UNIVERSITY OF TRIESTE

Department of Engineering and Architecture

Master's Degree in
Computer & Electronic Engineering

Shape optimization with discrete adjoint method applied to RBF-FD meshless method

May 20, 2024

Candidate
Kevin Marzio

Supervisor
Prof. Andrea De Lorenzo

Co-supervisors
Dr. Mauro Munerato
Dr. Riccardo Zamolo
Dr. Davide Miotti

A.Y. 2023/2024

Lorem ipsum
dolor sit amet

- Cicero

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

Introduction

Spiegazione/storia mesh-based.
Perché siamo arrivati ai meshless e differenze in generale
Perché diff. automatica e aggiunto nel meshless

Chapter 1

Meshless methods

Meshless or Meshfree Methods (MMs) were developed to overcome the drawbacks of traditional mesh-based methods for the solution of Partial Differential Equations (PDE), where their true advantage is that "the approximation of unknowns in the PDE is constructed based on scattered points without mesh connectivity" [1].

They appeared for the first time in 1977 with the Smooth Particle Hydrodynamics (SPH) method [2], initially used to modeling astrophysical phenomena such as exploding stars and dust clouds was later applied in solid mechanics to overcome limitations of mesh-based methods [3]. To improve accuracy, tensile instability and spatial instability of SPH many more modern MMs have been developed: the introduction of Reproducing Kernel Particle Method (RKPM) [4] is a prime example of enhanced consistency and stability. Generalized Finite Difference (GFD) methods is another branch of numerical methods for solving PDEs that do not rely on a grid structure and many modern MMs originate from the employment of this approximation for solving PDEs.

The typical use case for MMs is the construction of an approximating field u^h for the sought solution $u\colon \Omega \subset \mathbb{R}^d \to \mathbb{R}$ of the following boundary value problem:

$$\begin{cases} \mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ \mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}) & \text{on } \partial\Omega \end{cases} \tag{1.0.1}$$

where $\mathcal{L}$ is a generic linear differential operator and $\mathcal{B}$ is a different linear operator used to enforce some Boundary Condition (BC) that does not necessarily involves partial derivatives. f and g are known functions. Usually in the Computational

Fluid Mechanics (CFD) world the encountered BC are:

$$\text{Dirichlet BC:} \qquad u = g \tag{1.0.2}$$

$$\text{Neumann BC:} \qquad \frac{\partial u}{\partial \boldsymbol{n}} = g \tag{1.0.3}$$

$$\text{Robin BC:} \qquad au + b\frac{\partial u}{\partial \boldsymbol{n}} = g \tag{1.0.4}$$

where $\frac{\partial u}{\partial \boldsymbol{n}}$ indicate the normal derivative of u whereas a and b are known functions. It can also be noticed that Dirichlet and Neumann BCs are special cases of Robin BC respectively when $b(\boldsymbol{x}) = 0$ and $a(\boldsymbol{x}) = 0$.

Regardless on the Meshfree approach, the following approximation for any solution u can be written:

$$u^h(\boldsymbol{x}) = \sum_{k=1}^{N} \alpha_k B_k(\boldsymbol{x}) \tag{1.0.5}$$

where $B_k \colon \Omega \to \mathbb{R}$ are suitable basis functions and α_k are the expansions coefficients that must be determined. Different choices for the basis functions B_k leads to different formulations and in literature can be found several of these, some examples are: RKPM [4], moving least square (MLS) [5], radial basis function (RBF) [6, 7] and partition of unity (PU) [8]. Furthermore solving the PDE in (1.0.1) with the approximated solution u^h, in general, yields to a non-zero error function ϵ^h given by:

$$\epsilon^h(\boldsymbol{x}) = \mathcal{L}u^h(\boldsymbol{x}) - f(\boldsymbol{x}) \tag{1.0.6}$$

Once the general form of the approximated solution in (1.0.5) has been properly defined, it can be employed for discretizing the PDE, reported in (1.0.1), over a set of N generated nodes $\{ \boldsymbol{x}_1, \dots, \boldsymbol{x}_N \}$ distributed in the physical domain $\Omega \cup \partial\Omega$. The Weighted Residual Method is used to do so: a set of test functions $\{ \Gamma_1, \dots, \Gamma_N \}$ orthogonal to ϵ^h are used to integrate the error to zero:

$$\begin{aligned} \int_\Omega \Gamma_i \epsilon^h \, d\Omega &= \int_\Omega \Gamma_i (\mathcal{L}u^h(\boldsymbol{x}) - f(\boldsymbol{x})) \, d\Omega \\ &= \int_\Omega \Gamma_i \Big[\mathcal{L}\Big(\sum_{k=1}^{N} \alpha_k B_k(\boldsymbol{x}) \Big) - f(\boldsymbol{x}) \Big] \, d\Omega = 0 \end{aligned} \tag{1.0.7}$$

where $i = 1, \dots, N$. From the choice of functions Γ_i the following two formulations are obtained [1]:

Galerkin Meshless Methods that find a weak solution for the PDE by using as test functions the basis functions B_k. These formulations require domain integration and special techniques to enforce boundary conditions;

Collocation Meshless Methods that find a strong solution for the PDE by using Dirac delta functions centered at the discretization nodes as test functions. Basically they enforce equaitons (1.0.1) on a finite set of nodes, and by doing so, they do not require any domain integration.

Finally, once constraints (1.0.7) are enforced, the solution of problem (1.0.1), i.e. the values $\{ u^h(\boldsymbol{x}_1), \dots, u^h(\boldsymbol{x}_N) \}$, can be found solving the linear system resulting from PDE's discretization. We remark that the system obtained is not linear in general, but it is in this case since the PDE is linear. Nevertheless, also non-linear PDEs can be reduced to the aforementioned case thanks to a proper linearization.

1.1 Motivation of case study

In this section we motivate the specific case that will be studied in the following chapters: the generic heat equation with internal heat generation [9] at steady state given by

$$\begin{cases} -\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ u(\boldsymbol{x}) = g(\boldsymbol{x}) & \text{on } \partial\Omega \end{cases} \tag{1.1.1}$$

The reason behind this choice are due to both its wide range of applications in different areas (e.g. electrostatic, chemistry, gravitation and others) and the simplicity of its analytical manipulation. Equation (1.1.1) is one of the many PDEs that can be solved using the meshless approaches.

In particular for this thesis we will:

- consider domains $\Omega \subset \mathbb{R}^d$ with $d = 1, 3$;
- use RBFs augmented with polynomial terms as basis functions B_k to approximate the solution u in a similar way to what has been done in [10];
- formulate the problem in its strong form, so we will use the collocation technique in combination with the Weighted Residual Method.

thus we will apply the RBF-generated Finite Differences (RBF-FD) [11, 12] to solve (1.1.1) in one and three dimensional physics domain.

Mono dimensional physics domain are used to gain confidence with the techniques explained while three dimensional domain are used for their application in real case problems as those faced at Esteco. 2D cases, instead, have been skipped in the analysis since they would only be used to bridge the 2 previous scenarios and therefore would have had no practical utility.

RBFs are used for scattered data interpolation both for their physical foundation[overview 20, Hardy 1990] and for their profitable use in applications like meteorology, turbulence analysis and neural network [1]

Finally, collocation technique is employed because thanks to its ability to discretize the Boundary Value problem (1.1.1) expressed in strong form leads to a real fully meshless approach [13].

Chapter 2

RBF-FD method

In this chapter we explain in more details the Meshless Method (MM) used within this work: the Radial Basis Function generated Finite Differences (RBF-FD) method. To do so we show how it can be used to solve the general boundary value problem defined in chapter 1 on page 1. From now on, in order to be able to discretize the PDE, we consider to have a disposal a set of N distinct nodes, $\mathcal{X}$, defined as follow:

$$\mathcal{X} := \{ \boldsymbol{x}_1, \dots, \boldsymbol{x} \mid \boldsymbol{x}_i \in \Omega \cup \partial\Omega,\, i = 1, \dots, N \} \tag{2.0.1}$$

where $\Omega \cup \partial\Omega \subset \mathbb{R}^d$ is the physical domain of the problem, Ω and $\partial\Omega$ are respectively its open subset and boundary, and $d \in \mathbb{N}$ its dimension.

2.1 Scattered Data Interpolation

Every MM, as we have already seen, on its core, is just a way to approximate the solution of a PDE and RBF-FD method is no exception; the tool used to do so is called *scattered data interpolation*. In this section we first see what scattered data interpolation is and then how it is related to PDEs solution. In the second part of the explanation we see how it is applied in general by MMs so to avoid to becoming fixated on a single implementation and losing generality; moreover this approach is still useful since it establishes all the steps that are also followed by RBF-FD. To avoid from the very beginning any kind of ambiguity we underline that scattered data interpolation, is inherently connected to data fitting and not to PDEs approximation, consequently, it finds many other applications beyond the realm of MMs. An example can be found in [14].

In general the interpolation problem has the following, simple, formulation. Given:

- a finite set of nodes, $\mathcal{X} \subset \mathbb{R}^d$, which could be the one reported in (2.0.1) and;

- a set of known real values $u(\boldsymbol{x}_1), \dots, u(\boldsymbol{x}_N)$, which may be obtained from a function

we want to find a continuous function $u^h \colon \Omega \cup \partial\Omega \subset \mathbb{R}^d \to \mathbb{R}$ that satisfy:

$$u^h(\boldsymbol{x_i}) = u(\boldsymbol{x}_i) \qquad \forall \boldsymbol{x}_i \in \mathcal{X} \tag{2.1.1}$$

If the locations, the nodes in $\mathcal{X}$ where the measurements $u(\boldsymbol{x}_1), \dots, u(\boldsymbol{x}_N)$ are taken, are placed on a uniform or regular grid we talk about interpolation, otherwise the process above is called *scattered* data interpolation. Here the main idea is to find a function u^h which is a "good" fit to the given data, where with "good" we mean a function that exactly match the given measurements at the corresponding locations.

MMs also aim to provide an approximation for an unknown function defined as a linear combination of a set of basis functions as reported in equation (1.0.5); we report here for clarity its definition:

$$u^h(x) = \sum_{k=1}^{N} \alpha_k B_k(\boldsymbol{x}) \tag{2.1.2}$$

and we remark that coefficients α_k are unknown. Here is where the theory of scattered data interpolation is applied: it tells us that we are able to find the numerical values for $\alpha_1, \dots \alpha_N$ if we impose a number of conditions equal to the number of coefficients that we are looking for. These conditions must have a form like that shown in equation (2.1.1). Therefore if we replace the generic meshless approximation within each of the N interpolation conditions we can write the following linear system:

$$\underbrace{\begin{bmatrix} B_1(\boldsymbol{x}_1) & \dots & B_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ B_1(\boldsymbol{x}_N) & \dots & B_N(\boldsymbol{x}_N) \end{bmatrix}}_{\boldsymbol{B}} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} = \begin{bmatrix} u(\boldsymbol{x}_1) \\ \vdots \\ u(\boldsymbol{x}_N) \end{bmatrix} \tag{2.1.3}$$

that, once solved, provides us the desired coefficients that uniquely define u^h given the set of basis functions $\{ B_1, \dots, B_N \}$. We would like to stress the fact that, however, the right hand side vector is made up of values of the unknown exact solution of problem (1.0.1).

During the solution of problem (1.0.1) the following constraints are enforced instead:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_j) &= f(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \Omega \\ u^h(\boldsymbol{x}_j) &= g(\boldsymbol{x}_j) \quad \text{if } \boldsymbol{x}_j \in \partial\Omega \end{aligned} \tag{2.1.4}$$

and if we arrange the nodes such that the first N_I nodes belongs to Ω and the last N_B to $\partial\Omega$, we can find the values of the approximated solution at the points in $\mathcal{X}$ by solving:

$$\begin{bmatrix} c_{1,1} & \dots & c_{1,N_I} \\ \vdots & \ddots & \vdots \\ c_{N_I,1} & \dots & c_{N_I,N_I} \end{bmatrix} \begin{bmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{bmatrix} = \boldsymbol{f} - \begin{bmatrix} c_{1,N_I+1} & \dots & c_{1,N_B} \\ \vdots & \ddots & \vdots \\ c_{N_I,N_I+1} & \dots & c_{N_I,N_B} \end{bmatrix} \boldsymbol{g} \tag{2.1.5}$$

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

$$\begin{bmatrix} c_{1,1} & \dots & c_{N_I,1} \\ \vdots & \ddots & \vdots \\ c_{1,N} & \dots & c_{N_I,N} \end{bmatrix} = \boldsymbol{B}^{-T} \begin{bmatrix} \mathcal{L}B_1(\boldsymbol{x}_1) & \dots & \mathcal{L}B_1(\boldsymbol{x}_{N_I}) \\ \vdots & \ddots & \vdots \\ \mathcal{L}B_N(\boldsymbol{x}_1) & \dots & \mathcal{L}B_N(\boldsymbol{x}_{N_I}) \end{bmatrix} \tag{2.1.6}$$

Analyzing each row $\boldsymbol{c}_i = [c_{i,1}, \dots c_{i,N}]$ of matrix $\boldsymbol{C}$ we can notice that are computed solving the following linear systems:

$$\boldsymbol{B}^T c_i = \begin{bmatrix} \mathcal{L}B_1(\boldsymbol{x_i}) \\ \vdots \\ \mathcal{L}B_N(\boldsymbol{x_i}) \end{bmatrix} \qquad i = 1, \dots, N_I \tag{2.1.7}$$

which are closely related to the ones obtained from scattered data interpolation reported in (2.1.3) due to the presence of the same matrix $\boldsymbol{B}$. We conclude by commenting that equation (2.1.6) with matrix $\boldsymbol{B}$ defined as in (2.1.3) holds true only in case of Dirichlet boundary conditions, otherwise $\boldsymbol{B}$ would take on a different form.

2.2 The Mairhuber-Curtis theorem

From the previous discussion, in particular from equation (2.1.6), it can be noticed that matrix $\boldsymbol{B}$ has to be non singular in order to be able to solve the linear system associated to the boundary value problem, and this must hold for each node placement $\mathcal{X}$ (to be read as every possible discretization of the problem domain) as long nodes are distinct. This property of $\boldsymbol{B}$ turns out to be dependent on the choice of the particular set of basis functions: for example if we assume $B_k(\boldsymbol{x}) \in \Pi_P^d$ and $\{ B_1(\boldsymbol{x}), \dots, B_N(\boldsymbol{x}) \}$ to be a polynomial basis of the space Π_P^d of polynomials of degree at most P in $\mathbb{R}^d$, then we are not able to guarantee that $\boldsymbol{B}$ is invertible for $d > 1$.

This issue is explained in more detail by the Mairhuber-Curtis theorem [15]; when dealing with the multidimensional case it is possible to continuously move

two nodes along a closed path P, that does not interfere with any other node in $\mathcal{X}$, such that they end up by interchanging their original positions without one crossing the path of the other. In the event that these two are the only nodes of $\mathcal{X}$ that are moved, $\boldsymbol{B}$ ends up with two rows exchanged leading to a change in the sign of its determinant, and, since the determinant is a continuous function, this means that there is a moment when the latter vanishes making the matrix singular.

The inconvenience arises from the fact that the set of basis functions is independent from the node position and could be solved by simply choosing a basis that is function of nodes position. By doing so we no more fall in the case of the Mairhuber-Curtis theorem since whenever we move nodes also the base itself changes and if two nodes switches their positions not only their respective rows in $\boldsymbol{B}$ are switched, but also their columns, forcing the determinant not to change in sign.

2.3 Radial Basis Functions

Up to now, when talking about the approximated solution of the PDE, u^h, we have not specified the type of basis functions which define it. However these must be done in order to be able to find the coefficients α_k in equation (2.1.2) and thus its numerical values; furthermore it would be appropriate to select a set of functions that allow the avoidance of the aforementioned Mairhuber-Curtis' theorem case. In this section we will define the ones used by the RBF-FD method: the Radial Basis Functions (RBFs).

RBFs are defined as:

$$\Phi(\boldsymbol{x}, \boldsymbol{x}_k) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_k\|_2) \tag{2.3.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^d$ is a given and known point , $\|\cdot\|_2$ is the euclidean distance and $\varphi\colon \mathbb{R} \to \mathbb{R}$, named *basic* function, is a (univariate) function which takes the radius $r_k = \|\boldsymbol{x} - \boldsymbol{x}_k\|_2$ as input and it is used as generator for all the (multivariate) *basis* functions associated to different $\boldsymbol{x}_k$. In general different basic functions φ can be used, some of them are reported in table 2.1. Sometimes the notations $\Phi(\boldsymbol{x} - \boldsymbol{x}_k)$ or $\Phi_k(\boldsymbol{x})$ are also used instead of $\Phi(\cdot, \boldsymbol{x}_k)$.

To further clarify the RBFs name we can notice that they are called:

Radial since the value of each $\Phi(\cdot, \boldsymbol{x}_k)$ at each point $\boldsymbol{x}$ depends only on the distance between that point and $\boldsymbol{x}_k$ through $\|\cdot\|_2$, they satisfy radial symmetry, i.e. $\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_j, \boldsymbol{x}_i)$ for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$;

Basis since in case of a set of N distinct nodes in $\mathbb{R}^d$, as can be $\mathcal{X}$ given in (2.0.1), the set of radial functions $\Phi_k(\boldsymbol{x})$ with $k = 1, \dots, N$ associated to each point

Table 2.1: Examples of basic functions where r is a real number greater than or equal to zero, and ϵ, called shape factor, is a suitable parameter

Name	$\varphi(r)$
Multiquadratic	$\sqrt{1+(\epsilon r)^2}$
Inverse multiquadratic	$\left(\sqrt{1+(\epsilon r)^2}\right)^{-1}$
Thin plate splines	$r^{2l}\log l,\ l \in \mathbb{N}$
Gaussian	$e^{-(\epsilon r)^2}$
Polyharmonics	$r^{2l+1},\ l \in \mathbb{N}$

of the set, form a basis for the space of functions:

$$F_\Phi := \left\{ \sum_{k=1}^{N} \alpha_k \Phi_k(\boldsymbol{x}), \quad \alpha_k \in \mathbb{R},\ \boldsymbol{x}_k \in \mathcal{X} \right\}$$

In case of RBF-FD method the implementation of scattered data interpolation and the solution of governing equation remain the same as explained in the previous section and leads to a symmetric matrix $\boldsymbol{B}$ defined as:

$$\boldsymbol{B} = \begin{bmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \Phi(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \tag{2.3.2}$$

We conclude by observing that this functions are particularly convenient since they depend on nodes position through $\boldsymbol{x}_k$, thus they allow to avoid the non-invertibility of matrix $\boldsymbol{B}$ in case of singular node arrangements (Mairhuber-Curtis theorem).

2.4 Polynomial augmentation

Setting aside the issue of the invertibility of matrix $\boldsymbol{B}$ in case of particular nodes arrangement discussed in previous subsection, we should also take into account the accuracy of the interpolation that we are able to achieve, which also depends upon the type of functions that we are supposed to approximate. Indeed RBFs approximation schemes alone are not able to exactly interpolate (i.e. with an accuracy only depending on roud-off errors) constant, linear or higher degree polynomials fields. This is an issue in different important engineering applications such as modeling of constant strain in elastic bodies and steady temperature fields in differentially heated walls [16].

To overcome this limitation, a polynomial augmentation of degree P is required, leading to the overall formulation for the RBF interpolant:

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \Phi_j(\boldsymbol{x}) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \tag{2.4.1}$$

where $M = \binom{P+D}{D}$ is the number of polynomial basis functions with degree $P \leq D$, $\{\, p_1(\boldsymbol{x}) \dots p_M(\boldsymbol{x}) \,\}$ is a complete polynomial basis of Π_P^d and β_j are the corresponding coefficients. An example of polynomial basis for polynomials of degree $P = 1$ in $2D$ has the following $M = 3$ elements: $p_1(x, y) = 1$, $p_2(x, y) = x$, $p_3(x, y) = y$.

We must also note that using an interpolant with the introduced polynomial augmentation leads to an underdetermined system in (2.1.3). In order to obtain a square $\boldsymbol{B}$ and thus having a solvable system, the following orthogonality conditions have to be imposed:

$$\sum_{i=1}^{N} \alpha_i p_k(\boldsymbol{x}_i) = 0, \qquad i = 1, \dots, M \tag{2.4.2}$$

The coefficients of u^h, which are now composed not only by $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_N]$, but also by $\boldsymbol{\beta} = [\beta_1 \dots \beta_M]$, can then be found by solving the following system:

$$\underbrace{\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} \tag{2.4.3}$$

where:

$$\begin{aligned} \boldsymbol{P} &= \begin{bmatrix} p_0(\boldsymbol{x}_1) & \dots & p_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_0(\boldsymbol{x}_N) & \dots & p_M(\boldsymbol{x}_N) \end{bmatrix} \\ \boldsymbol{u} &= [u(\boldsymbol{x}_1) \dots . \, u(\boldsymbol{x}_N)] \end{aligned} \tag{2.4.4}$$

From its formulation is easy to understand that the system above is simply the composition of the system in equation (2.1.3), in the first row, with constraints (2.4.2) written in compact form, in the second row.

In practice the addition of polynomial basis to the RBF interpolant let us perfectly fit $u(\boldsymbol{x})$ not only on collocation points where we have $u^h(\boldsymbol{x}_i) = u(\boldsymbol{x}_i)$, but also across the rest of the domain provided that data $u(\boldsymbol{x}_1) \dots u(\boldsymbol{x}_N)$ come from a polynomial of total degree less than or equal to P. Nevertheless this procedure has as a side effect since not all set of nodes $\mathcal{X}$ nor all basic functions φ leads to a well-posed RBF interpolation with a non singular matrix M. We will discuss this limitation in more detail in the next section.

2.5 Problem solution

At the beginning of section 2.2 we mentioned that the system in equation (2.1.6) might not be solvable in case of a combination of non-point-dependent basis functions and particular nodes arrangements, but, up to now, we did not discuss in general in which cases the interpolation problem is solvable. In this section we will address this shortcoming.

We start by recalling that, in case of pure RBF interpolant, the system that we aim to solve has the following compact form:

$$\boldsymbol{B}\boldsymbol{\alpha} = \boldsymbol{u} \tag{2.5.1}$$

where $\boldsymbol{B}$ is a symmetric matrix. This means that the above system require a positive definite $\boldsymbol{B}$ in order to be solved and this property depends on the choice of the basic function φ used to define the RBFs. By definition only *strictly* positive definite [17] φ are associated to a positive definite $\boldsymbol{B}$ and, this, restrict our choices: of those shown in table 2.1 on page 9 only the Inverse Multiquadratic satisfy this requirement. The same attributes of φ are also inherited by the associated RBFs Φ_k

However in previous section we have seen that, beyond the solvability issue, a polynomial augmentation of degree P is beneficial for the accuracy of the interpolant u^h. This means that the system that we have to solve, in general, is no more in the form shown in equation (2.5.1), but rather in the following:

$$\boldsymbol{M}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{0}\end{bmatrix} \tag{2.5.2}$$

In this case the matrix that has to be positive definite would be $\boldsymbol{M}$, which is symmetric as well, that we recall being defined as:

$$\boldsymbol{M} = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{P}\\ \boldsymbol{P}^T & \boldsymbol{0}\end{bmatrix} \tag{2.5.3}$$

To be such the following conditions have to be met [18]:

1. basic function φ has to be *strictly conditionally* positive definite function of order P [17];
2. matrix $\boldsymbol{P}$ has to be full-rank.

Condition 1 allows a greater freedom on the choice of the basic function, compared to the pure RBF interpolant: in fact, the set of strictly conditionally positive definite functions of order P is a superset of strictly positive definite functions.

These functions are defined as those that require a polynomial augmentation of order at least $P-1$ in order to give a non singular $\boldsymbol{M}$. Strictly conditionally positive definite functions of order P are also strictly conditionally positive definite of any higher order. This means that, in case u^h include a polynomial augmentation of order 1, we can also use Multiquadratic, Thin plate splines with $l=0$ and Polyharmonics with $l=1$ (refer again to table 2.1 for their definitions) as basic funtions as these are strictly conditionally definite of order 1. At this point someone might consider increasing the degree P of the polynomial augmentation up to the theoretical limit for the size of matrix $\boldsymbol{P}$, i.e. $M=N$, in order to increase the accuracy of the interpolant, but doing so results in ill-conditioning and singularity issues related to $\boldsymbol{P}$ that, since non-singularity of $\boldsymbol{P}$ is a necessary condition for the non singularity of $\boldsymbol{M}$, also affect $\boldsymbol{M}$.

This is why condition 2 is also required. It can be shown that to have a full-rank $\boldsymbol{P}$ the set $\mathcal{X}$, containing the nodes respect to which we are carrying out the interpolation, must be P-unisolvent [17], where P is the degree of the polynomial augmentation. This dependency of $\boldsymbol{P}$'s rank on the node locations should not surprise since the matrix columns consist on the elements of the polynomial basis evaluated at the different points in $\mathcal{X}$, and they are required to be linearly independent.

Given the node generation technique used in this work a safe rule for a stable implementation (in the sense of a well-posed interpolation probelm) of the polynomial augmentation is to respect the inequality $2M \leq N$ where M is the number of terms in the polynomial basis and N is the number of nodes in $\mathcal{X}$.

2.6 RBF-FD

In this section we are going to discuss in detail how RBF-FD meshless method is used to solve Partial Differential Equations (PDEs). Its first implementation was developed by Kansa in [6, 7]. His approach, that we denote as *global method*, due to its computational inefficiencies (that we are going to reference later on this section), was eventually dropped in preference for the one suggested by Tolstykh in [19]. We will denote the latter as *local method*. In recent years RBF-FD local method has been developed and applied with success [20, 21, 22, 23].

The goal of both methods is to approximate solutions to PDEs, i.e., to find a function (or some discrete approximation to this function) which satisfies a given relationship between various of its derivatives on some given region of space along with some boundary conditions along the edges of this domain. In most cases an analytic formula for the solution can not be found. What RBF-FD methods do is replacing derivatives in the differential equation by Finite Difference approximations in such a way as to obtain a large algebraic system of equations to be solved in

place of the differential equation; this could be easily solved with a computer.

2.6.1 Finite difference method

Before tackling the approximation of PDEs solution, we first consider the more basic task of approximating the derivatives of a known function by Finite Difference (FD) formulas based only on values of the function itself at discrete points. Given u, in the simplest case a function of one variable assumed to be sufficiently smooth, we want to approximate its derivatives at a given point $\overline{x}$ relying solely on its values at a finite number of points close to $\overline{x}$. In general its k-th derivative is approximated by the following FD formula:

$$\left.\frac{d^k u}{dx^k}\right|_{x=\overline{x}} = u^{(k)}(\overline{x}) \approx \sum_{i=1}^{n} c_i^k u(x_i) \tag{2.6.1}$$

where $\{\, u(x_1), \ldots, u(x_n) \,\}$ are the function's samples and $\{\, c_1^k, \ldots, c_n^k \,\}$, which can be computed in different ways such as the method of undetermined coefficients or via polynomial interpolation [24], are called FD weights.

To give a concrete example we could approximate $u'(\overline{x})$ with the following one-sided approximations:

$$D_+ u(\overline{x}) = \frac{u(\overline{x}+h) - u(\overline{x})}{h} \tag{2.6.2a}$$

$$D_- u(\overline{x}) = \frac{u(\overline{x}) - u(\overline{x}-h)}{h} \tag{2.6.2b}$$

for some value of h. This is motivated by the standard definition of the derivative as the limiting value of this expression as $h \to 0$. In these cases the same FD weights, $\{\, 1/h, -1/h \,\}$, are associated to function values coming from different discrete points: $\{\, u(\overline{x}+h), u(\overline{x}) \,\}$ for equation (2.6.2a) and $\{\, u(\overline{x}), u(\overline{x}-h) \,\}$ for (2.6.2b). Another possibility is to use the centered approximation:

$$D_0 u(\overline{x}) = \frac{u(\overline{x}+h) - u(\overline{x}-h)}{h} = \frac{1}{2}\big(D_+ u(\overline{x}) + D_- u(\overline{x})\big) \tag{2.6.3}$$

To derive approximations to higher order derivatives, besides the two method mentioned above, is also possible to repeatedly apply first order differences. Just as the second order derivatives is the derivative of u', we can view $D^2 u(\overline{x})$, the second order derivative approximant, as being a finite difference of first differences: $D^2 u(\overline{x}) = D_+ D_- u(\overline{x})$ or $D^2 u(\overline{x}) = D_- D_+ u(\overline{x})$. If we use a step size $h/2$ in each centered approximation to the first derivative we could also define $D^2 u(\overline{x})$ as a centeered difference of centered differences and obtain:

$$D^2 u(\overline{x}) = \frac{1}{h}\left(\left(\frac{u(\overline{x}+h) - u(\overline{x})}{h}\right) - \left(\frac{u(\overline{x}) - u(\overline{x}-h)}{h}\right)\right) \tag{2.6.4}$$

where FD weights $\{ 1/h^2, -2/h^2, 1/h^2 \}$ are associated to $\{ u(\overline{x}+h), u(\overline{x}), u(\overline{x}-h) \}$

2.6.2 Formulation

Now that we have gained a certain level of familiarity with Finite Difference (FD) methods we can see how they are generalized to PDEs differential operators by RBF-FD in order to obtain easily solvable linear systems.

We start by recalling that the boundary value problem that we have to solve is defined as:

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega \\ \mathcal{B}u = g & \text{on } \partial\Omega \end{cases} \tag{2.6.5}$$

where $\mathcal{L}$ and $\mathcal{B}$ are linear operators and its solution u could be approximated by a function u^h defined as reported in equation (2.4.1):

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \Phi_j(\boldsymbol{x}) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}) \tag{2.6.6}$$

which holds true all over the domain. Coefficients of the expansions are found by solving equation (2.4.3):

$$\underbrace{\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} \tag{2.6.7}$$

In order to streamline the explanation of the procedure we split the set of nodes $\mathcal{X}$ reported in (2.0.1) into the two following sets:

$$\mathcal{X}_I := \{ \boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_I} \mid \boldsymbol{x}_i \in \Omega,\ i = 1, \dots, N_I \} \tag{2.6.8a}$$

$$\mathcal{X}_B := \{ \boldsymbol{x}_{N_I+1}, \dots, \boldsymbol{x}_{N_B} \mid \boldsymbol{x}_i \in \partial\Omega,\ i = N_I+1, \dots, N_B \} \tag{2.6.8b}$$

where N_I and N_B indicate respectively the number of nodes inside and on the boundary of the physical domain. Of course $N_I + N_B = N$ and $\mathcal{X}_I \cup \mathcal{X}_B = \mathcal{X}$.

The idea that Kansa used to discretize the problem reported above is to use the theory of interpolation to approximate the differential operator $\mathcal{L}$ in the PDE, with an operator $\mathcal{L}^h$ represented by a matrix $\boldsymbol{C}_I$, using a FD-like method. Tolstykh's approach follows the same concept as Kansa's, but adds *stencil* as novelty. A stencil of m nodes is associated to each node $\boldsymbol{x}_i \in \mathcal{X}_I$ and it is defined as the set $\mathcal{X}_i = \{ \boldsymbol{x}_1, \dots, \boldsymbol{x}_m \} \subseteq \mathcal{X}$ formed by the neighbors of $\boldsymbol{x}_i$; in addition $\mathcal{X}_i$ can be interpreted as the union of $\mathcal{X}_{i,I} = \{ \boldsymbol{x}_1, \dots, \boldsymbol{x}_{m_I} \}$ and $\mathcal{X}_{i,B} = \{ \boldsymbol{x}_{m_I+1}, \dots, \boldsymbol{x}_m \}$, respectively the set of its m_I nodes belonging to Ω and of its other m_B nodes belonging to $\partial\Omega$. An example of stencil could be found in Figure 2.1.

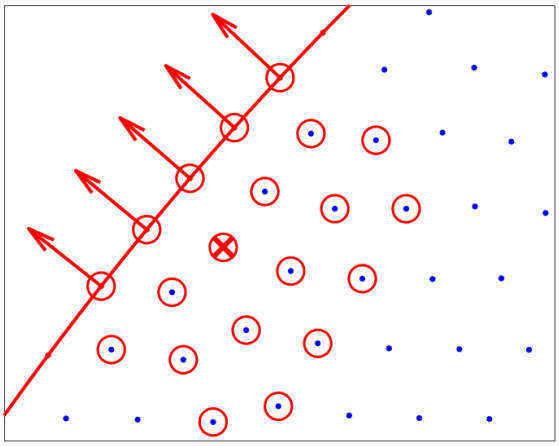

Figure 2.1: Example of 2D stencil. The nodes which belong to it are marked with a red circle, its central node with a red cross. Red arrows represent the boundary normals $\boldsymbol{n}$ of those nodes belonging to $\partial\Omega$. Figure taken from [18]

In Tolstykh's local method the interpolation scheme is local, i.e. u^h is expanded using a basis that changes depending on the position $\boldsymbol{x}$ and it is made valid only inside the stencil centered at $\boldsymbol{x}$ rather than globally across the entire domain. Thus, given a point $\boldsymbol{x}_i \in \mathcal{X}_I$ along its related stencil $\mathcal{X}_i = \{ \boldsymbol{x}_1, \dots, \boldsymbol{x}_m \}$, equation (2.6.6) is rewritten as:

$$u^h(\boldsymbol{x}_i) = \sum_{j=1}^{m} \alpha_j \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k p_k(\boldsymbol{x}_i) \tag{2.6.9}$$

Applying operator $\mathcal{L}$ to the definition of u^h results in:

$$\begin{aligned} \mathcal{L}u^h(\boldsymbol{x}_i) &= \sum_{j=1}^{m} \alpha_j \mathcal{L}\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{k=1}^{M} \beta_k \mathcal{L}p_k(\boldsymbol{x}_i) \\ &= \begin{bmatrix} \boldsymbol{\alpha} & \boldsymbol{\beta} \end{bmatrix} \begin{bmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{bmatrix} \end{aligned} \tag{2.6.10}$$

where different vectors of coefficients $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1, \dots, \alpha_m \end{bmatrix} \in \mathbb{R}^m$ and $\boldsymbol{\beta} = \begin{bmatrix} \beta_1, \dots, \beta_M \end{bmatrix} \in \mathbb{R}^M$ are associated to different $\boldsymbol{x}_i$. In order to find them we can note that, since equation (2.6.9) is still an approximated solution of the boundary value problem (at least locally), it must satisfy:

$$\begin{aligned} u^h(\boldsymbol{x}_i) &= u(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_{i,I} \\ \mathcal{B}u^h(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_{i,B} \end{aligned} \tag{2.6.11}$$

which rewritten in matrix form read as:

$$\underbrace{\begin{bmatrix} \boldsymbol{\Phi}_I & \boldsymbol{P}_I \\ \boldsymbol{\mathcal{B}}\boldsymbol{\Phi}_B & \boldsymbol{\mathcal{B}}\boldsymbol{P}_B \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{M_{BC}}} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_I \\ \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} \tag{2.6.12}$$

where $\boldsymbol{u}_I = \{\, u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_{m_I}) \,\} \in R^{m_I}$ and $\boldsymbol{g} = \{\, g(\boldsymbol{x}_{m_I+1}), \ldots, g(\boldsymbol{x}_m) \,\} \in \mathbb{R}^{m_B}$ and the new terms in $\boldsymbol{M_{BC}} \in \mathbb{R}^{(m+M)\times(m+M)}$ are defined as follow:

$$\begin{aligned}
\boldsymbol{\Phi}_I &= \begin{bmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \Phi(\boldsymbol{x}_{m_i}, \boldsymbol{x}_1) & \ldots & \Phi(\boldsymbol{x}_{m_I}, \boldsymbol{x}_m) \end{bmatrix} \in \mathbb{R}^{m_I \times m} \\
\boldsymbol{P}_I &= \begin{bmatrix} p_1(\boldsymbol{x}_1) & \ldots & p_M(\boldsymbol{x}_{m_I}) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_1) & \ldots & p_M(\boldsymbol{x}_{m_I}) \end{bmatrix} \in \mathbb{R}^{m_I \times M} \\
\boldsymbol{\mathcal{B}}\boldsymbol{\Phi}_B &= \begin{bmatrix} \mathcal{B}\Phi(\boldsymbol{x}_{m_I+1}, \boldsymbol{x_1}) & \ldots & \mathcal{B}\Phi(\boldsymbol{x}_{m_I+1}, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \mathcal{B}\Phi(\boldsymbol{x}_m, \boldsymbol{x_1}) & \ldots & \mathcal{B}\Phi(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{bmatrix} \in \mathbb{R}^{m_B \times m} \\
\boldsymbol{\mathcal{B}}\boldsymbol{P}_B &= \begin{bmatrix} \mathcal{B}p_1(\boldsymbol{x}_{m_I+1}) & \ldots & \mathcal{B}p_M(\boldsymbol{x}_{m_I+1}) \\ \vdots & \ddots & \vdots \\ \mathcal{B}p_1(\boldsymbol{x}_m) & \ldots & \mathcal{B}p_M(\boldsymbol{x}_m) \end{bmatrix} \in \mathbb{R}^{m_B \times M}
\end{aligned} \tag{2.6.13}$$

Substituting the recently determined $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into equation (2.6.10) we obtain:

$$\mathcal{L}u^h(\boldsymbol{x}_i) = \begin{bmatrix} \boldsymbol{u}_I & \boldsymbol{g} & \boldsymbol{0} \end{bmatrix} \boldsymbol{M_{BC}}^{-T} \begin{bmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{bmatrix} \tag{2.6.14}$$

We might now observe that the last two factors of equation (2.6.14) are known, thus we can represent their product with a vector $\boldsymbol{c}(\boldsymbol{x}_i) = \left[\boldsymbol{c}_I(\boldsymbol{x}_i), \boldsymbol{c}_B(\boldsymbol{x}_i), \boldsymbol{c}_p(\boldsymbol{x}_i)\right] \in \mathbb{R}^{m+M}$ that can be obtained solving the dual problem:

$$\boldsymbol{M_{BC}}^T \begin{bmatrix} \boldsymbol{c}_I(\boldsymbol{x}_i) \\ \boldsymbol{c}_B(\boldsymbol{x}_i) \\ \boldsymbol{c}_p(\boldsymbol{x}_i) \end{bmatrix} = \begin{bmatrix} \mathcal{L}\boldsymbol{\Phi}(\boldsymbol{x}_i, \mathcal{X}_{i,I}) \\ \mathcal{L}\boldsymbol{p}(\boldsymbol{x}_i) \end{bmatrix} \tag{2.6.15}$$

where $\boldsymbol{c}_I(\boldsymbol{x}_i)$, $\boldsymbol{c}_B(\boldsymbol{x}_i)$ and $\boldsymbol{c}_p(\boldsymbol{x}_i)$ denote the first m_I, the next m_B and the last M elements of $\boldsymbol{c}(\boldsymbol{x}_i)$ respectively. Once this is done (2.6.14) can be rewritten as:

$$\begin{aligned}
\mathcal{L}u^h(\boldsymbol{x}_i) &= \boldsymbol{c}_I(\boldsymbol{x}_i)^T \boldsymbol{u}_I + \boldsymbol{c}_B(\boldsymbol{x}_i)^T \boldsymbol{g} \\
&= \sum_{j=1}^{m_I} c_j(\boldsymbol{x}_i) u(\boldsymbol{x}_j) + \sum_{k=m_I+1}^{m} c_k(\boldsymbol{x}_i) g(\boldsymbol{x}_k)
\end{aligned} \tag{2.6.16}$$

which is nothing else but the FD-like approximation of $\mathcal{L}u^h$ at point $\boldsymbol{x}_i$ with FD weights given by the elements of the vectors $\boldsymbol{c}_I(\boldsymbol{x}_i)$ and $\boldsymbol{c}_P(\boldsymbol{x}_i)$. We stress that it can not be properly called FD since not only values of u^h (remember that $u(\boldsymbol{x}_j) = u^h(\boldsymbol{x}_j)$ thanks to conditions (2.6.11)), but also of g are used for the approximation.

The values of the approximated solution at the N_I nodes of $\mathcal{X}$, i.e. the solution of the PDE, can be finally found by requiring u^h to approximate the exact solution at each of these points:

$$\mathcal{L}u^h(\boldsymbol{x}_i) = \mathcal{L}u(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) \quad \text{if } \boldsymbol{x}_i \in \mathcal{X}_{\mathcal{I}} \tag{2.6.17}$$

which takes the matrix form:

$$\boldsymbol{C}_I \begin{bmatrix} u^h(\boldsymbol{x}_1) \\ \vdots \\ u^h(\boldsymbol{x}_{N_I}) \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_{N_I})) \end{bmatrix} - \boldsymbol{C}_B \begin{bmatrix} g(\boldsymbol{x}_{N_I+1}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{bmatrix} \tag{2.6.18}$$

where rows of matrices $\boldsymbol{C_I}$ and $\boldsymbol{C_B}$ are formed by the elements of the vectors $\boldsymbol{c}_I(\boldsymbol{x}_i)$ and $\boldsymbol{c}_B(\boldsymbol{x}_i)$ found by solving solving equation (2.6.15) N_I times:

$$\begin{aligned} \boldsymbol{C}_I &= \begin{bmatrix} c_1(\boldsymbol{x}_1) & \dots & c_{N_I}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_1(\boldsymbol{x}_{N_I}) & \dots & c_{N_I}(\boldsymbol{x}_{N_I}) \end{bmatrix} \in \mathbb{R}^{N_I \times N_I} \\ \boldsymbol{C}_B &= \begin{bmatrix} c_{N_I+1}(\boldsymbol{x}_1) & \dots & c_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ c_{N_I+1}(\boldsymbol{x}_{N_I}) & \dots & c_N(\boldsymbol{x}_{N_I}) \end{bmatrix} \in \mathbb{R}^{N_I \times N_B} \end{aligned} \tag{2.6.19}$$

What we just obtained in (2.6.18) is the algebraic system that once solved, in place of the PDE, using an iterative program, gives the values of the PDE approximated solution $\{ u^h(\boldsymbol{x}_1), \dots, u^h(\boldsymbol{x}_{N_I}) \}$. In Kansa's global method the process is followed with the same steps as long as equation (2.6.6) is used to approximate the PDE solution instead of the interpolant in (2.6.9).

An important remark has to be done about computational efficiencies of global and local methods since linear system (2.6.15) has to be solved at any point $\boldsymbol{x}_i \in \mathcal{X}_I$, therefore requiring the inversion of N_I different $\boldsymbol{M_{BC}}^T$ matrices. In case of local method each $\boldsymbol{M_{BC}}$ has size $(m+M) \times (m+M)$; instead in global method they have size $(N+M) \times (N+M)$ since conditions (2.6.11) must be enforced not only in the stencil, but over the entire domain due to the global nature of the approximated solution. Remembering that the computational cost of inverting a matrix is at least of $\mathcal{O}(n^3)$, where here n denotes the number of rows and columns of the matrix

to be inverted, is clear how local method is computationally advantageous respect the global one. Furthermore in global method derivative approximation of u^h at point $\boldsymbol{x}_i$ depends on the values of u across the entire domain even if derivatives are local properties of functions: this is clearly suboptimal since leads to a matrix $\boldsymbol{C}_I$, which represent the discretized differential operator $\mathcal{L}$, that is full even though it should not be in principle. On the contrary, in local method, $\boldsymbol{C}_I$ is sparse since in each row i the number of non-zero entries is equal to the number of internal nodes of stencil $\mathcal{X}_I$. The two reason just listed are the ones that explain the benefits of stencil introduction and that led to the abandonment of Kansa's method in favor of the one proposed by Tolstykh.

What we have not covered yet is the computational cost of the stencils creation: if too costly, it might negate all the benefits of the local approach. In fact this would be the case when using a brute force approach (which require a computational cost of $\mathcal{O}(N^2)$ for each $\boldsymbol{x}_i$) where all pairwise distances between nodes are computed and then sorted in order to keep just the m-nearest neighbors. To avoid this issue more efficient algorithms has to be used. An example of these is the k-d tree algorithm [25] which require $\mathcal{O}(N \log N)$ operations for rearranging the nodes and other $\mathcal{O}(N \log N)$ for finding a fixed number of neighbors. Regarding the effects of stencil size

Finally, in addition to the stencil we note that the effect of its size are still under research: one of the last activities on this regard is the one of Kolar-Požun et al. [28] where they found that changing stencil's size induce oscillations in both the solution and discretization errors for the Poisson equation.

Chapter 3

Automatic differentiation and adjoint method

Cos'è e come funziona la differenz. automatica
Cos'è e come funziona l'aggiunto e legami con la diff. automatica
Come viene applicato il metodo dell'aggiunto nel metodo RBF-FD

Chapter 4

Results

Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4.1 Lorem Ipsum

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

4.1.1 Dolor sit amet

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Appendix A

Sit Amet

Bibliography

[1] Michael Hillman Jiun-Shyan Chen and Sheng-Wei Chi. "Meshfree Methods: Progress Made after 20 Years". In: *Journal of Engineering Mechanics* (2017).

[2] T. Belytschko et al. "Meshless methods: An overview and recent developments". In: *Computer Methods in Applied Mechanics and Engineering* (1996).

[3] W. Benz and E. Asphaug. "Simulations of brittle solids using smooth particle hydrodynamics". In: *Computer Physics Communications* (1995).

[4] S. Jun W. K. Liu and Y. F. Zhang. "Reproducing kernel particle methods". In: *International Journal for Numerical Methods in Fluids* (1996).

[5] P. Lancaster and K. Salkauskas. "Surfaces Generated by Moving Least Squares Methods". In: *Mathematics of Computation* (1981).

[6] E. J. Kansa. "Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—I surface approximations and partial derivative estimates". In: *Computers & Mathematics with Applications* (1990).

[7] E. J. Kansa. "Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations". In: *Computers & Mathematics with Applications* (1990).

[8] M. A. Schweitzer. *Partition of Unity Method*. Lecture Notes in Computational Science and Engineering. 2003. URL: https://doi.org/10.1007/978-3-642-59325-3_2.

[9] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. 2011. URL: https://doi.org/10.1007/978-0-387-70914-7.

[10] G. R Liu and Y. T. GU. *An Introduction to Meshfree Methods and Their Programming*. Cambridge University Press, 2005.

[11] B. Fornberg and N. Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*. 2015.

[12] B. Fornberg and N. Flyer. "Solving PDEs with radial basis functions". In: *Acta Numerica* (2015).

[13] D. Miotti, R. Zamolo, and E. Nobile. "A Fully Meshless Approach to the Numerical Simulation of Heat Conduction Problems over Arbitrary 3D Geometries". In: *Energies* (2021).

[14] I. Amidror. "Scattered data interpolation methods for electronic imaging systems: a survey". In: *Journal of Electronic Imaging* (2002).

[15] J. C. Mairhuber. "On Haar's Theorem Concerning Chebychev Approximation Problems Having Unique Solutions". In: *Proceedings of the American Mathematical Society* (1956).

[16] R. Zamolo. "Radial Basis Function-Finite Difference Meshless Methods for CFD Problems". PhD thesis. Università degli studi di Trieste, 2018.

[17] G. Fasshauer. *Meshfree Approximation Methods with Matlab*. World Scientific Publishing Company, 2007.

[18] D. Miotti. "Stable meshless methods for 3D CFD simulation on complex geometries based on Radial Basis Functions". PhD thesis. Università degli studi di Trieste, 2024.

[19] A. Tolstykh. *On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grid calculations*. 2000. URL: https://scholar.google.com/scholar_lookup?title=On+using+RBF-based+differencing+formulas+for+unstructured+and+mixed+structured%E2%80%93unstructured+grid+calculations&author=Tolstykh+A.+I.&publication+year=2000&journal=Proc.+16th+IMACS+World+Congress&volume=228&pages=4606-4624.

[20] L. A. Bueno, E. A. Divo, and A. J. Kassab. "A coupled localized RBF meshless/DRBEM formulation for accurate modeling of incompressible fluid flows". In: *International Journal of Computational Methods and Experimental Measurements* (2017).

[21] L. A. Bueno, E. A. Divo, and A. J. Kassab. "Multi-scale cardiovascular flow analysis by an integrated meshless-lumped parameter model". In: *International Journal of Computational Methods and Experimental Measurements* (2018).

[22] G. Kosec and J. Slak. "Rbf-fd based dynamic thermal rating of overhead power lines". In: *Advances in Fluid Mechanics XII* (2018).

[23] G. Kosec and J. Slak. "Radial basis function-generated finite differences solution of natural convection problem in 3D". In: *AIP Conference Proceedings 2293* (2020).

[24] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-dependent Problems*. Society for Industrial and Applied Mathematics, 2007.

[25] J. L. Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Communications of the ACM* (1975).

[26] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[27] R. Zamolo, D. Miotti, and E. Nobile. "Accurate stabilization techniques for RBF-FD meshless discretizations with neumann boundary conditions". In: *SSRN Electronic Journal* (2022).

[28] A. Kolar-Požun et al. "Oscillatory behaviour of the RBF-FD approximation accuracy under increasing stencil size". In: *Computational Science – ICCS 2023* (2023).

[29] L. Bacer. "Fully meshless approaches for the numerical solution of partial differential equations: radial basis function finite difference method and physics-informed neural network". MA thesis. Università degli studi di Trieste, 2023.

[30] K. C. Jeyanthi. "Efficient node generation over complex 3D geometries for meshless RBF-FD method". MA thesis. Università degli studi di Trieste, 2022.

[31] Wikipedia contributors. *Finite difference method*. Online; accessed 21-February-2024. 2024. URL: https://en.wikipedia.org/wiki/Finite_difference_method#Further_reading.

Ei fu. Siccome immobile,
Dato il mortal sospiro,
Stette la spoglia immemore
Orba di tanto spiro

Alessandro Manzoni – *Il Cinque Maggio*