energies

MDPI

Article

A Fully Meshless Approach to the Numerical Simulation of Heat Conduction Problems over Arbitrary 3D Geometries

Davide Miotti , Riccardo Zamolo and Enrico Nobile *

Department of Engineering and Architecture, University of Trieste, Via Alfonso Valerio 10, 34127 Trieste, Italy; davide.miotti@PHD.units.it (D.M.); rzamolo@units.it (R.Z.)
* Correspondence: nobile@units.it

Abstract: One of the goals of new CAE (Computer Aided Engineering) software is to reduce both time and costs of the design process without compromising accuracy. This result can be achieved, for instance, by promoting a "plug and play" philosophy, based on the adoption of automatic mesh generation algorithms. This in turn brings about some drawbacks, among others an unavoidable loss of accuracy due to the lack of specificity of the produced discretization. Alternatively it is possible to rely on the so called "meshless" methods, which skip the mesh generation process altogether. The purpose of this paper is to present a fully meshless approach, based on Radial Basis Function generated Finite Differences (RBF-FD), for the numerical solution of generic elliptic PDEs, with particular reference to time-dependent and steady 3D heat conduction problems. The absence of connectivity information, which is a peculiar feature of this meshless approach, is leveraged in order to develop an efficient procedure that accepts as input any given geometry defined by a stereolithography surface (`.stl` file format). In order to assess its performance, the aforementioned strategy is tested over multiple geometries, selected for their complexity and engineering relevance, highlighting excellent results both in terms of accuracy and computational efficiency. In order to account for future extensibility and performance, both node generation and domain discretization routines are entirely developed using `Julia`, an emerging programming language that is rapidly establishing itself as the new standard for scientific computing.

Keywords: meshless; RBF-FD; heat conduction; CAE; Julia

Citation: Miotti, D.; Zamolo, R.; Nobile, E. A Fully Meshless Approach to the Numerical Simulation of Heat Conduction Problems over Arbitrary 3D Geometries. *Energies* **2021**, *14*, 1351. https://doi.org/10.3390/en14051351

Academic Editor: Alessandro Mauro

Received: 5 February 2021
Accepted: 22 February 2021
Published: 2 March 2021

1. Introduction

Engineering design often involve the solution of some boundary value problem, e.g., heat conduction problems, with the aid of numerical approaches. Well-known Finite Volume (FVM) [1] and Finite Element (FEM) [2] methods are usually employed to discretize the governing equations, i.e., partial differential equations (PDEs), in order to obtain a finite dimensional problem that can be solved numerically.

The solution attained in the end is indeed bound to be just an approximation, due to an unavoidable loss of information that takes place already at this stage, and the quality of the discretization determines how much information is lost and how much is preserved.

Usually the discretization process begins with the creation of a mesh, i.e., a subdivision of the domain in many discrete cells, which is often generated by some computer algorithm under the supervision of a specialist. Sometimes, due to time and/or budget constraints, the mesh is automatically generated, and the designer is left free to focus on other topics. In both cases the key point is the introduction of a mesh in the formulation stage; therefore, any numerical method applied afterwards relies on the related connectivity information.

This procedure presents some important weaknesses [3], two of them are:

- the cost of the mesh creation and the size of the consequent data structure,
- the inability to allow large (geometric) deformations of the domain.

The first one refers to both computational and economical costs; indeed operator costs now outweigh the cost of CPU time for the computer [3], and if the operator intervention is avoided it becomes mandatory to use a finer automated mesh generation algorithm. This in turn gives as output a greater data structure and thus slows down successive numerical manipulations.

The second weakness becomes of interest in many practical cases, for example when the solver is paired with some optimization algorithm and the geometry is changed many times. At each step the mesh is degraded in quality, and even when morphing algorithms are employed, the solution becomes less reliable at every iteration. In some cases a periodic complete re-meshing of the domain becomes necessary.

These facts led to the development of various meshless or meshfree methods for the solution of partial differential equations over complex shaped domains encountered in practical problems [4–7]. Some authors have successfully experimented with the use of meshfree methods in order to enforce specific boundary conditions, for example Absorbing Boundary Conditions (ABCs), while the state equation is solved using the more traditional Finite Element Method (FEM) [8]. Meshfree methods are also destined to play a key role in the development of new implementations of peridynamic (PD) models, as shown in [9]. Numerical methods belonging to this class allow to skip the mesh generation procedure altogether and, therefore, overcome its inherent shortcomings.

Most meshless methods (also called meshfree methods) can be classified according to three different criteria [3]:

- according to the formulation procedure,
- according to the function approximation schemes, and
- according to the domain representation.

The common characteristic is the use of the so-called field nodes: at first a set of nodes is scattered over the domain (and/or over the boundaries). Then, shape functions are constructed for each node of interest and are based on selected local nodes; the shape functions can change when the node of interest changes [3].

This paper discusses the implementation of one such method and its integration with a novel node generation procedure. In order to reach maximum compatibility with commercial software, the domain is assumed as defined by a generic stereolithography (`.stl`) file, like those used in the fields of computer graphics and 3D printing. Furthermore the obtained data structure is more efficient than that of a traditional mesh, since it carries no information about the connectivity between nodes. To the best of the authors' knowledge, this is the first time a fully meshless approach, based on the RBF-FD method, is employed for the accurate solution of heat conduction problems over arbitrary 3D domains defined by a generic stereolithography CAD file.

The greater geometric flexibility of truly meshless approaches is therefore fully leveraged, with the aim of making them more accessible and user-friendly in the future. More precisely, the RBF-generated Finite Differences (RBF-FD) method is employed [10,11], similar to the one described in [5]. This method has been successfully employed for solving various heat transfer problems [12–15]. Interested readers might also refer to [16] for further details and comparisons between the performance attained with different RBF schemes.

This method belongs to the following classes, according to the aforementioned classification criteria:

- formulation procedure: collocation technique. The problem to be solved is formulated in the strong form, i.e., strong forms of governing equations and boundary conditions are directly discretized at the field nodes.
- function approximation scheme: point interpolation (PIM). All functions are approximated at a certain node using RBFs defined on a compact support around the field nodes. In order to achieve greater accuracy and stability the RBF scheme is augmented with a polynomial term [17,18].
- domain representation: both domain and boundaries are represented by field nodes.

See Figure 1 for a schematic representation of the differences between the RBF-FD meshless method and traditional mesh-based methods. It must be remarked that the solution procedure is totally independent of the particular shape of such domain.

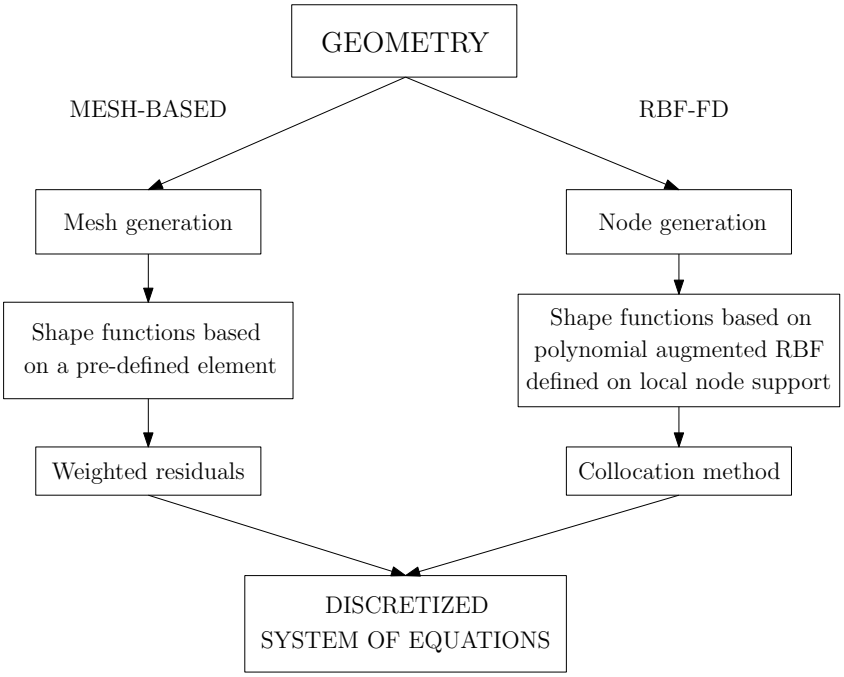


Figure 1. Flowchart for FEM and meshless methods, inspired by that reported in [3].

The whole code is developed using Julia programming language [19] and can be executed in parallel on multiple cores, allowing extensive code reuse and excellent computational performances already at the development stage.

Several tests over different 3D geometries with different boundary conditions will be presented in order to illustrate the numerical properties of the proposed truly meshless approach. Excellent results in terms of both accuracy and computational efficiency have been obtained in each of the presented cases, confirming the ability of RBF-FD method to easily deal with problems of engineering relevance.

2. Materials and Methods

2.1. Governing Equation

The generic heat equation with internal heat generation (see [20]) can be written in the form

$$\frac{\partial u}{\partial t} - \triangle u = f \tag{1}$$

In this paper the steady-state version of Equation (1) will be considered; therefore, the governing equations are given by the following boundary value problem:

$$\begin{cases} -\triangle u = f & in\ \Omega \\ au + b\,\frac{\partial u}{\partial n} = g & on\ \partial\Omega \end{cases} \tag{2}$$

where Ω is an open subset of $\mathbb{R}^3$ and $\partial\Omega$ its boundary; f, a, b, and g are known functions; and n is the unit normal vector of $\partial\Omega$. The second equation in (2) represents, therefore, a boundary condition (b.c.).

The presented method can also be employed to solve transient heat conduction problems, as well as generic heat transfer problems with time-dependent behavior, allowing the implementation of standard explicit or implicit time integration schemes as shown in [7,13,21].

2.2. Node Generation

The first step of the employed meshless approach is to generate a 3D node distribution in Ω and on $\partial\Omega$, i.e., nodes are placed both inside the domain Ω and on the boundary $\partial\Omega$ according to a prescribed spacing function s. Although traditional mesh generation software could be employed, it is obvious that truly meshless node generation could bring great benefits in the whole meshless simulation chain. Different meshless-based node generation algorithms have been proposed [22–25], with particular reference to the ones based on the node-repel approach [26–28], which is employed in this work. The latter approach consists of two sequential steps:

- generation of a volumetric node distribution in Ω that satisfies the spacing function s on average;
- refinement of the initial node distribution through a node-repel approach that provides a suitable node distribution also on $\partial\Omega$.

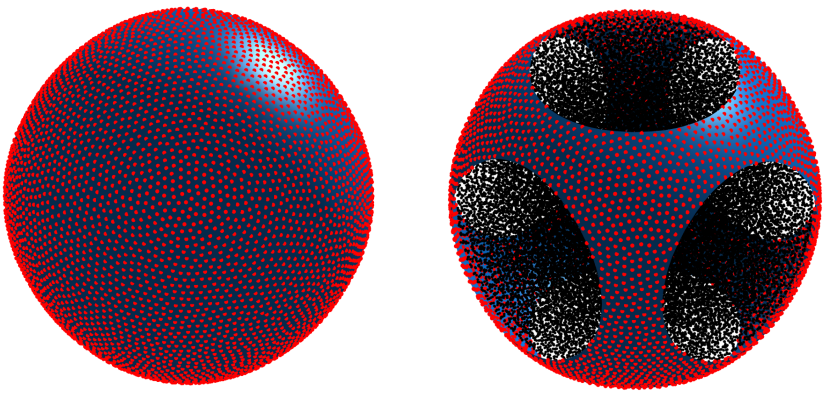
In the last step, i.e., the refinement step, nodes are moved according to the mutual repulsion forces of the nearest neighbor nodes in order to obtain isotropic node arrangements, i.e., the spacing between the nodes is independent of the direction and matches the prescribed spacing function s. This node-repel approach can be viewed as an iterative process that minimizes the total potential energy U of the node distribution by moving one node at a time. In this work 50 refinement iterations and a constant spacing function s have been considered [27,28].

The nodes leaving the domain due to the repulsion forces during the refinement step are projected onto the nearest boundary point, resulting in a boundary-conforming node distribution. This projection operation is efficiently performed by exploiting an octree data structure [29,30] for the boundary triangles of the stereolitography surface (`.stl` file), which is used to determine the distance between a given node and the nearest boundary triangle, i.e., the minimum distance between the node and the boundary.

The triangle-based octree partitioning of the space around the object is performed by recursive subdivision of the space into octants while fulfilling the following rules:

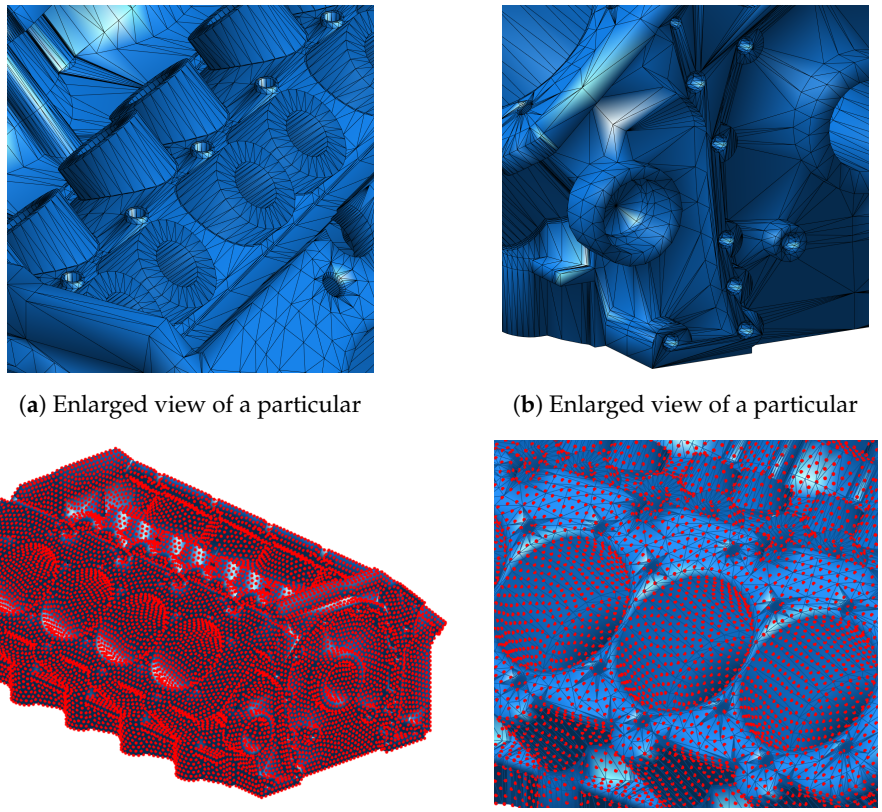
- each leaf-box, i.e., box with no children, can not contain more than $T_M = 5$ triangles;
- the minimum size of each leaf-box is $h_m = 5 \times 10^{-3}$ (the domain is assumed to lie in the unit cube);
- the first constraint can be ignored if a box and its parent-box both contain exactly one vertex of the triangulated surface, which is also the same vertex;

where the last rule avoids infinite octree subdivisions around nodes connected to more than T_M triangles. Graphical representations of generated node distributions are shown in Figures 2 and 3 for the employed models.



(**a**) Boundary nodes (**b**) Inner nodes

Figure 2. Example of node distribution with $N \approx$ 50k nodes in the sphere: boundary nodes are represented in red, inner nodes in black.



(**a**) Enlarged view of a particular (**b**) Enlarged view of a particular

(**c**) Boundary nodes (**d**) Enlarged view of a particular

Figure 3. Enlarged views of some details of the `.stl` crankcase model and an example of node distribution with $N \approx 75$k nodes.

2.3. Function Approximation

The value of a function at a generic point is approximated applying the Radial Point Interpolation Method (RPIM) using Radial Basis Functions (RBFs) augmented with polynomial terms [3]. Given a boundary-conforming node distribution as outlined in Section 2.2, this method is based on the assumption that the value $u(x)$ taken by the function u at the generic point $x \in \Omega$ can be locally approximated by the quantity $u^h(x)$ defined as follows:

$$u(x) \approx u^h(x) := \sum_{i=1}^{\bar{n}} \alpha_i \varphi(\|x - x_i\|_2) + \sum_{j=1}^{m} \beta_j p_j(x) \tag{3}$$

where

- $\{\varphi(\|x - x_i\|_2) : i = 1, \ldots, \bar{n}\}$ is a set of $\bar{n}$ RBFs, each of which is associated to the corresponding node x_i. The set of the considered nodes is local, i.e., $x_1, \ldots, x_{\bar{n}}$ are the $\bar{n}$ nearest nodes to x. For brevity of notation, $\varphi(x, x_i)$ will be used instead of $\varphi(\|x - x_i\|_2)$.
- $\{p_j : j = 1, \ldots, m\}$ is a complete polynomial basis of degree P.

The adoption of the polynomial is motivated by the need to solve those accuracy and stability issues affecting pure RBF schemes [17].

In this implementation of the method, multiquadric functions [31] have been used:

$$\varphi(r) := \sqrt{1 + (\varepsilon r)^2} \tag{4}$$

where the actual shape factor $\varepsilon := \bar{\varepsilon}/s(x)$ is obtained by normalizing the shape factor parameter $\bar{\varepsilon}$ with the local nodal spacing defined by the spacing function $s(x)$.

Since the number of nodes is finite, u^h is a finite-dimensional approximation of u; therefore, great care must be taken in the choice of such number in order to balance

efficiency and accuracy. From now on we will assume that a small number of neighbors $\bar{n} \ll N$ located within a neighborhood of the point x is used, where N is the total number of nodes generated. From this assumption it is implicitly understood that the RBFs have compact support, i.e., they are zero for $\|x - x_i\| > \delta$ for a certain δ. The employment of compactly supported RBFs was made compulsory by two closely related reasons: the great number of nodes required to properly describe complex shaped domains, such as those presented in the following sections, and the consequent necessity to deal with large-scale matrices. Local RBF expansions lead to sparse linear systems that can be efficiently solved, in contrast to global approaches that lead to dense and ill-conditioned matrices, which preclude any application to large scale problems [14,32–34]. Accuracy and stability issues affecting global methods are clearly addressed in [35,36], where further possible solutions are discussed.

This in turn means that the value assumed by u^h at a certain point x only depends on $u^h(x_i)$ estimated at the closest nodes $\{x_1, \ldots, x_{\bar{n}}\}$, from now on called the stencil:

$$u(x) \approx u^h(x) := \sum_{i=1}^{\bar{n}} \alpha_i \varphi(x, x_i) + \sum_{j=1}^{m} \beta_j p_j(x) \tag{5}$$

The calculation of the weights $\{\alpha_1, \ldots, \alpha_{\bar{n}}\}$ and $\{\beta_1, \ldots, \beta_m\}$ depends on the position of the stencil. Therefore, if some nodes of the stencil belongs to the boundary $\partial\Omega$, then boundary conditions must be taken into account.

2.3.1. Stencil Contained within Ω

In this case none of the nodes belonging to the stencil lies on the boundary $\partial\Omega$. Assuming the values of the function u at the nodes of the stencil $\{x_1, \ldots, x_{\bar{n}}\}$ are known, then a linear system of equations can be solved by imposing the exactness of the approximation u^h at each of those nodes:

$$u^h(x_i) = u(x_i), \quad i = 1, \ldots, \bar{n} \tag{6}$$

Substituting the definition of $u^h(x)$ given in Equation (5) and evaluating at the $\bar{n}$ points $\{x_1, \ldots, x_{\bar{n}}\}$ leads to the following linear system of equations:

$$\boldsymbol{\Phi}\,\boldsymbol{\alpha} + \boldsymbol{P}\,\boldsymbol{\beta} = \bar{\boldsymbol{u}} \tag{7}$$

where $\boldsymbol{\Phi}$ is a $(\bar{n} \times \bar{n})$ square matrix with entries $\boldsymbol{\Phi}_{i,j} = \varphi(x_i, x_j)$ as defined in Equation (4), $\boldsymbol{P}$ is the $(\bar{n} \times m)$ matrix of the polynomial coefficients, $\boldsymbol{\alpha} \in \mathbb{R}^{\bar{n}}$ and $\boldsymbol{\beta} \in \mathbb{R}^m$ are the vectors with a total of $(\bar{n} + m)$ unknowns. $\bar{\boldsymbol{u}}$ is instead the $(\bar{n} \times 1)$ column vector with entries $u(x_i)$ for each x_i node of the stencil.

In order to make the system solvable, m more rows are needed, this is why the following additional orthogonality conditions are imposed:

$$\sum_{i=1}^{\bar{n}} \alpha_i p_j(x_i) = 0, \quad j = 1, \ldots, m \tag{8}$$

The final linear system can thus be written using block matrices notation in the following form:

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{Bmatrix} = \begin{Bmatrix} \bar{\boldsymbol{u}} \\ \boldsymbol{0} \end{Bmatrix} \tag{9}$$

which has a unique solution in the set of weights $\{\alpha_1 \ldots \alpha_{\bar{n}}, \beta_1, \ldots \beta_m\}$ if the $\bar{n}$ nodes $x_1, \ldots, x_{\bar{n}}$ are distinct.

2.3.2. Stencil with Boundary Nodes

When some boundary nodes are included in the stencil, then instead of assuming the true value of the function u, the appropriate condition must be imposed. In the attempt of

providing a general description we shall assume that a Robin b.c., as defined in Equation (2), is applied to such nodes:

$$a(x_i)\, u^h(x_i) + b(x_i)\, \frac{\partial u^h}{\partial n}(x_i) = g(x_i), \quad x_i \in \partial\Omega \tag{10}$$

where a, b, g are all known functions defined on the boundary.

Now suppose the nodes are listed in the following order: the first $\bar{n}_I$ indices correspond to the nodes lying within the domain Ω, the following $\bar{n}_B$ indices correspond instead to the nodes lying on $\partial\Omega$, and $\bar{n}_I + \bar{n}_B = \bar{n}$. More formally:

$$\begin{cases} x_i \in \Omega & \iff i \in \mathcal{I} := \{1, \dots, \bar{n}_I\} \\ x_i \in \partial\Omega & \iff i \in \mathcal{B} := \{\bar{n}_I + 1, \dots, \bar{n}_I + \bar{n}_B\} \end{cases} \tag{11}$$

Building on what was done when dealing with inner nodes, the problem to be solved is the following:

$$\begin{cases} u^h(x_i) = u(x_i), & i \in \mathcal{I} \\ a(x_i)\, u^h(x_i) + b(x_i)\, \frac{\partial u^h}{\partial n}(x_i) = g(x_i), & i \in \mathcal{B} \end{cases} \tag{12}$$

By substituting once again the definition of u^h given in Equation (5) we get

$$\sum_{k=1}^{\bar{n}} \alpha_i \varphi(x_i, x_k) + \sum_{j=1}^{m} \beta_j p_j(x_i) = u(x_i), \quad i \in \mathcal{I} \tag{13}$$

$$\begin{aligned} a(x_i) \left[\sum_{k=1}^{\bar{n}} \alpha_k \varphi(x_i, x_k) + \sum_{j=1}^{m} \beta_j p_j(x_i) \right] + \\ + b(x_i) \left[\sum_{k=1}^{\bar{n}} \alpha_k \frac{\partial \varphi}{\partial n}(x_i, x_k) + \sum_{j=1}^{b} \frac{\partial p_j}{\partial n}(x_i) \right] = g(x_i), \quad i \in \mathcal{B} \end{aligned} \tag{14}$$

By reordering the terms for case $i \in \mathcal{B}$ of Equation (14) in order to isolate unknown coefficients (α's and β's), we obtain the following linear system:

$$\begin{aligned} \sum_{k=1}^{\bar{n}} \alpha_k \left(a(x_i)\varphi(x_i, x_k) + b(x_i) \frac{\partial \varphi}{\partial n}(x_i, x_k) \right) + \\ + \sum_{j=1}^{m} \beta_j \left(a(x_i) p_j(x_i) + b(x_i) \frac{\partial p_j}{\partial n}(x_i) \right) = g(x_i), \quad i \in \mathcal{B} \end{aligned} \tag{15}$$

Once again we can now impose the orthogonality conditions of Equation (8) and obtain a linear system of the following form:

$$\underbrace{\begin{bmatrix} \boldsymbol{\Phi'} & \boldsymbol{P'} \\ \boldsymbol{P^T} & \mathbf{0} \end{bmatrix}}_{\boldsymbol{M}} \begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{Bmatrix} = \begin{Bmatrix} \bar{\boldsymbol{u}} \\ \bar{\boldsymbol{g}} \\ \mathbf{0} \end{Bmatrix} \tag{16}$$

where the block matrix $\boldsymbol{P}^T$ and the vector of unknowns $\{\alpha_1, \dots, \alpha_{\bar{n}}\}$ are the only unchanged terms inherited from the previous system (9). The other terms in Equation (16) are fully expanded below for clarity.

$$\left[\boldsymbol{\Phi'}\right] = \begin{bmatrix} \varphi(x_1, x_1) & \dots & \varphi(x_1, x_{\bar{n}}) \\ \vdots & \ddots & \vdots \\ \varphi(x_{\bar{n}_I}, x_1) & \dots & \varphi(x_{\bar{n}_I}, x_{\bar{n}}) \\ a\varphi(x_{\bar{n}_I+1}, x_1) + b\frac{\partial \varphi}{\partial n}(x_{\bar{n}_I+1}, x_1) & \dots & a\varphi(x_{\bar{n}_I+1}, x_{\bar{n}}) + b\frac{\partial \varphi}{\partial n}(x_{\bar{n}_I+1}, x_{\bar{n}}) \\ \vdots & \ddots & \vdots \\ a\varphi(x_{\bar{n}}, x_1) + b\frac{\partial \varphi}{\partial n}(x_{\bar{n}}, x_1) & \dots & a\varphi(x_{\bar{n}}, x_{\bar{n}}) + b\frac{\partial \varphi}{\partial n}(x_{\bar{n}}, x_{\bar{n}}) \end{bmatrix}$$

$$\left[\boldsymbol{P'}\right] = \begin{bmatrix} p_1(x_1) & \dots & p_m(x_1) \\ \vdots & \ddots & \vdots \\ p_1(x_{\bar{n}_I}) & \dots & p_m(x_{\bar{n}_I}) \\ a\, p_1(x_{\bar{n}_I+1}) + b\frac{\partial p_1}{\partial n}(x_{\bar{n}_I+1}) & \dots & a\, p_m(x_{\bar{n}_I+1}) + b\frac{\partial p_m}{\partial n}(x_{\bar{n}_I+1}) \\ \vdots & \ddots & \vdots \\ a\, p_1(x_{\bar{n}}) + b\frac{\partial p_1}{\partial n}(x_{\bar{n}}) & \dots & a\, p_m(x_{\bar{n}}) + b\frac{\partial p_m}{\partial n}(x_{\bar{n}}) \end{bmatrix}$$

$$\begin{bmatrix} \bar{\boldsymbol{u}} \\ \bar{\boldsymbol{g}} \\ \boldsymbol{0} \end{bmatrix} = \begin{Bmatrix} u(x_1) \\ \vdots \\ u(x_{\bar{n}_I}) \\ g(x_{\bar{n}_I+1}) \\ \vdots \\ g(x_{\bar{n}}) \\ 0_1 \\ \vdots \\ 0_m \end{Bmatrix}$$

Please note that a and b are two known functions; therefore, they may have different values in different rows. Finally we remark that Equation (9) can be considered as a special case of Equation (16) with no boundary nodes, i.e., where none of the neighbors lie on the boundary.

2.4. Collocation Technique

Considering the boundary value problem in (2), an approximation of the solution u with a finite dimensional function u^h leads to a certain error. u^h is now defined over the whole domain Ω, and its value at each point is interpolated using local information, as explained in Equation (5). The numerical procedure adopted in order to solve the linear problem in (2) aims at finding a "good" finite dimensional representation of the solution u. The Weighted Residual Method in combination with the Collocation Technique is employed, here follows a brief discussion of such procedure, see [3] for further details.

The linear PDE in (2) is applied to the approximate solution u^h given by Equation (5) instead of u. Therefore, the initial PDE will not be solved exactly, in general, and a nonzero residual function R_s will appear:

$$R_s(x) := \triangle u^h(x) + f(x) \neq 0, \quad x \in \Omega \tag{17}$$

The Weighted Residual Method now consists in forcing to zero the following integrals, where N_I is the total number of nodes generated inside the domain Ω:

$$\int_\Omega W_i R_s \, d\Omega = 0, \quad i = 1, \dots, N_I \tag{18}$$

Substituting the definition of the residual functions leads to the following system of equations:

$$\int_{\Omega} W_i(\triangle u^h + f)\, d\Omega = 0, \quad i = 1, \ldots, N_I \tag{19}$$

In order to get rid of the integrals in the above equations, many techniques are proposed, among others the Collocation Method has been chosen. The main advantage is its ability to discretize the Boundary Value problem expressed in strong form, thus leading to a real fully meshless approach (see again [3,28]).

The basic idea behind the Collocation Method is to choose the solution with vanishing residuals at the nodes only. Formally, this is achieved by defining the weight functions W_i as Dirac Delta Functions as follows:

$$W_i(x) = \delta(x - x_i), \quad i = 1, \ldots, N_I \tag{20}$$

As a result, u^h is required to send the residual function R_s to zero at the nodes x_i:

$$\int_{\Omega} \delta(x - x_i)(\triangle u^h + f)\, d\Omega = \triangle u^h \Big|_{x=x_i} + f(x_i) = 0, \quad i = 1, \ldots, N_I \tag{21}$$

At this point we are allowed to substitute the definition of u^h, as defined in Equation (5), into Equation (21). From now on let us focus our attention on a specific node $x_i \in \Omega$:

$$\triangle \left(\sum_{k=1}^{\bar{n}} \alpha_k \varphi(x, x_k) + \sum_{j=1}^{m} \beta_j p_j(x) \right) \Bigg|_{x=x_i} + f(x_i) = 0 \tag{22}$$

Finally, because of the linearity of the Laplace differential operator, it can be distributed over the basis functions (radial and polynomial), thus yielding the following equation:

$$\sum_{k=1}^{\bar{n}} \alpha_k \triangle \varphi(x, x_k)\big|_{x=x_i} + \sum_{j=1}^{m} \beta_j \triangle p_j(x)\big|_{x=x_i} + f(x_i) = 0 \tag{23}$$

Equation (23) can be recast into the following compact form:

$$\begin{Bmatrix} \boldsymbol{\Psi}(x_i) \\ \boldsymbol{\Pi}(x_i) \end{Bmatrix}^T \begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{Bmatrix} + f(x_i) = 0 \tag{24}$$

where $\boldsymbol{\Psi}(x_i)$ and $\boldsymbol{\Pi}(x_i)$ are defined as follows:

$$\boldsymbol{\Psi}(x) := \begin{Bmatrix} \triangle\varphi(x, x_1) \\ \vdots \\ \triangle\varphi(x, x_{\bar{n}}) \end{Bmatrix}, \quad \boldsymbol{\Pi}(x) := \begin{Bmatrix} \triangle p_1(x) \\ \vdots \\ \triangle p_m(x) \end{Bmatrix} \tag{25}$$

whereas $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_{\bar{n}}\}$ and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_m\}$ are unknown vectors of the expansion coefficients. Then, Equation (16) can be formally solved for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, assuming the value of u^h to be formally known at the nodes belonging to the stencil:

$$\begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{Bmatrix} = [\boldsymbol{M}]^{-1} \begin{Bmatrix} \bar{\boldsymbol{u}}^h \\ \bar{\boldsymbol{g}} \\ \mathbf{0} \end{Bmatrix} \tag{26}$$

This result can then be substituted back into Equation (24), thus yielding the following equation:

$$\begin{Bmatrix} \boldsymbol{\Psi}(x_i) \\ \boldsymbol{\Pi}(x_i) \end{Bmatrix}^T [\boldsymbol{M}]^{-1} \begin{Bmatrix} \bar{\boldsymbol{u}}^h \\ \bar{\boldsymbol{g}} \\ \mathbf{0} \end{Bmatrix} + f(x_i) = 0 \tag{27}$$

We might now observe that the first two factors of Equation (27) are known and their product yields to a row vector $\boldsymbol{c}^T$. Therefore, instead of performing the inversion of matrix $\boldsymbol{M}$, we can therefore solve the following adjoint linear system for $\boldsymbol{c}$, where the right-hand side is known:

$$\left[\boldsymbol{M}\right]^T\{\boldsymbol{c}\} = \begin{Bmatrix} \boldsymbol{\Psi}(x_i) \\ \boldsymbol{\Pi}(x_i) \end{Bmatrix} \tag{28}$$

The scalar Equation (27) written in terms of the known vector $\boldsymbol{c}$ reads as follows:

$$\{\boldsymbol{c}\}\begin{Bmatrix} \bar{\boldsymbol{u}}^h \\ \bar{\boldsymbol{g}} \\ \boldsymbol{0} \end{Bmatrix} + f(x_i) = 0 \tag{29}$$

which represents the ith scalar equation of the following sparse linear system:

$$\left[\boldsymbol{C}\right]\{\boldsymbol{u}^h\} = \boldsymbol{q} - \boldsymbol{f} \tag{30}$$

where $\boldsymbol{u}^h = \{u^h(x_1), \ldots, u^h(x_{N_I})\}$, $\boldsymbol{f} = \{f(x_1), \ldots, f(x_{N_I})\}$ and $\boldsymbol{q} = \{q_1, \ldots, q_{N_I}\}$ are referred to all the N_I internal nodes, i.e., those contained in Ω, while vector $\boldsymbol{q}$ comes from the imposition of the boundary conditions at the interpolant level.

2.5. Solution Procedure

The final sparse system of Equation (30) can now be solved using an iterative method, thus obtaining the value of unknowns $\boldsymbol{u}^h = \{u^h(x_1), \ldots, u^h(x_{N_I})\}$ at all internal nodes.

In the current implementation, a LU decomposition is performed for the solution of the small linear systems (28) associated to each inner node x_i, while the final sparse system (30) is solved instead by performing an Incomplete LU Factorization (ILU) [37] (package `IncompleteLU` in `Julia`) of the sparse matrix $\boldsymbol{C}$ followed by the application of the Biconjugate Gradient Stabilized Method [38] (package `IterativeSolvers` in `Julia`) employing a relative norm $tol = 10^{-14}$ on the residuals.

2.6. Julia Programming Language

The `Julia` Programming Language [19] was announced as an open source project in February 2012, in the attempt of taking advantage of modern techniques for executing dynamic languages effectively and providing a single solution, capable of being both fast and dynamic at the same time.

High-level dynamic languages on one side provide great advantages in terms of convenience and productivity and are often preferred even in the fields of data analysis, applied math, engineering, and other sciences. On the other hand, lower-level languages like `C` and `Fortran` remain the gold standard for computationally intensive problems, since high-level dynamic languages tend to lack sufficient performance [39].

`C` and `Fortran`, however, tend to be harder to work with and require more specific skills; therefore, they might become less productive and sometimes slow down the whole development process.

If a compromise between convenience and performance is desired, then one of the following two approaches were available at the time `Julia` came out.

- Two-tiered architecture solutions: programmers express high-level logic in a dynamic language (like `MATLAB`, `Octave`, `R`, or `SciPi`), while heavy lifting is done in `C` and `Fortran` [39].
- Enhanced versions of existing dynamic languages: specific libraries need to be used for this purpose; however, the same programmer is given the possibility to contribute to the whole process without completely loosing understanding.

`Julia` was therefore designed from the ground up and succeeded in providing a new and radically different alternative for the aforementioned compromise. It has indeed the performance of a statically compiled language while providing interactive dynamic

behavior and productivity comparable to those of `Python`, `LISP`, or `Ruby` [39]. On top of that, a large community rapidly grew around the initial project, and many open source packages and excellent integration with text editors have become available to date.

For these reasons, `Julia` was chosen for the development of all the components of the RBF-FD solver presented in this paper, allowing optimal management of data structures and also taking into account future extensibility and high performance.

2.7. MATLAB PDE Toolbox

In order to compare the results of the presented approach with other numerical methods, MATLAB Partial Differential Equation Toolbox was employed [40]. This toolbox offers the user a complete package (including both discretization and solution routines) based on FEM with tetrahedral elements in 3D.

Quadratic elements, i.e., 10-node tetrahedrals, have been used. The aforementioned toolbox was chosen in order to allow a comparison with the FEM, which is one of the most employed methods for the numerical simulation of engineering problems. It is important to point out that the tetrahedral mesh that was generated is patch-conforming, i.e., the dimension of the elements adapts to the size of the features of the geometry. On the other side, the meshless node distributions are obtained with constant spacing functions for simplicity. Therefore, FEM results are expected to be truly optimal since they are attained using a fully developed adaptive meshing, while meshless results, on the other side, are expected to improve substantially in the case of the engine domain. The accuracy of the meshless method would indeed be improved significantly with the adoption of an advanced node generation routine, capable of performing some local reduction in the nodal spacing in proximity to complex geometrical features.

3. Results

3.1. Code Verification

In order to assess correctness, accuracy, and efficiency of the presented meshless implementation, the Method of Manufactured Solution (MMS) [41] is applied:

1. a sufficiently smooth function $u(x)$ is chosen,
2. the boundary conditions and the internal heat generation f are analytically computed in order to ensure $u(x)$ to be the exact solution of the Boundary Value Problem (2),
3. the RBF-FD method is employed to solve the Boundary Value Problem, thus leading to an approximated solution $u^h(x)$. Much information concerning order of accuracy and efficiency is collected during the calculations,
4. $u^h(x)$ is finally compared to $u(x)$ according to various metrics, and the performance of the method is assessed.

3.2. Analytical Solution

The analytical solution chosen for the verification of the correctness of the code is simply any polynomial $u(x)$ of degree $\bar{P}$. For simplicity, the following polynomial has been chosen:

$$u(x) = x_1^{\bar{P}} + x_2^{\bar{P}} + x_3^{\bar{P}} \tag{31}$$

with $\bar{P} = 1, \ldots, 6$ and which can be exactly solved by the RBF-FD method as long as the degree P of the augmented polynomial fulfills the condition $P \geq \bar{P}$.

The analytical solution chosen for the assessment of the accuracy of the presented implementation of the RBF-FD method is the following function, defined for every $x \in \mathbb{R}^3$ and continuously differentiable infinitely many times:

$$u(x) = \exp(x_1 + x_2 + x_3) \tag{32}$$

3.3. Choice of the Domain

By taking into account the final accuracy alone, one might underestimate the importance of the node generation routine, since the performances of the solver strongly depend

on the quality of this previous step. In our case both elements, i.e., node generator and solver, are written using `Julia` and developed from the very beginning to be seamlessly integrated. For this reason two very different domains are selected: one is very regular and the other is complex and can be considered to be a good representative for the class of problems usually met in engineering design. Both domains are described by standard stereolitography surfaces encoded in the `.stl` format:

- The first is given by the 3D sphere highlighted in Figure 2. This surface is chosen for its regularity, and the absence of any sharp edge is an especially forgiving feature when the stencil around each node is chosen. The accuracy and the robustness of the whole approach are therefore expected to be very high: the final accuracy depends almost exclusively on the solver.
- The second is given by a 3D model of the crankcase of a $V8$ ICE (Internal Combustion Engine). This surface, shown in Figure 3, is chosen for its complexity since it presents different features: there are both regular surfaces (cylindrical and flat) as well as sharp and rounded edges. If the integration between node generator and solver is capable of providing reliable results on this scenario, then it is safe to conclude its applicability to many other shapes of engineering relevance.

The quality of the generated node distributions can be assessed visually, for , by carefully inspecting the plotted result. For instance, one might check whether the normal vectors associated to the surface nodes are correctly oriented, especially in correspondence with the edges or vertices at the intersection between different triangles of the stereolithography.

Another visual test is to check whether there are inner nodes too close to the surface since such nodes repel those belonging to the boundary, thus leading to a suboptimal enforcement of the boundary conditions.

We also point out that, for simplicity, the spacing function s is always set to be uniform for all the results presented in this paper. This is not an issue for regular domains, like the sphere, but might harm the accuracy when the geometry presents some strong 3D curvature localized at certain spots, for example in the case of the crankcase domain.

3.4. RBF-FD Parameters

The achieved performances mainly depend on four parameters:

- $\tilde{n}$, number of nodes included in each stencil, see Equation (5),
- $\bar{\varepsilon}$, shape parameter of the Radial Basis Function, see Equation (4),
- N, total number of nodes generated on both Ω and $\partial\Omega$,
- P, degree of the polynomial term in the definition of u^h, see Equation (5).

The number of nodes included in each stencil $\tilde{n}$ is linked to P in the sense that a minimum number of nodes is required for any given order of the polynomial term. It has been shown that the choice of larger stencils, like $\bar{n} = 2m$ given in Equation (33), overcomes stability and accuracy problems due to Runge's phenomenon [17]:

$$\bar{n} = 2m = 2\binom{P+d}{P} \tag{33}$$

where m is the number of terms in the polynomial basis function, and d is the number of dimensions ($d = 3$ in 3D problem).

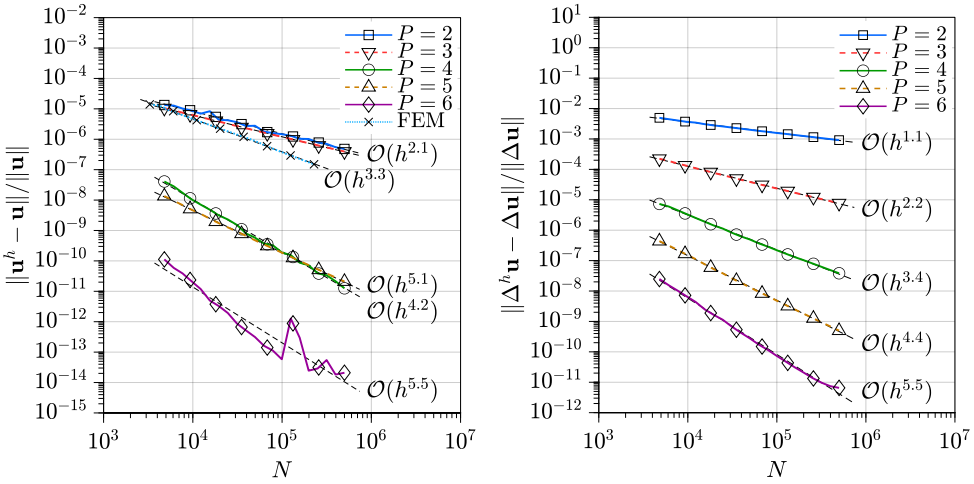
The shape parameter $\bar{\varepsilon}$ defines the flatness of the Radial Basis Function (RBF) reported in Equation (4); its value must be selected carefully due to the following trade-off:

- in the limit $\bar{\varepsilon} \longrightarrow 0$ the RBF becomes increasingly flat, until some typical oscillations appears due to the Runge's phenomenon [28], on top of that, too little values of $\bar{\varepsilon}$ might also lead to numerical issues, due to an ill-conditioning of the interpolation matrix,
- as $\bar{\varepsilon}$ increases, on the other side, the solution becomes more stable but less accurate.

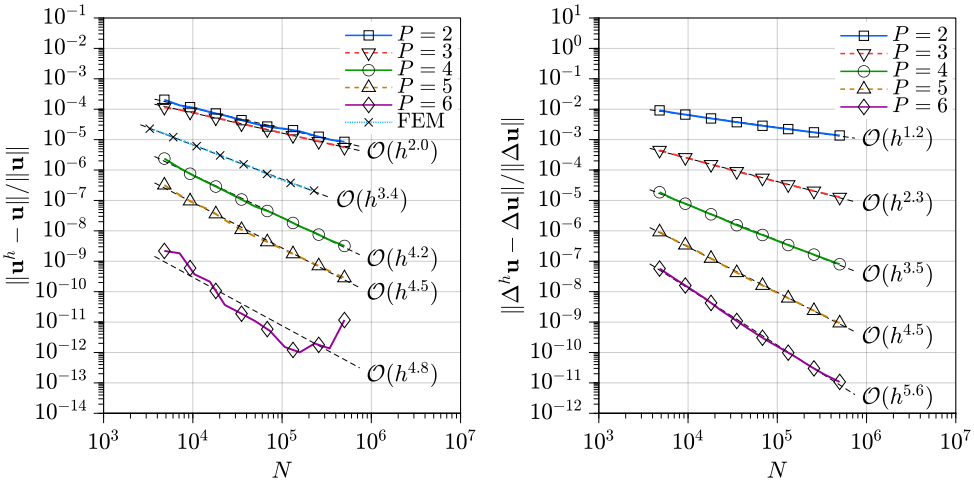
A reasonable choice based on previous numerical studies [21,28,42] is $\bar{\varepsilon} = 0.3$, while the influence of parameters N and P is instead discussed in succeeding paragraphs, since they are strongly related to the order of accuracy.

3.5. Sphere

In Figure 4 the convergence curves for the solution (Figure 4a,c) and for the Laplacian (Figures 4b,d) are shown for polynomial degrees $P = 2, \ldots, 6$ on the spherical domain, where the total number of nodes ranges from $N = 5$k to $N = 500$k nodes. Dirichlet b.c., i.e., $a = 1$ and $b = 0$ in Equation (2), and Robin b.c., i.e., $a = 1$ and $b = 1$, have been considered. Errors are normalized with respect to the norm of analytical values.



(**a**) Solution error, Dirichlet b.c. (**b**) Discretization error, Dirichlet b.c.



(**c**) Solution error, Robin b.c. (**d**) Discretization error, Robin b.c.

Figure 4. Convergence curves for solution and Laplace operator (or Discretization Error) for Dirichlet and Robin b.c., Ω: Sphere.

Errors in the approximation of the Laplacian operator, presented in Figure 4b,d , are simply attained as the result of the matrix-vector product reported in Equation (34), where $\boldsymbol{C}$ is the sparse matrix from (30) and $\boldsymbol{u}$ is the analytical solution evaluated at the inner nodes. Therefore Figure 4b,d represents an estimation of the mean discretization error, calculated as

$$\{\triangle^h \boldsymbol{u}\} := [\boldsymbol{C}]\{\boldsymbol{u}\} - \{\boldsymbol{q}\} \tag{34}$$

Figure 4a shows an increase in the order of accuracy, which ranges from $\sigma = 2.2$ to $\sigma = 5.5$, when the polynomial degree P is increased from $P = 2$ to $P = 6$. The order of accuracy σ is defined by the following equation:

$$error \propto h^{\sigma} \tag{35}$$

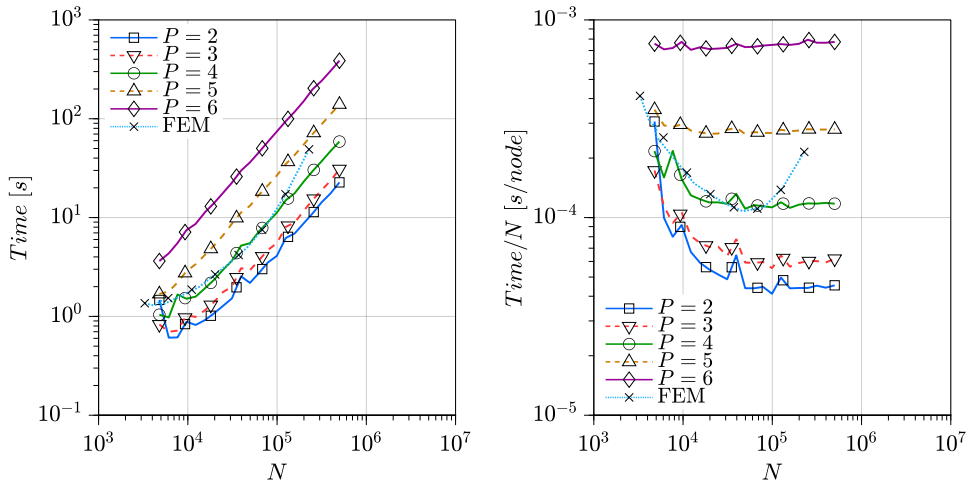
where $h \propto N^{-1/3}$ is a reference value for the nodal spacing in 3D.

Figure 4c shows slightly lower σ's when Robin b.c. are enforced, even if present, the degradation in performance with these boundary conditions can still be considered more than acceptable. We believe that the apparent lack of convergence appearing in the case $P = 6$ for $N > 300$k nodes is due to numerical instabilities caused by the large size of the local stencils when Robin boundary nodes are included. Indeed, with $P = 6$ quite a large local stencil is employed, i.e., $\bar{n} = 168$ nodes, leading to an ill-conditioned dense local matrix $\boldsymbol{M}$. However, this is not a problem since $P = 6$ is a limit case in 3D, and such instabilities seem to only play a non-negligible role once the method has come to convergence near machine epsilon in the global error.

From Figure 4a,c it can also be observed that it is always convenient to choose a polynomial basis of even degree P in order to solve a Poisson equation, which is a second-order PDE: odd degrees not only provide negligible improvements in terms of solution errors, but even lead to worse orders of accuracy. Furthermore, we point out that, in the case $P = 6$ and $N > 200$k nodes, the normalized error becomes quite close to machine epsilon in double precision, and any further increase in N will not reduce the error. FEM convergence curves are also displayed in Figure 4a,c for comparison; it can be observed that the FEM curve is very similar to the ones for the cases $P = 2, 3$ for Dirichlet b.c., while for Robin b.c. the FEM curve lies between the cases $P = 3$ and $P = 4$.

The convergence curves represented in Figure 4b,d show a continuous improvement in both the order of accuracy σ and absolute value of the discretization error when P is increased, and they seem to follow the rule of thumb $\sigma \approx 1.1 \times (P - 1)$.

Figure 5 can instead be used to compare the different choices of P under the aspect of time efficiency. Figure 5a gives a rough idea of the actual computational time required for the calculation of the solution u^h. The values are referred to a portable computer equipped with Intel *i*7-6700 *HQ* CPU (4 cores) and 16 GB of RAM. The total computational time accounts for the time required for node generation, calculation of RBF-FD stencil, Equation (28), and final sparse linear system solution, Equation (30). Both node generation and RBF-FD stencils calculations are explicitly parallelized in `Julia` using `Distributed` package and `Base.Threads` module for multithreaded computations distributed over multiple cores.



(**a**) Total Time (**b**) Time per Node

Figure 5. Total time required for the final solution, Ω: sphere, Dirichlet b.c.

Figure 5b shows instead the specific total times in terms of seconds per node, where it can be seen that, whatever the value of P, the specific time tends to constant values for sufficiently large N, where the total time is dominated by node generation, and RBF-FD stencils calculations which are perfectly linear in N. As N increases further, the specific time shows a slight increase due to the asymptotic cost of the ILU-preconditioned BiCGStabl algorithm, which scales more than linearly. The total and specific computational times required for the FEM solution are also displayed in Figure 5, showing a significant increase beyond $N = 100$k nodes.

Figure 6 shows an analysis of the time efficiency allowed by each choice of the polynomial degree P. For better visualization only the results attained with more than 20k nodes are included. It is possible to see that, when the geometry is sufficiently smooth as the spherical one, choosing a higher even polynomial degree is always advisable for the solution of a Poisson equation. Figure 6 also shows that the time efficiency of the FEM approach is very similar to the cases $P = 2, 3$.

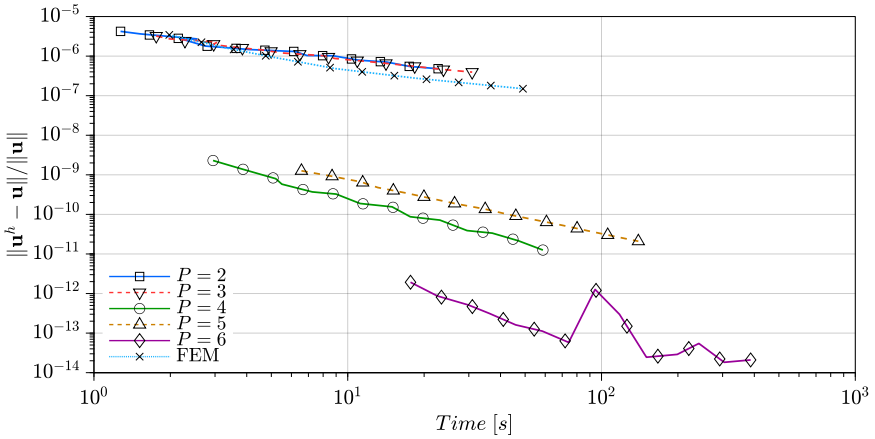
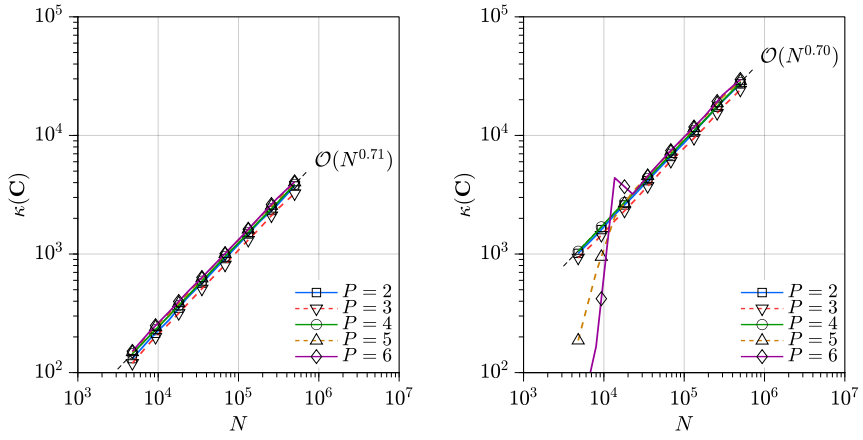


Figure 6. Error vs time (the lower the better), allowed by different values of P, Ω: sphere, b.c.: Dirichlet, $N >$ 20k nodes.

The estimation of the condition number $\kappa(\mathbf{C})$ of the final sparse matrix $\mathbf{C}$ is shown in Figure 7 for both Dirichlet and Robin b.c., where $\kappa(\mathbf{C})$ is defined as $\kappa(\mathbf{C}) = \|\mathbf{C}\|_2\|\mathbf{C}^{-1}\|_2$ using the 2-norm. The condition number grows approximately with order $\mathcal{O}(N^{0.7})$ in both cases, regardless of P, thus very much comparable with the order $\mathcal{O}(N^{2/3})$ of classic finite difference schemes for the 3D Laplacian operator on uniform grids [43]. The condition number for Robin b.c. ($a = b = 1$ in Equation (2)), Figure 7b, is approximately 7 times larger than the condition number for Dirichlet b.c., Figure 7a, which is again comparable to the value 5.78 of classic finite difference schemes with the same b.c.

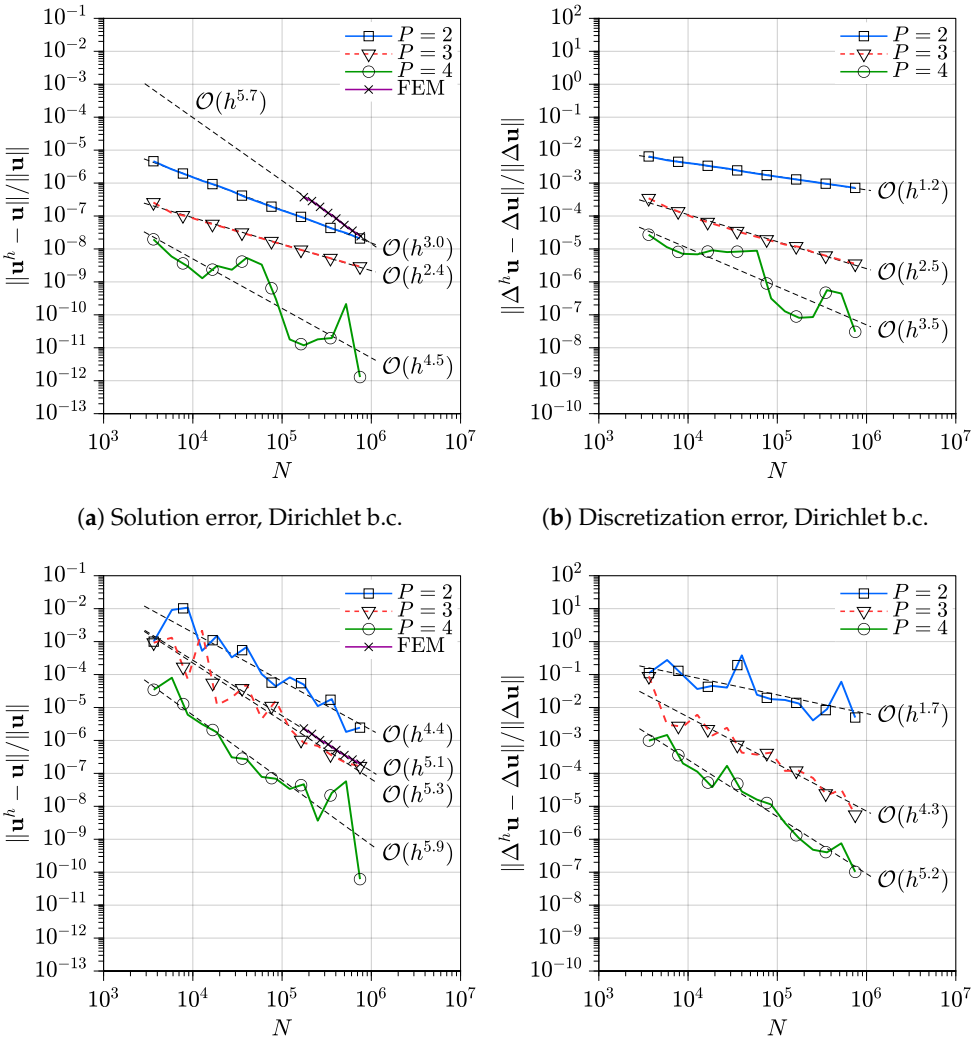


(**a**) Dirichlet b.c (**b**) Robin b.c

Figure 7. Condition number of the sparse matrix $\mathbf{C}$, Ω: sphere.

3.6. Engine

Convergence curves for the solution and for the Laplacian are shown in Figure 8, again for both Dirichlet b.c., i.e., $a = 1$ and $b = 0$ in Equation (2), and Robin b.c., i.e., $a = 1$ and $b = 1$. The chosen values for the polynomial degree are $P = 2, 3, 4$, and the total number of nodes ranges from $N = 5$k to $N = 750$k nodes.



(**a**) Solution error, Dirichlet b.c.

(**b**) Discretization error, Dirichlet b.c.

(**c**) Solution error, Robin b.c.

(**d**) Discretization error, Robin b.c.

Figure 8. Convergence curves for solution and Laplace operator (or discretization error) for Dirichlet and Robin b.c., Ω: Engine.

In the case of Dirichlet b.c., the order of accuracy for the solution, Figure 8a, varies from $\sigma = 3.0$ to $\sigma = 4.5$ when the polynomial degree is increased from $P = 2$ to $P = 4$. Similarly to the sphere, passing from $P = 2$ (even) to $P = 3$ (odd) does not increase the order of accuracy, which instead decreases from $\sigma = 3.0$ to $\sigma = 2.4$, while an improvement in absolute terms appears. A further increase to $P = 4$ (odd) results in both increased order of accuracy, i.e., $\sigma = 4.5$, and reduced errors in absolute terms. FEM convergence curves are also shown in Figure 8a, revealing an order of accuracy $\sigma = 5.7$, which is significantly higher than the ones for $P = 2, 3, 4$, but the FEM error has larger absolute value than $P = 2$ below $N = 1$M nodes.

The convergence curves for the Laplacian, Figure 8b, show instead a continuous increase in the order of accuracy when P grows, analogously to the corresponding results on the spherical domain. Unlike the case of the sphere, the convergence curves for $P = 4$ in both the previous figures exhibit some irregularities compared to the curves for

$P = 2, 3$. This is probably due to the fact that the large stencil, required by the condition of Equation (33) in the case $P = 4$, i.e., $\bar{n} = 70$ nodes, leads to very distorted 3D nodal arrangements in the proximity of particular features of the complex surface, giving rise to unstable RBF interpolants. This problem, which is mainly due to the employment of a constant spacing function, can be alleviated through some proper refinement of the node distribution, for instance some reduction in the nodal spacing in proximity to such geometrical features.

In the case of Robin b.c., it has been necessary to impose three additional requirements for the inclusion of boundary nodes in the stencil:

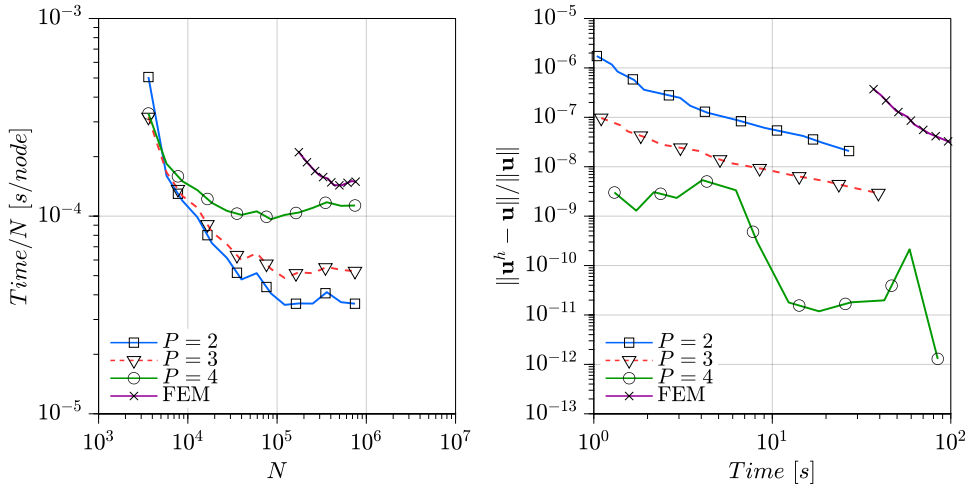
- Angle rule:

$$\frac{x_b - x_i}{\|x_b - x_i\|} \cdot n(x_b) > 0.6 \tag{36}$$

  where x_b is a boundary node and x_i is an inner node where the Laplacian is required to be approximated. Equation (36) states that a boundary node x_b can be included in the stencil only if the angle γ between the outer normal $n(x_b)$ associated to the boundary node x_b and the line connecting x_b to x_i satisfies $\gamma < \arccos(0.6) \approx 53°$.
- Distance rule: $\|x_b - x_i\| < 2s$, i.e., only boundary nodes closer to the center x_i than twice the spacing function can be included in the stencil.
- Number rule: no more than 10 boundary nodes can be included in the stencil.

The resulting convergence curves for the solution and for the Laplacian are shown in Figure 8c,d, respectively. The comparison with the corresponding curves for Dirichlet b.c., Figure 8a, shows a general loss in accuracy in absolute terms due to the effects of Robin b.c., i.e., the curves are shifted upwards, while the apparent order of accuracy for the solution ranges from $\sigma = 4.4$ to $\sigma = 5.9$ when the polynomial degree is increased from $P = 2$ to $P = 4$. FEM results are very much comparable to the case $P = 3$ in terms of both absolute error and order of accuracy. The convergence curves for the Laplacian highlight a similar behavior, with a continuous improvement in the order of accuracy when P is increased. The graphical comparison between the convergence curves of Dirichlet b.c. and Robin b.c. also reveals a higher sensitivity to boundary node arrangements for the latter case, as expected. Again, local node refinement in the neighborhood of particular geometrical features of the boundary should mitigate this problem.

Figure 9 shows the analysis of computational times. The curves for the specific time, i.e., time per node, depicted in Figure 9a, are very similar to the ones obtained for the spherical domain, depicted in Figure 5b. The presented approach is thus almost perfectly linear in the range N = 100k–1M nodes; the specific times for the FEM solution are instead always slightly larger than the case $P = 4$.



(**a**) (**b**)

Figure 9. Time per node (**a**) and error vs. time (**b**), Ω: Engine, Dirichlet b.c.

Figure 9b shows the curves of the solution error versus computational time in the case of Dirichlet b.c. These curves reveal that it is always convenient to employ high-degree polynomials, e.g., $P = 4$, also in this case where a very complex-shaped domain is subjected to analysis. Figure 9b shows that the FEM approach appears to be the less efficient. Indeed it required longer times than $P = 4$, as can be seen in Figure 9a, still not being able to further reduce the corresponding errors, see Figure 8a.

4. Conclusions

In this work a fully meshless approach is applied to the simulation of heat conduction problems. The code is verified according to the Method of Manufactured Solutions, and it is proven to be reliable and suitable for practical applications, even with convective boundary conditions. Indeed, the results presented in this paper exceeded the most optimistic expectations, especially on complex domains.

The potential of the RBF-FD meshless method lies both in its geometrical flexibility and its inclination towards parallelization: the root of its advantages can be found in the lack of connectivity information between the nodes.

As many simulations were carried over it became apparent that a key role was played by the node generation algorithm and the nearest-neighbors search algorithm. Different implementations of these routines yielded very different results when the code was verified over the engine domain. The proposed node generator allowed to properly leverage the great flexibility of the RBF-FD method, favoring performances over complex domains very much comparable to the ones attained on the spherical domain, both in terms of accuracy and consistency. Very interestingly such improvements were reached without any measurable increase in the computational cost of the whole solution procedure.

Speaking of computational cost, from the comparison between different polynomial basis it emerged an advantage of the ones with higher degrees when error vs. time was considered. We remark that most mesh-based commercial CFD CAE software only allow second-order accuracy and therefore cannot take advantage of the gain in performance permitted by higher orders. Such an advantage seems somehow reduced when Robin b.c. are applied to complex geometries; however, the results exposed above were attained with uniform spacing between nodes. Great improvements in the results are expected by the application of some local refinement algorithm, as usually employed also in mesh-based solvers. Such a refinement routine would be required to generate more inner nodes in close proximity to those geometrical features where the available space is otherwise insufficient, thus providing better RBF interpolants. Nonetheless, the presented meshless approach allowed better performances in terms of both absolute errors and computational efficiency when compared with patch-conforming quadratic FEM, highlighting the great advantages of the employed RBF-FD approach.

`Julia` language proved to be very convenient and highly efficient at the same time, allowing extensive code reuse and making parallelization fairly easy. Furthermore, the use of a single programming language ensured better integration between the node generator and the solver and made it possible to measure the total computational times reported in the results.

Future developments in the short term are the implementation of an autonomous node-refinement algorithm and an algebraic multilevel solver. Finally, the simulation of problems based on Navier–Stokes equations is planned for the near future.

Overcoming the need to generate a mesh constitute a great advantage over traditional CAE software, potentially making the design process far more convenient and democratic. This is why, also encouraged by the promising results reported in the present work, we believe that RBF-FD method will rapidly grow in popularity and become a major resource in the years to come.

Author Contributions: Conceptualization, R.Z.; Data curation, D.M.; Formal analysis, R.Z.; Funding acquisition, E.N.; Methodology, R.Z.; Project administration, E.N.; Software, R.Z.; Visualization, D.M.;

Writing—original draft, D.M.; Writing—review and editing, E.N. and R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Versteeg, H.K.; Malalasekera, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*; Pearson Education: Upper Saddle River, NJ, USA, 2007.
2. Zienkiewicz, O.C. *The Finite Element Method*; McGraw-Hill: London, UK, 1977; Volume 3.
3. Liu, G.R.; Gu, Y.T. *An Introduction to Meshfree Methods and Their Programming*; Springer Science & Business Media: Boston, NY, USA, 2005.
4. Zamolo, R.; Nobile, E.; Šarler, B. Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations. *J. Comput. Phys.* **2019**, *392*, 311–334, doi:10.1016/j.jcp.2019.04.064.
5. Liu, G.R. *Meshfree Methods: Moving beyond the Finite Element Method*; Taylor & Francis: Boca Raton, FL, USA, 2009.
6. Li, H.; Mulay, S.S. *Meshless Methods and Their Numerical Properties*; CRC Press: Boca Raton, FL, USA, 2013.
7. Zamolo, R.; Nobile, E. Solution of incompressible fluid flow problems with heat transfer by means of an efficient RBF-FD meshless approach. *Numer. Heat Transf. Part B Fundam.* **2019**, *75*, 19–42, doi:10.1080/10407790.2019.1580048.
8. Shojaei, A.; Mossaiby, F.; Zaccariotto, M.; Galvanetto, U. A local collocation method to construct Dirichlet-type absorbing boundary conditions for transient scalar wave propagation problems. *Comput. Methods Appl. Mech. Eng.* **2019**, *356*, 629–651, doi:10.1016/j.cma.2019.07.033.
9. Shojaei, A.; Hermann, A.; Seleson, P.; Cyron, C.J. Dirichlet absorbing boundary conditions for classical and peridynamic diffusion-type models. *Comput. Mech.* **2020**, *66*, 773–793, doi:10.1007/s00466-020-01879-1.
10. Fornberg, B.; Flyer, N. *A Primer on Radial Basis Functions with Applications to the Geosciences*; SIAM: Philadelphia, PA, USA, 2015; doi:10.1137/1.9781611974041.
11. Fornberg, B.; Flyer, N. Solving PDEs with radial basis functions. *Acta Numer.* **2015**, *24*, 215–258, doi:10.1017/S0962492914000130.
12. Divo, E.; Kassab, A.J. An efficient localized radial basis function meshless method for fluid flow and conjugate heat transfer. *J. Heat Transf.* **2007**, *129*, 124–136, doi:10.1115/1.2402181.
13. Šarler, B.; Vertnik, R. Meshfree explicit local radial basis function collocation method for diffusion problems. *Comput. Math. Appl.* **2006**, *51*, 1269–1282, doi:10.1016/j.camwa.2006.04.013.
14. Waters, J.; Pepper, D.W. Global Versus Localized RBF Meshless Methods for Solving Incompressible Fluid Flow with Heat Transfer. *Numer. Heat Transf. Part B Fundam.* **2015**, *68*, 185–203, doi:10.1080/10407790.2015.1021590.
15. Kosec, G.; Slak, J. Radial basis function-generated finite differences solution of natural convection problem in 3D. *AIP Conf. Proc.* **2020**, *2293*, 420094, doi:10.1063/5.0027289.
16. Wang, L. Radial basis functions methods for boundary value problems: Performance comparison. *Eng. Anal. Bound. Elem.* **2017**, *84*, 191–205, doi:10.1016/j.enganabound.2017.08.019.
17. Bayona, V.; Flyer, N.; Fornberg, B.; Barnett, G.A. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.* **2017**, *332*, 257–273, doi:10.1016/j.jcp.2016.12.008.
18. Flyer, N.; Fornberg, B.; Bayona, V.; Barnett, G.A. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *J. Comput. Phys.* **2016**, *321*, 21–38, doi:10.1016/j.jcp.2016.05.026.
19. Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V.B. Julia: A fresh approach to numerical computing. *SIAM Rev.* **2017**, *59*, 65–98, doi:10.1137/141000671.
20. Brezis, H. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*; Springer Science & Business Media: Boston, NY, USA, 2010.
21. Zamolo, R.; Parussini, L. Analysis of geometric uncertainties in CFD problems solved by RBF-FD meshless method. *J. Comput. Phys.* **2020**, *421*, 109730, doi:10.1016/j.jcp.2020.109730.
22. van der Sande, K.; Fornberg, B. Fast Variable Density 3-D Node Generation. *SIAM J. Sci. Comput.* **2021**, *43*, A242–A257, doi:10.1137/20M1337016.
23. Slak, J.; Kosec, G. On Generation of Node Distributions for Meshless PDE Discretizations. *SIAM J. Sci. Comput.* **2019**, *41*, A3202–A3229, doi:10.1137/18M1231456.
24. Duh, U.; Kosec, G.; Slak, J. Fast variable density node generation on parametric surfaces with application to mesh-free methods. *arXiv* **2020**, arxiv:2005.08767.

25. Duh, U.; Depolli, M.; Slak, J.; Kosec, G. Parallel point sampling for 3D bodies. In Proceedings of the 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 28 September–2 October 2020; pp. 219–223, doi:10.23919/MIPRO48935.2020.9245354.
26. Fornberg, B.; Flyer, N. Fast generation of 2-D node distributions for mesh-free PDE discretizations. *Comput. Math. Appl.* **2015**, *69*, 531–544, doi:10.1016/j.camwa.2015.01.009.
27. Zamolo, R.; Nobile, E. Two algorithms for fast 2D node generation: Application to RBF meshless discretization of diffusion problems and image halftoning. *Comput. Math. Appl.* **2018**, *75*, 4305–4321, doi:10.1016/j.camwa.2018.03.031.
28. Zamolo, R. Radial Basis Function-Finite Difference Meshless Methods for CFD Problems. Ph.D. Thesis, Università degli Studi di Trieste, Trieste, Italy, 2019.
29. Frey, P.; George, P. *Mesh Generation: Application to Finite Elements*; ISTE, Wiley: London, UK, 2013.
30. De Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M. *Computational Geometry: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2008.
31. Hardy, R.L. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.* **1971**, *76*, 1905–1915, doi:10.1029/jb076i008p01905.
32. Šarler, B. From Global to Local Radial Basis Function Collocation Method for Transport Phenomena. In *Advances in Meshfree Techniques*; Leitao, V.M.A., Alves, C.J.S., Armando Duarte, C., Eds.; Springer: Dordrecht, The Netherlands, 2007; pp. 257–282, doi:10.1007/978-1-4020-6095-3_14.
33. Yao, G.; Siraj-ul-Islam.; Šarler, B. A Comparative Study of Global and Local Meshless Methods for Diffusion-Reaction Equation. *CMES Comp. Model. Eng.* **2010**, *59*, 127–154, doi:10.3970/cmes.2010.059.127.
34. Yao, G.; Siraj-ul-Islam.; Šarler, B. Assessment of global and local meshless methods based on collocation with radial basis functions for parabolic partial differential equations in three dimensions. *Eng. Anal. Bound. Elem.* **2012**, *36*, 1640–1648, doi:10.1016/j.enganabound.2012.04.012.
35. Wang, L.; Qian, Z. A meshfree stabilized collocation method (SCM) based on reproducing kernel approximation. *Comput. Methods Appl. Mech. Eng.* **2020**, *371*, 113303, doi:10.1016/j.cma.2020.113303.
36. Hu, H.; Chen, J.; Hu, W. Weighted radial basis collocation method for boundary value problems. *Int. J. Numer. Meth. Eng.* **2007**, *69*, 2736–2757, doi:10.1002/nme.1877.
37. Saad, Y. Iterative Methods for Sparse Linear Systems. In *Iterative Methods for Sparse Linear Systems. Chapter 10: Preconditioning Techniques*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2003; pp. 297–368.
38. Van der Vorst, H. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.* **1992**, *13*, 631–644, doi:10.1137/0913035.
39. Bezanson, J.; Karpinski, S.; Shah, V.B.; Edelman, A. Julia: A fast dynamic language for technical computing. *arXiv* **2012**, arxiv:1209.5145.
40. MathWorks®. Partial Differential Equation Toolbox™: User's Guide (R2020b). 2020. Available online: https://www.mathworks.com/help/pdf_doc/pde/pde.pdf (accessed on 18 February 2021).
41. Roache, P.J. Code verification by the method of manufactured solutions. *J. Fluids Eng.* **2002**, *124*, 4–10, doi:10.1115/1.1436090.
42. Zamolo, R.; Parussini, L. Geometric uncertainty propagation in laminar flows solved by RBF-FD meshless technique. *J. Phys. Conf. Ser.* **2020**, *1599*, 012045, doi:10.1088/1742-6596/1599/1/012045.
43. Gregory, R.; Karney, D. *A Collection of Matrices for Testing Computational Algorithms*; Wiley-Interscience: New York, NY, USA, 1969.