributions are considered only. Isotropic node distributions are characterized by equal nodal spacing along all spatial directions, while the nodal spacing is variable in space allowing locally refined node distributions.

Three significant 2D examples are shown in Figure 2.1, where a hexagonal, a uniform cartesian and a non-uniform cartesian node distributions are depicted. The hexagonal distribution is isotropic since the nodal spacing is equal in all directions. The uniform cartesian distribution is not strictly isotropic since the nodal spacing along x and y directions is smaller than the spacing along the 45 degrees directions. The non-uniform distribution is strictly non-isotropic since the spacing along x is different from the spacing along y.

The distinction between isotropic (hexagonal) and not strictly isotropic (cartesian) is not unique since it depends upon the choice of neighbouring nodes: the cartesian distribution can be considered isotropic when 4 neighbouring nodes are considered. However, since the cartesian arrangement is not stable within a node-repel process, i.e., it does not maximize the number of nodes per area, only locally hexagonal or nearly-hexagonal node distributions will be considered as 2D isotropic in this context. For such arrangements, each node has 6 nearest neighbours. In

3D the corresponding highest density node arrangement follows the face-centered cubic (FCC) or the hexagonal close-packed (HCP) lattices where each node has 12 neighbours. Each of these 3D arrangements can be obtained by packing planes filled with 2D hexagonal node arrangements.

The dependency of the nodal spacing upon the position is given by a spacing function $s(\mathbf{x})$ which defines the point-wise nodal spacing. An isotropic node distribution $X(s)$ inside the domain Ω can be formally defined as a function of the prescribed spacing function $s(\mathbf{x})$ as follows:

$$X(s) = \{\mathbf{x}_i \in \Omega : \text{nodal spacing satisfies } s;\ i = 1, \dots, N\} \tag{2.1}$$

Let us define the nodal density $\delta(\mathbf{x})$ as the number of nodes per unit area/volume:

$$\delta(\mathbf{x}) = \lim_{\substack{\mathcal{P}\to\mathbf{x} \\ k\to+\infty}} \frac{\#nodes(X(s/\kappa), \mathcal{P})}{\kappa^D \mu(\mathcal{P})} \tag{2.2}$$

where κ is a positive real number, $\mathcal{P}$ is a portion of Ω, $\mu(\mathcal{P})$ is the area/volume of $\mathcal{P}$, $\#nodes(X, \mathcal{P})$ gives the number of nodes of distribution X lying inside $\mathcal{P}$ and D is the number of dimensions, e.g., $D = 3$ in 3D. In the limit (2.2) the notation $\mathcal{P} \to \mathbf{x}$ means $\mu(\mathcal{P}) \to 0$ with $\mathbf{x} \in \mathcal{P}$, while $\mathcal{P}$ must satisfy $\mu(\mathcal{P}) \geq c\kappa^{-\alpha}$ for some constants $\alpha < D$ and $c > 0$.

The definition of the spacing function s is obtained by using the definition of the nodal density in the case of isotropic node distributions:

$$\delta(\mathbf{x}) = \frac{number\ of\ nodes}{unit\ area/volume} = \frac{2}{\zeta s^D(\mathbf{x})} \tag{2.3}$$

where $\zeta = \sqrt{3}$ in 2D and $\zeta = \sqrt{2}$ in 3D ($\zeta = 2$ in 1D).

The explicit definition of the spacing function is therefore:

$$s(\mathbf{x}) = \sqrt[D]{\frac{2}{\zeta \delta(\mathbf{x})}} \tag{2.4}$$

Given a prescribed spacing function $s(\mathbf{x})$, the corresponding prescribed number of nodes $N_\mathcal{P}$ contained in a portion $\mathcal{P}$ of the domain is therefore given by the integration of Eq. (2.3) over $\mathcal{P}$:

$$N_\mathcal{P} = \int_\mathcal{P} \delta(\mathbf{x})\ \mathrm{d}\mathcal{P} = \int_\mathcal{P} \frac{2}{\zeta s^D(\mathbf{x})}\ \mathrm{d}\mathcal{P} \tag{2.5}$$

where $N_\mathcal{P}$ is non-integer in general.

For a generic node distribution $\tilde{X} = \{\tilde{\mathbf{x}}_i \in \Omega;\ i = 1, \ldots, N\}$ the error $E_\mathcal{P}$ between the actual number of nodes in $\mathcal{P}$ and the prescribed number $N_\mathcal{P}$ is simply given by:

$$E_\mathcal{P} = \#nodes(\tilde{X}, \mathcal{P}) - N_\mathcal{P} \tag{2.6}$$

The error $E_\mathcal{P}$ has an average meaning when the size of $\mathcal{P}$ is larger than the maximum spacing function s_M encountered in $\mathcal{P}$, i.e., $\sqrt[D]{\mu(\mathcal{P})} \gg s_M$. When the size of $\mathcal{P}$ is comparable with the prescribed spacing $s(\mathcal{P})$, $E_\mathcal{P}$ assumes the meaning of quantization error between the integer number of nodes in $\mathcal{P}$ and the prescribed non-integer number $N_\mathcal{P}$ (nodal quantization error).

Given a domain Ω with boundary Γ and a prescribed spacing function $s(\mathbf{x})$, the node generation problem consists in creating an isotropic node distribution of the type of Eq. (2.1) which is conformal to the boundary Γ, i.e., nodes are required to lie also on the boundary whilst fulfilling the prescribed spacing function.

2.2. Initial node positioning

2.2.1. Some remarks on the node-repel technique

The developed node generation algorithms are composed by two phases: an initial node placing phase and a consequential iterative refinement phase. The initial phase creates a distribution whose node spacing matches the prescribed spacing function $s(\mathbf{x})$ except for some high-frequency error that is smoothed out in the refinement phase which improves the quality of the distribution. In the refinement phase, which is based upon an iterative node-repel algorithm, nodes move according to the mutual radial repulsion forces of the nearest neighboring nodes.

The initial node placing phase is required because the node-repel algorithm efficiently smooths out only the high-frequency component of the error between the actual spacing of the node distribution and the prescribed spacing $s(\mathbf{x})$, while any low-frequency component in space requires a very high number of iterations to be significantly reduced [33].

This property is highlighted by Figure 2.2, where two initial 1D node distributions with high-frequency and low-frequency errors are subjected to node-repel iterations using 2 neighboring nodes. The initial node distributions are obtained from a uniform distribution which is perturbed with small random deviations in the first case, while in the second case the perturbations are larger and are given by a positive linear function along x. The spacing function is $s(x) = 1/(N - 1)$.

In these 1D cases the spacing deviation $\sigma[\Delta x]$ quantifies the global deviation between the prescribed nodal spacing $s(x)$ and the actual nodal spacing:

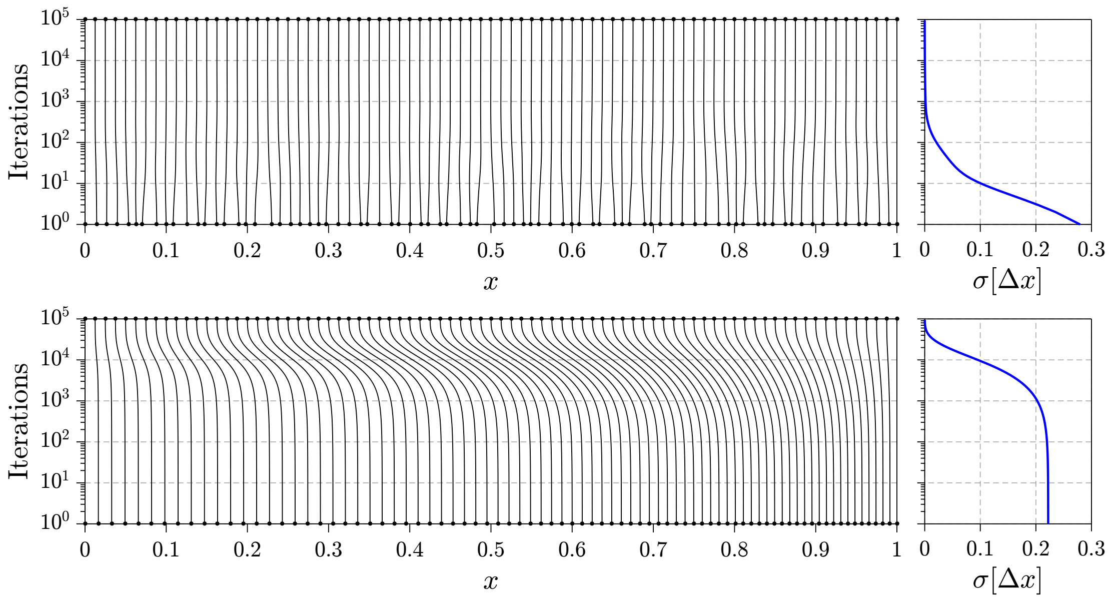


Figure 2.2: Evolution of the node-repel phase for 1D node distributions with an initially high-frequency (top) and low-frequency (bottom) error in space in the case of a constant spacing function and $N = 80$ nodes.

$$\sigma[\Delta x] = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N-1} \Big[\Delta x_i / s(x_i) - 1\Big]^2} \tag{2.7}$$

where Δx_i is the spacing between nodes x_i.

In the first case (high-frequency error) the deviation $\sigma[\Delta x]$ rapidly decrease to 0 within a small number of repel iterations. In the second case (low-frequency error) the deviation stagnates for a large number of repel iterations before a significant decrease.

Therefore it is very important to develop some method which is capable of generating an initial node distribution which satisfies the prescribed spacing function $s(\mathbf{x})$ in average sense, i.e., the error $E_{\mathcal{P}}$ defined by Eq. (2.6) should be sufficiently small for any portion $\mathcal{P} \in \Omega$ which is larger than the spacing function. The quantization error, i.e., $E_{\mathcal{P}}$ for $\mathcal{P}$ with size comparable to the spacing function, can be high since this high-frequency error is efficiently reduced in the node-repel refinement phase. Local deviations from isotropy are also allowed in this phase.

Two different node placing algorithms have been developed and are presented as follows.

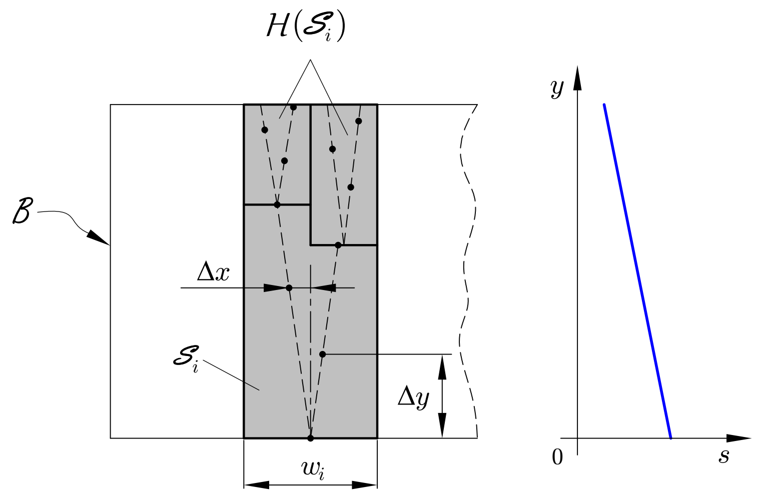


Figure 2.3: Node placing through the VS algorithm in the case of a decreasing spacing function s along y.

2.2.2. Vertical strips (VS) algorithm

This algorithm has been developed for 2D cases only. The smallest rectangle $\mathcal{B}$ bounding domain Ω is considered, with axis-aligned sides. This rectangle is partitioned into vertical strips $\mathcal{S}_i$ with width $w_i = \max s$ in $\mathcal{S}_i$. Then the algorithm proceeds by filling each vertical strip $\mathcal{S}_i$ in a recursive way using the VS algorithm whose pseudocode is given by Algorithm 1, where:

- GenerateNodesVS($Nodes, \mathcal{S}, s$) fills the whole strip $\mathcal{S}$ using spacing function s, appending new nodes to the existing $Nodes$;

- LinearDistribution($Nodes, \mathcal{S}, s$) fills the strip $\mathcal{S}$ as long as $w_i/2 < s \leq w_i$, where w is the width of $\mathcal{S}$. The filling takes place with an alternating horizontal nodal offset $\Delta x = \pm(w_i - s)/2$ from the vertical midline of the strip, while the vertical offset between nodes Δy is chosen in order to follow the prescribed node density $\delta = 2/(\sqrt{3}s^2) = 1/(w_i \Delta y) \Rightarrow \Delta y = \sqrt{3}s^2/(2w_i)$;

- $\mathcal{H}(\mathcal{S})$ are the two vertical substrips of the unfilled portion of $\mathcal{S}$;

- ParentStrip($\mathcal{S}$) is the unfilled portion of the parent strip of $\mathcal{S}$.

The function LinearDistribution($\cdot, \mathcal{S}, s$) fills the strip $\mathcal{S}$ in such a way that the error $E_{\mathcal{P}}$ between the actual number of placed nodes and the prescribed number in the portion $\mathcal{P} = \mathcal{S}$ is very small if the spacing function is sufficiently smooth.

Algorithm 1 Vertical strips (VS) node generation inside strip $\mathcal{S}$

```
Input: strip $\mathcal{S}$, prescribed spacing function $s$
Output: node distribution $Nodes$ inside $\mathcal{S}$
1: function GenerateNodesVS($Nodes$,$\mathcal{S}$,$s$)
2:     LinearDistribution($Nodes$,$\mathcal{S}$,$s$)
3:     if $s < w/2$ then
4:         for each $SubStrip \in \mathcal{H}(\mathcal{S})$ do
5:             GenerateNodesVS($Nodes$,$SubStrip$,$s$)
6:     else
7:         $ParentStrip \leftarrow$ ParentStrip($\mathcal{S}$)
8:         GenerateNodesVS($Nodes$,$ParentStrip$,$s$)
9: end function
```

A graphical representation of the working principle for this algorithm is depicted in Figure 2.3 in the case of a linear spacing function s decreasing with y. Starting from the bottom of the strip $\mathcal{S}_i$, nodes are placed in an alternated fashion along y according to the spacing function: the smaller the spacing, the higher the horizontal offset from the strip vertical midline. When the spacing s is less than half of the strip width w_i, the remaining upper portion of the strip is subdivided

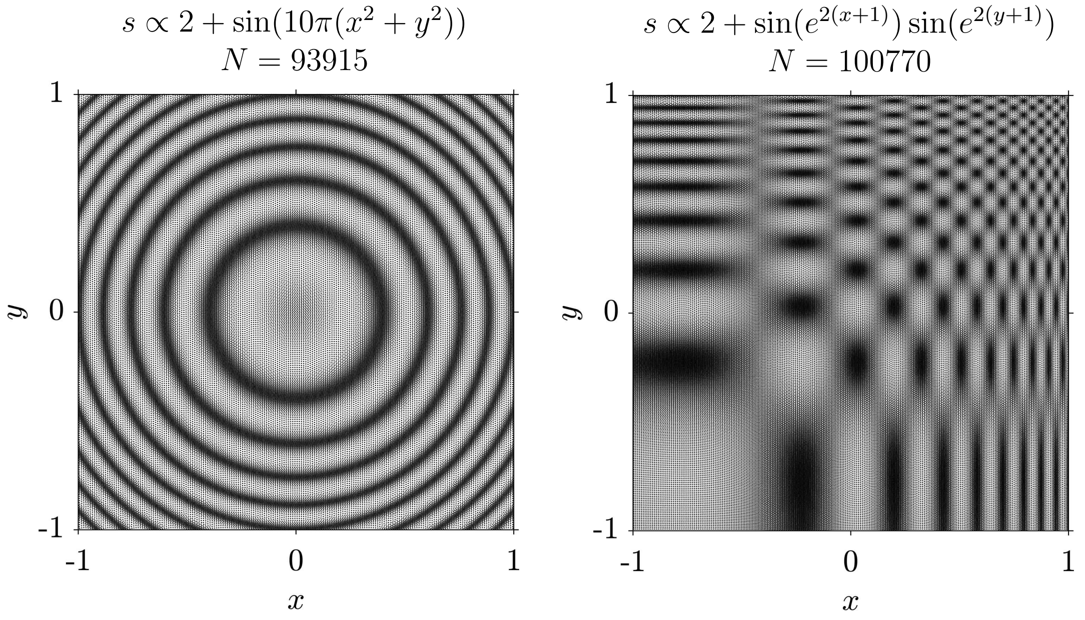


Figure 2.4: Examples of node distributions generated by the VS algorithm in the case of simple spacing functions s.

into two substrips $\mathcal{H}(\mathcal{S}_i)$ for which the VS algorithm is recursively called.

The VS algorithm is very easy to implement and gives very good results in the case of sufficiently smooth spacing functions s. The computational cost and the memory requirement are both linear in the number of nodes N. Nevertheless, if s has steep variations or discontinuities, the VS algorithm in its easiest implementation may produce artifacts and is not robust in general.

Some examples of node distributions generated by the VS algorithm are shown in Figure 2.4 and Figure 2.5 in the case of simple analytic spacing functions $s(x, y)$ in the square $[-1, 1]^2$. The examples of Figure 2.4 employ a high number of nodes, i.e., $N \approx 10^5$, and spacing functions with some high-frequency components. The examples of Figure 2.5 employ a low number of nodes, i.e., $N \approx 5000$, and spacing functions with very smooth variations in order to better evaluate the graphical patterns due to the working principle of the VS algorithm.

The computing times for two limit cases represented by the highly-varying function $s \propto 2 + \sin(e^{2(x+1)}) \sin(e^{2(y+1)})$ of Figure 2.4 and the smooth function $s \propto 1 + 5(x^2 + y^2)$ of Figure 2.4 are 0.16s/(mln nodes) and 0.059s/(mln nodes), respectively, using a C implementation and 1 core. The VS algorithm is thus extremely fast and therefore represents a very efficient approach for 2D node generation in the case of sufficiently smooth spacing functions.

The extension to 3D cases is made possible by considering vertical parallelepipeds as volume strips which are filled by the VS algorithm, as shown in Figure 2.6. The projection of the bounding box $\mathcal{B} = [-1, 1]^3$ onto the $x - y$ plane is partitioned into a cartesian grid according to the maximum value of the spacing function, and the corresponding vertical volume-strips along z are considered. The VS algorithm is then applied to each of these vertical strips which can be recursively partitioned into 4 smaller parallelepipeds when the spacing function drops below $w/2$, where w is the side length of the initial vertical strip. However, since the VS algorithm is not robust in its easiest implementation, its 3D extension has not been considered.

2.2.3. Quadtree/Octree algorithms

The 2D quadtree (QT) and 3D octree (OT) algorithms [86, 87] are widely used space partitioning techniques which are characterized by high efficiency and robustness. Quadtrees and octrees proceed by recursive partitioning of the space into four or eight equally sized child boxes as shown in Figure 2.7 where l is the QT level.

In the context of meshless methods, QTs and OTs can be employed for fast and robust node generation. An example of quadtree node generator for 2D meshless discretizations can be found in [102]. This task can be accomplished during the recursive space partitioning process by considering a single node placement at the

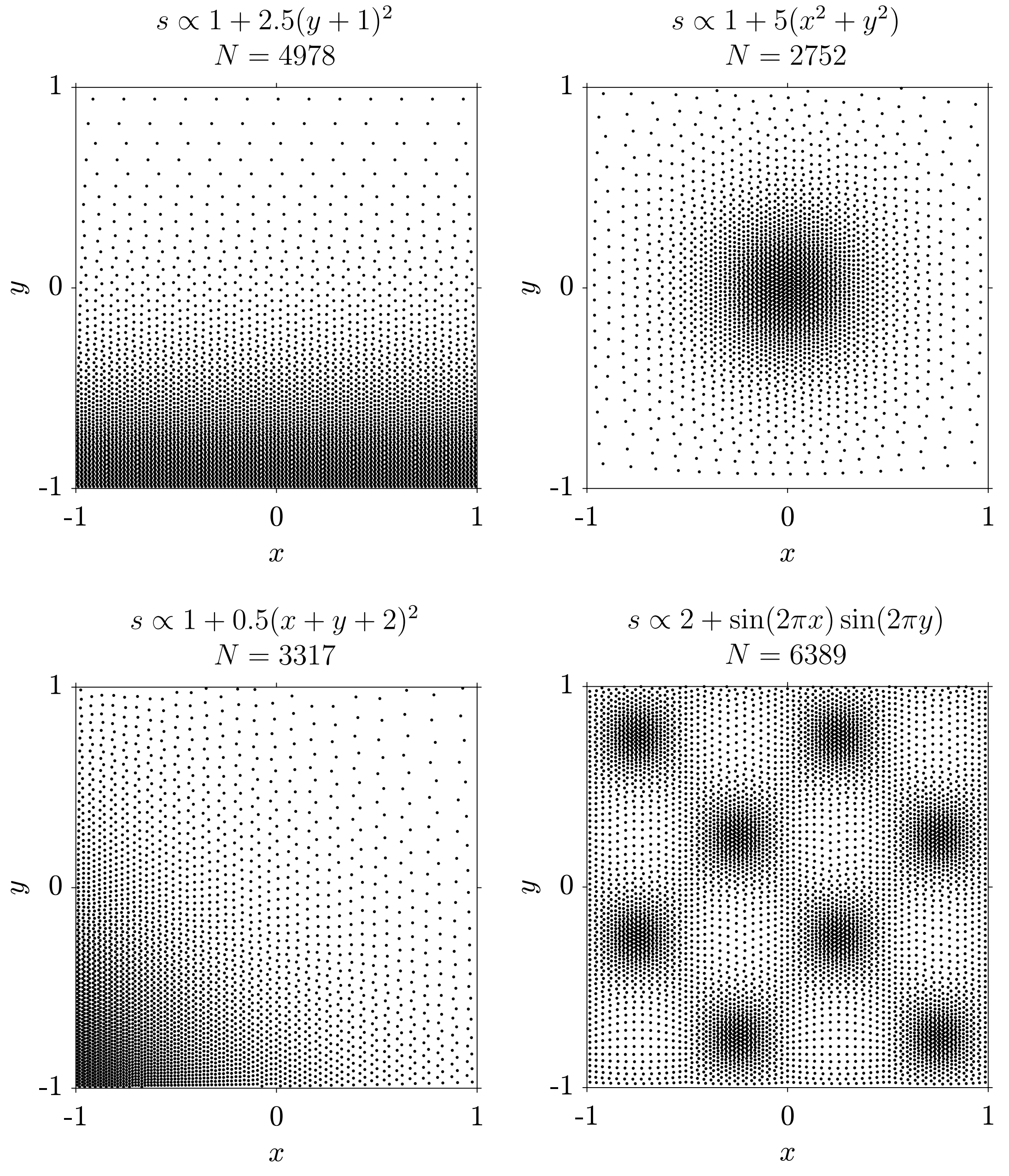


Figure 2.5: Examples of node distributions generated by the VS algorithm in the case of simple spacing functions s.

center of the boxes when their size w_l at level l is comparable to the prescribed spacing function. These boxes where node are placed represent therefore the leaf nodes in the tree (leaf boxes). The pseudocode for this algorithm is given by Algorithm 2, where:

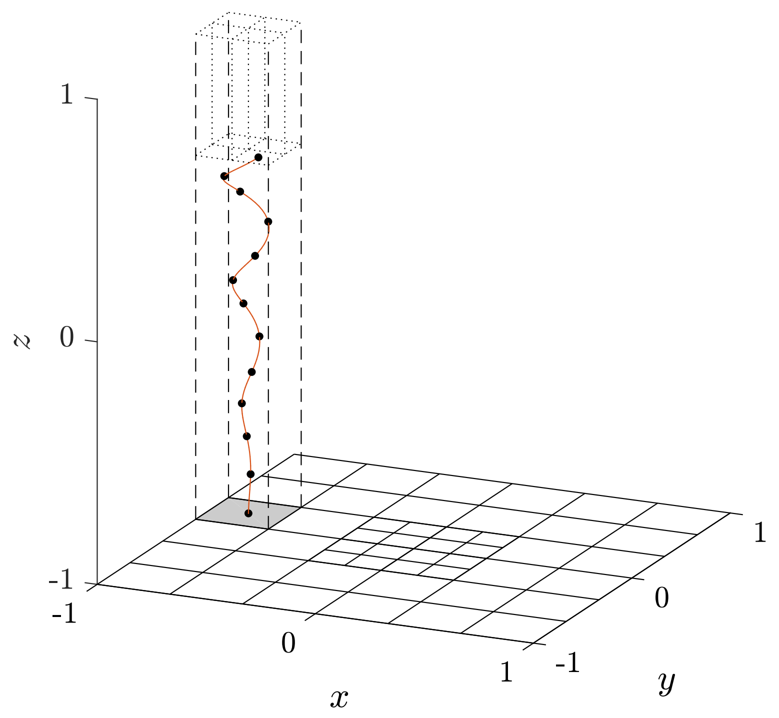


Figure 2.6: Possible working principle of the VS algorithm for 3D cases.

- GenerateNodesTree($Nodes, \mathcal{P}, s, m$) fills the whole box $\mathcal{P}$ using the QT/OT algorithm and spacing function s, appending new nodes to the existing $Nodes$. Up to m nodes are inserted into the leaf boxes;
- InsertNodes($Nodes, \mathcal{P}, n$) inserts n nodes in the box $\mathcal{P}$;
- $N_{\mathcal{P}}$ is the prescribed number of nodes inside the box $\mathcal{P}$;
- $\mathcal{C}(\mathcal{P})$ are the four/eight child boxes of $\mathcal{P}$ for QT and OT, respectively.

The choice of placing one single node at the center of the leaf boxes thus corresponds to the case $m = 1$ in Algorithm 2. An example of the application of this simple algorithm to 2D node generation is given in Figure 2.8(a) where the spacing function decreases going towards the center of the box $\mathcal{B}$. The node generation is obtained by calling GenerateNodesTree on the smallest square/cubic box $\mathcal{B}$ (root box) bounding the boundary Γ.

This simple version of the QT/OT algorithm for node generation suffers from one main problem: high nodal quantization errors. Nodal quantization error (see Eq. (2.6)) is due to the fact that only finite nodal densities $\delta = 1/w_l^D$ can be

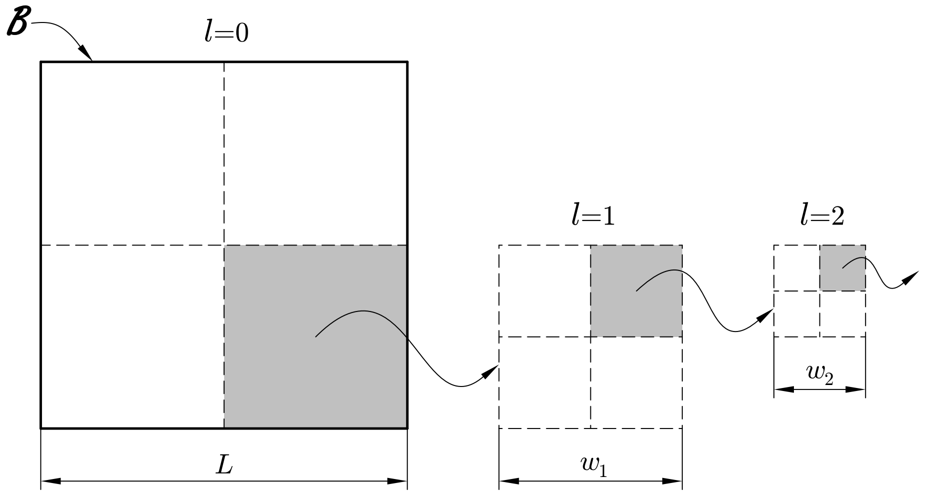


Figure 2.7: Recursive space partitioning in 2D using quadtree.

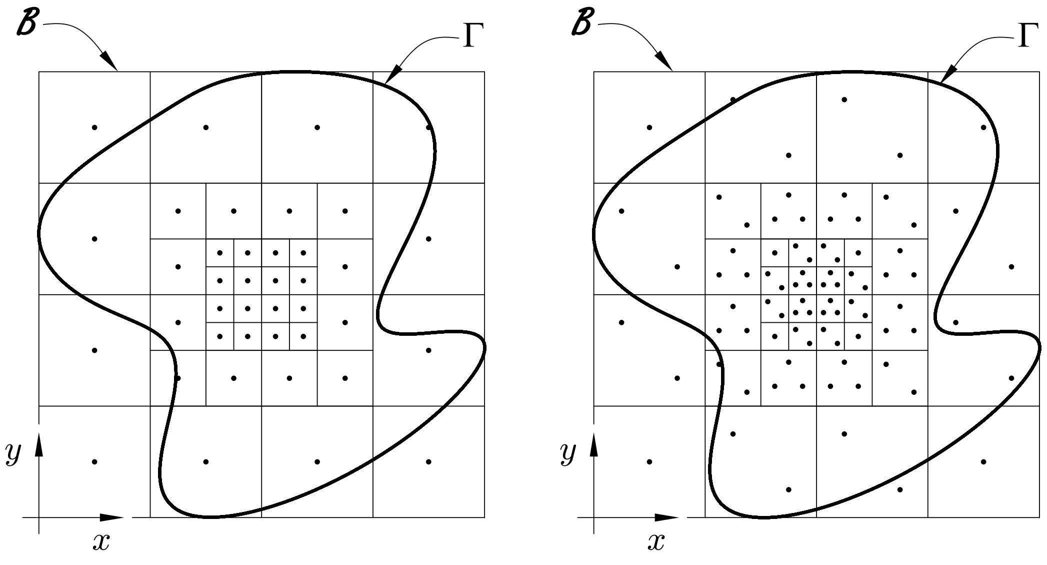
Algorithm 2 Quadtree/octree (QT/OT) node generation

Input: box $\mathcal{P}$, prescribed spacing function s, maximum node insertions m
Output: node distribution $Nodes$ inside $\mathcal{P}$

```
1: function GenerateNodesTree($Nodes$,$\mathcal{P}$,$s$,$m$)
2:     if $N_\mathcal{P} < m + 1/2$ then
3:         InsertNodes($Nodes$,$\mathcal{P}$,$\lfloor N_\mathcal{P} \rceil$)
4:     else
5:         for each $ChildBox \in \mathcal{C}(\mathcal{P})$ do
6:             GenerateNodesTree($Nodes$,$ChildBox$,$s$,$m$)
7: end function
```

obtained, i.e., one node is placed over boxes which have fixed size $w_l = L/2^l$, therefore nodal quantization errors are always present. Nodal quantization errors, which are defined at the small scales of the nodal spacing, can also imply large-scale errors, i.e. large differences between the prescribed number of nodes and the actual number of node placed over large areas of the domain. This can occur when nodal quantization errors accumulates over large portions of the domain as shown in the 1D example of Figure 2.9(a). The QT algorithm in 1D is simply obtained by recursive bisection of a segment, and therefore we will refer to it as bisection tree (BT) algorithm.

In Figure 2.9(a) the prescribed spacing function $s(x) = w_5(1 + x/L)$ increases linearly from $s(0) = w_5$ to $s(L) = 2w_5 = w_4$, where L is the length of the domain and $w_l = L/2^l$ is the size of the 1D boxes $\mathcal{P}$ at level l. For this spacing function the

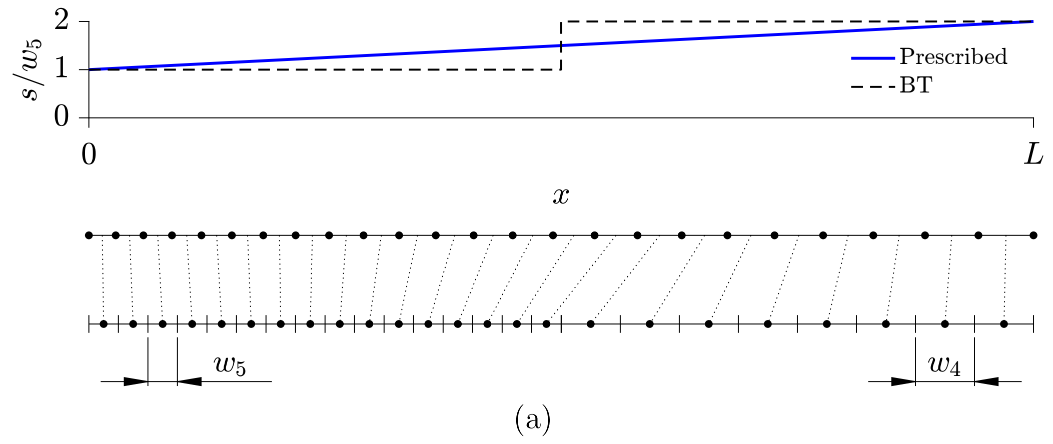


(a) (b)

Figure 2.8: 2D node generation through the quadtree algorithm: (a) basic application; (b) modified version with multiple node insertions.

BT algorithm generates small boxes with width w_5 for the left half of the domain, while the width $w_4 = 2w_5$ doubles in the right half. The width of the boxes thus matches the prescribed spacing function at both ends $x = 0, L$, while it is smaller and larger than the prescribed spacing function in the left and right halves of the domain, respectively, as reported by the graphs of Figure 2.9(a). This obviously results in a larger number of nodes than the prescribed number in the left half, while the opposite situation occurs in the right half, as reported by the diagram of Figure 2.9(b) which shows the error $E_{\mathcal{P}}$ per unit length for each box $\mathcal{P}$ at each level l.

The BT node distribution thus shows a large low-frequency component, i.e., at level $l = 1$, for the error $E_{\mathcal{P}}$. As showed previously, the low-frequency components of $E_{\mathcal{P}}$ are slowly reduced by the node-repel algorithm that is intended to follow this initial node placing phase. In fact, the nodes from the BT distribution in Figure 2.9(a) have to sustain large displacements in order to reach the exact distribution given in the same figure. Such displacements can be larger than the spacing function itself for nodes near $x = L/2$, while the node-repel algorithms is efficient only in the case of small displacements which are comparable to the spacing function. Furthermore, the more nodes are employed, the larger the displacements if compared to the spacing function, and therefore the more iterations the node-repel algorithm will require to reach the equilibrium.



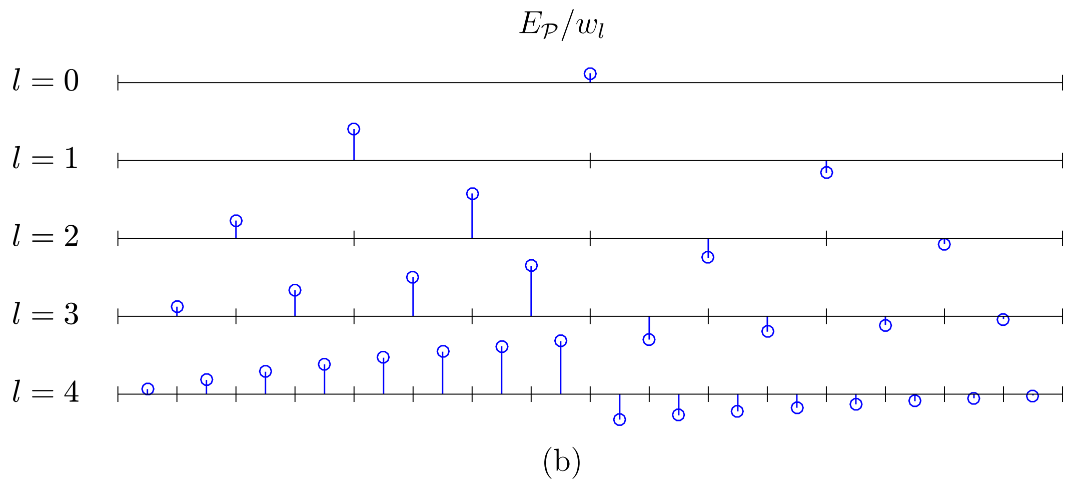


Figure 2.9: (a) Example of large differences between 1D node distributions obtained from the same spacing function s: exact (top) and BT generated (bottom). (b) Error per unit length $E_{\mathcal{P}}/w_l$ between the actual number of BT nodes and the prescribed number for each box $\mathcal{P}$ at each level l.

The analysis of the diagrams of Figure 2.9(b) for $l = 1, 2$ also reveals that in this case the excess of nodes in the left half of the domain is not balanced by the lack of nodes in the right half, and the total number of nodes generated by the BT algorithm is slightly larger than the exact value, i.e., 24 BT nodes vs. $N_{\mathcal{P}} = w_5^{-1} \log 2 = 22.18$ exact nodes.

Multiple node insertions

The previous analysis highlights the necessity of an improved QT/OT node placing algorithms in order to eliminate or reduce the nodal quantization errors. A simple technique which is capable of reducing such problem is the employment of multiple node insertions within the leaf boxes, according to the spacing function. The only modification in the QT/OT Algorithm 2 involves the function InsertNodes which is modified for the suitable insertion of $m \geq 1$ nodes, according to the spacing function s.

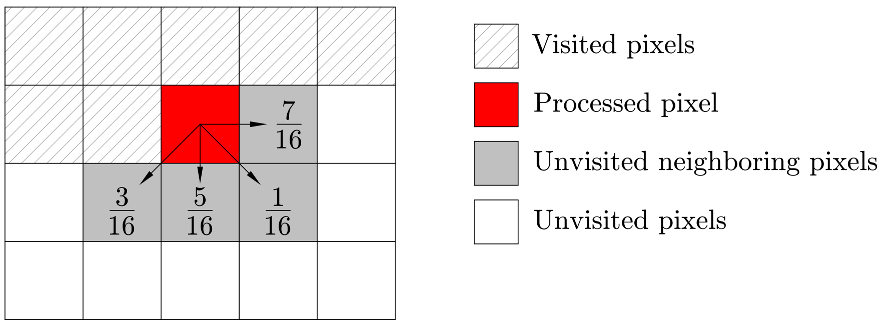
With this modification the realizable nodal densities grow from $\delta = 1/w_l^D$ to $\delta = 1/w_l^D, \ldots, m/w_l^D$, thus increasing in number. The problem of nodal quantization is apparently overcome by using a large number m of nodal insertions, but this choice simply shifts the nodal quantization problem to the choice of correct nodal insertions. However, small numbers $1 \leq m < 2^D$ for multiple nodal insertions can bring significant improvements. In this case these few nodes are distributed uniformly in the leaf box. An example of the application of this technique to a 2D problem is depicted in Figure 2.8(b) where $m = 3$ nodal insertions are employed: each leaf box can therefore contain up to $m = 3$ nodes. Variations in the nodal spacing are smoother and less steep than the ones obtained with the original QT algorithm.

Dithering

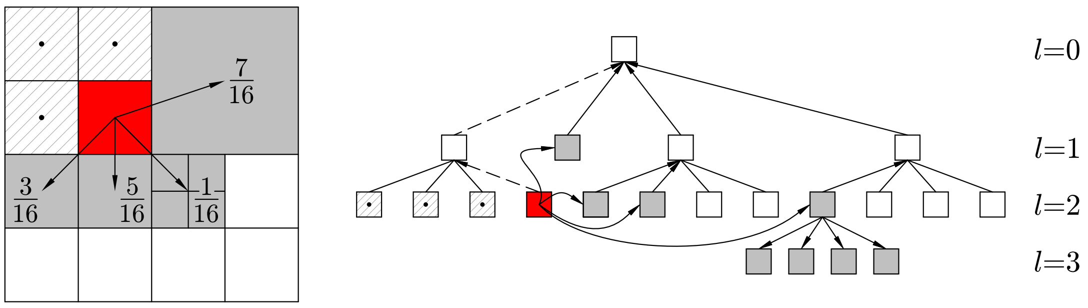
Besides the technique of multiple node insertions reduces the nodal quantization error, the problem is still present, requiring unnecessary iterations in the node-repel refinement phase. In order to overcome this issue, the dithering algorithm used in signal and image processing can be used [30, 96]. The aim of dithering is to diffuse quantization errors in space in order to maintain an almost zero quantization error on average. In the context of node generation, this translates into an almost zero error $E_{\mathcal{P}}$ for length scales from L (size of the domain) to the small scales close to the spacing function. This operation is performed by properly diffusing $E_{\mathcal{P}}$, i.e., the local excess/lack of nodes, over the domain.

The Floyd-Steinberg algorithm [30], which is a widely used dithering technique for 2D image processing, is employed for diffusing nodal quantization error. This algorithm scans the pixels by rows and distributes the quantization error onto the unvisited neighbouring pixels using the scheme depicted in Figure 2.10(a). Figure 2.11 shows an example of the application of the Floyd-Steinberg dithering algorithm to the colour quantization of a greyscale image with a black and white palette: the quantization without an appropriate error diffusion leads to large-scale errors.

The application of the Floyd-Steinberg algorithm to the node generation using the QT/OT algorithm with multiple node insertions gives rise to the dithered quadtree/octree (DQT/DOT) algorithm, whose pseudocode is given by Algorithm 3. Two new elements are required: a numerosity variable M_s and a QT/OT dithering correction function DitheringTree.



(a)



(b)

Figure 2.10: Floyd-Steinberg dithering algorithm: (a) original scheme for image processing; (b) adaptation to the quadtree data structure.

The numerosity variable M_s, which uses the QT/OT hierarchical data structure, retains the number of nodes for each box in the tree. The value of M_s for any box equals the sum of the values of M_s for each of its child boxes (summation property), if any, and therefore the value of M_s for the root box $\mathcal{B}$ equals the total number of nodes to be generated.

The dithering correction function DITHERINGTREE diffuses the nodal quantization error $E_{\mathcal{P}}$ by modifying the values of numerosity M_s for the unvisited neighboring boxes using the Floyd-Steinberg algorithm, which has been adapted to the QT/OT hierarchical data structure as showed in Figure 2.10(b). The (red) processed box, i.e., where node insertion(s) is occurring (leaf box), transfers the opposite of its quantization error to the four (grey) unvisited neighbouring boxes using the Floyd-Steinberg coefficients shown in Figure 2.10(b). In order to preserve the summation property for M_s, each of these (grey) neighbouring boxes then recursively distributes the received fraction of the error to its child boxes in equal parts, until the leaf boxes are reached. Analogously, the same fraction of the error is recursively added to the parent box for each of these (grey) neighbouring

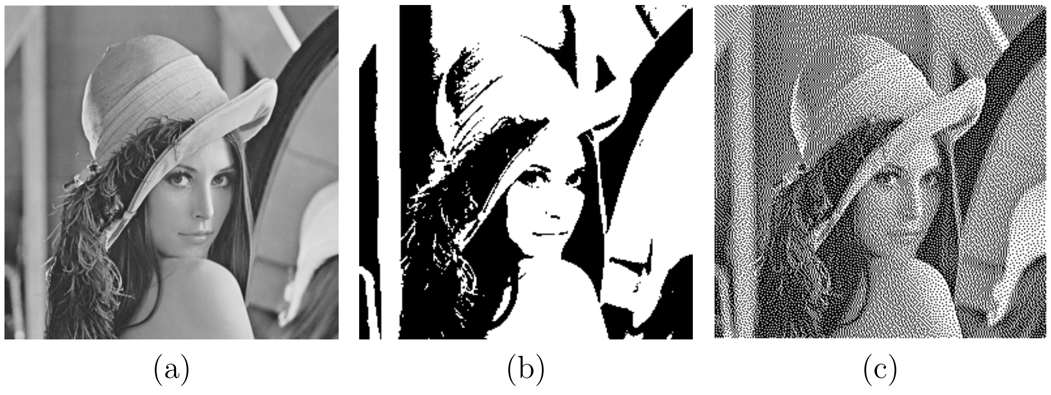


Figure 2.11: Quantization of image colours: (a) original greyscale image; (b) black and white palette; (c) black and white palette with Floyd-Steinberg dithering (256×256 pixels).

box, until the root box is reached. To conclude, the same recursive procedure is also performed by the (red) processed box for the summation of the quantization error towards its parent boxes in order to guarantee the perfect conservation of numerosity M_s, i.e., the total number of nodes and the summation property are maintained after each node insertion.

Algorithm 3 Dithered quadtree/octree (DQT/DOT) node generation

```
Input: box $\mathcal{P}$, numerosity $M_s$, maximum node insertions $m$
Output: node distribution $Nodes$ inside $\mathcal{P}$
 1: function GenerateNodesDitheredTree($Nodes$,$\mathcal{P}$,$M_s$,$m$)
 2:     $n \leftarrow M_s(\mathcal{P})$
 3:     if $n < m + 1/2$ then
 4:         InsertNodes($Nodes$,$\mathcal{P}$,$\lfloor n \rceil$)
 5:         $QuantizationError \leftarrow \lfloor n \rceil - n$
 6:         DitheringTree($M_s$,$QuantizationError$,$\mathcal{P}$)
 7:     else
 8:         for each $ChildBox \in \mathcal{C}(\mathcal{P})$ do
 9:             GenerateNodesDitheredTree($Nodes$,$ChildBox$,$M_s$,$m$)
10: end function
```

The numerosity M_s is initialized by traversing the tree from the root box and assigning the prescribed number of nodes $N_{\mathcal{P}}$ for each box until the leaf boxes are reached, i.e., when $N_{\mathcal{P}} < m + 1/2$. The node generation is then obtained by the application of the actual DQT/DOT algorithm, whose pseudocode is given by Algorithm 3, where:

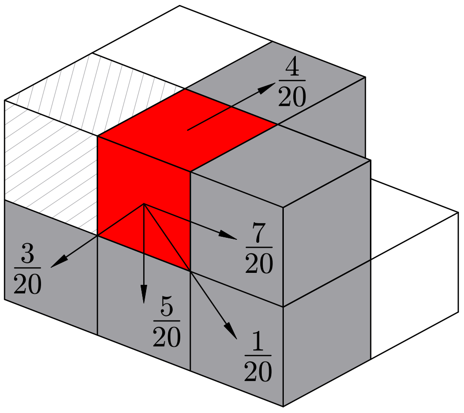


Figure 2.12: Coefficients for 3D Floyd-Steinberg error diffusion through octree boxes at the same level.

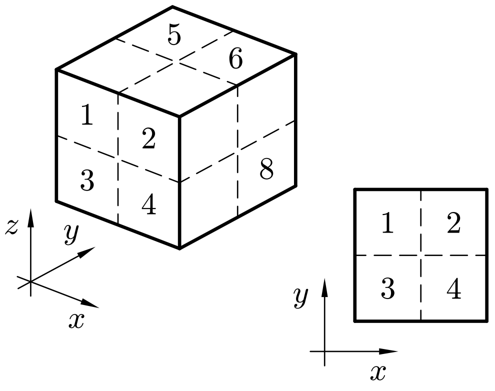


Figure 2.13: Order for the visits of child boxes in 3D octree (left) and 2D quadtree (right).

– GenerateNodesDitheredTree($Nodes$, $\mathcal{P}$, M_s, m) fills the whole box $\mathcal{P}$ using the DQT/DOT algorithm and numerosity M_s, appending new nodes to the existing $Nodes$. Up to m nodes are inserted into the leaf boxes;

– DitheringTree(M_s,$QuantizationError$,$\mathcal{P}$) diffuses the $QuantizationError$ of box $\mathcal{P}$ across the tree data structure of numerosity M_s, as depicted in Figure 2.10(b).

In 3D cases, the Floyd-Steinberg coefficients for the diffusion of the quantization error are shown in Figure 2.12, where the coefficient 4/20 accounts for the error diffusion from the (red) processed box to the (grey) unvisited neighbour box sharing a face along the $+y$ direction.

The dithered versions of both QT and OT algorithms must follow the order reported in Figure 2.13 when visiting each of the child boxes in order to diffuse the quantization error onto unvisited boxes only, similarly to the processing order in the standard Floyd-Steinberg dithering algorithm which proceeds by rows of pixels.

Two comparisons between the proposed node generation algorithms for the initial node placing phase are reported in Figure 2.14 and Figure 2.15 in the case of two smooth spacing functions in the square $[-1, 1]^2$.

In Figure 2.14 the spacing function s is proportional to $1 + (y + 1)/2$, therefore the spacing is constant along horizontal lines. In this case the VS algorithm produces very smooth nodal spacing since s depends upon y only, which is the direction of the vertical strips in the VS algorithm. The QT node distribution shows evident bands due to the limited number of realizable nodal densities, for which nodes appear to be grouped into large regions with constant nodal densities. The spacing function is therefore highly discontinuous with steep variations moving from one region to the other. This issue is partially addressed by the use of $m = 3$ multiple node insertions, for which the bands are less evident but still present.

Ultimately, the coupling of $m = 3$ multiple node insertions with dithering for the QT algorithm (DQT) solves the problem almost completely: the regions with constant spacing are discontinued by the periodical appearance of small zones with different spacing which guarantee an almost exact nodal density on average. The fulfillment of the prescribed nodal density for the DQT algorithm is also highlighted by the total number of generated nodes $N = 9452$ which is very close to the exact value $N_{\mathcal{P}} = \int_0^1\int_0^1 \delta(x, y)\ \mathrm{d}x\ \mathrm{d}y = 9459.31$. The number of nodes generated by the VS algorithm, i.e., $N = 9522$, is also close to the exact value since the VS algorithm also fulfills the prescribed nodal densities almost exactly. Because of the large quantization errors occurring with the QT algorithm without dithering or multiple node insertions, the number of generated nodes $N = 10432$ is very far from the exact number.

The same considerations can be made for the comparison depicted in Figure 2.15, where the spacing function is proportional to $\propto 1 + (x^2 + y^2)$. Although the spacing function depends also upon x, the VS algorithm performs well anyway, because s is sufficiently smooth along both x and y. The QT node distribution shows two distinct regions and therefore it is again unacceptable, while the application of multiple node insertions and dithering bring remarkable improvements.

Although the VS algorithm produced very smooth node distributions which are also aesthetically pleasant in the previous 2D examples, we point out that this approach is not robust in the case of non-smooth spacing functions and therefore has not been employed any further, and the DQT algorithm with $m = 2^D - 1$ multiple insertions has been chosen as the best candidate for the initial node positioning phase.

Computing times and memory requirements

Table 2.1 reports the complexity, memory requirements and computing times for QT/OT and DQT/DOT algorithms implemented in C.

The computing times refer to the generation of $N \approx 10^6$ nodes for the spacing function $s \propto 3 + 2\sin(\pi x)\sin(\pi y)$ in the square $[-1, 1]^2$ for the 2D case, while $s \propto 3 + 2\sin(\pi x)\sin(\pi y)\sin(\pi z)$ in the cube $[-1, 1]^3$ for the 3D case. The employment of the dithering technique doubles the computing times in both 2D and 3D cases. Anyway, such computing times are almost negligible when compared to the computing times required by the successive node-repel refinement phase, therefore

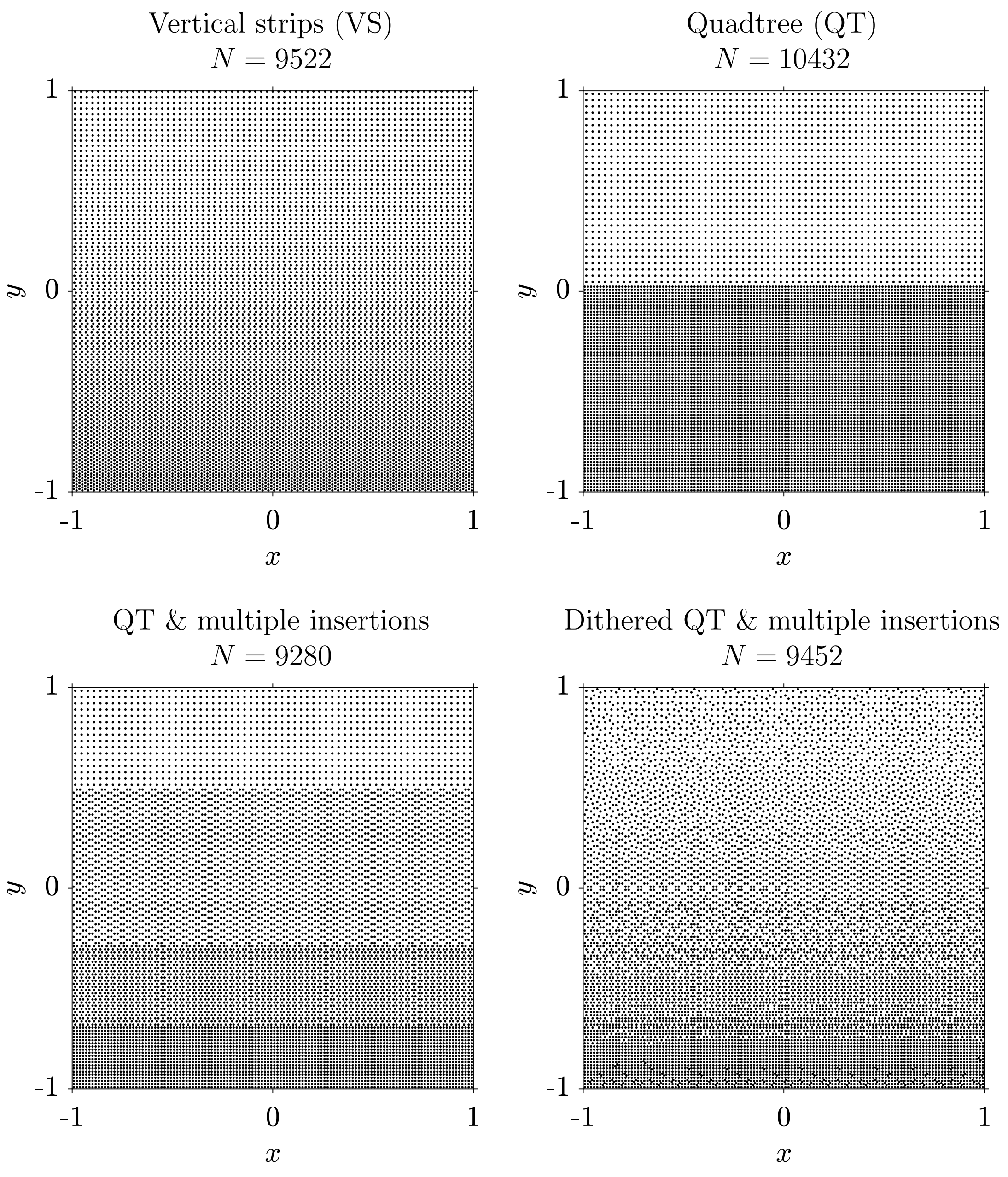


Figure 2.14: Comparison between different node generation algorithms for $s \propto 1 + (y+1)/2$. The exact number of nodes is $N_{\mathcal{P}} = 9459.31$.

the use of the dithering technique is highly recommended and justified.

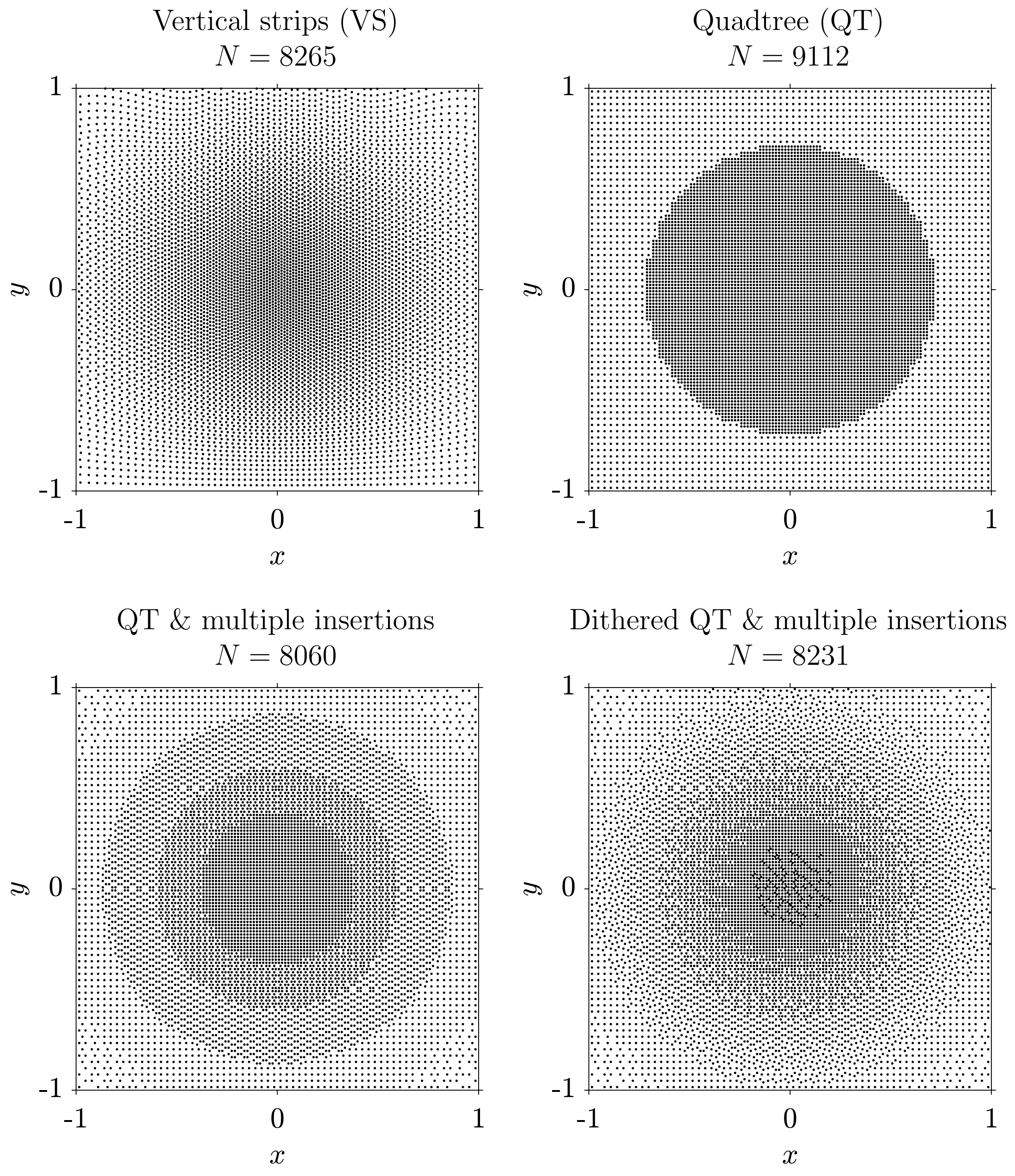


Figure 2.15: Comparison between different node generation algorithms for $s \propto 1 + (x^2 + y^2)$. The exact number of nodes is $N_{\mathcal{P}} = 8233.57$.

2.3. Node-repel refinement

2.3.1. Theoretical background

The node distributions generated by the DQT algorithm have correct nodal densities which fulfill the prescribed ones, as previously presented, but the local ar-

Table 2.1: Features of OXC and MQT algorithms

			Computing time[1]	
Algorithm	**Complexity**	**Memory**	2D	3D
QT/OT	$\mathcal{O}(N)$	$\mathcal{O}(N)$	0.18s	0.85s
DQT/DOT	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$	0.33s	1.45s

[1]For $N \approx 10^6$ nodes and using 1 core.

rangement of nodes does not fulfill the isotropy requirement which is beneficial for the RBF-FD meshless discretizations that will be employed. The local isotropy for nodal spacing, i.e., local arrangements which are nearly hexagonal in 2D or nearly FCC/HCP in 3D, can be achieved by the application of an iterative refinement process which moves the nodes according to the mutual repulsion forces of the nearest neighbouring nodes.

This node-repel approach can be viewed as an iterative process which minimizes the total potential energy U of the node distribution $X = \{\mathbf{x}_i, i = 1, \dots, N\}$ by moving one node at a time. The total potential energy U is built using neighbouring nodes only, taking the spacing function s into account as follows:

$$U(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{i=1}^{N} \sum_{j \in J(i)} \Phi\Big(\frac{r_{ij}}{s_{ij}}\Big) \tag{2.8}$$

where $J(i)$ is the set of the indices of the neighbours of node $\mathbf{x}_i$, Φ is a suitable potential function, $r_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|$ is the distance between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, and s_{ij} is the mean spacing function in between these two nodes. Any suitable symmetrical expression for s can be employed, e.g.:

$$s_{ij} = \frac{s(\mathbf{x}_i) + s(\mathbf{x}_j)}{2}\ , \quad s_{ij} = s\Big(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}\Big) \tag{2.9}$$

where the first expression is more convenient because spacing function is usually evaluated at node locations.

By taking the gradient of the potential U with respect to the cartesian coordinates of node $\mathbf{x}_i = (x_i, y_i, z_i)^T$ only, we obtain:

$$\frac{\partial U}{\partial \mathbf{x}_i} = 2 \sum_{j \in J(i)} F_\Phi\Big(\frac{r_{ij}}{s_{ij}}\Big) \frac{\partial}{\partial \mathbf{x}_i}\Big(\frac{r_{ij}}{s_{ij}}\Big) \tag{2.10}$$

where $F_\Phi = \Phi'$ is the derivative of the potential function Φ while the gradient operator $\partial/\partial \mathbf{x}_i$ is defined as follows:

$$\frac{\partial U}{\partial \mathbf{x}_i} = \Big(\frac{\partial}{\partial x_i}, \frac{\partial}{\partial y_i}, \frac{\partial}{\partial z_i}\Big)^T \tag{2.11}$$

The simple expression of $\partial U/\partial \mathbf{x}_i$ in Eq. (2.10) is due to the assumption of symmetry in the construction of neighbours $J(i)$: if $\mathbf{x}_j$ is a neighbour of $\mathbf{x}_i$, then $\mathbf{x}_i$ is a neighbour of $\mathbf{x}_j$. This assumption is obviously fulfilled in the case of nearly isotropic node distributions, which is the aim of the presented node generation algorithms.

For the sake of simplicity, the dependence of s_{ij} upon $\mathbf{x}_i$ is neglected in the derivative on the RHS of Eq. (2.10), yielding:

$$\frac{\partial}{\partial \mathbf{x}_i}\Big(\frac{r_{ij}}{s_{ij}}\Big) \approx \frac{1}{s_{ij}}\frac{\partial r_{ij}}{\partial \mathbf{x}_i} = \frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}s_{ij}} \tag{2.12}$$

Eq. (2.10) can then be recast in the following form:

$$\frac{\partial U}{\partial \mathbf{x}_i} = 2\sum_{j\in J(i)} \frac{1}{s_{ij}} F_\Phi\Big(\frac{r_{ij}}{s_{ij}}\Big)\frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}} \tag{2.13}$$

which states that the direction of maximum growth of the potential U equals the sum of the radial forces with magnitude $\frac{1}{s_{ij}}|F_\Phi(\frac{r_{ij}}{s_{ij}})|$ exerted on node $\mathbf{x}_i$ by the neighbouring nodes $\mathbf{x}_j, j \in J(i)$. Therefore, this is the direction along which node $\mathbf{x}_i$ should be moved in order to minimize U efficiently.

To calculate the actual magnitude of the displacement $\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}$ at iteration k, let us suppose U can be approximated by the following quadratic form with a minimum at the unknown node $\mathbf{x}_i^{(k+1)}$:

$$U(\mathbf{x}) \approx \alpha\Big(\frac{\|\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k+1)}\|}{\bar{s}}\Big)^2 \tag{2.14}$$

where $\bar{s} = s(\mathbf{x}_i^{(k)})$ and α is a positive constant.

The gradient of Eq. (2.14) with respect to $\mathbf{x}_i^{(k)}$ is:

$$\frac{\partial U}{\partial \mathbf{x}_i^{(k)}} = 2\alpha\frac{\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k+1)}}{\bar{s}^2} \tag{2.15}$$

which gives the actual displacement for node $\mathbf{x}_i$:

$$\frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\bar{s}} = -\frac{\bar{s}}{2\alpha}\frac{\partial U}{\partial \mathbf{x}_i^{(k)}} = -\frac{1}{\alpha}\sum_{j\in J(i)} \frac{\bar{s}}{s_{ij}} F_\Phi\Big(\frac{r_{ij}}{s_{ij}}\Big)\frac{\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}}{r_{ij}} \tag{2.16}$$

where the derivative $\partial U/\partial \mathbf{x}_i^{(k)}$ is obtained from Eq. (2.13) at iteration k.

Eq. (2.16) is then modified by neglecting the term $\bar{s}/s_{ij}$ as follows:

$$\frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\bar{s}} = -\frac{1}{\alpha} \sum_{j \in J(i)} F_\Phi\left(\frac{r_{ij}}{s_{ij}}\right) \frac{\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}}{r_{ij}} \tag{2.17}$$

which is the operative expression for the iterative procedure. The removal of the term $\bar{s}/s_{ij}$ from Eq. (2.16) is not a mere approximation, it is required in order to obtain a consistent process which converges to a node distribution which fulfill the prescribed spacing function. The iterative process defined by Eq. (2.16) is not consistent because even in the case $r_{ij} = s_{ij}$, i.e., nodal spacing matching the prescribed spacing, the denominator of ratio $\bar{s}/s_{ij}$ leads to nonzero displacements which cause the nodal spacing to slowly drift from the prescribed value.

The final Eq. (2.17) states that the displacement of node $\mathbf{x}_i$, normalized with the the spacing $\bar{s}$, equals the sum of the radial forces exerted by its neighbours $\mathbf{x}_j$, followed by a multiplication by the factor $1/\alpha$. α is unknown, but it is approximately independent upon nodal arrangements and it can be considered constant over the domain. Therefore the value $1/\alpha$ acts as a gain factor which has to be tuned by trial and error in order to maximize the convergence speed while avoiding instabilities.

The determination of the neighbours for each node is performed using a fast nearest neighbour search (NNS, see Appendix A). The efficiency of this task is crucial since the list of the neighbours is updated at each iteration. This choice is necessary during the early stages of the refinement process when local arrangements change radically, requiring a continuous update of the list of neighbouring nodes. For the latest refinement iterations, the list of neighbours can be updated less frequently, since the nodal arrangements change moderately. However, since the cost of the employed NNS is negligible when compared to the cost of the repel iteration, it is still performed at each iteration.

It is convenient to employ potential functions Φ with repulsive forces $-F_\Phi = -\Phi' > 0$, e.g., electric potential $\Phi = kr^{-1}$ with $-F_\Phi = kr^{-2} > 0$ for $k > 0$. With the choice of repulsive potentials, which has proven to be an effective choice for node refinement, the iterative procedure defined by Eq. (2.16) is actually a node-repel technique.

As outlined in Subsection 2.2.1, this node-repel procedure is not effective in the refinement of large-scale errors $E_\mathcal{P}$ since it operates locally: one node at a time with node displacements which are smaller than the local spacing. A "good" initial node distribution, i.e., with small errors $E_\mathcal{P}$ for large scales down to the small scales of the spacing function, is therefore essential for the efficiency of the overall node generation process.

The refinement process is iterated till a certain convergence criterion is met. In the limit of $k \to \infty$ iterations, a stable node distribution is reached, corresponding to a minimum for U which can be a simple local minimum. In these limit configurations, each nodes is in perfect equilibrium with the radial forces exerted by its neighbours, as stated by Eq. (2.17) with a zero LHS. If sufficiently smooth spacing functions s are employed, it is likely that almost each node is surrounded by a nearly isotropic arrangement of neighbouring nodes. Since the error $E_{\mathcal{P}}$ is assumed to be negligible, as shown for the DOT/DQT algorithm, it is concluded that the generated spacing matches the prescribed value. Anyway, few singular node arrangements which are not isotropic can also occur.

2.3.2. Boundary

The node-repel process obviously requires a strategy for boundary confinement, otherwise nodes would drift outside the domain. The simplest boundary confinement technique requires the generation of a boundary node distribution before the application of the refinement process. This distribution is not allowed to move during the repel iterations (fixed boundary nodes), while its nodes contribute to the repulsion of the nodes inside the domain (internal nodes). The boundary distribution has also to fulfill the prescribed spacing function s.

In 2D this technique is very simple since the boundary is composed by one or more closed curves and therefore the fulfillment of the prescribed spacing function is straightforward. Let us consider Eq. (2.5) in the 1D case along the boundary, which consists in a single curve Γ for the sake of simplicity:

$$N_\Gamma(t) = \int_{t_0}^{t} \frac{\mathrm{d}L}{s} \tag{2.18}$$

where t is a parameter for $\Gamma(t)$, e.g., the curvilinear length itself, and $\mathrm{d}L$ is the corresponding infinitesimal boundary length. $N_\Gamma(t)$ therefore gives the cumulative number of nodes along Γ starting from $\Gamma(t_0)$. Boundary nodes $\check{\mathbf{x}}_k = \Gamma(t_k), k = 1, \dots, N_B$ are then generated when $N_\Gamma(t_k) = k$.

The extension of this technique to 3D cases is not straightforward since the boundary Γ is a surface which requires an appropriate 2D node distribution itself. Therefore a more general approach for boundary confinement must be employed.

A possible solution for the 3D boundary confinement is given by the projection technique. This strategy projects nodes onto the nearest boundary during each node-repel iteration when these nodes cross the boundary, as depicted in Figure 2.16. The projected nodes become boundary nodes and can implicitly move along the boundary only, while the prescribed spacing is fulfilled as well as for the internal nodes. This is perhaps the most natural technique for boundary confinement in the case of the node-repel refinement process. Problems may arise in the case of sharp concave boundaries, but this is beyond the scope of the presented work.

The projection technique requires two new elements: the detection of the nodes crossing the boundary and suitable projection techniques. In the case of boundaries

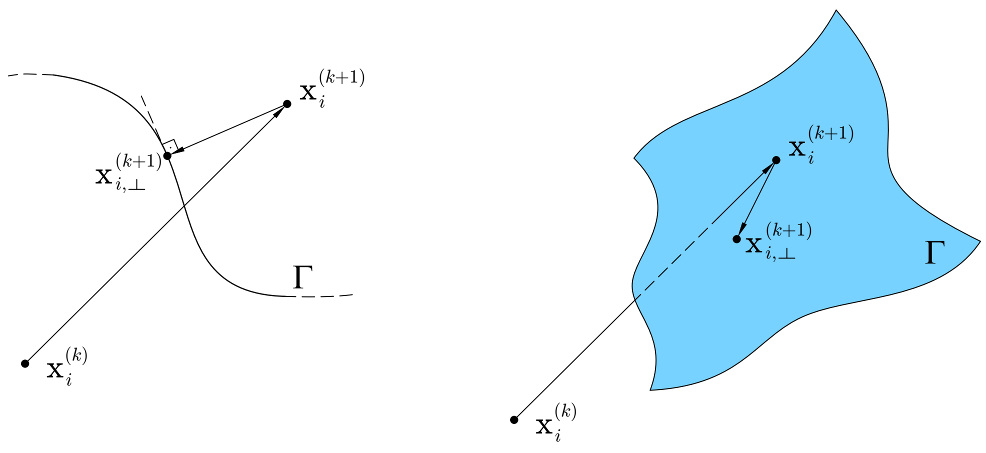


Figure 2.16: Projection of a node $\mathbf{x}_i$ crossing the boundary Γ at iteration k in 2D (a) and 3D cases (b).

defined by few simple geometric entities, these new elements are straightforward to implement, which is the case of the 3D domains employed in the present work. In the case of general-shaped boundaries, efficient techniques can be employed for these tasks. Obviously, the projection technique can also be employed in 2D cases if required.

2.3.3. 2D CASES

Figure 2.17 illustrates the different phases involved in the node generation process for a simple 2D case. A circular domain Ω is considered together with a prescribed spacing function $s(\mathbf{x})$ which decreases from s_{max} at the center of the domain to s_{min} at the boundary Γ, as depicted in Figure 2.17. The initial node distribution is obtained for the whole bounding box $\mathcal{B}$ with the DQT algorithm previously introduced. The spacing function outside the domain is considered to be $s = s_{max}$ in order not to generate a high number of unnecessary nodes. Nodes outside the boundary Γ are then eliminated and a boundary distribution fulfilling s is generated. The node generation process concludes with the application of a certain number of node-repel iterations which refine the distribution.

For 2D cases, the following type of radial force $-F_\Phi$ is found to give satisfactory results:

$$-F_\Phi(r) = \frac{1}{(r^2+\beta)^2} \tag{2.19}$$

where β is an adjustable parameter. The repulsion force defined by Eq. (2.19) is

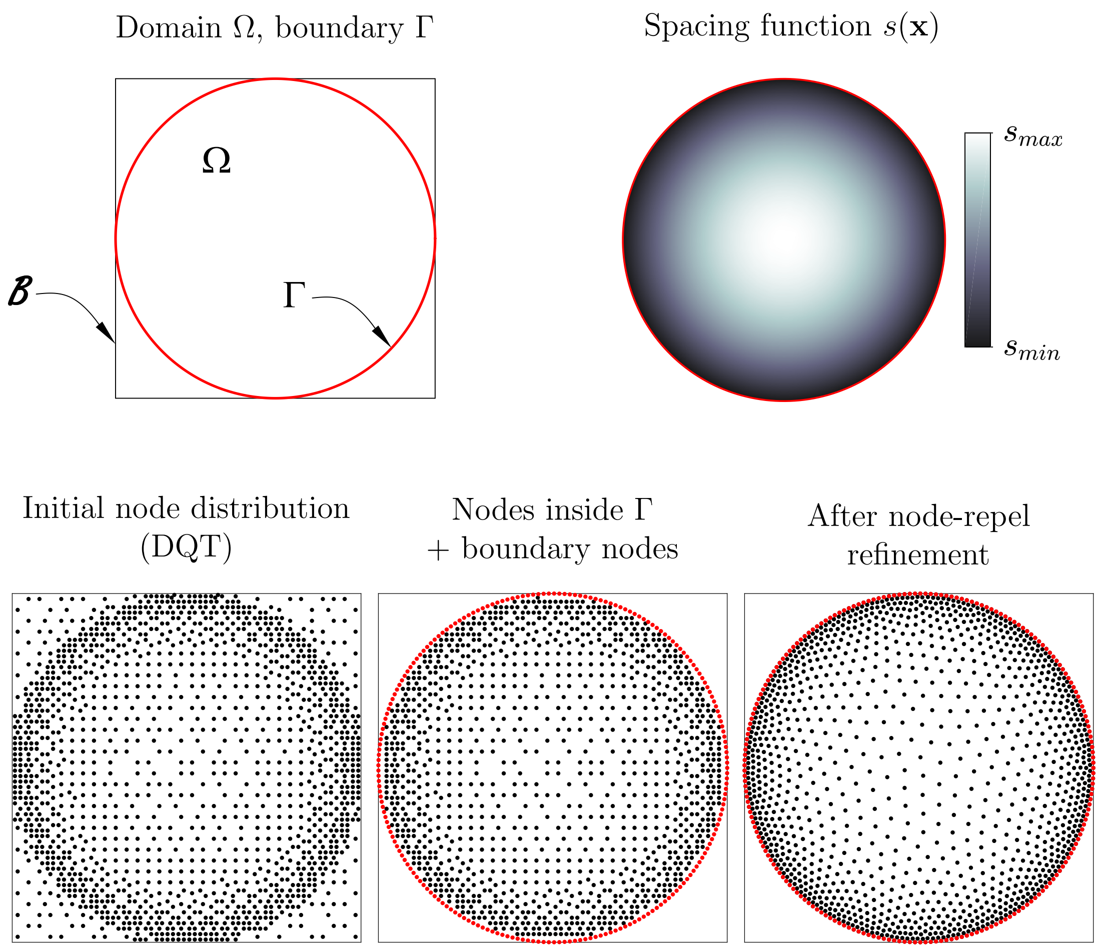

Domain Ω, boundary Γ
Ω
$\mathcal{B}$
Γ
Spacing function $s(\mathbf{x})$
s_{max}
s_{min}
Initial node distribution (DQT)
Nodes inside Γ + boundary nodes
After node-repel refinement

Figure 2.17: 2D node generation process for a circular domain Ω with a spacing function s.

decreasing with r and decays as r^{-4} for large nodal distances r. The parameter β acts as a force limiter in the case of close nodes or coincident nodes, i.e., $r \to 0$, in which cases the force is limited to the finite value $-F_\Phi(0) = \beta^{-2}$.

Another parameter which has to be tuned is the number of nearest neighbours n_R employed in the calculation of the the radial forces acting on each node, Eq. (2.17). The choice of this parameter affects both the computational costs and the quality (isotropy) of the generated distribution: a small value for n_R yields unsatisfactory results, e.g., cartesian-like patterns and holes, while big values for n_R require unnecessary computational effort since the contributions of the furthest nodes are almost negligible due to the decreasing nature of the force $-F_\Phi$.

The influence of these parameters on the characteristics of the generated distribution is studied for a node distribution with $N \approx 10^5$ nodes in a square and a

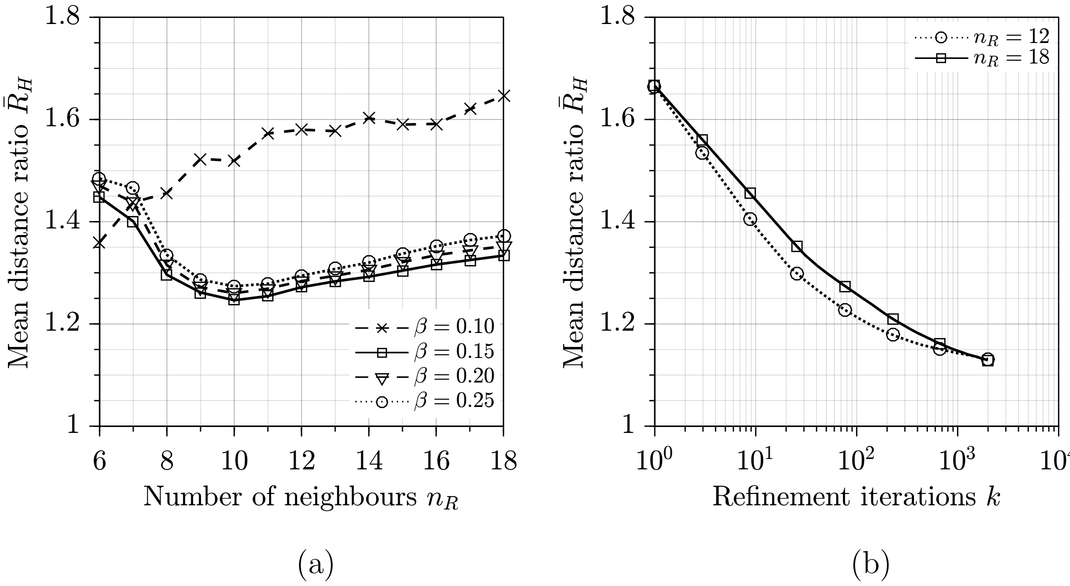


Figure 2.18: (a) Mean distance ratio $\bar{R}_H$ after $500/n_R$ refinement iterations; (b) $\bar{R}_H$ convergence history for $\beta = 0.20$. $N \approx 10^5$ nodes.

constant spacing function. A distance ratio R_H is defined for each node as the ratio of maximum to minimum distances between the 6 nearest neighbouring nodes. R_H can be interpreted as an isotropic quality index with respect to hexagonal distributions for which $R_H = 1$, while $R_H \geq 1$ for generic distributions, e.g., $R_H = \sqrt{2}$ for a cartesian distribution. The value of $1/\alpha$ which defines the gain factor for the nodal displacements is chosen to be 0.1 in order to maximize the convergence speed while avoiding instabilities, regardless of the remaining parameters.

Figure 2.18(a) shows the influence of the number of neighbours n_R upon the mean distance ratio $\bar{R}_H$, i.e., the mean value of R_H over the whole distribution, after $k = 500/n_R$ repel refinement iterations. The choice $k = 500/n_R$ iterations takes account of the increasing computational cost with respect to n_R, i.e., each repel iteration requires $N \cdot n_R$ operations. The mean distance ratio shows a minimum for $n_R = 10$ and $\beta = 0.15$ which are the optimal parameters for the minimization of $\bar{R}_H$ for a given computational work. Anyway, in order to avoid instabilities and other side effects arising in different situations, $n_R = 12$ and $\beta = 0.20$ have been preferred and employed from now on.

A geometrical interpretation of the influence of the number of neighbouring nodes n_R upon the distance ratio can be derived from Figure 2.19 which depicts a 2D hexagonal node distribution, which represents the ideal node distribution to be reached in the case of constant spacing. Obviously, the minimum number of neighbours to consider is $n_R \geq 6$, while a non-negligible contribution is also

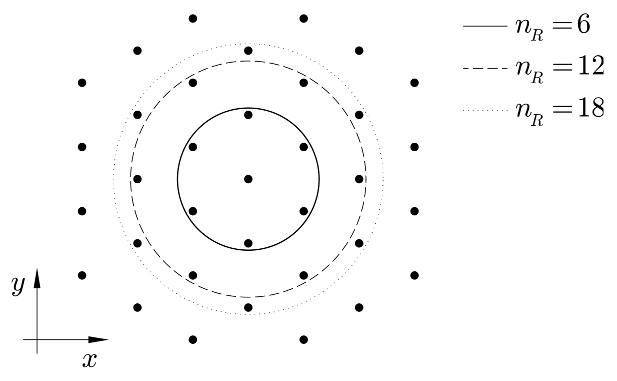


Figure 2.19: Neighbouring nodes for the node-repel phase with a 2D hexagonal node distribution.

given by the 6 nearest neighbours contained in the strip between $n_R = 6$ and $n_R = 12$. Therefore a significant improvement in the reduction of the distance ratio is expected by increasing n_R from 6 to 12. Above $n_R = 12$ the contribution of the 6 successive nearest neighbours, contained in the strip between $n_R = 12$ and $n_R = 18$, is negligible since the repulsive force $-F_\Phi$ decays rapidly with the distance, therefore representing unnecessary operations.

Figure 2.18(b) shows the convergence histories of the mean distance ratio $\bar{R}_H$ for different values of $n_R = 12, 18$. $\bar{R}_H$ decreases rapidly during the first 20 node-repel iterations, while an apparently asymptotic behaviour is reported for large k. A significant element is represented by the slow convergence towards a unitary mean distance ratio. This is due to the appearance of a non-negligible number of nearly stable nodal arrangements which are not hexagonal. The appearance of these arrangements is reported in Figure 2.20 where nodes with $n_R > \sqrt{2}$ are coloured in red. The enlarged view reveals that such singular arrangements are typically characterized by a pentagonal structure for the nearest neighbours with the 6th nearest neighbour at a larger distance.

The asymptotic behaviour of $\bar{R}_H$ is also confirmed by the distributions of the distance ratio, which are reported in Figure 2.21 for $k = 2, 20, 200$ and 2000 repel iterations. At the beginning of the iterative process, i.e., $k = 2$, most of the nodes have large distance ratios $R_H > 1.4$, while the application of a small number of $k = 20$ iterations leads to a significant shift of the distribution towards smaller values $R_H < 1.4$. At this point, significant improvements can be obtained by the application of $k = 200$ iterations, for which most of the nodes have $R_H < 1.25$. Successive iterations have limited effect: after $k = 2000$ iterations the fraction of nodes at $R_H = 1.05$ is increased while a significant fraction at $R_H = 1.2$ is still present.

Despite the slow asymptotic convergence of the node-repel process, the pre-

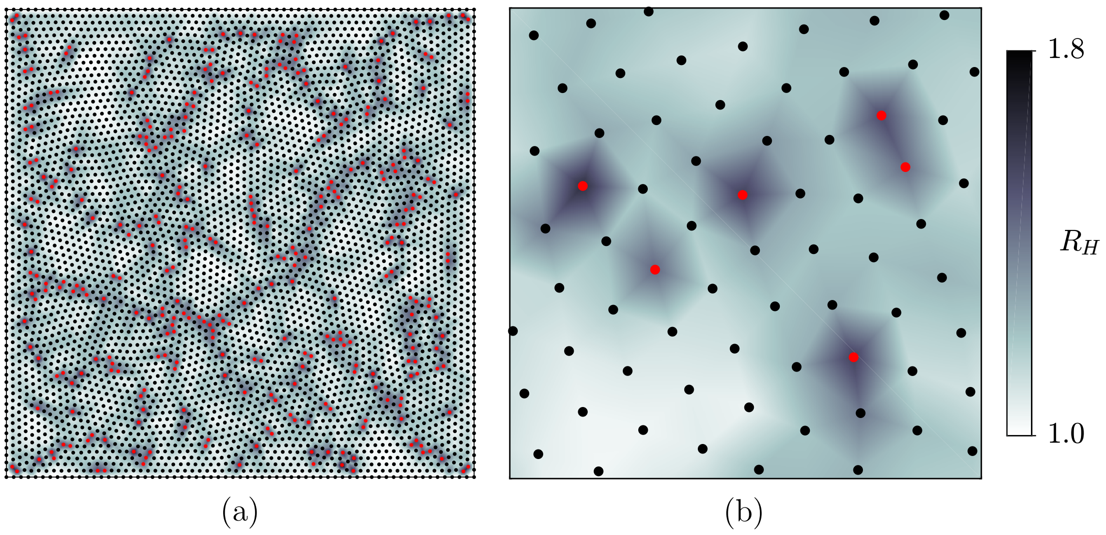


Figure 2.20: (a) Node distribution and the corresponding distance ratio R_H for a constant spacing function after 200 refinement iterations; (b) enlarged view. $R_H > \sqrt{2}$ for red nodes, $N \approx 5000$ nodes.

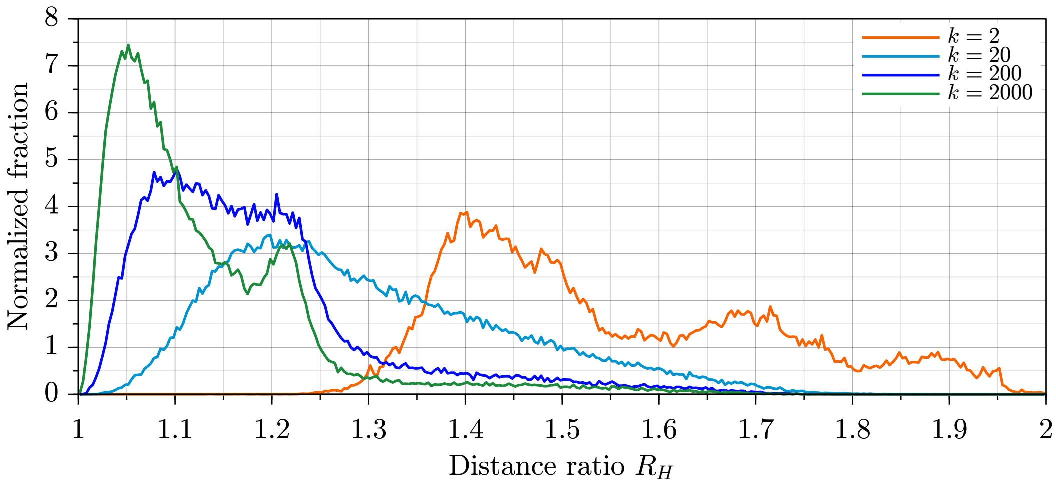


Figure 2.21: Distributions of the distance ratio R_H for $N \approx 10^5$ nodes and $n_R = 12$.

sented procedure can efficiently generate "good" node distributions with a reasonably small mean distance ratio $\bar{R}_H < 1.3$ by using a moderate number of node-repel iterations, e.g., $k = 50 - 200$. The generated distributions are therefore excellent candidates for their use in RBF-FD meshless discretizations in order to yield an accurate and flexible discretization process.

Number of nodes N	
67824	60889
Computing time (4 cores, $k = 100$ iterations)	
0.90s	0.95s
Mean distance ratio $\bar{R}_H$	
1.17	1.22

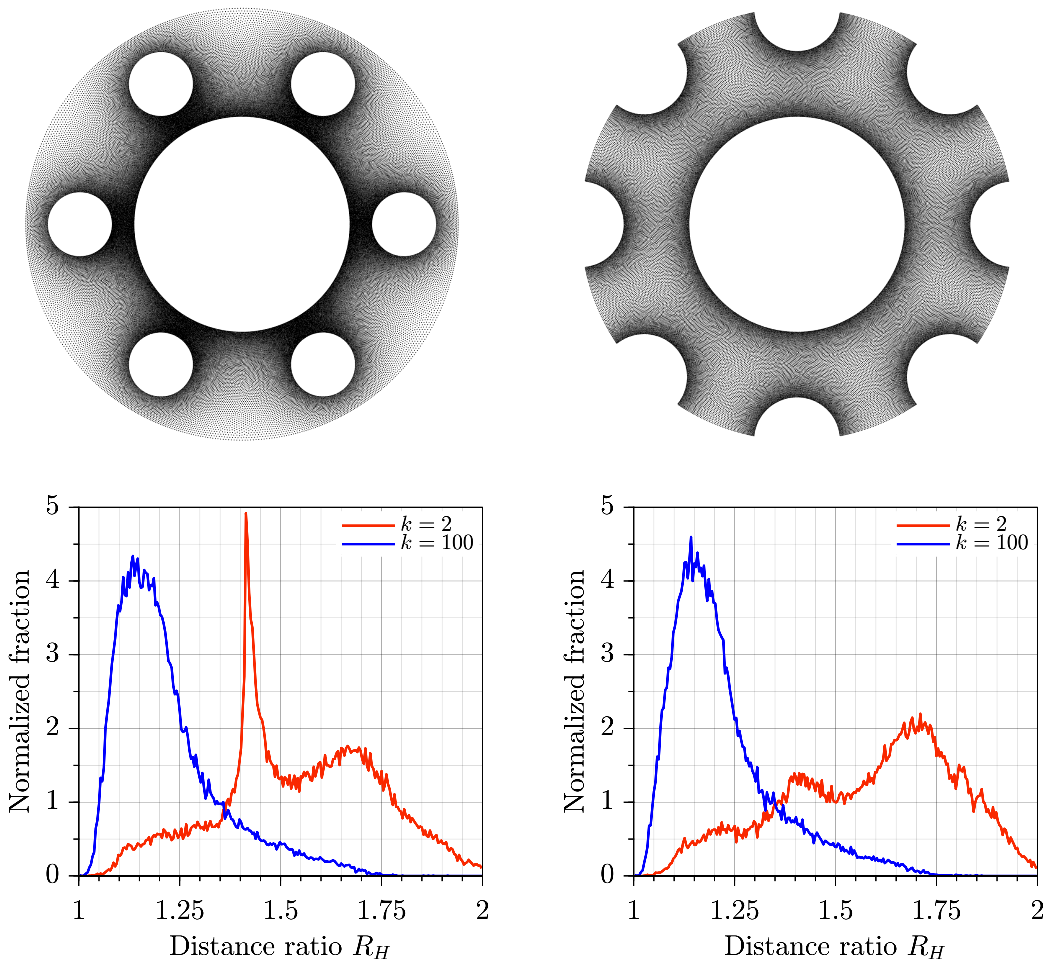


Figure 2.22: Examples of node distributions over 2D complex-shaped domains (top) and the corresponding distributions of the distance ratio R_H (bottom).

Two examples of practical node distributions generated over complex-shaped domains are given in Figure 2.22 together with the corresponding distributions of the distance ratio at the beginning and at the end of the node-repel refinement.

The employed node-repel process has been capable of an effective reduction of the mean distance ratio from large values $\bar{R}_H > 1.6$ to a reasonable value $\bar{R}_H \approx 1.2$ within $k = 100$ repel iterations in both cases where $N \approx 65000$ nodes have been employed.

2.3.4. 3D cases

The node generation process in 3D cases employs the same approach used in 2D cases. An initial distribution is generated for the whole box bounding the domain Ω using the DOT algorithm previously introduced. The spacing function outside the domain is again considered to be $s = s_{max}$ in order not to generate a high number of unnecessary nodes. Nodes outside the boundary Γ are then eliminated, while no boundary distribution is generated since the boundary projection technique presented in Subsection 2.3.2 is employed. This technique greatly simplifies the whole node generation approach since no additional boundary distribution is required, which would need the employment of a 2D node generator for each boundary surface. Instead, the boundary confinement is implicitly introduced within each node-repel iteration, where nodes crossing the boundary are projected onto it and therefore they are implicitly constrained onto the boundary itself.

For the node-repel process, the employed radial force has the same expression of the one employed in 2D cases, Eq. (2.19), for which the number of neighbouring nodes n_R and the parameter β have to be optimized again in order to maximize the effectiveness of the refinement process while minimizing the computational cost.

The influence of these parameters on the characteristics of the generated distribution is studied for a node distribution with $N \approx 50000$ nodes in a sphere and a constant spacing function. The distance ratio R_H for 3D cases is defined for each node as the ratio of maximum to minimum distances between the 12 nearest neighbouring nodes. R_H can be interpreted as an isotropic quality index with respect to FCC/HCP lattices for which each node has 12 equally distanced neighbours and thus $R_H = 1$. For a cartesian distribution $R_H = \sqrt{2}$. The optimal value of $1/\alpha$ which defines the gain factor for the nodal displacements is found to be 0.1 again, as in the 2D case. Such optimal value maximizes the convergence speed while avoiding instabilities, regardless of the remaining parameters.

Figure 2.23(a) shows the influence of the number of neighbours n_R upon the mean distance ratio $\bar{R}_H$, i.e., the mean value of R_H over the whole distribution, after $k = 1000/n_R$ repel refinement iterations. The choice $k = 1000/n_R$ iterations takes account of the increasing computational cost with respect to n_R, i.e., each repel iteration requires $N \cdot n_R$ operations. The mean distance ratio shows a minimum for $n_R \approx 32$ and $\beta = 0.10$. Anyway, in order to avoid possible instabilities arising in different situations, $n_R = 36$ and $\beta = 0.15$ have been preferred and employed from now on.

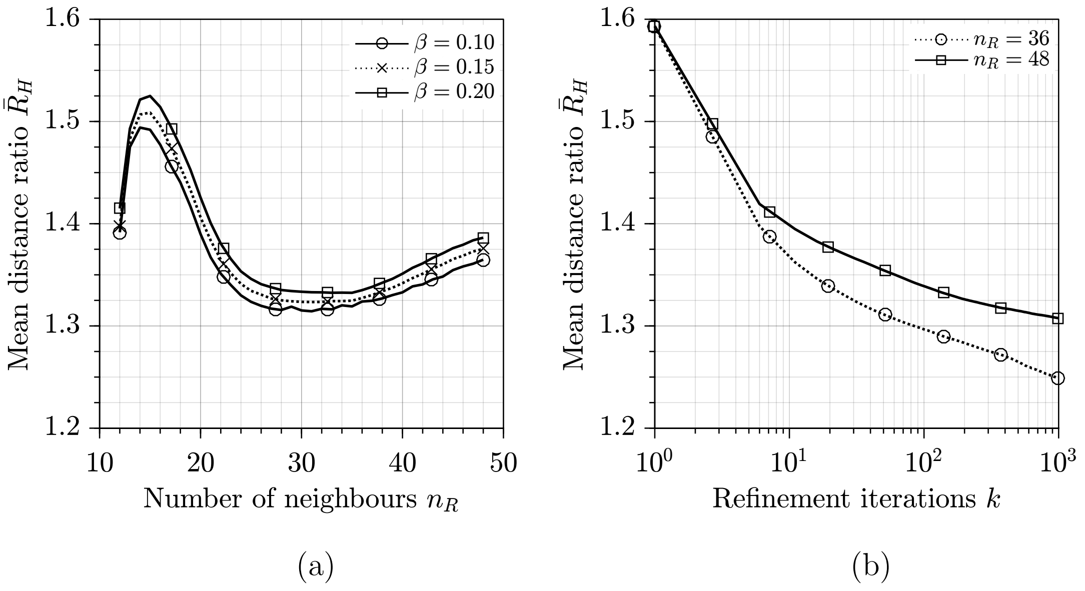


Figure 2.23: (a) Mean distance ratio $\bar{R}_H$ after $1000/n_R$ refinement iterations; (b) $\bar{R}_H$ convergence history for $\beta = 0.15$. $N \approx 50000$ nodes.

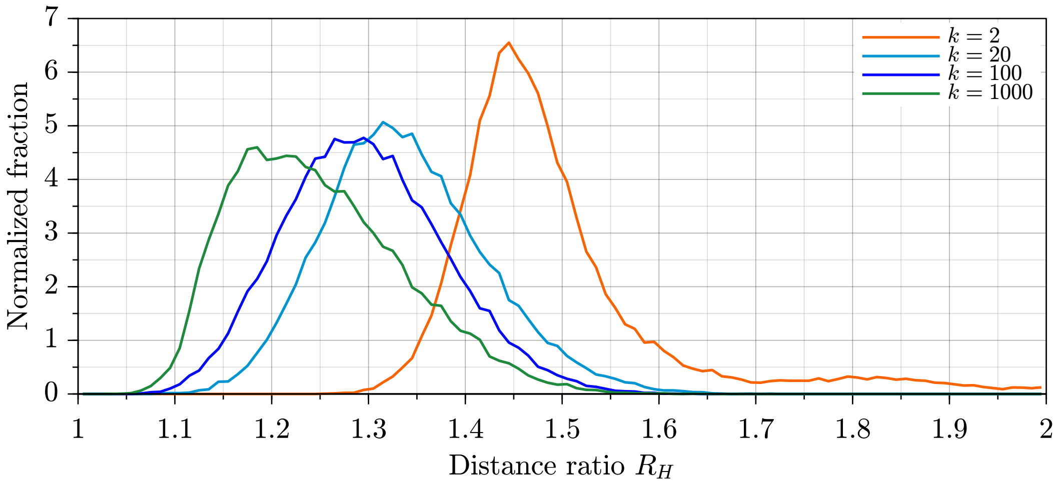


Figure 2.24: Distributions of the distance ratio R_H for $N \approx 50000$ nodes and $n_R = 36$.

A geometrical interpretation of the influence of the number of neighbouring nodes n_R upon the distance ratio can be made by considering the FCC/HCP lattices which are obtained by packing planes of hexagonal arrangements. The minimum number of neighbours to consider is then $n_R = 12$, from which the addition of a small number of neighbours is not effective since their contribution is highly non-symmetric. Non-negligible contributions are then given by $n_R > 20$

neighbours which can restore a spatial symmetry, bringing significant improvements. Above $n_R = 40$ the contributions of the successive nearest neighbours are negligible because of the large distances and should not be employed since they require unnecessary operations.

The convergence histories for the mean distance ratio $\bar{R}_H$ are shown in Figure 2.23(b) for different values of $n_R = 36, 48$. The employment of a large number $n_R = 48$ of neighbours appears to be even less effective than the cheaper choice $n_R = 36$. In both cases $\bar{R}_H$ decreases rapidly during the first 10-20 node-repel iterations, while a slow convergence towards a unitary mean distance ratio is reported.

The distributions of the distance ratio are shown in Figure 2.24 for $k = 2, 20, 100$ and 1000 repel iterations. At the beginning of the iterative process, i.e., $k = 2$, most of the nodes have large distance ratios centred on $R_H \approx 1.45$, while the application of a small number of $k = 20$ iterations leads to a significant reduction towards smaller values centred at $R_H \approx 1.3$. The application of $k = 100$ iterations leads to almost negligible improvements and the application of a high number $k = 1000$ of iterations brings to a limited reduction of R_H.

Figure 2.25 shows two 3D examples of generated distributions together with the corresponding distributions of the distance ratio. In the first example a simple spherical domain and a constant spacing function are employed.

In the second example the domain is defined by using the following function:

$$f_\Omega = 4(x^2 + y^2) - \cos(3\pi z) + 3z^4 - 3 \tag{2.20}$$

for which the domain Ω is implicitly defined by $f_\Omega < 0$ and the boundary Γ is implicitly defined by $f_\Omega = 0$. The employed spacing function is $s(\mathbf{x}) \propto 2 - e^{f_\Omega/2}$ which is minimum at the boundary and doubles at the center of the domain.

Using the previous definitions, the projection technique required for the boundary confinement of a node $\mathbf{x}_i$ can be easily implemented at each iteration k as follows:

$$\begin{aligned}&\textbf{if}\ \ f_\Omega(\mathbf{x}_i^{(k+1)}) > 0\\ &\qquad \mathbf{x}_{i,\perp}^{(k+1)} - \mathbf{x}_i^{(k+1)} = -\left(\frac{f_\Omega \nabla f_\Omega}{\|\nabla f_\Omega\|^2}\right)_{\mathbf{x}=\mathbf{x}_i^{(k+1)}}\end{aligned}$$

where $\mathbf{x}_{i,\perp}^{(k+1)}$ is the first order projection of node $\mathbf{x}_i^{(k+1)}$ onto the boundary Γ. With the previous projection formulation, nodes escaping from the domain because of the repulsive forces are continuously constrained onto the boundary.

The distributions of the distance ratio depicted in Figure 2.25 confirm that in 3D cases the node-repel technique is slightly less effective in the reduction of R_H than in 2D cases. However, for both the presented examples the employment of a moderate number $k = 100$ of node-repel iterations leads to a uniform reduction

Number of nodes N	
99426	100624
Computing time (4 cores, $k = 100$ iterations)	
3.20s	3.00s
Mean distance ratio $\bar{R}_H$	
1.30	1.33

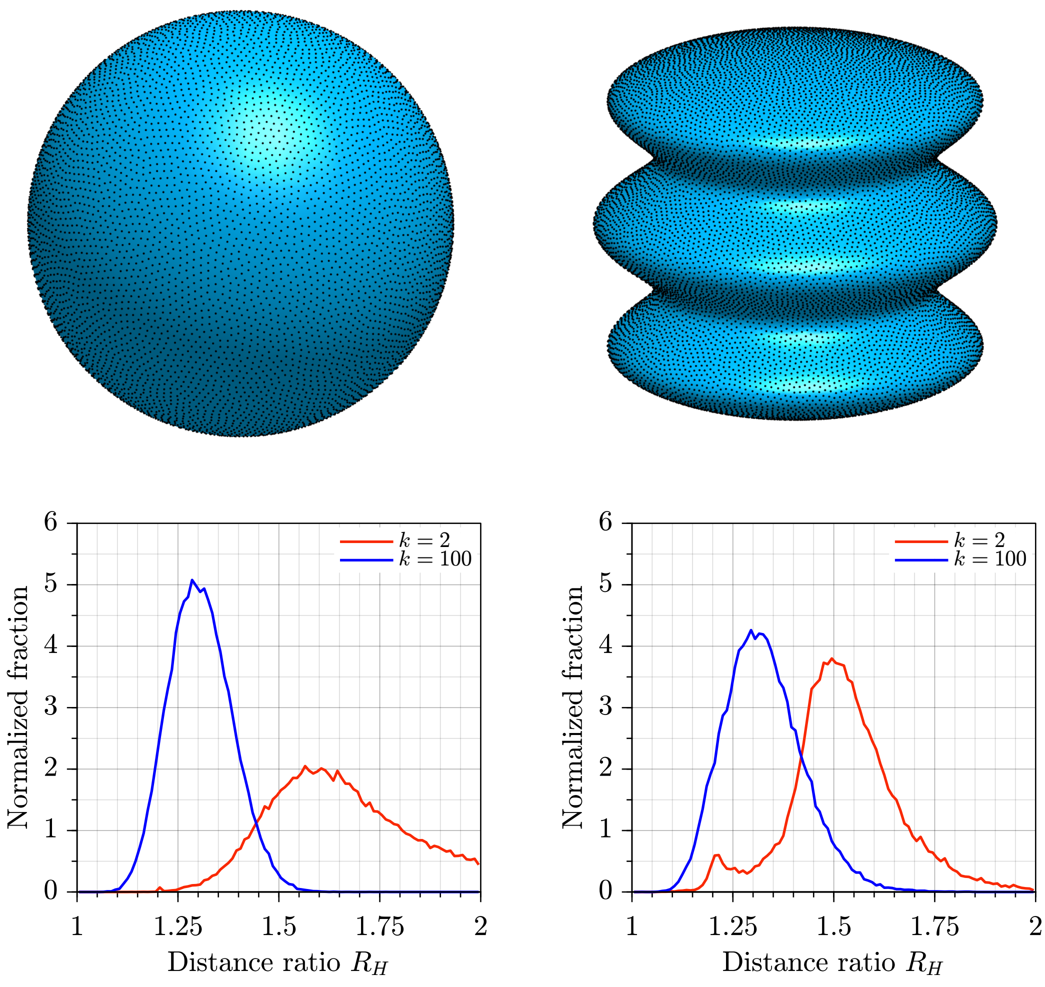


Figure 2.25: Examples of node distributions over 3D domains (top) and the corresponding distributions of the distance ratio R_H (bottom).

of the distance ratio for the whole set of nodes, i.e., there are no regions where nodes exhibit excessive anisotropy. These node distributions therefore represent

Table 2.2: Computing times(speedup) for $k = 100$ node-repel iterations.

		A (2D)				**B** (2D)
N	**Cores**:	1	2	4	4 (8 threads)	4 (8 threads)
100,000		4.15	2.20(1.9)	1.40(3.0)	1.00(4.2)	1.50
250,000		10.5	5.50(1.9)	3.50(3.0)	2.40(4.4)	3.30
500,000		22.0	11.3(2.0)	6.70(3.3)	4.70(4.7)	6.45
		C (3D)				**D** (3D)
N	**Cores**:	1	2	4	4 (8 threads)	4 (8 threads)
100,000		15.4	7.80(2.0)	4.80(3.2)	3.20(4.8)	3.00
250,000		40.1	20.8(1.9)	12.8(3.1)	8.40(4.8)	7.50
500,000		87.5	46.0(1.9)	26.5(3.3)	17.8(4.9)	15.25

2D cases: **A**: circle, constant spacing
B: first example of Figure 2.22

3D cases: **C**: sphere, constant spacing
B: second example of Figure 2.25

good candidates for their use in RBF-FD meshless discretizations.

2.3.5. Computing times

The complexity of the node-repel algorithm is linear in the number of nodes N in both 2D and 3D cases if a fixed number of iterations k is employed, which is recommended. The computing times for $k = 100$ node-repel iterations and different 2D and 3D cases are reported in Table 2.2 together with the speedup values obtained by using an OpenMP parallelized C code. In 2D cases $n_R = 12$ neighbours are employed, while $n_R = 36$ in 3D cases.

The computing times are comparable for cases **A** (2D circle, constant spacing) and **B** (first 2D example of Figure 2.22), despite the latter case employs a more complex domain and a more complex spacing function than the former. The same applies when comparing cases **C** (3D sphere, constant spacing) and **D** (second 3D example of Figure 2.25), where the latter case performs even better than the former, despite its non-trivial geometry.

The speedup values are also satisfying for both 2D and 3D cases, with a maximum speedup value of 4.9 for the 3D case where $N = 500,000$ nodes and 4 cores (8 OpenMP threads) are employed.

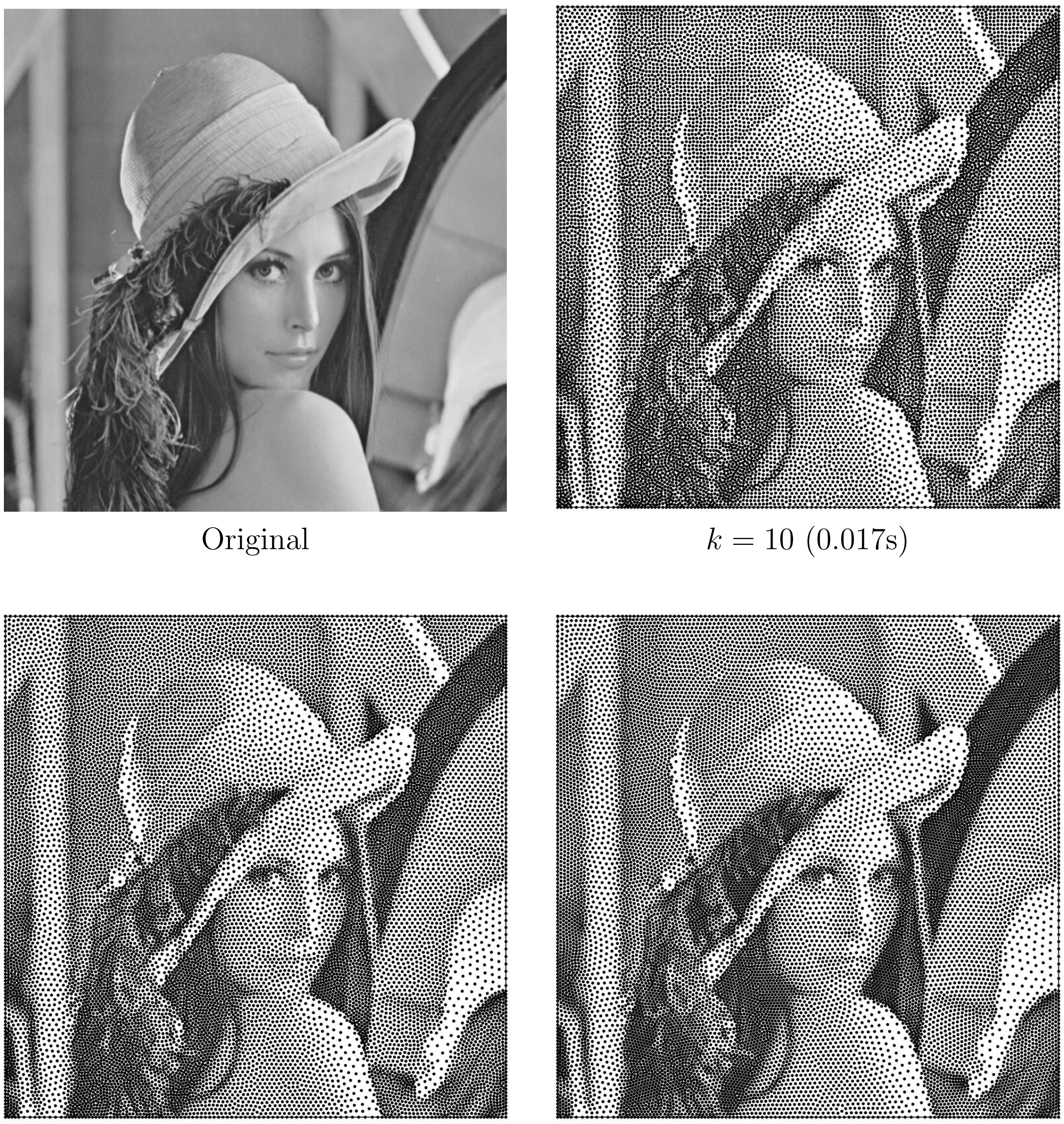


Figure 2.26: Halftone approximation of a greyscale test image through stippling with k node-repel iterations. $N = 16543$ nodes.

2.4. Stippling

The proposed node generation procedure in 2D cases can also be used for the halftone approximation of black/white images through the stippling technique [75]. This technique is employed to approximate the continue halftones of a greyscale image with an adequate distribution of equally sized black dots over a white background. It differs from the dithering techniques which are widely used in image processing where the black dots can only have fixed positions and sizes, i.e., pixels.

In order to employ the presented node generation algorithm for stippling, the prescribed spacing function s has to be expressed as a function of the image brightness $\nu \in [0, 1]$ ($\nu = (R + G + B)/3$ if the brightness is obtained from a coloured image). For simplicity, we assume that the complement of the brightness, i.e., the blackness, is proportional to the nodal density δ through the area A_D of each dot:

$$1 - \nu = \frac{number\ of\ nodes}{unit\ area/volume} \cdot A_D = \delta A_D = \frac{2}{\sqrt{3}s^2} \tag{2.21}$$

which does not account for overlapping dots.

The spacing function is then obtained from Eq. (2.21):

$$s = \sqrt{\frac{2A_D}{\sqrt{3}(1 - \nu)}} \tag{2.22}$$

Obviously, the smaller the dots, the smaller the spacing function and therefore more and more dots will be needed for the approximation of the image.

Figure 2.26 shows an example of stippling of a test image using the presented node generation algorithm with $k = 10, 100$ and 1000 node-repel iterations and $N = 16543$ nodes. $n_R = 12$ nearest neighbours have been employed, while the gain factor $1/\alpha$ is reduced to 0.05 in order to prevent possible instabilities arising from the brighter regions where the spacing is large.

The resulting stippled images show a remarkable graphic quality for $k = 100$ and $k = 1000$ iterations, while $k = 10$ iterations are not sufficient to eliminate the graphical patterns due to the DQT algorithm employed for the generation of the initial node distribution. This example also shows that the presented node generation algorithm can effectively handle nodal densities with steep variations with robustness.

2.5. Conclusions

Different algorithms for the 2D/3D node generation problem over arbitrarily-shaped domains have been proposed. Such algorithms are characterized by a meshless approach where no polygonization of the domain is required, resulting in flexible and efficient procedures. The node generation process is composed of an initial node placing phase which is followed by a refinement phase based on the mutual repulsion of the nodes. This kind of approach is very general and robust, and it can be applied to a wide range of node generation problems with no limitations.

These procedures are developed in order to generate the node distributions required by generic meshless methods, with particular reference to the localized radial basis function (RBF) approach which is employed in this work.

Chapter 3

RBF-FD method

The basis of each of the methods employed in the numerical solution of PDEs is the construction of suitable approximations for the solution itself and for its derivatives. These formal approximations are then introduced into the involved PDE in order to obtain the sought solution.

In the FDM the solution is assumed to be known at some grid points only, while the derivatives are obtained from local polynomial functions which interpolate the solution at the grid points. The PDE is then made valid at grid points through the collocation technique. In the FEM the solution is assumed to be piecewise continuous within each element, typically using a polynomial expansion which matches the solution at the element nodes. The derivatives are obtained by deriving the polynomial expansion and the weak form of the PDE is properly satisfied. In the FVM the solution is assumed to be known at the cell centers and the PDE is averaged on each cell, requiring suitable interpolation schemes for the solution and its derivatives.

In the radial basis function-generated finite difference method (RBF-FD), the solution is assumed to be known at some scattered nodes and a local interpolant is defined in the neighbourhood of each node by using a RBF expansion. The derivatives are formally obtained by the analytic differentiation of the RBF expansion, while the PDE is made valid at node locations using the collocation technique. The RBF-FD denomination is due to the fact that its approach resembles the FD method, but using arbitrarily scattered nodes and a RBF approach for the construction of the interpolant. The RBF-FD approach is also known as local RBF collocation method (LRBFCM) [59, 76, 77, 89] and localized RBF meshless method [22, 105].

Since the RBF-FD method is composed by a RBF interpolation and a following collocation approach, it is therefore important to analyze the characteristics of the RBF-FD discretization scheme by studying the properties of these two elements in a consequential order.

3.1. Scattered data RBF interpolation

3.1.1. Problem definition

The problem of scattered data interpolation is of primary importance in meshless applications for obvious reasons. In D dimensions it can be stated as follows: given a set of n nodes $\mathbf{x}_i \in \mathcal{R}^D$ and n real values f_i, find a continuous function g such that:

$$g(\mathbf{x}_i) = f_i \tag{3.1}$$

for $i = 1, \ldots, n$. We suppose the values $f_i = f(\mathbf{x}_i)$ are generated by an unknown function f.

When the nodes have regular arrangements, e.g., cartesian, or the problem is 1D with distinct nodes, polynomial interpolants can be successfully employed since the interpolation problem is always well-posed. In the case of uniform cartesian arrangements, the use of polynomials brings great advantages in terms of implementation ease and accuracy: in 1D cases the interpolation error is $|g - f| < c_n h^n$ where h is the cartesian spacing and the function f has continuous derivatives up to order n.

The problem of polynomial interpolation arises when the nodes have arbitrary arrangements in more than one dimension. Singular node arrangements leads to ill-posed interpolation problems, e.g., interpolation with a plane in 2D for $n = 3$ nodes lying on a line.

The ill-posedness is common to every interpolation scheme of the type:

$$g(\mathbf{x}) = \sum_{i=1}^{n} a_i \phi_i(\mathbf{x}) \tag{3.2}$$

which employs basis functions ϕ_j that are independent upon the nodal positions $\mathbf{x}_i$. This is explained by the Mairhuber-Curtis theorem [26]; let us consider the system of equations obtained from the interpolation conditions expressed by Eq. (3.1) using the expansion defined in Eq. (3.2):

$$\begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_n(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_n) & \cdots & \phi_n(\mathbf{x}_n) \end{bmatrix} \begin{Bmatrix} a_1 \\ \vdots \\ a_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{3.3}$$

where a_i are the unknown coefficients of the interpolating function g. The compact notation of Eq. (3.3) is:

$$\boldsymbol{\phi}(\mathbf{x}_1, \ldots, \mathbf{x}_n)\mathbf{a} = \mathbf{f} \tag{3.4}$$

Consider now a closed path along which two nodes are continuously moved without interfering with the remaining nodes, until the two nodes exchange their position. This corresponds to a row exchange in the interpolation matrix in Eq. (3.3), resulting in a change of the sign of its determinant. Therefore the determinant is 0 somewhere along this nodal exchange, since the determinant is a continuous function of $\mathbf{x}_i$ (ϕ_j are also continuous).

This ill-posed problem arises for singular node arrangements that are not easily predictable in general situations, which is the case of the node distributions employed in meshless methods. Despite different criteria for a proper choice of the interpolation nodes have been proposed in order to address this problem, nearly singular node arrangements can still be encountered causing large numerical errors. Another possibility is to use more interpolation nodes than the number of the basis function and employing a least squares approach [70, 84].

A different choice can be made in order to overcome this problem by using basis functions ϕ_j depending upon the nodal positions. Radial basis functions represent very good candidates for scattered data interpolation in multiple dimensions [26] since the basis functions depend upon a distance, which is typically the Euclidean distance.

3.1.2. Radial basis functions

Radial basis functions are defined as follows:

$$\phi_j(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{x}_j\|) \tag{3.5}$$

for which there is a dependence upon the distance from the interpolation node $\mathbf{x}_j$, as suggested by the Mairhuber-Curtis theorem. We note that the interpolation matrix $\boldsymbol{\phi}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ is now symmetric since its entries are $\varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. The function φ must be chosen in order to guarantee a unique solution of the interpolation problem expressed by Eq. (3.3) for each set of n distinct nodes $\mathbf{x}_i$.

This solvability condition is met in the case of a strictly positive definite function φ, which means that the associated quadratic form satisfies:

$$\mathbf{a}^T \boldsymbol{\phi}(\mathbf{x}_1, \dots, \mathbf{x}_n)\mathbf{a} > 0 \tag{3.6}$$

for any set of distinct nodes $\mathbf{x}_i$ and for any $\mathbf{a} \neq \mathbf{0}$. The condition expressed by Eq. (3.6) implies the non-singularity of the interpolation problem. Strictly positive definite functions include the Gaussian (GA) and the Inverse multiquadric (IMQ) functions reported in Table 3.1.

The interpolation with strictly positive definite radial functions therefore leads to well-posed interpolation problems for any set of distinct nodes. This important result, however, does not supply any additional information about the accuracy of the interpolation, which also depends upon the type of functions that are intended to approximate. From an engineering point of view, the exact reproduction of

Table 3.1: Most used types of RBF generating functions.

Name	**Abbreviation**	$\varphi(r)$
Multiquadric	MQ	$\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$1/\sqrt{1+(\varepsilon r)^2}$
Thin plate splines	TPS	$r^k \log r, k$ even
Gaussian	GA	$e^{-\varepsilon^2 r^2}$
Polyharmonics	PHS	r^k, k odd

constant, linear or higher degree polynomial fields must also be required, e.g., constant strain in an elastic body, steady temperature field in a differentially heated wall, translations and rotations. The use of polynomials is not only recommended but also necessary for the convergence under certain assumptions.

An expansion with only radial basis functions can not reproduce polynomial fields exactly, therefore the following augmentation of the RBF interpolant with an appended polynomial of degree P is required:

$$g(\mathbf{x}) = \sum_{i=1}^{n} a_i \varphi_i(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^{m} b_j p_j(\mathbf{x}) \tag{3.7}$$

where p_j are the basis of the polynomial expansion, b_j are the corresponding coefficients and $m = \binom{P+D}{P}$ is the number of the polynomial basis functions of degree less than or equal to P in D dimensions. For example, the basis for linear polynomials in 2D has $m = 3$ elements: $p_1 = 1, p_2 = x, p_3 = y$.

When the polynomial augmentation is employed, it is convenient to shift the nodal coordinates to their mean point $\bar{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{x}_i/n$ in order to avoid numerical instabilities when the nodes have large magnitudes $\|\mathbf{x}_i\|$. Each polynomial component $p_j(\mathbf{x})$ in Eq. (3.7) is therefore replaced by $p_j(\mathbf{x} - \bar{\mathbf{x}})$. Since this shift does not affect any interpolation property, it is omitted in the following for the sake of simplicity, while it is considered in the numerical implementations.

The interpolation system in Eq. (3.3) must also be augmented with additional conditions in order to obtain a square interpolation matrix which guarantees the polynomial reproduction. This requirement is obtained by imposing the following orthogonality conditions between the polynomial basis functions and the RBF coefficients $\mathbf{a}$:

$$\left\{ p_j(\mathbf{x}_1)\ , \ldots ,\ p_j(\mathbf{x}_n) \right\} \begin{Bmatrix} a_1 \\ \vdots \\ a_n \end{Bmatrix} = 0 \tag{3.8}$$

for $j = 1, \ldots, m$.

The final interpolation system is:

$$\left[\begin{array}{ccc|ccc} \varphi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_1-\mathbf{x}_n\|) & p_1(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{x}_n-\mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_n-\mathbf{x}_n\|) & p_1(\mathbf{x}_n) & \cdots & p_m(\mathbf{x}_n) \\ \hline p_1(\mathbf{x}_1) & \cdots & p_1(\mathbf{x}_n) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_m(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_n) & 0 & \cdots & 0 \end{array}\right] \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_m \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{3.9}$$

where the symmetry of the coefficient matrix is maintained.

The compact notation of Eq. (3.9) is:

$$\mathbf{M} \begin{Bmatrix} \mathbf{a} \\ \mathbf{b} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix} \tag{3.10}$$

where $\mathbf{0} \in \mathcal{R}^m$ is a zero column-vector and $\mathbf{M}$ is the interpolation matrix.

The functions φ to be employed for a well-posed RBF interpolation with the polynomial augmentation can be found in the set of strictly conditionally positive definite functions of order $P+1$, which is a superset of the strictly positive definite functions. Strictly conditionally positive definite functions of order $P+1$ satisfy the positive definite condition of Eq. (3.6) under the orthogonality constraints of Eq. (3.8), thus guaranteeing a well-posed RBF interpolation with polynomials of degree P for each set of distinct nodes. Strictly conditionally positive definite functions of order $P+1$ are also strictly conditionally positive definite of any higher order.

Strictly conditionally definite functions of order 1, i.e., RBF interpolation with an appended constant, include Polyharmonics (PHS) with $k = 1$ (distance), Thin plate splines (TPS) with $k = 0$ ($\log r$), and the Multiquadric (MQ), and are reported in Table 3.1. The corresponding RBFs grow with the distance from the corresponding node, which is a counterintuitive behaviour of the basis functions employed for interpolation. Contrary to these RBFs, the Inverse multiquadric and the Gaussian RBFs decrease with the distance from the corresponding node.

A comprehensive and practical analysis of the theory of radial basis functions can be found in [26].

3.1.3. RBF interpolation with boundary conditions

The interpolation system in Eq. (3.9) solves the classic scattered data interpolation problem defined in Subsection 3.1.1, where a set of values f_i are interpolated at the nodes. Since the RBF interpolation is intended to be employed in the solution of PDEs, we introduce a different interpolation condition which takes account of the boundary conditions (BCs).

Boundary conditions are imposed on the interpolant g along the boundary Γ:

$$\mathcal{B}_{\mathcal{H}}\big(g(\mathbf{x})\big) = \bar{f}(\mathbf{x}) \qquad \text{on } \Gamma \tag{3.11}$$

where $\mathcal{B}_{\mathcal{H}}$ is the homogeneous part of the boundary condition and $\bar{f}$ is a known function along Γ. For example, the Robin BC $\alpha g + \beta \partial g/\partial \mathbf{n} = \gamma$ is expressed by $\mathcal{B}_{\mathcal{H}} = \alpha + \partial/\partial \mathbf{n}$ and $\bar{f} = \gamma$, where $\mathbf{n}$ is the unit outward vector.

Since we suppose that the node distribution is conformal to the boundary, a certain number $n_B < n$ of nodes can lie on Γ. For these boundary nodes $\hat{\mathbf{x}}_k, k = 1, \ldots, n_B$, the interpolation condition expressed by Eq. (3.1) is dropped and replaced by the corresponding BC:

$$\mathcal{B}_{\mathcal{H}}\big(g(\mathbf{x})\big)_{\mathbf{x}=\hat{\mathbf{x}}_k} = \sum_{i=1}^{n} a_i \varPsi_i(\hat{\mathbf{x}}_k) + \sum_{j=1}^{m} b_j \varPi_j(\hat{\mathbf{x}}_k) = \bar{f}(\hat{\mathbf{x}}_k) \tag{3.12}$$

where $\varPsi_i(\mathbf{x}) = \mathcal{B}_{\mathcal{H}}\big(\varphi_i(\|\mathbf{x} - \mathbf{x}_i\|)\big)$ and $\varPi_j(\mathbf{x}) = \mathcal{B}_{\mathcal{H}}\big(p_j(\mathbf{x})\big)$ are given by the application of the homogeneous part of the BC on the basis functions, and are known functions.

Eq. (3.12) is made valid on each of the n_B boundary nodes, which are supposed to follow the remaining $n_I = n - n_B$ internal nodes:

$$\left[\begin{array}{ccc|ccc}
\varphi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_1 - \mathbf{x}_n\|) & p_1(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\varphi(\|\mathbf{x}_{n_I} - \mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_{n_I} - \mathbf{x}_n\|) & p_1(\mathbf{x}_{n_I}) & \cdots & p_m(\mathbf{x}_{n_I}) \\
\hline
\varPsi_1(\hat{\mathbf{x}}_1) & \cdots & \varPsi_n(\hat{\mathbf{x}}_1) & \varPi_1(\hat{\mathbf{x}}_1) & \cdots & \varPi_m(\hat{\mathbf{x}}_1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\varPsi_1(\hat{\mathbf{x}}_{n_B}) & \cdots & \varPsi_n(\hat{\mathbf{x}}_{n_B}) & \varPi_1(\hat{\mathbf{x}}_{n_B}) & \cdots & \varPi_m(\hat{\mathbf{x}}_{n_B}) \\
\hline
p_1(\mathbf{x}_1) & \cdots & p_1(\mathbf{x}_n) & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
p_m(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_n) & 0 & \cdots & 0
\end{array}\right]
\begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_m \end{Bmatrix}
=
\begin{Bmatrix} f_1 \\ \vdots \\ f_{n_I} \\ \bar{f}_1 \\ \vdots \\ \bar{f}_{n_B} \\ 0 \\ \vdots \\ 0 \end{Bmatrix}
\tag{3.13}$$

whose compact form is still in the form of Eq. (3.10), where the last n_B components of the vector $\mathbf{f}$ are now the known values $\bar{f}_1, \ldots, \bar{f}_{n_B}$.

3.1.4. Derivatives approximation with RBFs

Any required partial derivative $\mathcal{D}$ is obtained by differentiating the interpolation expansion in Eq. (3.7):

$$\mathcal{D}\big(g(\mathbf{x})\big) = \sum_{i=1}^{n} a_i \Psi_i(\mathbf{x}) + \sum_{j=1}^{m} b_j \Pi_j(\mathbf{x}) = \boldsymbol{\Psi}(\mathbf{x})^T \mathbf{a} + \boldsymbol{\Pi}(\mathbf{x})^T \mathbf{b} \tag{3.14}$$

where $\varPsi_i(\mathbf{x}) = \mathcal{D}\big(\varphi_i(\|\mathbf{x}-\mathbf{x}_i\|)\big)$ and $\varPi_j(\mathbf{x}) = \mathcal{D}\big(p_j(\mathbf{x})\big)$ are the derivatives of the basis functions, which are known functions and are organized in column vectors $\boldsymbol{\Psi}(\mathbf{x})$ and $\boldsymbol{\Pi}(\mathbf{x})$, respectively.

When the interpolation system defined by Eq. (3.9) (if no boundary nodes) or Eq. (3.13) (with boundary nodes) is solved for the RBF coefficients a_i and polynomial coefficients b_j, the required derivative can therefore be directly evaluated using Eq. (3.14). This approach can be employed for the reconstruction of the derivatives from given values of the function f.

A formal expression for the required derivative can still be computed by solving Eq. (3.10) for the coefficients $\mathbf{a}$ and $\mathbf{b}$ even if the coefficients are not known:

$$\begin{Bmatrix}\mathbf{a}\\ \mathbf{b}\end{Bmatrix} = \mathbf{M}^{-1}\begin{Bmatrix}\mathbf{f}\\ \mathbf{0}\end{Bmatrix} \tag{3.15}$$

which is then used in Eq. (3.14):

$$\mathcal{D}\big(g(\mathbf{x})\big) = \begin{Bmatrix}\boldsymbol{\Psi}(\mathbf{x})\\ \boldsymbol{\Pi}(\mathbf{x})\end{Bmatrix}^T \begin{Bmatrix}\mathbf{a}\\ \mathbf{b}\end{Bmatrix} = \begin{Bmatrix}\boldsymbol{\Psi}(\mathbf{x})\\ \boldsymbol{\Pi}(\mathbf{x})\end{Bmatrix}^T \mathbf{M}^{-1}\begin{Bmatrix}\mathbf{f}\\ \mathbf{0}\end{Bmatrix} \tag{3.16}$$

Eq. (3.16) can be recast in the following form in order to clarify the role of the derivatives of the basis functions and the role of the function values $\mathbf{f}$:

$$\mathcal{D}\big(g(\mathbf{x})\big) = \left((\mathbf{M}^T)^{-1}\begin{Bmatrix}\boldsymbol{\Psi}(\mathbf{x})\\ \boldsymbol{\Pi}(\mathbf{x})\end{Bmatrix}\right)^T \begin{Bmatrix}\mathbf{f}\\ \mathbf{0}\end{Bmatrix} = \mathbf{d}(\mathbf{x})^T\begin{Bmatrix}\mathbf{f}\\ \mathbf{0}\end{Bmatrix} \tag{3.17}$$

which is the sought expression for the required derivative as a function of the values $\mathbf{f}$ at the scattered nodes, which comprehends the contributions of possible boundary nodes.

In the case of no boundary nodes, the column vector $\mathbf{d}(\mathbf{x})$ represents the vector of the shape functions for the derivative $\mathcal{D}$ since each of its components is multiplied by the value of the function at the corresponding node, Eq. (3.17):

$$\mathcal{D}\big(g(\mathbf{x})\big) = f_1 d_1(\mathbf{x}) + \cdots + f_n d_n(\mathbf{x}) \tag{3.18}$$

In the case of the identity operator $\mathcal{D}(g) = g$, i.e., no derivation, the components of $\mathbf{d}(\mathbf{x})$ are the actual shape functions which satisfy the interpolation conditions $d_i(\mathbf{x}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta.

The expression of the sought derivative changes if boundary nodes are considered, Eq. (3.17):

$$\mathcal{D}\big(g(\mathbf{x})\big) = f_1 d_1(\mathbf{x}) + \cdots + f_{n_I} d_{n_I}(\mathbf{x}) + \underbrace{\bar{f}_1 d_{n_I+1}(\mathbf{x}) + \cdots + \bar{f}_{n_B} d_n(\mathbf{x})}_{\text{known terms}} \tag{3.19}$$

where the known terms represent the contributions of boundary nodes.

The column vector $\mathbf{d}(\mathbf{x})$ can be directly evaluated since it is obtained from the solution of the following linear system:

$$\mathbf{M}^T \mathbf{d}(\mathbf{x}) = \begin{Bmatrix} \boldsymbol{\Psi}(\mathbf{x}) \\ \boldsymbol{\Pi}(\mathbf{x}) \end{Bmatrix} \tag{3.20}$$

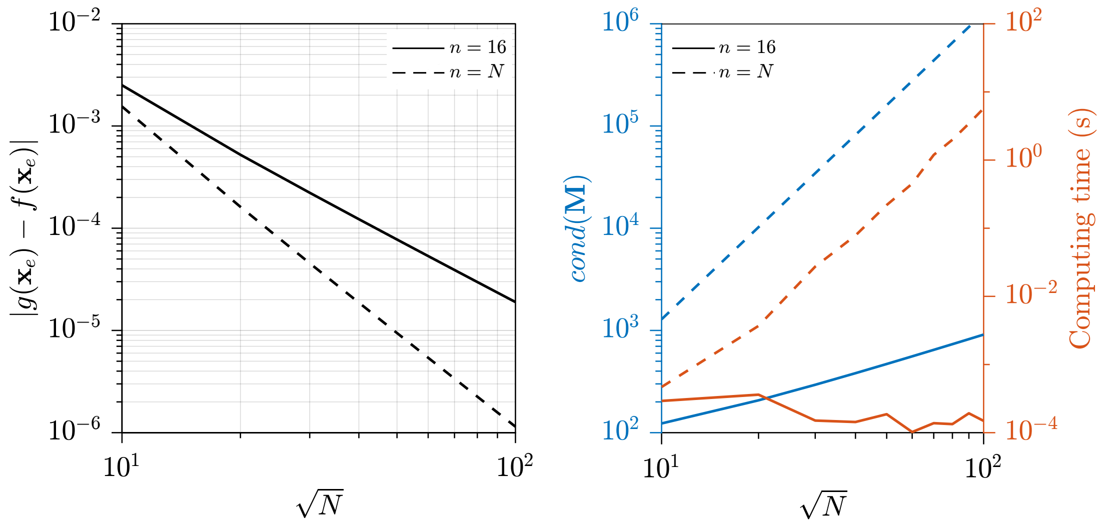
where the interpolation matrix $\mathbf{M}$ and the RHS vector of the derivatives of the basis functions are both known. The accurate solution of the system (3.20) is the main element of the RBF interpolation since good interpolation properties are often associated with ill-conditioned interpolation matrices [25, 26, 36, 62, 106]. For this purpose, in this work we employed a LDL^T factorization in the case of the symmetric interpolation matrix of Eq. (3.9) (no boundary nodes), while in the case of the non-symmetric interpolation matrix of Eq. (3.13) a Schur complement [48] is performed on the non-symmetric part of $\mathbf{M}$, followed by a LDL^T factorization on the remaining symmetric part (see Appendix B). Different techniques can be used to overcome the ill-conditioning problem [36, 106] and are discussed in the following.

3.1.5. Global versus local interpolation

The previously introduced RBF interpolation can be employed using two approaches: global and local. Given a set of N nodes, the global approach employs all the available nodes $n = N$ in the construction of the interpolant g, while in the local approach only a small number of local nodes $n \ll N$ is considered. These n local nodes represent the local support and are usually chosen between the closest nodes to the evaluation point $\mathbf{x}_e$. Therefore, in the global approach a full and large ($\approx N \times N$) interpolation system must be solved only once, while in the local approach a small ($\approx n \times n$) interpolation system has to be solved for each evaluation point.

Since the interpolation matrix in the global approach is large and full, this approach suffers from efficiency issues and ill-conditioning problems if a moderately high number of nodes is employed, while for small N it gives very accurate solutions. The employment of local interpolants greatly reduces both problems since only small interpolation matrices are involved, at the cost of slightly less accurate results [88, 105].

An example of comparison between a global and a local approach is given in Figure 3.1, where a 2D cartesian arrangements of N nodes in the unit square $[0, 1]^2$ is considered. The considered interpolant employs the distance RBFs, i.e., $\varphi(r) = r$, augmented with a constant. In the local approach the number of local nodes is $n = 16$. The evaluation point $\mathbf{x}_e$ is chosen to be the mid-point of the 4 nodes closest to the center of the unit square $x = 0.5, y = 0.5$, while the chosen



(a) (b)

Figure 3.1: Comparison between local ($n = 16$) and global ($n = N$) RBF interpolation. Error (a), condition number of the interpolation matrix and computing times (b).

function to approximate is:

$$f(x, y) = e^{-(x^2+y^2)} \tag{3.21}$$

The error curves shown in Figure 3.1a reveal a faster decay for the global approach, while the computing times[1] and the condition number of the interpolation matrix for the global approach, Figure 3.1b, exhibit a considerably larger growth when compared to the local approach. For these reasons the local approach has been preferred and it is considered from now on.

3.1.6. Multiquadric RBFs

Among the possible choices for the type of RBF to employ, Table 3.1, the Multiquadric RBF has been chosen because it is proven to be the best choice for scattered data interpolation in an extensive comparison performed by Franke [38, 39].

Hardy's Multiquadric (MQ) [45, 46] is defined by:

$$\varphi(r) = \sqrt{1 + (\varepsilon r)^2} \tag{3.22}$$

where ε is the shape factor parameter which defines the flatness of the RBF. In the limit $\varepsilon \to 0$ the RBFs become increasingly flat and the corresponding interpolant

[1]With a MATLAB implementation

without polynomial augmentation converges to the minimum degree polynomial which interpolates the data, under certain conditions on the node arrangement [25, 62]. For example, in 1D with two nodes $x_1 = 0,\ x_2 = 1$ and data $f_1,\ f_2$, the interpolation system of Eq. (3.9) is:

$$\begin{bmatrix} 1 & \sqrt{1+\varepsilon^2} \\ \sqrt{1+\varepsilon^2} & 1 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{3.23}$$

for which the corresponding interpolant $g(x)$ can be computed explicitly:

$$\begin{aligned} g(x) &= \frac{\left(f_1 - f_2\sqrt{1+\varepsilon^2}\right)\sqrt{1+(\varepsilon x)^2} + \left(f_2 - f_1\sqrt{1+\varepsilon^2}\right)\sqrt{1+\varepsilon^2(1-x)^2}}{-\varepsilon^2} \\ &= \frac{(f_1 + f_2 x - f_1 x)\varepsilon^2 + \mathcal{O}(\varepsilon^4)}{\varepsilon^2} \xrightarrow[\varepsilon\to 0]{} f_1 + (f_2 - f_1)x \end{aligned} \tag{3.24}$$

which is the linear interpolant.

Examples of MQ shape functions for different shape factors ε are depicted in Figure 3.2 for a 5×5 cartesian node arrangement with constant spacing s. The smoothness of the shape function grows as ε decreases, with the appearance of the typical oscillations due to a polynomial nature, i.e., Runge's phenomenon. When ε increases, the oscillations decreases because the interpolant loses its smooth polynomial character in favour of a sharp behaviour. Therefore it is expected to obtain better interpolation properties when using small ε, especially in the approximation of derivatives. On the other hand, the use of small shape factors involves numerical issues because of the ill-conditioned interpolation matrix due to the increasingly flat basis functions. This fact leads to the so-called trade-off principle, for which the best results are obtained using an intermediate ε value which avoids numerical issues while ensuring enough smoothness.

Nonetheless, interesting methods have been proposed for the exact evaluation of the interpolant in the limit $\varepsilon \to 0$, including the limit value $\varepsilon = 0$. These methods are based on the extension of ε to the complex plane together with a Contour-Padè algorithm [36], or on the use of vector-valued rational approximations [106]. However, the employing of these methods has not been investigated and a proper choice of the shape factor has been preferred.

Influence of RBF parameters

It is of great importance to study the effects of the different RBF parameters on the accuracy and stability of the interpolation. These parameters include the number n of interpolation nodes, the degree P of the augmented polynomial, the MQ shape factor ε and the number n_B of boundary nodes.

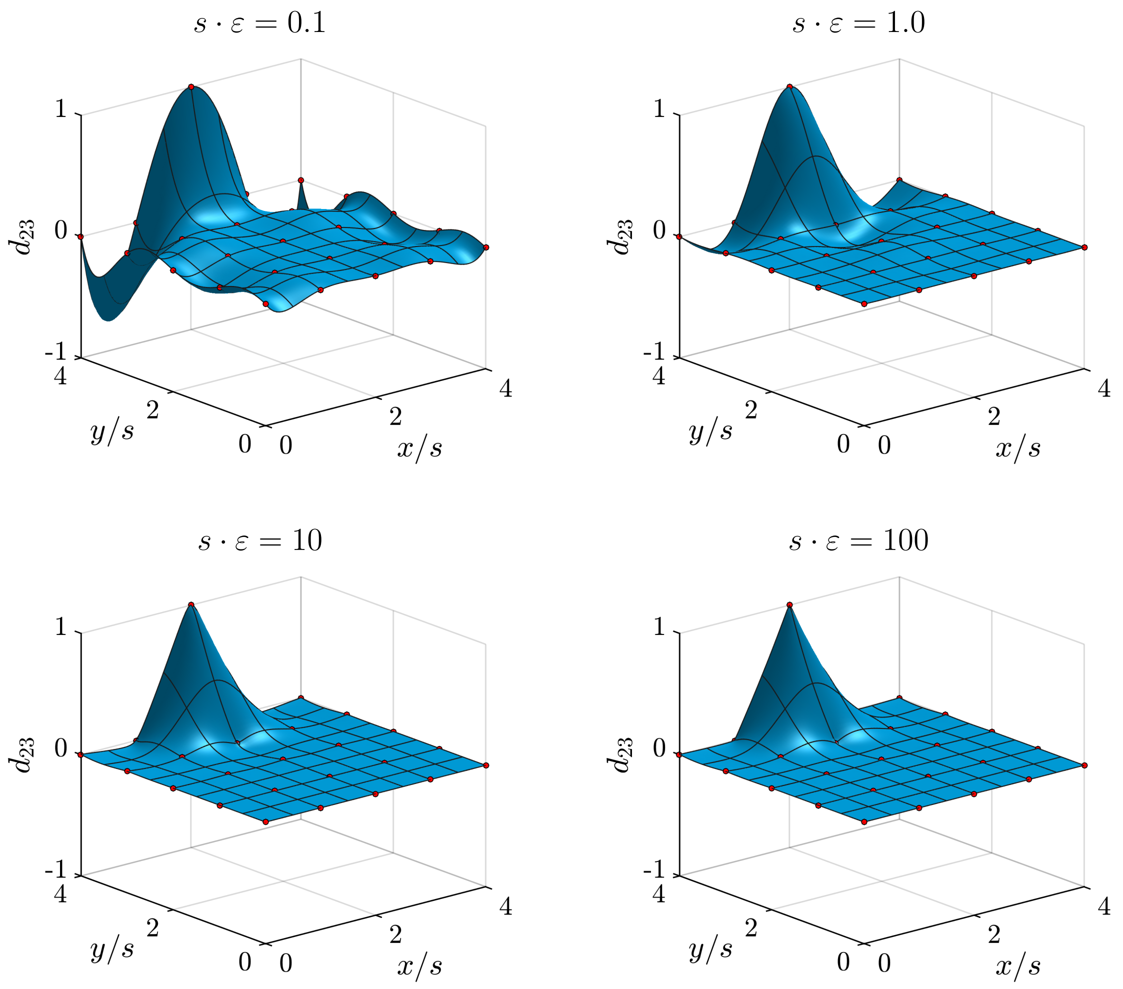


Figure 3.2: Shape function d_{23} for the node $\mathbf{x}_{23} = (2s, 4s)^T$. s is the cartesian spacing.

Influence of n, P, ε. The influence of these parameters is studied by carrying out different numerical experiments using the following test function in the 2D case:

$$f(x, y) = \cos(x)\cos(y) \tag{3.25}$$

The interpolation nodes are displaced in a hexagonal arrangement with spacing s which is then perturbed with random displacements in the order of $s/10$ to avoid distorted results due to symmetries. The nodes are then shifted in order to position the first node at $\mathbf{x}_1 = (\pi/8, \pi/8)^T$, while the n nodes closest to $\mathbf{x}_1$ are chosen as interpolation nodes. The error for the partial derivative $\partial/\partial x$ and for the laplacian ∇^2 at the central node $\mathbf{x}_1$ are chosen to measure the accuracy of the interpolation g.

The first set of numerical results is reported in Figure 3.3 and highlights the effects of the shape factor ε for three numbers of interpolation nodes $n = 7, 13, 19$ and different polynomial degrees $P \leq 4$, using a spacing $s = \pi/64$. The choice of these particular numbers of interpolation nodes is due to the hexagonal arrangement, Figure 2.19, where each node has 6 nearest neighbours at distance s, 6 more neighbours at distance $\sqrt{3}s$ and 6 more neighbours at distance $2s$, corresponding to the choices $n = 7, 13, 19$, respectively, which includes the central node. For these values $n = 7, 13, 19$ the maximum allowable degree for the augmented polynomial is $P = 2, 3$ and 4, respectively.

The most evident element is the appearance of numerical instabilities for small ε due to the ill-conditioned interpolation matrix, as expected. The more the ε gap between the instabilities and the reaching of a sufficiently small error, the more reliable a combination (n, P) is. In the case $n = 7$ and considering the partial derivative, the choice of no polynomial augmentation or $P = 0$ is not reliable since a small error is reached near the instable region, while $P = 1, 2$ yield small errors for almost each ε. In the case of the laplacian, the only reliable choice is $P = 2$. The choice $P > 2$ is not possible since the number of monomials $m = \binom{P+2}{P}$ is larger than the number of nodes n.

The same principle applies to the case $n = 13$ where $P = 3$ is the best choice for both errors, while $P = 2$ is also a reasonable choice. In the case $n = 19$ the best choice is again obtained for the highest degree $P = 4$ which can be employed, followed by $P = 3$.

The second set of numerical results is reported in Figure 3.4 where convergence tests are carried out by gradually reducing the spacing s starting from $s_0 = \pi/4$. A non-stationary interpolation is employed, i.e., the shape factor $\varepsilon = 2$ is kept constant and it is not rescaled with the spacing s. The non-stationary interpolant maintains its smoothness regardless of the spacing s, allowing the convergence of the process. Ill-conditioning instabilities arise when the spacing decreases under a certain threshold value, precluding possible applications where a small spacing is required.

In the case $n = 7$ for the partial derivative, the choices of no polynomial augmentation and $P = 0$ perform equally, as well as $P = 1$ and $P = 2$. Instabilities appear below the threshold spacing for which $s \cdot \varepsilon = 2s \approx 3 \cdot 10^{-3}$ in the corresponding graph of Figure 3.3. The order of accuracy p_E, defined by an error proportional to s^{p_E}, is 2 for each P. In the case of the laplacian, p_E is also approximately 2 except for $P = 2$ which shows a slower decrease, but with smaller errors than any other P until the threshold spacing is reached. Similar behaviours are found in the remaining cases, where the increase of P is followed by a reduction of the absolute value of both errors without an increase in the order p_E. For example, the case $n = 19$ is characterized by $p_E \approx 4$ for the partial derivative and $p_E \approx 3$ for the laplacian, regardless of the employed polynomial degree P. Approximately,

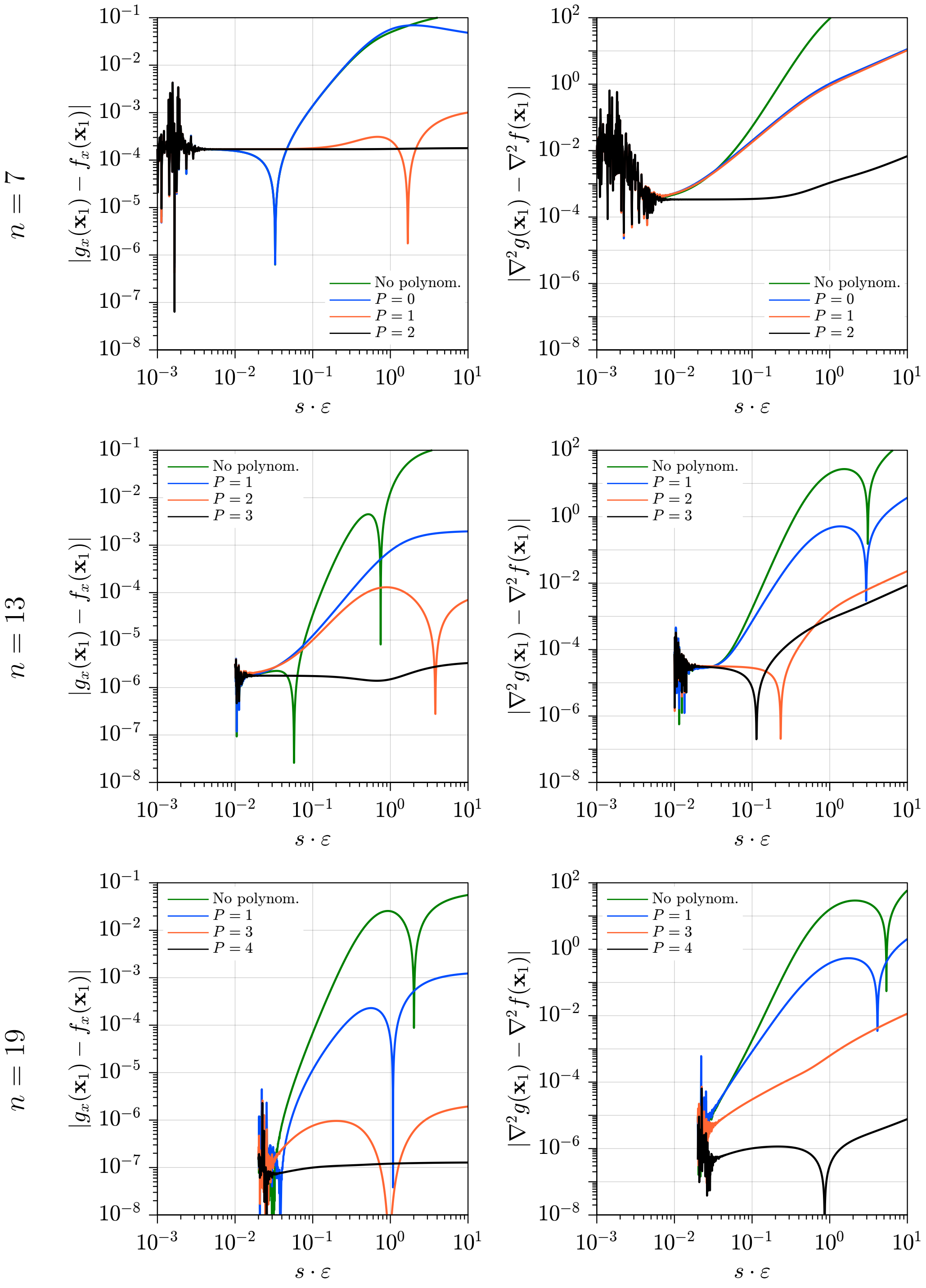


Figure 3.3: Influence of the shape factor ε on the interpolation error for the partial derivative w.r.t. x (left) and for the laplacian (right), 2D case, $s = \pi/64$.

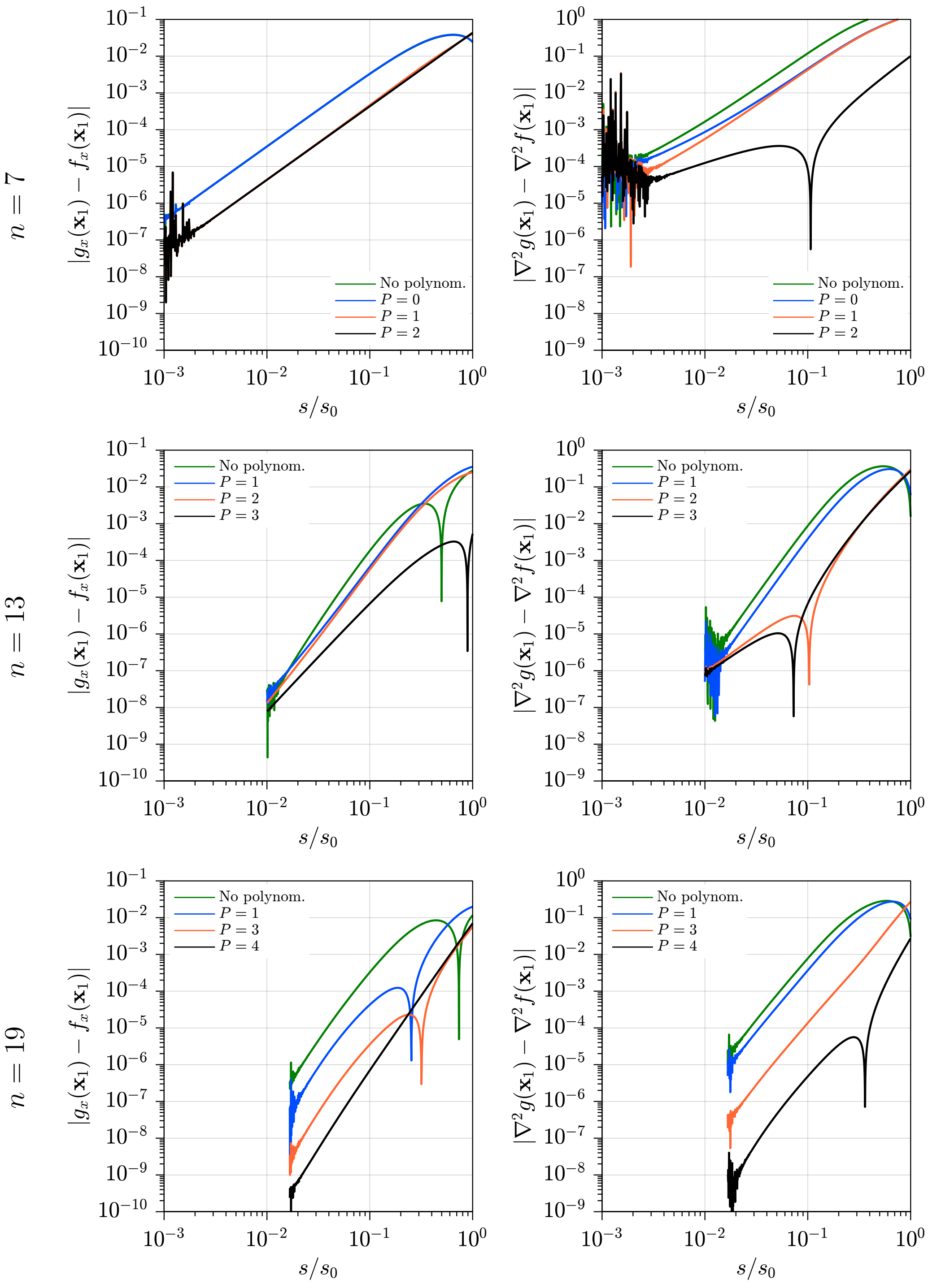


Figure 3.4: Convergence curves for the partial derivative w.r.t. x (left) and for the laplacian (right), 2D case, non-stationary interpolation ($\varepsilon = 2$).

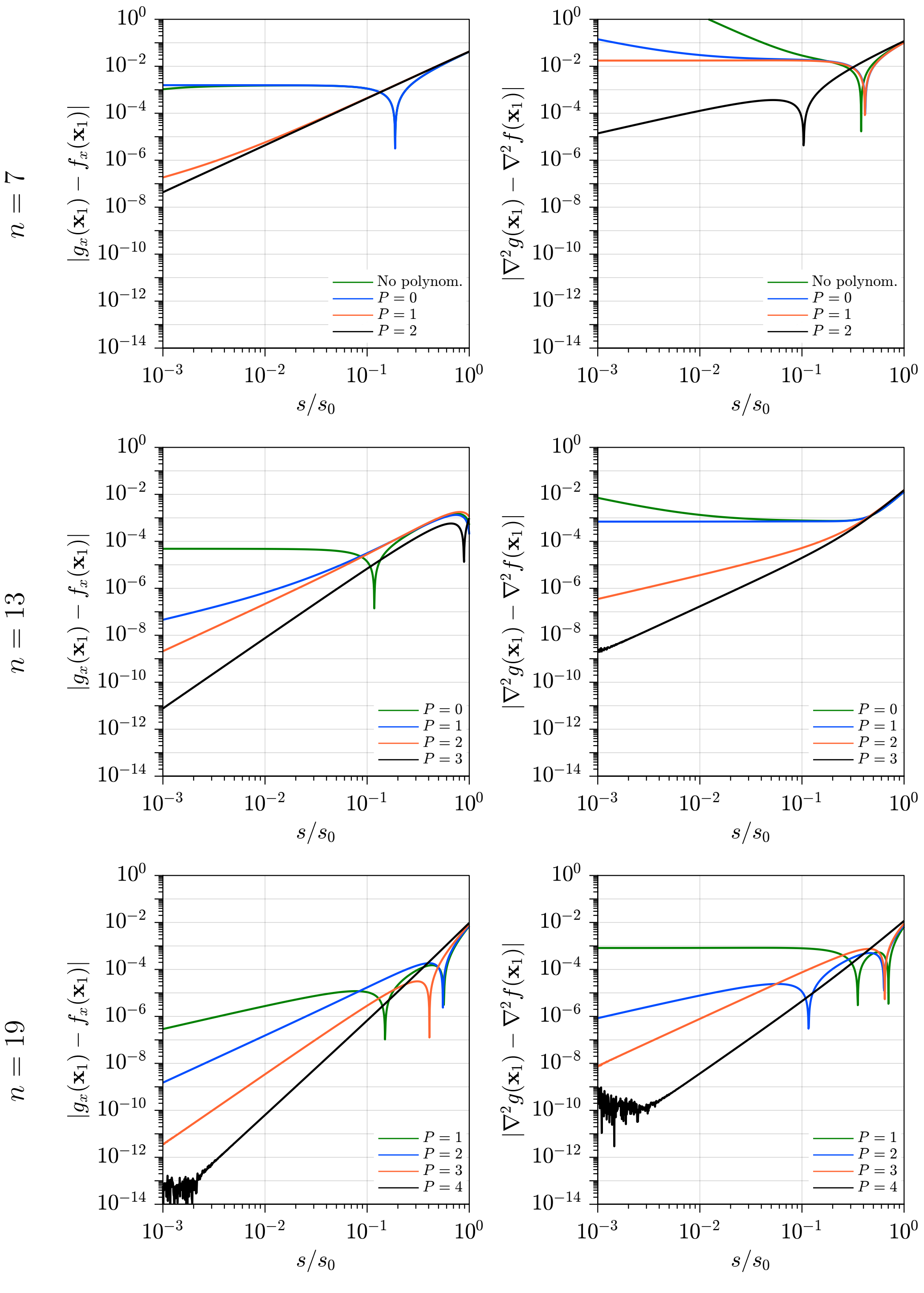


Figure 3.5: Convergence curves for the partial derivative w.r.t. x (left) and for the laplacian (right), 2D case, stationary interpolation ($s \cdot \varepsilon = 0.1$).

p_E increases in steps of 1 going from $n = 7$ to $n = 13$ to $n = 19$ for both the first derivative and the laplacian.

The third set of numerical results is reported in Figure 3.5 where convergence tests are carried out with a stationary interpolation. In stationary interpolation the shape factor is rescaled with the spacing as $s \cdot \varepsilon = constant$ in order to avoid the instabilities arising for small spacing s. Therefore, the smaller the spacing s, the larger the shape factor and the sharper the MQ basis functions. This can lead to stagnation errors, i.e., errors failing to decrease to zero under continuing spacing reduction [32], as reported in the case of $n = 7$ for low polynomial degrees P. This phenomenon occurs because the smooth function which is interpolated can not be approximated by increasingly sharper MQ basis functions, which may also cause the divergence in the approximation of the derivatives as reported in the case of the laplacian for $n = 7$ and no polynomial augmentation or $P = 0$. However, this problem can be solved by the polynomial augmentation if $P \geq 1$ in the case of the partial derivative and $P \geq 2$ in the case of the laplacian, as shown in Figure 3.5.

The constant for the shape factor rescaling is chosen to be $s \cdot \varepsilon = 0.1$ since it is a reasonably small value while it is sufficiently larger than the threshold value at which instabilities occur for all the cases $n = 7, 13, 19$, Figure 3.3. The convergence curves shown in Figure 3.5 highlight the increase of the order of accuracy p_E when P is increased, while the instabilities for small spacing are no longer present (the instabilities for $n = 19$ and $P = 4$ are due to machine precision errors in the final computation of the derivatives and are not due to an ill-conditioned interpolation matrix).

When $P \geq 2$, the order of accuracy is $p_E = P$ for the partial derivative and $p_E = P - 1$ for the laplacian, as expected.

In the 3D case, the test function is:

$$f(x, y, z) = \cos(x)\cos(y)\cos(z) \tag{3.26}$$

The interpolation nodes are displaced in a HCP node arrangement (see Section 2.1) with spacing s and perturbed with random displacements in the order of $s/10$. The nodes are then shifted in order to position the first node at $\mathbf{x}_1 = (\pi/8, \pi/8, \pi/8)^T$, while the n nodes closest to $\mathbf{x}_1$ are chosen as interpolation nodes. Similarly to the 2D case, the error for the partial derivative $\partial/\partial x$ and for the laplacian ∇^2 at the central node $\mathbf{x}_1$ are chosen to measure the accuracy of the interpolation g.

The chosen values for the number of neighbours is $n = 13, 21, 39$. These choices are due to the HCP node arrangement where each node has 12 nearest neighbours at distance s, 8 more neighbours at distance less than $\sqrt{8/3}s$ (6 at distance $\sqrt{2}s$ and 2 at distance $\sqrt{8/3}s$) and 18 more neighbours at distance $\sqrt{3}s$, corresponding to the choices $n = 13, 21, 39$, respectively, which includes the central node. For

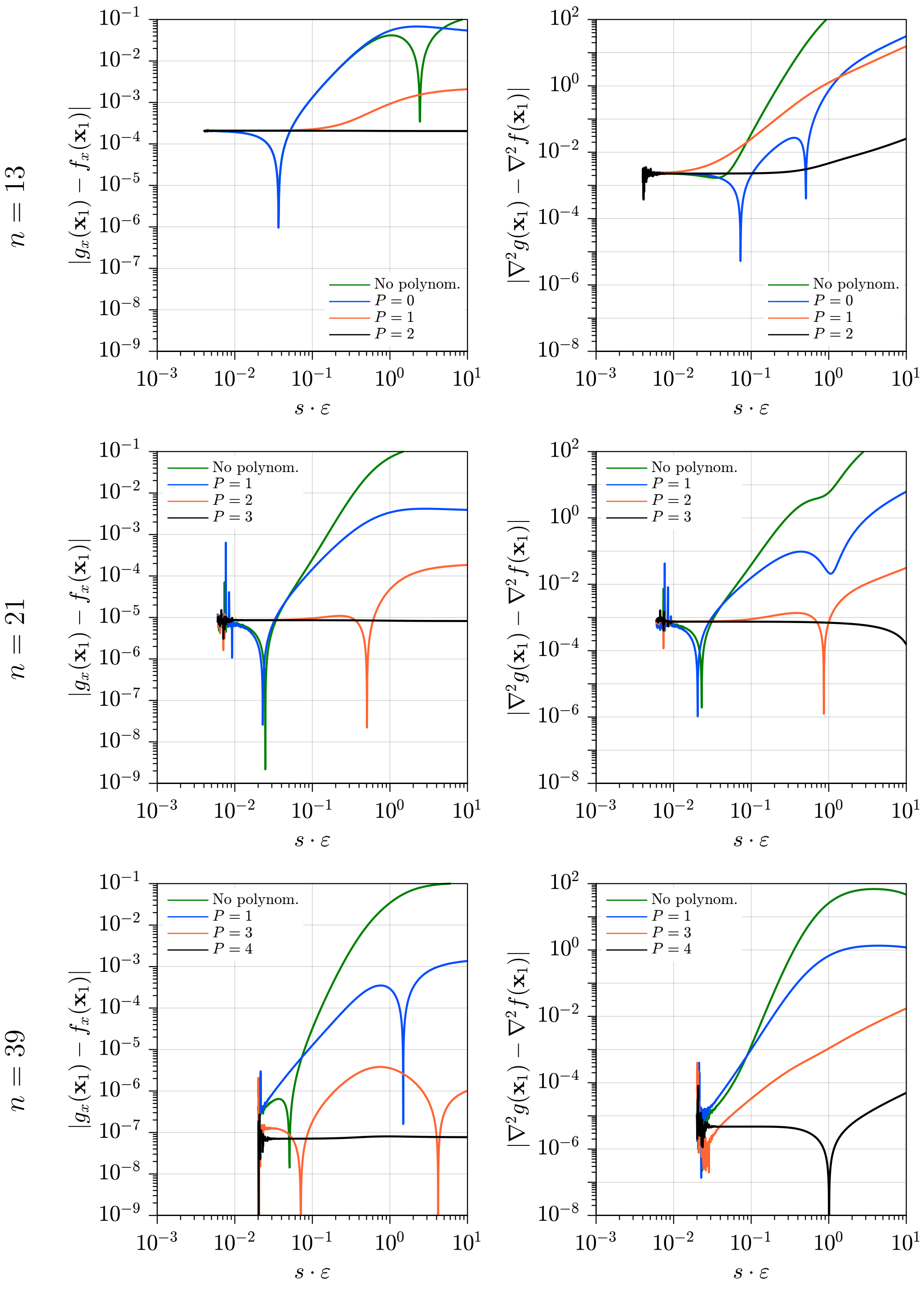


Figure 3.6: Influence of the shape factor ε on the interpolation error for the partial derivative w.r.t. x (left) and for the laplacian (right), 3D case, $s = \pi/64$.

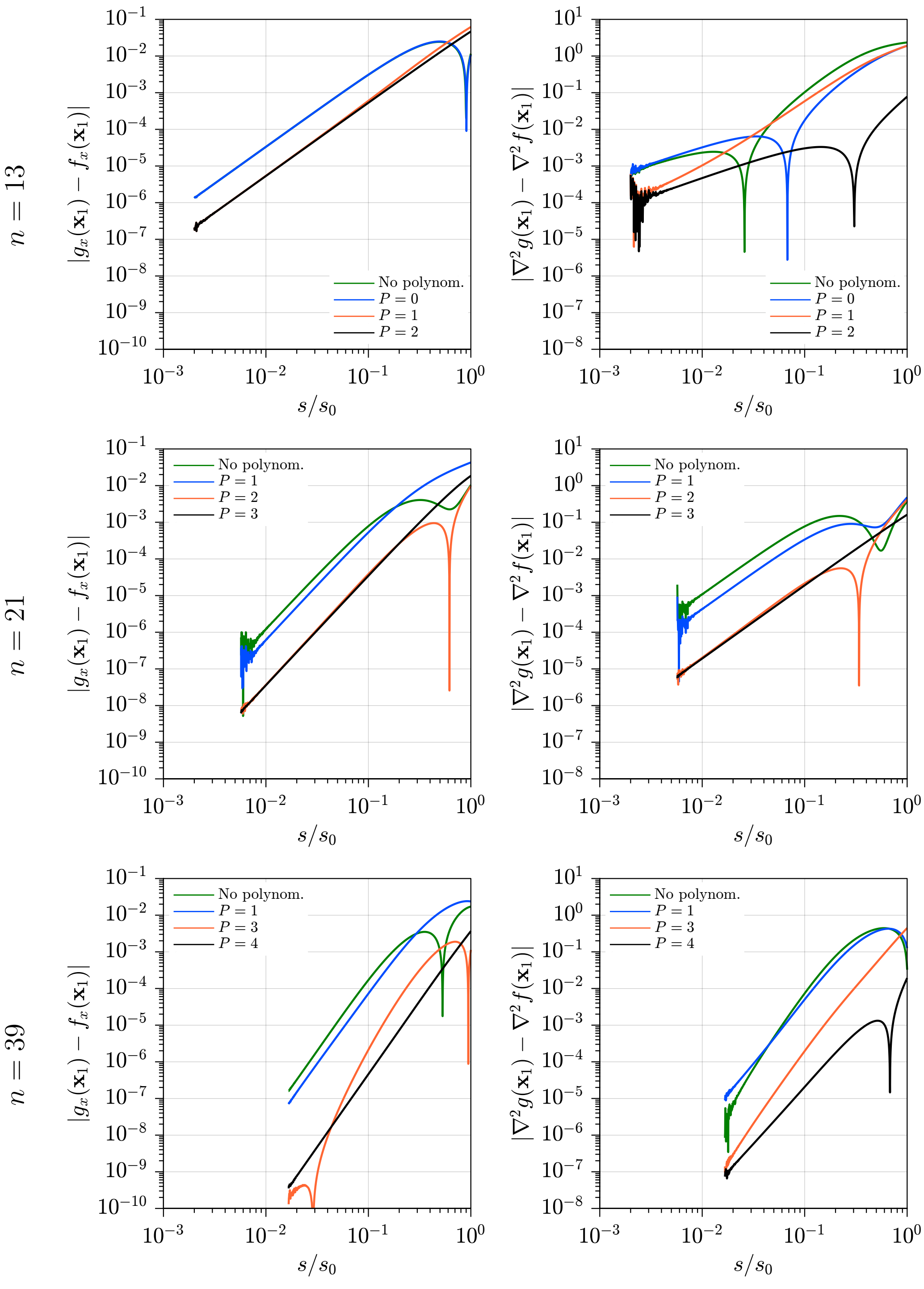


Figure 3.7: Convergence curves for the partial derivative w.r.t. x (left) and for the laplacian (right), 3D case, non-stationary interpolation ($\varepsilon = 2$).

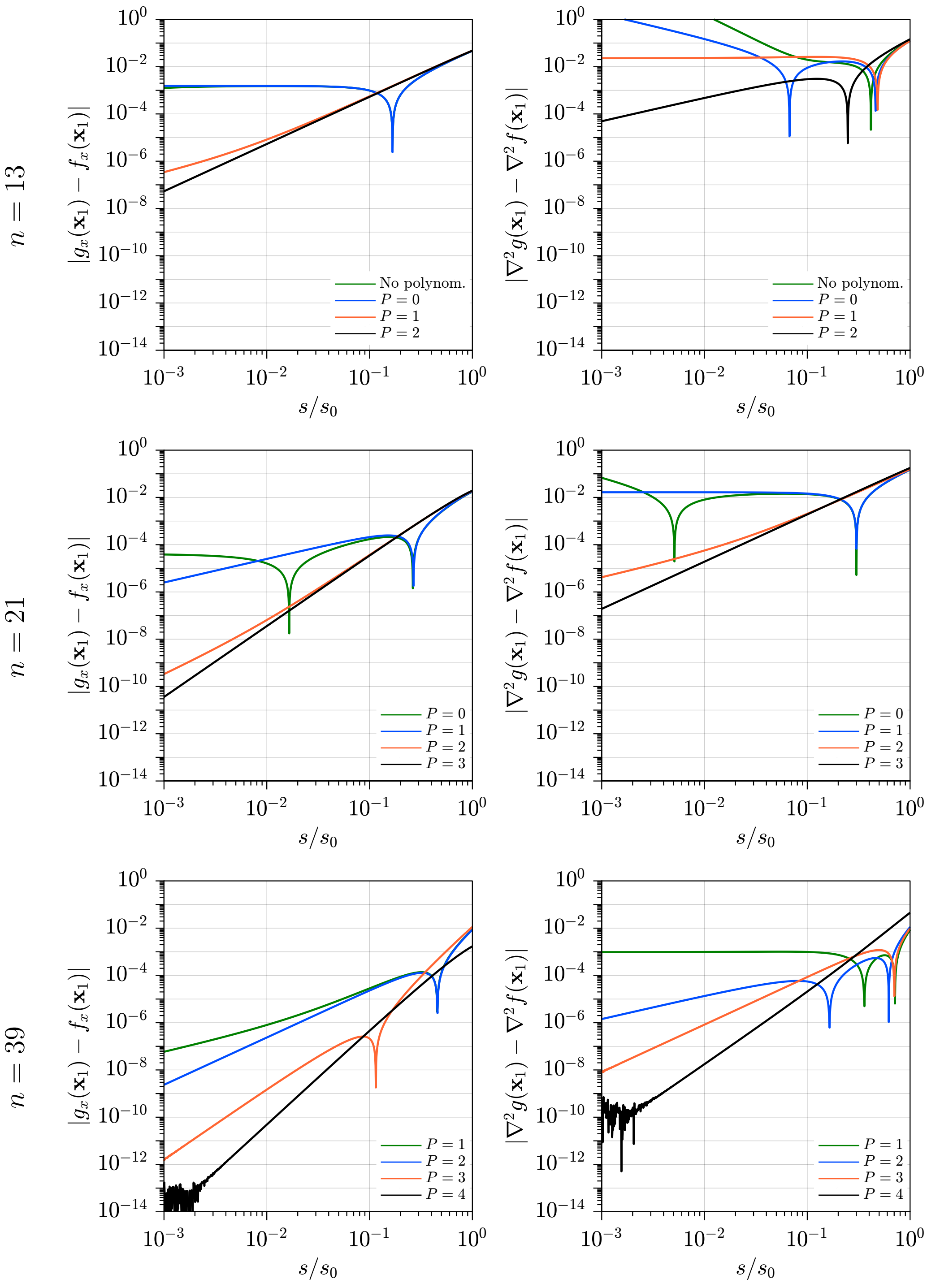


Figure 3.8: Convergence curves for the partial derivative w.r.t. x (left) and for the laplacian (right), 3D case, stationary interpolation ($s \cdot \varepsilon = 0.1$).

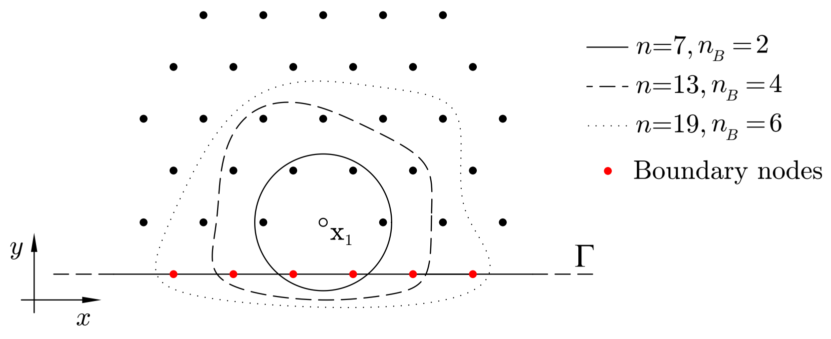


Figure 3.9: Interpolation nodes (local support) near the boundary Γ.

these values $n = 13, 21, 39$ the maximum allowable degree for the augmented polynomial is $P = 2, 3$ and 4, respectively.

The influence of the shape factor is reported in Figure 3.6, while the results of the convergence tests are reported in Figures 3.7 and 3.8 for the non-stationary and the stationary interpolations, respectively. The remaining parameters are the same of the 2D case: spacing $s = \pi/64$ in the study of the influence of ε, initial spacing $s_0 = \pi/4$ in the convergence tests, $\varepsilon = 2$ in the non-stationary interpolation, $s \cdot \varepsilon = 0.1$ in the stationary interpolation.

The results are very similar to the 2D ones in the corresponding cases: in the non-stationary convergence, Figure 3.7, the order of accuracy p_E increases approximately in steps of 1 going from $n = 7$ to $n = 13$ to $n = 19$ for both the first derivative and the laplacian, while in the stationary convergence, Figure 3.8, the order of accuracy for $P \geq 2$ is $p_E = P$ for the partial derivative and $p_E = P - 1$ for the laplacian, as expected.

Influence of n_B. In the 2D case the influence of the number of boundary nodes n_B is studied by considering the same numerical setup employed in the previous 2D studies where the test function f was expressed by Eq. (3.25). The employed node arrangement is depicted in Figure 3.9 where the evaluation node $\mathbf{x}_1$ is near the boundary Γ, while the n_B boundary nodes are also chosen between the nearest neighbours of $\mathbf{x}_1$. Random displacements in the range $s/10$ are again applied to this hexagonal arrangement. Since no nodes are available outside the boundary, more interpolation nodes have to be considered in the opposite direction and the local support is neither symmetric nor centered around $\mathbf{x}_1$ any longer.

The chosen numbers of interpolation nodes are again $n = 7, 13, 19$ and the corresponding polynomial degrees are chosen to be $P = 2, 3, 4$, respectively. Dirichlet and Neumann BCs have been chosen as meaningful conditions at the boundary nodes. Dirichlet BCs do not affect the interpolation system without BCs expressed

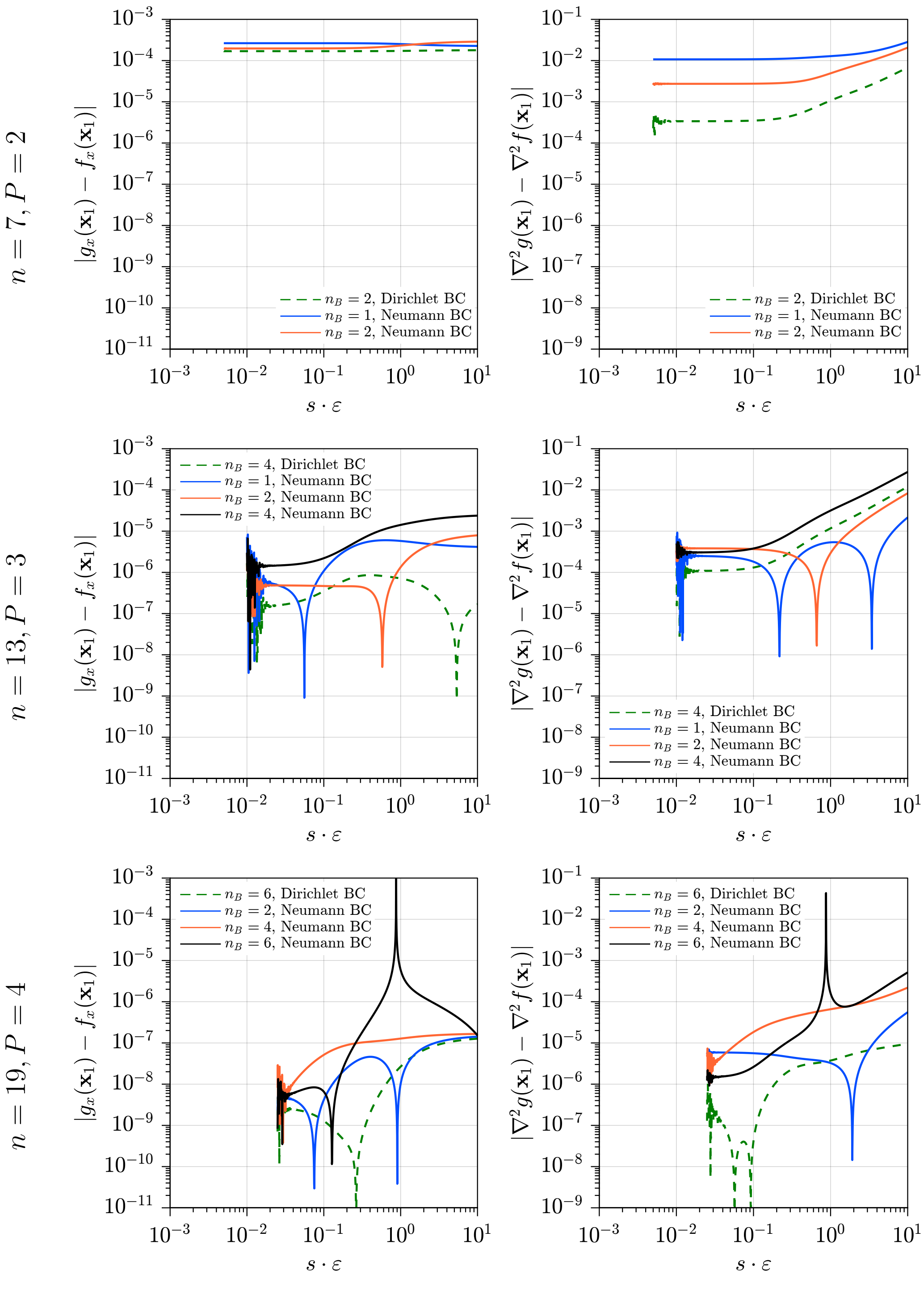


Figure 3.10: Influence of the number of boundary nodes n_B on the interpolation error for the partial derivative w.r.t. x (left) and for the laplacian (right), 2D case, $s = \pi/64$.

by Eq. (3.9) since the Dirichlet condition is expressed by the identity operator $\mathcal{B}_{\mathcal{H}}(g) = g$ in Eq. (3.11), matching the canonical interpolation condition of Eq. (3.1). Neumann BCs are expressed by $\mathcal{B}_{\mathcal{H}} = -\partial/\partial y$, $\bar{f} = -\partial f/\partial y$ in Eq. (3.11), for which the accuracy of the interpolation is altered with respect to the canonical interpolation conditions.

Similarly to the previous studies, the influence of the number n_B of boundary nodes on the errors is reported in Figure 3.10 for both the partial derivative with respect to x and the laplacian. As expected, in each case the errors increase when Neumann BCs are imposed. It is found that $n_B = 2$ is a good choice for each case. In the case $n = 7$ this is the most natural choice since the interpolation support remains symmetric and centered around $\mathbf{x}_1$, while in the cases $n = 13, 19$ the choice $n_B = 2$ implies an interpolation support which is increasingly shifted towards the internal part of the domain. With the choice $n_B = 2$ the reduction of accuracy due to Neumann BCs amounts to one order of magnitude in absolute value for each case, while slightly larger reductions are observed for $n_B > 2$, especially in the case $n = 19$. However, the choice $n_B > 2$ is also valid and further investigations are required for the actual applications.

In the 3D case the influence of n_B is studied by considering the same numerical setup employed in the previous 3D studies where the test function f was expressed by Eq. (3.26). The employed 3D node arrangement is the usual HCP where the evaluation node $\mathbf{x}_1$ is near the boundary Γ, which in this case is the $x - y$ plane, similarly to the 2D arrangement depicted in Figure 3.9. The n_B boundary nodes are chosen between the nearest neighbours of $\mathbf{x}_1$ and random displacements in the range $s/10$ are also applied to the nodes. Similarly to the 2D case, there are no nodes outside the boundary and therefore more interpolation nodes have to be considered in the opposite direction: the local support is neither symmetric nor centered around $\mathbf{x}_1$ any longer.

The numbers of interpolation nodes are again $n = 13, 21, 39$ and the corresponding polynomial degrees are $P = 2, 3, 4$, respectively. Dirichlet and Neumann BCs have been chosen on the boundary nodes, where Neumann BCs are expressed by $\mathcal{B}_{\mathcal{H}} = -\partial/\partial z$, $\bar{f} = -\partial f/\partial z$ in Eq. (3.11). The influence of the number n_B of boundary nodes on the errors is reported in Figure 3.11 for both the partial derivative with respect to x and the laplacian. Contrary to the 2D case, the effect of Neumann BCs is no longer monotone, i.e., errors can be larger or smaller than the case of Dirichlet BCs, depending upon the specific case. A good choice is $n_B = 3$ boundary nodes for $n = 13$ and $n = 21$, while for $n = 39$ the choice $n_B > 3$ results in smaller errors and therefore it is preferable. Again, further numerical investigations on the influence of n_B are required in the actual applications.

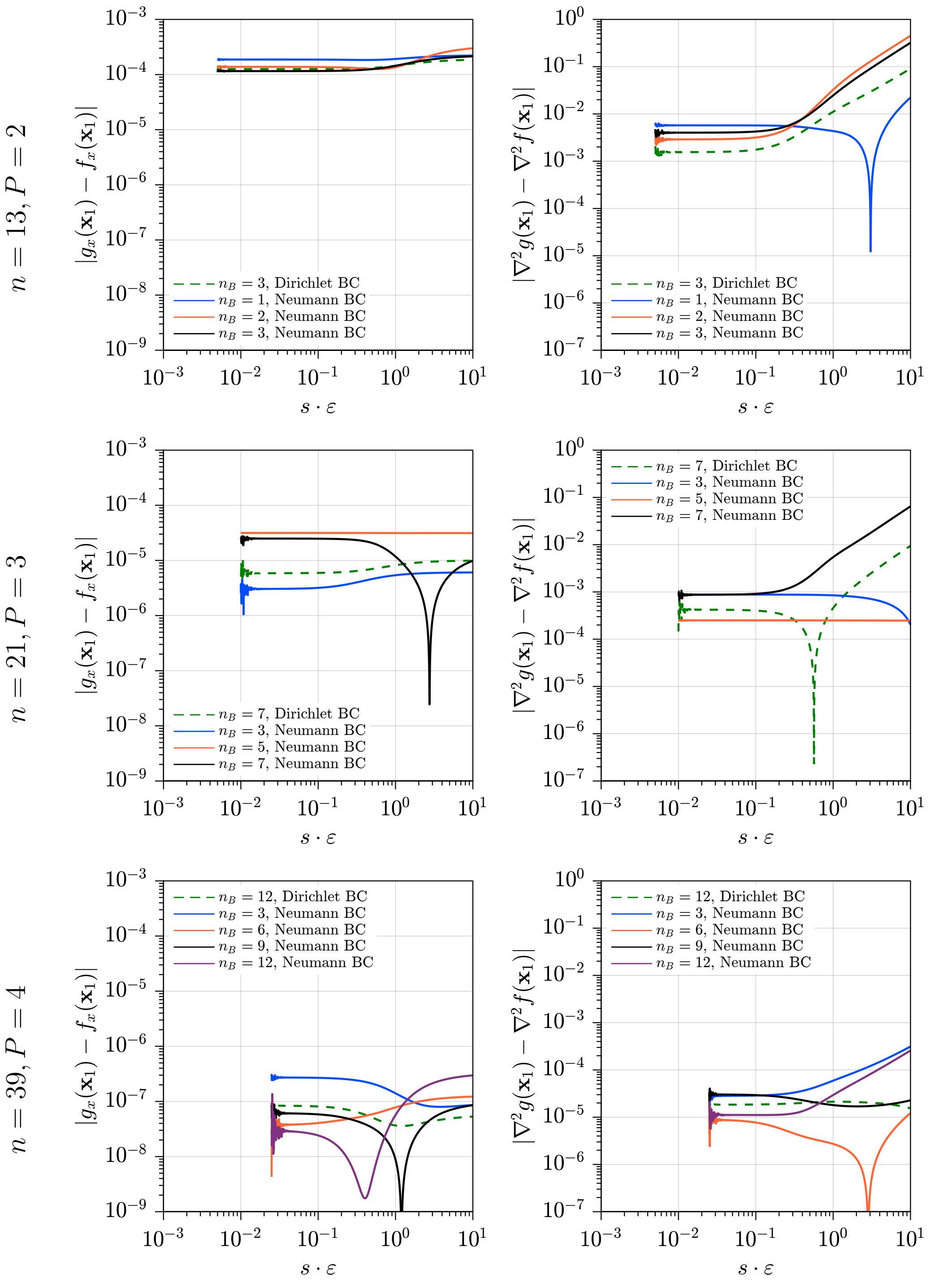


Figure 3.11: Influence of the number of boundary nodes n_B on the interpolation error for the partial derivative w.r.t. x (left) and for the laplacian (right), 3D case, $s = \pi/64$.

3.2. RBF-FD

In the previous Section the use of the local RBF interpolant $g(\mathbf{x})$, Eq. (3.7), provided an approximation for the sought derivative $\mathcal{D}\big(f(\mathbf{x})\big)$ as a function of the vector $\mathbf{f}$ of the function f at the n interpolation nodes, Eq. (3.17). If boundary nodes are present, the corresponding boundary conditions are directly imposed on the interpolant $g(\mathbf{x})$ and are thus exactly satisfied at the boundary nodes, regardless of the values $\mathbf{f}$.

In this Section the RBF interpolant and the collocation technique are employed to discretize a partial differential equation (PDE) in order to obtain a finite approximation of the PDE which can be solved numerically. The involved partial derivatives are expressed in terms of the nodal values through the RBF interpolant, while the collocation technique is used to obtain a pointwise approximation of the PDE at the nodes only, obtaining a finite number of equations N_I which equals the number of unknown nodal values.

This approach is known as RBF-generated finite difference method (RBF-FD) since it resembles the classic finite differences approach where the PDE is satisfied at the nodes only, using a polynomial interpolation through the neighbouring nodes. The RBF-FD technique therefore represents a “truly” meshless approach since approximations are required at the nodes only, without any connectivity information or integration requirement [71]. These features bring great advantages over mesh-based methods in terms of geometrical flexibility, making the RBF-FD method an effective tool for the solution of practical problems.

3.2.1. RBF collocation

Let us consider the following PDE:

$$\mathcal{D}\big(f(\mathbf{x})\big) = q(\mathbf{x}) \quad \text{in } \Omega \tag{3.27a}$$

$$\mathcal{B}_{\mathcal{H}}\big(f(\mathbf{x})\big) = \bar{f}(\mathbf{x}) \quad \text{on } \Gamma \tag{3.27b}$$

where $\mathcal{D}$ and $\mathcal{B}_{\mathcal{H}}$ are linear differential operators while q and $\bar{f}$ are known functions. Let us then consider a node distribution $X(s)$ which satisfies the prescribed spacing function s as defined by Eq. (2.1). $X(s)$ is composed by $N = N_I + N_B$ nodes: N_I internal nodes lying inside the domain Ω and N_B boundary nodes lying on the boundary Γ. The collocation technique is applied by writing Eq. (3.27a) for each of the N_I internal nodes $\mathbf{x}_i$ using the RBF approximation expressed by Eq. (3.17) for the required derivatives:

$$\mathcal{D}\big(g(\mathbf{x})\big)_{\mathbf{x}=\mathbf{x}_i} = \mathbf{d}(\mathbf{x}_i)^T \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix} = q(\mathbf{x}_i) \tag{3.28}$$

for $i = 1, \dots, N_I$ and where the global indexing of the nodes is such that the boundary nodes follow the internal nodes for the sake of simplicity. The basis functions $\mathbf{d}$ for the operator $\mathcal{D}$ and the values $\mathbf{f}$ are local quantities which refer to the local support for the node $\mathbf{x}_i$. The local support, i.e., the chosen nodes for the local RBF interpolation, is built by considering the n closest nodes to $\mathbf{x}_i$, including possible boundary nodes which are limited to be the $n_B \leq \bar{n}_B$ closest boundary nodes to $\mathbf{x}_i$. The choice $\bar{n}_B \geq n$ implies no limitations on the number of boundary nodes, while the influence of the values of n and $\bar{n}_B$ within the collocation approach is discussed in the following Subsection.

The boundary conditions expressed by Eq. (3.27b) are directly imposed on the interpolant g and the corresponding contributions are given by the last n_B components of the vector $\mathbf{f}$ in Eq. (3.28), as previously stated in Subsection 3.1.3. Recalling the explicit form given by Eq. (3.19), the collocation Eq. (3.28) becomes:

$$\sum_{j=1}^{n_I} f_j d_j(\mathbf{x}_i) = q(\mathbf{x}_i) - \sum_{k=1}^{n_B} \bar{f}_k d_{n_I+k}(\mathbf{x}_i) \tag{3.29}$$

where the RHS terms are known values while the LHS values f_j are unknown.

The compact notation of Eq. (3.29) for each internal node $\mathbf{x}_i$ can be written in the following global form, i.e., using the global indexing for the nodes:

$$\mathbf{D}\mathbf{f}_g = \mathbf{q} - \overline{\mathbf{D}}\bar{\mathbf{f}}_g \tag{3.30}$$

where $\mathbf{D} \in \mathcal{R}^{N_I \times N_I}$ is the sparse coefficients matrix, $\mathbf{q} \in \mathcal{R}^{N_I}$ is the vector of the known terms q at the n_I internal nodes and $\overline{\mathbf{D}} \in \mathcal{R}^{N_I \times N_B}$ is the sparse coefficient matrix for the BCs. $\mathbf{f}_g \in \mathcal{R}^{N_I}$ is the vector of internal nodal unknowns and $\bar{\mathbf{f}}_g \in \mathcal{R}^{N_B}$ is the vector of the known terms $\bar{f}$ of the BCs at the n_B boundary nodes, Eq. (3.27b), using the global indexing for the nodes.

Eq. (3.30) represents the final linear system which solved for $\mathbf{f}_g$ gives the nodal values of the sought solution f. For small size problems, e.g., $N < 50,000$ in 2D and $N < 10,000$ in 3D, a direct solution through a LU decomposition of the unsymmetric coefficient matrix $\mathbf{D}$ is employed, while for larger problems the BiCGSTAB iterative solver [97] coupled with an incomplete LU (ILU) factorization as preconditioner [85] are employed. In this last case the use of more accurate factorizations than ILU(0), i.e., ILU factorization with 0 level of fill-in, is found to be an effective choice for which the value of the drop tolerance for the incomplete factorization has to be optimized case by case. The Cuthill-McKee ordering [19] is used for the minimization of the bandwidth of the coefficient matrix $\mathbf{D}$. For stationary problems the relative tolerance for the residual is set to 10^{-14} in order to avoid any effect due to convergence errors, while for time-dependent problems the chosen relative tolerance is 10^{-10} for 2D and small size 3D problems; for large size 3D problems the relative tolerance for the residual is set to 10^{-8}.

3.2.2. Time stabilization

Theoretical considerations

When the RBF-FD technique is used to discretize time-dependent problems, it is important to ensure that the resulting scheme is stable when integrating in time. Numerical, non-physical instabilities may arise as a result of the coupling between the chosen time discretization scheme and the employed space discretization scheme [54]. With particular reference to CFD problems, stability requirements due to the time discretization are typically imposed on the time step Δt and are assumed to be satisfied.

Numerical instabilities may also arise due to the space discretization only, even if the stability requirements are satisfied or even in the case where the time is not discretized, i.e., by the use of the analysis of the associated eigenvalues. This means that the space discretization leads to a finite dimensional model that possesses non-physical modes which are not related to the physical behaviour of the system that is intended to simulate. Therefore these spurious modes appear regardless of the employed time discretization scheme, showing a typical exponential growth or decay in time.

In order to clarify this aspect, let us consider the following advection equation for the variable f:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0 \tag{3.31}$$

where t is the time and $\mathbf{u}$ is the advection velocity which is considered uniform and constant for the sake of simplicity. Eq. (3.31) therefore represents a transport equation which translates the initial profile $f_0(\mathbf{x}) = f(\mathbf{x}, t = 0)$ without any modification, i.e., $f(\mathbf{x}, t) = f_0(\mathbf{x} - \mathbf{u}t)$.

The application of the RBF-FD discretization to the spatial differential operator $\mathcal{D} = \mathbf{u} \cdot \nabla$ of Eq. (3.31) yields the following discrete model:

$$\frac{\partial \mathbf{f}_g}{\partial t} + \mathbf{D}\mathbf{f}_g = -\overline{\mathbf{D}}\bar{\mathbf{f}}_g \tag{3.32}$$

where the notation follows the one expressed in Eq. (3.30). If we suppose constant boundary conditions, i.e., $\bar{\mathbf{f}}_g$ is constant, Eq. (3.32) with the substitution $\mathbf{f}_g = \mathbf{f}' - \mathbf{D}^{-1}\overline{\mathbf{D}}\bar{\mathbf{f}}_g$ becomes:

$$\frac{\partial \mathbf{f}'}{\partial t} + \mathbf{D}\mathbf{f}' = \mathbf{0} \tag{3.33}$$

which is the transport equation for the deviation $\mathbf{f}'$ from the equilibrium state $\mathbf{f}_g = -\mathbf{D}^{-1}\overline{\mathbf{D}}\bar{\mathbf{f}}_g$.

The solution of Eq. (3.33) is $\mathbf{f}'(t) = e^{-\mathbf{D}t}\mathbf{f}'(0)$ and it should model a purely advective phenomenon with constant BCs. The stability of the RBD-FD discretization therefore depends upon the real part of the eigenvalues λ_i of $\mathbf{D}$. If

$\mathcal{R}e(\lambda_i) = 0$ for all i, the RBF-FD discretization does not exhibit spurious modes. If $\mathcal{R}e(\lambda_i) < 0$ for some i, the discretization is not stable because of the presence of diverging spurious modes with exponential growth in time. If $\mathcal{R}e(\lambda_i) \geq 0$ with the strict inequality for some i, the discretization is stable with some numerical diffusion because of the presence of converging spurious modes with exponential decay in time.

The RBF-FD approach as previously presented in this Chapter does not guarantee a stable discretization for each type of differential operator $\mathcal{D}$. For example, the distribution of the eigenvalues of matrix $\mathbf{D}$ obtained by the discretization of the advection Eq. (3.31) on the unit square $\Omega = [0,1]^2$ with a uniform distribution of $N = 2000$ nodes is shown in Figure 3.12a. The velocity is $\mathbf{u} = (1,0)^T$, the left vertical side is the inlet where Dirichlet BCs are imposed, Neumann BCs are imposed at the horizontal sides while the right vertical side requires no BCs since it is the outlet. The RBF-FD discretization employs $n = 30$ local interpolation nodes, polynomial degree $P = 4$ and $s \cdot \varepsilon = 0.4$.

The distribution of the eigenvalues in Figure 3.12a reveals an unstable discretization since there are eigenvalues with negative real part. The eigenvectors ψ_1 and ψ_{N_I} associated with λ_1 and λ_{N_I}, i.e., the eigenvalues with smaller and largest real part, respectively, are depicted in Figure 3.12b. λ_1 is the most unstable eigenvalue with $\mathcal{R}e(\lambda_1) = -1.84$ and the associated eigenvector ψ_1 is large near the inlet boundary where the RBF-FD stencils have an unsymmetric shape with downwind orientation, i.e., in the direction of $\mathbf{u}$, giving rise to convective instabilities. λ_{N_I} is the most stable and diffusive eigenvalue with $\mathcal{R}e(\lambda_{N_I}) = 31.99$ and the associated eigenvector ψ_{N_I} is large near the outlet boundary where the RBF-FD stencils have an unsymmetric shape with upwind orientation, i.e., in the opposite direction with respect to $\mathbf{u}$, which is responsible of the numerical diffusion.

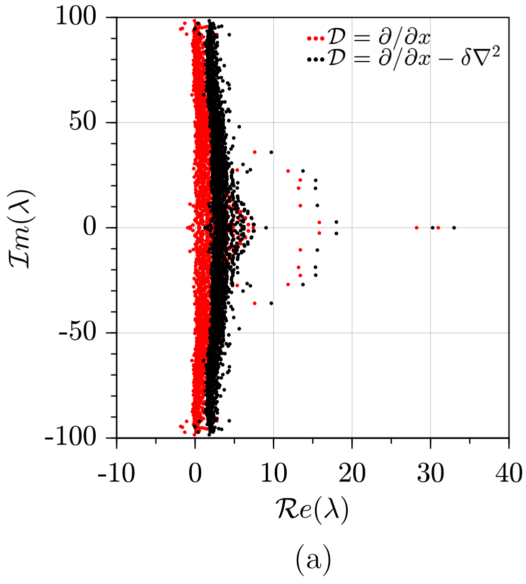
Artificial viscosity

The simplest way to stabilize an unstable discretization is to add a small amount of artificial diffusion in order to dampen the unstable modes. In the case of the advection equation Eq. (3.31), it is slightly modified as follows:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = \delta \nabla^2 f \tag{3.34}$$

where $\delta > 0$ is a small constant which has to be chosen in order to stabilize the discretization without adding unnecessary artificial diffusion, i.e., δ is the minimum value which guarantees $\mathcal{R}e(\lambda_i) \geq 0$ for all i for the RBF-FD discretization of the modified advection operator $\mathcal{D} = \mathbf{u} \cdot \nabla f - \delta \nabla^2 f$.

The distribution of the eigenvalues of the stabilized discretization is shown in Figure 3.12a. The effect of the artificial diffusion using the laplacian ∇^2 is



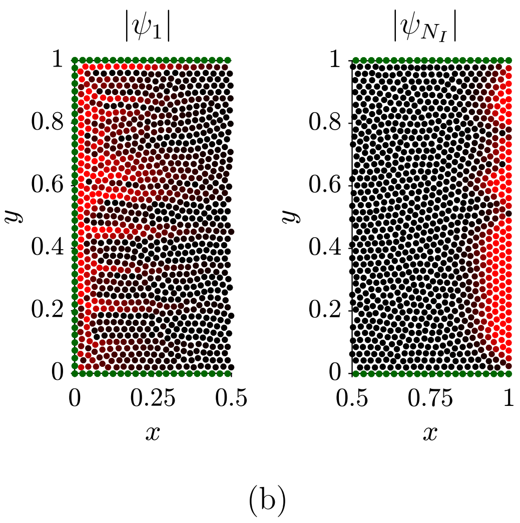


Figure 3.12: (a): eigenvalue distribution for the derivative operator (red) and for the stabilized derivative (black). (b): eigenvectors associated to the most unstable eigenvalue at the inlet (ψ_1) and to the most diffusive eigenvalue at the outlet (ψ_{N_I}); boundary nodes are coloured in green.

the shift of the eigenvalues towards the diffusive region $\mathcal{R}e(\lambda) > 0$, implying a certain amount of numerical diffusion. The amount of this artificial diffusion can be unacceptable in some case depending upon the type of equation that is intended to solve. For example, when the advection Eq. (3.31) is integrated over long time periods, the laplacian stabilization can excessively dampen the low space-frequency components of the initial profile f_0 causing large errors.

Hyperviscosity

Besides the classic laplacian diffusion, other types of artificial diffusion can be employed. An effective choice is hyperviscosity [35] which employs the powers of the laplacian ∇^{2k} instead of the laplacian ∇^2. This choice is due to the fact that the powers of laplacian are more effective in smoothing the high space-frequency components and preserving the low space-frequency components than the simple laplacian, which is the key element in the damping of unstable modes without introducing unacceptable numerical diffusion. The advection Eq. (3.31) with hyperviscosity becomes:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = \delta \nabla^{2k} f \tag{3.35}$$

In order to illustrate the smoothing properties of the hyperviscosity, let us consider Eq. (3.35) with no advection, i.e., $\mathbf{u} = \mathbf{0}$, for a harmonic component $f = e^{j\boldsymbol{\omega}\cdot\mathbf{x}/s-\lambda t}$, where $\boldsymbol{\omega}$ is the vector of spatial frequencies and s a generic spacing. The resulting decay rate λ is:

$$\lambda = \delta \frac{\|\boldsymbol{\omega}\|_2^{2k}}{s^{2k}}(-1)^{k+1} \tag{3.36}$$

Eq. (3.36) states that the larger k, the faster the decay of high space-frequency components with respect to the spacing s, i.e., $\|\boldsymbol{\omega}\|_2 = \mathcal{O}(1)$, and the slower the decay of low space-frequency components, i.e., $\|\boldsymbol{\omega}\|_2 \ll 1$. This implies very low numerical diffusion for the length scales which carry physical information and should not be damped, while non-physical spurious modes with frequency comparable to the spacing s are effectively damped. The laplacian diffusion is obtained when $k = 1$, for which the decay is proportional to $\|\boldsymbol{\omega}\|_2^2$ which shows that low space-frequency components $\|\boldsymbol{\omega}\|_2 \ll 1$ sustain a non-negligible damping.

Since we want a positive decay λ to be independent upon the spacing s for a given $\boldsymbol{\omega}$, from Eq. (3.36) we can write:

$$\bar{\delta}_k := \delta \frac{(-1)^{k+1}}{s^{2k}} > 0 \quad \Rightarrow \quad \delta = \bar{\delta}_k s^{2k}(-1)^{k+1} \tag{3.37}$$

where $\bar{\delta}_k > 0$ is the specific amount of artificial hyperviscosity and depends upon the problem and the RBF-FD parameters but not upon the spacing s. Eq. (3.37) states that the amount δ of artificial hyperviscosity decreases to zero with order $2k$ under nodal refining, i.e., increasing the number of nodes. This is another strength of hyperviscosity with $k > 1$ since the introduced artificial diffusion decreases rapidly to zero when the nodal spacing is reduced.

The formulas for the calculation of the powers of the laplacian for the multiquadric RBF are reported in Appendix C.

3.2.3. Test cases

In order to assess the numerical properties of the RBF-FD approach for the solution of 2D/3D CFD problems, several stationary and non-stationary tests are carried out. The chosen model problem for stationary cases is the Poisson equation which is representative of steady-state diffusion problems, while the model problem for non-stationary cases is an advection equation which is representative of convection-dominated transport phenomena, i.e., convection-diffusion problems where low or no diffusion is considered. The choice to consider a purely advective model problem is motivated by the necessity to assess the stability properties of the RBF-FD approach for time-dependent simulations where no diffusion is considered. These analyses will allow to develop stable RBF-FD schemes for the simulation of time-dependent flows which will be presented in Chapter 5.

The error norm employed for the required comparisons is:

$$\|u\| = \sqrt{\frac{1}{\mu(\Omega)}\int_\Omega u^2 \, \mathrm{d}\Omega} \approx \sqrt{\frac{\sum_{i=1}^{n_I} s^D(\mathbf{x}_i) u^2(\mathbf{x}_i)}{\sum_{i=1}^{n_I} s^D(\mathbf{x}_i)}} \tag{3.38}$$

where $\mu(\Omega)$ is the area/volume of domain Ω and $D = 2, 3$ is the number of dimensions.

Domains and node distributions. In 2D cases, a circular domain Ω with unit radius and centered at the origin is considered, while a constant nodal spacing is employed. In 3D cases, the considered domain Ω is a sphere with unit radius and centered at the origin, while a constant nodal spacing is employed. The node distributions are obtained through the node generation techniques presented in Chapter 2: the DQT (2D) and DOT (3D) algorithms are employed to generate an initial node distribution which is then refined by the application of $k = 100$ node-repel iterations. A fixed boundary node distribution is employed in 2D cases, while the boundary projection technique is employed in 3D cases.

Poisson equation

The Poisson equation is:

$$\nabla^2 f = q(\mathbf{x}) \tag{3.39}$$

for which the differential operator in Eq. (3.27a) is $\mathcal{D} = \nabla^2$ and the RHS term $q(\mathbf{x})$ is obtained from the Poisson Eq. (3.39) itself for a given analytical solution f. Robin boundary conditions are considered, i.e., the differential operator in Eq. (3.27b) is $\mathcal{B}_H = \alpha f + (1 - \alpha)\partial f/\partial \mathbf{n}$ and the RHS term $\bar{f}$ in Eq. (3.27b) is obtained from the same equation for a given analytical solution f. Dirichlet BCs are obtained for $\alpha = 1$ while Neumann BCs are obtained for $\alpha = 0$.

2D case. The chosen analytical solution is:

$$f(x, y) = \sin(\pi x)\sin(\pi y) \tag{3.40}$$

The influence of the number of the interpolation nodes n on the norm of the absolute error $g - f$ and on the norm of the discretization error for the laplacian is shown in Figure 3.13 for polynomial degrees $2 \leq P \leq 5$, $N \approx 10,000$ nodes and $s \cdot \varepsilon = 0.1$. The discretization error is $\tilde{\nabla}^2 f - \nabla^2 f$ where $\tilde{\nabla}^2$ is the RBF-FD discretized laplacian operator. The minimum number of interpolation nodes using a polynomial augmentation with degree P is $n \geq m = \binom{P+2}{P}$ which is the number of the polynomial basis functions in 2D.

The curves for both errors in Figure 3.13 show that for a given polynomial degree P the increase of the number of interpolation nodes from $n = m$ is initially followed by a strong decrease of the error, while larger n do not bring any significant improvement. For this Poisson problem is therefore convenient to employ $n = 15, 20, 25, 35$ interpolation nodes for $p = 2, 3, 4, 5$, respectively.

An important difference between the absolute error and the discretization error can be observed in Figure 3.13: the discretization error exhibits a monotonic decrease with the polynomial order P whereas the decrease of the absolute error is discontinuous since the curves for $P = 2, 3$ are very similar, as well as for the curves for $P = 4, 5$.

The influence of the shape factor ε is reported in Figure 3.14 for the values $10^{-3} < s \cdot \varepsilon < 1$ using the numbers of interpolation nodes n previously chosen and $N \approx 10,000$ nodes. As expected, the errors decrease for increasingly flat RBFs, i.e., $\varepsilon \to 0$, until the numerical errors due to the increasingly ill-conditioned interpolation matrix become significant. The optimal values for $s \cdot \varepsilon$ lie in the range $[0.01, 0.1]$, while reliable values can be chosen in the range $[0.05, 0.5]$.

The results of the convergence studies, i.e., increasing the total number of nodes N, are shown in Figure 3.15 using the stationary interpolation for which $s \cdot \varepsilon$ is kept constant for each P. Similarly to the previous observations, the convergence curves for the absolute error $g - f$ in Figure 3.15a do not exhibit a monotonic decrease when P is increased: the increase from $P = 2$ to $P = 3$ and from $P = 4$ to $P = 5$ does not improve neither the absolute error nor the order of accuracy p_E which is $p_E \approx 2$ for $P = 2, 3$ and $p_E \approx 4$ for $P = 4, 5$. This observation does not hold for the convergence curves of the discretization error in Figure 3.15b, showing an order accuracy $p_E \approx P - 1$ as expected.

The influence of the refinement iterations k in the node-repel phase of the node generation process is shown in Figure 3.16 for both errors and $N \approx 10,000$ nodes. The effect of the iterations k is very limited in each case with non-negligible effects only for $P = 2$. These graphs show that in this stationary case the accuracy of the RBF-FD discretization is unaffected by the quality of the node distribution, especially when large stencils n with high degree polynomials are employed.

The last influence analysis regards the effects of the parameter α defining the Robin boundary conditions. The results for $\alpha \in [0, 1]$ are reported in Figure 3.17 for both errors and $N \approx 50,000$ nodes. The decrease of α, i.e., moving towards Neumann BCs, results in a significant deterioration of the solution, Figure 3.17a, especially in the case $P = 5, n = 35$ where the accuracy of the solution is reduced by almost three orders of magnitude. As expected, Figure 3.17b shows a limited effect of α on the discretization error since the deterioration of the accuracy occurs at the boundary only.

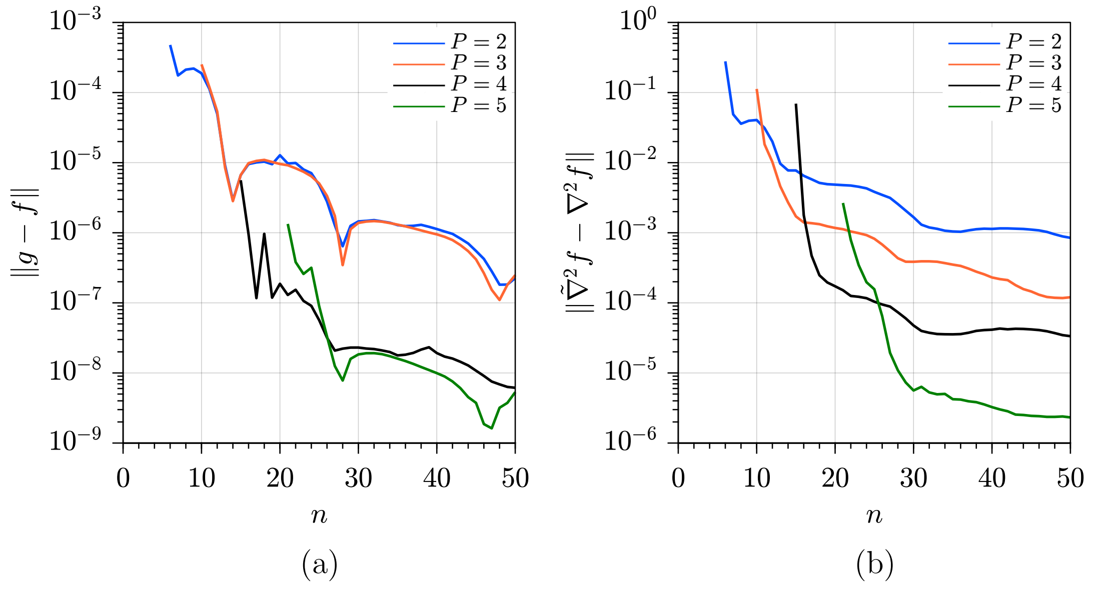


Figure 3.13: Influence of the number of interpolation nodes n on the errors: (a) solution, (b) laplacian. 2D case, $s \cdot \varepsilon = 0.1$, $N \approx 10,000$ nodes.

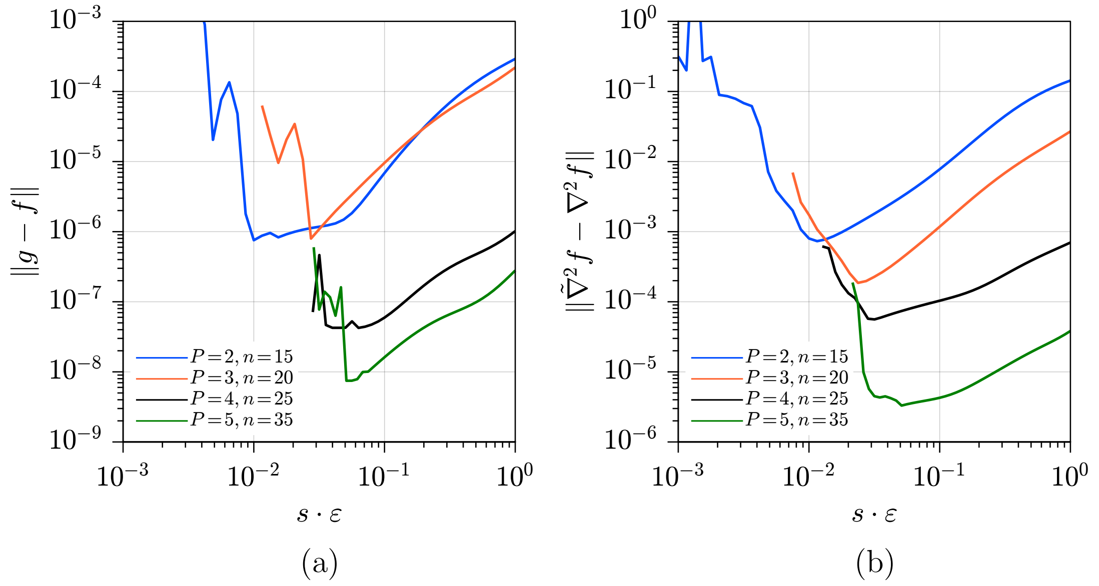


Figure 3.14: Influence of the shape factor ε on the errors: (a) solution, (b) laplacian. 2D case, $N \approx 10,000$ nodes.

3D CASE. The chosen analytical solution is:

$$f(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z) \tag{3.41}$$

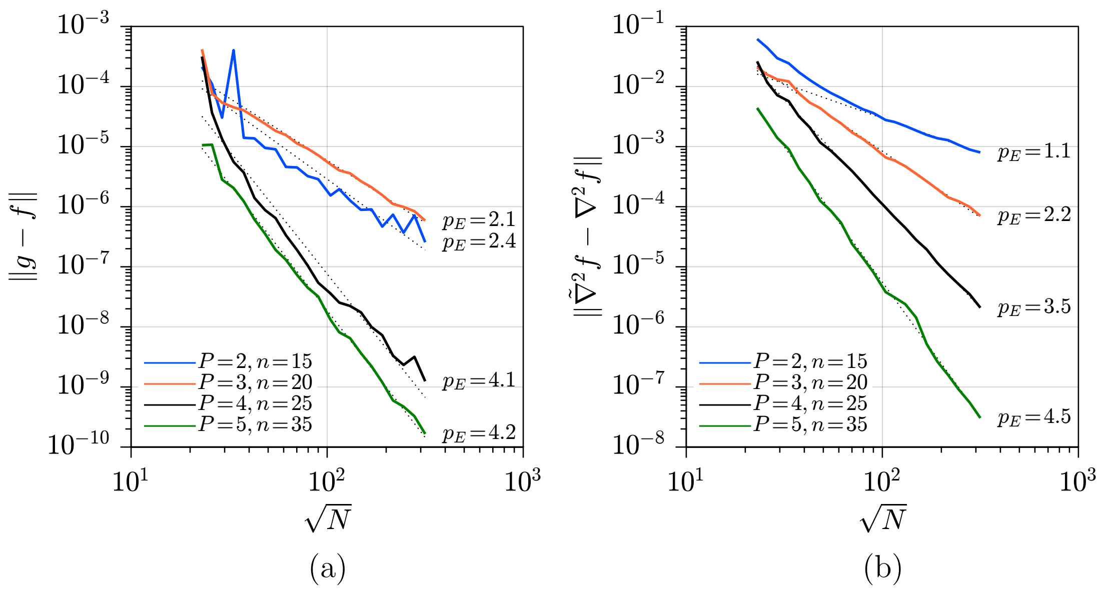


Figure 3.15: Convergence curves for (a) solution and (b) laplacian. 2D case, $s \cdot \varepsilon = 0.05$ for $P = 2$, $s \cdot \varepsilon = 0.075$ for $P = 3$ and $s \cdot \varepsilon = 0.1$ for $P = 4, 5$.

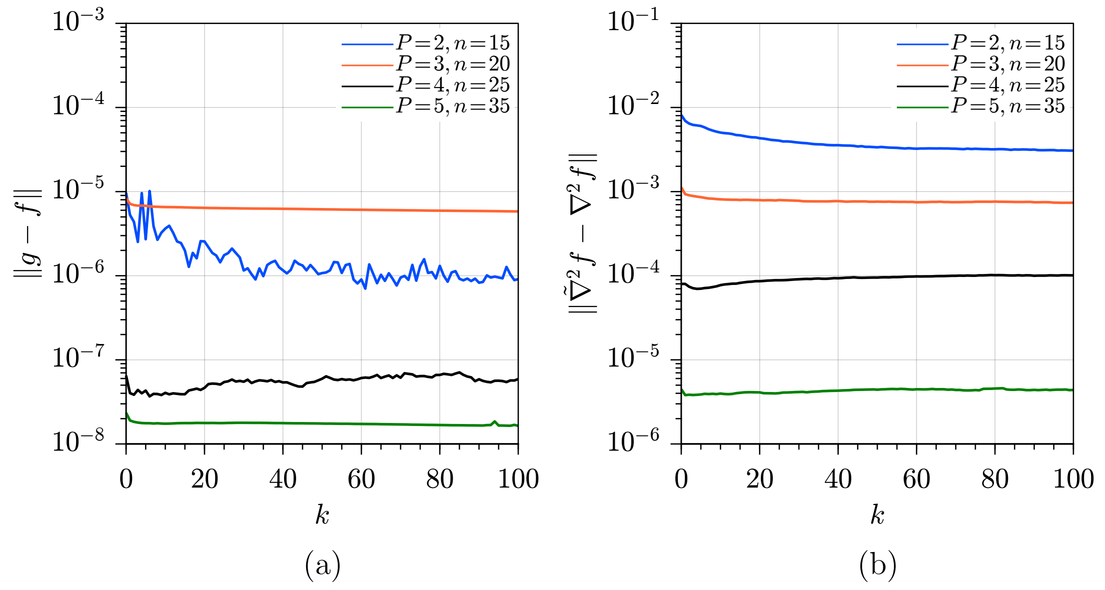


Figure 3.16: Influence of the number of refinement iterations k on the errors: (a) solution, (b) laplacian. 2D case, $N \approx 10,000$ nodes, $s \cdot \varepsilon = 0.05$ for $P = 2$, $s \cdot \varepsilon = 0.075$ for $P = 3$ and $s \cdot \varepsilon = 0.1$ for $P = 4, 5$.

The influence of the number of the interpolation nodes n on the norm of the

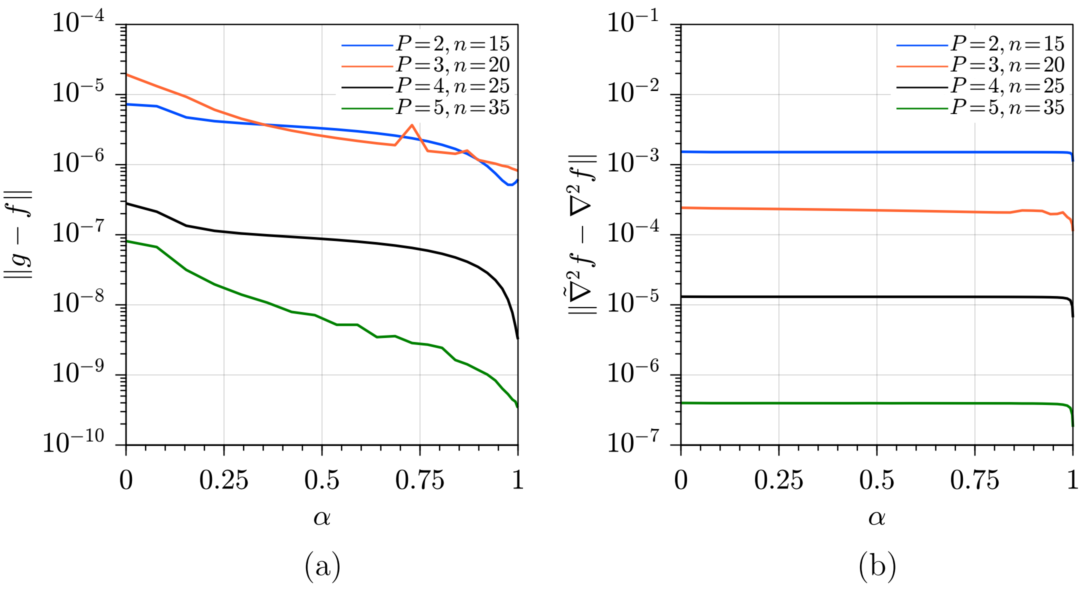


Figure 3.17: Influence of the boundary condition $\mathcal{B}_{\mathcal{H}}(g) = \alpha g + (1-\alpha)\partial g/\partial \mathbf{n}$ on the errors: (a) solution, (b) laplacian. 2D case, $N \approx 50,000$ nodes, $s \cdot \varepsilon = 0.05$ for $P = 2$, $s \cdot \varepsilon = 0.075$ for $P = 3$ and $s \cdot \varepsilon = 0.1$ for $P = 4, 5$.

absolute error $g - f$ and on the norm of the discretization error for the laplacian is shown in Figure 3.18 for polynomial degrees $2 \leq P \leq 5$, $N \approx 50,000$ nodes and $s \cdot \varepsilon = 0.2$. The minimum number of interpolation nodes using a polynomial augmentation with degree P is $n \geq m = \binom{P+3}{P}$ which is the number of the polynomial basis functions in 3D.

Similarly to the 2D case, the curves for the absolute error in Figure 3.18a show that for a given polynomial degree P the increase of the number of interpolation nodes from $n = m$ is initially followed by a strong decrease of the error, while for $P = 5$ this effect is not evident because the cases $m < n < 64$ have not been calculated due to issues with the ILU factorization. The curves for the discretization error in Figure 3.18b also reveal a positive effect of the increase of the stencil size n for a given polynomial degree P, although less evident. In this case it is therefore convenient to employ $n = 30, 35, 55, 80$ interpolation nodes for $p = 2, 3, 4, 5$, respectively.

The influence of the shape factor ε is reported in Figure 3.19 for the values $10^{-3} < s \cdot \varepsilon < 1$ using the numbers of interpolation nodes n previously chosen and $N \approx 50,000$ nodes. The errors decrease for increasingly flat RBFs as predicted by the RBF theory, while the deterioration effect due to the ill-conditioned interpolation matrix for smaller values of $s \cdot \varepsilon$ has not been calculated because it resulted in convergence issues also. The optimal values for $s \cdot \varepsilon$ lie in the range $[0.01, 0.1]$,

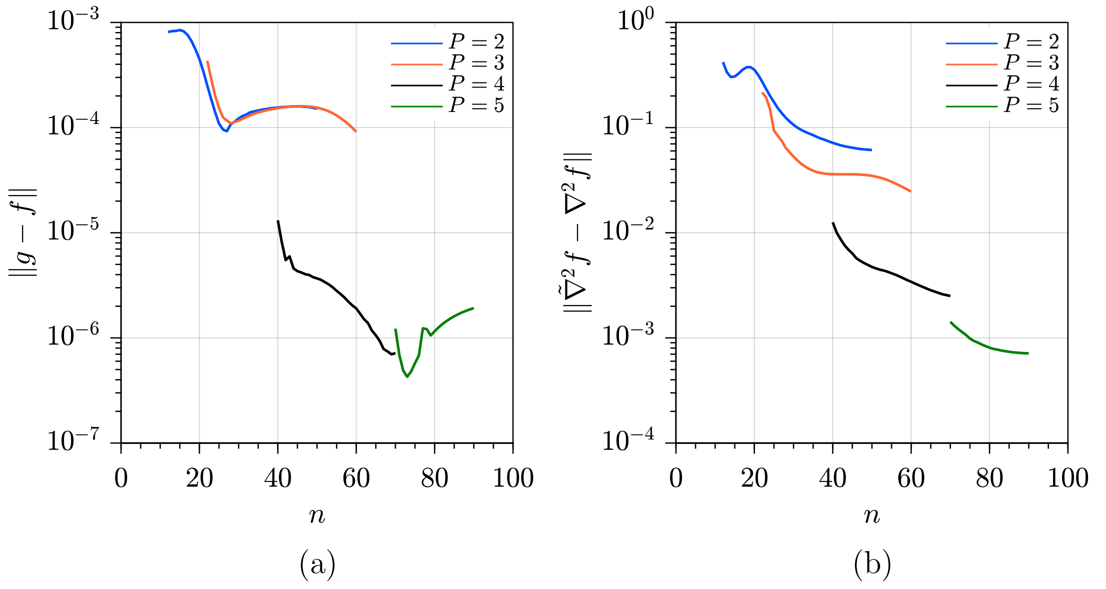


Figure 3.18: Influence of the number of interpolation nodes n on the errors: (a) solution, (b) laplacian. 3D case, $s \cdot \varepsilon = 0.2$, $N \approx 50,000$ nodes.

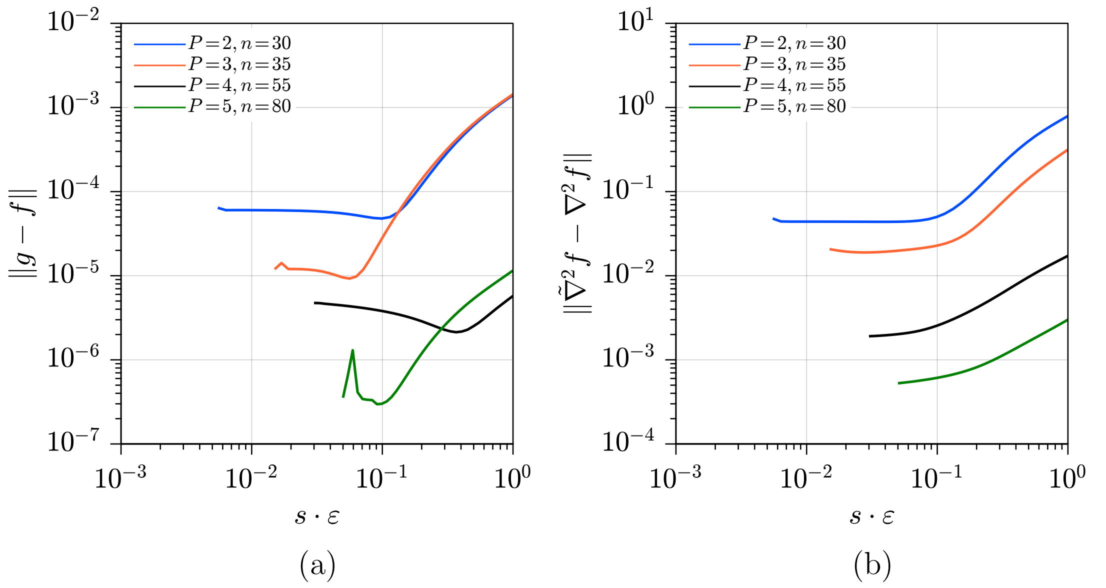
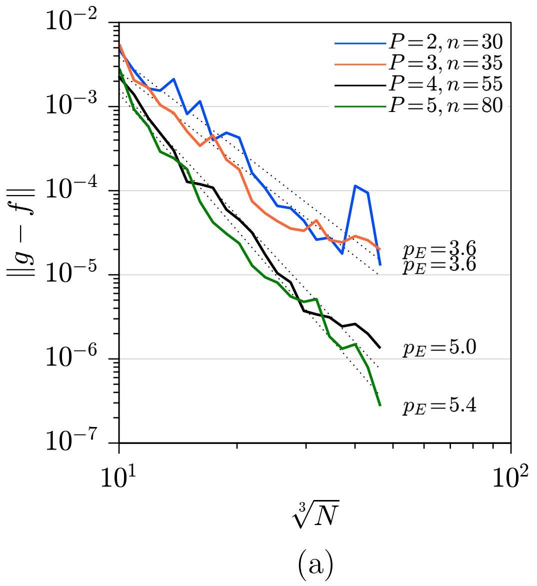


Figure 3.19: Influence of the shape factor ε on the errors: (a) solution, (b) laplacian. 3D case, $N \approx 50,000$ nodes.

while reliable values can be chosen in the range $[0.05, 0.5]$, which is the same range employed in the 2D case.



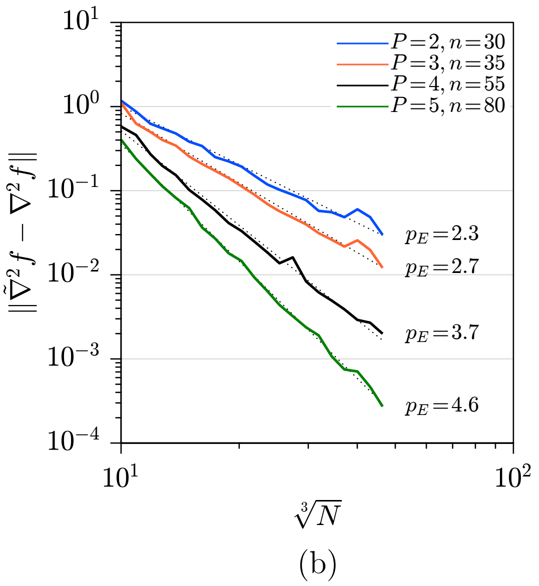


Figure 3.20: Convergence curves for (a) solution and (b) laplacian. 3D case, $s \cdot \varepsilon = 0.1$ for $P = 2, 3$; $s \cdot \varepsilon = 0.2$ for $P = 4, 5$.

The results of the convergence studies are shown in Figure 3.20 using the stationary interpolation for which $s \cdot \varepsilon$ is kept constant for each P. Similarly to the 2D case, the convergence curves for the absolute error $g - f$ in Figure 3.20a do not exhibit a monotonic decrease when P is increased. The increase from $P = 2$ to $P = 3$ and from $P = 4$ to $P = 5$ does not improve neither the absolute error nor the order of accuracy p_E which is $p_E \approx 3.5$ for $P = 2, 3$ and $p_E \approx 5$ for $P = 4, 5$. Again, this observation does not hold for the convergence curves of the discretization error in Figure 3.20b which reveals a monotonic increase of the order of accuracy p_E with P, going from $p_E \approx 2.3$ for $P = 2$ to $p_E \approx 4.6$ for $P = 5$.

The last influence analysis for the 3D Poisson equation regards the effect of the node-repel refinement iterations k on the errors. The results for both errors and $N \approx 10,000$ nodes are shown in Figure 3.21, revealing a slightly difference with respect to the 2D case. In each of the reported cases, the application of the first $k = 10$ iterations results in non-negligible improvements, probably because of the boundary projection technique employed in the 3D refinement phase only. In the case $P = 5$ it has been found that the limitation of the number of boundary nodes $n_B < \bar{n}_B = 10$ in the RBF-FD discretization resulted in a decreased sensitivity to the node distribution. The effects of this boundary limitation can be observed in Figure 3.21 where this strategy is denoted by (B).

Anyway, the effects of the node-repel iterations are limited and the RBF-FD approach proves to be robust also in this 3D stationary case.

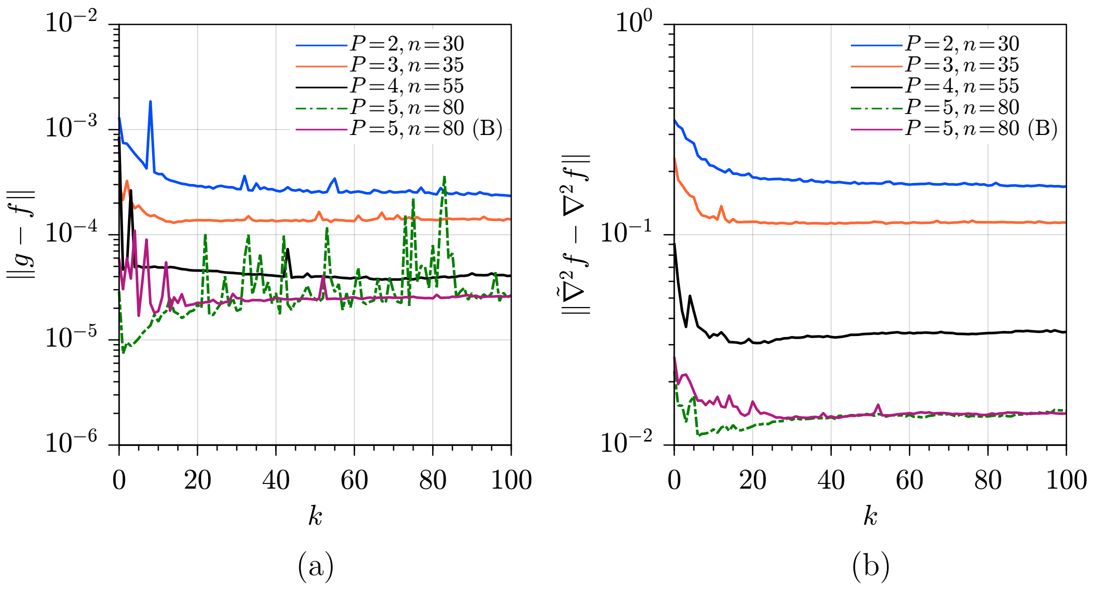


Figure 3.21: Influence of the number of refinement iterations k on the errors: (a) solution, (b) laplacian. 3D case, $N \approx 10,000$ nodes, $s \cdot \varepsilon = 0.1$ for $P = 2, 3$; $s \cdot \varepsilon = 0.2$ for $P = 4, 5$.

Advection equation

The advection Eq. (3.31) is here considered in the stabilized version with hyperviscosity as expressed by Eq. (3.35):

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = \delta \nabla^{2k} f \tag{3.42}$$

where $\delta = \bar{\delta}_k s^{2k}(-1)^{k+1}$ and $\bar{\delta}_k$ has to be found in order to guarantee the stability while minimizing the numerical diffusion. A rotational field around the z axis is considered for the advective velocity which is $\mathbf{u} = (-y, x)^T$ in the 2D case and $\mathbf{u} = (-y, x, 0)^T$ in the 3D case.

The following gaussian wave is chosen for the initial profile:

$$f_0(\mathbf{x}, \mathbf{x_0}) = e^{-50\|\mathbf{x}-\mathbf{x_0}\|_2^2} \tag{3.43}$$

where $\mathbf{x_0} = (0.3, 0)^T$ in the 2D case and $\mathbf{x_0} = (0.3, 0, 0)^T$ in the 3D case. The analytical solution is therefore given by $f(\mathbf{x}, t) = f_0(\mathbf{x}, \mathbf{x}_0(t))$ where $\mathbf{x}_0(t) = 0.3 \cdot (\cos t, \sin t)^T$ in the 2D case and $\mathbf{x}_0(t) = 0.3 \cdot (\cos t, \sin t, 0)^T$ in the 3D case.

The differential operator in Eq. (3.27a) for the stabilized advection equation is $\mathcal{D} = \mathbf{u} \cdot \nabla - \delta \nabla^{2k}$ and the RHS term is $q(\mathbf{x}) = 0$. Dirichlet boundary conditions are employed on the circular/spherical boundary. Since the maximum value of the

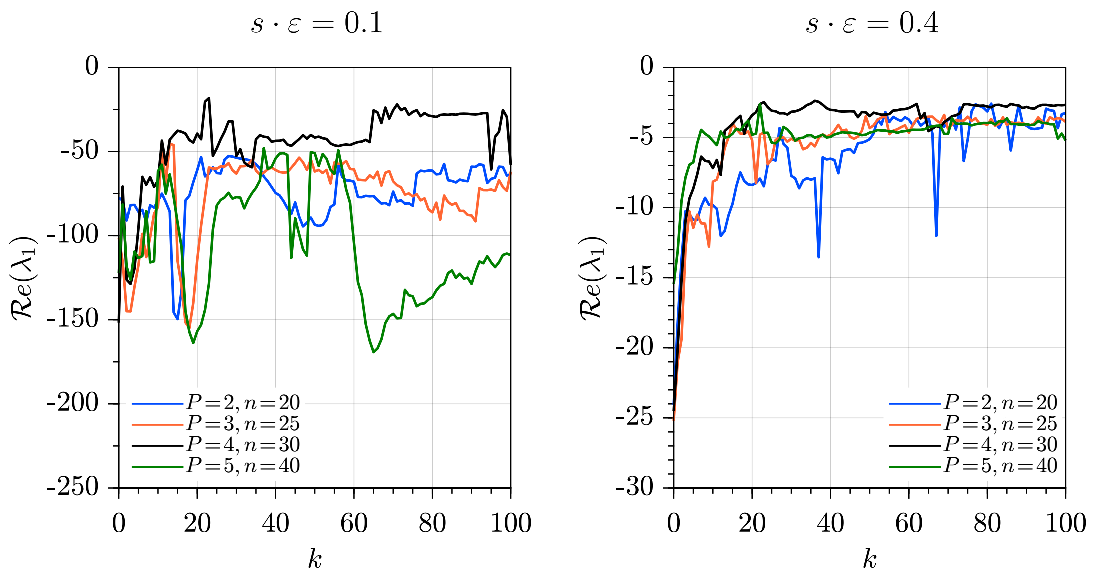


Figure 3.22: Influence of the number of refinement iterations k on the real part of the most unstable eigenvalue of $\mathbf{D}$ with $N \approx 2000$ nodes, 2D case.

employed gaussian profile at the boundary is 2×10^{-11}, it is approximated by a zero-valued Dirichlet BC with negligible errors.

With the previous assumptions, the RBF-FD discretization of Eq. (3.42) leads to the following vector equation:

$$\frac{\partial \mathbf{f}_g}{\partial t} + \mathbf{D}\mathbf{f}_g = \mathbf{0} \tag{3.44}$$

Before moving on to the time discretization, it is useful to understand how the quality of the node distribution affects the stability of the discretization for this non-stationary problem without any stabilization, i.e., $\delta = 0$. This effect can be quantified by studying the influence of the node-repel refinement iterations k on the real part of the most unstable eigenvalue λ_1 of the coefficient matrix $\mathcal{D}$, i.e., the eigenvalue with the smaller real part. The results for the 2D case with $N \approx 2000$ nodes and for two values $s \cdot \varepsilon = 0.1$ and $s \cdot \varepsilon = 0.4$ for the RBF-FD discretization are reported in Figure 3.22.

The curves for $s \cdot \varepsilon = 0.1$, Figure 3.22a, indicate no clear influence of k on the stability, while in the case $s \cdot \varepsilon = 0.4$, Figure 3.22b, there is a significant stability improvement in each case since $\mathcal{R}e(\lambda_1)$ is significantly increased after the first $k = 20 - 40$ iterations. Regardless of the influence of the refinement iterations, the case $s \cdot \varepsilon = 0.1$ is "more unstable" than the case $s \cdot \varepsilon = 0.4$ since the values $\mathcal{R}e(\lambda_1)$ are smaller in the former case. These observations are confirmed by numerical experiments where discretizations with $s \cdot \varepsilon < 0.1$ are found extremely unstable, while the node-repel iterations are found to enhance the stability with larger shape factors, e.g., $s \cdot \varepsilon = 0.4$ which is chosen for the following computations.

The time discretization is obtained by integrating Eq. (3.44) over a time step Δt using a third order approximation for the term $\mathbf{D}\mathbf{f}_g$:

$$\mathbf{f}_g^{(n+1)} - \mathbf{f}_g^{(n)} + \Delta t \mathbf{D}\big(\beta_1 \mathbf{f}_g^{(n+1)} + \beta_2 \mathbf{f}_g^{(n)} + \beta_3 \mathbf{f}_g^{(n-1)}\big) = \mathbf{0} \tag{3.45}$$

where n is the time level and the weights β_i are $\beta_1 = 5/12$, $\beta_2 = 8/12$, $\beta_3 = -1/12$.

The integration is performed over a single revolution with period $T = 2\pi$ using 3000 time steps $\Delta t = 2\pi/3000$ which are sufficiently small to avoid any appreciable time discretization error. Three cases $k = 1, 2, 3$ for the hyperviscosity are considered, corresponding to the stabilization terms $\nabla^2, \nabla^4, \nabla^6$. The specific amount of hyperviscosity $\bar{\delta}_k$ defined in Eq. (3.37) is found in each case by trial and error in order to prevent the appearance of unstable modes over the whole revolution for small number of nodes $N \approx 5000$. This value is kept constant for each specific discretization approach, i.e., for a given P and n, while the scaling with the number of nodes N follows Eq. (3.37) resulting in an artificial diffusion which decreases to zero with order $2k$.

2D CASE. Figure 3.23 shows the results in the case $P = 2$ and $n = 20$ interpolation nodes. The time histories of the norm of the absolute errors $g - f$ over the revolution period T are depicted in Figure 3.23a for $N \approx 100,000$ nodes, where the specific amount of hyperviscosity is $\bar{\delta}_1 = 25$ for $k = 1$, $\bar{\delta}_2 = 15$ for $k = 2$ and $\bar{\delta}_3 = 20$ for $k = 3$. The initial growth of the error is due to the initial gaussian shape, while the employment of the $k = 2$ hyperviscosity leads to a significant reduction of the error because of the reduced artificial diffusion. The use of the $k = 3$ hyperviscosity does not improve the solution since the totality of the error is now due to the RBF-FD discretization with $P = 2$. This behaviour is confirmed by the convergence curves at $t = T$, i.e., after a whole revolution, which are depicted in Figure 3.23b. The $k = 1$ hyperviscosity (laplacian) introduces a lot of numerical diffusion which enters the asymptotic regime only for $N \gtrsim 100,000$ ($\sqrt{N} \approx 300$). The $k = 2$ and $k = 3$ hyperviscosities differ only for small N since their diffusive effect rapidly decrease to zero with order 4 and 6, respectively. The convergence curve for $k = 3$ is therefore representative of the advective RBF-FD discretization error only, having order $p_E \approx 2$ as expected.

Figure 3.24a shows the convergence curves for the $k = 3$ hyperviscosity and different polynomial degrees $2 \leq P \leq 5$. The chosen number of interpolation nodes n is slightly larger than the recommended values presented in the 2D stationary case in Subsection 3.2.3 since this choice resulted in slightly more stable discretizations. The specific amount of hyperviscosity is $\bar{\delta}_3 = 20$ for $P = 2$, $\bar{\delta}_3 = 10$ for $P = 3$, $\bar{\delta}_3 = 5$ for $P = 4$ and $\bar{\delta}_3 = 3.5$ for $P = 5$. Since $2k = 6 > P$ in each case, i.e.,

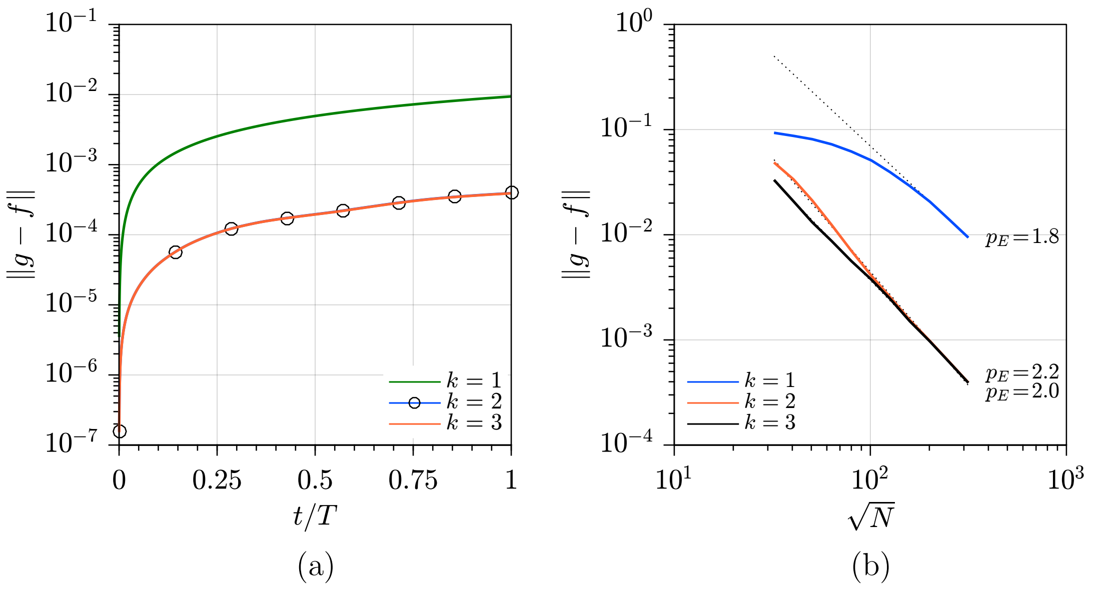


Figure 3.23: Time history of the error norm for $N \approx 100,000$ nodes (a) and convergence curves at $t = T$ (b) for polynomial degree $P = 2$, $n = 20$ interpolation nodes, $s \cdot \varepsilon = 0.4$, 2D case. k is the power of the laplacian for the hyperviscosity $\nabla^{2k} = \Delta^k$.

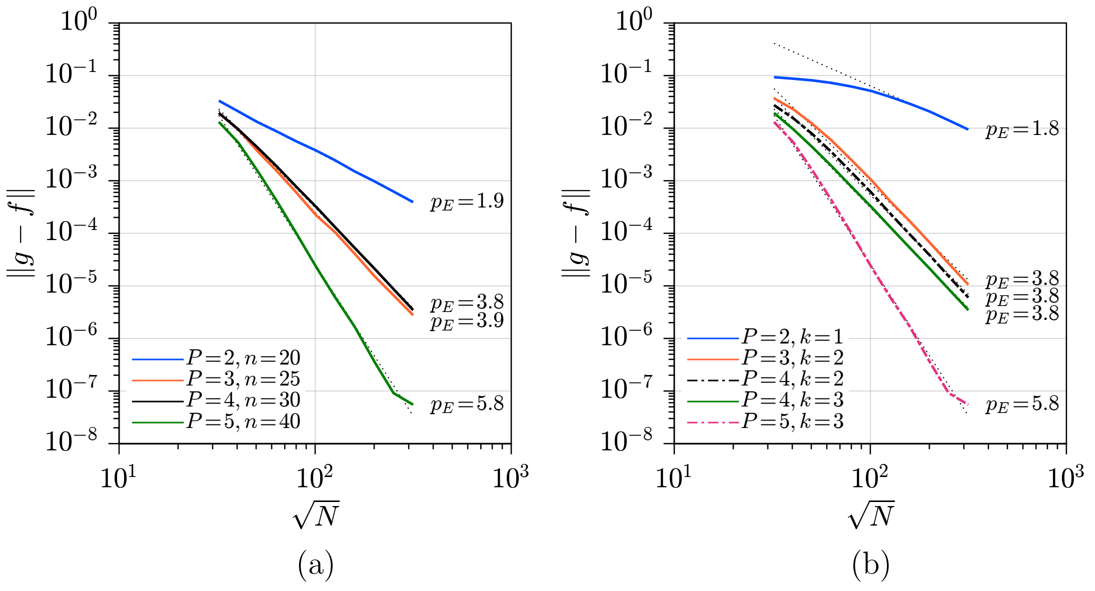


Figure 3.24: Convergence curves at $t = T$ for (a) hyperviscosity ∇^6 ($k = 3$) and (b) different combinations of hyperviscosity and polynomial degree, $2k \geq P$. 2D case.

the order $2k$ of the hyperviscosity is higher than the polynomial degree P, it is

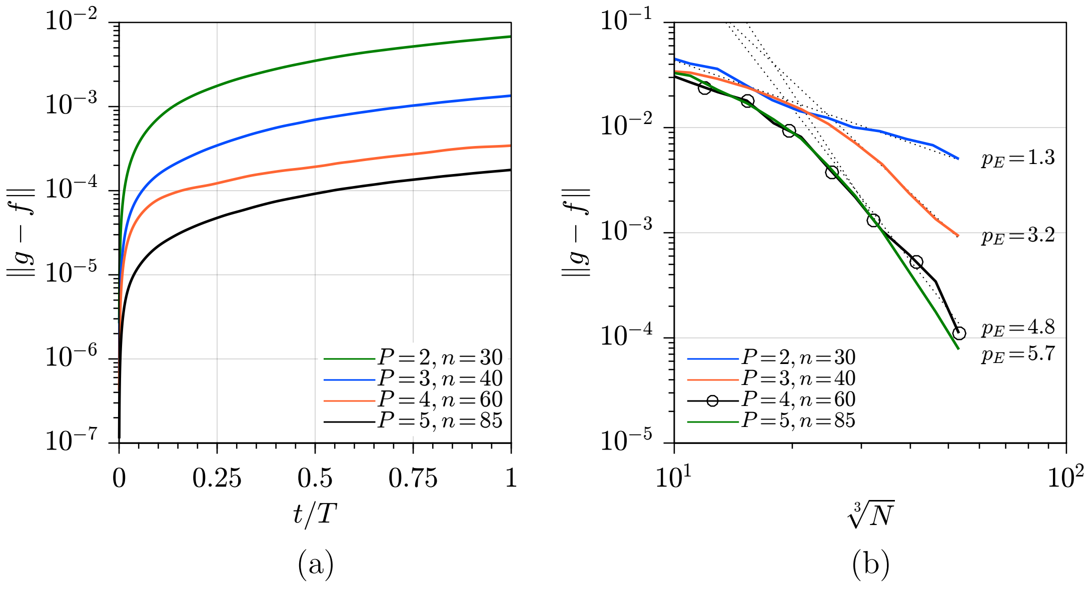


Figure 3.25: Time history of the error norm for $N \approx 100,000$ nodes (a) and convergence curves at $t = T$ (b). 3D case, $s \cdot \varepsilon = 0.4$, $k = 3$ hyperviscosity (∇^6).

expected that the convergence curves are representative of the advective RBF-FD discretization errors only. The obtained orders of accuracy are $p_E \approx 2$ for $P = 2$, as expected, $p_E \approx 4$ for $P = 3, 4$ and $p_E \approx 6$ for $P = 5$.

Figure 3.24b shows the convergence curves for various combinations of polynomial order P and and hyperviscosities k, $2k \geq P$. The cases $2k < P$ make no sense for convergence studies since a larger order of the artificial hyperviscosity $2k$ than the polynomial order P would introduce asymptotic diffusion errors. The curve $P = 2, k = 1$ is discussed before and it is reported for comparison. The increase from $P = 3$ to $P = 4$ with $k = 2$ does not bring any improvement, as reported in Figure 3.24a for $k = 3$, as well as for the increase from $k = 2$ to $k = 3$ for the same degree $P = 4$. A further increase from $P = 4$ to $P = 5$ with $k = 3$ leads to an increase in the order of accuracy from $p_E \approx 4$ to $p_E \approx 6$.

The comparison of the previous convergence curves therefore suggests that a good strategy is to employ hyperviscosities with $2k > P$ for which the stabilizing effect allows the use of high polynomial degrees P, e.g., up to $P = 5$ in these tests, for the solution of non-stationary problems with very low artificial diffusion.

3D CASE. In this case no stability analysis is carried out with respect to the shape factor ε and the same value $s \cdot \varepsilon = 0.4$ employed in the 2D case is considered here. Polynomial degrees $P = 2, 3, 4, 5$ are considered, for which the corresponding number of employed interpolation nodes is $n = 30, 40, 60, 85$, respectively. Similarly to the 2D case, these values n are slightly larger than the recommended values presented in the 3D stationary case in Subsection 3.2.3 since this choice resulted in slightly more stable discretizations. A stabilization through hyperviscosity with $k = 3$, i.e., ∇^6, is employed since negligible numerical diffusion was obtained in the 2D case.

The time histories of the norm of the absolute errors $g - f$ over the revolution period T are depicted in Figure 3.25a for $N \approx 100,000$ nodes. The specific amount of hyperviscosity which has been employed is $\bar{\delta}_3 = 1.25$ for $P = 2$, $\bar{\delta}_3 = 0.5$ for $P = 3$ and $\bar{\delta}_3 = 0.4$ for $P = 4$ and $P = 5$. Since the artificial diffusion due to the $k = 3$ hyperviscosity can be assumed to be negligible, the growth of the error curves in Figure 3.25a is only due to the RBF-FD discretization errors which show a monotonic decrease when P is increased from $P = 2$ to $P = 5$.

The corresponding convergence curves at $t = T$, i.e., after a whole revolution, are depicted in Figure 3.25b and reveal that the asymptotic regime is only partially reached (the maximum number of nodes is $N \approx 150,000$). Nonetheless, the increase of P is followed by a monotonic increase of the order of accuracy p_E, especially going from $P = 2$ ($p_E \approx 1.3$) to $P = 4$ ($p_E \approx 4.8$), while the accuracy in the case $P = 5$ is very similar to the case $P = 4$. Since the asymptotic regime is not fully reached, the case $P = 5$ is expected to outperform the case $P = 4$ when larger number of nodes $N > 150,000$ are considered.

3.3. Conclusions

In this chapter the RBF-FD approach is presented and several 2D/3D test cases are considered in order to highlight the numerical properties of this meshless approach. In particular, a stationary diffusion equation (Poisson equation) and a non-stationary transport equation (advection equation) are considered as model problems in order to assess the accuracy of the RBF-FD discretizations for which the influence of the discretizations parameters is thoroughly investigated.

The findings presented in this chapter represent an important basis on which to rely when facing more complex problems which are characterized by a diffusive component and an advective component, e.g., the incompressible Navier-Stokes equations.

Chapter 4

Multicloud techniques

In the case of linear or linearized PDEs, the linear system arising from the RBF-FD discretization presented in Chapter 3, Eq. (3.30), is characterized by a large, sparse and unsymmetric coefficient matrix. For small problem sizes, e.g., $N < 50000$ in 2D and $N < 10000$ nodes in 3D, the direct solution through a LU factorization [85] can be employed, while large size problems require the use of iterative methods in order to overcome the limitations of direct methods regarding memory and computing time requirements. A typical choice [4] for an iterative approach is the BiCGSTAB method [97] with an incomplete LU factorization (ILU) as preconditioner [85]. When the size of the problem becomes very large, e.g., $N > 10^6$, even this choice can be prohibitive in terms of both time and memory consumption, and therefore more efficient solvers have to be used. In order to achieve such desired convergence acceleration, the basic concepts of multigrid (MG) methods [78] are in this Chapter extended to the linear systems arising from RBF-FD discretization of a 2D Poisson equation in the case of $n = 7$ interpolation nodes. We use the name multicloud (MC) for the developed techniques since the multilevel approach is coupled to the meshless discretization and data structure. The multicloud denomination is also used by Katz and Jameson [55] to define meshless coarse-level operators within a similar multilevel approach.

Although algebraic multigrid (AMG) [94, 95] could be employed with RBF-FD discretizations since it operates at equation level without any additional geometrical information ("black box" matrix solver), it is useful to develop simple MC approaches which are specifically designed and tuned for the RBF-FD discretizations. This is of utmost importance for the use of the RBF-FD approach in large scale problems.

The working principle of the MG approach, formally defined in the paper of Brandt [9], is to optimally reduce the various frequency components of the error on a hierarchy of coarser grids by means of proper interpolation/restriction and smoothing operators. The key idea behind this working principle was somehow previously discovered by some authors that implicitly introduced a two-level correction which is the foundation of the recursive definition of the MG algorithm. Southwell [92] proposed block/group relaxations for mechanical frameworks and stated that such techniques are “almost essential to practical success”. Stiefel [93] proposed an analogous block relaxation for a FD discretization of a Poisson problem and observed that the steepest descent for the quadratic function of residuals is equivalent to impose zero-mean residuals on the block, i.e., the summation of block equations; it is then suggested to use smaller blocks within the starting block, i.e., a MG approach. Fedorenko [27, 28] formalized the first correction scheme with a two-level FD discretization of a 2D Poisson problem and bilinear interpolation. De la Vallee Poussin [20] obtained an additive block correction strategy from the case of a FD discretized heat conduction problem with internal “slots” with infinite thermal conductivity and reported convergence acceleration also for the case of constant thermal conductivity, while Settari and Aziz [91] formalized a general additive correction strategy showing its feasibility to practical problems.

Even after the formalization of the MG approach, many additive correction MG approaches have been proposed and employed to solve practical problems [6–8, 49] because of their easy implementation, since the involved interpolation/restriction operators are given by piecewise constant functions. Another slightly more complex, but still simple choice, for interpolation/restriction operators is given by smoothing the piecewise constant operators using the problem equations themselves [98–101]. These two types of AMG are particularly attractive for meshless applications because of their ease of implementation, independence upon geometric discretization and because they require only a single fine-grid discretization.

The application of MG principles to meshless methods has been previously investigated only with a limited number of works: Leem, Oliveira and Stewart [66] studied the application of AMG with smoothed transfer operators to a Poisson problem discretized with the Reproducing Kernel Particle Method (RKPM). Seibold [90] studied the same problem with AMG but using a Generalized Finite Difference Method (GFDM). More recently, Katz and Jameson [55] developed a multigrid technique with meshless transfer operators at coarse levels, and called it multicloud (MC).

The work presented in this Chapter regards the extension of the multigrid solution approach to RBF-FD meshless methods in the case of straight additive correction and smoothed transfer operators strategies. Such strategies are employed to develop two simple MC techniques: additive correction multicloud (ACMC) and smoothed restriction multicloud (SRMC) which differ for the type of restriction strategy only. In ACMC the restriction operator is piecewise constant over the restriction support while in SRMC it is smoothed from constant by applying one Jacobi iteration using the problem equations themselves, resulting in an increased support size. The implementation of such procedures is very simple and straightforward within the RBF-FD meshless data structure. Both techniques can be used as standalone solvers and as preconditioners for iterative solvers such as BiCGSTAB, allowing the convergence acceleration in the case of RBF-FD discretizations of a 2D Poisson equation.

A particular attention is given to the study of the factors influencing the convergence properties of the proposed MC approaches. Tests are carried out with two different complex-shaped domains and different boundary conditions for a 2D Poisson equation in order to assess the characteristics of the developed techniques from a practical point of view, in the simple case of a multiquadric RBF-FD discretization with $n = 7$ interpolation nodes and linear polynomial augmentation $P = 1$. High benefits in terms of savings in computing time and amount of work to convergence have been achieved when compared to a standard BiCGSTAB solver with ILU preconditioning in the case of large number of nodes, e.g., $N \approx 10^5 - 10^6$. The coupling of the developed MC algorithms with a classical iterative solver such as BiCGSTAB allows additional gains in the performances and in the reduction of the sensitivity to MC parameters.

4.1. Domains, node distributions, BCs and analytical solutions

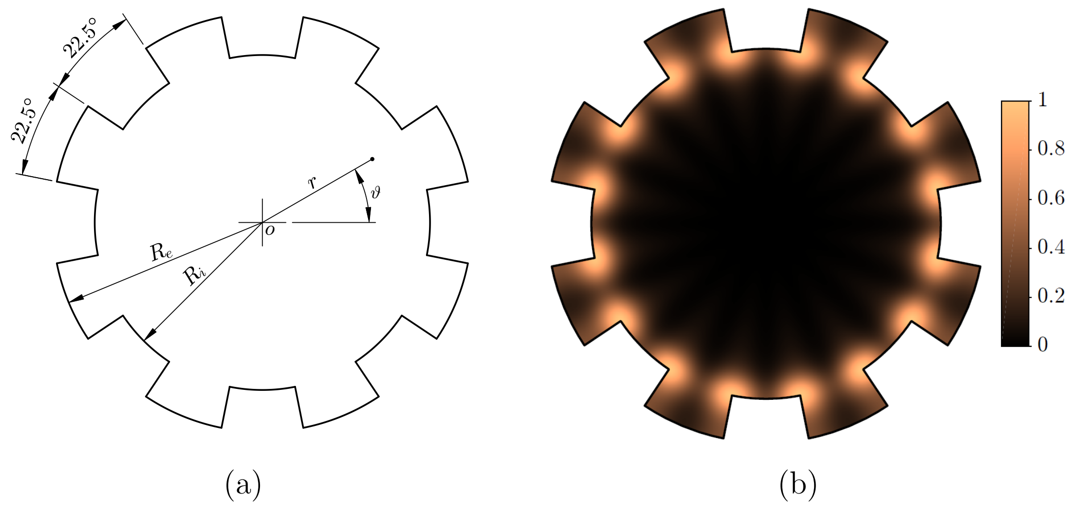
Two complex shaped domains are employed in order to demonstrate the application of the proposed MC techniques to practical problems which are characterized by complex geometries and non-trivial node arrangements. The introduction of the domains is required here because preliminary analysis will be carried out in the following Section 4.3 (stencil positivity).

4.1.1. Case 1

Geometry The geometry of the domain Ω is a circle with $N_g = 8$ periodical angular grooves, as depicted in Figure 4.1a together with the coordinate system. The ratio R_i/R_e between the internal and the external radii of the grooves has been chosen to be 0.8.

Node distributions The following spacing function $s(\mathbf{x})$ is employed for the generation of node distributions over this domain:

$$\frac{s_M}{s(\mathbf{x})} = 1 + \bar{r}^2 \frac{2 - \cos(2N_g\vartheta)}{3} \Big[1 + k_E \exp\big(-k_S(\bar{r} - \bar{r}_S)^2\big)\Big]^2 \tag{4.1}$$



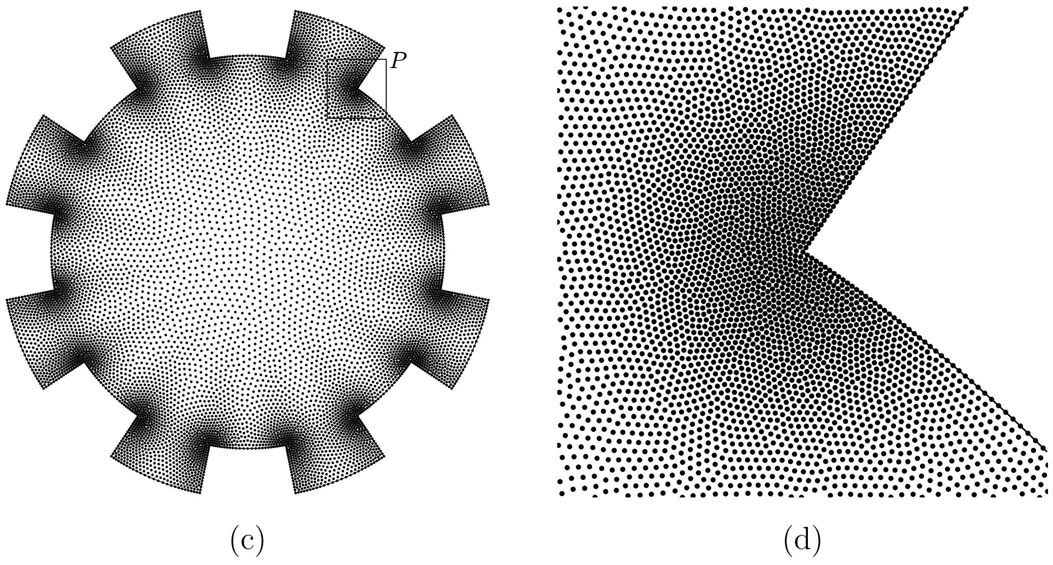


Figure 4.1: Case 1: geometry (a), analytical solution (b), example of node distribution with $N \approx 10^4$ nodes (c) and a snapshot of the particular part P of the domain (d).

where s_M is the maximum spacing function over the domain, $\bar{r} = r/R_i$ is the dimensionless radius, $k_E = 0.75$, $k_S = 50$ and $\bar{r}_S = 1 - (1 + k_E)/(2k_E k_S)$. The ratio between the maximum and the minimum nodal spacing is $s_M/s_m \approx 4$. An example of node distribution with $N \approx 10^4$ nodes is displayed in Figure 4.1c. A snapshot of the particular part P of the node arrangement in the neighbourhood of a corner is displayed in Figure 4.1d.

Boundary conditions Dirichlet BCs on the whole boundary Γ are considered.

Analytical solution The following analytical solution is used for BCs and for the definition of the error norms:

$$f(\mathbf{x}) = \bar{r}^3 \Big[2 - \cos(2N_g \vartheta) \Big] \Big[1 + k_E \exp\big(- k_S (\bar{r} - \bar{r}_S)^2 \big) \Big]^3 \tag{4.2}$$

A graphical representation of this analytical solution is reported in Figure 4.1b, after scaling it to the interval $[0, 1]$.

4.1.2. Case 2

Geometry The domain Ω is implicitly defined as the set $\{\mathbf{x} \in \mathcal{R}^2 : G(\mathbf{x}) \geq 0\}$ where G is defined as:

$$G(\mathbf{x}) = \frac{2}{3} \Big[\cos(6\pi x) \cos(6\pi y) - 2(x^4 + y^4)^2 + \frac{1}{2} \Big] \tag{4.3}$$

The geometry of such complex shaped domain with multiple holes is depicted in Figure 4.2a together with the coordinate system. The boundary Γ is simply given by the level set $G(\mathbf{x}) = 0$.

Node distributions The spacing function $s(\mathbf{x})$ employed for the generation of node distributions over this domain is:

$$s(\mathbf{x}) = s_M \Big[1 + k_G G^2(\mathbf{x}) \Big] \tag{4.4}$$

where s_M is the maximum spacing function over the domain and $k_G = 3$. The ratio between the maximum and the minimum nodal spacing is $s_M/s_m = 1 + k_G = 4$. An example of node distribution with $N \approx 15000$ nodes is displayed in Figure 4.2c. A snapshot of the particular part P of the domain is displayed in Figure 4.2d, showing the node arrangement in the neighbourhood of a double boundary.

Boundary conditions Two types of BCs have been considered for this domain:

- Case 2Dir: Dirichlet BCs on the whole boundary;
- Case 2Mix: Mixed BCs: Dirichlet BCs on the "external" boundary, Neumann BCs on the boundary of the 40 "internal" holes.

Analytical solution The analytical solution that has been used for BCs and error norms is simply defined as $f(\mathbf{x}) = G(\mathbf{x})$. A graphical representation of this analytical solution is shown in Figure 4.2b, after scaling it to the interval $[0, 1]$.

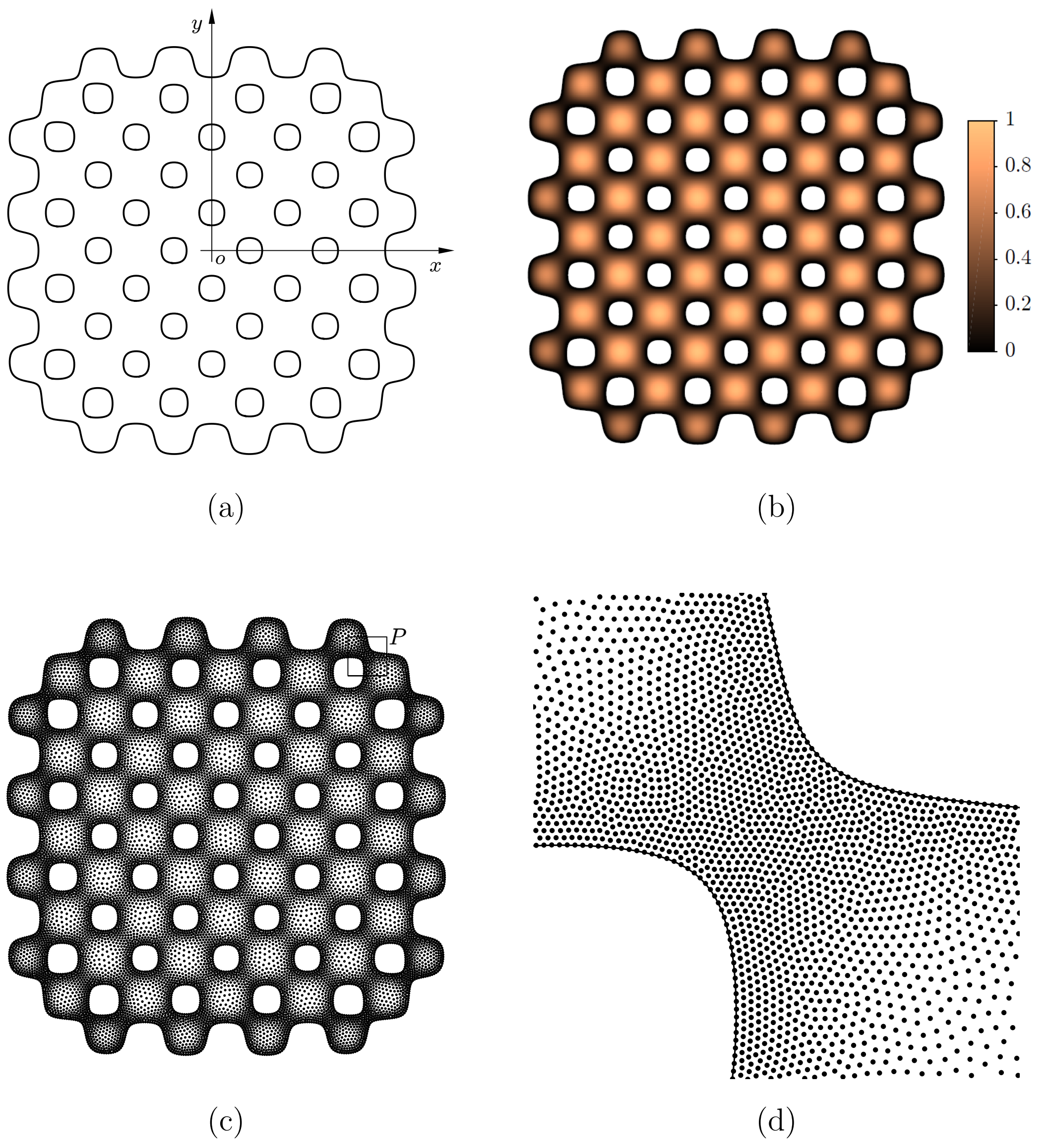


Figure 4.2: Case 2: geometry (a), analytical solution (b), example of node distribution with $N \approx 15000$ nodes (c) and a snapshot of the particular part P of the domain (d).

4.2. Model problem and RBF-FD discretization features

The chosen model problem is the following 2D Poisson equation for which a negative laplacian is considered in order to deal with positive-definite matrices in the following:

$$-\nabla^2 f = q(\mathbf{x}) \tag{4.5}$$

which is subjected to mixed boundary conditions: Dirichlet BCs on the Dirichlet boundary Γ_D and Neumann BCs on the Neumann boundary Γ_N.

The required node distributions are obtained through the node generation techniques presented in Chapter 2. The DQT algorithm is employed to generate an initial node distribution which is then refined by the application of $k = 100$ node-repel iterations with a fixed boundary node distribution.

For the sake of simplicity in the development of the MC algorithms, a RBF-FD discretization with $n = 7$ interpolation nodes and linear augmentation $P = 1$, Eq. (3.7), is considered only.

A non-stationary MQ-RBF interpolation is employed, where the shape factor is rescaled as:

$$\varepsilon = s_M \bar{\varepsilon} / s(\bar{\mathbf{x}}) \tag{4.6}$$

where $\bar{\varepsilon}$ is the rescaled shape factor, s_M is the maximum prescribed spacing function on the domain Ω and $\bar{\mathbf{x}}$ is the mean point for each local support as defined in Subsection 3.1.2. In the non-stationary interpolation the shape factor ε is kept constant under continuing node refinement, which is the case of Eq. (4.6) since the ratio $s_M/s(\bar{\mathbf{x}})$ is constant for each point regardless of the spacing s, i.e., regardless of the total number of nodes N.

4.3. Multicloud techniques

In this Section the basic concepts of MG methods are introduced and extended to the employed RBF-FD discretization in the case of the considered Poisson problem. The analyses of MG methods are elaborated in [9, 78, 94].

4.3.1. Multigrid

Mathematical formulation

Let us consider the following sparse linear system arising from the employed RBF-FD discretization of the Poisson Eq. (4.5), which is the case of Eq. (3.30) with the differential operator $\mathcal{D} = -\nabla^2$:

$$\mathbf{D}\mathbf{f} = \mathbf{b} \tag{4.7}$$

The solution of system (4.7) using classical iterative methods, e.g., Jacobi, Gauss-Seidel, successive over-relaxation (SOR), leads to an efficient reduction of the high-frequency error components in space, while the low-frequency error components are slowly reduced [28]. This property can be seen by considering the Jacobi iteration:

$$\mathbf{M}\mathbf{f}_{k+1} = \mathbf{N}\mathbf{f}_k + \mathbf{b} \tag{4.8}$$

where $\mathbf{M} = diag(\mathbf{D})$ and $\mathbf{N} = \mathbf{M} - \mathbf{D}$. Let us consider the local, continuous-space approximation of the previous Jacobi iteration where the vector quantities are replaced by the corresponding continuous variables:

$$f_{k+1} = f_k + \alpha(q + \nabla^2 f_k) \tag{4.9}$$

where $\alpha = \mathcal{O}(h^2) > 0$ with grid spacing h. Then, considering a Fourier component of the error $e_k = f - f_k = E_k e^{j\boldsymbol{\omega}\cdot\mathbf{x}}$ with $\boldsymbol{\omega}$ the vector of spatial frequencies, Eq. (4.9) gives the following convergence factor $\mu(\boldsymbol{\omega})$:

$$\mu(\boldsymbol{\omega}) = \left|\frac{E_{k+1}}{E_k}\right| = 1 - \mathcal{O}(\|\boldsymbol{\omega}\|_2^2 h^2) \tag{4.10}$$

where $\mathcal{O}(1) = \omega_m \leq \|\boldsymbol{\omega}\|_2 \leq \omega_M = \mathcal{O}(h^{-1})$ since the realizable frequencies for f on a given grid lie in the range $[\omega_m, \omega_M]$ on local basis. Therefore, rapidly fluctuating errors, i.e., $\|\boldsymbol{\omega}\|_2 \approx \omega_M$, have favourable convergence factors $\mu \ll 1$, while low-frequency components, i.e., $\|\boldsymbol{\omega}\|_2 \approx \omega_m$, are very slowly decaying because $\mu \approx 1$. The Jacobi iteration (4.8) acts as an error smoother and thus cannot be used as a practical single-grid solver for discretizations with a large number of unknowns.

Nevertheless, this smoothing property can be somehow exploited to improve this iterative method: the application of k smoothing iterations (typically SOR) leaves only a smooth error $\mathbf{e}_k = \mathbf{f} - \mathbf{f}_k$ with both geometric and algebraic sense [94] since the Poisson equation is isotropic and the grid is also assumed to be isotropic. Then, the linear system (4.7) in terms of error becomes

$$\mathbf{D}\mathbf{e}_k = \mathbf{b} - \mathbf{D}\mathbf{u}_k =: \mathbf{r}_k \tag{4.11}$$

which states that the error $\mathbf{e}_k$ can alternatively be computed from the residual $\mathbf{r}_k$. Since $\mathbf{e}_k$ is smooth because of the k Jacobi/SOR iterations, it can therefore be well approximated using a smaller number of meaningful unknowns $\tilde{\mathbf{e}}_k$ through a suitable interpolation operator $\mathbf{I}_H^h$:

$$\mathbf{e}_k = \mathbf{I}_H^h \tilde{\mathbf{e}}_k \tag{4.12}$$

The application of a restriction operator $\mathbf{I}_h^H$ to Eq. (4.11) with the smooth approximation of Eq. (4.12) gives the following coarse grid correction equation:

$$\underbrace{\left(\mathbf{I}_h^H \mathbf{D} \mathbf{I}_H^h\right)}_{\tilde{\mathbf{D}}} \tilde{\mathbf{e}}_k = \underbrace{\mathbf{I}_h^H \mathbf{r}_k}_{\tilde{\mathbf{r}}_k} \tag{4.13}$$

where $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{r}}_k$ are the coarse correction matrix and the coarse residual, respectively.

Eq. (4.13), solved for $\tilde{\mathbf{e}}_k$, allows the final coarse grid correction:

$$\mathbf{f}'_k = \mathbf{f}_k + \mathbf{I}_H^h \tilde{\mathbf{e}}_k \tag{4.14}$$

where $\mathbf{f}'_k$ is the sought approximation for $\mathbf{f}$ obtained by improving $\mathbf{f}_k$ through the addition of the coarse grid correction $\mathbf{I}_H^h \tilde{\mathbf{e}}_k$.

This correction is then followed by further k smoothing iterations needed to smooth the high-frequency error components introduced by the interpolation operator $\mathbf{I}_H^h$ in Eq. (4.14). If the coarse linear system of Eq. (4.13) is still too large to be solved directly or by simple iterative methods, the previous two-level approach can then be recursively applied to the solution of the coarse grid correction system itself. This iterative approach yields the classical V-cycle MG algorithm [9].

The pseudocode of the V-cycle MG algorithm is given by Algorithm 4, where:

- k is the number of smoothing iterations;
- r_I is the iteration multiplier between successive MG levels. r_I thus defines the amount of smoothing work at coarse levels;
- t is the overcorrection factor;
- Smooth($\mathbf{f}, \mathbf{D}, \mathbf{b}, k, \omega$) performs k smoothing iterations (e.g., SOR with relaxation factor ω) on vector $\mathbf{f}$;
- N_{MG} is the threshold size for vector $\mathbf{f}_0$ under which a direct solution is calculated.

Algorithm 4 Multigrid

```
Input: vector $\mathbf{f}_0$, matrix $\mathbf{D}$, RHS vector $\mathbf{b}$, parameters $k, \omega, t, r_I$
Output: solution $\mathbf{f}''_k$
 1: function Multigrid($\mathbf{f}_0, \mathbf{D}, \mathbf{b}, k, \omega, t, r_I$)
 2:     if $size(\mathbf{f}_0) > N_{MG}$ then
 3:         $\mathbf{f}_k \leftarrow$ Smooth($\mathbf{f}_0, \mathbf{D}, \mathbf{b}, k, \omega$)
 4:         $\tilde{\mathbf{r}}_k \leftarrow I_h^H(\mathbf{b} - \mathbf{D}\mathbf{f}_k)$
 5:         $\tilde{\mathbf{e}}_k \leftarrow$ Multigrid($\mathbf{0}, \tilde{\mathbf{D}}, \tilde{\mathbf{r}}_k, \lfloor r_I k \rceil, \omega, t, r_I$)
 6:         $\mathbf{f}'_k \leftarrow \mathbf{f}_k + t I_H^h \tilde{\mathbf{e}}_k$
 7:         $\mathbf{f}''_k \leftarrow$ Smooth($\mathbf{f}'_k, \mathbf{D}, \mathbf{b}, k, \omega$)
 8:     else
 9:         $\mathbf{f}''_k \leftarrow \mathbf{D}^{-1}\mathbf{b}$
        return $\mathbf{f}''_k$
10: end function
```

Transfer operators

The performances of MG methods depends upon the choice for the transfer operators between MG levels, i.e., interpolation $\mathbf{I}_H^h$ and restriction $\mathbf{I}_h^H$, and their interaction with the smoothing operator. A recommended choice [94] is:

$$\mathbf{I}_h^H = (\mathbf{I}_H^h)^T \tag{4.15}$$

which states that the restriction is obtained from interpolation in order to fulfill variational principles [94]. With this choice the sign definiteness and symmetry (if any) of the coarse matrix $\tilde{\mathbf{D}} = (\mathbf{I}_H^h)^T \mathbf{D} \mathbf{I}_H^h$ are maintained between the MG levels, as well as the squared error energy norm $\mathbf{e}_k^T \mathbf{M} \mathbf{e}_k = \tilde{\mathbf{e}}_k^T \tilde{\mathbf{M}} \tilde{\mathbf{e}}_k$. This condition is not always respected in the present work in favour of a simpler numerical implementation.

For second order PDEs such as the Poisson Eq. (4.5), the following condition has to be satisfied [9] in order to obtain the real MG convergence, i.e., convergence rate independent from the size of the problem:

$$r_R + r_I > 2 \tag{4.16}$$

where r_R and r_I are the order of restriction and interpolation operators, respectively. For this reason piecewise linear interpolation ($r_I = 2$), with its transpose piecewise linear restriction ($r_R = 2$), is very common in geometric MG [78]. Eq. (4.16) can also be satisfied if only one operator is piecewise linear while the other is piecewise constant, as suggested by Gjesdal [41] with a piecewise linear restriction which is easier to implement.

We briefly present two AMG approaches which are particularly attractive for RBF-FD discretizations because of their simple implementation, independence upon geometric discretization, low memory requirements and their need of the fine grid discretization only, i.e., no coarse-level discretizations are needed. Therefore boundary conditions are needed only at the finest level and do not need any explicit treatment at the coarse levels.

Aggregation-type MG The simplest AMG approach, also known as aggregation -type MG or additive correction MG, is given by piecewise constant transfer operators and it can be implemented very easily. Nonetheless, a straight additive correction implementation does not produce a true MG convergence for diffusion problems because of an incomplete reduction of the smooth error components [7, 94], as reported in Appendix D. This lack of MG convergence is also confirmed by Eq. (4.16) which is not satisfied since $r_R = r_I = 1$ for piecewise constant transfer operators.

A partial solution to this problem is overcorrection [7], which amplifies the coarse grid correction (4.14) with an overcorrection factor $t > 1$ as follows:

$$\mathbf{f}'_p = \mathbf{f}_p + t\mathbf{I}_H^h \tilde{\mathbf{e}}_p \tag{4.17}$$

where $\mathbf{I}_H^h$ is piecewise constant. For a Poisson equation the optimal value of t depends upon the size of the aggregates and therefore it depends upon the specific geometric discretization.

MG WITH SMOOTHED TRANSFER OPERATORS Instead of overcorrection, it is possible to overcome the problem of the incomplete reduction of the smooth error components encountered with the aggregation-type MG by smoothing transfer operators [98–101]. An under-relaxed homogeneous Jacobi iteration, i.e., with a relaxation smoothing factor $\omega_S < 1$ [28], is applied to the piecewise constant interpolation using the matrix coefficients $\mathbf{D}$ of the problem itself [95] at each MG level, while the restriction operator is given by Eq. (4.15). This choice allows higher convergence rates with a moderate increase in computational effort and memory requirements since the interpolation (and the restriction) radius will increase.

4.3.2. Multicloud

C/F-splitting

A coarsening strategy is required to define a set of coarse-level variables required by interpolation, Eq. (4.12), resulting in a $Coarse/Fine$-splitting (C/F-splitting), using AMG terminology. In the RBF-FD context, variables and equations are associated to nodes, therefore a C/F-splitting is simply given by a set of coarse nodes obtained from a set of fine nodes by using a nodal coarsening strategy. We assume that the coarse-level node set is a subset of the fine-level node set.

In the presented work we employ the nodal coarsening strategy proposed by Katz and Jameson [55] which is similar to the one proposed by Chan et al [11]: starting from a fine-level set where each node has a null flag F (free node), each free node is visited sequentially, its flag is set to 2 and the free nodes from its 6 nearest neighbors are set to have $F = 1$. Then the coarse-level set is given by the nodes with $F = 2$. This choice leads to coarse sets whose number of nodes is approximately 4 times less than the number of fine-level nodes in 2D cases. Such isotropic coarsening strategy is suitable for a RBF-FD discretization of a Poisson equation with isotropic node distribution.

Additive correction multicloud technique (ACMC)

The first MC technique which has been developed is additive correction multicloud (ACMC), which is based on the aggregation-type MG with overcorrection,

●	fine-level nodes	$\longrightarrow$	interpolation/restriction
○	coarse-level nodes	$--$	equation links
✦	fine-level support nodes for restriction to central coarse-level node	$\dashrightarrow$	modified interpolation support

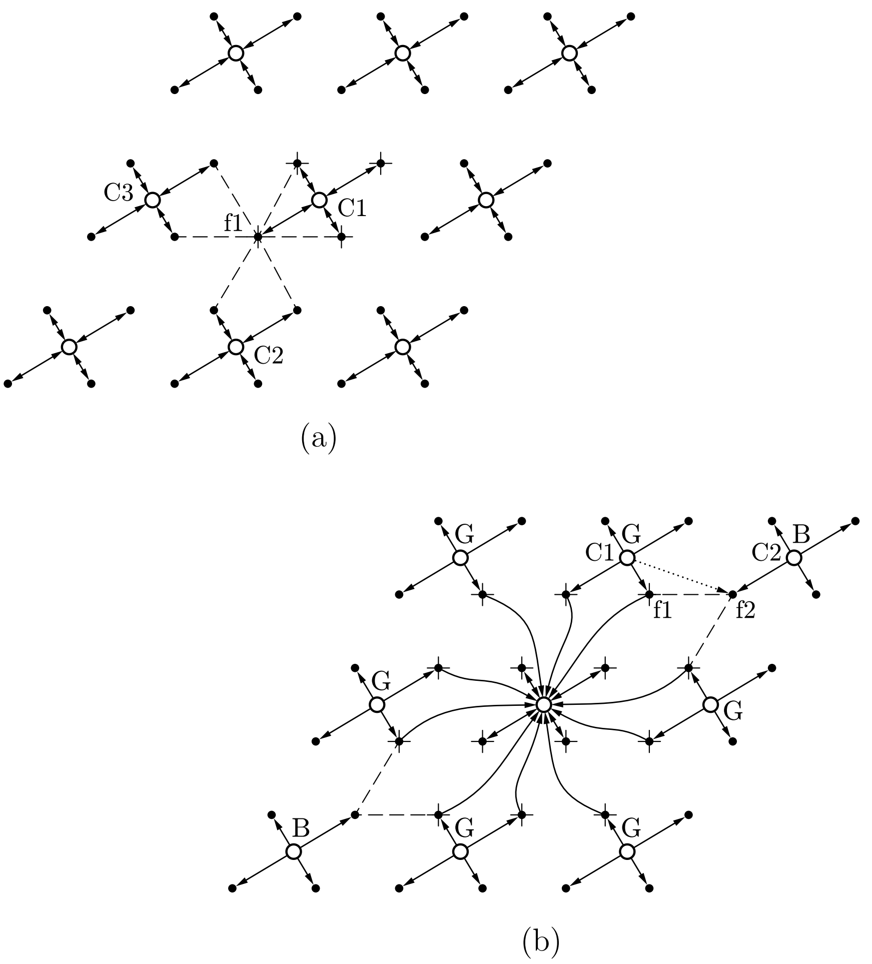


Figure 4.3: Interpolation and restriction support nodes for ACMC (a) and SRMC (b).

described in Subsection 4.3.1, where both transfer operators are piecewise constant over their support. Given a C/F-splitting, the interpolation support is defined by choosing which coarse-level variables/nodes are employed to interpolate a fine-level variable, Eq. (4.12). Similarly, the restriction support is defined by choosing which fine-level equations are employed to obtain a coarse-level correction equation by restriction, Eq. (4.13).

One nearest neighboring coarse-level node is used as the support for the constant interpolation. The restriction is given by Eq. (4.15), i.e., the constant restriction support is given by the fine-level nodes employing the same coarse-level node as interpolation support. This strategy is depicted in Figure 4.3a where the arrows indicate interpolation (coarse to fine nodes) or restriction (fine to coarse nodes) with uniform and isotropic coarse/fine node distributions. For the sake of graphic clarity, the coarse-level nodes have been moved in order not to coincide with fine/coarse-level nodes. The coincidence of some fine/coarse-level nodes is actually assumed in the nodal coarsening strategy described in Subsection 4.3.2 since the coarse-level node set is a subset of the fine-level node set.

By considering a stencil with $n = 7$ neighboring nodes for the RBF-FD approach (this particular choice is explained in Subsection 4.3.3), the corresponding MC strategy maintains a constant stencil size, on average, at each level. This property can be deduced from Figure 4.3a by considering the central coarse-level node C1 and its related equation: its stencil is given by the coarse-level nodes employed to interpolate all the fine-level nodes (marked with symbol ◆ in Figure 4.3a) which are involved in restriction to C1. For example, the equation for fine-level node f1, which is one of the nodes used for the restriction to C1, depends upon its 6 fine-level neighbors whose interpolation support employs only coarse-level nodes C1, C2 and C3. Finally, it is possible to observe that the coarse correction equation for C1 depends upon 6 coarse-level neighbors, and therefore the stencil size $n = 7$ is maintained at all coarse levels on the average.

Smoothed restriction multicloud technique (SRMC)

The second MC technique proposed in this work is smoothed restriction multicloud (SRMC), which is obtained from ACMC by smoothing the piecewise constant restriction operator only, using one under-relaxed homogeneous Jacobi iteration as described in Subsection 4.3.1. The interpolation is still maintained piecewise constant and therefore Eq. (4.15) does not hold anymore.

As in the previous technique, we used one nearest neighboring coarse-level node as support for the constant interpolation, while the restriction operator is obtained in two steps: an initial restriction operator is obtained from Eq. (4.14) which is then smoothed using one under-relaxed homogeneous Jacobi iteration with under-relaxed factor ω_S, resulting in an increased radius for the restriction support.

This strategy is depicted in Figure 4.3b where the solid line arrows indicate interpolation (coarse to fine nodes) or restriction (fine to coarse nodes) with uniform and isotropic coarse/fine node distributions. Again, for the sake of graphic clarity, the coarse-level nodes have been moved in order not to have coincident fine/coarse-level nodes.

By considering the central coarse-level node of Figure 4.3b and a RBF-FD stencil with $n = 7$ neighboring nodes, we observe that the radius of the restriction support is increased from 4 fine-level nodes of ACMC approach to 14 fine-level nodes (marked with symbol ✦ in Figure 4.3b) because of the smoothing operation on the piecewise constant operator. This "extended" restriction support is given by the fine-level nodes whose stencils include any of the "original" four fine-level nodes defining the piecewise constant restriction support.

We also observe that this MC strategy does not maintain a constant stencil size, on average, across the levels. This property can be deduced from Figure 4.3b by considering again the central coarse-level node and its related equation: its stencil is given by the coarse-level nodes employed to interpolate all the fine-level nodes involved in the restriction to the central coarse-level node. It is then possible to observe that the coarse correction equation for the central coarse-level node depends upon 9 coarse-level neighbors, and therefore the stencil size $n = 7$ is not maintained across levels, on average, but will grow at coarse levels. To avoid a growing stencil size that would imply larger memory and computational requirements, we propose a modified restriction strategy which ensures a constant stencil size: considering Figure 4.3b, a coarse-level node is marked G ("good") if it is involved in the interpolation of any of the 14 fine-level nodes defining the central coarse-level node restriction support, otherwise it is marked B ("bad"). Then, the restriction contribution of B coarse-level nodes is replaced by G coarse-level nodes only. For example, when the fine-level equation for node f1 is considered by restriction, the contribution of interpolation of the fine-level node f2, which is included in f1 stencil, is now given by the coarse-level node C1 and not by C2 anymore, because C1 is the G coarse-level node employed for the interpolation of f1. Numerical tests confirmed that this strategy ensures a constant stencil size $n = 7$ at all coarse levels, on average.

4.3.3. Multicloud convergence

As previously outlined, the RBF-FD discretization employed in this work leads to an unsymmetric coefficient matrix. Therefore the classic positive definiteness requirement for symmetric matrices cannot be used to guarantee the convergence of the proposed MC techniques. An analogous requirement for the convergence of MG methods in the unsymmetric case is that the matrix be an M-matrix [43]. We briefly present the sufficient conditions leading to a M-matrix structure (see Seibold [90] and Hackbusch [43] for further details) in the context of the present meshless approach.

M-matrices

Definition 1 (Z-matrices and L-matrices)**.** *A square matrix $\boldsymbol{D} = (D_{ij}) \in \mathcal{R}^{N_I \times N_I}$ is called Z-matrix if $D_{ij} \leq 0 \ \ \forall i \neq j$. A Z-matrix is called L-matrix if $D_{ii} > 0 \ \ \forall i$.*

The notation $\mathbf{D} \geq 0$ indicates that its matrix entries satisfy $D_{ij} \geq 0 \ \ \forall i, j$.

Definition 2 (M-matrices)**.** *A Z-matrix is called M-matrix if it is nonsingular and $\boldsymbol{D}^{-1} \geq 0$.*

Definition 3 (Essentially diagonally dominance)**.** *A square matrix $\boldsymbol{D}$ is called essentially diagonally dominant if it is weakly diagonally dominant:*

$$|D_{ii}| \geq \sum_{j \neq i} |D_{ij}| \ \ \forall i$$

and every node is connected (directly or indirectly) through stencil entries to a node k which satisfies the strict diagonal dominance relation $|D_{kk}| > \sum_{j \neq k} |D_{kj}|$.

Definition 4 (Stencil)**.** *Given the RBF approximation of Eq.* (3.17) *at the first node $\mathbf{x}_1$:*

$$\mathcal{D}\big(g(\mathbf{x})\big)_{\mathbf{x}=\mathbf{x}_1} = f_1 d_1(\mathbf{x}_1) + \cdots + f_{n_I} d_{n_I}(\mathbf{x}_1) + \bar{f}_1 d_{n_I+1}(\mathbf{x}_1) + \cdots + \bar{f}_{n_B} d_n(\mathbf{x}_1) \quad (4.18)$$

the corresponding stencil is defined by the internal nodes $\mathbf{x}_1, \ldots, \mathbf{x}_{n_I}$ and by the coefficients $d_1(\mathbf{x}_1), \ldots, d_{n_I}(\mathbf{x}_1)$.

The first node $\mathbf{x}_1$ of a stencil is the "central" node where the derivatives are sought.

Definition 5 (Positive stencil)**.** *A stencil is called positive if its coefficients $d_1, \ldots, d_{n_I}$ satisfy $d_1 > 0$ and $d_2, \ldots, d_{n_I} \leq 0$, where $n_I \leq n$ is the number of internal nodes.*

Benefits on the use of positive stencils in the context of generalized-FD methods can be found in Demkowicz et al. [21].

Theorem 1. *Matrix $\boldsymbol{D}$ of Eq.* (4.7) *is essentially diagonally dominant if at least one Dirichlet node is employed and if positive stencils are employed for every internal node.*

Proof. Consider the interpolation system of Eq. (3.13) with constant data, i.e., $f_i = 1$, $\bar{f}_i = 1$ if $\hat{\mathbf{x}}_i$ is a Dirichlet boundary node and $\bar{f}_i = 0$ if $\hat{\mathbf{x}}_i$ is a Neumann boundary node. Since such interpolation is exact for constant data because a linear augmentation $P = 1$ is employed, we get $a_i = 0$, $(b_1, b_2, b_3) = (1, 0, 0)$, i.e., $g(\mathbf{x}) = 1$. Eq. (4.18) with $\mathcal{D} = -\nabla^2$ then becomes $\sum_{i \in D}^{n} d_i = 0$ where D is the set of internal and Dirichlet boundary nodes. Since the stencil is positive, the previous relation can be written as $d_1 = \sum_{i \in D,\, i>1}^{n} |d_i|$. If the stencil has no Dirichlet nodes, the last relation implies a weakly diagonally dominant row in terms of matrix coefficients, otherwise a strictly diagonally dominant row is obtained since $d_1 = \sum_{i \in D,\, i>1}^{n} |d_i| > \sum_{i=2}^{n_I} |d_i|$. Finally, in the present RBF-FD approach every node is always directly or indirectly connected to a Dirichlet node through stencil entries because of the employed node distributions and strategy for the choice of stencil support nodes (nearest neighbours). □

Theorem 2. *An essentially diagonally dominant L-matrix is an M-matrix.*

Proof. See Hackbusch [43], pp. 154-155. □

The coefficient matrix $\mathbf{D}$ of Eq. (4.7) resulting from the present RBF-FD discretization is an M-matrix if the conditions of Theorem 1 hold.

Positive stencils

In the previous subsection the RBF-FD coefficient matrix $\mathbf{D}$ is proved to be an M-matrix under the hypothesis of positive stencils for every internal node. The validity of such hypothesis depends upon the number of interpolation nodes n for the MQ-RBF interpolation, their geometrical arrangement and upon the MQ shape factor ε. The investigation of these factors on stencil positivity is here investigated.

First of all, we investigate the effect of the number of support nodes n and MQ shape factor ε on stencil positivity for a local node arrangement given by a central node surrounded by $n-2$ nodes which are distributed over a circle with angular intervals $\Delta\alpha = 2\pi/(n-1)$, while the position $\mathbf{y}$ of the remaining node is free. The positions of such a free node allowing a positive stencil are depicted as hatched areas in the diagrams of Figure 4.4 for $n = 7, 8, 9$ and $\varepsilon = 0.1, 1, 10$, while the areas delimited by red-dashed curves represent "safe" areas where the stencil coefficients $d_1(\mathbf{y}), \ldots, d_n(\mathbf{y})$ satisfy the condition $G(\mathbf{y}) > G_M/2$, where $G(\mathbf{y}) = \min\{d_1(\mathbf{y}), -d_2(\mathbf{y}), \ldots, -d_n(\mathbf{y})\}$ and G_M is the maximum value for G over the whole plane. The larger these areas, the more likely it is to get a positive stencil from general node arrangements with n support nodes.

From Figure 4.4 it is possible to deduce that increasing n from 7 to 9 nodes will decrease the positive stencil area, also due to obvious geometric reasons, while the effect of shape factor ε is non-monotonic: the positive stencil area decreases and then increases considering an increasing shape factor from $\varepsilon = 0.1$ to $\varepsilon = 10$. This is unwanted because "small" ε will produce numerical instabilities when solving the interpolation system, while "big" ε will produce bad interpolants as outlined in

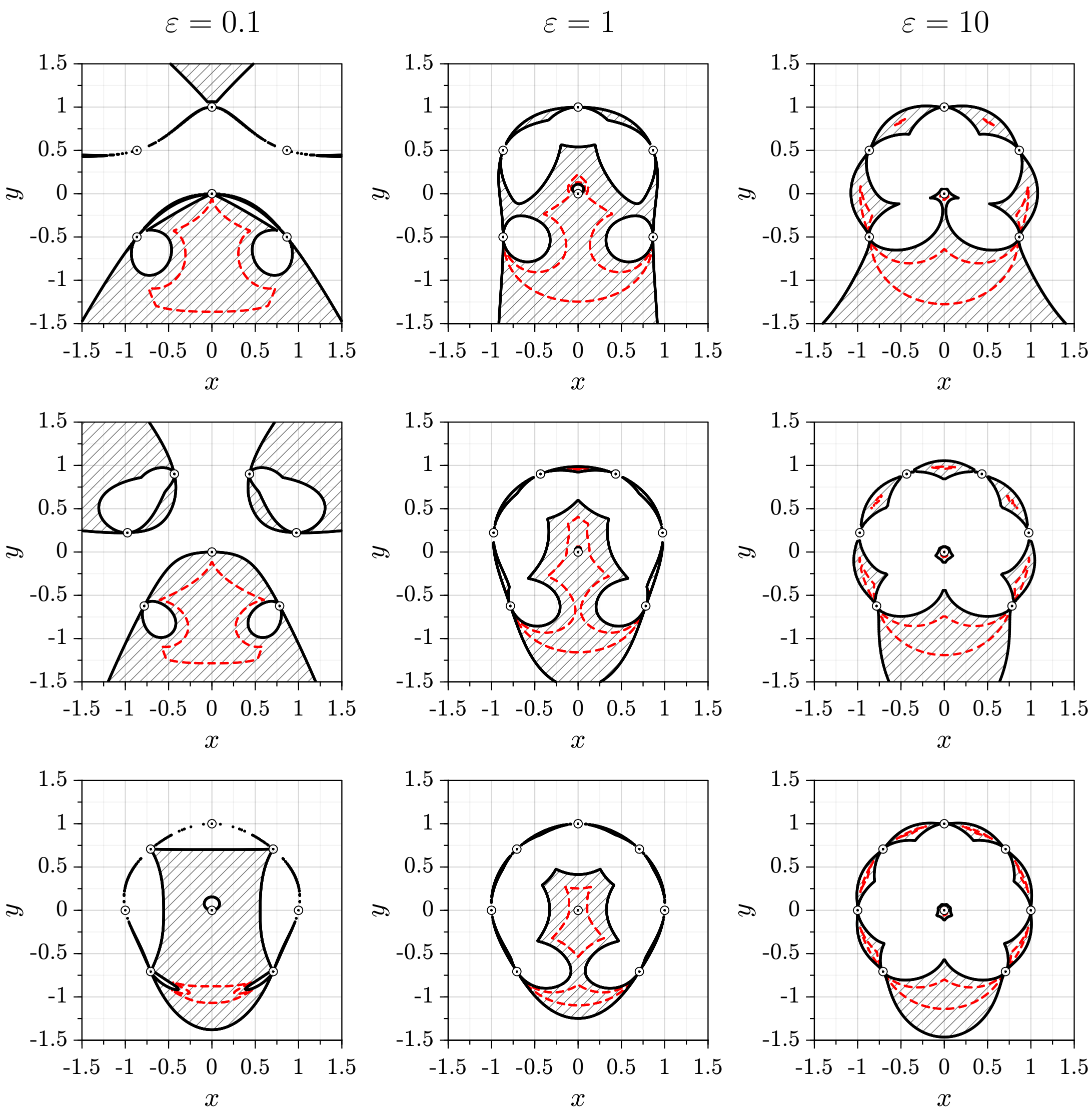
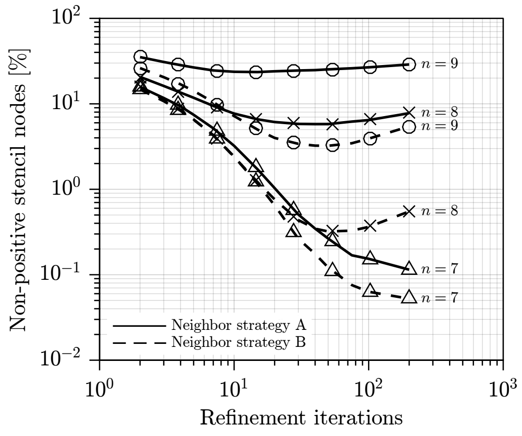


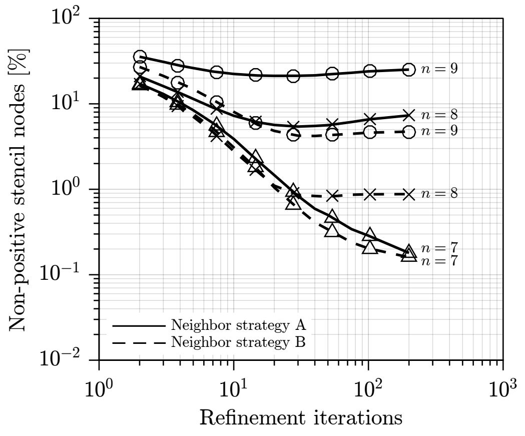
Figure 4.4: Stencil positivity plots for $n = 7$ (top row), $n = 8$ (middle row) and $n = 9$ (bottom row): grey hatched areas represent positive stencil areas, while red dashed curves delimit “safe” areas for stencil positivity (see Subsection 4.3.3).

Chapter 3. Nonetheless, ε has to be chosen on the basis of discretization properties for a specific problem, and therefore it is not an adjustable parameter [62].

Lastly, we investigate the effect of the node distribution on stencils positivity for node distributions with $N \approx 10^5$ nodes for Case 1 and Case 2Mix. The fraction of nodes with non-positive stencils is reported in Figure 4.5 as a function of the number of refinement iterations in the node generation phase for $n = 7, 8, 9$ and for two strategies for the choice of RBF neighboring support nodes:



(a)



(b)

Figure 4.5: Effect of refinement iterations on stencils positivity for Case 1 (a) and Case 2Mix (b), $N \approx 10^5$ nodes and $\bar{\varepsilon} = 1$.

- Neighbor strategy A: n nearest neighboring nodes are always chosen;
- Neighbor strategy B: all the nodes within a circle of radius proportional to $s(\mathbf{x})\sqrt{n}$ are chosen.

The difference between the two previous strategies is that the number of support nodes is exactly n with Neighbor strategy A, while with Neighbor strategy B the number of support nodes is n only on average and a small deviation from n (1 or 2 nodes) can occur depending upon the local node arrangement.

The curves reported in Figure 4.5 reveal that the refinement iterations have a positive effect in the reduction of the fraction of non-positive stencils for $n = 7$ only, while for $n = 8, 9$ this effect is limited and non-monotonic. Neighbor strategy B is also more effective than Neighbor strategy A in the reduction of the fraction of non-positive stencils, especially with $n = 8, 9$. Nonetheless this strategy is found to produce unacceptable discretization errors and therefore only Neighbor strategy A (n nearest neighbors) is employed.

To conclude, $n = 7$ is found to be the most appropriate choice, for which $50 - 100$ refinement iterations are a reasonable balance between computational effort and an adequate reduction of non-positive stencils. For this choice the fraction of non-positive stencils drops significantly below 1% for both cases 1 and 2, and therefore the presence of these few non-positive stencils destroys the M-matrix property. However, this is not a necessary condition [90] for the convergence of the proposed MC techniques.

Table 4.1: MC parameters.

Parameter	Symbol	ACMC	SRMC	Range
- MC pre/post smoothing SOR iterations	k	1**	1**	$\mathcal{N}^+$
- SOR relaxation factor	ω	1.2*	1.2*	$[1, 1.4]$
- ACMC overcorrection factor	t	variable	–	$[1, 3]$
- SRMC under-relaxation smoothing factor	ω_S	–	variable	$[0, 1]$
- SOR iteration multiplier at coarse levels	r_I	1*	1*	$[1, 4]$

* unless otherwise specified.
** at finest level.

4.3.4. Preconditioning

Similarly to the traditional MG methods, both the proposed MC techniques can be used as standalone solvers and as preconditioners for traditional iterative solvers. As suggested by Stüben [95], the coupling of MG methods as preconditioners with reliable and robust iterative methods is more effective than trying to fine-tune a standalone MG solver, especially in cases where the exact convergence proofs cannot be stated. This is our situation because of the presence of non-positive stencils.

When the proposed MC techniques are used as preconditioners, then the BiCGSTAB method [97] is used as an iterative solver. Such iterative solver is one of the most common choices for the solution of nonsymmetric linear systems. The resulting approaches are denoted by BiCGSTAB/ACMC and BiCGSTAB/SRMC.

4.3.5. Multicloud parameters

The working parameters that completely define the proposed MC approaches are briefly summarized in Table 4.1. The coarsest level is assumed to be reached when the number of equations at this level is less than $N_{MG} = 4000$, and the corresponding linear system is solved by direct LU solution.

4.3.6. Work count and residual norm

The comparison of the amount of computational work, i.e., the number of floating point operations in the different algorithms is measured in terms of work units

(WU), which is defined as the amount of work needed for one residual evaluation at the finest level. For the MC techniques, the amount of work is automatically evaluated by considering the pre/post smoothing SOR iterations, residual evaluation, restriction and interpolation phases. For BiCGSTAB solver, each full iteration is composed by two semi-iterations, each of which has a cost of 1 WU+10N_I operations [97], where N_I is the number of internal nodes (number of fine-level equations).

The convergence histories are plotted in terms of normalized RMS residual, which is defined as $\|\mathbf{r}\|_2/\|\mathbf{b}\|_2$, where $\mathbf{b}$ is the RHS of Eq. (4.7) and $\mathbf{r}$ is the corresponding residual vector. A null vector is employed as a starting solution vector.

4.3.7. Error norm

The comparison between the computed solution g and the corresponding analytical solution f is done by computing the normalized RMS norm of the error:

$$\begin{aligned}\text{Normalized RMS error} &= \sqrt{\frac{1}{A(\Omega)}\int_\Omega \left(\frac{g-f}{f_{max}-f_{min}}\right)^2 \mathrm{d}\Omega} \approx \\ &\approx \frac{1}{|f_{max}-f_{min}|}\sqrt{\frac{\sum_{i=1}^{n_I} s^2(\mathbf{x}_i)(g(\mathbf{x}_i)-f(\mathbf{x}_i))^2}{\sum_{i=1}^{n_I} s^2(\mathbf{x}_i)}}\end{aligned} \tag{4.19}$$

where $A(\Omega)$ is the area of Ω.

4.4. Results

4.4.1. Preliminary analyses

First of all, we investigated the effect of the rescaled shape factor $\bar{\varepsilon} \in [0.1, 10]$ on the normalized RMS error as reported in Figures 4.6a and 4.6c for both domains and for $N \approx 10^5$ and $N \approx 10^6$ nodes. As expected, the error reduces as $\bar{\varepsilon}$ decreases, while below $\bar{\varepsilon} = 1$ the error reaches an asymptotic behaviour and therefore $\bar{\varepsilon} = 1$ has been chosen for the following results. The choice $\bar{\varepsilon} < 1$ would also imply numerical stability issues for large N in the local RBF interpolation.

Figures 4.6b and 4.6d show the convergence curves in the case $\bar{\varepsilon} = 1$ for both domains, revealing an order of accuracy[1] $p \approx 1.8$ for Case 1 and $p \approx 1.7$ for Case 2Dir. Convergence curves for Case 2Mix confirm the sensitivity of the RBF-FD approach to the imposition of Neumann BCs.

[1]For 2D cases the order of accuracy p is defined by a normalized RMS error proportional to $N^{-p/2}$

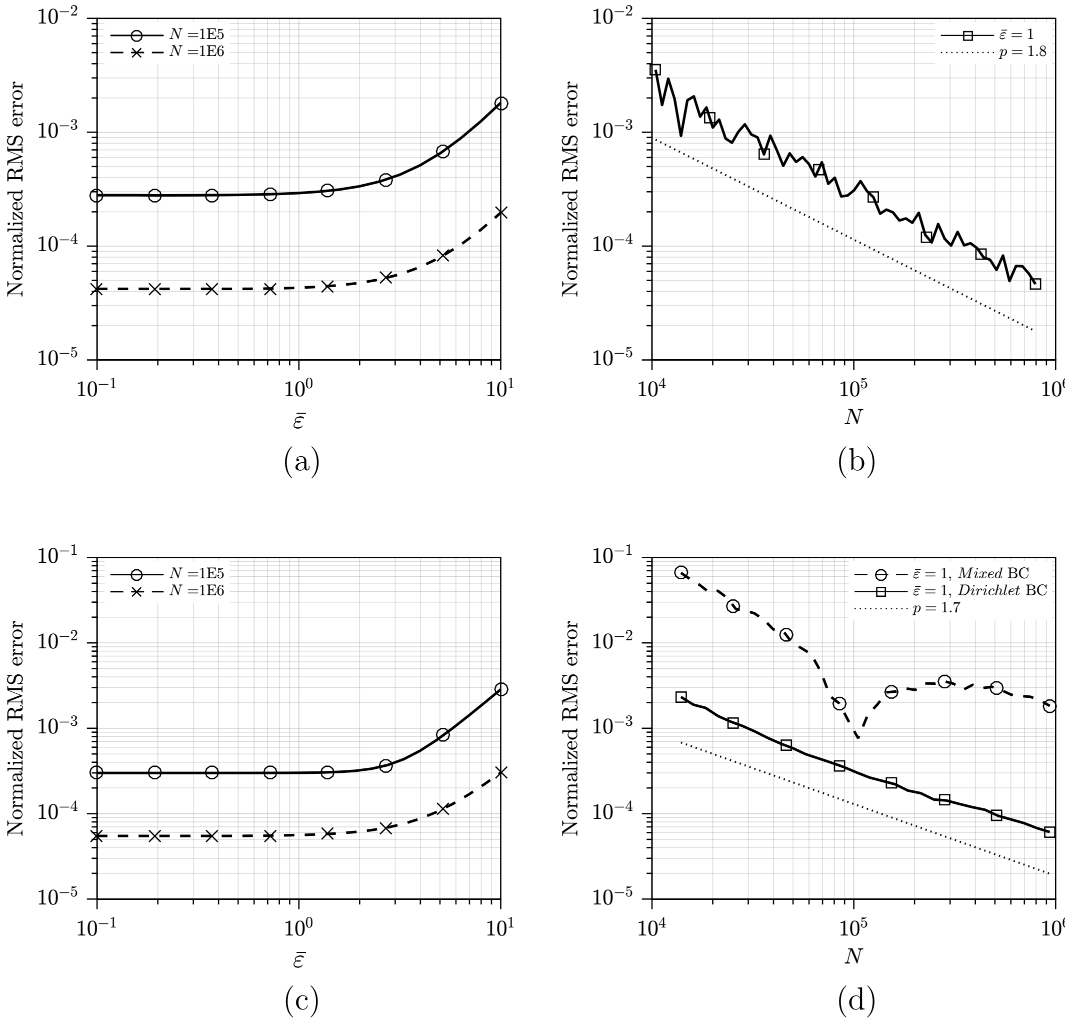


Figure 4.6: Normalized RMS error as function of rescaled shape factor $\bar{\varepsilon}$, Case 1 (a) and Case 2Dir (c), and as function of number of the nodes N, Case 1 (b) and Case 2 (d).

4.4.2. Case 1

Multicloud parameters

The influence of MC parameters on the convergence work, i.e., the work needed to reach a normalized RMS residual less than 10^{-14}, is reported in Figure 4.7 for ACMC and SRMC employed both as standalone solver and as preconditioners for BiCGSTAB in the case $N \approx 10^5$ nodes.

Figure 4.7a shows that for the standalone ACMC the convergence work decreases with the overcorrection factor t as expected, until a minimum is reached

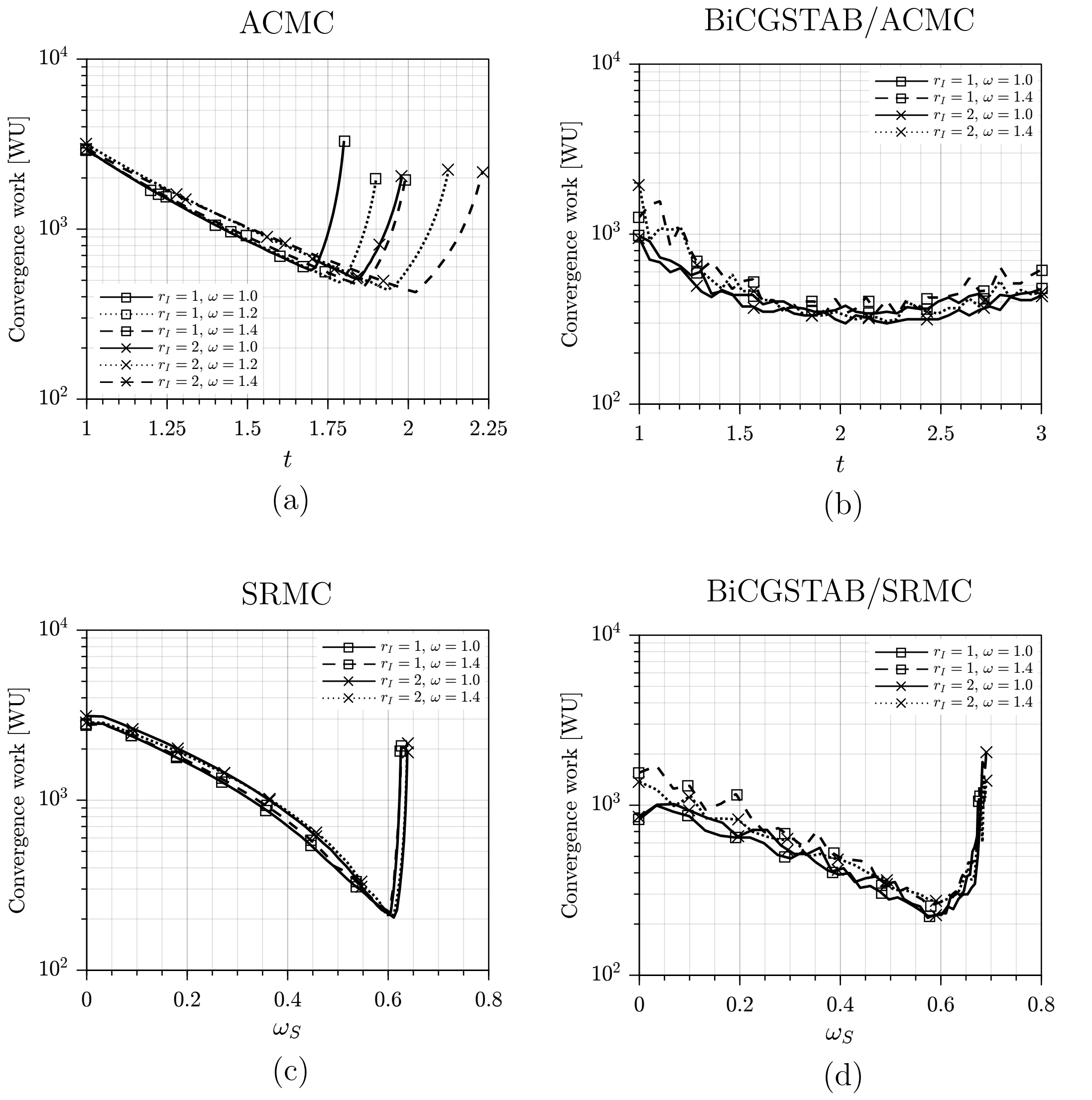


Figure 4.7: Effect of ACMC overcorrection parameter t (a,b) and SRMC smoothing factor ω_S (c,d) on the convergence work, $N \approx 10^5$ nodes, Case 1.

for an optimal value of t depending upon the choice for SOR iteration multiplier r_I and SOR relaxation factor ω. Beyond this optimal value, which lies in the range $[1.7, 2.0]$ for the MC parameters considered here, the convergence work quickly increases and therefore an accurate choice for t is crucial. The convergence works for ACMC used as preconditioner for BiCGSTAB are reported in Figure 4.7b, where the almost flat curves reveal that this strategy is more effective than the standalone ACMC strategy because its sensitivity to MC parameters is very low, especially for an overcorrection factor $t \in [1.5, 3.0]$.

Convergence works for standalone SRMC are reported in Figure 4.7c where it can be observed that the effect of r_I and ω is very limited and therefore there is a "single" optimal smoothing factor $\omega_S \approx 0.6$, beyond which the growth of the convergence work is very steep. Once again, the effect of using SRMC as preconditioner for BiCGSTAB, shown in Figure 4.7d, is beneficial since the curves show a smoothed behaviour near the optimal $\omega_S \approx 0.6$, but the steep growth beyond this optimal value is present as well.

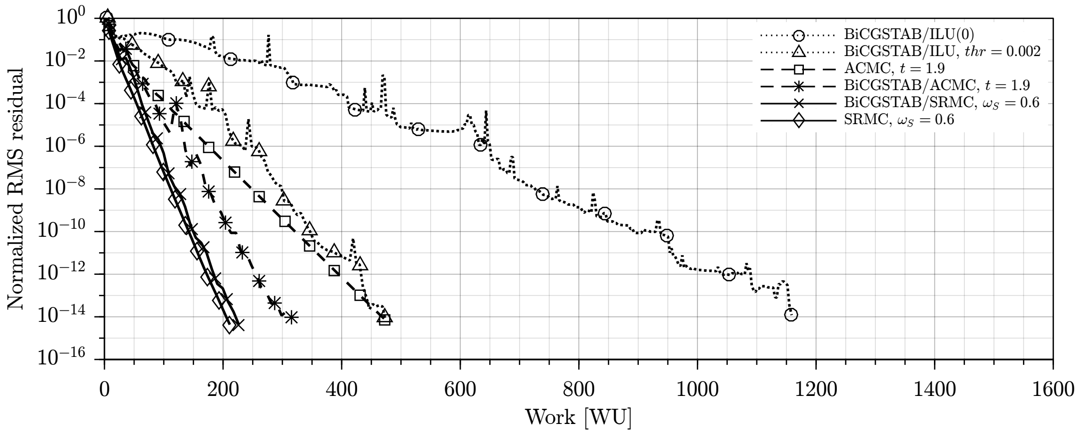
An initial comparison between the previous strategies reveals that the preconditioned versions are always convenient. BiCGSTAB/ACMC strategy is less sensitive than BiCGSTAB/SRMC to the choice of MC parameters, while BiCGSTAB/SRMC strategy can offer a slightly smaller optimal convergence work than BiCGSTAB/ACMC.

Multicloud results

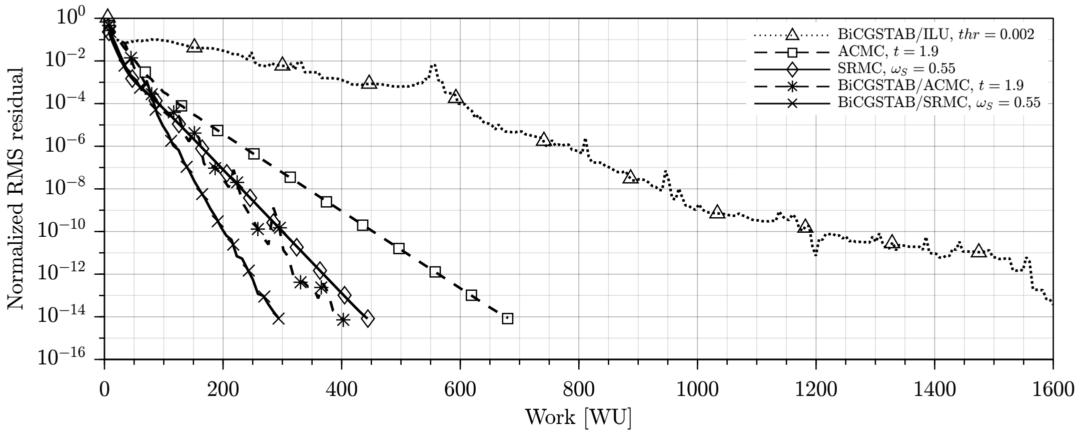
Figure 4.8 shows the convergence histories in the case of $N \approx 10^5$ and $N \approx 10^6$ nodes for all the MC strategies previously proposed and also for BiCGSTAB with incomplete LU factorization (ILU) preconditioning and reverse Cuthill-McKee ordering [19]. In the case of $N \approx 10^5$ nodes, Figure 4.8a, each MC strategy has shown to be much more effective in the reduction of the residual than BiCGSTAB with ILU(0) preconditioning, i.e., 0 level of fill in. Comparable performances between ACMC and BiCGSTAB/ILU can be obtained employing a small ILU factorization drop tolerance $thr = 0.002$, which implies non-negligible memory requirements and time consumptions in the initial factorization phase. In the case of $N \approx 10^6$ nodes, Figure 4.8b, shows that BiCGSTAB/ILU(0) is practically unfeasible (its convergence history is not even reported since it converged in much more than 1600 WU), the convergence work for BiCGSTAB/ILU with the same drop tolerance $thr = 0.002$ grows rapidly over 1600 WU while the convergence works for the MC strategies show a moderate growth which is subsequently analysed in a greater detail.

The convergence works for each MC strategy and also for two BiCGSTAB/ILU cases ($thr = 0.002$ and $thr = 0.0005$, the latter requiring twice the memory required by the former) are reported in Figure 4.9 for N ranging from $2 \cdot 10^4$ to $8 \cdot 10^6$. The most evident fact is that the growth of convergence work for both BiCGSTAB/ILU cases is considerably larger than the MC ones, and such strategies become uncompetitive with MC strategies for $N > 10^6$ nodes, taking also account of the fact that drop tolerances below $thr = 0.0005$ are practically unfeasible due to high memory requirements for such large problem sizes. For $N < 10^5$ nodes both BiCGSTAB/ILU cases are comparable or even better than the proposed MC approaches.

BiCGSTAB/SRMC with $\omega_S = 0.55$ turns out to be the most efficient strategy, whose convergence work grows very slowly from 300 WU for $N \approx 2 \cdot 10^4$ nodes



(a)



(b)

Figure 4.8: Comparison of convergence histories for $N \approx 10^5$ nodes (a) and $N \approx 10^6$ nodes (b), Case 1.

to 400 WU for $N \approx 8 \cdot 10^6$ nodes. BiCGSTAB/ACMC with $t = 1.9$ performs almost identically. The less efficient strategy, as expected, is standalone ACMC with $t = 1.9$, whose convergence work grows from 400 WU for $N \approx 2 \cdot 10^4$ nodes to 900 WU for $N \approx 8 \cdot 10^6$ nodes.

The comparison between the specific times, i.e., time per node, is reported in Figure 4.10 for each of the previous strategies considering the time required to reach a normalized RMS residual less than 10^{-14}, and also for two ILU factorization cases ($thr = 0.002$ and $thr = 0.0005$) considering the initial factorization time. 8 OpenMP threads have been employed for the parallelized C code. Similarly to the behaviour of convergence work, specific times for both BiCGSTAB/ILU cases

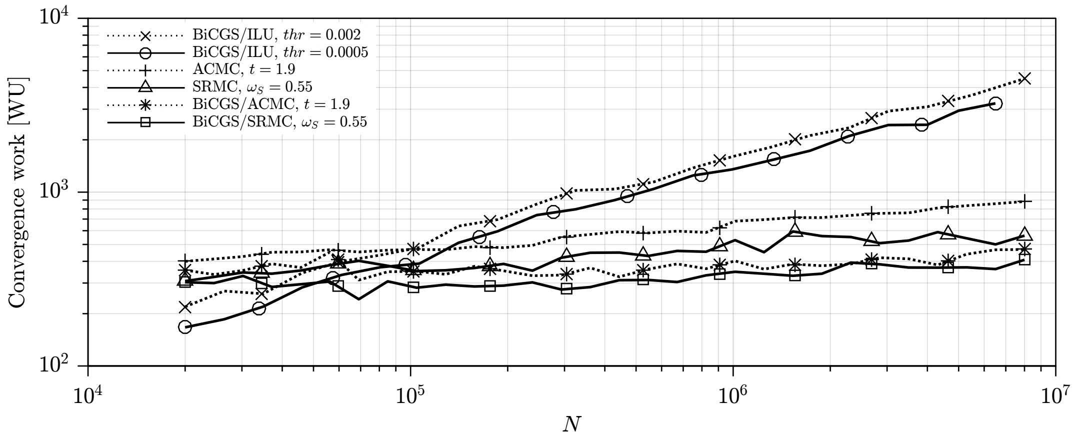


Figure 4.9: Convergence work vs number of nodes N, Case 1.

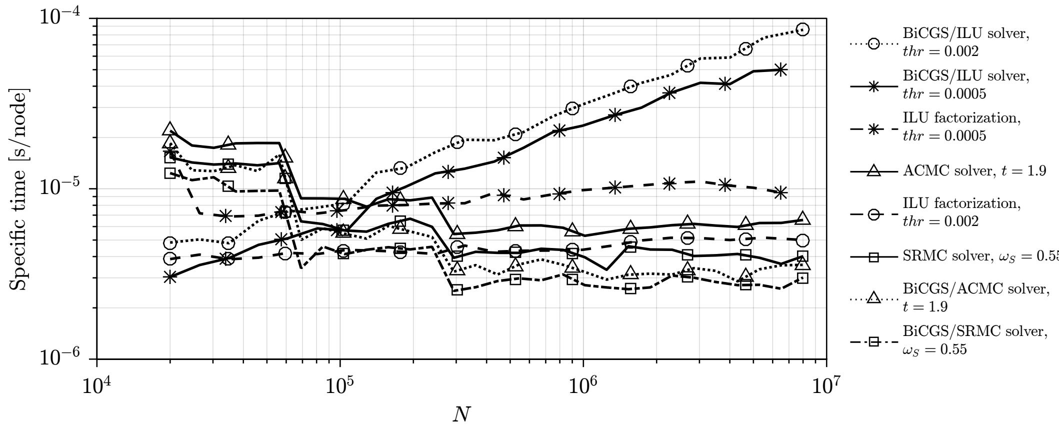


Figure 4.10: Specific time vs number of nodes N, Case 1.

show a larger growth than MC ones and become uncompetitive compared to each MC strategy beyond $N \approx 2 \cdot 10^5$ nodes, while the specific times for MC strategies turn out to be almost constant for $N > 3 \cdot 10^5$. Again, BiCGSTAB/SRMC with $\omega_S = 0.55$ turns out to be the best (faster) strategy, requiring approximately $3 \cdot 10^{-6}$ s/node for $N > 3 \cdot 10^5$. BiCGSTAB/ACMC with $t = 1.9$ performs almost identically, while standalone ACMC with $t = 1.9$ is the less performing and shows a very moderate growth in specific time for high N. The speedup between a traditional approach like BiCGSTAB/ILU with drop tolerance $thr = 0.0005$ and BiCGSTAB/SRMC with $\omega_S = 0.55$ ranges from 4 for $N \approx 3 \cdot 10^5$ nodes, to almost 20 for $N \approx 8 \cdot 10^6$ nodes.

The specific times for the remaining main tasks are almost constant and are as follows: $3.5 \cdot 10^{-7}$ s/node for matrix coefficients calculation (discretization) and

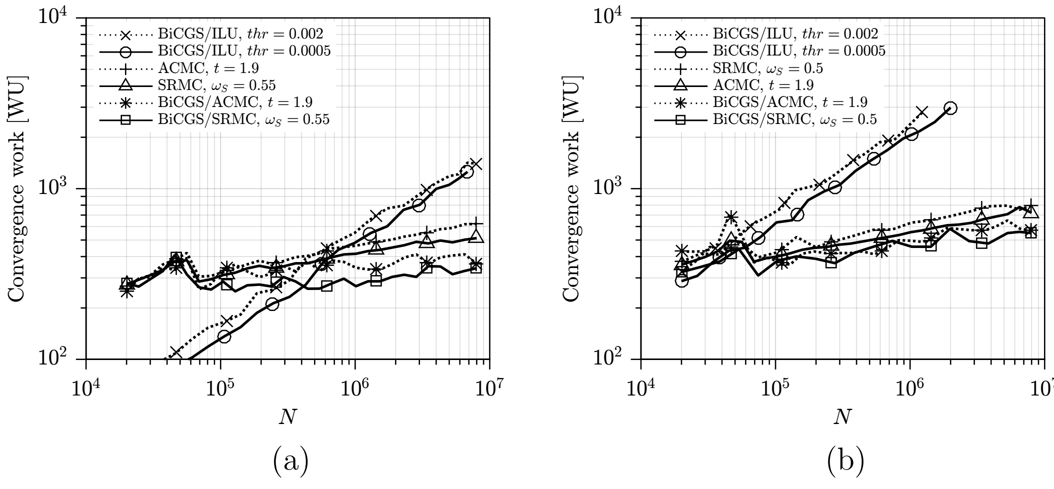


Figure 4.11: Convergence work vs number of nodes N for Case 2Dir (a) and Case 2Mix (b).

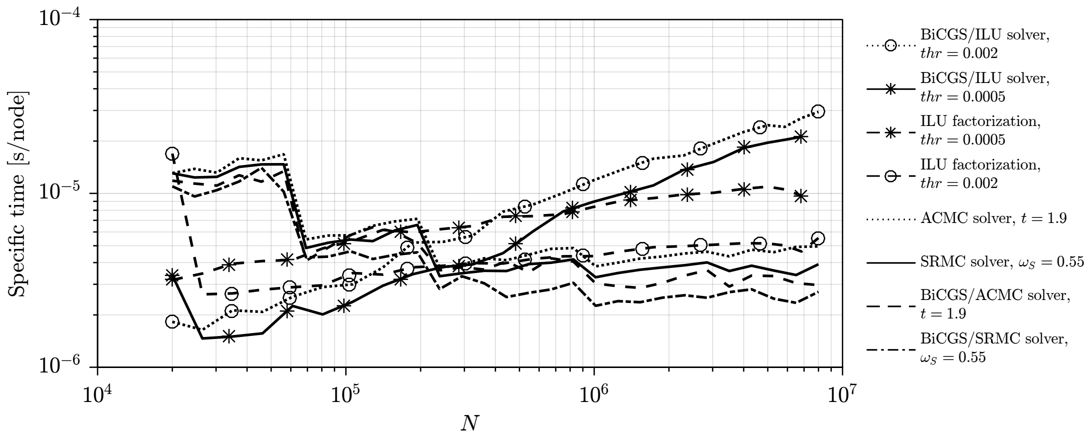


Figure 4.12: Specific time vs number of nodes N for Case 2Dir.

$1.5 \cdot 10^{-7}$ s/node for the MC setup phase (coarsening and coarse-level coefficients calculation).

4.4.3. Case 2

Multicloud results

The convergence works for each MC strategy and also for two BiCGSTAB/ILU cases ($thr = 0.002$ and $thr = 0.0005$) are reported in Figure 4.11 for N ranging

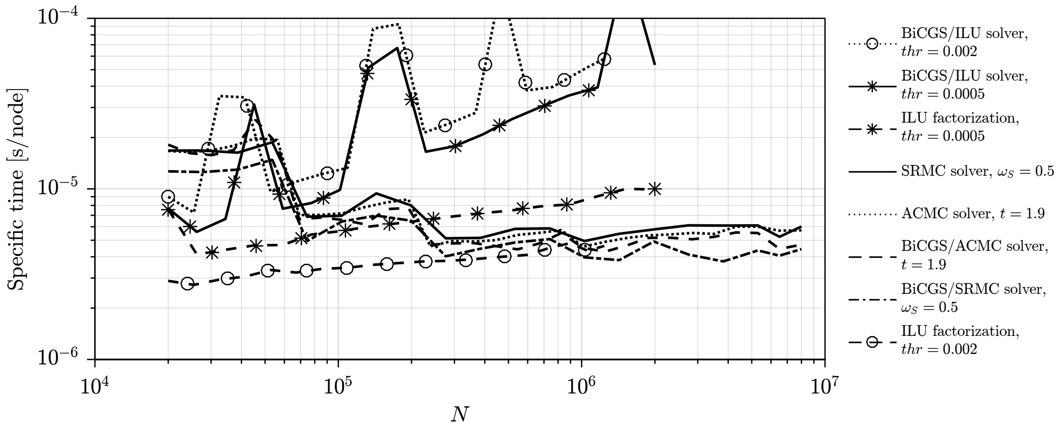


Figure 4.13: Specific time vs number of nodes N for Case 2Mix.

from $2 \cdot 10^4$ to $8 \cdot 10^6$, for both Case 2Dir and Case 2Mix. Again, as in Case 1, the most evident fact is that the growth rate of convergence work for BiCGSTAB/ILU cases is considerably larger than the MC ones for both types of BCs. In Case 2Dir, Figure 4.11a, MC strategies outperform BiCGSTAB/ILU strategies only for $N > 4 \cdot 10^5$ nodes, while in Case 2Mix, Figure 4.11b, MC strategies outperform BiCGSTAB/ILU strategies from $N > 6 \cdot 10^4$ nodes already. Both BiCGSTAB/ILU strategies turned out to be unfeasible due to both high time consumption and large memory requirements for $N > 2 \cdot 10^6$ nodes in Case 2Mix.

BiCGSTAB/SRMC with $\omega_S = 0.55$ is again the most efficient strategy for both BCs and BiCGSTAB/ACMC with $t = 1.9$ perform almost identically. In Case 2Dir the convergence work for these strategies is almost constant ($\approx 300 - 400$ WU), while in Case 2Mix the convergence work shows a moderate growth from 300 WU for $N \approx 2 \cdot 10^4$ nodes to 550 WU for $N \approx 8 \cdot 10^6$ nodes. Therefore the less efficient strategies are again standalone ACMC and SRMC, especially in Case 2Dir where their convergence work shows a moderate but constant growth from $N \approx 3 \cdot 10^5$ nodes.

The comparison between the specific times is reported in Figures 4.12 and 4.13 for each of the previous strategies (normalized RMS residual less than 10^{-14}), and also for two ILU factorizations ($thr = 0.002$ and $thr = 0.0005$) for both cases 2Dir and 2Mix. 8 OpenMP threads have been employed for the parallelized C code. Similarly to the behaviour of convergence work, specific times for BiCGSTAB/ILU cases show a larger growth than MC ones and become uncompetitive compared to each MC strategy beyond $N \approx 2 \cdot 10^5$ nodes in Case 2Dir, (Figure 4.12), while in Case 2Mix, (Figure 4.13), the MC strategies outperforms BiCGSTAB/ILU strategies from $N \approx 7 \cdot 10^4$ nodes already. Specific times for all MC strategies are practically constant for $N > 3 \cdot 10^5$ nodes. The specific times for the remaining main tasks (node generation, matrix coefficients calculation and MC setup phase) remain the same as in Case 1.

The speedup between a traditional approach like BiCGSTAB/ILU with drop tolerance $thr = 0.0005$ and BiCGSTAB/SRMC with $\omega_S = 0.55$ ranges from 4 for $N \approx 10^6$ nodes, to 8 for $N \approx 8{\cdot}10^6$ nodes, in Case 2Dir. In Case 2Mix such speedup ranges from 2 for $N \approx 10^5$ nodes, to 10 for $N \approx 2 \cdot 10^6$ nodes. Therefore the use of MC strategies can bring great advantages especially in case of Neumann BCs, which is a typical condition encountered in many practical applications such as Poisson problems in computational fluid dynamics. Furthermore, in these cases the proposed MC strategies are able to deal with very large problems where traditional BiCGSTAB/ILU approaches can not deal with.

4.4.4. OpenMP speedup

The measured speedup values obtained using 2, 4 and 8 OpenMP threads for the parallelized C code for the MC algorithms are reported in Table 4.2 for Case 1 and cases 2Dir and 2Mix where $N \approx 10^6$ nodes. The speedup values are listed for each of the main tasks separately.

From the previous tables it can be observed that the matrix coefficients calculation is the task that exhibit the largest speedup ($4.3 - 4.7$ with 8 OpenMP threads), while the achieved speedup for the MC solvers is lower and does not exceed 2.7. In particular, the speedup for BiCGSTAB with MC preconditioning is always lower than the speedup for the corresponding standalone MC solver because BiCGSTAB is always parallelized on all available cores in MATLAB by default, as well as other vector operations (vector sums) that are carried out in MATLAB in the present implementation.

4.5. Conclusions

The proposed multicloud techniques for the fast solution of linear systems arising from RBF-FD discretizations of Poisson problems have proven to bring great advantages over traditional approaches, e.g., BiCGSTAB with incomplete LU factorization as preconditioner, in terms of both reduction of the convergence work and reduction of the required computational time. Such reductions can reach a factor 10 for convergence work and even a factor 20 for computational time in the case of extremely large problem sizes, e.g., $N \approx 8 \cdot 10^6$ nodes. These advantages over traditional approaches have proven to be particularly evident in the case of Neumann boundary conditions, which often characterize many problems of engineering relevance. Furthermore, in cases where Neumann boundary conditions

Table 4.2: Speedup values, $N \approx 10^6$ nodes.

	ACMC			SRMC		
Cores	**Cff**	**Slv-A**	**Bi/A**	**Cff**	**Slv-S**	**Bi/S**
			Case 1			
2	1.9	1.9	1.6	1.9	1.6	1.5
4	3.7	2.5	2.0	2.6	2.2	2.0
4 (8 threads)	4.7	2.7	2.1	4.4	2.4	2.3
			Case 2Dir			
2	1.8	1.6	1.5	1.8	1.8	1.5
4	2.7	2.1	1.9	2.7	2.4	2.0
4 (8 threads)	4.3	2.3	1.9	4.4	2.6	2.0
			Case 2Mix			
2	1.8	1.8	1.6	1.9	1.7	1.6
4	3.1	2.3	2.1	2.8	2.3	2.2
4 (8 threads)	4.3	2.5	2.2	4.3	2.4	2.1

Cff = matrix coefficients phase (meshless discretization)
Slv-A, **Slv-S** = solution phase with **A**=ACMC, **S**=SRMC
Bi-A, **Bi-S** = BiCGSTAB precond. with **A**=ACMC, **S**=SRMC

are employed, the proposed MC strategies have proven to be able to deal with very large size problems where traditional solver approaches can not deal with because of excessive memory and time requirements. Therefore, the proposed MC strategies are extremely attractive for problems where complex-shaped domains and large number of nodes are employed.

The extension of the proposed algorithms to 3D cases is straightforward since the RBF-FD data structure is implicitly unstructured, while further extensions can include higher order RBF-FD discretizations, i.e., larger stencil sizes $n > 7$ with $P > 1$, as well as the extension to different PDEs.

Chapter 5

RBF-FD solution of CFD problems

The employment of the RBF-FD method for solving fluid flow problems has received increasing attention in both academic and industrial fields since the early 2000s, when such approach has been initially proposed and proved to be competitive with traditional mesh-based methods [12, 14, 15, 22–24, 58]. Many works dealing with a RBF-FD solution of CFD problems followed since then [31, 34, 60, 104, 105].

In this chapter the RBF-FD meshless approach presented in Chapter 3 are employed for the numerical solution of 2D and 3D incompressible fluid flow problems with possible heat transfer. The time dependent Navier-Stokes equations are solved using primitive variables, i.e., velocity and pressure, since this is the most generic formulation which can be easily employed to characterize and solve fluid flow problems with engineering relevance.

Different benchmark test cases will be considered in the 2D case: a laminar flow inside a circular domain, the lid-driven cavity, the differentially heated cavity and the flow past a circular cylinder between parallel walls. The 3D lid-driven cavity problem is then considered. Besides these test cases are characterized by simple geometries, the use of generic node arrangements as obtained with the node generation algorithms proposed in Chapter 2 will allow the assessment of the numerical properties of the RBF-FD approach to be extended to arbitrary 2D/3D geometries.

5.1. Governing equations

An incompressible and non-isothermal flow is considered, for which the conservation equations of mass, momentum and energy in the unsteady form are the following:

$$\nabla \cdot \mathbf{u} = 0 \tag{5.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 \mathbf{u} - \mathbf{g}\beta(T - T_0) \tag{5.2}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \alpha \nabla^2 T \tag{5.3}$$

where $\mathbf{u}$ is the velocity vector, ν is the kinematic viscosity, α is the thermal diffusivity, T is the temperature, $\mathbf{g} = -g\mathbf{e}_y$ is the gravity acceleration and p is the pressure, deprived of the hydrostatic component. The Boussinesq approximation is employed, i.e., constant thermophysical properties except for the density in the buoyancy term which has a linearized dependence on the temperature:

$$\frac{\rho}{\rho_0} = -\beta(T - T_0) \tag{5.4}$$

where T_0 and ρ_0 are the reference values for temperature and density in the linearization, respectively. In the case of isothermal flows, the energy Eq. (5.3) is not considered and the buoyancy term is dropped, i.e., $\beta = 0$.

Eqs. (5.1)-(5.3) can be made nondimensional using the following reference values: L for length, u_0 for velocities, L/u_0 for time, ΔT for temperature and $\rho_0 u_0^2$ for pressure. For isothermal flows, the flow Reynolds number is defined as $Re = u_0 L/\nu$, while for non-isothermal flows the Rayleigh number $Ra = g\beta\Delta T L^3/(\nu\alpha) = u_0^2 L^2/(\nu\alpha)$ and the Prandtl number $\mathrm{Pr} = \nu/\alpha$ are chosen as nondimensional groups. A Prandtl number $Pr = 0.71$ is chosen, which is representative of air.

The nondimensional form of Eqs. (5.1)-(5.3) is therefore:

$$\nabla \cdot \mathbf{u} = 0 \tag{5.5}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{D_u}\nabla^2 \mathbf{u} + \mathbf{B} \tag{5.6}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \frac{1}{D_T}\nabla^2 T \tag{5.7}$$

where $\mathbf{B} = (T - T_0)\mathbf{e}_y$ in the non-isothermal cases; T_0 is assumed to be 0 without loss of generality. The values for D_u, D_T and $\mathbf{B}$ are reported in Table 5.1 for isothermal and non-isothermal cases.

Table 5.1: Parameters for Eqs. (5.6)-(5.7) for isothermal and non-isothermal cases.

Case	D_u	D_T	$\mathbf{B}$
Isothermal	Re	–	$\mathbf{0}$
Non-isothermal	$\sqrt{Ra/Pr}$	$\sqrt{Ra \cdot Pr}$	$T\mathbf{e}_y$

5.2. Numerical procedure

5.2.1. Time discretization and solution procedure

A second order backward Euler scheme (three-level Gear scheme) is employed for the time discretization of Eqs. (5.6)-(5.7). Such choice is found to be much more stable than the second order Crank-Nicolson scheme for which divergent spurious oscillations with period $2\Delta t$ appeared in some case, and it is therefore preferred.

A segregated approach is chosen for the decoupling of the system of Eqs. (5.5)-(5.7) which are solved separately for each variable. The projection approach [16] is employed for the decoupling of mass and momentum equations. In this approach the pressure p assumes the role of a distributed Lagrange multiplier which enforces the continuity constraint expressed by Eq. (5.5). At each iteration k within the time step Δt, a tentative velocity $\mathbf{u}^*$ is computed by implicitly solving the momentum Eq. (5.6) with a linearized advective (non-linear) term as follows:

$$\frac{3\mathbf{u}_k^*}{2\Delta t} + \mathbf{u}_{k-1}^{n+1}\nabla\mathbf{u}_k^* - \frac{1}{D_u}\nabla^2\mathbf{u}_k^* = \frac{4\mathbf{u}^n - \mathbf{u}^{n-1}}{2\Delta t} - \nabla p_{k-1}^{n+1} + \mathbf{B}_{k-1}^{n+1} \tag{5.8}$$

where the values with iteration index $k-1$ are extrapolated at the first iteration $k=1$ as follows: $\mathbf{u}_0^{n+1} = (\gamma+1)\mathbf{u}^n - \gamma\mathbf{u}^{n-1}$, $\mathbf{B}_0^{n+1} = (\gamma+1)\mathbf{B}^n - \gamma\mathbf{B}^{n-1}$, while the pressure is always initialized as $p_0^{n+1} = p^n$. $\gamma = 1$ is employed for the accurate simulation of time-dependent flows, while $\gamma = 0$ is employed when a steady-state solution is expected, allowing the use of large time steps.

The tentative velocity $\mathbf{u}_k^*$ is then projected onto the space of divergence-free fields in order to satisfy the continuity constraint [16], i.e., $\mathbf{u}_k^*$ is deprived from its irrotational component $\nabla\phi$:

$$\mathbf{u}_k^{n+1} = \mathbf{u}_k^* - \nabla\phi \tag{5.9}$$

The equation for the potential ϕ is obtained by taking the divergence of Eq. (5.9) and enforcing the continuity constraint $\nabla \cdot \mathbf{u}_k^{n+1} = 0$, obtaining the following Poisson equation:

$$\nabla^2\phi = \nabla \cdot \mathbf{u}_k^* \tag{5.10}$$

subject to the boundary condition $\nabla\phi \cdot \mathbf{n} = 0$ on the whole boundary Γ in order not to modify the normal component of the tentative velocity $\mathbf{u}_k^*$ at the boundary.

Therefore, after the velocity correction of Eq. (5.9), the normal velocity still satisfies the boundary condition (BC) imposed for the solution of Eq. (5.8) since $\mathbf{u}_k^{n+1} \cdot \mathbf{n} = \mathbf{u}_k^* \cdot \mathbf{n}$. This does not hold for the remaining components of the velocity for which the exact satisfaction of the BCs occurs only when the iterative process within the time step converges. The Poisson Eq. (5.10) has to be solved at each time step in order to compute the velocity correction of Eq. (5.9).

The pressure is then updated as:

$$p_k^{n+1} = p_{k-1}^{n+1} + \frac{\phi}{\Delta t} \tag{5.11}$$

Lastly, the temperature is computed by implicitly solving the energy Eq. (5.7) for T_k^{n+1}:

$$\frac{3T_k^{n+1}}{2\Delta t} + \mathbf{u}_k^{n+1}\nabla T_k^{n+1} - \frac{1}{D_T}\nabla^2 T_k^{n+1} = \frac{4T^n - T^{n-1}}{2\Delta t} \tag{5.12}$$

where the divergence-free velocity $\mathbf{u}_k^{n+1}$ is now considered for the advective term.

The whole system of Eqs. (5.8)-(5.12) is iterated k_{SUB} times for each time step in order to properly address the non-linear coupling between the different variables, especially when using large time-steps. $k_{SUB} = 4$ subiterations are typically employed for the accurate simulation of time-dependent flows, while $k_{SUB} = 1$ iteration is employed when a steady-state solution is expected, resulting in a very cheap time-marching procedure for the reaching of the steady-state.

Since implicit solvers are employed for both the computation of velocities, Eq. (5.8), and temperature, Eq. (5.12), there is no stability requirement for the choice of the time step Δt, i.e., CFL condition, and therefore it is possible to employ large time steps, i.e., large Courant numbers $C = \Delta t\|\mathbf{u}\|/s \gg 1$ where s is the nodal spacing. Large time steps are typically employed for a fast integration in time when a steady-state solution is expected.

5.2.2. Solution techniques

The RBF-FD discretization of the Poisson Eq. (5.10) leads to a constant coefficient matrix, therefore it is convenient to perform a single LU factorization for this matrix at the beginning of the simulation and use it at each time step for the calculation of the potential ϕ, allowing a very fast and accurate solution for both 2D and 3D problems. Since Neumann boundary conditions are employed, the Poisson equation is undetermined up to a constant and the uniqueness of the solution is obtained by using a Lagrange multiplier λ:

$$\nabla^2\phi + \lambda = \nabla \cdot \mathbf{u}_k^* \tag{5.13a}$$

$$\int_\Omega \phi = 0 \tag{5.13b}$$

which corresponds to adding a constant row and a constant column to the coefficient matrix, with a 0 on the main diagonal.

The RBF-FD coefficient matrices for momentum and energy Eqs. (5.8), (5.12) are time-dependent because of the advection term which depends upon the velocity, therefore the previous LU approach cannot be used. The BiCGSTAB iterative solver [97] is employed, using an incomplete LU (ILU) factorization [85] as preconditioner and reverse Cuthill-McKee ordering [19]. The relative tolerance for the residuals is set to 10^{-10} for 2D and small size 3D problems, while for large size 3D problems the relative tolerance for the residual is set to 10^{-8}.

During the transients, i.e., starting from resting fluid or restarting from a previous solution at lower Re/Ra numbers, as well as for time-dependent flows with strong recirculations, the ILU factorization is performed whenever the number of iterations IT needed by the BiCGSTAB to converge exceeds $IT > 2IT_0$, where IT_0 is the number of iterations required immediately after the ILU factorization. The employed ILU drop tolerance tol_{ILU} varies depending upon the size of the problem N; a typical value is $tol_{ILU} = 10^{-3}$. The typical computing time for 2D fluid-flow problems is $0.2 - 0.8$ s/(time step) when $N \approx 40,000$ nodes are employed, depending upon the size of the stencil n, the Re/Ra number and the time step Δt. For the 3D lid-driven cavity, the computing time is roughly 3 to 6 times larger than the time required by the corresponding 2D lid-driven problem.

5.2.3. Pressure-velocity coupling and stabilization

It is known that the discretization schemes for the pressure-velocity coupling in the incompressible Navier-Stokes equations must fulfill certain conditions in order to ensure the uniqueness of the discrete solution. For example, the unequal-order interpolation is employed in the FEM in order to satisfy the Ladyzhenskaya–Babuška–Brezzi (LBB) condition; staggered grids are employed in the FVM to avoid the appearance of spurious pressure modes.

Spurious pressure modes also appear with RBF-FD discretizations. This problem was initially addressed by considering an additional node distribution for the pressure, which has twice the spacing of the starting node distributions. The potential ϕ was still solved on the original node distribution and was also used for the velocity correction on the starting node distribution, Eq. (5.9). ϕ was then interpolated onto the coarse node distribution for the pressure update, Eq. (5.11), and for the calculation of pressure gradient in Eq. (5.8). A very small pressure smoothing was anyway required for the obtaining of a stable scheme. Such procedure was then abandoned because its implementation is impractical since two node distributions and two RBF-FD discretizations are required; furthermore, the pressure discretization is always less accurate than the velocity discretization since a coarse node distribution is involved.

A stable “equal-order” RBF-FD discretization scheme is then obtained by applying the following explicit hyperviscosity smoothing (see Subsection 3.2.2) to the pressure:

$$\tilde{p} = p + \delta \nabla^{2k} p \tag{5.14}$$

where $\tilde{p}$ is the smoothed pressure and δ is a small constant. Similarly to the analysis carried out in Subsection 3.2.2, let us consider a harmonic component $p = Ae^{j\boldsymbol{\omega}\cdot\mathbf{x}/s}$ with amplitude A for the pressure. Eq. (5.14) for this harmonic component gives:

$$\tilde{p} = A\left[1 + \delta \frac{\|\boldsymbol{\omega}\|_2^{2k}}{s^{2k}}(-1)^{k+1}\right] e^{j\boldsymbol{\omega}\cdot\mathbf{x}/s} \tag{5.15}$$

which states that the amplification factor for the harmonic component with frequency $\boldsymbol{\omega}$ is given by the term between the square brackets. The effect of explicit smoothing with hyperviscosity is very similar to the hyperviscosity stabilization: high-frequency components are effectively reduced while low-frequency components sustain small reductions; the quality of the smoothing depends upon k. Since we want the amplification factor to be independent upon the spacing s, i.e., independence upon space scaling, we obtain $\delta = \bar{\delta}_k s^{2k}(-1)^{k+1}$ which is the same expression obtained in Subsection 3.2.2. The higher the hyperviscosity exponent k (k applications of the laplacian), the faster the decay of the artificial hyperviscosity when decreasing s, i.e., increasing the number of nodes N. The explicit hyperviscosity smoothing is found to be very effective in avoiding the appearance of spurious pressure modes even with very small hyperviscosity factors $\bar{\delta}_k$, e.g., $\bar{\delta}_3 = \mathcal{O}(10^{-6})$ with $k = 3$ and $N \approx 100,000$ nodes in 2D.

The explicit hyperviscosity smoothing is also employed for the velocities, allowing the use of different number of interpolation nodes n for the RBF-FD discretization of the Navier-Stokes equations and n_H for the RBF-FD discretization of the hyperviscosity term ∇^{2k}. $n_H > n$ is typically chosen because the hyperviscosity smoothing is performed explicitly while the solution of the momentum equation (5.8) is performed implicitly and therefore the costs of a larger stencil are not justified. Furthermore, $n_H > n$ is usually required also because a high-order hyperviscosity $k \geq 2$ is preferable (the laplacian smoothing $k = 1$ introduces unacceptable numerical diffusion). The application of the explicit hyperviscosity smoothing to the velocity can also overcome the problem of convective instabilities arising when dealing with convection-dominated flows as presented in Subsection 3.2.2. The need of more complex upwinding techniques [108] is therefore avoided, still maintaining small or even negligible artificial effects. The use of the same amount of artificial hyperviscosity $\bar{\delta}_k$ for both pressure and velocity is found to be an effective choice. A practical way to determine the minimal $\bar{\delta}_k$ which guarantees the stability in each specific case is to perform very short test runs employing zero boundary conditions for the velocity and starting from a small random initial field for velocity and pressure, e.g., in the order of 10^{-3}. $\bar{\delta}_k$ is then increased from 0 until all the variables exhibit a uniform decrease to 0 after the initial overshoot due to the initial conditions.

5.2.4. Auxiliary computations

The calculation of the streamfunction ψ, required for vortices detection and graphical evaluation of the flow features for the cavity problems and the flow past a cylinder between parallel walls, is carried out by solving the following Poisson equation:

$$\nabla^2 \psi = -\nabla \times \mathbf{u} \tag{5.16}$$

subject to the boundary condition $\psi = \bar{\psi}$. The boundary streamfunction $\bar{\psi}$ is constant at all solid walls, while $\bar{\psi}$ is obtained from the imposed velocities at the inlet and from the solved velocities at the outlet, by integrating the normal component. Eq. (5.16) is discretized through the same RBF-FD approach employed for the discretization of the Poisson Eq. (5.10) and solved using a direct LU decomposition.

The local Nusselt number Nu_y at the cold wall for the the differentially heated cavity is obtained using the following expression:

$$\mathrm{Nu}_y = \frac{T_m - T_C}{1 - x_m} \tag{5.17}$$

where m is the index of the internal node $\mathbf{x}_m = (x_m, y_m)^T$ which is closest to the point $\mathbf{x} = (1, y)^T$ and T_m is its temperature. The mean Nusselt number $\overline{\mathrm{Nu}}$ on the cold wall is therefore given by $\overline{\mathrm{Nu}} = \int_0^1 \mathrm{Nu}_y \, \mathrm{d}y$.

The drag coefficient C_D for the case of the flow past a circular cylinder between parallel walls is defined as follows:

$$C_D = \frac{2F_x}{\rho_0 u_0^2 L d_{cyl}} \tag{5.18}$$

where F_x is the x-component of the force exerted by the fluid on the cylinder, which is time-averaged over a period $\bar{t}$ in the case of unsteady periodic flows.

5.2.5. Node generation

The required node distributions are obtained through the node generation techniques presented in Chapter 2: the DQT (2D) and DOT (3D) algorithms are employed to generate an initial node distribution which is then refined by the application of $k = 100$ node-repel iterations. A fixed boundary node distribution is employed in 2D cases, while the boundary projection technique is employed in the 3D case.

5.3. 2D CASES

For all the 2D cases, the $k = 3$ explicit hyperviscosity smoothing (∇^6) with $n_H = 40$ interpolation nodes is employed, except for the case of the flow past a circular cylinder between parallel walls where a laplacian smoothing with $n_H = n = 7$ is employed.

5.3.1. Isothermal laminar flow inside a circular domain

Geometry, parameters, initial and boundary conditions

A circular domain with radius L is considered. No-slip conditions are imposed on the fixed circular boundary, while the nondimensional initial conditions at $t = 0$ are given by the analytical solution reported in Appendix E for $M = 50$ terms in the expansion. Such analytical solution is characterized by a tangential velocity profile which is depicted in Figure 5.2a for the chosen value $Re = 1000$ and for three different times. The reference velocity u_0 is the maximum of the initial tangential velocity at $t = 0$ in the limit $M \to \infty$.

Node distributions

The node distribution is obtained by using the following spacing function:

$$\frac{s(r)}{s_m} = 1 + 4\frac{\operatorname{atan}\big(20(1-r)\big)}{\operatorname{atan}(20)} \tag{5.19}$$

where r is the nondimensional radius and s_m and s_M are the minimum and the maximum spacing function. The radial profile of the spacing function is depicted in Figure 5.1b while an example of a node distribution with $N \approx 7500$ nodes is depicted in Figure 5.1a. A refined distribution is employed near the boundary since the initial velocity profile u_ϑ exhibits a large gradient at $r = 1$ and a large curvature near $r = 0.95$ which is rapidly smoothed by the viscous dissipation.

Results

The isothermal Navier-Stokes equations are integrated starting from the initial conditions at $t = 0$ for π time units, corresponding to half a revolution for a constant tangential velocity $u_\vartheta(r = 1) = 1$. The chosen time step is $\Delta t = \pi/500$ and $k_{SUB} = 4$ subiterations are employed. With these parameters the time discretization error is very small and negligible when compared to the RBF-FD discretization errors.

Calculations are carried out for polynomial degree $P = 2, 3, 4, 5$ using $n = 20, 25, 30, 40$ interpolation nodes, respectively and $s \cdot \varepsilon = 0.4$. The employed

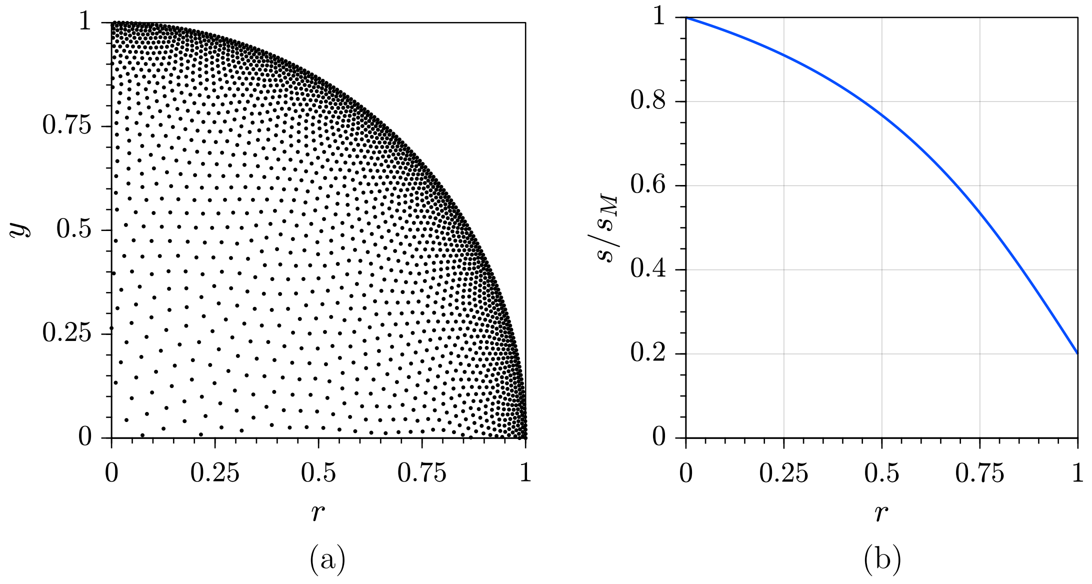


Figure 5.1: Node distribution with $N \approx 7500$ nodes (a) and the corresponding spacing function (b).

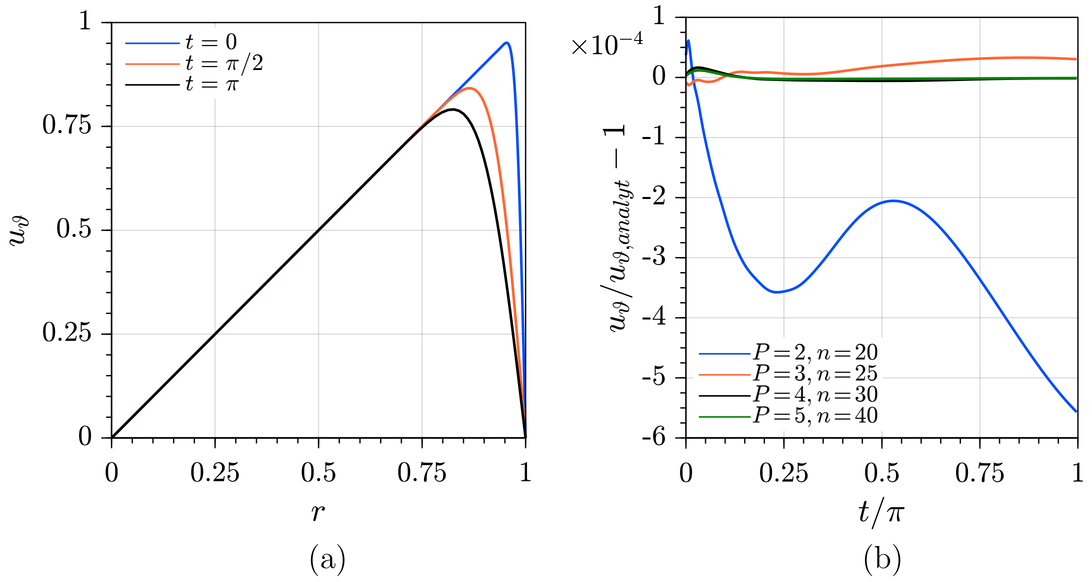
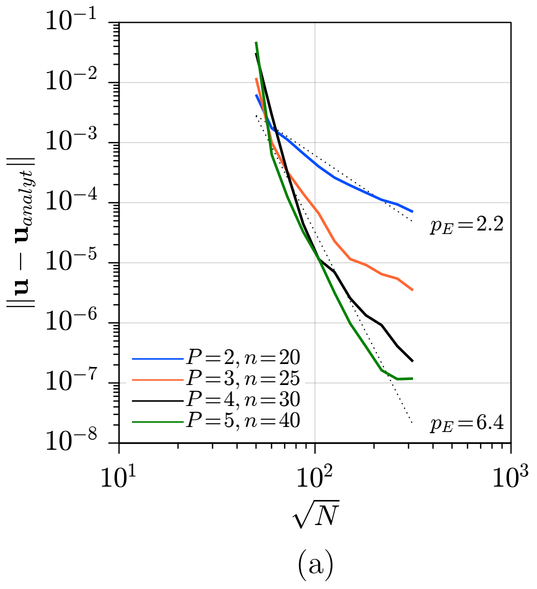


Figure 5.2: (a) analytical velocity profiles ($Re = 1000, M = 50$) at different times. (b) relative error of the tangential velocity at $r = 0.95$ for $N \approx 100,000$ nodes.

amount of explicit hyperviscosity is $\bar{\delta}_3 = 4 \cdot 10^{-4}$. An example of the time history of the relative error between the computed solution and the analytical solution at



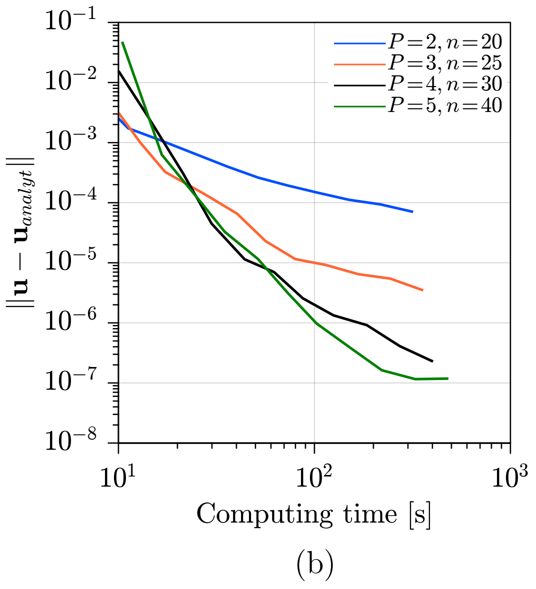


Figure 5.3: Error norms at $t = \pi$: convergence curves (a) and error norm as a function of the computing time (b).

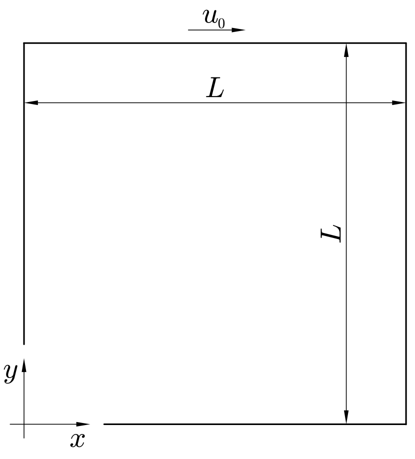
$r = 0.95$ is given in Figure 5.2b for $N \approx 100,000$ nodes. From this figure it can be seen how the relative error is drastically reduced by increasing the polynomial order P.

Convergence analysis are carried out by considering the following error norm for the difference $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{analyt}$ between the computed solution $\mathbf{u}$ and the analytical solution $\mathbf{u}_{analyt}$ at $t = \pi$:

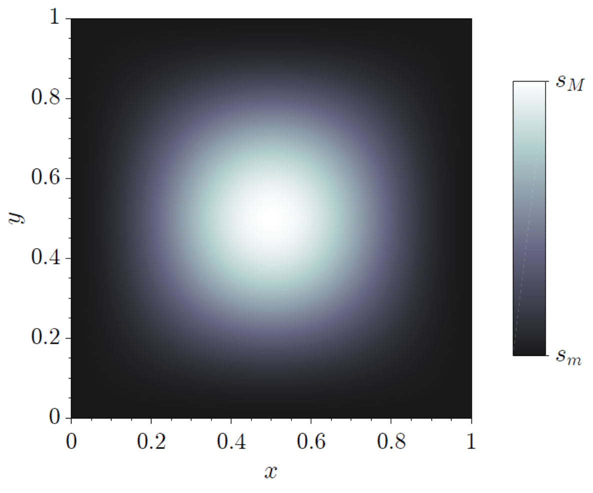
$$\|\delta\mathbf{u}\| = \sqrt{\frac{1}{2\pi}\int_\Omega \|\delta\mathbf{u}\|_2^2 \, d\Omega} \approx \sqrt{\frac{\sum_{i=1}^{n_I} s^2(\mathbf{x}_i)[\delta u^2(\mathbf{x}_i) + \delta v^2(\mathbf{x}_i)]}{2\sum_{i=1}^{n_I} s^2(\mathbf{x}_i)}} \tag{5.20}$$

where $\delta\mathbf{u} = (\delta u, \delta v)^T$.

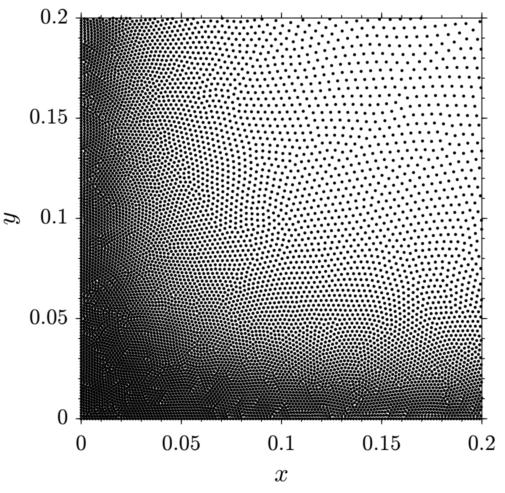
Figure 5.3a shows the convergence curves at $t = \pi$ for different polynomial orders P. The resulting order of accuracy p_E exhibits a monotone behaviour and increases from $p_E = 2.2$ for $P = 2$ to $p_E = 6.4$ for $P = 5$. Figure 5.3b shows the error norm at $t = \pi$ as a function of the computing time. These curves, which show a behaviour very similar to the convergence curves of Figure 5.3a, suggest that for this problem the increase of the polynomial order P always results in an increase of the numerical efficiency of the whole procedure.



(a)



(b)



(c)

Figure 5.4: Geometry (a), spacing function (b) and enlarged view of the node distribution for the bottom left corner (c) with $N \approx 80,000$ nodes for the lid-driven cavity problem.

5.3.2. Lid driven cavity

Geometry and boundary conditions

The lid-driven cavity problem, Figure 5.4a, is defined by a square cavity with side length L where the top wall moves to the right with velocity u_0. The boundary conditions in terms of nondimensional variables are $\mathbf{u} = 0$ at $x = 0, 1$ and $y = 0$, $\mathbf{u} = (1, 0)^T$ at $y = 1$.

Spacing function

The spacing function employed for the node generation is the following:

$$\frac{s(x,y)}{s_M} = \frac{1}{5} + \frac{1}{5}\Big[1 + \cos\big(\pi(2x-1)^4\big)\Big]\Big[1 + \cos\big(\pi(2y-1)^4\big)\Big] \tag{5.21}$$

for which the maximum spacing at the cavity center is s_M, while the minimum spacing at the walls is $s_m = s_M/5$ in order to accurately resolve the boundary layers. A graphical representation of the spacing function is given in Figure 5.4b, while 5.4c shows an enlarged view of the node distribution for the bottom left corner in the case $N \approx 80,000$ nodes.

Results

The lid-driven cavity problem has been solved for Reynolds numbers $Re = 1000$ and $Re = 5000$ with three different node distributions with $N \approx 20,000$, $N \approx 40,000$ and $N \approx 80,000$ nodes. The employed polynomial order is $P = 2,3,4,5$ using $n = 20,25,30,40$ interpolation nodes, respectively, with $s \cdot \varepsilon = 0.4$. The employed amount of explicit hyperviscosity is $\bar{\delta}_3 = 5 \cdot 10^{-6}$ for $Re = 1000$ and $\bar{\delta}_3 = 1 \cdot 10^{-5}$ for $Re = 5000$.

The calculation at $Re = 1000$ is started from rest using a time step $\Delta t = 0.1$, reaching an asymptotic steady solution after an appropriate long time integration, i.e., over 500 time units. Steady solutions are also found for $Re = 5000$ using the same time step $\Delta t = 0.1$, starting from the steady solution at the $Re = 1000$ and integrating for additional 500 time units. These steady solutions are in perfect agreement with the findings of Fortin et al. [37] and Bruneau and Saad [10], which predicted a critical Reynolds number $Re_{cr} \approx 8000$. The extrema of the streamfunction for the primary vortex at the center of the cavity and for the secondary vortices at bottom corners are reported in Table 5.3, where the reference results of Bayona et al. [4] and AbdelMigid et al. [1] are also reported. The reference results of Bayona et al. are obtained using a high order RBF-FD meshless approach with $n = 90$ local support nodes, $N \approx 40000$ total nodes and a steady-state streamfunction formulation. The reference results of AbdelMigid et al. are obtained using a second order FV scheme with a 1301^2 grid. Good agreement is found in each case, with less than 3% deviations in each case. Nonetheless, the results for $Re = 5000$ exhibit a strange behaviour when N is increased, especially for $P = 4$ and $P = 5$, while the cases $P = 2$ and $P = 3$ show a very good agreement with the reference results on the fine distribution with $N \approx 80,000$ nodes.

Table 5.2: Streamfunction values for lid-driven cavity streamline plots.

-1×10^{-1}	-8×10^{-2}	-6×10^{-2}	-4×10^{-2}	-2×10^{-2}	-1×10^{-2}	-3×10^{-3}	-1×10^{-3}
-3×10^{-4}	-1×10^{-4}	-3×10^{-5}	-1×10^{-5}	-3×10^{-6}	-1×10^{-6}	-1×10^{-7}	-1×10^{-8}
-1×10^{-9}	-1×10^{-10}	-1×10^{-11}	-1×10^{-12}	-1×10^{-13}	-1×10^{-14}	0	1×10^{-14}
1×10^{-13}	1×10^{-12}	1×10^{-11}	1×10^{-10}	1×10^{-9}	1×10^{-8}	1×10^{-7}	1×10^{-6}
3×10^{-6}	1×10^{-5}	3×10^{-5}	1×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-3}	1×10^{-2}

Figure 5.5 shows the velocity profiles along the centerlines for $P = 3, n = 25$ and two node distributions $N \approx 20,000$ and $N \approx 80,000$. The comparison with the reference results of AbdelMigid et al. [1] shows a very good agreement to graphical accuracy.

The streamlines for $P = 3$, $n = 25$ and $N \approx 80,000$ nodes at steady state for the streamfunction values reported in Table 5.2 are depicted in Figure 5.6 for $Re = 1000$ and $Re = 5000$, with enlarged views of the corner regions. Such streamline figures agree, to graphical accuracy, to the ones reported in [4, 80].

A time dependent solution for $Re = 10000$ is then calculated using $N \approx 40,000$ nodes, $P = 3$, $n = 25$ interpolation nodes and $k_{SUB} = 3$ subiterations, expecting a flow bifurcation between a steady solution at $Re = 7500$ and a time-dependent, periodic solution for $Re = 10000$, as suggested by different authors [10, 37, 80]. The employed amount of explicit hyperviscosity is $\bar{\delta}_3 = 5 \cdot 10^{-5}$. The calculation is started from the steady solution at $Re = 5000$ using a time step $\Delta t = 0.05$, reaching an apparently periodic solution after about 1000 time units. Such periodic behaviour is deduced from the analysis of the time trace of the x-component of the velocity at the cavity center $x = y = 0.5$, reported in Figure 5.7. The frequency of the strongest harmonic component of this time trace is $f = 0.57$, which agrees with the reference value $f = 0.59$ reported in [80]. Good agreement with the same reference is also found for the mean value and the amplitude of this periodic signal. Streamfunction contours for the streamfunction values reported in Table 5.2 are reported in Figure 5.8 for four equally spaced time intervals along the main period $1/f$. From the analysis of Figure 5.8 it can be deduced that the time-dependent flow behaviour is originated from the periodic growing and detaching of secondary and tertiary vortices from the bottom and left cavity walls. Again, the streamline plots agree, to graphical accuracy, to the ones reported in [80] for the same time instants.

5.3.3. Differentially heated cavity

Geometry and boundary conditions

The differentially heated cavity problem, Figure 5.9a, is defined again by a square cavity with side length L where the horizontal walls are adiabatic while the vertical

Table 5.3: Comparison of streamfunction extrema for $Re = 1000$ and $Re = 5000$.

	Primary vortex, ψ (location)	Secondary vortex BR, ψ (location)	Secondary vortex BL, ψ (location)
		$Re = 1000$	
$N \approx 20,000$			
$P = 2$	-0.1195 (0.5294,0.5651)	1.760E-3 (0.8637,0.1118)	2.445E-4 (0.0834,0.0786)
$P = 3$	-0.1211 (0.5290,0.5647)	1.808E-3 (0.8630,0.1118)	2.526E-4 (0.0836,0.0789)
$P = 4$	-0.1226 (0.5288,0.5642)	1.861E-3 (0.8622,0.1118)	2.664E-4 (0.0840,0.0795)
$P = 5$	-0.1244 (0.5283,0.5638)	1.923E-3 (0.8616,0.1117)	2.827E-4 (0.0844,0.0800)
$N \approx 40,000$			
$P = 2$	-0.1190 (0.5313,0.5670)	1.737E-3 (0.8640,0.1119)	2.376E-4 (0.0834,0.0783)
$P = 3$	-0.1189 (0.5313,0.5671)	1.737E-3 (0.8637,0.1120)	2.349E-4 (0.0833,0.0782)
$P = 4$	-0.1192 (0.5313,0.5670)	1.748E-3 (0.8636,0.1119)	2.376E-4 (0.0834,0.0783)
$P = 5$	-0.1198 (0.5313,0.5670)	1.769E-3 (0.8630,0.1118)	2.431E-4 (0.0835,0.0786)
$N \approx 80,000$			
$P = 2$	-0.1191 (0.5329,0.5664)	1.736E-3 (0.8639,0.1116)	2.365E-4 (0.0835,0.0782)
$P = 3$	-0.1194 (0.5329,0.5664)	1.747E-3 (0.8637,0.1116)	2.378E-4 (0.0835,0.0782)
$P = 4$	-0.1199 (0.5328,0.5664)	1.761E-3 (0.8636,0.1116)	2.413E-4 (0.0836,0.0783)
$P = 5$	-0.1206 (0.5328,0.5663)	1.785E-3 (0.8634,0.1116)	2.474E-4 (0.0838,0.0786)
Reference [4]	-0.1189 (0.5308,0.5652)	1.730E-3 (0.8641,0.1118)	2.334E-4 (0.0832,0.0781)
Reference [1]	-0.1189 (0.5308,0.5657)	1.732E-3 (0.8636,0.1115)	2.334E-4 (0.0832,0.0782)
		$Re = 5000$	
$N \approx 20,000$			
$P = 2$	-0.1183 (0.5150,0.5359)	3.039E-3 (0.8062,0.0735)	1.386E-3 (0.0729,0.1372)
$P = 3$	-0.1226 (0.5146,0.5357)	3.146E-3 (0.8027,0.0726)	1.413E-3 (0.0726,0.1373)
$P = 4$	-0.1237 (0.5142,0.5355)	3.228E-3 (0.8021,0.0720)	1.456E-3 (0.0722,0.1380)
$P = 5$	-0.1262 (0.5121,0.5340)	3.320E-3 (0.8012,0.0716)	1.501E-3 (0.0718,0.1388)
$N \approx 40,000$			
$P = 2$	-0.1209 (0.5228,0.5275)	3.075E-3 (0.8053,0.0730)	1.393E-3 (0.0728,0.1374)
$P = 3$	-0.1175 (0.5230,0.5275)	3.008E-3 (0.8054,0.0731)	1.358E-3 (0.0731,0.1366)
$P = 4$	-0.1186 (0.5230,0.5275)	3.045E-3 (0.8050,0.0728)	1.372E-3 (0.0729,0.1369)
$P = 5$	-0.1180 (0.5230,0.5275)	3.059E-3 (0.8048,0.0728)	1.384E-3 (0.0728,0.1370)
$N \approx 80,000$			
$P = 2$	-0.1224 (0.5193,0.5291)	3.104E-3 (0.8052,0.0731)	1.393E-3 (0.0726,0.1376)
$P = 3$	-0.1225 (0.5193,0.5291)	3.099E-3 (0.8047,0.0729)	1.390E-3 (0.0728,0.1371)
$P = 4$	-0.1237 (0.5193,0.5291)	3.134E-3 (0.8038,0.0725)	1.404E-3 (0.0726,0.1374)
$P = 5$	-0.1256 (0.5193,0.5291)	3.189E-3 (0.8029,0.0721)	1.430E-3 (0.0723,0.1380)
Reference [4]	-0.1223 (0.5151,0.5352)	3.077E-3 (0.8046,0.0727)	1.379E-3 (0.0728,0.1371)
Reference [1]	-0.1221 (0.5155,0.5355)	3.078E-3 (0.8052,0.0729)	1.375E-3 (0.0729,0.1369)

BR = bottom right; BL = bottom left.

walls are isothermal. The temperature of the left wall (hot wall) is $T = T_H$ while the temperature of the right wall (cold wall) is $T_C < T_H$. The reference temperature for the buoyancy linearization is chosen to be the mean temperature $T_0 = (T_H + T_C)/2$ while the reference temperature scale is chosen to be $\Delta T = T_H - T_C$.

The boundary conditions in terms of nondimensional variables are the follow-

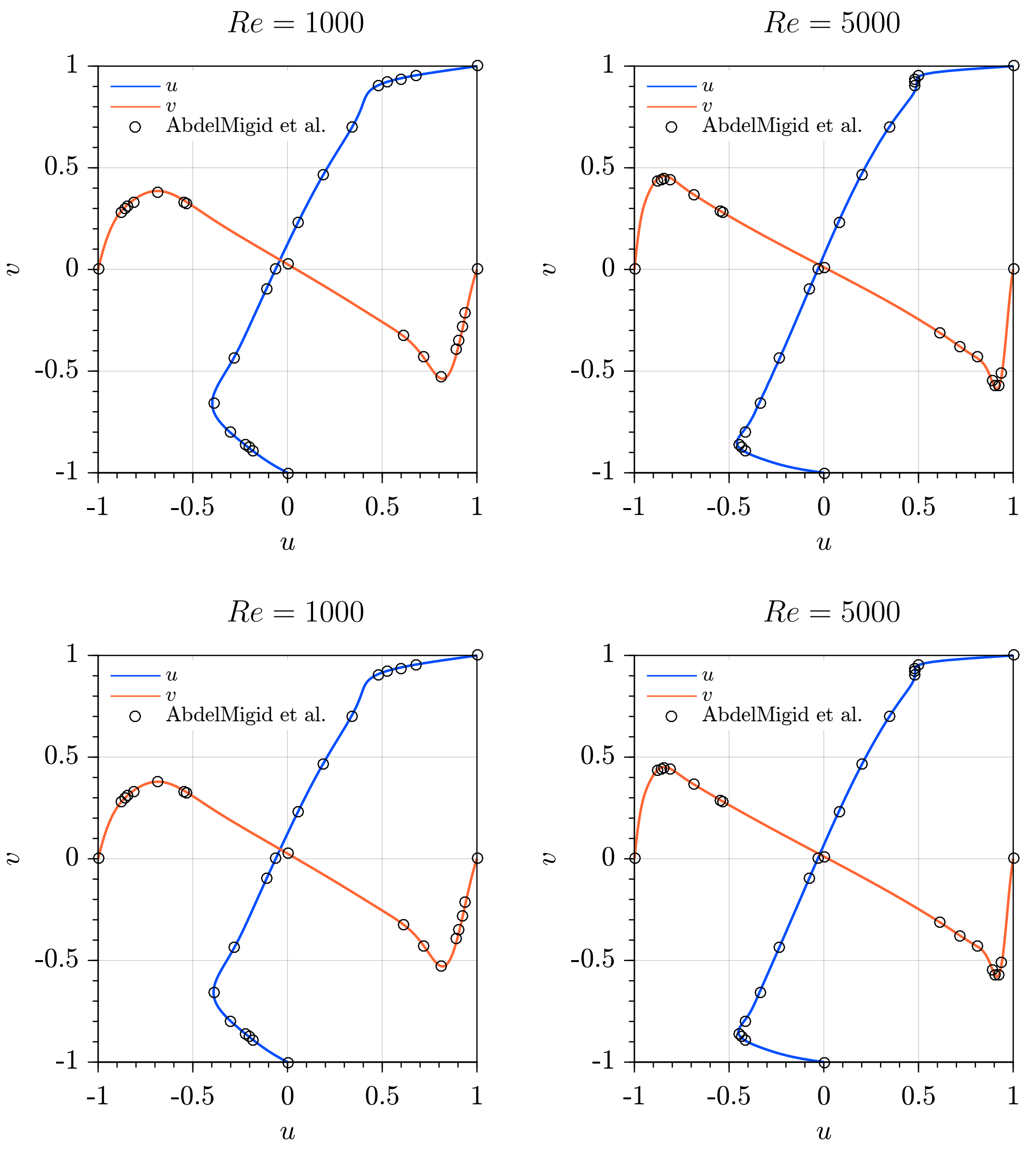


Figure 5.5: Normalized velocity profiles along centerlines: $N \approx 2 \times 10^4$ nodes (top), $N \approx 8 \times 10^4$ nodes (top), $P = 3$, $n = 25$.

ing:

$$\begin{cases} \mathbf{u} = \mathbf{0} & \text{at } x, y = 0, 1 \\ T = 1/2 & \text{at } x = 0 \\ T = -1/2 & \text{at } x = 1 \\ \dfrac{\partial T}{\partial y} = 0 & \text{at } y = 0, 1 \end{cases} \tag{5.22}$$

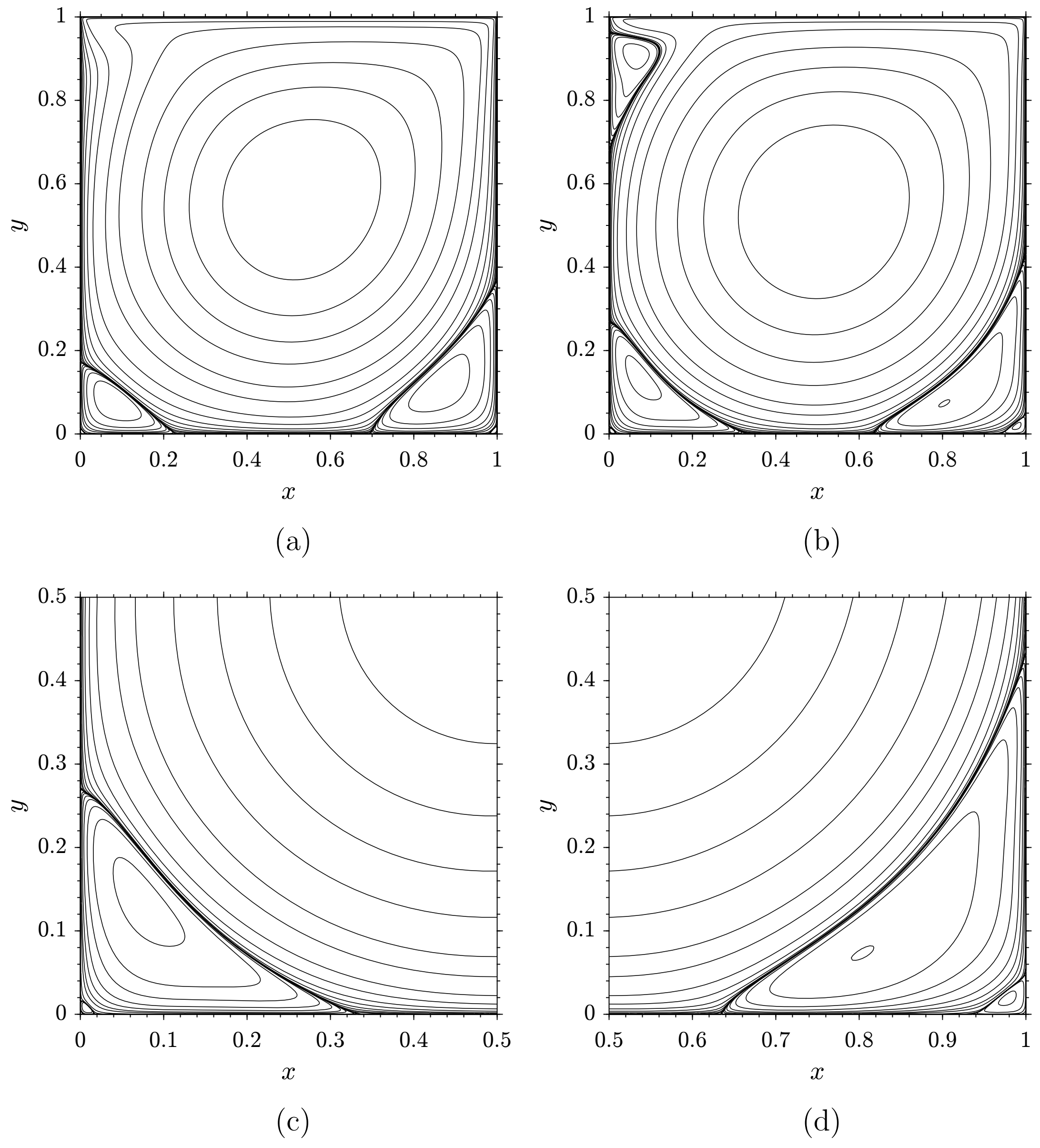


Figure 5.6: Streamfunction contours for $Re = 1000$ (a), $Re = 5000$ (b) with enlarged view of the bottom left (c) and bottom right (d) corners at $Re = 5000$. $N \approx 8 \times 10^4$ nodes, $P = 3$, $n = 25$.

Node distributions

The spacing function employed for the differentially heated cavity problem at $Ra = 10^6$, $Ra = 10^7$ and $Ra = 10^8$ is the following:

$$\frac{s(x,y)}{s_M} = \frac{1}{\tau} + \frac{\tau - 1}{4.2\tau}\Big[1 + \cos\big(\pi(2x-1)^8\big)\Big]\Big[1.1 + \cos\big(\pi(2y-1)^8\big)\Big] \tag{5.23}$$

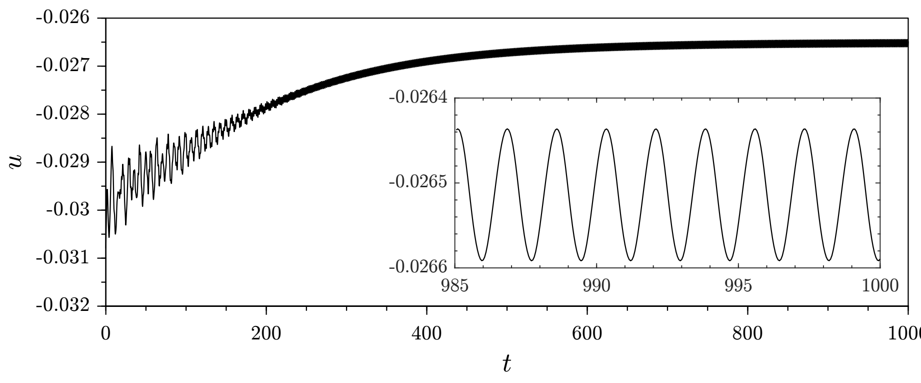


Figure 5.7: Time history of the x-component of the velocity at the cavity center for $Re = 10000$. $N \approx 4 \times 10^4$ nodes, $P = 3$, $n = 25$.

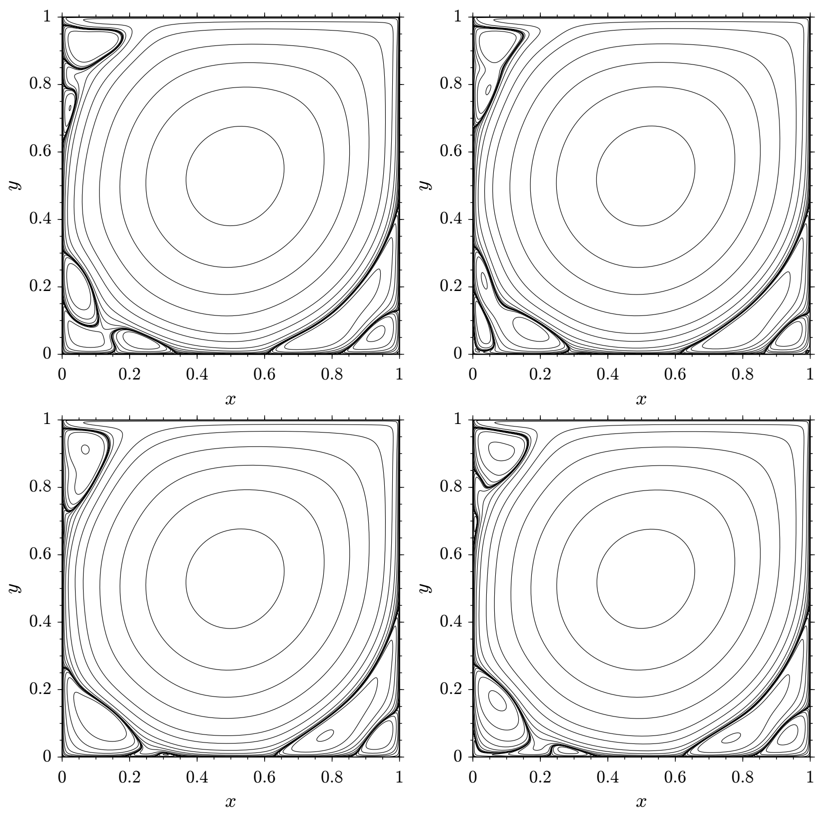
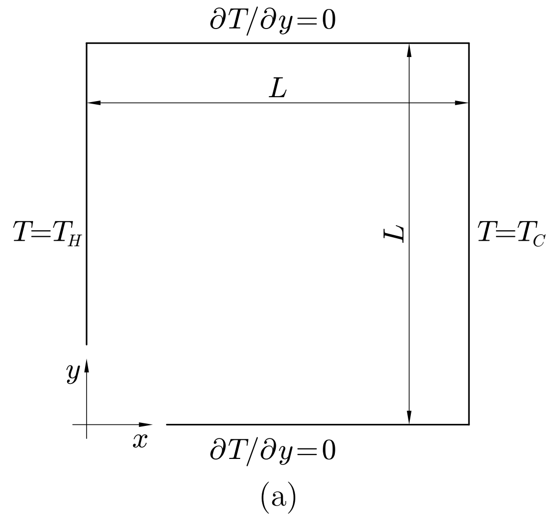


Figure 5.8: Streamfunction contours for $Re = 10000$ for four equally spaced times along the main period $1/f = 1.75$ (left to right, top to bottom). $N \approx 4 \times 10^4$ nodes, $P = 3$, $n = 25$.

for which the maximum spacing at the cavity center is s_M and $\tau = 40$. The minimum spacing at the vertical walls is $s_m = s_M/\tau = s_M/40$, while the spacing



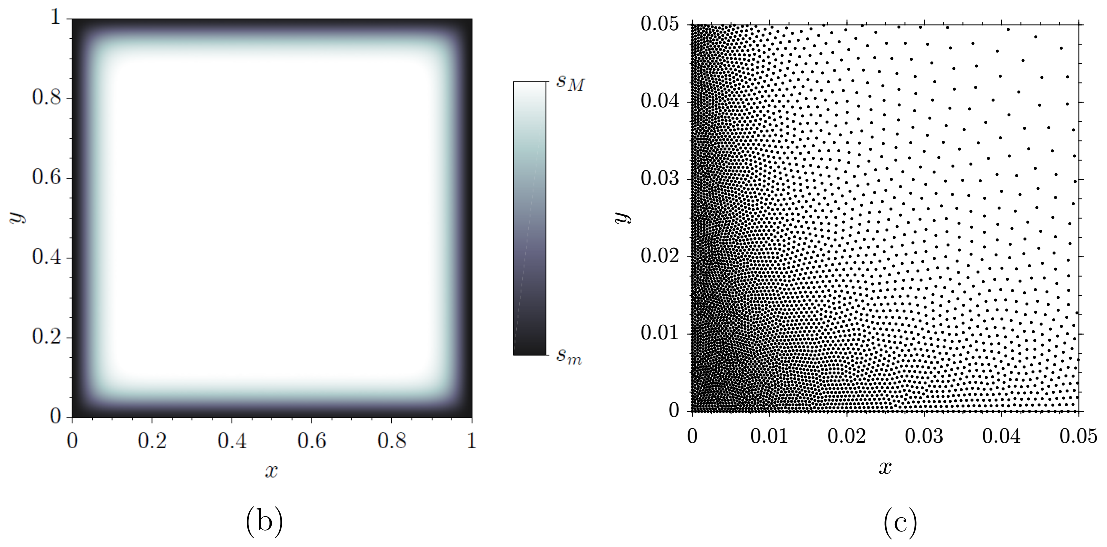


Figure 5.9: Geometry (a), spacing function (b) and enlarged view of the node distribution for the bottom left corner (c) with $N \approx 100,000$ nodes for the differentially heated cavity problem.

at the horizontal walls is larger than s_m. This choice is motivated by the necessity of an accurate resolution of the thin boundary layers occurring at the isothermal vertical walls, especially for high Ra numbers. A graphical representation of the spacing function is given in Figure 5.9b, while 5.9c shows an enlarged view of the node distribution for the bottom left corner.

The solutions of the differentially heated cavity at $Ra = 2 \times 10^8$ and $Ra = 4 \times 10^8$, which are expected to be time-dependent, are obtained on highly stretched, non-uniform cartesian node arrangements. This choice is due to the strong influence of the accurate resolution of the thin boundary layer on the time-dependent flow.

The employed cartesian coordinates of the nodes for both directions are:

$$\frac{x_i}{L} = \frac{i}{i_{MAX}} - \frac{1}{2\pi} \sin\left(\frac{2\pi i}{i_{MAX}}\right), \quad i = 0, \ldots, i_{MAX} \tag{5.24}$$

as suggested in [50]. The spacing defined by Eq. (5.24) is extremely small at the walls: $2L\pi^2/(3i^3_{MAX})$ orthogonally to the wall. Two cartesian arrangements with $i_{MAX} = 200$ ($N \approx 40,000$ nodes) and $i_{MAX} = 320$ ($N \approx 100,000$ nodes) have been employed, for which the distance of the first node from the wall is 8.2×10^{-7} and 2.0×10^{-7} when $L = 1$.

Results

The differentially heated cavity problem has been solved for Rayleigh numbers $Ra = 10^6$, $Ra = 10^7$ and $Ra = 10^8$ on an isotropic node distribution with $N \approx 100,000$ nodes, and for $Ra = 2 \times 10^8$ and $Ra = 4 \times 10^8$ on the stretched cartesian node distributions with 200×200 and 320×320 nodes. A transition between a steady solution at $Ra = 10^8$ and a time-dependent, periodic solution for $Ra = 2 \times 10^8$ is expected, while the solution at $Ra = 4 \times 10^8$ is expected to be weakly turbulent (chaotic), as suggested by different authors [50, 83].

In the cases $Ra = 10^6$, $Ra = 10^7$ and $Ra = 10^8$, a linear polynomial $P = 1$ and $n = 7$ interpolation nodes are employed with $s \cdot \varepsilon = 0.1$. The employed amount of explicit hyperviscosity is $\bar{\delta}_3 = 1 \cdot 10^{-4}$.

Starting from rest, a steady-state solution is found for $Ra = 10^6$ after an appropriate long time integration for 300 time units using a time step $\Delta t = 0.1$. Steady solutions are also found for the cases $Ra = 10^7$ and $Ra = 10^8$ using a time step $\Delta t = 0.05$, starting from the steady solution at the Ra value immediately below and integrating for more than 500 time units. These steady solutions are in perfect agreement with the findings of Janssen and Henkes [50] and Paolucci and Chenoweth [83], which predicted a critical Rayleigh number $\mathrm{Ra}_{cr} \approx 1.93 \times 10^8$. Characteristic values such as mean, maximum and minimum Nusselt number at the cold wall are reported in Table 5.4 for each of these cases, where the reference results of Contrino et al. [17] are also reported. Such reference results are obtained using a thermal lattice Boltzmann approach with fine meshes up to 2043^2. Very good agreement is found in each case, with slightly larger deviations from reference values only for $Ra = 10^8$. These deviations are probably due to the employed isotropic node distribution which is not refined enough to accurately solve the very thin boundary layers at the isothermal walls at such high Ra numbers.

Table 5.4: Comparison of characteristic values for the differentially heated cavity.

	$\overline{\text{Nu}}$	Nu_{max}	y	Nu_{min}	y
		$Ra = 10^6$			
Contrino et al. [17]	8.8252	17.5360	0.9608	0.9795	0.0006
Present results	8.8280	17.5611	0.9614	0.9793	0.0030
		$Ra = 10^7$			
Contrino et al. [17]	16.5231	39.3950	0.9820	1.3659	0.0006
Present results	16.5159	39.3889	0.9818	1.3755	0.0022
		$Ra = 10^8$			
Contrino et al. [17]	30.2251	87.2454	0.9917	1.9195	0.0010
Present results	30.0887	86.7845	0.9914	1.9694	0.0014

Figure 5.11a depicts the local Nusselt number Nu_y at steady-state along the cold wall for the Ra numbers previously considered. The comparison with reference values from Contrino et al. [17] shows excellent agreement along the whole wall. The contour plots for streamfunction and temperature at steady state are reported in Figure 5.10, showing a very good agreement, to graphical accuracy, to the ones reported in [17, 63].

In the cases $Ra = 2 \times 10^8$ and $Ra = 4 \times 10^8$, a linear polynomial $P = 1$ and $n = 5$ interpolation nodes (the classic five points stencil for cartesian arrangements) are employed with $s \cdot \varepsilon = 0.1$. The employed amount of explicit hyperviscosity is $\bar{\delta}_3 = 2.5 \cdot 10^{-4}$ and $k_{SUB} = 3$ subiterations are employed. Since the node distribution is not isotropic, the value of the spacing s employed for the scaling of the RBF and for the scaling of the hyperviscosity is the geometric mean of the spacing along the x and y axes: $s = \sqrt{s_x \cdot s_y}$.

The case $Ra = 2 \times 10^8$, which is a slightly larger Ra number than the critical value $Ra_{cr} \approx 1.93 \times 10^8$, is started from the computed steady solution at $Ra = 10^8$ which is interpolated onto the 200×200 cartesian node distribution. After an integration over 300 time units using a time step $\Delta t = 0.03$, an apparently periodic solution is found, albeit not completely developed. The periodic behaviour is deduced from the analysis of the time trace of the temperature for the node which is closest to the point $(0.1032, 0.8036)$, suggested in [83], and is reported in Figure 5.12a. In order to investigate the influence of the node distribution on the time dependent solution at this Ra number, the calculated solution is interpolated onto the fine 320^2 cartesian node distribution for a successive integration over 300 additional time units with the same time step $\Delta t = 0.03$. The analysis of time trace of the temperature at the same point $(0.1032, 0.8036)$ is reported in Figure

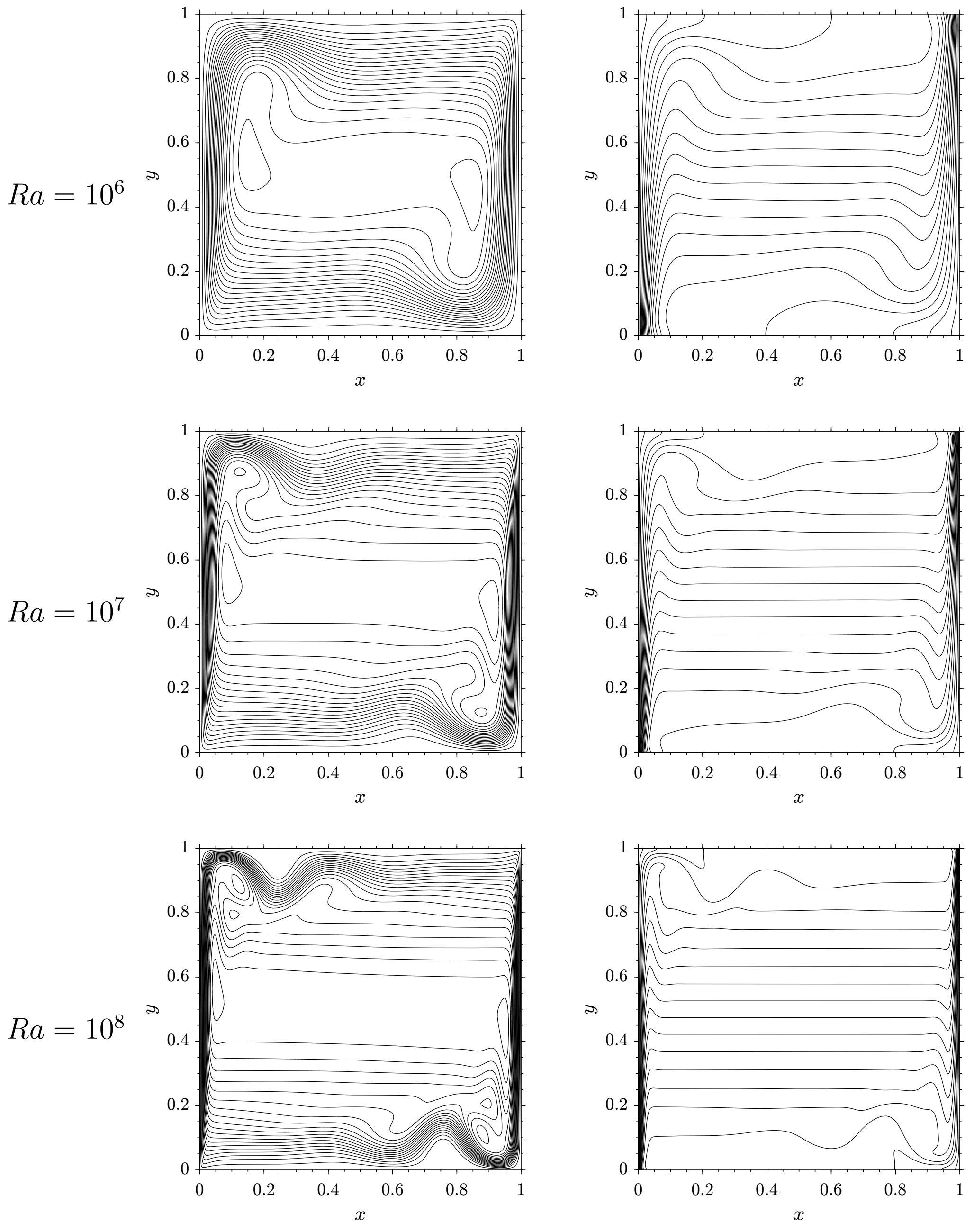


Figure 5.10: Streamfunction contours (left) and temperature contours (right) for $Ra = 10^6$, $Ra = 10^7$ and $Ra = 10^8$.

5.12b and confirms the periodic behaviour calculated with the coarse distribution. The frequency $f = 0.0532$ of the strongest component is in close agreement with

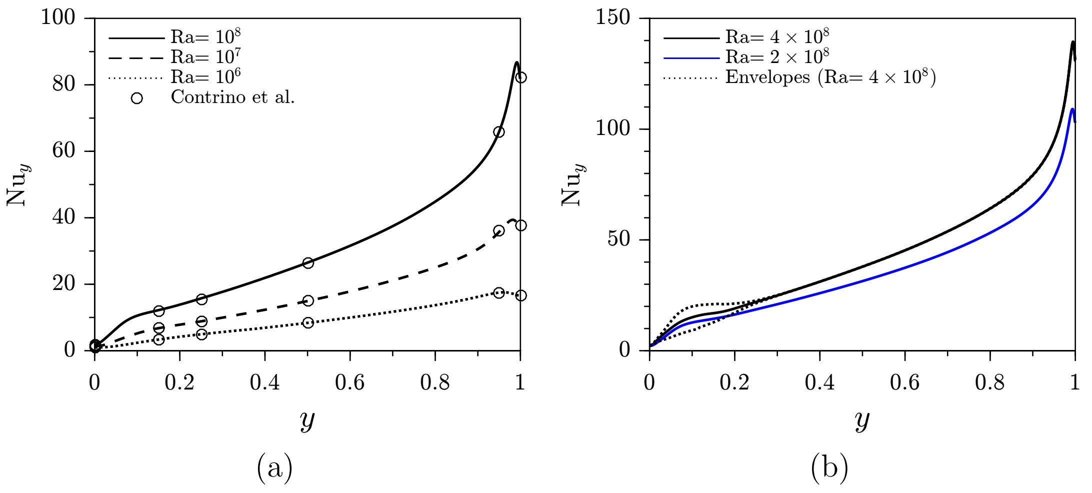


Figure 5.11: Local Nusselt number Nu_y along the cold wall for (a) steady-state solutions and (b) time dependent solutions (time averaged).

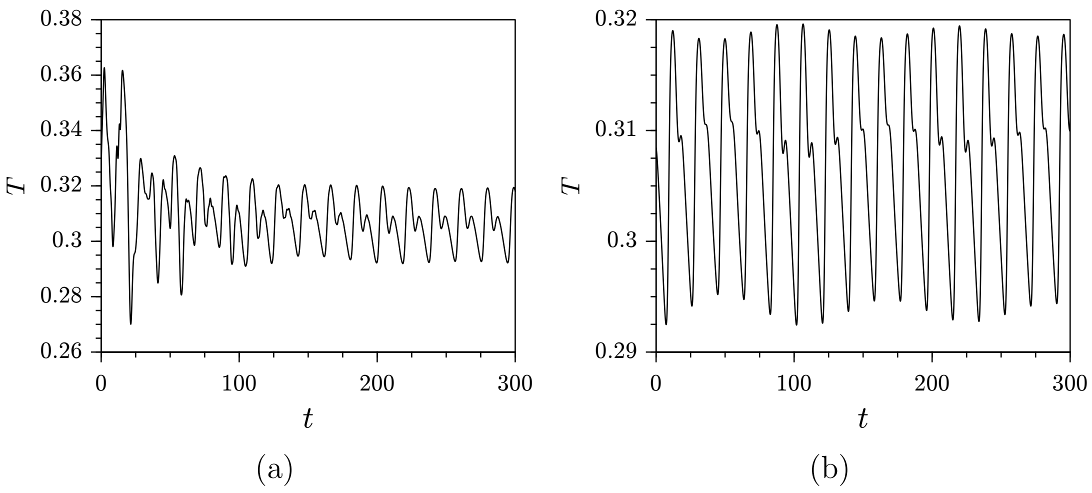


Figure 5.12: Time traces of the temperature at point (0.1032,0.8036) for $Ra = 2 \times 10^8$ using (a) 200×200 and (b) 320×320 highly stretched cartesian grids.

the value $f = 0.0527$ obtained by Janssen and Henkes [50] with a fourth order finite volume scheme and a 360^2 grid. The time trace also reveals a small harmonic component with frequency $2f = 0.1064$ and the presence of a low-frequency component whose frequency is estimated to be $f' \approx 0.008$, which is also in good agreement with the value $f' = 0.0078$ obtained by Janssen and Henkes. They also showed that this low-frequency component is damped and has a very slow decay, requiring long integration periods to be eliminated, e.g., 3000 time units.

The solution at $Ra = 4 \times 10^8$ is calculated on the fine distribution only, starting

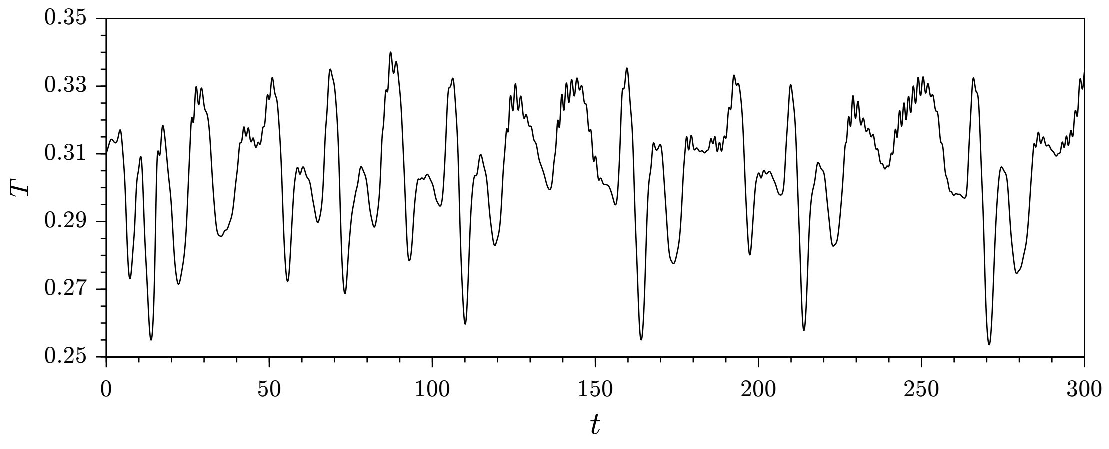


Figure 5.13: Time trace of the temperature at point (0.1032,0.8036) for $Ra = 4 \times 10^8$.

from the last available solution at $Ra = 2 \times 10^8$ and integrating for 300 time units with $\Delta t = 0.02$. The time trace of the temperature at the point $(0.1032, 0.8036)$ is reported in Figure 5.13, from which it can be observed that the solution can not be described by a limited number of harmonic components and the flow is described as chaotic or weakly turbulent [83]. This chaotic behaviour agrees with the findings of Paolucci and Chenoweth [83] which performed calculations for the same Ra number using a finite difference scheme on a 121^2 grid.

Figure 5.11b shows the time averaged local Nusselt number Nu_y along the cold wall for $Ra = 2 \times 10^8$ and $Ra = 4 \times 10^8$. The envelopes for the highest Ra value are also shown, highlighting the activity at the bottom of the cold wall.

5.3.4. Flow past a circular cylinder between parallel walls

Geometry and boundary conditions

The problem of the flow past a circular cylinder between parallel walls, Figure 5.14, is defined by a rectangular channel with height L and length $11L$ with a circular obstacle with diameter $d_{cyl} = L/5$ placed at half of the channel height and $3L = 15d_{cyl}$ downstream from the left inlet. The inlet velocity profile is parabolic with u_0 as mean value, while completely developed flow conditions are imposed at the outlet. The outlet is placed at $8L = 40d_{cyl}$ downstream from the cylinder in order to avoid any spurious influence of the outlet boundary conditions on the flow near the cylinder. The boundary conditions in terms of nondimensional variables

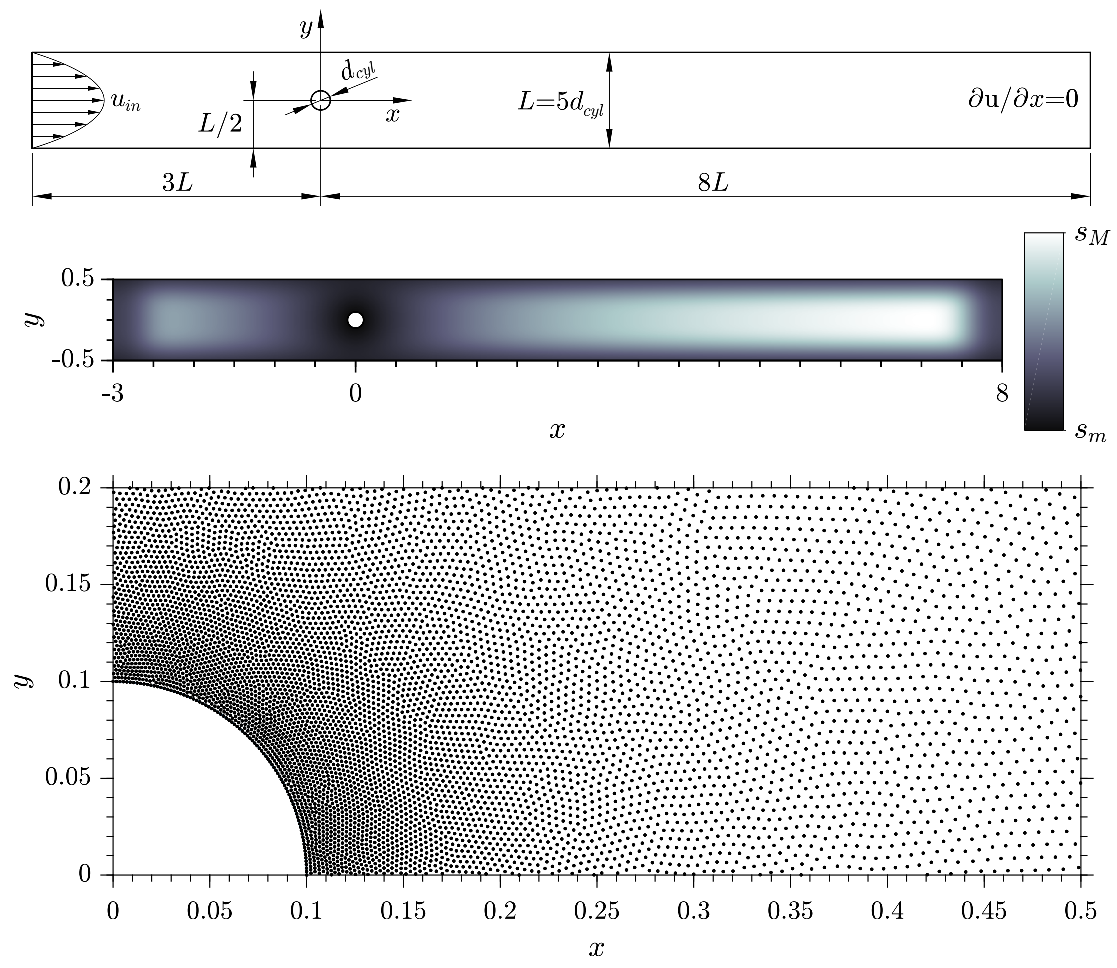


Figure 5.14: Geometry (top), spacing function (middle) and enlarged view of the node distribution around the cylinder with $N \approx 100,000$ nodes (bottom) for the flow past a circular cylinder between parallel walls.

are the following:

$$\begin{cases} \mathbf{u} = \mathbf{0} & \text{at } y = -1/2, 1/2; \text{ on the cylinder} \\ \mathbf{u} = \{3/2 - 6y^2, 0\}^T & \text{at } x = -3 \\ \dfrac{\partial \mathbf{u}}{\partial x} = \mathbf{0} & \text{at } x = 8 \end{cases} \tag{5.25}$$

Spacing function

The spacing function employed for the node generation is the following:

$$\frac{s(x,y)}{s_M} = \left[1 + k_W\left(\frac{2y}{H}\right)^{2e_W} + k_C\frac{d_{cyl}}{2r} + k_W e^{-8(x+3)^2} + k_W e^{-8(x-8)^2}\right]^{-1} \tag{5.26}$$

where $k_W = 3$, $e_W = 1.75$, $k_C = 24$, $r = \sqrt{x^2+y^2}$ is the distance from the center of the cylinder and s_M is the maximum spacing. The ratio between the spacing at the channel walls and the maximum spacing s_M, encountered towards the outlet, is approximately $1/(k_W+1) = 1/4$, while the ratio between the spacing at the cylinder wall and the maximum spacing is approximately $1/(k_C+1) = 1/25$. Again, a very small nodal spacing is employed near the cylinder in order to accurately resolve the boundary layers, while the inlet and the outlet are also refined in order to prevent the arising of instabilities. Instabilities at the inlet are due to the downstream orientation of the stencils near the boundary, while possible instabilities at the outlet may arise because of the imposed condition of completely developed flow. A graphical representation of the spacing function is given in Figure 5.14 together with an enlarged view of the node distribution around the cylinder for $N \approx 100,000$ nodes.

Results

The case of the flow past a circular cylinder between parallel walls has been solved for Reynolds numbers $Re = 200$, $Re = 300$, $Re = 500$ and $Re = 1000$, expecting a time-dependent flow for the cases $Re \geq 300$, as suggested by different authors [13, 111]. A linear polynomial augmentation $P = 1$ and $n = 7$ interpolation nodes are employed, while the employed amount of explicit viscosity (laplacian) is $\bar{\delta}_1 = 4$ for each Re number. $k_{SUB} = 2$ subiterations are employed. The results for this case are obtained with the coarse distribution strategy for the pressure stabilization reported in Subsection 5.2.3. $N \approx 100,000$ nodes are employed.

The calculation at $Re = 200$ is started from rest using a time step $\Delta t = 0.025$, reaching an asymptotic steady solution after an appropriate time integration for over 100 time units. Periodic solutions are found in the remaining cases, using the time steps reported in Table 5.5, starting from the solution at the Re value immediately below and integrating for over 200 time units for each case. These periodic solutions are in perfect agreement with the findings of Zovatto and Pedrizzetti [111] and Chen et al. [13], which predicted a critical Reynolds number $Re_{cr} \approx 231$. The periodic flow is due to the onset of an unsteady periodic shedding regime in the cylinder wake. The values of the nondimensional shedding period $\bar{t}$ and drag coefficient C_D are reported in Table 5.6, where the reference results of Zovatto and

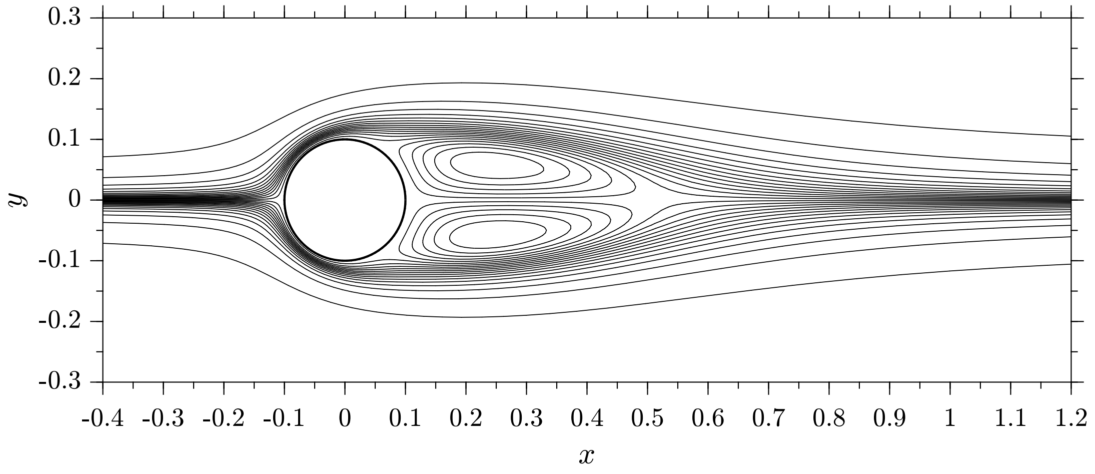


Figure 5.15: Streamlines for the steady flow at $Re = 200$.

Table 5.5: Time steps Δt at different Re numbers for the flow past a cylinder.

Re	200	300	500	1000
Δt	0.025	0.020	0.015	0.015

Pedrizzetti [111] are also reported. Good agreement for both $\bar{t}$ and C_D is found for low Re numbers, while larger deviations of C_D are found for higher Re and are due to the use of the simple laplacian smoothing $k = 1$ which introduces large numerical diffusion errors.

A streamline plot for the steady solution at $Re = 200$ is reported in Figure 5.15, which shows the typical recirculating zones with perfect symmetry behind the cylinder.

5.4. 3D CASE

For all the 2D cases, the $k = 3$ explicit hyperviscosity smoothing (∇^6) with $n_H = 40$ interpolation nodes is employed, except for the case of the flow past a circular cylinder between parallel walls where a laplacian smoothing with $n_H = n = 7$ is employed.

Table 5.6: Comparison of nondimensional period $\bar{t}$ and time-averaged drag coefficient C_D.

	Re	200	300	500	1000
Present	$\bar{t}$	–	0.80	0.72	0.65
results	C_D	3.41	2.99	2.77	2.74
Zovatto and	$\bar{t}$	–	0.81	0.73	0.67
Pedrizzetti [111]	C_D	3.40	2.94	2.68	2.62

5.4.1. Lid-driven cavity

Geometry and boundary conditions

The domain is a cube with side length L. No-slip conditions are imposed at all faces and the face $x = 0$ moves in the $+z$ direction with velocity u_0.

Spacing function

The spacing function employed for the node generation is:

$$\frac{s(x,y,z)}{s_M} = \frac{1}{1 + d(x,y,z)} \tag{5.27}$$

where $d(x,y,z)$ is:

$$d(x,y,z) = 7 - \sum_{i=1}^{3} L_i \frac{\operatorname{atan}(2l_i x_i)}{\operatorname{atan}(l_i)} + \frac{\operatorname{atan}\big(2g(1-x_i)\big)}{\operatorname{atan}(g)} \tag{5.28}$$

where $x_1 = x$, $x_2 = y$, $x_3 = z$, $g = 5$, $l_i = 5$ except for $l_1 = 10$, $L_i = 1$ except for $L_1 = 2$.

The maximum spacing function s_M is encountered at the cavity center, while the spacing at the walls is $s \approx s_M/2$ except for the moving face $x = 0$ where the spacing is $s \approx s_M/3$. Three node distributions with $N = 40,000$, $N = 80,000$ and $N = 140,000$ nodes are depicted in Figure 5.16 where the yellow face is the moving face.

Results

The 3D lid-driven cavity problem has been solved for Reynolds numbers $Re = 100$, $Re = 400$ and $Re = 1000$ using three node distributions with $N \approx 40,000$, $N \approx 80,000$ and $N \approx 140,000$ nodes. The finer distribution is employed for the case $Re = 1000$ only. The employed polynomial order is $P = 3$ and $n = 40$ interpolation nodes are used with $s \cdot \varepsilon = 0.4$. The $k = 3$ hyperviscosity is employed

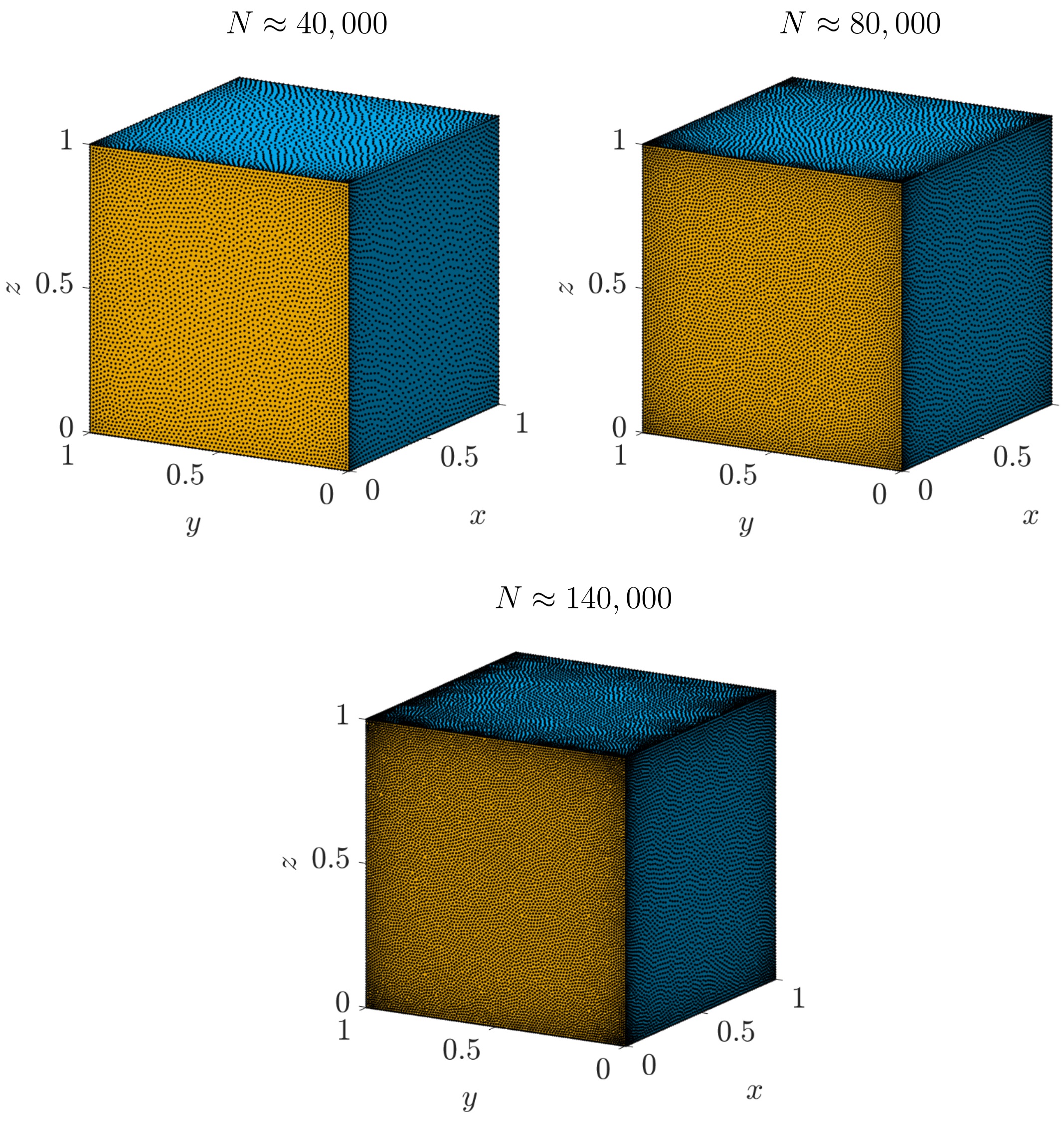


Figure 5.16: Three node distributions employed for the 3D lid-driven cavity problem. The yellow face moves in the $+z$ direction with velocity $w = 1$.

using $n_H = 55$ interpolation nodes and the amount of explicit hyperviscosity is $\bar{\delta}_3 = 2 \cdot 10^{-5}$ for $Re = 100$ and $Re = 400$, while $\bar{\delta}_3 = 1 \cdot 10^{-4}$ for $Re = 1000$.

The calculation at $Re = 100$ is started from rest using a time step $\Delta t = 0.1$, reaching an asymptotic steady solution after an appropriate long time integration, i.e., over 300 time units. Steady solutions are also found for $Re = 400$ using the same time step $\Delta t = 0.1$, starting from the steady solution at the $Re = 100$ and integrating for additional 300 time units. The velocity profiles along the centerlines

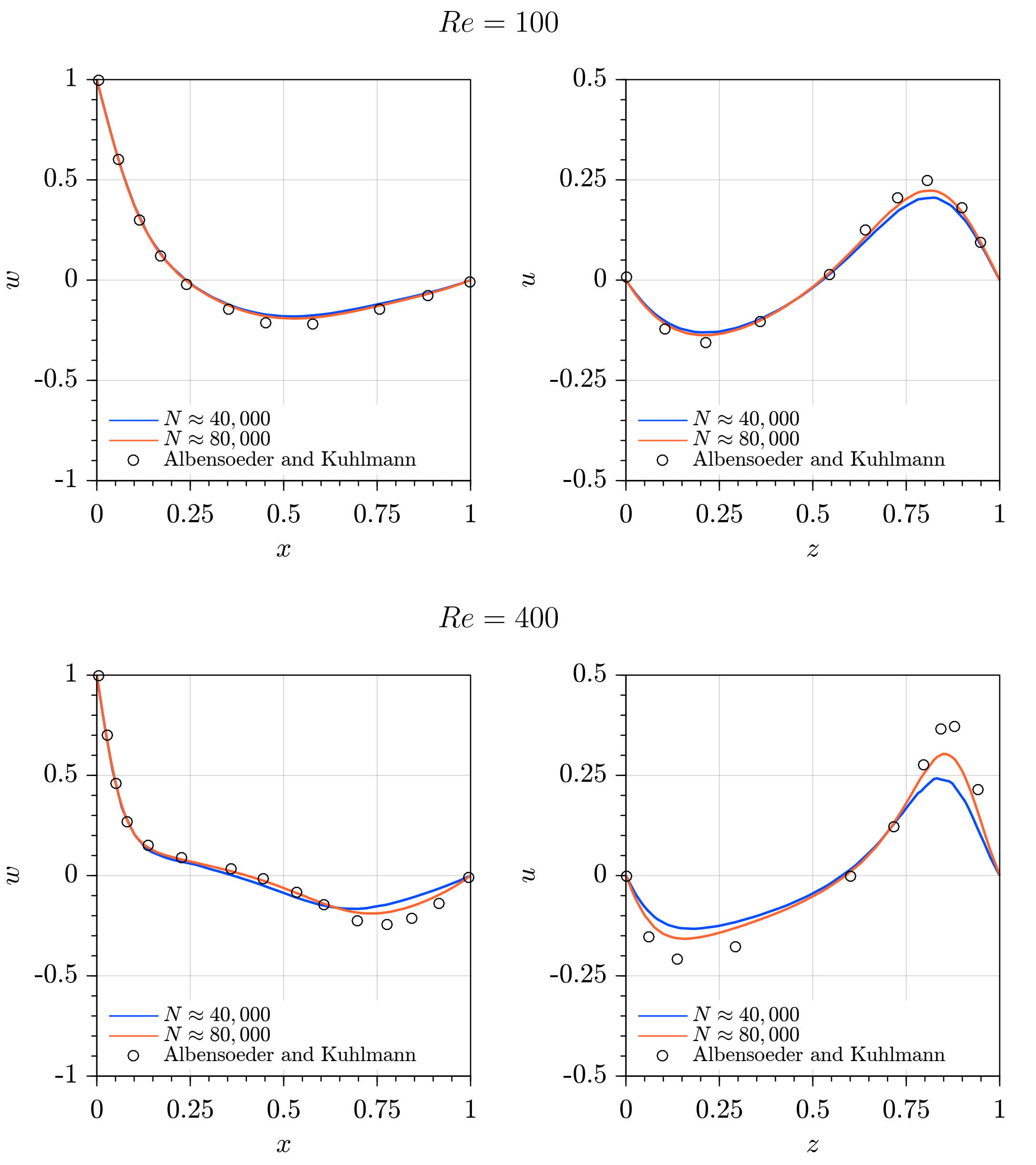


Figure 5.17: Normalized velocity profiles $w(x, 0.5, 0.5)$ (left) and $u(0.5, 0.5, z)$ (right) along centerlines. $P = 3, n = 40$.

at steady-state are depicted in Figure 5.17 where the results of Albensoeder and Kuhlmann [3] are also reported as reference. Such reference results are obtained using an accurate spectral method with $32 \times 24 \times 32$ polynomial modes. A good agreement with reference results is obtained in the $Re = 100$ case, while for $Re = 400$ the deviations are larger. Nonetheless, there is a significant improvement when

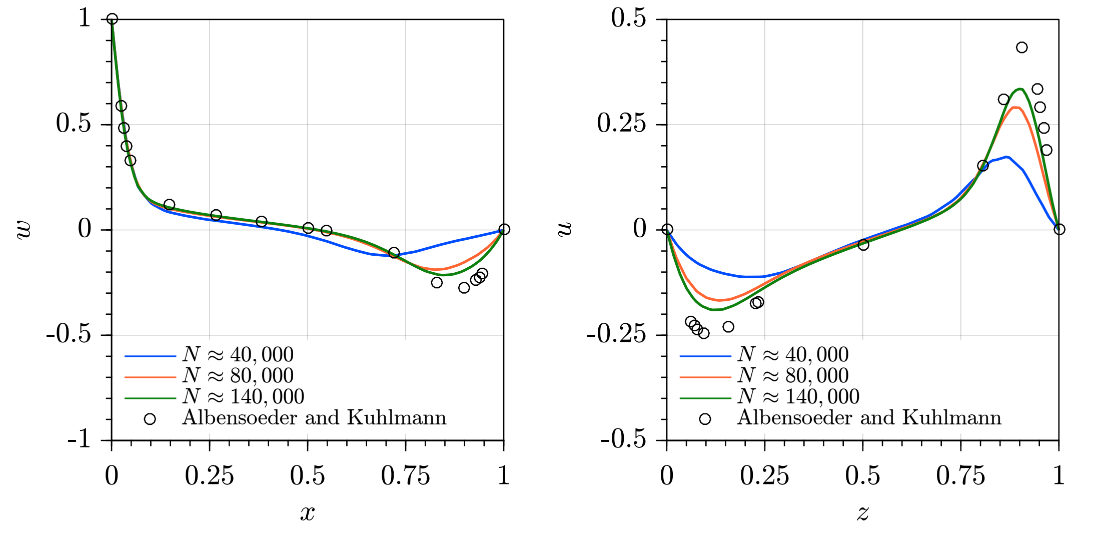


Figure 5.18: Normalized velocity profiles $w(x, 0.5, 0.5)$ (left) and $u(0.5, 0.5, z)$ (right) along centerlines, $Re = 1000$. $P = 3, n = 40$.

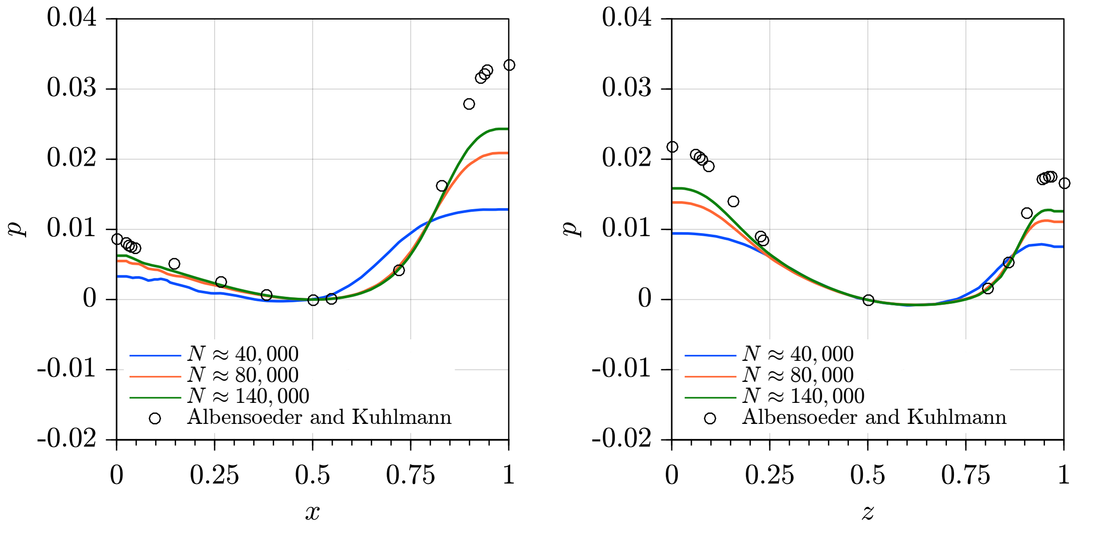


Figure 5.19: Pressure profiles $p(x, 0.5, 0.5)$ (left) and $p(0.5, 0.5, z)$ (right) along centerlines, $Re = 1000$. $P = 3, n = 40$.

increasing the number of nodes from $N \approx 40,000$ to $N \approx 80,000$.

The case $Re = 1000$ is calculated using the three distributions with $N \approx 40,000$, $N \approx 80,000$ and $N \approx 140,000$ nodes. Steady state solutions have been found after an appropriate time integration over 400 time units in each case, starting from the steady solution at $Re = 400$ using $N \approx 40,000$ and $N \approx 80,000$

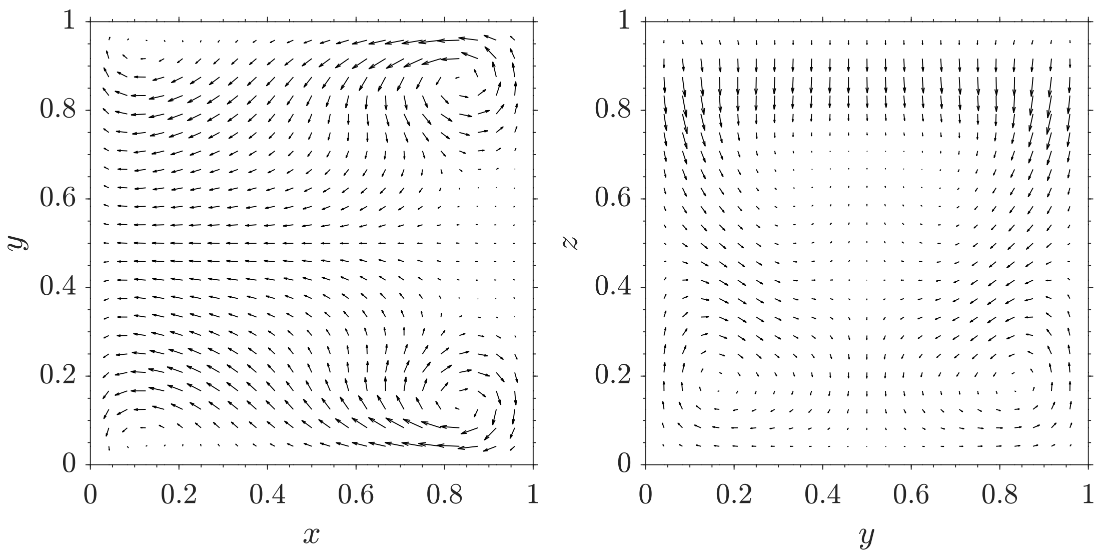


Figure 5.20: Velocity vector plot for (u, v) (left) and (v, w) (right) on midplanes $z = 0.5$ and $x = 0.5$, respectively. $Re = 1000, P = 3, n = 40$.

nodes, while for $N \approx 140,000$ nodes the calculation is started from rest. A time step $\Delta t = 0.075$ is employed in each case. The pressure (shifted to 0 at the cavity center) and the velocity profiles along the centerlines at steady-state are depicted in Figure 5.19 and Figure 5.18, respectively, where the results of Albensoeder and Kuhlmann [3] are again reported as reference. In the case $Re = 1000$ the reference results are obtained using $96 \times 64 \times 96$ polynomial modes. The results for $N \approx 40,000$ nodes show poor agreement with the reference values, while the increase in the number of nodes from $N \approx 40,000$ to $N \approx 80,000$ to $N \approx 140,000$ leads again to a significant improvement of the results. Significant deviations from the reference values are still visible even using the fine distribution, especially for the pressure near the walls, suggesting that $N \gg 140,000$ nodes are probably required for the accurate solution of this problem using this RBF-FD scheme ($P = 3, n = 40$).

Figure 5.20 shows the velocity vector plot on midplanes $z = 0.5$ and $x = 0.5$ for the case $Re = 1000$ using $N \approx 140,000$ nodes. These vector plots reveal the 3D nature of the lid-driven cavity flow at $Re = 1000$ with the presence of secondary flows. These observations are in perfect accordance to the findings of Albensoeder and Kuhlmann [3].

5.5. Conclusions

In this chapter the RBF-FD approach previously presented has been successfully employed for the discretization of the incompressible Navier-Stokes equations with possible heat transfer using primitive variables and a projection technique for the decoupling of mass and momentum equations. A second order backward Euler scheme for the time integration has been employed, resulting in an accurate and efficient procedure for the numerical simulation of time-dependent fluid-flow problems on 2D and 3D domains. The hyperviscosity technique for the stabilization of transport equations is extended to the set of Navier-Stokes equations where there is no transport equation for the pressure. The stabilization of such velocity-pressure coupling is obtained through an explicit hyperviscosity stabilization which is applied to both velocity and pressure at each time step, allowing the use of an "equal-order" RBF-FD discretization scheme for each variable. Such stabilization technique has proven to be particularly effective when using high order RBF expansions, i.e., large stencils and polynomial order $P > 2$, and for moderately high Reynolds or Rayleigh numbers occurring in engineering relevant problems. Different 2D/3D benchmark test cases have been successfully carried out in order to highlight the properties of the RBF-FD method when it is applied to the full set of Navier-Stokes equations with possible heat transfer using primitive variables; the comparison with reference results showed very good agreement in each case. In the case of the laminar flow inside a circular domain, where the analytical solution is available, it is shown that the computational efficiency of the RBF-FD method is increased when using high order RBF expansions, e.g., P =4-5 and n =30-40 nodes; this is in perfect accordance with similar results obtained for the simple test cases of Chapter 3. Even though simple geometries have been considered, the presented approach can be applied to arbitrary 2D/3D domains since the RBF-FD discretization does not rely upon any polygonization and no connectivity information is required. Alongside the favourable numerical properties highlighted in this chapter, these features confirm that the presented RBF-FD meshless approach is an excellent candidate for the efficient and accurate numerical simulation of 2D and 3D CFD problems over complex-shaped domains with engineering relevance.

Chapter 6

Summary and conclusions

6.1. Summary of the presented work

The objective of this dissertation was to investigate the numerical properties of the RBF-FD meshless approach when it is employed for the solution of CFD problems, with particular reference to fluid-flow problems defined over complex-shaped domains. This objective has been accomplished by developing a MATLAB code which is composed by several elements characterizing the meshless approach to a CFD problem: node generation, RBF-FD discretization of the Navier-Stokes equations with possible heat transfer and solution of the systems of equations. The work presented in this thesis has focused on the analysis and development of these characterizing elements which are essential in developing an innovative, robust, accurate and flexible numerical method.

The node generation is the first problem that has been tackled and a significant portion of this work is dedicated to this element since it is the foundation of every meshless approach. Different algorithms have been successfully proposed for the generation of high quality node distributions on 2D and 3D complex-shaped domains. Such node distributions then proved to be very suitable for the use with RBF-FD discretizations. The proposed approaches are characterized by an initial node positioning phase followed by a refinement phase based on the mutual repulsion of neighbouring nodes. The developed node generation algorithms are extremely efficient, i.e., $0.5 - 1.0 \cdot 10^5$ nodes/second, and are based on very simple principles: this is an important insight since the possibility to easily deal with complex geometries represents the main theoretical advantage of meshless methods over mesh-based methods. This advantage is particularly valuable in the 3D case where no boundary discretization is needed thanks to a boundary projection technique.

The RBF interpolation, which is the key element of the RBF-FD method, is then thoroughly studied by exploring all the variables which influence the construction of an accurate and robust interpolant over scattered nodes: number of interpolation nodes, polynomial degree, shape factor, type of rescaling, influence of boundary nodes. The multiquadric RBF has been chosen and extensive numerical tests are conducted for 2D and 3D cases. On the base of these analysis, a RBF-FD code has been developed for the local approximation of the partial derivatives of an unknown function which is defined only at scattered nodes distributed over the 2D/3D domain. The coupling of a node generation algorithm to a RBF-FD scheme leads to a truly meshless approach which can be used to discretize a given PDE on a domain with possible arbitrary shape. Such meshless approach has been employed to perform several test cases for different 2D/3D model problems which have fundamental importance in CFD applications. These model problems include a Poisson equation, which is representative of steady-state diffusive phenomena, and an advection equation, which is representative of convection-dominated transport phenomena.

The solution phase then follows the RBF-FD discretization, and its role in the simulation chain is just as important as the previous aspects, since an efficient solver is essential for any numerical approach. For this purpose, novel multicloud techniques based on the multigrid principles have been proposed for the acceleration of the convergence in the solution of the system of equations arising from RBF-FD discretizations, in the case of a 2D Poisson equation. Such multicloud techniques have proven to bring substantial improvements in terms of reduction of computational effort over the traditional solvers employed with the RBF-FD discretizations, e.g., BiCGSTAB with incomplete LU (ILU) preconditioning.

Finally, the RBF-FD approach is employed to solve actual 2D/3D fluid-flow problems in the case of the time-dependent, incompressible Navier-Stokes equations with possible heat transfer using a projection approach. These problems include a laminar flow inside a circular domain, a lid-driven cavity, a differentially heated cavity and a flow past a circular cylinder between parallel walls in the 2D, while a 3D lid driven cavity is also considered. Very good results have been obtained in each case when comparing the computed results to the reference ones. Important problems are addressed, e.g., the development of stable discretizations when dealing with the pressure-velocity coupling using primitive variables with moderately high Reynolds or Rayleigh numbers. This important result has been obtained employing an explicit stabilization through the hyperviscosity technique for all the involved fields, i.e., velocities, pressure and temperature, leading to an efficient and stable RBF-FD approach which can be used for the accurate solution of time-dependent fluid-flow problems over arbitrarily shaped domains in 2D and in 3D.

6.2. Conclusions and further developments

The main conclusions resulting from the conducted work can be summarized as follows:

- 2D/3D node generation for RBF-FD meshless approaches can be more effective and easier than the meshing process required by mesh-based methods: the proposed algorithms are very simple and efficient and are able to automatically generate high-quality node distributions over arbitrary geometries. Nonetheless, since a limited number of cases have been faced only, especially in 3D, stronger statements can not be made;

- a proper choice of the RBF-FD parameters, i.e., polynomial order P, number of interpolation nodes n and shape factor ε, results in very accurate and efficient discretizations and is therefore essential for the practical success of RBF-FD methods;

- if proper RBF-FD parameters are chosen, the sensitivity of the discretization upon the nodal arrangement is small. This is another significant advantage over mesh-based methods;

- when high order RBF-FD discretizations are employed, i.e., $P > 3$, the accuracy and the stability of the discretization are affected by the nonsymmetric nodal arrangement in the neighbourhood of the boundary (no ghost nodes are employed in this work);

- for problems with no time dependence, e.g., Poisson equation, the employment of small shape factors $\varepsilon \to 0$ increases the accuracy of the discretization in the limit of the appearance of ill-conditioned interpolants. This is in perfect accordance with the theoretical predictions stating that the polynomial interpolant with minimal degree is recovered in the limit $\varepsilon \to 0$;

- for time-dependent problems, e.g., advection-diffusion equation, the stability of the discretization is heavily compromised by the use of small shape factors, probably because of boundary effects/boundary conditions due to the unsymmetrical boundary stencils;

- hyperviscosity is a powerful smoother which can be employed for an effective stabilization of RBF-FD discretizations, especially with high order RBF expansions. The introduced artificial numerical diffusion can be very small or even negligible if proper hyperviscosity parameters are employed, i.e., hyperviscosity exponent and amount;

- the explicit hyperviscosity for the pressure allows the use of "equal-order" discretization schemes for both pressure and velocity. Theoretically, the pressure field could even be discretized with higher order RBF schemes than velocities;

- the RBF-FD approach can be employed for the accurate, robust and flexible simulation of fluid-flow problems over 2D and 3D complex-shaped domains. The order of accuracy of the discretization can be increased by simply considering large stencils with more nodes and a polynomial with high degree. For the considered problems, a polynomial degree $P = 4$ with $n \approx 30$ local nodes allowed the best results in terms of computational efficiency.

Considering both the theoretical and the heuristic motivations acquired during the present work, the ranges and the recommended values for the main parameters defining the entire RBF-FD meshless approach presented here, including the node generation phase, are reported in Table 6.1.

Further developments of the RBF-FD method include a very large variety of aspects. If we limit to the basic aspects, the following elements are believed to be important:

- robust treatment and enforcement of boundary conditions at the boundary stencils (without ghost nodes) for steady-state problems, especially for high orders P. Such aspect should also be developed for time-dependent problems by considering stability issues;

- automatic and robust choice of the stabilization parameter for the explicit hyperviscosity;

- development of specific and efficient solvers for different equation types and regimes, i.e., convection dominated vs. diffusion dominated, especially for high orders P;

- handling of geometrical anisotropy for the efficient treatment of boundary layers at the walls. This aspect includes the development of anisotropic node generation techniques and, consequently, the employment of anisotropic RBF expansions;

- employment of turbulence models for the simulation of practical fluid-flow problems of industrial relevance.

Table 6.1: Ranges for the main RBF-FD parameters ($\underline{\text{recommended values}}$).

Node-repel refinement

k_R: number of refinement iterations
n_R: number of neighbouring nodes
α, β: repel parameters

k_R	β	n_R	$1/\alpha$
20-100	$\underline{0.20}$ (2D)	$\underline{12}$ (2D)	maximize until insatabilities appear
($\underline{50}$)	$\underline{0.15}$ (3D)	$\underline{36}$ (3D)	($\underline{0.1}$)

RBF-FD

P: polynomial degree
n: number of local nodes (stencil size)
$s \cdot \varepsilon$: spacing function · shape factor
D: number of dimensions

P	n	$s \cdot \varepsilon = constant$
> 1	$\geq 2 \cdot \binom{P+D}{P}$	as small as possible (steady-state problems)
($\underline{3\text{-}4}$)		$\underline{0.4}$ (time-dependent problems)

Explicit hyperviscosity stabilization $(p, \mathbf{u}, T)$

k: hyperviscosity exponent (∇^{2k})
$\bar{\delta}_k$: amount of hyperviscosity
P_H: polynomial degree
n_H: number of local nodes (stencil size)
$s \cdot \varepsilon_H$: spacing function · shape factor

k	$\bar{\delta}_k > 0$	P_H	n_H	$s \cdot \varepsilon_H = constant$
$\geq P/2$	minimum value	≥ 2	$> \binom{P_H+D}{P_H}$	$\underline{s \cdot \varepsilon}$
($\underline{3\text{-}4}$)	guaranteeing	$\leq 2k$	$\geq n$	
	stability	($\underline{4\text{-}6}$)		

6.3. Publications

1 Conference Paper (2017), R. Zamolo, E. Nobile, Numerical Analysis of Heat Conduction Problems on Irregular Domains by Means of a Collocation Meshless Method, *Journal of Physics: Conference Series*, 796 (1), 012006.

2 Conference Paper (2017), R. Zamolo, E. Nobile, Numerical Solution of Heat Conduction Problems by Means of a Meshless Method with Proper Point Distributions, PROCEEDINGS OF CHT-17 ICHMT 7th International Symposium on Advances in Computational Heat Transfer, pp. 727-742.

3 Conference Paper (2017), R. Zamolo, E. Nobile, Numerical Analysis of Heat Conduction Problems on 3D General-shaped Domains by Means of a RBF Collocation Meshless Method, *Journal of Physics: Conference Series*, 923 (1), 012034.

4 (2017), R. Zamolo, L. Parussini, V. Pediroda, Distributed Lagrange Multiplier Functions for Fictitious Domain Method with Spectral/hp Element Discretization, *Journal of Scientific Computing*, 74(2), pp. 805-825.

5 (2018), R. Zamolo, 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), pp. 4305-4321.

6 Conference Paper (2018), R. Zamolo, E. Nobile, Numerical Analysis of Advection-diffusion Problems on 2D General-shaped Domains by Means of a RBF Collocation Meshless Method, *Conference Paper*, 36th UIT Heat Transfer Conference.

7 Under review (2018), R. Zamolo, E. Nobile, B. Šarler, Novel Multicloud Techniques for Convergence Acceleration in the Solution of Systems of Equations Arising from RBF-FD Meshless Discretizations.

8 Accepted (2019), R. Zamolo, 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.*

Appendix A

Nearest neighbour search

In the context of meshless methods, the task of efficient nearest neighbour search (NNS) for a given node distribution is of utmost importance. The specific problem of efficient NNS arising in meshless methods consists in finding the n nodes (local nodes) from a set of N nodes (the meshless node distribution) which are closest to a given node, where $n \ll N$. This search is required to be performed in an efficient way, i.e., its cost should be $\mathcal{O}(1)$ or $\mathcal{O}(\log N)$ at most.

With particular reference to the meshless approach developed in this thesis, NNS is required in two different phases: node generation and RBF-FD discretization. NNS is required in node generation because a node-repel algorithm is employed, therefore it is required to find the nearest neighbours for each node in order to efficiently compute the corresponding nodal displacement due to repulsion. NNS is also required in RBF-FD discretization because the RBF approximation is chosen to be built using a local support, therefore nearest neighbouring nodes have to be found.

There exist many common techniques developed in the field of computer science that can be employed for the efficient NNS. These techniques include 2D quadtrees and 3D octrees [29, 87], k-d trees [5] and R-trees [42]. These techniques can be used with arbitrary node distributions where node clustering and/or large zones with no nodes can be encountered. However, since the node distributions employed in this work are characterized by a complete filling of the domain and a moderate node clustering, typically near the walls, two simpler techniques have been successfully employed. These techniques, which have a straightforward implementation, are both based on spatial binning and are described as follows.

A.1. NNS with cartesian binning

This is the most straightforward and fast technique for NNS in the case of nearly uniform distributions of nodes. Given a domain Ω in D dimensions where nodes are scattered, the smallest box $\mathcal{B}$ bounding Ω is considered. The box has the faces aligned with the cartesian axes and its dimensions are L_x, L_y (and L_z in 3D). For the sake of simplicity we assume that $\mathcal{B}$ is completely contained in the first quadrant/octant and has one vertex at the origin of the axes, without loss of generality.

The bounding box $\mathcal{B}$ is then spatially subdivided using a cartesian partition with N_x, N_y, N_z bins along each axis. Therefore each bin has dimensions $h_x = L_x/N_x$, $h_y = L_y/N_y$, $h_z = L_z/N_z$. The number of bins along each axis is chosen to have a total number of bins $N_x \times N_y$ $(\times N_z) \approx N$ and $h_x \approx h_y (\approx h_z)$ in order to have nearly square (cubic in 3D) bins.

For each bin, which is identified by a spatial indexing $\mathcal{B}_{i,j,k}$, $i = 1, \dots, N_x$, $j = 1, \dots, N_y$, $k = 1, \dots, N_z$, a list $\mathcal{L}_{i,j,k}$ of the nodes which are contained in $\mathcal{B}_{i,j,k}$ is created. The NNS for a generic point $\mathbf{x} = (x, y, z)$ is then performed as follows:

1. Evaluation of the spacing function at point $\mathbf{x}$: $\bar{s} = s(\mathbf{x})$;
2. Evaluation of the radius of the circle (or sphere in 3D) centered in $\mathbf{x}$ in which approximately n nodes are contained: $\bar{r} = \chi \bar{s} \sqrt[D]{n}$, where $\chi = \sqrt[D]{\zeta/(2\pi\upsilon)}$ is a constant which depends upon the dimensionality D: $\zeta = \sqrt{3}, \upsilon = 1$ in 2D and $\zeta = \sqrt{2}, \upsilon = 4/3$ in 3D;
3. Determination of the bins $\mathcal{B}_{i,j,k}$, $i = i_1, \dots, i_2$, $j = j_1, \dots, j_2$, $k = k_1, \dots, k_2$, which are likely to contain the nodes inside the circle (sphere) of radius $\bar{r}$:
$$\begin{aligned} i_1 &= \lfloor (x - \bar{r})/h_x \rfloor, \quad & i_2 &= \lceil (x + \bar{r})/h_x \rceil \\ j_1 &= \lfloor (y - \bar{r})/h_y \rfloor, \quad & j_2 &= \lceil (y + \bar{r})/h_y \rceil \\ k_1 &= \lfloor (z - \bar{r})/h_z \rfloor, \quad & k_2 &= \lceil (z + \bar{r})/h_z \rceil \end{aligned} \tag{A.1}$$
4. Determination of $\bar{n} \le n$ nodes closest to $\mathbf{x}$ and contained in the circle (sphere) of radius $\bar{r}$ from the list of nodes $\mathcal{L}_{i,j,k}$ of each of the previous bins $\mathcal{B}_{i,j,k}$;
5. If the number of nodes provided by point 4 is not enough, i.e., $\bar{n} < n$, the radius $\bar{r}$ is properly increased till n nodes are found.

The computational expensive part of the previous NNS procedure is given by point 4 where coordinates are compared and squared distances from $\mathbf{x}$ are actually computed. Since a nearly uniform node distribution is assumed, each bin contains $\mathcal{O}(1)$ nodes and the total number of distance evaluations is proportional to n.

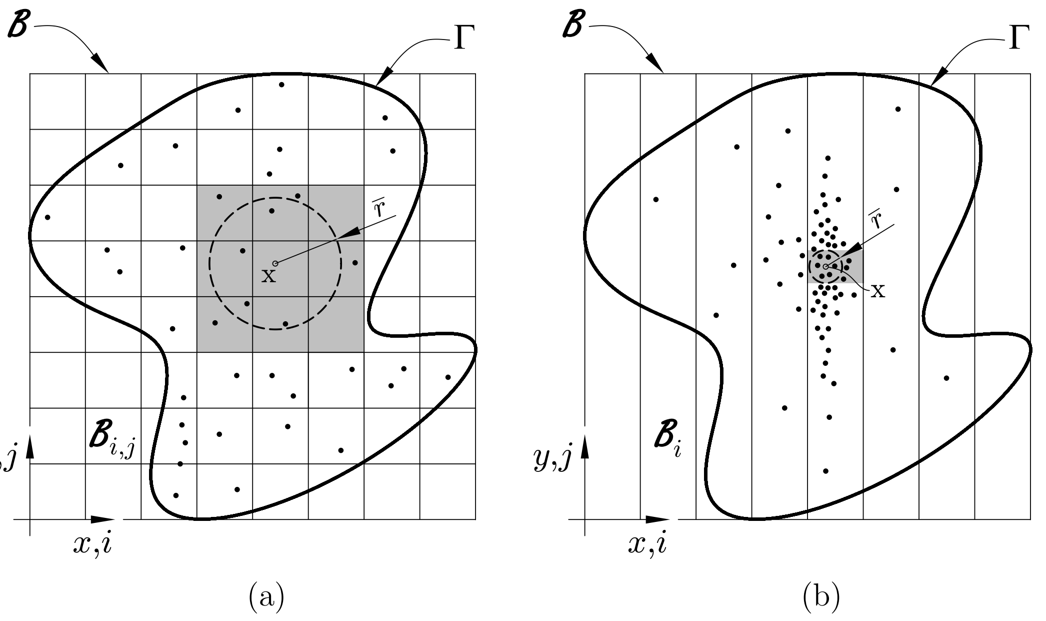


Figure A.1: 2D examples of the employed NNS algorithms: (a) cartesian binning, (b) cartesian binning with ordering along y.

Considering a small and constant number $n \ll N$, the cost of this NNS algorithm is therefore $\mathcal{O}(n) = \mathcal{O}(1)$. In the case of strong node clustering, the cost of this algorithm grows to $\mathcal{O}(N)$.

An example of the application of the NNS with cartesian binning is given in Figure A.1(a) for a 2D case where Γ is the domain boundary. In this example the algorithm is searching for $n = 4$ neighboring nodes from point $\mathbf{x}$, while the bins which are likely to contain the nodes inside the dashed circle of radius $\bar{r}$ are coloured in grey. These are the only bins queried by the NNS algorithm.

It is clear that this NNS algorithm can not be applied in the case of irregular node distributions characterized by areas where nodes are densely clustered. In such cases the list of nodes $\mathcal{L}_{i,j,k}$ of the corresponding bins may contain a very high number of nodes, requiring a lot of unnecessary coordinates comparisons and distance calculations and therefore reducing the efficiency of this algorithm. However, if the node clustering is not excessive the performances of this algorithm are satisfactory. Most of the node distributions considered in this thesis fall in the previous case and this NNS technique can be successfully employed. Furthermore, its implementation is incredibly straightforward and efficient.

A.2. NNS with cartesian binning and ordering

If node clustering is excessive, e.g., very small nodal spacing near a wall to accurately resolve thin boundary layers, the previous NNS technique can not be employed. A simple way to alleviate the clustering problem is to couple the binning technique along one (or two in 3D) dimensions to a sorting technique along the remaining dimension. This strategy alleviates the clustering problem along the dimension of the sorting since this dimension is not partitioned into equally spaced bins, while the clustering problem along the other dimensions remains.

Let us consider a 2D domain Ω for clarity of exposition. The smallest rectangle $\mathcal{B}$ bounding Ω is then considered, whose dimensions are L_x, L_y and having axis-aligned sides. For the sake of simplicity we assume that $\mathcal{B}$ is completely contained in the first quadrant and has one vertex at the origin of the axes, without loss of generality.

The rectangle $\mathcal{B}$ is then spatially subdivided into N_x bins along one dimension, which is assumed to be x. Therefore each bin is elongated along y and has dimensions $h_x = L_x/N_x$, $h_y = L_y$. The number of bins along each axis is chosen to be $N_x \propto \sqrt{N}$ in order to have $N/N_x \propto \sqrt{N}$ nodes for each bin in the case of a nearly uniform node distribution.

For each bin, which is identified by a spatial indexing $\mathcal{B}_i$ along x only, a list $\mathcal{L}_i$ of the nodes which are contained in $\mathcal{B}_i$ is created and the nodes are sorted along y. A cumulative function $Y_i(j)$ is then built using the y coordinates of the sorted nodes, allowing a fast identification of the nodes of each bin by means of their y coordinate. The NNS for a generic point $\mathbf{x} = (x, y)$ is then performed as follows:

1. Evaluation of the spacing function at point $\mathbf{x}$: $\bar{s} = s(\mathbf{x})$;

2. Evaluation of the radius of the circle centered in $\mathbf{x}$ in which approximately n nodes are contained: $\bar{r} = \chi\bar{s}\sqrt{n}$, where $\chi = \sqrt[4]{3}/\sqrt{2\pi} = 0.525$;

3. Determination of the bins $\mathcal{B}_i$, $i = i_1, \ldots, i_2$, which are likely to contain the nodes inside the circle of radius $\bar{r}$:

$$i_1 = \lfloor (x - \bar{r})/h_x \rfloor, \quad i_2 = \lceil (x + \bar{r})/h_x \rceil \tag{A.2}$$

4. For each of the previous bins $\mathcal{B}_i$, nodes (x_l, y_l) with $y - \bar{r} < y_l < y + \bar{r}$ are efficiently selected from the corresponding bin list $\mathcal{L}_i$ using the cumulative function $Y_i(j)$. From this reduced node set, $\bar{n} \le n$ nodes closest to $\mathbf{x}$ and contained in the circle of radius $\bar{r}$ are determined;

5. If the number of nodes provided by point 4 is not enough, i.e., $\bar{n} < n$, the radius $\bar{r}$ is properly increased till n nodes are found.

The selection of the nodes contained in the horizontal strip $y \pm \bar{r}$, performed in point 4, is efficiently carried out by applying the bisection method on $Y_i(j)$, therefore its total cost is $\mathcal{O}(\sqrt{n}\log(\varrho N_x))$, where $\varrho = s_M/s_m$ is the ratio between maximum and minimum node spacing. The total number of operations, i.e., coordinates comparisons, is $\mathcal{O}(\varrho\sqrt{n})$, regardless of the direction of node clustering. However, the constant hidden by the asymptotic notation is bigger when the node clustering does not occurs along the sorting direction.

Considering a small and constant number $n \ll N$ and a constant spacing ratio ϱ, the cost of the NNS algorithm with cartesian binning and ordering is therefore $\mathcal{O}(\sqrt{n}\log(\varrho N_x)) + \mathcal{O}(\varrho\sqrt{n}) = \mathcal{O}(\log N)$. The use of the sorting process can thus yield a great reduction of computational costs in the case of strong node clustering, while its implementation is slightly more complicated than the NNS algorithm with simple cartesian binning.

The presented NNS technique can also be extended to 3D domains considering a 2D cartesian binning along one plane coupled with the sorting technique along the remaining dimension, preferably the dimension along which the node clustering occurs.

An example of the application of the NNS with cartesian binning with ordering is given in Figure A.1(b) for a 2D case where Γ is the domain boundary and a node distribution with a node clustering along y is employed. In this example the algorithm is searching for $n = 6$ neighboring nodes from point $\mathbf{x}$, while the vertical portion of the bin which is likely to contain the nodes inside the dashed circle of radius $\bar{r}$ is coloured in grey. The nodes contained in such portion are the only nodes queried by the NNS algorithm, avoiding unnecessary calculations.

The NNS with cartesian binning and ordering has been used in this thesis in the cases where strong node clustering was required.

Appendix B

Solution of the RBF interpolation system

The square interpolation matrix $\mathbf{M}$ defined by the RBF interpolation system of Eq. (3.13) in the generic case of n_B boundary nodes is:

$$\mathbf{M} = \left[\begin{array}{ccc|ccc} \varphi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_1 - \mathbf{x}_n\|) & p_1(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{x}_{n_I} - \mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_{n_I} - \mathbf{x}_n\|) & p_1(\mathbf{x}_{n_I}) & \cdots & p_m(\mathbf{x}_{n_I}) \\ \hline \Psi_1(\hat{\mathbf{x}}_1) & \cdots & \Psi_n(\hat{\mathbf{x}}_1) & \Pi_1(\hat{\mathbf{x}}_1) & \cdots & \Pi_m(\hat{\mathbf{x}}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \Psi_1(\hat{\mathbf{x}}_{n_B}) & \cdots & \Psi_n(\hat{\mathbf{x}}_{n_B}) & \Pi_1(\hat{\mathbf{x}}_{n_B}) & \cdots & \Pi_m(\hat{\mathbf{x}}_{n_B}) \\ \hline p_1(\mathbf{x}_1) & \cdots & p_1(\mathbf{x}_n) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_m(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_n) & 0 & \cdots & 0 \end{array}\right] \tag{B.1}$$

In the case of the multiquadric RBF $\varphi(r) = \sqrt{1 + (\varepsilon r)^2}$ with small shape factors ε, the RBF entries of $\mathbf{M}$, i.e., the top left submatrix in Eq. (B.1), are very close to 1 and it is convenient to store these entries as $m_{ij} = 1 + b_{ij}$, where the entries b_{ij} are evaluated from the Maclaurin expansion of $\varphi(r)$ around $r = 0$:

$$b_{ij} = \sum_{k=1}^{K} a_k \frac{(\varepsilon r_{ij})^{2k}}{k!} \tag{B.2}$$

where $a_k = \left(\frac{1}{2}\right)\left(\frac{1}{2} - 1\right)\left(\frac{1}{2} - 2\right) \cdots \left(\frac{1}{2} - k + 1\right)$ and $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$. The number of terms K to consider must satisfy $(\varepsilon r_{ij})^{2K}/(K+1) < err$ where $err = 10^{-16}$ in double precision.

Lastly, in the evaluation of the polynomial entries in $\mathbf{M}$, i.e., the top right and the bottom left submatrices in Eq. (B.1), it is convenient to perform a shift to the mean point $\bar{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{x}_i/n$ followed by a scaling with respect to the spacing function at the mean point $\bar{s} = s(\bar{\mathbf{x}})$ in order not to have unnecessary large entries which affect the conditioning of the matrix:

$$p_k(\mathbf{x}_i) \leftarrow p_k\left(\frac{\mathbf{x}_i - \bar{\mathbf{x}}}{\bar{s}}\right) \tag{B.3}$$

The approximation of the RBF derivatives, Eq. (3.17), requires the solution of the linear system:

$$\mathbf{M}^T \mathbf{d} = \boldsymbol{\chi} \tag{B.4}$$

where $\mathbf{d}$ is the unknown vector of the shape functions for the sought derivatives and $\boldsymbol{\chi}$ is the known vector of the derivatives of the basis functions, as expressed by Eq. (3.17).

For the sake of simplicity, it is convenient to perform some row/column permutations in $\mathbf{M}$ in order to have the largest top left submatrix which is symmetric. This is accomplished by shifting the rows corresponding to the boundary conditions towards the bottom, followed by a shift of the columns associated to the boundary nodes towards the right:

$$\mathbf{M}' = \left[\begin{array}{ccc|ccc|ccc}
\varphi(r_{1,1}) & \cdots & \varphi(r_{1,n_I}) & p_1(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_1) & \varphi(r_{1,n_I+1}) & \cdots & \varphi(r_{1,n}) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\varphi(r_{n_I,1}) & \cdots & \varphi(r_{n_I,n_I}) & p_1(\mathbf{x}_{n_I}) & \cdots & p_m(\mathbf{x}_{n_I}) & \varphi(r_{n_I,n_I+1}) & \cdots & \varphi(r_{n_I,n}) \\
\hline
p_1(\mathbf{x}_1) & \cdots & p_1(\mathbf{x}_{n_I}) & 0 & \cdots & 0 & p_1(\mathbf{x}_{n_I+1}) & \cdots & p_1(\mathbf{x}_n) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
p_m(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_{n_I}) & 0 & \cdots & 0 & p_m(\mathbf{x}_{n_I+1}) & \cdots & p_m(\mathbf{x}_n) \\
\hline
\Psi_1(\hat{\mathbf{x}}_1) & \cdots & \Psi_{n_I}(\hat{\mathbf{x}}_1) & \Pi_1(\hat{\mathbf{x}}_1) & \cdots & \Pi_m(\hat{\mathbf{x}}_1) & \Psi_{n_I+1}(\hat{\mathbf{x}}_1) & \cdots & \Psi_n(\hat{\mathbf{x}}_1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\Psi_1(\hat{\mathbf{x}}_{N_B}) & \cdots & \Psi_{n_I}(\hat{\mathbf{x}}_{N_B}) & \Pi_1(\hat{\mathbf{x}}_{N_B}) & \cdots & \Pi_m(\hat{\mathbf{x}}_{N_B}) & \Psi_{n_I+1}(\hat{\mathbf{x}}_{N_B}) & \cdots & \Psi_n(\hat{\mathbf{x}}_{N_B})
\end{array}\right] \tag{B.5}$$

which can be written as:

$$\mathbf{M}' = \begin{bmatrix} \mathbf{S} & \mathbf{Q}_{IB}^T \\ \mathbf{Q}_{BI}^T & \mathbf{Q}_{BB}^T \end{bmatrix} \tag{B.6}$$

where $\mathbf{S}$ is the symmetric top left matrix of dimension $n_I + m$ and the remaining matrices follow.

The linear system of Eq. (B.4) with the previous row/column permutations becomes:

$$\mathbf{M}'^T \mathbf{d}' = \boldsymbol{\chi}' \tag{B.7}$$

which can be written as:

$$\begin{bmatrix} \mathbf{S} & \mathbf{Q}_{BI} \\ \mathbf{Q}_{IB} & \mathbf{Q}_{BB} \end{bmatrix} \begin{Bmatrix} \mathbf{d}'_I \\ \mathbf{d}'_B \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\chi}'_I \\ \boldsymbol{\chi}'_B \end{Bmatrix} \tag{B.8}$$

In order to provide an accurate solution of the linear system of Eq. (B.8), the symmetry of $\mathbf{S}$ is exploited by calculating its Schur complement [48], i.e., matrix elimination. The first matrix row is multiplied by $\mathbf{Q}_{IB}\mathbf{S}^{-1}$ and subtracted from the second row, obtaining the following square system:

$$\left(\mathbf{Q}_{BB} - \mathbf{Q}_{IB}\mathbf{S}^{-1}\mathbf{Q}_{BI}\right)\mathbf{d}'_B = \boldsymbol{\chi}'_B - \mathbf{Q}_{IB}\mathbf{S}^{-1}\boldsymbol{\chi}'_I \tag{B.9}$$

which is solved for $\mathbf{d}'_B$ by Gaussian elimination with pivoting. $\mathbf{d}'_I$ is then obtained from the first row of Eq. (B.8):

$$\mathbf{d}'_I = \mathbf{S}^{-1}\boldsymbol{\chi}'_I - \mathbf{S}^{-1}\mathbf{Q}_{BI}\mathbf{d}'_B \tag{B.10}$$

In the case of no boundary nodes ($n_B = 0$), the previous procedure reduces to the solution of the symmetric system of Eq. (B.4) where $\mathbf{M} = \mathbf{S}$.

The whole procedure relies on the accurate solution of linear systems where the coefficient matrix is $\mathbf{S}$ which is symmetric but not (positive) definite. In fact, in the case of the polynomial augmentation of the MQ RBF interpolant, it can be shown that $\mathbf{S}$ has m strictly positive eigenvalues and $n_I - m$ strictly negative eigenvalues. Gaussian elimination is avoided because $\mathbf{S}$ can be very ill-conditioned for small shape factors ε and large stencil sizes n, therefore a LDL^T factorization of $\mathbf{S}$ is required. Such factorization is defined by:

$$\mathbf{S} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{B.11}$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{L}$ is a lower triangular matrix with unitary diagonal. Once $\mathbf{S}$ is factorized, the accurate solution of any linear system is carried out by the usual forward-backward substitution.

Appendix C

Powers of the laplacian for the MQ RBF

The multiquadric (MQ) RBF defined by Eq. (3.22) is:

$$\phi_j(\mathbf{x}) = \varphi(r) = \sqrt{1 + (\varepsilon r)^2} \tag{C.1}$$

where $r = \|\mathbf{x} - \mathbf{x}_j\|$.

C.1. 2D case

The laplacian (del) operator $\nabla^2 = \Delta$ for a generic RBF in 2D (cylindrical coordinates) is:

$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\Big(r\frac{\partial}{\partial r}\Big) \tag{C.2}$$

The laplacian of the MQ RBF $\varphi(r)$ is a polynomial in φ^{-1} itself, therefore the powers of the laplacian $\nabla^{2k} = \Delta^k$, i.e., k repeated applications of the laplacian, are again polynomials in φ^{-1}. It is therefore convenient to obtain a formula for the laplacian of a generic power p of φ:

$$\nabla^2\varphi^p(r) = p\varepsilon^2\big[(2-p)\varphi^{p-4}(r) + p\varphi^{p-2}(r)\big] \tag{C.3}$$

which allows the recursive calculation of the powers of the laplacian $\nabla^{2k} = \Delta^k$

starting from the simple laplacian for $k = 1$ and $p = 1$:

$$\begin{aligned}
\nabla^2\varphi(r) &= \varepsilon^2\big[\varphi^{-3}(r) + \varphi^{-1}(r)\big]\\
\nabla^4\varphi(r) &= \varepsilon^4\big[-15\varphi^{-7}(r) + 6\varphi^{-5}(r) + \varphi^{-3}(r)\big]\\
\nabla^6\varphi(r) &= \varepsilon^6\big[945\varphi^{-11}(r) - 945\varphi^{-9}(r) + 135\varphi^{-7}(r) + 9\varphi^{-5}(r)\big]\\
\nabla^8\varphi(r) &= \varepsilon^8\big[-135135\varphi^{-15}(r) + 207900\varphi^{-13}(r)+\\
&\qquad - 85050\varphi^{-11}(r) + 6300\varphi^{-9}(r) + 225\varphi^{-7}(r)\big]
\end{aligned} \tag{C.4}$$

C.2. 3D case

The laplacian (del) operator $\nabla^2 = \Delta$ for a generic RBF in 3D (spherical coordinates) is:

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) \tag{C.5}$$

The laplacian of the MQ RBF $\varphi(r)$ is a polynomial in φ^{-1} itself, therefore the powers of the laplacian $\nabla^{2k} = \Delta^k$, i.e., k repeated applications of the laplacian, are again polynomials in φ^{-1}. It is therefore convenient to obtain a formula for the laplacian of a generic power p of φ:

$$\nabla^2\varphi^p(r) = p\varepsilon^2\big[(2-p)\varphi^{p-4}(r) + (p+1)\varphi^{p-2}(r)\big] \tag{C.6}$$

which allows the recursive calculation of the powers of the laplacian $\nabla^{2k} = \Delta^k$ starting from the simple laplacian for $k = 1$ and $p = 1$:

$$\begin{aligned}
\nabla^2\varphi(r) &= \varepsilon^2\big[\varphi^{-3}(r) + 2\varphi^{-1}(r)\big]\\
\nabla^4\varphi(r) &= \varepsilon^4\big[-15\varphi^{-7}(r)\big]\\
\nabla^6\varphi(r) &= \varepsilon^6\big[945\varphi^{-11}(r) - 630\varphi^{-9}(r)\big]\\
\nabla^8\varphi(r) &= \varepsilon^8\big[-135135\varphi^{-15}(r) + 166320\varphi^{-13}(r) - 45360\varphi^{-11}(r)\big]
\end{aligned} \tag{C.7}$$

Appendix D

ACMC, SRMC and linear Multicloud

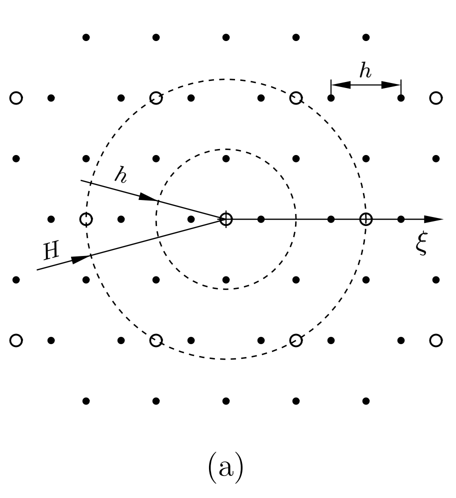
This Appendix is intended to give some analytical insight about the MC approaches and to show the different effectiveness between ACMC, SRMC and linear MC (not employed in this work) within a two-level MC approach for a Poisson problem.

Let us consider a continuous-space approximation of the restriction operator $\mathbf{I}_h^H$, i.e., a weighted integral where the weight w_R is the restriction shape function. The application of such weighted integral to Eq. (4.5) in terms of the sought error $e = f - f_k$, i.e., $-\nabla^2 e = r$ where $r = q + \nabla^2 f_k$ is the residual, gives:

$$\begin{aligned} -\int_W w_R \nabla^2 e \, \mathrm{d}W &= \int_W \Big[\nabla w_R \cdot \nabla e \, \mathrm{d}W - \nabla \cdot (w_R \nabla e)\Big] \, \mathrm{d}W \\ &= \int_W \nabla w_R \cdot \nabla e \, \mathrm{d}W - \underbrace{\int_{\partial W} w_R \nabla e \cdot \mathbf{n} \, \mathrm{d}\partial W}_{C=0} = \int_W w_R r \, \mathrm{d}W \end{aligned} \tag{D.1}$$

where W is the restriction support, i.e., where $w_R > 0$, and the divergence theorem has been applied to C, which always vanishes because $w_R = 0$ on ∂W by definition. Let us consider the node distributions of Figure D.1a, where h is the fine-level nodal spacing, the ratio between the fine-level nodes and coarse-level nodes is 4 and therefore the coarse-level nodal spacing is $H = 2h$; the coarse-level nodes have been moved in order not to have coincident fine/level nodes. Let us also consider the radial functions w_c and w_l defined by the radial profiles of Figure D.1b where ξ is the radial coordinate of Figure D.1a. w_c and w_l are, respectively, the restriction shape functions for constant and linear MC, considered here as radial functions for sake of simplicity, while J_c is defined as:

$$J_c(\xi) = \frac{1}{2\pi h} \int_{\bar{C}(\xi)} w_c \, \mathrm{d}\bar{C} \tag{D.2}$$



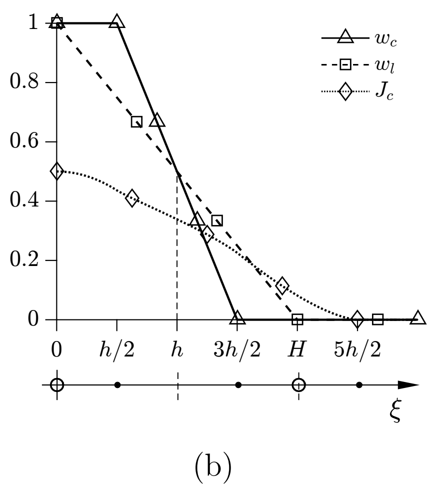


Figure D.1: Fine and coarse-level node distributions for continuous-space approximation (a), continuous-space approximation weight functions (b).

where $\bar{C}(\xi)$ is a circle of radius h centered at radial coordinate ξ. The (radial) restriction shape function for SRMC is then given by $w_{sr} = (1 - \omega_S)w_c + \omega_S J_c$, since this is the continuous-space approximation of one ω_S-Jacobi iteration on the restriction shape function w_c in the case of a Poisson problem.

Finally, let us consider a constant residual $r = 1$ and an error of the form $e = \bar{e}w_I$, where $\bar{e} \in \mathcal{R}$ is the sought coarse-level correction and w_I is the continuous-space approximation of the interpolation operator $\mathbf{I}_H^h$, i.e., the interpolation shape function which is also radial. Therefore $w_I = w_c$ for ACMC and SRMC, while $w_I = w_l$ for the linear MC.

In the end, the coarse-level correction $\bar{e}$ is obtained from Eq. D.1 as follows:

$$\bar{e} = \frac{\displaystyle\int_W w_R \, \mathrm{d}W}{\displaystyle\int_W \nabla w_R \cdot \nabla w_I \, \mathrm{d}W} \tag{D.3}$$

Therefore the coarse-level correction $\bar{e}$ depends upon the choices for w_R and w_I, i.e., the choices for interpolation and restriction operators. The coarse-level corrections for ACMC and SRMC are reported in Table D.1, where $\bar{e}_{lin}$ is the linear MC coarse-level correction, $\bar{d} = 3/4 + (h/H)^2/4 \approx 0.41$ and $\Delta J_c = J_c(h/2) - J_c(3h/2) \approx 0.21$.

These results indicate that ACMC without overcorrection leads to an incomplete reduction of the smooth error components [7, 94] since $\bar{e} \ll \bar{e}_{lin}$ and therefore a rough estimate for the overcorrection factor is $t = 1/\bar{d} \approx 2.5$, which is slightly larger than the optimal overcorrection factor $t = 1.7 - 2.0$ suggested by the numerical results of Chapter 4. The compensation for the incomplete reduction of

Table D.1: ACMC and SRMC coarse corrections ratios.

	ACMC	**SRMC**
$\bar{e}/\bar{e}_{lin}$	$\bar{d}$	$\bar{d}/[1-\omega_S(1-\Delta J_c)]$

the smooth error components in the SRMC case is obtained by using an under-relaxation smoothing factor $0 < \omega_S < 1$, for which the results of Table D.1 provide a rough estimate $\omega_S \approx 0.75$ which is again slightly larger than the optimal relaxation parameter $\omega_S = 0.55 - 0.60$ suggested by the numerical results.

Appendix E

An analytical solution in cylindrical coordinates

Let us consider the set of incompressible Navier-Stokes equations expressed by Eqs. (5.5)-(5.6) in the 2D isothermal case using cylindrical coordinates (r, ϑ). If we suppose a laminar flow with tangential velocity $u_\vartheta(r,t)$, radial velocity $u_r = 0$ and pressure $p(r,t)$, the continuity Eq. (5.5) is always satisfied while the momentum Eq. (5.6) reduces to the following scalar equations:

$$\frac{\partial u_\vartheta}{\partial t} = \frac{1}{Re}\left[\frac{1}{r}\frac{\partial}{\partial r}\Big(r\frac{\partial u_\vartheta}{\partial r}\Big) - \frac{u_\vartheta}{r^2}\right] \tag{E.1a}$$

$$\frac{\partial p}{\partial r} = \frac{u_\vartheta^2}{r} \tag{E.1b}$$

Since Eqs. (E.1a)-(E.1b) are decoupled, i.e., the pressure p does not appear in the first equation, it is possible to solve Eq. (E.1a) for u_ϑ while the pressure is then obtained by integrating Eq. (E.1b) along the radius r:

$$p = \int \frac{u_\vartheta^2}{r}\,\mathrm{d}r = u_\vartheta^2 \log r - 2\int u_\vartheta \frac{\partial u_\vartheta}{\partial r}\log r\,\mathrm{d}r + c \tag{E.2}$$

where c is an additive constant.

The solution for Eq. (E.1a) is sought in the form $u_\vartheta(r,t) = e^{-\lambda t}R(r)$ for which Eq. (E.1a) becomes:

$$\lambda R = \frac{1}{Re}\Big(R'' + \frac{R'}{r} - \frac{R}{r^2}\Big) \tag{E.3}$$

which can be recast in the following form:

$$r^2R'' + rR' + (r^2\lambda Re - 1)R = 0 \tag{E.4}$$

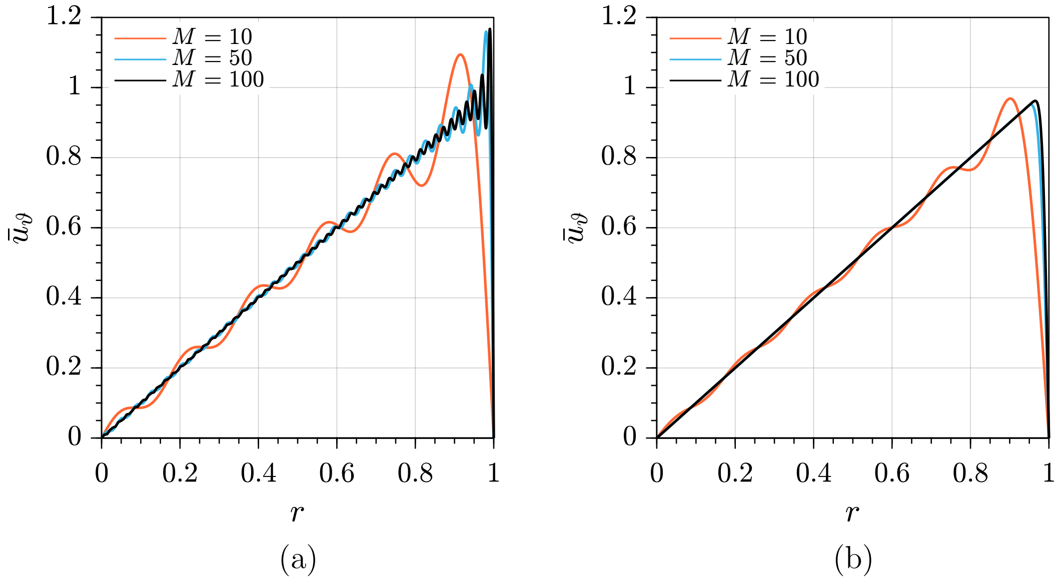


Figure E.1: Approximation of the discontinuous function $\bar{u}_\vartheta(r < 1) = r$, $\bar{u}_\vartheta(r = 1) = 0$: (a) minimization on $[0, 1]$; (b) minimization on $[0, 0.95]$.

Defining a new variable $\bar{r} = r\sqrt{\lambda Re}$ and a new function $J(\bar{r}) = R(r)$, Eq. (E.4) becomes:

$$\bar{r}^2 J'' + \bar{r} J' + (\bar{r}^2 - 1)J = 0 \tag{E.5}$$

whose solutions for this problem are given by the first order Bessel functions of the first kind $J(\bar{r}) = J_1(\bar{r})$ (first order because of the constant term -1^2 in the coefficient $(\bar{r}^2 - 1)$, while first kind Bessel functions are finite in $\bar{r} = 0$, contrary to second kind Bessel functions which can not therefore be considered [2]).

Let us consider a circular domain with radius $r = 1$ and a Dirichlet boundary condition $u_\vartheta(r = 1, t) = 0$, i.e., no-slip conditions at the circular boundary. This condition corresponds to $R(r = 1) = J_1(\bar{r} = \sqrt{\lambda Re}) = 0$ which can be satisfied if and only if $\sqrt{\lambda Re} = z_i$ where $z_i > 0$ are the roots of J_1, from which we obtain $\lambda = z_i^2/Re$. Since the number of roots is infinite, the solution to the problem is therefore given by an infinite series:

$$u_\vartheta(r, t) = \sum_{i=1}^{\infty} A_i e^{-z_i^2 t/Re} J_1(z_i r) \tag{E.6}$$

where the expansion coefficients A_i must be determined in order to satisfy the following initial condition:

$$u_\vartheta(r, t = 0) = \sum_{i=1}^{\infty} A_i J_1(z_i r) = \bar{u}_\vartheta(r) \tag{E.7}$$

where $\bar{u}_\vartheta(r)$ is the initial profile for the velocity.

Since the analytical solution is employed for numerical comparisons, it is therefore useful to consider a limited number M of terms in the expansion defined by (E.6) as follows:

$$u_\vartheta(r,t) = \sum_{i=1}^{M} A_i e^{-z_i^2 t/Re} J_1(z_i r) \tag{E.8}$$

and the expansion coefficients A_i are determined by a numerical minimization of the error between the initial profile $\bar{u}_\vartheta(r)$ and the sought solution.

The discontinuous case $\bar{u}_\vartheta(r) = r$ is considered, which corresponds to an initial velocity profile with rigid rotation subjected to an instantaneous no-slip condition at $r = 1$ and $t \geq 0$ with a zero tangential velocity $u_\vartheta = 0$. The approximation of this initial profile is depicted in Figure E.1a for the minimization of the error over the whole interval $r \in [0, 1]$, while Figure E.1b shows the approximation when the minimization is performed on the interval $r \in [0, 0.95]$ which excludes the singularity at $r = 1$ and avoids the appearance of the typical oscillations due to a discontinuity.

Appendix F

Code

In order to appreciate the ease of coding with a meshless data structure, some of the main sections of the developed code for the node generation process will be listed in the following.

F.1. Cartesian binning for nearest neighbour search

The following C code performs the 2D cartesian binning needed for the nearest neighbour search presented in Section A.1; nodes are limited to be contained in the unit square $[0, 1]^2$.

- N_cell: vector containing the indices of the nodes, ordered by cartesian binning
- N_ix: vector containing the starting indices in N_cell for each cartesian cell
- x, y: nodal coordinates
- N: number of nodes
- Ngrid_xy: number of cartesian bins for each direction

```
void SpaceInversion( int *N_cell , int *N_ix ,
                     int N , float *x , float *y , int Ngrid_xy ) {

 int i , j , ix ;
 int Ngrid = Ngrid_xy * Ngrid_xy ;

 // 0 initialization
 for ( i = 0 ; i <= Ngrid ; i++ ) {
   N_ix[i] = 0 ;
 }
```

```
  // Number of nodes for each cell (N_ix)
  int *kij = (int *) malloc( N * sizeof(int) ) ;
  for ( ix = 0 ; ix < N ; ix++ ) {
    i = (int) ( x[ix] * ( Ngrid_xy - 1.0f ) + 0.5f ) ;
    j = (int) ( y[ix] * ( Ngrid_xy - 1.0f ) + 0.5f ) ;
    k_ij[ix] = i + Ngrid_xy * j ;
    N_ix[k_ij[ix]+1]++ ;
  }

  // Starting indices for each cell in N_cell (cumulative sum, N_ix)
  N_ix[0] = 0 ;
  for ( i = 0 ; i < Ngrid ; i++ ) {
    N_ix[i+1] += N_ix[i] ;
  }

  // Copy from N_ix to N_ix_temp
  int    *N_ix_temp = (int *) malloc( (Ngrid+1) * sizeof(int) ) ;
  memcpy( N_ix_temp , N_ix ,          (Ngrid+1) * sizeof(int) ) ;

  // Node number assignment (N_cell)
  for ( ix = 0 ; ix < N ; ix++ ) {
    N_cell[N_ix_temp[k_ij[ix]]++] = ix ;
  }

  free( N_ix_temp ) ;
  free( k_ij ) ;
  return ;
}
```

F.2. Node-repel refinement

The following C code performs the 2D node-repel refinement presented in Section 2.3; nodes are limited to be contained in the unit square $[0,1]^2$.

- N_cell: vector containing the indices of the nodes, ordered by cartesian binning
- N_ix: vector containing the starting indices in N_cell for each cartesian cell
- x, y: nodal coordinates
- N: number of nodes
- Ngrid_xy: number of cartesian bins for each direction
- P_type: vector defining boundary nodes
- beta: β repel parameter
- n_nb: number of neigbouring nodes for repulsion calculation

- `fct`: $1/\alpha$ repel parameter

```
#include <math.h>

float minf( float a , float b ) {
a = b<a ? b : a ;
return a ;
}

int limits( int i , int max ){
if( i<0 )   { i = 0 ; }
if( i>max ) { i = max ; }
return i ;
}

float limits_f( float x , float max ){
if( x<0.0f ) { x = 0.0f ; }
if( x>max  ) { x = max ; }
return x ;
}

// Spacing function
float s( float x , float y ) {
float d ;
d = // add spacing function
return d ;
}

void NodeRefinement( int *N_cell , int *N_ix ,
                     int N , int Ngrid_xy ,
                     float *x , float *y , char *P_type ,
                     int steps , float beta , int n_nb , float fct
                     ) {

 float fct_s = 1.155f ; // = 2 / sqrt(3)
 float pig   = 3.141f ;
 int   i , j , k ,

 int    step   , node    , node2    , ix , i0 , i1 , j0 , j1 , ix0 , ix1 ;
 float  temp_x , temp_y , f_x , f_y , xp , yp , ri , dc , x2 , y2 ;
 float  dx , dy   , d , isqd , s_loc ;
 float  h    = 1.0f / ( Ngrid_xy - 1.0f ) ;
 float  lbd = sqrtf( n_nb / ( pig * fct_s ) ) , sq ;
 int    ix_max     = Ngrid_xy * Ngrid_xy ;
 int    Ngridxy_max = Ngrid_xy-1 ;

 // Loop over steps refinement iterations
 for( step = 0 ; step < steps ; step++ ) {

  // Cartesian binning for nearest neighbour search
  SpaceInversion( N_cell , N_ix , N , x , y , Ngrid_xy ) {

  // Loop over each free node
  for( node = 0 ; node < N ; node++ ) {
   if ( P_type[node] == 0 ) {
    // Node coordinates
    xp = x[node] ;
    yp = y[node] ;

    // Cartesian indices for nearest cells
    temp_x = xp * ( Ngrid_xy - 1.0f ) + 0.5f ;
    temp_y = yp * ( Ngrid_xy - 1.0f ) + 0.5f ;
    dc = lbd * s( xp , yp ) ;
    ri = dc / h ;
    i0 = (int) ( temp_x - ri ) ; i0 = limits( i0 , Ngridxy_max ) ;
    i1 = (int) ( temp_x + ri ) ; i1 = limits( i1 , Ngridxy_max ) ;
    j0 = (int) ( temp_y - ri ) ; j0 = limits( j0 , Ngridxy_max ) ;
    j1 = (int) ( temp_y + ri ) ; j1 = limits( j1 , Ngridxy_max ) ;

    // Computation of forces through nearest neighbour search
    f_x = 0.0f ;
    f_y = 0.0f ;
    // Starting indices for nearest cells
    for ( j = j0 ; j <= j1 ; j++ ) {
     ix0 = Ngrid_x * j + i0 ;
     ix1 = Ngrid_x * j + i1 + 1 ;
     ix0 = N_ix[ix0] ;
     ix1 = N_ix[ix1] ;
     // Indices for nearest neighbour nodes
     for ( ix = ix0 ; ix < ix1 ; ix++ ) {
      node2 = N_cell[ix] ;
      if ( node2 != node ) {
       x2 = x[node2] ;
       y2 = y[node2] ;
       dx = x2 - xp ;
       dy = y2 - yp ;
       d  = hypotf( dx , dy ) ;
       if ( d < dc ) {
        // Repulsive force calculation
        s_loc = s( x2 , y2 ) ;
        isqd = d / s_loc ;
        isqd *= isqd ;
        isqd = isqd + beta ;
        isqd *= isqd ;
        isqd = 1.0f / (d * isqd) ;
        f_x += isqd * dx ;
        f_y += isqd * dy ;
       }
      }
     }
    }
    sq = fct_iter * s( xp , yp ) ;
    // Coordinates update
    x[node] -= sq * f_x ; x[node] = limits_f( x[node] , 1.0f ) ;
    y[node] -= sq * f_y ; y[node] = limits_f( y[node] , 1.0f ) ;
   }
  }
 }
 return ;
}
```

Bibliography

[1] T. A. AbdelMigid, K. M. Saqr, M. A. Kotb, and A. A. Aboelfarag, *Revisiting the lid-driven cavity flow problem: Review and new steady state benchmarking results using gpu accelerated code*, Alexandria Engineering Journal, 56 (2017), pp. 123 – 135.

[2] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover Publications, Inc., New York, NY, USA, 9th dover printing, 10th gpo printing ed., 1964.

[3] S. Albensoeder and H. C. Kuhlmann, *Accurate three-dimensional lid-driven cavity flow*, Journal of Computational Physics, 206 (2005), pp. 536–558.

[4] V. Bayona, N. Flyer, B. Fornberg, and G. Barnett, *On the Role of Polynomials in RBF-FD Approximations: II. Numerical Solution of Elliptic PDEs*, Journal of Computational Physics, 332 (2017), pp. 257–273.

[5] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM, 18 (1975), pp. 509–517.

[6] R. Blaheta, *A multilevel method with correction by aggregation for solving discrete elliptic problems*, Applications of Mathematics, 31 (1986), pp. 365–378.

[7] ——, *A multilevel method with overcorrection by aggregation for solving discrete elliptic problems*, Journal of Computational and Applied Mathematics, 24 (1988), pp. 227–239.

[8] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.

[9] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Mathematics of Computation, 31 (1977), pp. 333–390.

[10] C.-H. Bruneau and M. Saad, *The 2D Lid-Driven Cavity Problem Revisited*, Computers & Fluids, 35 (2006), pp. 326–348.

[11] T. Chan and B. Smith, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Electronic Transactions On Numerical Analysis, 2 (1994), pp. 171–182.

[12] G. Chandhini and Y. Sanyasiraju, *Local RBF-FD Solutions for Steady Convection-diffusion Problems*, International Journal for Numerical Methods in Engineering, 72 (2007), pp. 352–378.

[13] J.-H. Chen, W. G. Pritchard, and S. J. Tavener, *Bifurcation for Flow Past a Cylinder Between Parallel Planes*, Journal of Fluid Mechanics, 284 (1995), pp. 23–41.

[14] P. Chinchapatnam, K. Djidjeli, P. Nair, and M. Tan, *A Compact RBF-FD Based Meshless Method for the Incompressible Navier-Stokes Equations*, Proc. IMechE, Part M: J. Engineering for the Maritime Environment, 223 (2009), pp. 275–290.

[15] P. P. Chinchapatnam, *Radial basis function based meshless methods for fluid flow problems*, PhD thesis, University of Southampton, 2006.

[16] A. Chorin, *Numerical Solution of the Navier-Stokes Equations*, Mathematics of Computation, 22 (2015), pp. 745–762.

[17] D. Contrino, P. Lallemand, P. Asinari, and L.-S. Luo, *Lattice-Boltzmann Simulations of the Thermally Driven 2D Square Cavity at High Rayleigh Numbers*, Journal of Computational Physics, 275 (2014), pp. 257–272.

[18] R. Courant, K. Friedrichs, and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik*, Mathematische Annalen, 100 (1928), pp. 32–74.

[19] E. Cuthill and J. McKee, *Reducing the Bandwidth of Sparse Symmetric Matrices*, in Proceedings of the 1969 24th National Conference, ACM '69, New York, NY, USA, 1969, ACM, pp. 157–172.

[20] F. de la Vallee Poussin and W. Timlake, *An accelerated relaxation algorithm for iterative solution of elliptic equations*, SIAM Journal of Numerical Analysis, 5 (1968), pp. 340–351.

[21] L. Demkowicz, A. Karafiat, and T. Liszka, *On some convergence results for FDM with irregular mesh*, Computer Methods in Applied Mechanics and Engineering, 42 (1984), pp. 343–355.

[22] E. Divo and A. J. Kassab, *An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer*, ASME Journal of Heat Transfer, 129 (2006), pp. 124–136.

[23] ——, *Efficient localized meshless modeling of natural convective viscous flows*, 9th AIAA/ASME Joint Thermophysics and Heat Transfer Conference. San Francisco, California, 2006.

[24] ——, *Localized meshless modeling of natural-convective viscous flows*, Numerical Heat Transfer, Part B: Fundamentals, 53 (2008), pp. 487–509.

[25] T. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Computers & Mathematics with Applications, 43 (2002), pp. 413 – 422.

[26] G. Fasshauer, *Meshfree Approximation Methods with Matlab*, vol. 6 of Interdisciplinary Mathematical Sciences, World Scientific, Singapore, 2007.

[27] R. Fedorenko, *The speed of convergence of one iterative process*, USSR Computational Mathematics and Mathematical Physics, 4 (1964), pp. 227–235.

[28] ——, *Iterative methods for elliptic difference equations*, Russian Mathematical Surveys, 28 (1973), pp. 129–195.

[29] R. Finkel and J. Bentley, *Quad trees a data structure for retrieval on composite keys*, Acta Informatica, 4 (1974), pp. 1–9.

[30] R. W. Floyd and L. Steinberg, *An Adaptive algorithm for spatial greyscale*, Proceedings of the Society for Information Display, 17 (1976), pp. 75–77.

[31] N. Flyer and B. Fornberg, *Radial basis functions: Developments and applications to planetary scale flows*, Computers & Fluids, 46 (2011), pp. 23 – 32. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[32] N. Flyer, B. Fornberg, V. Bayona, and G. Barnett, *On the Role of Polynomials in RBF-FD Approximations: I. Interpolation and Accuracy*, Journal of Computational Physics, 321 (2016), pp. 21–38.

[33] B. Fornberg and N. Flyer, *Fast generation of 2-d node distributions for mesh-free pde discretizations*, Computers & Mathematics with Applications, 69 (2015), pp. 531 – 544.

[34] B. Fornberg and N. Flyer, *Solving PDEs with Radial Basis Functions*, Acta Numerica, 24 (2015), pp. 215–258.

[35] B. Fornberg and E. Lehto, *Stabilization of rbf-generated finite difference methods for convective pdes*, Journal of Computational Physics, 230 (2011), pp. 2270 – 2285.

[36] B. Fornberg and G. B. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Computers & Mathematics with Applications, 48 (2004), pp. 853 – 867.

[37] A. Fortin, M. Jardak, J. Gervais, and R. Pierre, *Localization of Hopf Bifurcations in Fluid Flow Problems*, International Journal for Numerical Methods in Fluids, 24 (1997), pp. 1185–1210.

[38] R. Franke, *A Critical Comparison of Some Methods for Interpolation of Scattered Data (Technical Report)*, Tech. Report NPS-53-79-003, Naval Postgraduate School, Monterey, California, 1980.

[39] ——, *Scattered Data Interpolation: Tests of Some Methods*, Mathematics of Computation, 38 (1982), pp. 181–200.

[40] P. J. Frey and P.-L. George, *Mesh Generation*, ISTE, London, UK, 2010.

[41] T. Gjesdal, *A note on the additive correction multigrid method*, International Communications in Heat and Mass Transfer, 23 (1996), pp. 293–298.

[42] A. Guttman, *R-trees: A dynamic index structure for spatial searching*, ACM SIGMOD Record, 14 (1984), pp. 47–57.

[43] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations.*, vol. 95 of Applied Mathematical Sciences, Springer-Verlag, Berlin, Germany, 1994, ch. 6: Analysis for M-Matrices.

[44] U. Hanoglu, S. ul Islam, and B. Sarler, *Thermo-mechanical analysis of hot shape rolling of steel by a meshless method*, Procedia Engineering, 10 (2011), pp. 3173 – 3178. 11th International Conference on the Mechanical Behavior of Materials (ICM11).

[45] R. Hardy, *Multiquadric equations of topography and other irregular surfaces*, Journal of Geophysical Research, 76 (1971), pp. 1905–1915.

[46] R. Hardy, *Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968-1988*, Computers & Mathematics with Applications, 19 (1990), pp. 163 – 208.

[47] V. Hatić, B. Mavrič, N. Košnik, and B. Šarler, *Simulation of direct chill casting under the influence of a low-frequency electromagnetic field*, Applied Mathematical Modelling, 54 (2018), pp. 170 – 188.

[48] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[49] B. Hutchinson and G. Raithby, *A multigrid method based on the additive correction strategy*, Numerical Heat Transfer, Part A: Applications, 9 (1986), pp. 511–537.

[50] R. J. A. Janssen and R. A. W. M. Henkes, *Accuracy of Finite-Volume Discretizations for the Bifurcating Natural-Convection Flow in a Square Cavity*, Numerical Heat Transfer, Part B: Fundamentals, 24 (1993), pp. 191–207.

[51] P. S. Jensen, *Finite difference techniques for variable grids*, Computers & Structures, 2 (1972), pp. 17 – 29.

[52] 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 (1990), pp. 127–145.

[53] ——, *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 (1990), pp. 147–161.

[54] G. Karniadakis and S. Sherwin, *Spectral/hp element methods for computational fluid dynamics*, Oxford University Press, 2 ed., 2005.

[55] A. Katz and A. Jameson, *Multicloud: multigrid convergence with a meshless operator*, Journal Computational Physics, 228 (2009), pp. 5237–5250.

[56] A. J. Katz, *Meshless Methods for Computational Fluid Dynamics*, PhD thesis, Stanford, CA, USA, 2009. AAI3351453.

[57] G. Kosec, *A local numerical solution of a fluid-flow problem on an irregular domain*, Advances in Engineering Software, 120 (2018), pp. 36 – 44. Civil-Comp - Part 2.

[58] G. Kosec and B. Šarler, *Solution of Thermo-fluid Problems by Collocation with Local Pressure Correction*, International Journal Of Numerical Methods For Heat & Fluid Flow, 18 (2008), pp. 868–882.

[59] G. Kosec and B. Šarler, *H-adaptive local radial basis function collocation meshless method*, Computers, Materials & Continua, 26 (2011), pp. 227–253.

[60] G. Kosec and B. Šarler, *Solution of a Low Prandtl Number Natural Convection Benchmark by a Local Meshless Method*, International Journal Of Numerical Methods For Heat & Fluid Flow, 23 (2013), pp. 189–204.

[61] K. Y. Lam, Q. X. Wang, and H. Li, *A novel meshless approach -local kriging (lokriging) method with two-dimensional structural analysis*, Computational Mechanincs, 33 (2004), pp. 235–244.

[62] E. Larsson and B. Fornberg, *Theoretical and Computational Aspects of Multivariate Interpolation with Increasingly Flat Radial Basis Functions*, Computers & Mathematics with Applications, 49 (2005), pp. 103–130.

[63] Le Quéré, P. , *Accurate Solutions to the Square Thermally Driven Cavity at High Rayleigh Number*, Computers & Fluids, 20 (1991), pp. 29–41.

[64] C. Lee, *A new finite point generation scheme using metric specifications*, International Journal for Numerical Methods in Engineering, 48 (2000), pp. 1423–1444.

[65] C. Lee, X. Liu, and S. Fan, *Local Multiquadric Approximation for Solving Boundary Value Problems*, Computational Mechanics, 30 (2003), pp. 396–409.

[66] K. Leem, S. Oliveira, and D. Stewart, *Algebraic multigrid (AMG) for saddle point systems from meshfree discretizations*, Numerical Linear Algebra with Applications, 11 (2004), pp. 293–308.

[67] R. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in Mathematics. ETH Zürich, Birkhäuser, Basel, 2nd ed., 1992.

[68] H. Li and S. Mulay, *Meshless Methods and Their Numerical Properties*, CRC Press, Boca Raton, Florida, 2013.

[69] T. Liszka, C. Duarte, and W. Tworzydlo, *hp-meshless cloud method*, Computer Methods in Applied Mechanics and Engineering, 139 (1996), pp. 263 – 288.

[70] T. Liszka and J. Orkisz, *The finite difference method at arbitrary irregular grids and its application in applied mechanics*, Computers & Structures, 11 (1980), pp. 83 – 95. Special Issue-Computational Methods in Nonlinear Mechanics.

[71] G. Liu, *Meshfree Methods: Moving Beyond the Finite Element Method*, CRC Press, Boca Raton, Florida, 2009.

[72] W. K. Liu, S. Jun, and Y. F. Zhang, *Reproducing kernel particle methods*, International Journal for Numerical Methods in Fluids, 20 (1995), pp. 1081–1106.

[73] R. Loehner and E. Oñate, *An advancing front point generation technique*, Communications in Numerical Methods in Engineering, 14 (1998), pp. 1097–1108.

[74] ——, *A general advancing front technique for filling space with arbitrary objects*, International Journal for Numerical Methods in Engineering, 61 (2004), pp. 1977–1991.

[75] D. Martín, G. Arroyo, A. Rodríguez, and T. Isenberg, *A survey of digital stippling*, Computers & Graphics, 67 (2017), pp. 24 – 44.

[76] B. Mavrič, *Meshless modeling of thermo-mechanics of low-frequency electromagnetic direct chill casting*, PhD thesis, University of Nova Gorica, Graduate School, 2017.

[77] B. Mavrič and B. Šarler, *Local radial basis function collocation method for linear thermoelasticity in two dimensions*, International Journal of Numerical Methods for Heat & Fluid Flow, 25 (2015), pp. 1488–1510.

[78] S. McCormick, *Multigrid Methods*, Frontiers in Applied Mathematics. , SIAM, Philadelphia, Pennsylvania, 1987.

[79] E. Mitteff, *Automated adaptive data center generation for meshless methods*, master's thesis, Orlando, Florida, 2006.

[80] E. Nobile, *Simulation of Time-dependent Flow in Cavities with the Additive-correction Multigrid Method, part II: Applications*, Numerical Heat Trasfer, Part B: Fundamentals, 30 (1996), pp. 351–370.

[81] E. Oñate, S. Idelsohn, O. C. Zienkiewicz, and R. L. Taylor, *A finite point method in computational mechanics. applications to convective transport and fluid flow*, International Journal for Numerical Methods in Engineering, 39 (1996), pp. 3839–3866.

[82] S. Owen, *A survey of unstructured mesh generation technology*, in 7th International Meshing Roundtable, 1998, pp. 239–267.

[83] S. Paolucci and D. R. Chenoweth, *Transition to Chaos in a Differentially Heated Vertical Cavity*, Journal of Fluid Mechanics, 201 (1989), pp. 379–410.

[84] N. Perrone and R. Kao, *A general finite difference method for arbitrary meshes*, Computers & Structures, 5 (1975), pp. 45 – 57.

[85] Y. Saad, *Iterative Methods for Sparse Linear Systems* , SIAM, Philadelphia, Pennsylvania, 2nd ed., 2003, ch. 10: Preconditioning Techniques.

[86] H. Samet, *The quadtree and related hierarchical data structures*, ACM Computing Surveys, 16 (1984), pp. 187–260.

[87] ——, *An overview of quadtrees, octrees, and related hierarchical data structures*, in Theoretical Foundations of Computer Graphics and CAD, R. A. Earnshaw, ed., Berlin, Heidelberg, 1988, Springer Berlin Heidelberg, pp. 51–68.

[88] B. Šarler, *From Global to Local Radial Basis Function Collocation Method for Transport Phenomena*, in Advances in Meshfree Techniques, V. Leitão, C. Alves, and C. Duarte, eds., Computational Methods in Applied Sciences, vol. 5, Springer, Dordrecht, Netherlands, 2007, pp. 257–282.

[89] B. Šarler and R. Vertnik, *Meshfree Explicit Local Radial Basis Function Collocation Method for Diffusion Problems*, Computers and Mathematics with Applications, 51 (2006), pp. 1269–1282.

[90] B. Seibold, *Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods*, Numerical Linear Algebra with Applications, 17 (2009), pp. 433–451.

[91] A. Settari and K. Aziz, *A generalization of the additive correction methods for the iterative solution of matrix equations*, SIAM Journal of Numerical Analysis, 10 (1973), pp. 506–521.

[92] R. Southwell, *Relaxation Methods in Theoretical Physics*, The Oxford engineering science series. , Oxford University Press, London, 1946.

[93] E. Stiefel, *Über einige methoden der relaxationsrechnung*, Zeitschrift fur Angewandte Mathematik Und Physik, 3 (1952), pp. 1–33.

[94] K. Stüben, *Algebraic multigrid (AMG): an introduction with applications (Technical Report)*, Tech. Report GMD-70, Institute for Algorithms and Scientic Computing (SCAI), German National Research Center for Information Technology (GMD). , Sankt Augustin, Germany, 1999.

[95] ——, *A review of algebraic multigrid*, Journal of Computational and Applied Math, 128 (2001), pp. 281–309.

[96] R. A. Ulichney, *Review of halftoning techniques*, 1999.

[97] H. van der Vorst, *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 631–644.

[98] P. Vaněk, *Acceleration of convergence of a two-level algorithm by smoothing transfer operators*, Applications of Mathematics, 37 (1992), pp. 265–274.

[99] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.

[100] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid on unstructured meshes*, tech. report, University of Colorado at Denver, Denver, CO, USA, 1994.

[101] ——, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[102] M. U. Varma, R. S. Rao, and S. Deshpande, *Point distribution generation using hierarchical data structures*, ECCOMAS 2004, 2004, pp. 1–6.

[103] O. Vlasiuk, T. Michaels, N. Flyer, and B. Fornberg, *Fast high-dimensional node generation with variable density*, Computers & Mathematics with Applications, 76 (2018), pp. 1739 – 1757.

[104] Z. Wang, Z. Huang, W. Zhang, and G. Xi, *A Meshless Local Radial Basis Function Method for Two-Dimensional Incompressible Navier-Stokes Equations*, Numerical Heat Transfer, Part B: Fundamentals, 67 (2015), pp. 320–337.

[105] J. Waters and D. Pepper, *Global Versus Localized RBF Meshless Methods for Solving Incompressible Fluid Flow with Heat Transfer*, Numerical Heat Transfer, Part B: Fundamentals, 68 (2015), pp. 185–203.

[106] G. B. Wright and B. Fornberg, *Stable computations with flat radial basis functions using vector-valued rational approximations*, Journal of Computational Physics, 331 (2017), pp. 137 – 156.

[107] G. Yao, Siraj-ul-Islam, and B. Šarler, *Assessment of Global and Local Meshless Methods Based on Collocation with Radial Basis Functions for Parabolic Partial Differential Equations in Three Dimensions*, Engineering Analysis with Boundary Elements, 36 (2015), pp. 1640–1648.

[108] D. Yun and Y. Hon, *Improved localized radial basis function collocation method for multi-dimensional convection-dominated problems*, Engineering Analysis with Boundary Elements, 67 (2016), pp. 63 – 80.

[109] 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 (2018), pp. 4305–4321.

[110] O. C. Zienkiewicz, *The finite element method*, McGraw-Hill London, New York, 3rd expanded and rev. ed., 1977.

[111] L. Zovatto and G. Pedrizzetti, *Flow About a Circular Cylinder Between Parallel Walls*, Journal of Fluid Mechanics, 440 (2001), pp. 1–25.