POLITECNICO DI TORINO
VON KARMAN INSTITUTE FOR FLUID DYNAMICS

Master Course
In Aerospace Engineer

Master thesis

A Meshless Method to Measure Pressure Fields from Image Velocimetry via Radial Basis Functions

Supervisors:
Prof. Pieraccini Sandra
Prof. Mendez A. Miguel

Candidate:
Sperotto Pietro
Matriculation no. S263648

Academic Year 2020/2021

Contents

1 Introduction **3**
- 1.1 Motivations and aims . . . 3
- 1.2 Methodology and structure of the Thesis . . . 3

2 Image velocimetry and pressure computation **5**
- 2.1 Particle Image Velocimetry . . . 5
- 2.2 Particle Tracking Velocimetry . . . 8
- 2.3 Pressure computation . . . 8

3 From function approximation to PDE integration via RBF **13**
- 3.1 Radial Basis Functions fundamentals . . . 13
- 3.2 Function approximation using RBF . . . 15
- 3.3 Constrained RBF function approximation . . . 16
- 3.4 Solving Poisson equation . . . 18

4 Implementation of the meshless pressure evaluation **19**
- 4.1 Including physical constraints and penalties . . . 19
- 4.2 Defining collocation points and shape parameter . . . 21
- 4.3 Solving the linear system . . . 22
  - 4.3.1 Efficient method to identify a suitable regularization . . . 22
  - 4.3.2 Solving the linear system with standard Numpy routine . . . 23
  - 4.3.3 Penrose-Moore Pseudo Inverse . . . 23
  - 4.3.4 Pseudo Inverse mixed with Cholesky . . . 24
  - 4.3.5 Full Cholesky . . . 24
  - 4.3.6 Comparison . . . 24
- 4.4 Pressure evaluation algorithm . . . 25
- 4.5 Python implementation . . . 25
  - 4.5.1 `Matrix.py` . . . 25
  - 4.5.2 `MESH_LAB.py` . . . 25

5 Test Cases **31**
- 5.1 Gaussian Vortex . . . 31
- 5.2 Cylinder at Re=5 . . . 32

6 Results **33**
- 6.1 Vortex . . . 33
  - 6.1.1 Vortex with standard collocation . . . 33
  - 6.1.2 Clustering on Vortex . . . 37
- 6.2 Cylinder at Re=5 . . . 40
  - 6.2.1 Cylinder with standard collocation . . . 40

6.2.2 Cylinder with clustering method 44

7 Conclusion and perspectives **47**

A Theorems **49**

List of Figures

2.1 PIV standard setup (Raffel et al. (2018)). 5
2.2 Operating modes (Discetti and Ianiro (2017)). 6
2.3 Example of a couple of PIV images. 7
2.4 Left to right: a small sample I of 32x32 pixel is compared to a larger sample I' of 64x64 pixel; the last figure reports the cross-correlation plane where is evident that the displacement is of 12 pixels. (Raffel et al. (2018)). 7
2.5 Flow chart implementation convolution theorem (Raffel et al. (2018)). 8
2.6 Example of omnidirectional integration (Liu and Katz (2006)). 10
2.7 Behaviour of Lagrangian and Eulerian methods (De Kat and Van Oudheusden (2012)). 11

3.1 Parameter influence on RBF (Gaussian). 14
3.2 Example of function approximation via RBF. 16
3.3 Example of constrained function approximation via RBF. 17

4.1 Flow chart of the clustering method. 27
4.2 Flow chart of the function approximator solver. 28
4.3 Flow chart of the boundary condition function. 29
4.4 Flow chart of the Poisson solver function. 30

5.1 Geometries of the flow, Rao et al. (2020). 32

6.1 Velocity approximation. 34
6.2 Pressure comparison. 35
6.3 Error with varying particle N=[1000, 2500, 5000, 7500, 10000, 15000, 20000], noise=0%. 36
6.4 Fourier transform of the approximation and the original data with different c's. 36
6.5 Effect of regularization on the spectra. 37
6.6 Error with varying Noise=[0%,1%,2%,5%,10%,20%], N=20000. 37
6.7 Forcing Terms. 38
6.8 Error(Clustering) with varying particle N=[1000, 2500, 5000, 7500, 10000, 15000, 20000], noise=0%. 39
6.9 Error(Clustering) with varying Noise=[0%,1%,2%,5%,10%,20%], N=20000. 40
6.10 Velocity in x direction, Rao et al. (2020). 41
6.11 Velocity in y direction, Rao et al. (2020). 41
6.12 Pressure, Rao et al. (2020). 41
6.13 Velocity in x direction. 42
6.14 Velocity in y direction. 42
6.15 Comparisons between computed and real velocities. The black line has a 45^o slope and represent the approximation with zero error. The further the blue points are from the black line worse is the computed value. 43
6.16 u at different y behind the cylinder. 43

6.17 Pressure field of the cylinder. 44
6.18 Pressure comparison. 44
6.19 Pressure around the cylinder. 45
6.20 Boundary condition around the circle, 0^o is the stagnation point which faces the inlet. 46
6.21 Pressure around the cylinder. 46

Abstract

In the past years the interest in computing pressure from image velocimetry has increased. However, only few researchers tried to implement meshless methods (see Gesemann et al. (2016)). Such techniques appear to be increasingly promising following the recent advances in both computational power and machine learning algorithms.

Furthermore, no meshless methods exploiting Radial Basis Functions have been found in the literature, at least applied in this context. The meshless method proposed here adds information from fluid dynamics theory directly to the data, using the former to correct the latter. The core of this thesis is to develop the mathematical basis of such method and to implement and test it. Two test cases are presented, with the objective of validating both theory and the implementation of this work. The obtained results show some interesting features of the method, which well treats noisy data, for instance.

This method is still at an early stage, and there is surely large margin for improvement

Chapter 1

Introduction

1.1 Motivations and aims

Pressure is one of the most relevant quantities describing a flow in fluid dynamics. Together with the velocity field, this quantity enables the computation of aerodynamic forces exerted on objects. Therefore pressure is an important flow quantity when sizing airfoils, wings and any structure subject to aerodynamic loadings.

Pressure can be measured using a variety of probes (e.g. Pitot or Prandtl tubes) and pressure transducers (piezoeletric or capacitance). However, these methods only provide discrete measurements only in the point where the probe is placed, and interfere with the flow during the experiment.

On the other hand, the velocity field can be measured non-intrusively using Particle Image Velocimetry (PIV) and Particle Tracking Velocimetry (PTV). Leveraging the high spatial resolution available from these measurements, Gurka et al. (1999) proposed to compute pressure fields from PIV/PTV velocity fields, hence enabling non-intrusive measurements of aerodynamic loading. Since then, many researchers have investigated techniques to refine these methods in different directions. When treating velocity measurements from PTV or PIV, velocity fields can be interpolated onto a new computational grid, and then used to solve the governing equations with standard PDE solvers to retrieve pressure fields.

These methods suffer from high sensitivity to measurement noise, and are especially challenging because they require a computational grid. The definition of such grid, together with the related interpolation step, can become particularly cumbersome in the presence of complex geometries such as airfoils or wings, as the velocity measurements might not be available sufficiently close to the solid boundaries.

This thesis aims at proposing a methodology that circumvents both of the aforementioned problems. The goal is to develop a method that allows computing an analytic approximation of the velocity field. This allows for mitigating noise, enables analytical derivatives of the velocity fields, and allows for solving PDEs in a meshless approach. Moreover, it was of interest to complement the approach with a method that could add physical penalties and constraints.

1.2 Methodology and structure of the Thesis

This work consists of seven chapters, including this introduction. This structure follows the logical path through which this work has been developed.

Initially, the project aimed at developing a toolbox to compute PIV-based pressure fields using standard methodologies proposed in the literature. Then, studying Radial Basis Functions (RBF) as possible function approximators, it was discovered that these tools could also be used to solve PDEs while easily accommodating boundary conditions in the form of regression constraints. Because the PDE solution can be cast in the form of a function approximator, the solution strategy turned out to be mesh-free. Moreover, these tools could also

be used to super-sample the original data while imposing physical constraints, thus enhancing the robustness of the approximation.

Next, this technique has to be exploited for the problem of pressure evaluation, analyzing the best constraints and penalizations to add. After refining the theory behind this technique, a Python implementation was proposed. Finally, some test cases were chosen to validate the code, and encouraging results were obtained.

This thesis closes with a discussion regarding the future development of this ongoing project.

Chapter 2

Image velocimetry and pressure computation

In this chapter, image velocimetry techniques are briefly recalled and the focus on the state of the art for pressure integration in this field is then sketched.

2.1 Particle Image Velocimetry

The standard setup for Particle Image Velocimetry (PIV) is presented in figure 2.1.

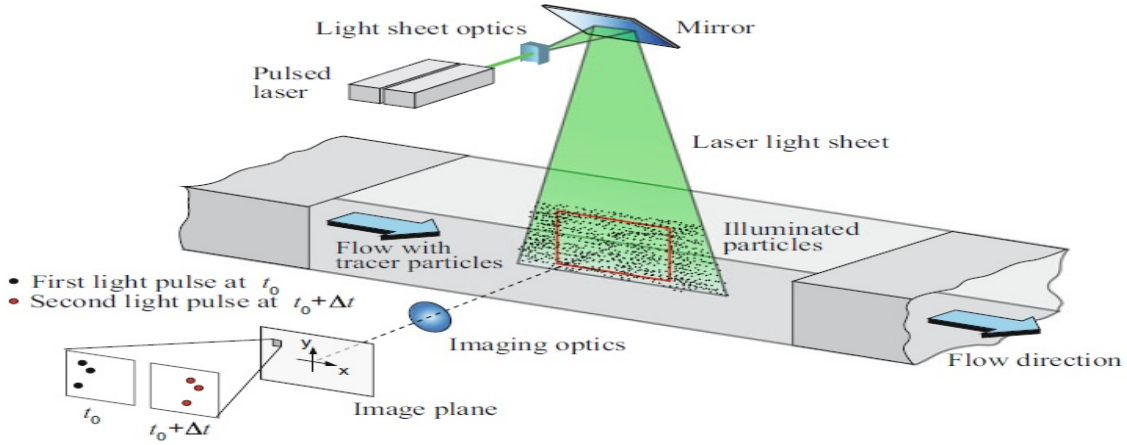


Figure 2.1: PIV standard setup (Raffel et al. (2018)).

Two main devices distinguish PIV (or in general image velocimetry) from other techniques, namely a laser and a camera. The laser creates a light sheet that hits the flow. This light illuminates some particles, which scatter the light. The light is then captured by a high-quality camera.

PIV has been made possible by the recent improvements of cameras that allow for capturing frames with a frequency of kHz. Charge Coupled Device (CCD) cameras are capable of capturing more than 100 PIV recording per minute. However, the highest frequencies possible to capture are reached with Complementary Metal-Oxide Semiconductor (CMOS).

The acquired images are 2D projections of the volume imaged by a series of lens optics. The coordinate in the imaged volume is referred to as $\mathbf{x}$ and on the planar sensor as $\mathbf{X}$. Usually the following relationship is assumed to hold:

$$\begin{pmatrix} X \\ Y \end{pmatrix} \approx M \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.1}$$

where $M \in \mathbb{R}^+$ is the magnification.

There are several operating modes for a camera, presented in figure 2.2.

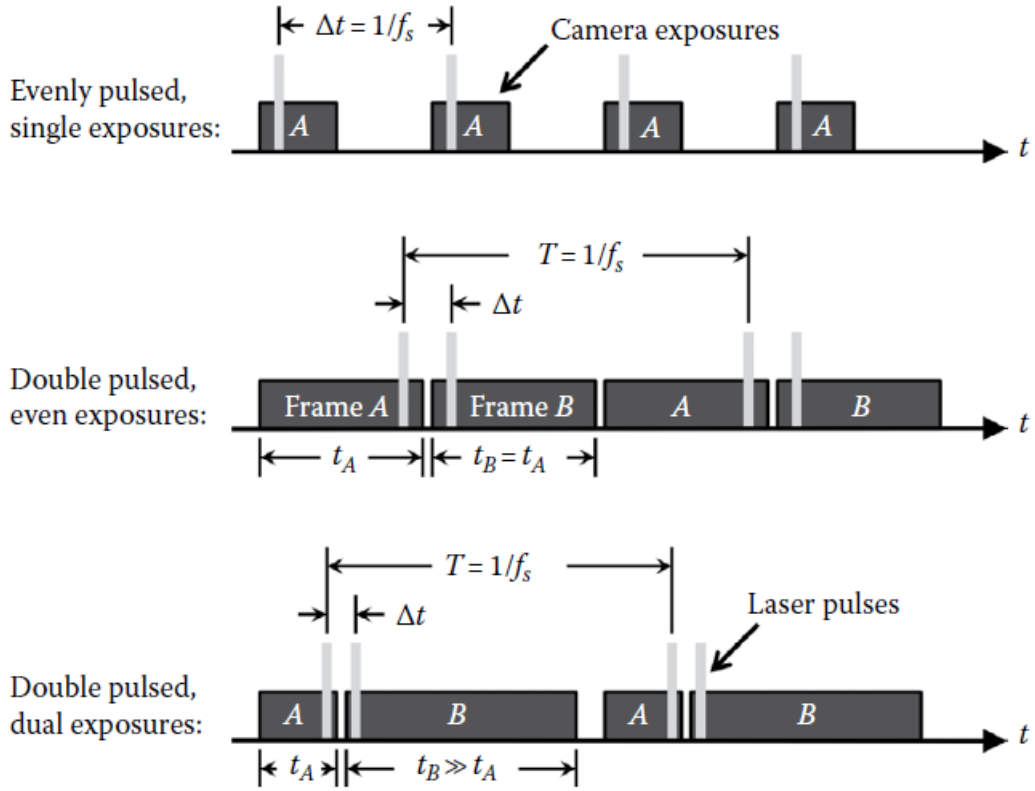


Figure 2.2: Operating modes (Discetti and Ianiro (2017)).

.

The specific mode to adopt mainly depends on the type of the used camera. Indeed, for the first two modes a CMOS is recommended, whereas the third could work also with a CCD.

Another key point is the type of flow that needs to be represented. The first option fits time-resolved flows, while the second option is a trade-off that permits to represent high-velocity flows. The third mode represents the cheapest option, as it requires fewer camera specifications with respect to the others, but could be used when a high acquisition rate is not necessary.

When the acquisition is over, in the general case coupled images are created. For example, in Figure 2.3 Frame A and Frame B of a Stokes flow around a cylinder is presented.

This images are framed with a time gap Δt, therefore if the displacements are known the velocity could be easily obtained:

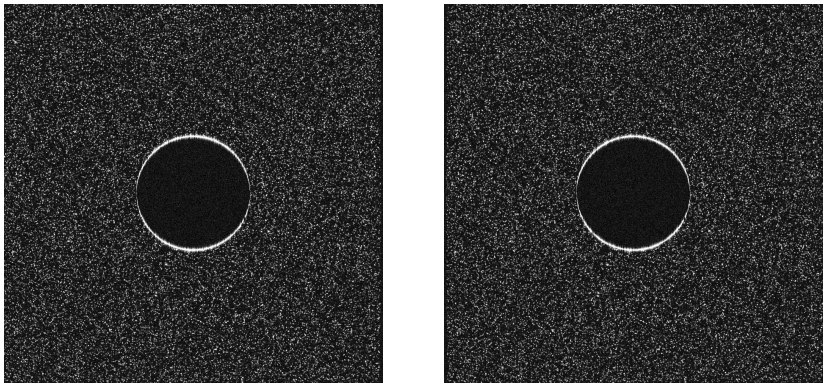
$$\mathbf{U} = \frac{\Delta \mathbf{s}(x, y)}{\Delta t} \tag{2.2}$$

The displacement is computed using a statistical approach. The cross-correlation is defined in equation (2.3).

$$R_{\mathrm{II}}(x, y) = \sum_{i=-K}^{K} \sum_{j=-L}^{L} I(i, j) I'(i + x, j + y) \tag{2.3}$$

where I and I' are the intensities associated with images A and B, respectively. An important feature of the PIV has to be pointed out. The cross-correlation is not computed for every particle.

Indeed, the domain of the first frame is reduced in multiple interrogation windows and it is assumed that all the particles inside that interrogation windows move with the average velocity of all the particles inside that window.



(a) Image A (b) Image B

Figure 2.3: Example of a couple of PIV images.

Intuitively this mechanism is equivalent to take a general window in the first frame, move it all around the second frame until the best fitting windows in the second frame is found. Then, the displacement is equal to the displacement of the windows between the two frames. A visualization of this concept is proposed in figure 2.4.

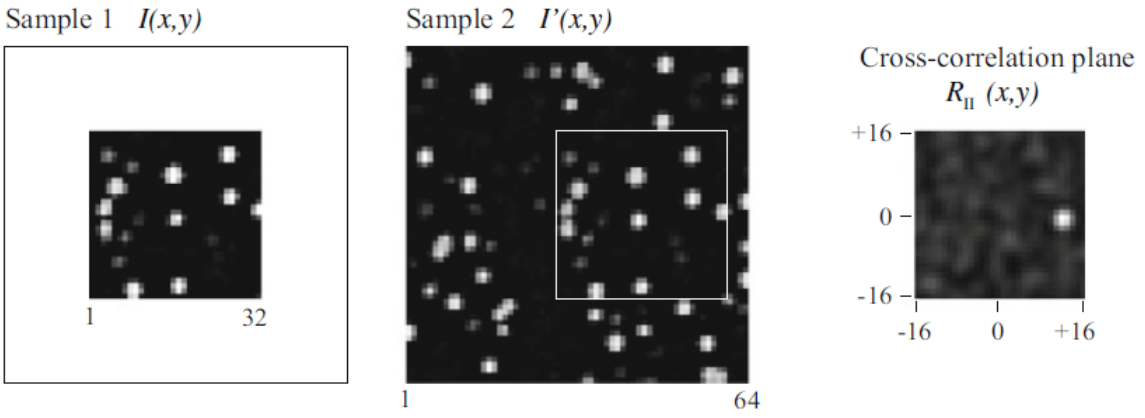


Figure 2.4: Left to right: a small sample I of 32x32 pixel is compared to a larger sample I' of 64x64 pixel; the last figure reports the cross-correlation plane where is evident that the displacement is of 12 pixels. (Raffel et al. (2018)).

Instead of solving (2.3) directly, the correlation theorem is recalled in equation (2.4).

$$\mathcal{F}(I * I') = \mathcal{F}(I)\,\mathcal{F}(I') \tag{2.4}$$

where $\mathcal{F}(\cdot)$ is the Fourier transform. But it can be shown that:

$$\mathcal{F}(R_{\mathrm{II}}) = \overline{\mathcal{F}(I)}\mathcal{F}(I') \tag{2.5}$$

This formulation is usually used because of its lower computational cost: taking a domain with same spatial dimension $N \times N$, the computational cost of the standard cross-correlation is $O\left(N^4\right)$ while with (2.5) is $O\left(N^2 log_2 N\right)$.

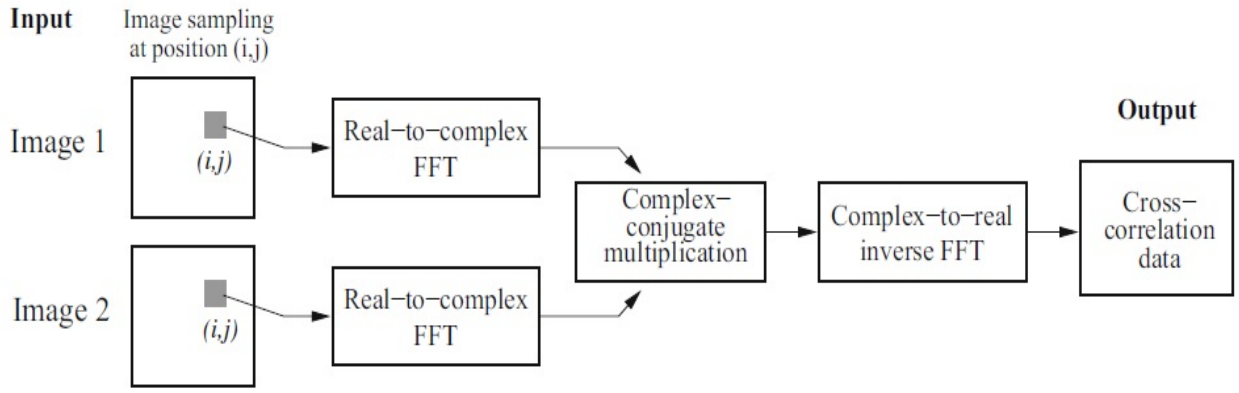


Figure 2.5: Flow chart implementation convolution theorem (Raffel et al. (2018)).

2.2 Particle Tracking Velocimetry

If in PIV the velocity was computed looking for the mean displacement of the particles in a windows, PTV characterize itself for tracking directly one particle at the time.

The setup does not change significantly between the two techniques. Indeed PIV and PTV images could be used interchangeably. The only feature in the parameters that may change, is the particle density which usually is smaller in PTV to better recognize the single particle.

The PTV algorithm consist of different steps. Firstly, the particle is labelled in every image and time and some feature, such as size and shape, is assigned to it.

Finally, the particles are compared to find the matching pairs. The most basic criteria is searching for the nearest matching particle in the region of interest. It is ineluctable that the velocity would be evaluated over scattered points.

One of the points of interest in this techniques is that when time-resolved data are available. It is possible to directly compute Lagrangian quantities which are of great interest, especially for the pressure integration.

Besides these considerations the two methods are similar and use one instead of the other is simply driven by the situation. For example, PTV would be more suitable for a time-resolved boundary layer as PTV is less influenced by the reflection of the wall defined as 'flairs', see Discetti and Ianiro (2017).

2.3 Pressure computation

Pressure computation from image velocimetry has been of wide interest during the last twenty years. There are different strategies to attack the problem.

The first concern is to outline the equations involved in the pressure computation. The roots of any further development are the momentum equations of a incompressible flow:

$$\nabla p = -\rho \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} + \mu \nabla^2 \mathbf{u} \tag{2.6}$$

where ρ and μ are density and viscosity, respectively, and the gravity force term is neglected. The term

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} \tag{2.7}$$

is the material derivative, that is the key difference between the Eulerian and Lagrangian approach. This point

will be discussed further on in this section. First, examine a 2D case:

$$\frac{\partial p}{\partial x} = -\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{2.8}$$

$$\frac{\partial p}{\partial y} = -\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{2.9}$$

The Poisson equation could be derived by differentiating and summing these two terms:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho\left(\left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2\right) \tag{2.10}$$

The momentum equation could be presented in a 3D formulation :

$$\begin{aligned}\frac{\partial p}{\partial x} = &-\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\\ &+ \left(-\rho w\frac{\partial u}{\partial z} + \mu\frac{\partial^2 u}{\partial z^2}\right)\end{aligned} \tag{2.11}$$

$$\begin{aligned}\frac{\partial p}{\partial y} = &-\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\\ &+ \left(-\rho w\frac{\partial v}{\partial z} + \mu\frac{\partial^2 v}{\partial z^2}\right)\end{aligned} \tag{2.12}$$

And so does the Poisson equation as proposed in De Kat et al. (2008):

$$\begin{aligned}\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = &-\rho\left\{\left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2\right\}\\ &- \rho\left\{\frac{\partial \operatorname{div}_{xy}}{\partial t} + u\frac{\partial \operatorname{div}_{xy}}{\partial x} + v\frac{\partial \operatorname{div}_{xy}}{\partial y}\right\}\\ &+ \mu\left\{\frac{\partial^2 \operatorname{div}_{xy}}{\partial x^2} + \frac{\partial^2 \operatorname{div}_{xy}}{\partial y^2}\right\}\\ &- \rho\left\{\frac{\partial w}{\partial x}\frac{\partial u}{\partial z} + \frac{\partial w}{\partial y}\frac{\partial v}{\partial z} + w\frac{\partial \operatorname{div}_{xy}}{\partial z}\right\}\end{aligned} \tag{2.13}$$

where div_{xy} is the in-plane divergence.

The target of the following paragraph is to determine the environment in which it is possible or preferable to use a planar approach, instead of a three-dimensional one.

Obviously the 3D case is more difficult as it requires the out-of-plane gradients which are addressed with a 3D PIV such as tomographic PIV, see Elsinga et al. (2006); Scarano (2012), or a dual-plane approach, see Kähler and Kompenhans (2000).

In general, if a strong three-dimensional motion is not present, it is perfectly justified to use 2D techniques. This has been widely reported in experiments where the two approaches have been confronted: De Kat and Van Oudheusden (2012); Violato et al. (2011) and Koschatzky et al. (2012), indipendently from the dimensionality of the problem. Two strategies could be adopted, the first is to solve the Poisson equation Gurka et al. (1999); Ghaemi et al. (2012) and Pan et al. (2016), the second is the direct momentum integration, see Liu and Katz (2006). Reconnecting to the first strategy, one might believe that solving the Poisson equation corresponds to computing pressure without the need for any knowledge on time-resolved velocity field. On the contrary, in order to correctly compute pressure, the momentum equations have to be used as Neumann boundary conditions (see Gresho and Sani (1987)) that comprehend the time-dependent part of the acceleration term. Usually, a boundary edge with Dirichlet conditions is added to ensure the uniqueness of the solution. To determine these conditions, simplifications such as the one proposed by De Kat and Van Oudheusden (2012) are broadly accepted.

The second strategy is to directly integrate the momentum equation. The most important work in this direction is Liu and Katz (2006). Figure 2.6 shows the methodology. Firstly, pressure is evaluated on the image

boundary and then the pressure is computed through the core of the domain following some omnidirectional path integration along the lines obtained from the grid.

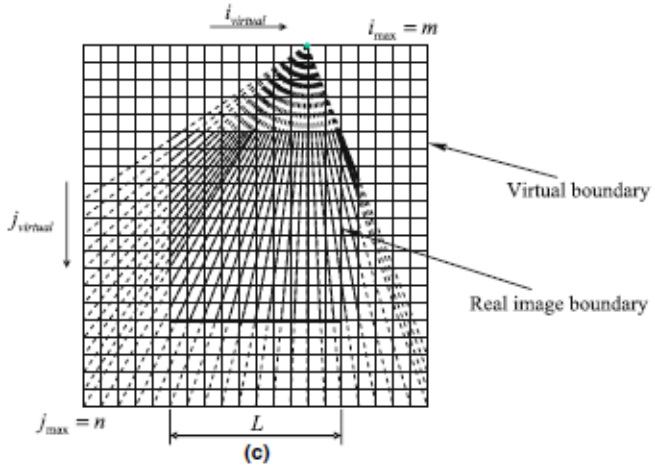


Figure 2.6: Example of omnidirectional integration (Liu and Katz (2006)).

Nonetheless, these methods can still be merged with well-known CFD techniques such as RANS (see Gurka et al. (1999) and van Oudheusden et al. (2007), which computes the averaged pressure for a turbulent and mainly-steady flow). Typically, to compute instantaneous pressure for a generic turbulent flow, time-resolved data and derivatives are required. However, De Kat and Ganapathisubramani (2012) proposed a simplifying assumption that allows the computation of pressure without the need for time derivatives, at least if specific conditions are met. This approach was successfully applied in Van der Kindere et al. (2019); Laskari et al. (2016).

There are still different philosophies on how to compute the pressure besides the equation utilized. This issue has to deal with how the material derivative is addressed, namely if a Lagrangian or an Eulerian approach is adopted. The former requires to follow the single-particle and hence it is a prerogative of PTV like method, here the material derivative of the velocity computed with a Lagrangian approach is shown:

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \frac{\mathrm{d}\mathbf{u}_p(t)}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{u}\left(\mathbf{x}_p(t), t\right)}{\mathrm{d}t} \tag{2.14}$$

where $\mathbf{x}_p(t)$ and $\mathbf{u}_p(t)$ are the position and the velocity of the particle in a specific instant.

In an Eulerian approach, the acceleration is built with the partial derivatives:

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}. \tag{2.15}$$

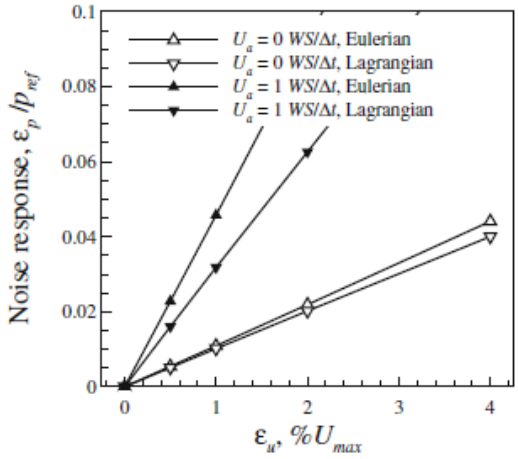
A very popular trade-off is the pseudo-LAgrangian (pLA) approach which uses the velocity field to iteratively achieve the material derivative (see Liu and Katz (2006)):

$$\mathbf{x}_p^k(t, \tau) = \mathbf{x} + \mathbf{u}(\mathbf{x}, t)\tau + \frac{1}{2}\frac{\mathrm{D}\mathbf{u}^k}{\mathrm{D}t}(\mathrm{x}, t)\tau^2 \tag{2.16}$$

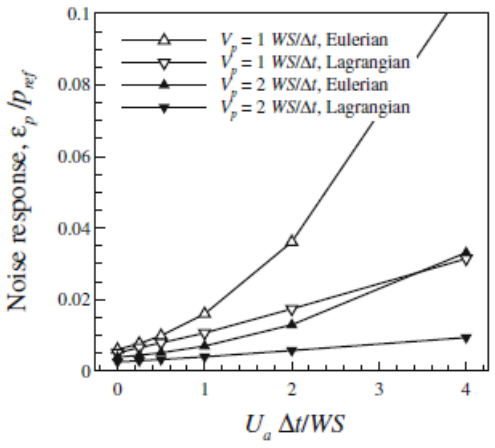
$$\frac{\mathrm{D}\mathbf{u}^{k+1}}{\mathrm{D}t}(\mathbf{x}, t) = \frac{\mathbf{u}\left(\mathbf{x}_p^k(t, \Delta t), t + \Delta t\right) - \mathbf{u}\left(\mathbf{x}_p^k(t, -\Delta t), t - \Delta t\right)}{2\Delta t}. \tag{2.17}$$

For the Lagrangian and Eulerian method an approximation of the noise transmitted from the original noise on the velocities to the pressure is provided in De Kat and Van Oudheusden (2012), here is possible to write the two obtained error for the Lagrangian approach ($\varepsilon_{p_{\mathrm{Lag}}}$) and for the Eulerian approach ($\varepsilon_{p_{\mathrm{Eul}}}$):

$$\varepsilon_{p_{\mathrm{Eul}}} \propto \varepsilon_u \sqrt{\frac{h^2}{2\Delta t^2} + |\nabla u|^2 h^2 + \frac{|\mathbf{u}|^2}{2}} \quad \text{Eulerian} \tag{2.18}$$



(a) Final noise versus the initial noise on the velocity



(b) Final noise versus the advective velocity U_a and the sample time

Figure 2.7: Behaviour of Lagrangian and Eulerian methods (De Kat and Van Oudheusden (2012)).

where h is the grid space and ε_u is the noise on the velocity.

$$\varepsilon_{p_{\text{Lag}}} \propto \varepsilon_u \sqrt{\frac{h^2}{2\Delta t^2} + \frac{|\nabla u|^2 h^2}{2}} \quad \text{Lagrangian} \tag{2.19}$$

As the authors of this article point out:*...the Eulerian approach is expected to be more sensitive to noise and advective motion, whereas the Lagrangian should have difficulties capturing rotational flow, because this complicates the flow path reconstruction*,(De Kat and Van Oudheusden (2012)). This statement was further supported by a numerical experiment (Figure 2.7) on a Gaussian vortex computed with both approaches.

Chapter 3

From function approximation to PDE integration via RBF

This chapter will introduce the basics of Radial Basis Functions (RBF). The first section presents the most conventional methods in which RBFs accomplish interpolation and solve PDEs. The second part focuses on the issue of function approximation using RBF.

The following section describes constrained function approximation, which is the fundamental concept of the whole method. Finally, in the last part, the approximation method will be applied and a PDE solver based on RBFs is proposed.

3.1 Radial Basis Functions fundamentals

The most common RBF are shown in table 3.1, where c is the shape parameter, $\mathbf{x_j}$ is the collocation point.

Name	Radial Function $r = \|\mathbf{x} - \mathbf{x_j}\|_2$
Linear	$\varphi(r) = cr$
Cubic	$\varphi(r) = (r + c)^3$
Thin plate spline	$\varphi(r) = r^2 \log\left(cr^2\right)$
Gaussian	$\varphi(r) = \exp\left(-cr^2\right)$
Multiquadratic	$\varphi(r) = \left(r^2 + c^2\right)^{1/2}$
Inverse Multiquadratic	$\varphi(r) = \left(r^2 + c^2\right)^{-1/2}$

Table 3.1: Most common RBF (Kim et al. (2009)).

We assume that s generic data array $\mathbf{h} \in \mathbb{R}^M$ is known, with h_i for $i = 1, \cdots, M$ representing a generic value; an array $\mathbf{x}$ is used to denote the grid points, i.e the points where the data are evaluated.

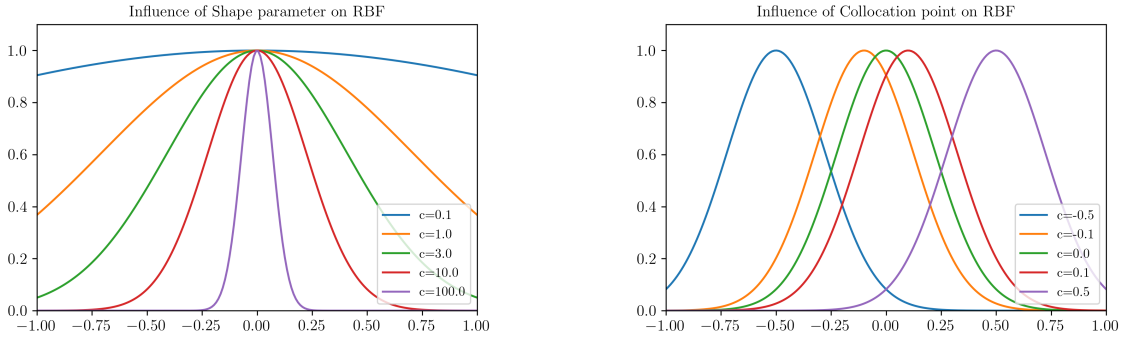
If an interpolation of $\mathbf{h}$ is saught, a linear combination of RBF might be used:

$$\mathbf{h} = \sum_{j=1}^{N} w_j \cdot \varphi\left(\|\mathbf{x} - \mathbf{x_j}\|^2\right) = \sum_{j=1}^{N} w_j \cdot \varphi_j, \tag{3.1}$$

where N is the number of RBF used. The mathematical formulation usually includes also polynomial terms, but they are neglected for the sake of clarity.

The previous equation can be rewritten as

$$\mathbf{h} = \mathbf{\Phi} \cdot \mathbf{w}, \tag{3.2}$$



(a) Effect of the shape parameter on the RBF.

(b) Effect of the collocation point on RBF.

Figure 3.1: Parameter influence on RBF (Gaussian).

where $\mathbf{w} \in \mathbb{R}^N$ is the array containing all the weights in (3.1) and $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ is defined as

$$\mathbf{\Phi} = \begin{pmatrix} \varphi_1(\mathbf{x}_1) & \cdots & \varphi_N(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{x}_M) & \cdots & \varphi_N(\mathbf{x}_M) \end{pmatrix}. \tag{3.3}$$

If $M = N$ and $\mathbf{\Phi}$ is invertible by solving the linear system (3.2) the weights are obtained.

The firsts attempts to solve PDEs with RBF were made in the early 1990s. The first method was proposed by Kansa (1990) and is based on a simple concept. Any linear differentiation operator applied to the RBF interpolation function leads to a linear relationship between the weights and the known differentiated value. The matrices defining this relationship are named differentiation matrix and are denoted here as

$$\partial_{\mathbf{x}}\mathbf{h} = \mathbf{D}_{\mathbf{X}} \cdot \mathbf{w} \qquad \partial_{\mathbf{y}}\mathbf{h} = \mathbf{D}_{\mathbf{Y}} \cdot \mathbf{w} \qquad \mathbf{\Delta h} = \mathbf{L} \cdot \mathbf{w}$$

where:

$$\mathbf{D}_{\mathbf{X}} = \begin{pmatrix} \partial_x\varphi_1(\mathbf{x}_1) & \cdots & \partial_x\varphi_N(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \partial_x\varphi_1(\mathbf{x}_M) & \cdots & \partial_x\varphi_N(\mathbf{x}_M) \end{pmatrix} \in \mathbb{R}^{M \times N} \qquad \mathbf{D}_{\mathbf{Y}} = \begin{pmatrix} \partial_y\varphi_1(\mathbf{x}_1) & \cdots & \partial_y\varphi_N(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \partial_y\varphi_1(\mathbf{x}_M) & \cdots & \partial_y\varphi_N(\mathbf{x}_M) \end{pmatrix} \in \mathbb{R}^{M \times N}$$

$$\mathbf{L} = \begin{pmatrix} \Delta\varphi_1(\mathbf{x}_1) & \cdots & \Delta\varphi_N(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \Delta\varphi_1(\mathbf{x}_M) & \cdots & \Delta\varphi_N(\mathbf{x}_M) \end{pmatrix} \in \mathbb{R}^{M \times N} \tag{3.4}$$

In this thesis the main purpose is to solve the Poisson equation, hence only Poisson solver is presented. However, it is important to point out that also different type of PDE could be solved as Burger equation (see for example Hosseini and Hashemi (2011)) in which Burger equation is considered). Focusing on the Poisson equation

$$\Delta h(\mathbf{x}) = f,$$

we assume a function $\tilde{h}$ which interpolates h; then, replacing in the Poisson equation we have:

$$\Delta \sum_{j=1}^{N} w_j \cdot \varphi_j(\mathbf{x}) = f$$

which in compact form can be written

$$\mathbf{L} \cdot \mathbf{w} = \mathbf{f} \tag{3.5}$$

Regarding how the boundary conditions should be imposed, it is sufficient to force the equation to satisfy them on some points of the boundary. This example includes only Dirichlet conditions, but if Neumann conditions were considered, these conditions can be introduced using the relative differentiation matrix. The RBF approximation is forced to satisfy Dirichlet boundary conditions at C points:

$$\mathbf{\Phi_C} \cdot \mathbf{w} = \mathbf{g} \tag{3.6}$$

where $\mathbf{\Phi_C} \in \mathbb{R}^{C \times N}$ and $\mathbf{g} \in \mathbb{R}^C$ and the subscript C differentiates this matrix from $\mathbf{\Phi}$ which is evaluated in M points. Collocation points do not change. The final matrix looks like:

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{\Phi_C} \end{pmatrix} \cdot \mathbf{w} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} \tag{3.7}$$

The main problem of this method is that is not possible to demonstrate that this last matrix is invertible as $\mathbf{\Phi}$. Furthermore, $M + C = N$ needs to hold true otherwise the matrix is not squared.

In Fasshauer (1996) an evolution of this method is proposed, called the symmetric Hermite method. In this variant, the Hermite interpolant is added to the previous method to ensure the non-singularity of the matrix. So equation (3.1) becomes:

$$\mathbf{h} = \sum_{j=1}^{N} w_j \cdot \varphi_j + \sum_{j=N+1}^{N+B} w_j \cdot \Delta^{\mathbf{x_j}} \varphi_j \tag{3.8}$$

where the superscript $\mathbf{x_j}$ implies that the Laplacian is computed with respect to the collocation point. N is the number of RBFs in the interior domain, while B is the number of collocation points on the boundary. Applying this idea to the problem with Dirichlet boundary conditions (3.7), rewrites:

$$\begin{pmatrix} \mathbf{L} & \mathbf{LL^{x_j}} \\ \mathbf{\Phi} & \mathbf{L^{x_j}} \end{pmatrix} \cdot \mathbf{w} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} \tag{3.9}$$

where the new matrices are

$$\mathbf{L^{x_j}} = \begin{pmatrix} \Delta^{\mathbf{x_j}} \varphi_{N+1}(\mathbf{x}_1) & \cdots & \Delta^{\mathbf{x_j}} \varphi_{N+B}(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \Delta^{\mathbf{x_j}} \varphi_{N+1}(\mathbf{x}_M) & \cdots & \Delta^{\mathbf{x_j}} \varphi_{N+B}(\mathbf{x}_M) \end{pmatrix} \in \mathbb{R}^{M \times B} \quad \mathbf{LL^{x_j}} = \begin{pmatrix} \Delta\Delta^{\mathbf{x_j}} \varphi_{N+1}(\mathbf{x}_1) & \cdots & \Delta\Delta^{\mathbf{x_j}} \varphi_{N+B}(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \Delta\Delta^{\mathbf{x_j}} \varphi_{N+1}(\mathbf{x}_M) & \cdots & \Delta\Delta^{\mathbf{x_j}} \varphi_{N+B}(\mathbf{x}_M) \end{pmatrix} \in \mathbb{R}^{M \times B} \tag{3.10}$$

the whole matrix would belong to $\mathbb{R}^{N+B \times N+B}$.

The main problem with these methods is how to find an optimal shape parameter and collocations. Furthermore, Kansa is not symmetric, neither is it Hermite if a changing of shape parameter is chosen. Additionally the matrix could be ill-conditioned and full, leading to several computational problems.

Several works have investigated these methods in several areas. One of these areas is to find a demonstration of the uniqueness of the solution, in this sense an important work was delivered by Hon and Schaback (2001). Unfortunately, presently the demonstration of uniqueness is still lacking. Other areas of research are to find stable algorithms to tackle this problem, which is well treated by Fasshauer (1999), or shape parameter optimization described by Rippa (1999) and Wang and Liu (2002).

3.2 Function approximation using RBF

The concepts on interpolation illustrated in the previous section do not represent the only possible way to represent data with the RBF. In the present section the function approximation via RBF is presented. We introduce a functional J defined as the squared euclidean norm of the residual of (3.2), namely:

$$J(\mathbf{w}) = \|\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{h}\|_2^2 .$$

In this case it is a consolidated practice to add a Tikhonov regularization:

$$J(\mathbf{w}) = \|\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{h}\|_2^2 + \alpha \|\mathbf{w}\|_2^2 \tag{3.11}$$

where α is an arbitrary strictly positive parameter. The regularization is added because it improves the behaviour of the problem as will be shown, and it smooths the results. Problem (3.2) is replaced by

$$\min_{\mathbf{w} \in \mathbb{R}^N} J(\mathbf{w}),$$

since J is a convex smooth functional, its minimum is attained at the unique stationary point:

$$\nabla J(\mathbf{w}) = 0 = 2\left(\mathbf{\Phi^T} \cdot \mathbf{\Phi} \cdot \mathbf{w} - \mathbf{\Phi^T} \cdot \mathbf{h} + \alpha \mathbf{w}\right). \tag{3.12}$$

It follows that $\mathbf{w}$ which minimizes the error should satisfy

$$\mathbb{H} \cdot \mathbf{w} = 2 \cdot \mathbf{\Phi} \cdot \mathbf{h} \tag{3.13}$$

with:

$$\mathbb{H} = 2\mathbf{\Phi^T} \cdot \mathbf{\Phi} + 2\alpha \mathbf{I} \tag{3.14}$$

in this matrix, $\mathbf{I} \in^{N \times N}$ is the identity matrix. The matrix $\mathbb{H} \in \mathbb{R}^{N \times N}$ has the following features:

- $\mathbb{H}$ is always positive definite if $\alpha > 0$;
- If $\alpha = 0$ but $dim(Ker\,(\mathbf{\Phi})) = 0$, then $\mathbb{H}$ is always positive definite.

If $dim(Ker\,(\mathbf{\Phi})) \neq 0$ it means that the information to build a RBF is redundant. This is not expected to happen, but in practice the matrix might be ill-conditioned; then, the regularization also improves the conditioning even if $2\mathbf{\Phi^T} \cdot \mathbf{\Phi}$ is positive definite.

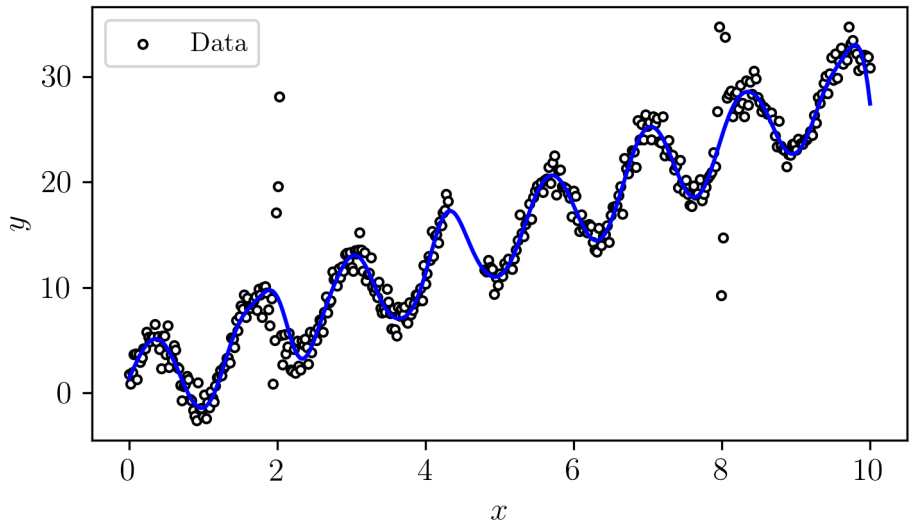


Figure 3.2: Example of function approximation via RBF.

3.3 Constrained RBF function approximation

In the framework of the function approximation is possible to add constraints. This is reasonable in term of additional computational effort because all the constraints for the RBFs are linear. The previous problem rewrites

$$\min_{\mathbf{w} \in \mathbb{R}^N} J(\mathbf{w}) \quad with \quad h(\mathbf{x_k}) = g_k \quad k = 1, \cdots, C$$

where C is the number of constraint and g_k is the constraint on the function in the point $\mathbf{x_k}$, this equation can be rewritten again into

$$\min_{\mathbf{w}\in\mathbb{R}^N} J(\mathbf{w}) \quad with \quad \mathbf{\Phi_C}\cdot\mathbf{w}=\mathbf{g}$$

where $\mathbf{g}$ is an array containing all the constraints g_k and $\mathbf{\Phi_C}\in\mathbb{R}^{N\times C}$ is the matrix $\mathbf{\Phi}$ evaluated in the constrained points. This constrained optimum can be calculated by recalling the associated Lagrangian

$$\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) = \|\mathbf{\Phi}\cdot\mathbf{w}-\mathbf{h}\|_2^2 + \alpha\|\mathbf{w}\|_2^2 + \boldsymbol{\lambda}^{\boldsymbol{T}}\cdot(\mathbf{\Phi_C}\cdot\mathbf{w}-\mathbf{g})\,. \tag{3.15}$$

where $\boldsymbol{\lambda}\in\mathbb{R}^C$ is an array containing all the Lagrange multiplier. In this context, the constraints are on the function values but might be of general type (derivative,divergence,*etc*). The theory of constrained optimization states that a necessary condition for $\mathbf{w}$ to be a constrained optimum is that $\mathbf{w}$ and $\boldsymbol{\lambda}$ satisfy $\nabla\mathcal{L}=\mathbf{0}$, therefore $\mathcal{L}$ is derived with respect to $\mathbf{w}$ and $\boldsymbol{\lambda}$ separately

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) &= \mathbb{H}\cdot\mathbf{w}-\mathbf{2}\cdot\mathbf{\Phi}^{\mathbf{T}}\cdot\mathbf{h}+\mathbf{\Phi}_{\mathbf{C}}^{\mathbf{T}}\cdot\boldsymbol{\lambda}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) &= \mathbf{\Phi_C}\cdot\mathbf{w}-\mathbf{g}\end{aligned}\,, \tag{3.16}$$

imposing $\nabla\mathcal{L}=\mathbf{0}$:

$$\begin{pmatrix}\mathbb{H} & \mathbf{\Phi}_{\mathbf{C}}^{\mathbf{T}}\\ \mathbf{\Phi_C} & \mathbf{0}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{w}\\ \boldsymbol{\lambda}\end{pmatrix}=\begin{pmatrix}\mathbf{2}\cdot\mathbf{\Phi}^{\mathbf{T}}\cdot\mathbf{h}\\ \mathbf{g}\end{pmatrix}. \tag{3.17}$$

Let $\mathbb{B}\in\mathbb{R}^{N\times C}$ denote the constraint matrix that could include the generic constraints which could be mixed, on the derivatives, *etc*. Hence the general linear system to solve is

$$\begin{pmatrix}\mathbb{H} & \mathbb{B}^{\mathbf{T}}\\ \mathbb{B} & \mathbf{0}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{w}\\ \boldsymbol{\lambda}\end{pmatrix}=\begin{pmatrix}\mathbf{2}\cdot\mathbf{\Phi}^{\mathbf{T}}\cdot\mathbf{h}\\ \mathbf{c}\end{pmatrix}. \tag{3.18}$$

For Nocedal and Wright (2006), if $dim(Ker(\mathbb{B}^{\mathbf{T}}))=0$ then only one solution exist.

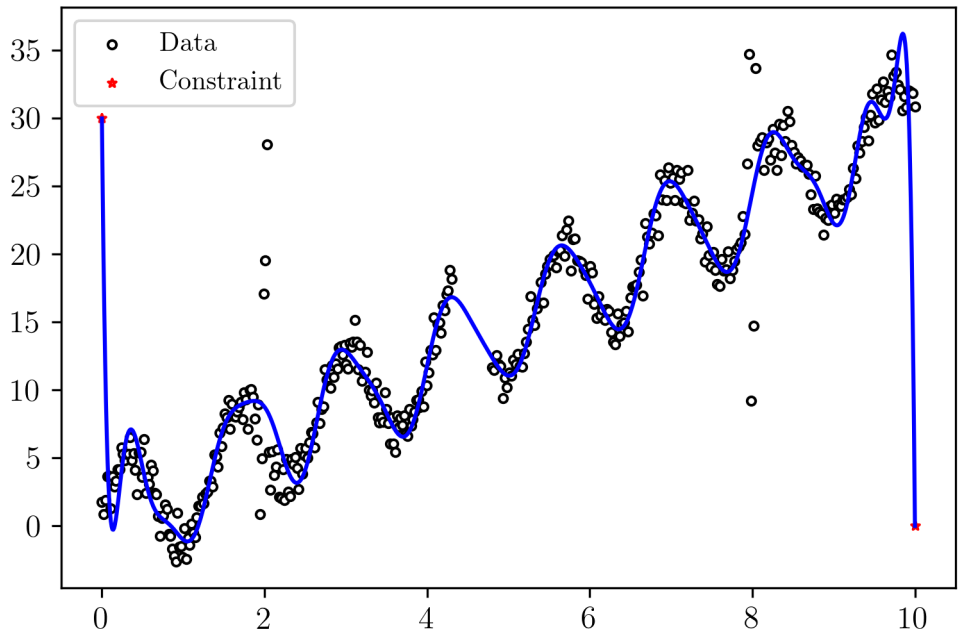


Figure 3.3: Example of constrained function approximation via RBF.

3.4 Solving Poisson equation

Similarly to how the Kansa's method uses the RBF interpolation to solve the PDEs, constrained function approximation could be used for the same purpose.

Let h be a solution to the Poisson problem:

$$\Delta h = f \tag{3.19}$$

subjected to some boundary condition. Supposing that there exists an approximation of h $\tilde{h}$, we have:

$$\tilde{h} = \sum_{j=1}^{N} w_j \cdot \varphi_j \approx h \tag{3.20}$$

This can be substituted into the Poisson equation. However, because this is just an approximation there will be some error:

$$\Delta\tilde{h} - f = \sum_{j=1}^{N} w_j \cdot \Delta\varphi_j - f = \varepsilon. \tag{3.21}$$

The previous formula can be rewritten in an L2-norm and evaluated at some grid point $\mathbf{x}$. Then using the matrix (3.2) the error can be written similarly as before:

$$J(\mathbf{w}) = \|\mathbf{L} \cdot \mathbf{w} - \mathbf{f}\|_2^2 . \tag{3.22}$$

To add the boundary conditions, constraints can be used. This leads to a matrix similar to the previous one:

$$\begin{pmatrix} \mathbb{L} & \mathbb{B}^{\mathbf{T}} \\ \mathbb{B} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{2} \cdot \boldsymbol{\Phi}^{\mathbf{T}} \cdot \mathbf{f} \\ \mathbf{b} \end{pmatrix} \tag{3.23}$$

where $\mathbf{b}$ is th vector containing the boundary data and $\mathbb{L}$ is defined as:

$$\mathbb{L} = 2\mathbf{L}^{\mathbf{T}} \cdot \mathbf{L} + 2\alpha\mathbf{I} \tag{3.24}$$

The uniqueness of a solution to (3.23) can be demonstrated similarly to case of the function approximation.

Chapter 4

Implementation of the meshless pressure evaluation

This chapter focuses on how to use the Radial Basis Functions to evaluate pressure from image velocimetry and how this approach has been implemented.

The first section deals with the application of physical constraints or penalties to the velocity approximation. Then, an automatic way to define collocation point and shape parameter is presented. The following section investigates the strategies applied to solve the linear system. Finally, the specific Python module created for this work is presented.

4.1 Including physical constraints and penalties

The target of the inclusion of constraints in the velocities is to add information that a normal image velocimetry does not carry, such as the no slip-condition or the known inlet velocity. Applying the divergence-free constraint on the boundary is less trivial. This last constraint could be seen as useless; however, it is very important, as it restores information on the derivatives without imposing any mathematical non-physical conditions.

Equation (3.15) can be properly changed to approximate a 2D velocity field where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^M$ are the velocities values in all the M points:

$$\begin{aligned}\mathcal{L}(\mathbf{w_u}, \mathbf{w_v}, \boldsymbol{\lambda_u}, \boldsymbol{\lambda_v}, \boldsymbol{\lambda_{DIV}}) = \|\boldsymbol{\Phi_u} \cdot \mathbf{w_u} - \mathbf{u}\|_2^2 + \|\boldsymbol{\Phi_v} \cdot \mathbf{w_v} - \mathbf{v}\|_2^2 + \alpha \|\mathbf{w_u}\|_2^2 + \alpha \|\mathbf{w_v}\|_2^2 + \boldsymbol{\lambda_u^T} \cdot (\mathbb{B_u} \cdot \mathbf{w_u} - \mathbf{g_u}) \\ + \boldsymbol{\lambda_v^T} \cdot (\mathbb{B_v} \cdot \mathbf{w_v} - \mathbf{g_v}) + \boldsymbol{\lambda_{DIV}^T} \cdot (\mathbf{D_{XC,u}} \cdot \mathbf{w_u} + \mathbf{D_{YC,v}} \cdot \mathbf{w_v}) \,.\end{aligned} \tag{4.1}$$

$\mathbf{D_{XC,u}}$ and $\mathbf{D_{YC,v}} \in \mathbb{R}^{CDIV \times N}$ are the differentiation matrices evaluated at $CDIV$ constraint points for the divergence-free. From here all subscripts in the matrices referring to $\mathbf{u}$ or $\mathbf{v}$ will be neglected, at the exception of $\boldsymbol{\lambda_u}$ and $\boldsymbol{\lambda_v}$, following the assumption that both velocities are calculated with the same RBFs and that $\mathbb{B}_u = \mathbb{B}_v \in \mathbb{R}^{C \times N}$ for construction, since any constraint applied to velocity in one direction would have its counterpart on the other direction. Finally, $\boldsymbol{\lambda_u}$, $\boldsymbol{\lambda_v} \in \mathbb{R}^C$, $\boldsymbol{\lambda_{DIV}} \in \mathbb{R}^{CDIV}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ as before.

Some decisions have been made to simplify the implementation of the algorithm: let $\mathbf{x_c} \in \mathbb{R}^C$ be the points where constraints other than the divergence-free are applied, then $\mathbf{x_c}$ is a subvector of $\mathbf{x_{cDIV}} \in \mathbb{R}^{CDIV}$, where $\mathbf{x_{cDIV}}$ are the points where the divergence-free constraint is applied. To explain this choice, let us consider a square domain with a wall in the lower edge. The only constraint different from divergence-free that can be applied is on this edge. Let have 1000 RBFs and 40 points per edge, which are supposed to be constraint points. Then $\mathbb{B} \in \mathbb{R}^{40 \times 1000}$ would be evaluated in the 40 points in correspondence of the lower edge to apply the no-slip condition, while $\mathbf{D_{XC}}, \mathbf{D_{YC}} \in \mathbb{R}^{160 \times 1000}$ are evaluated at all the boundaries of the domain, because it is always possible to apply the divergence-free condition.

Going back to (4.1), also in this case we look for points satisfying the optimality conditions by setting to zero

the gradient of $\mathcal{L}$. Again the derivatives are calculated with respect to the weights and the Lagrange multipliers:

$$\begin{aligned}
\nabla_{\mathbf{w_u}}\mathcal{L}(\mathbf{w_u},\mathbf{w_v},\boldsymbol{\lambda_u},\boldsymbol{\lambda_v},\boldsymbol{\lambda_{DIV}}) &= \mathbb{H}\cdot\mathbf{w_u} - \mathbf{2}\cdot\mathbf{\Phi^T}\cdot\mathbf{u} + \mathbb{B}^{\mathbf{T}}\cdot\boldsymbol{\lambda_u} + \mathbf{D}^{\mathbf{T}}_{\mathbf{X,C}}\cdot\boldsymbol{\lambda_{DIV}} \\
\nabla_{\mathbf{w_v}}\mathcal{L}(\mathbf{w_u},\mathbf{w_v},\boldsymbol{\lambda_u},\boldsymbol{\lambda_v},\boldsymbol{\lambda_{DIV}}) &= \mathbb{H}\cdot\mathbf{w_v} - \mathbf{2}\cdot\mathbf{\Phi^T}\cdot\mathbf{v} + \mathbb{B}^{\mathbf{T}}\cdot\boldsymbol{\lambda_v} + \mathbf{D}^{\mathbf{T}}_{\mathbf{Y,C}}\cdot\boldsymbol{\lambda_{DIV}} \\
\nabla_{\boldsymbol{\lambda_u}}\mathcal{L}(\mathbf{w_u},\mathbf{w_v},\boldsymbol{\lambda_u},\boldsymbol{\lambda_v},\boldsymbol{\lambda_{DIV}}) &= \mathbb{B}\cdot\mathbf{w_u} - \mathbf{g_u} \\
\nabla_{\boldsymbol{\lambda_v}}\mathcal{L}(\mathbf{w_u},\mathbf{w_v},\boldsymbol{\lambda_u},\boldsymbol{\lambda_v},\boldsymbol{\lambda_{DIV}}) &= \mathbb{B}\cdot\mathbf{w_v} - \mathbf{g_v} \\
\nabla_{\boldsymbol{\lambda_{DIV}}}\mathcal{L}(\mathbf{w_u},\mathbf{w_v},\boldsymbol{\lambda_u},\boldsymbol{\lambda_v},\boldsymbol{\lambda_{DIV}}) &= \mathbf{D_{X,C}}\cdot\mathbf{w_u} + \mathbf{D_{Y,C}}\cdot\mathbf{w_v}
\end{aligned} \tag{4.2}$$

Imposing the equality to zero we obtain:

$$\begin{pmatrix}
\mathbb{H} & \mathbf{0} & \mathbb{B}^{\mathbf{T}} & \mathbf{0} & \mathbf{D}^{\mathbf{T}}_{\mathbf{X,C}} \\
\mathbf{0} & \mathbb{H} & \mathbf{0} & \mathbb{B}^{\mathbf{T}} & \mathbf{D}^{\mathbf{T}}_{\mathbf{Y,C}} \\
\mathbb{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbb{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{D_{X,C}} & \mathbf{D_{Y,C}} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{pmatrix}\cdot\begin{pmatrix}
\mathbf{w_u} \\ \mathbf{w_v} \\ \boldsymbol{\lambda_u} \\ \boldsymbol{\lambda_v} \\ \boldsymbol{\lambda_{DIV}}
\end{pmatrix} = \begin{pmatrix}
\mathbf{2\Phi^T}\cdot\mathbf{u} \\ \mathbf{2\Phi^T}\cdot\mathbf{v} \\ \mathbf{g_u} \\ \mathbf{g_v} \\ \mathbf{0}
\end{pmatrix}. \tag{4.3}$$

The final linear system to be solved has a coefficient matrix $\in \mathbb{R}^{(2N+2C+CDIV)\times(2N+2C+CDIV)}$. The solution of this system is unique if both $\mathbf{\Phi}$ and the matrix

$$\begin{pmatrix}
\mathbb{B}^{\mathbf{T}} & \mathbf{0} & \mathbf{D}^{\mathbf{T}}_{\mathbf{X,C}} \\
\mathbf{0} & \mathbb{B}^{\mathbf{T}} & \mathbf{D}^{\mathbf{T}}_{\mathbf{Y,C}}
\end{pmatrix}$$

have kernel dimension equal to zero. If the regularization is used with α large enough, then only the second condition is needed. Let us introduce a divergence-free penalty to be evaluated at the same M point of $\mathbf{\Phi}$. We introduce

$$J_{DIV}(\mathbf{w_u},\mathbf{w_v}) = (\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{D_Y}\cdot\mathbf{w_v})^T\cdot(\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{D_Y}\cdot\mathbf{w_v}) = \left(\mathbf{w_u^T}\cdot\mathbf{D_X^T} + \mathbf{w_v^T}\cdot\mathbf{D_Y^T}\right)\cdot(\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{D_X}\cdot\mathbf{w_v}), \tag{4.4}$$

where $\mathbf{D_X}, \mathbf{D_Y} \in \mathbb{R}^{M\times N}$. Then, the functional J has to be incremented by J_{DIV}

$$J_{TOT}(\mathbf{w_u},\mathbf{w_v}) = J(\mathbf{w_u},\mathbf{w_v}) + J_{DIV}(\mathbf{w_u},\mathbf{w_v})$$

Expanding (4.4)

$$J_{DIV}(\mathbf{w_u},\mathbf{w_v}) = \mathbf{w_u^T}\cdot\mathbf{D_X^T}\cdot\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{w_v^T}\cdot\mathbf{D_Y^T}\cdot\mathbf{D_Y}\cdot\mathbf{w_v} + \mathbf{2w_u^T}\cdot\mathbf{D_X^T}\cdot\mathbf{D_Y}\cdot\mathbf{w_v} \tag{4.5}$$

$$\nabla_{w_u}J_{DIV}(\mathbf{w_u},\mathbf{w_v}) = \mathbf{2D_X^T}\cdot\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{2D_X^T}\cdot\mathbf{D_Y}\cdot\mathbf{w_v} \tag{4.6}$$

$$\nabla_{w_v}J_{DIV}(\mathbf{w_u},\mathbf{w_v}) = \mathbf{2D_Y^T}\cdot\mathbf{D_X}\cdot\mathbf{w_u} + \mathbf{2D_Y^T}\cdot\mathbf{D_Y}\cdot\mathbf{w_v} \tag{4.7}$$

In a more compact way, the equations rewrite:

$$\begin{pmatrix}
\mathbf{2D_X^T}\cdot\mathbf{D_X} & \mathbf{2D_X^T}\cdot\mathbf{D_Y} \\
\mathbf{2D_Y^T}\cdot\mathbf{D_X} & \mathbf{2D_Y^T}\cdot\mathbf{D_Y}
\end{pmatrix}\cdot\begin{pmatrix}
\mathbf{w_u} \\ \mathbf{w_v}
\end{pmatrix} = \begin{pmatrix}
\nabla_{w_u}J_{DIV}(\mathbf{w_u},\mathbf{w_v}) \\ \nabla_{w_v}J_{DIV}(\mathbf{w_u},\mathbf{w_v})
\end{pmatrix}. \tag{4.8}$$

According to Theorem 2 this matrix is always positive semi-definite. This gradient is added to the previously obtained (4.2), and the matrix rewrites:

$$\begin{pmatrix}
\mathbb{H}+\mathbf{2D_X^T}\cdot\mathbf{D_X} & \mathbf{2D_X^T}\cdot\mathbf{D_Y} & \mathbb{B}^{\mathbf{T}} & \mathbf{0} & \mathbf{D}^{\mathbf{T}}_{\mathbf{X,C}} \\
\mathbf{2D_Y^T}\cdot\mathbf{D_X} & \mathbb{H}+\mathbf{2D_Y^T}\cdot\mathbf{D_Y} & \mathbf{0} & \mathbb{B}^{\mathbf{T}} & \mathbf{D}^{\mathbf{T}}_{\mathbf{Y,C}} \\
\mathbb{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbb{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{D_{X,C}} & \mathbf{D_{Y,C}} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{pmatrix}\cdot\begin{pmatrix}
\mathbf{w_u} \\ \mathbf{w_v} \\ \boldsymbol{\lambda_u} \\ \boldsymbol{\lambda_v} \\ \boldsymbol{\lambda_{DIV}}
\end{pmatrix} = \begin{pmatrix}
\mathbf{2\Phi^T}\cdot\mathbf{u} \\ \mathbf{2\Phi^T}\cdot\mathbf{v} \\ \boldsymbol{c_u} \\ \boldsymbol{v_v} \\ \mathbf{0}
\end{pmatrix}. \tag{4.9}$$

The upper-left corner block, with dimension $\mathbb{R}^{2N\times 2N}$, is positive definite(summation of a positive definite matrix and a positive semi-definite matrix); therefore if the constraint part has $dim(Ker)$ equal to 0 then exist a solution for Nocedal and Wright (2006).

4.2 Defining collocation points and shape parameter

In this section an automatic way to define the collocation points and shape parameters is proposed. In this work, only Gaussians in the form of $\varphi(r) = \exp\left(-c^2 r^2\right)$ were used. Therefore when referring to this algorithm or to the implementation, the terms Gaussians and RBFs could be used interchangeably. The proposed method differs from Kansa's method, since choosing a changing shape parameter does not affect the symmetrical feature of the matrix.

However, there is still the issue of identifying these points and the shape parameters. First, a brief discussion on the criteria to find a good positioning are mentioned:

1. Enough points have to be under the dome of each Gaussian

2. The Gaussians have to be close enough to be correlated between them

3. Higher value of c means more accurate results

The first point is intuitive; in the critical situation where points are very distant from the center of the Gaussian, the Gaussian is not determined. It follows that, and when the matrix $\mathbf{\Phi}$ is constructed, the row corresponding to this Gaussian would be almost zero leading to an ill-conditioned matrix.

The second point is again demonstrated through an example. If the Gaussian shape parameter tends to infinity, the RBF turns into a Dirac delta function. When this happens, the function approximation becomes an interpolation, because the only points that could be represented are the ones that coincide with the collocation point. Furthermore, all the points that lay between one collocation point and another become untrustworthy to extrapolate. In a realistic situation the Gaussians would not be a Dirac function, but they can be very spiky. This kind of Gaussian need to be very near to represent the points which lay between two collocation points.

Regarding the last point, results will show that this function approximator works as a low-pass filter and the frequency which will be cut off is proportional to c. Hence, the number of RBFs used would become irrelevant after reaching a saturation number of RBFs. This behaviour has been widely reported in the literature of Kansa/Hermite method: Cheng et al. (2003); Fasshauer (1996).

All considered , these three criteria are discordant but could be summarized by one sentence: The best value for c is the highest value allowed by the data. As an example, imagine having a very dense dataset and enforcing the criterion that n points have to be under the Gaussian; in this case c will be rather big. On the other hand, if the dataset is coarse, to satisfy the same criterion c will be small. On this basis, the consequential idea is to use the dataset position to decide the placement of the Gaussians. In particular, a clustering method has been used. This method allows the creation of groups of points, each related to a specific centroid; centroids are calculated by maximizing the distance between them and minimizing the distance between the points of one group and their relative centroid. To calculate these centroids the `scikit-learn cluster module` has been used in particular `MiniBatchKmeans`[1]. As input it requires the number of groups to be created. The specific function developed in this works requires the average number of points per cluster instead, as it will be explained in the implementation section.

The idea at this point is to collocate the Gaussian in the centroid itself. To decide the relative c a rule of thumb has been found:

$$c = 0.5/(\sqrt{2}D_{min}) \tag{4.10}$$

where D_{min} is the distance between a centroid and the nearest centroid.

Larger Gaussians are required to approximate the part of the field characterized by smaller frequencies with less RBF. These are obtained by repeating clustering on the centroids and this could be done more than one time.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`

4.3 Solving the linear system

In this section some methods to solve the linear system involved in this problem are proposed, in particular to make a comparison all this method were used to resolve the velocity matrix of this test case 5.2. Based on the time required by the calculations and the obtained accuracy, these methods are ranked.

We start remarking that the matrices involved in this problem are ill-conditioned. As a result, the use of methods such as Penrose-Moore pseudo-inverse or QR factorization might be required. However, this does not appears to be always the case if proper regularization is used.

The method used here could be divided into two branches:

- standard;
- Schur Complement Approaches (SCA), see Nocedal and Wright (2006).

The first is referred to the standard case in which the whole system is solved, whereas SCAs manipulate the original matrix to solve two smaller problems. All the linear systems introduced so far have the structure

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix}. \tag{4.11}$$

It is supposed that no regularization is applied unless differently specified. If $\mathbf{A}$ is invertible, the vector $\mathbf{w}$ can be exploited:

$$\mathbf{w} = \mathbf{A}^{-1} \left(\mathbf{b_1} - \mathbf{B^T} \cdot \boldsymbol{\lambda} \right)$$

substituting it in the second block of equations:

$$\mathbf{B} \cdot \mathbf{A}^{-1} \left(\mathbf{b_1} - \mathbf{B^T} \cdot \boldsymbol{\lambda} \right) = \mathbf{b_2}$$

$$\mathbf{B} \cdot \mathbf{A}^{-1} \cdot \mathbf{B^T} \cdot \boldsymbol{\lambda} = \mathbf{B} \cdot \mathbf{A}^{-1} \cdot \mathbf{b_1} - \mathbf{b_2}$$

and defining:

$$\mathbf{M} = \mathbf{B} \cdot \mathbf{A}^{-1} \cdot \mathbf{B^T}$$

$$\mathbf{N} = \mathbf{B} \cdot \mathbf{A}^{-1}.$$

The final solution could be achieved by solving these linear systems in the following order:

$$\mathbf{M} \cdot \boldsymbol{\lambda} = \mathbf{N} \cdot \mathbf{b_1} - \mathbf{b_2} \tag{4.12}$$

$$\mathbf{A} \cdot \mathbf{w} = \mathbf{b_1} - \mathbf{B^T} \cdot \boldsymbol{\lambda} \tag{4.13}$$

It is worth to underline that $\mathbf{M}$ is always at least semidefinite positive.

4.3.1 Efficient method to identify a suitable regularization

In the previous section, it was mentioned that Tikhonov regularization has no importance in terms of smoothing because of the physical penalty on the divergence-free; nevertheless it is kept to treat the bad conditioning of the matrix. The regularization acts on the eigenvalues, adding α to them. Let $\mathbf{A}_R$ be the matrix $\mathbf{A}$ regularized with some α and $\sigma(\mathbf{A}_R)$ its eigenvalues, then we have:

$$\sigma(\mathbf{A}_R) = \sigma(\mathbf{A}) + \alpha.$$

However if $min\left(\sigma(\mathbf{A})\right) \ll \alpha$ then:

$$min\left(\sigma(\mathbf{A}_R)\right) \approx \alpha$$

Since $\mathbf{A}$ is symmetric postive definite, the conditioning number in spectral norm is given by:

$$k(\mathbf{A}) = \frac{max\left(\sigma(\mathbf{A})\right)}{min\left(\sigma(\mathbf{A})\right)}$$

However, for compatibility reasons with other solvers, in this thesis it is preferred to work with the inverse of this number:

$$rcond = \frac{1}{k(\mathbf{A})} = \frac{min\left(\sigma(\mathbf{A})\right)}{max\left(\sigma(\mathbf{A})\right)}$$

Therefore, if $min\left(\sigma(\mathbf{A})\right) \gg \alpha$, the regularization improves the conditioning of the matrix. However if α is chosen too big it completely change the original system. In this case there is no interest in the smoothing properties of the Tikhonov regularization, as this is ensured by the divergence-free penalty.

Hence α must be chosen to be the smallest possible to ensure a good conditioning. However, to get the correct α for this purpose the eigenvalue should be calculated, which would be pointless as in this case it would be easier to directly solve the system with a SVD.

Fortunately some approximations could be made,as it is always true that:

$$k(\mathbf{A_R}) = \frac{max\left(\sigma(\mathbf{A_R})\right)}{min\left(\sigma(\mathbf{A_R})\right)} \approx \frac{max\left(\sigma(\mathbf{A_R})\right)}{\alpha}$$

Moreover it is also true that if $\alpha \ll max\left(\sigma(\mathbf{A})\right)$ then:

$$max\left(\sigma(\mathbf{A}_R)\right) \approx max\left(\sigma(\mathbf{A})\right)$$

$$k(\mathbf{A}_R) \approx \frac{max\left(\sigma(\mathbf{A})\right)}{\alpha} \leq \frac{\|\mathbf{A}\|_\infty}{\alpha}.$$

Hence taking into account the worst situation possible, when the $k(\mathbf{A}_R)$ is the biggest allowed:

$$k(\mathbf{A}_R) = \frac{\|\mathbf{A}\|_\infty}{\alpha}.$$

The alpha could be estimated as:

$$\alpha = rcond\|\mathbf{A}\|_\infty$$

Where *rcond* must be decided by the user and should be around 10^{-10}.

4.3.2 Solving the linear system with standard Numpy routine

This is the easiest way to go through the problem, this method is the only one proposed for the standard approach. Simply, the `numpy.linalg.solve()` [2] is called.

Unfortunately, as $\mathbf{A}$ is not solved on its own it is not possible to identify α easily as suggested in the previous part. A brief table reporting the pros and cons is shown in table (4.1).

Pros	Cons
The most accurate	α must be identified by trial and error
	Expensive respect to the competitors

Table 4.1: Pros and cons of the solve method.

4.3.3 Penrose-Moore Pseudo Inverse

The following linear solvers belong to SCAs. In this solver, `numpy.linalg.pinv`[3] is recalled for both $\mathbf{M}$ and $\mathbf{A}$, calculating the pseudo inverse. This method has the advantage of not requiring regularization to work. Unfortunately it is also the worst in both accuracy and computational cost.

[2] `https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html`
[3] `https://numpy.org/doc/stable/reference/generated/numpy.linalg.pinv.html`

4.3.4 Pseudo Inverse mixed with Cholesky

This is the hybrid solution referred to as 'mixed'; $\mathbf{A}$ is regularized with the method mentioned before and factorized with Cholesky decomposition, whereas $\mathbf{M}$ is treated computing the pseudo inverse.

This is probably the universal solution because it is always mathematically sustained , as it can be used in every situation and is always valid, also for this case a pro and cons table is presented (4.2).

Pros	Cons
Accurate	With increasing complexity the pseudo-inverse could become very costful
Quite fast	
Universal	

Table 4.2: Pros and cons of the mixed method.

4.3.5 Full Cholesky

In this case also $\mathbf{M}$ is regularized with a new parameter called β and Cholesky decomposition is used, and for this reason is referred to as 'fullcho'. The results derived with this method represent the best trade-off between accuracy and computational speed. However, it could be demonstrated (starting from the equation and returning to the original error) that this regularization is the same as adding $-\beta\|\boldsymbol{\lambda}\|_2^2$ to the original error. Adding a negative term to the error means that it is no longer possible to demonstrate that the solution correspond to a maximization problem. Furthermore, there is no literature on Lagrange multiplier regularization, because it somehow changes the constraint. Nonetheless if $\|\boldsymbol{\lambda}\|_2 << 1$ this two situations should not occur. In table 4.3 the pros and cons.

Pros	Cons
The fastest	Weak mathematical basis
Very accurate	

Table 4.3: Pros and cons of this full Cholesky method.

4.3.6 Comparison

To enforce the previous affirmation, a comparison between the capabilities of these solvers is reported. For this comparison, the velocity matrix in 5.2 is used as a test case. The errors are calculated computing the norm of the residuals. For instance $\mathbf{b_1}$ error is:

$$\frac{\|\mathbf{A}\cdot\mathbf{w}+\mathbf{B}\cdot\boldsymbol{\lambda}-\mathbf{b_1}\|_2}{\|\mathbf{b_1}\|_2}$$

All computations were done in the same laptop and the time is evaluated as a mean of ten distinct runs. In table 4.4 the results are presented.

Method	Time[s]	$\mathbf{b_1}$ error	$\mathbf{b_2}$ error
Solve	4.23	$4.21\cdot10^{-11}$	$2.33\cdot10^{-8}$
Fullpinv	33.39	$3.83\cdot10^{-5}$	$6.15\cdot10^{-4}$
Mixed	3.44	$5.43\cdot10^{-9}$	$7.36\cdot10^{-7}$
Fullcho	3.33	$5.38\cdot10^{-9}$	$1.13\cdot10^{-7}$

Table 4.4: Results.

4.4 Pressure evaluation algorithm

At this point, all the devices needed to evaluate pressure are known. The first part of the algorithm would be to use the clustering method to obtain fine collocation points. The linear system in (4.9) is solved with one of the strategies depicted in section 4.3, hence the weights are easily obtained. The forcing term could be calculated by using the matrix (3.4) and becomes:

$$\mathbf{f} = -\rho\left((\mathbf{D_X}\cdot\mathbf{w_u})\circ(\mathbf{D_X}\cdot\mathbf{w_u}) + 2\,(\mathbf{D_Y}\cdot\mathbf{w_u})\circ(\mathbf{D_X}\cdot\mathbf{w_v}) + (\mathbf{D_Y}\cdot\mathbf{w_v})\circ(\mathbf{D_Y}\cdot\mathbf{w_v})\right). \tag{4.14}$$

The forcing term is known and the boundary condition can be divided in two types:

- known Dirichlet condition, which should be at least one to ensure the uniqueness of the solution;
- Neumann condition from the momentum equation.

Finally, by solving the linear system in (3.23), the weights for pressure are obtained.

Usually, the same grid point and RBF are used for both velocity and pressure, because as the information about the velocity is filtered from its approximation then the forcing term would not require different c to be described. Furthermore also changing the grid points would not have any meaning for the same reason.

4.5 Python implementation

This algorithms has been implemented in Python and is based on two main module:

- `Matrix.py`
- `MESH_LAB.py`

4.5.1 `Matrix.py`

`Matrix.py` is the module that manages the creation of all kind of matrices needed for the computation as (3.2) or (3.4). These matrices are then used to create the matrix to be solved to minimize the error, so (3.23) for the Poisson equation, or (4.3) for the velocities approximation.

4.5.2 `MESH_LAB.py`

This module is more complex than the previous and deals with the solution of the involved linear systems and the clustering. Therefore, it includes a function to solve the linear system associated with the Poisson solver and the function approximation. The Poisson solver cannot work by itself so a different function is needed to create the boundary conditions for the Poisson solver.

The clustering method function has a very complex flow chart shown in figure 4.2. The inputs are the data position and the average number of particles or collocation points per Gaussian. This last variable is particular: it is actually an array because it may change for every level of clustering, and therefore its length is equal to the number of level chosen by the user. Furthermore, two other variables can be set namely an upper limit on the c's to avoid unusual high shape parameters, and a boolean array named mincluster, which prevents the creation of fake high c's due to small clusters. If 'mincluster' is set to true for one level, every associated Gaussian with a cluster smaller than the average number of required particles, which should led to high c's, is associated with the minimum c of that level.

The 4.2 diagram provide a description of the algorithm. The data positions are elaborated by `MinBatchKMeans` and the collocation points are obtained. The closest collocation point is evaluated and exploiting the rule of thumb in (4.10) c's are calculated. Then, the c's that exceed the imposed limit are turned into the values of the `cap` itself. If `mincluster` hold true those clusters with less points than the average inside them are associated

with the lowest c of the same level. Finally, if all the levels are completed, the collocation are given as outputs; otherwise the presented process is repeated.

The second proposed function deals with the linear system of the approximation of the velocities; in figure 4.2 the flow chart of the function is shown. The inputs generally consist of collocation points, shape parameters, constraints and all the information about the data, such as grid points and values of the velocities in such points. Some other inputs are optional, but important nonetheless. The regularization is set to zero by default but it can be changed manually; similarly for the divergence-free penalty is not present by default but it can be added. If the regularization is set to zero, a way to solve the linear equation must be indicated. Recalling the tags given in 4.3, this variable, named `method`, can be set to 'fullcho', 'fullpinv' or 'mixed', with the default option being 'fullcho'.

Following what is prescribed in section 4.2, the inputs are used to create the matrices calling `Matrix.py`, obtaining the final matrix (4.3). If any penalties or regularization are present the matrix is build differently. In both cases if the regularization is not prescribed by the user one of the SCAs method is used, which can be decided through the `method` variable. On the other hand, if a regularization is imposed the code use `numpy.linalg.solve` to solve the whole matrix. In both cases the weights are calculated and given as outputs.

It was mentioned before that a function is needed to create the boundary conditions for the Poisson equation. This function, and the Poisson solver as well need both collocation points for velocity and pressure. Hence, it is important to point out that such collocation points do not always coincide. The weights previously calculated for the velocities are needed to compute the momentum equations and therefore the Neumann conditions. Moreover, an array containing the constrained points and an array containing the type of condition for every edge is required. Every specific condition type needs specific information: Neumann conditions need the direction normal to the edge where it is applied, and Dirichlet conditions need the relative assigned value. Finally physical properties are needed: viscosity and density.

In figure 4.3 the flow chart for the boundary conditions function is proposed. Firstly, the function looks whether a particular edge is subjected to Dirichlet or Neumann conditions. In the first case matrix (3.2) is built over the constrained point and an array containing the relative condition is created. In the Neumann case the differentiation matrices $\mathbf{D_X}$ and $\mathbf{D_Y}$ are created again over the constrained points. The velocity approximation is used to calculate $\partial_x p$ and $\partial_y p$ from the momentum equation. Then the differentation matrices and the pressure derivatives are projected to obtain $\mathbf{D_n}$ and $\partial_n p$. Finally, if all edges are assigned with a condition, the function stacks all of the matrices and boundary conditions into one matrix and one array; otherwise the cycle is repeated.

The final proposed function is the Poisson solver. This has several inputs: the collocation point and shape parameters for both velocity and pressure, the boundary matrix and conditions as obtained from the previous function, the weights for the velocities and the parameters defining which linear solver to use. As before, physical properties are needed.

In figure 4.4 a flow chart for this function is reported. This last implementation does not differ too much from the function for the function approximation; the main differences are that, int this case no penalties can be added, the linear system to be solved is (3.23), and the forcing term must be calculated as equation (4.14) prescribes.

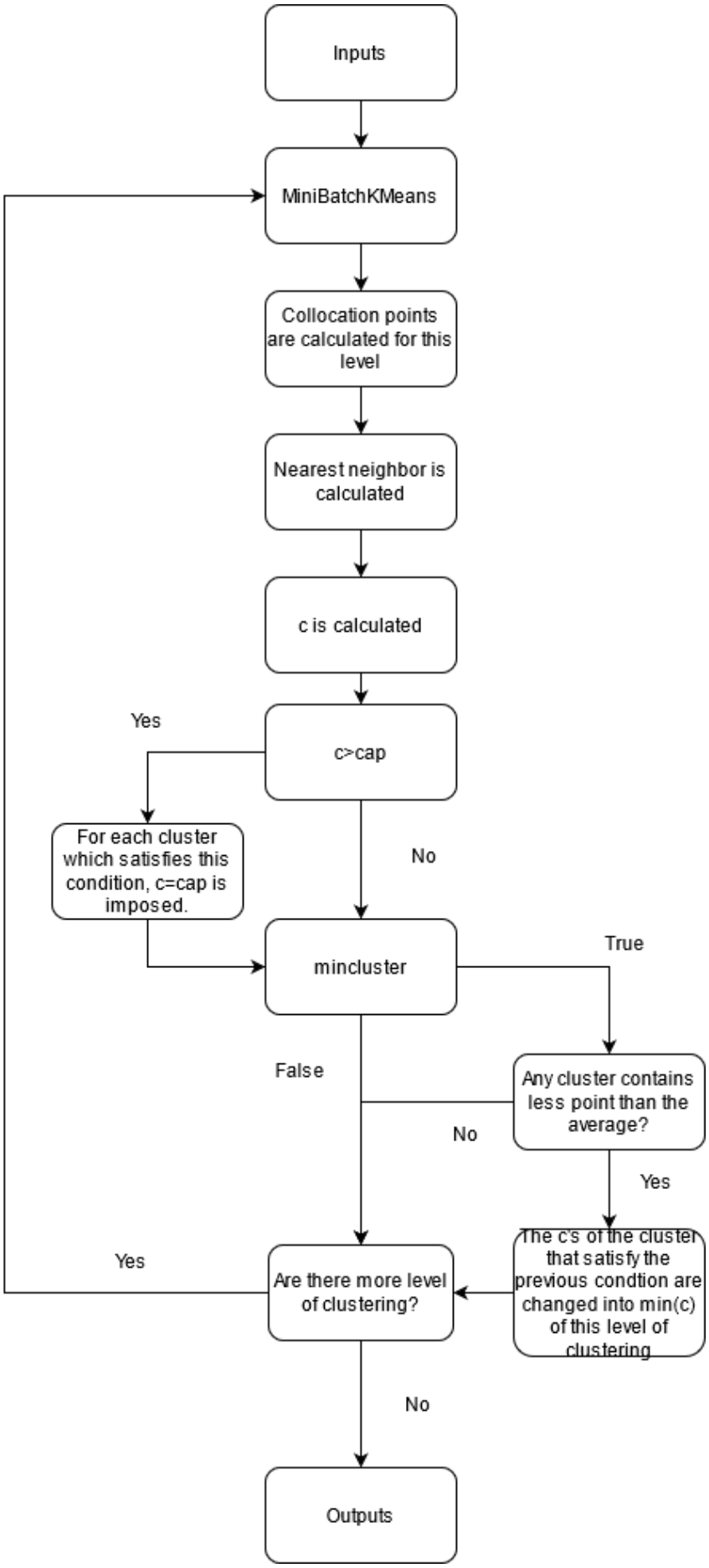


Figure 4.1: Flow chart of the clustering method.

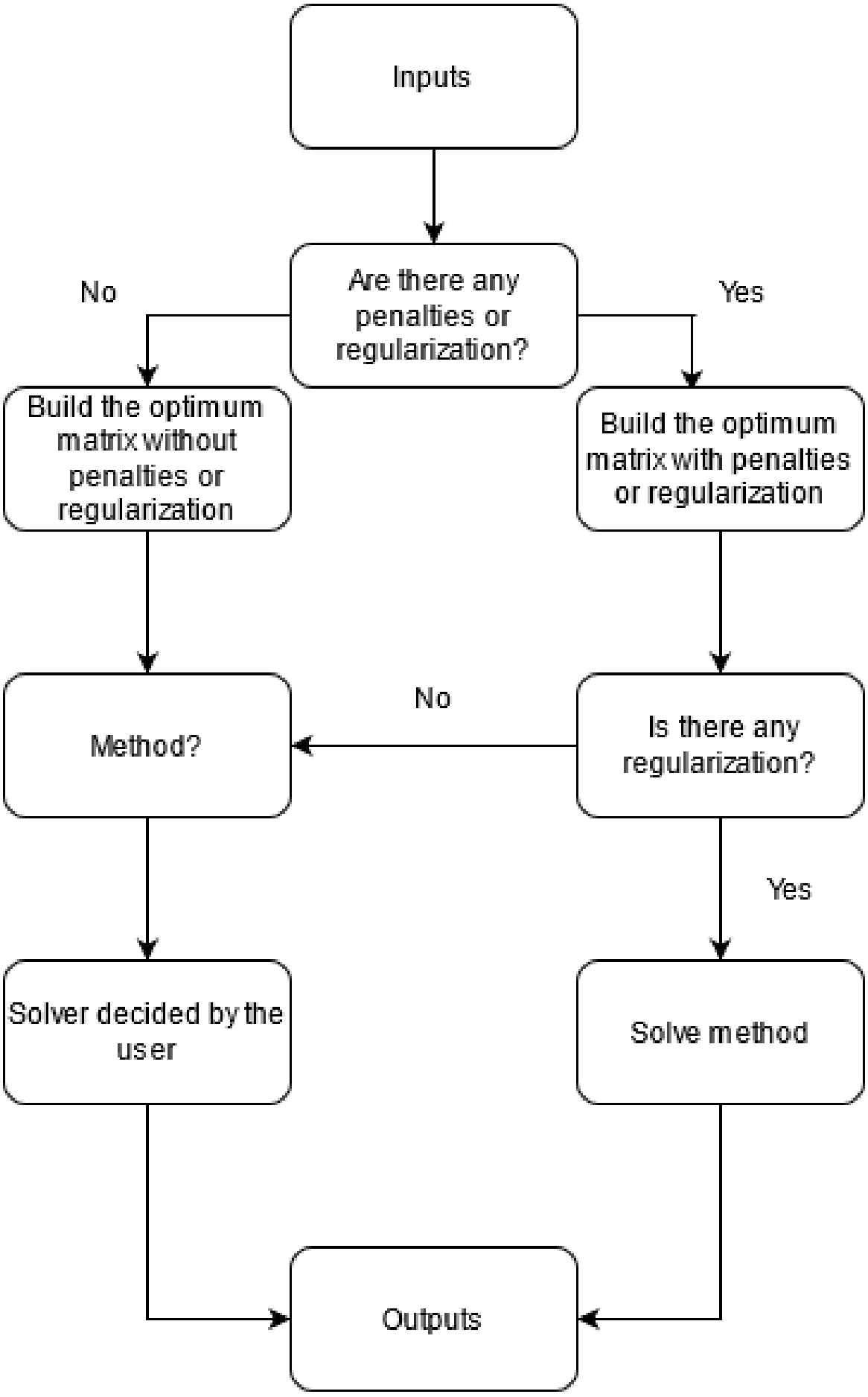


Figure 4.2: Flow chart of the function approximator solver.

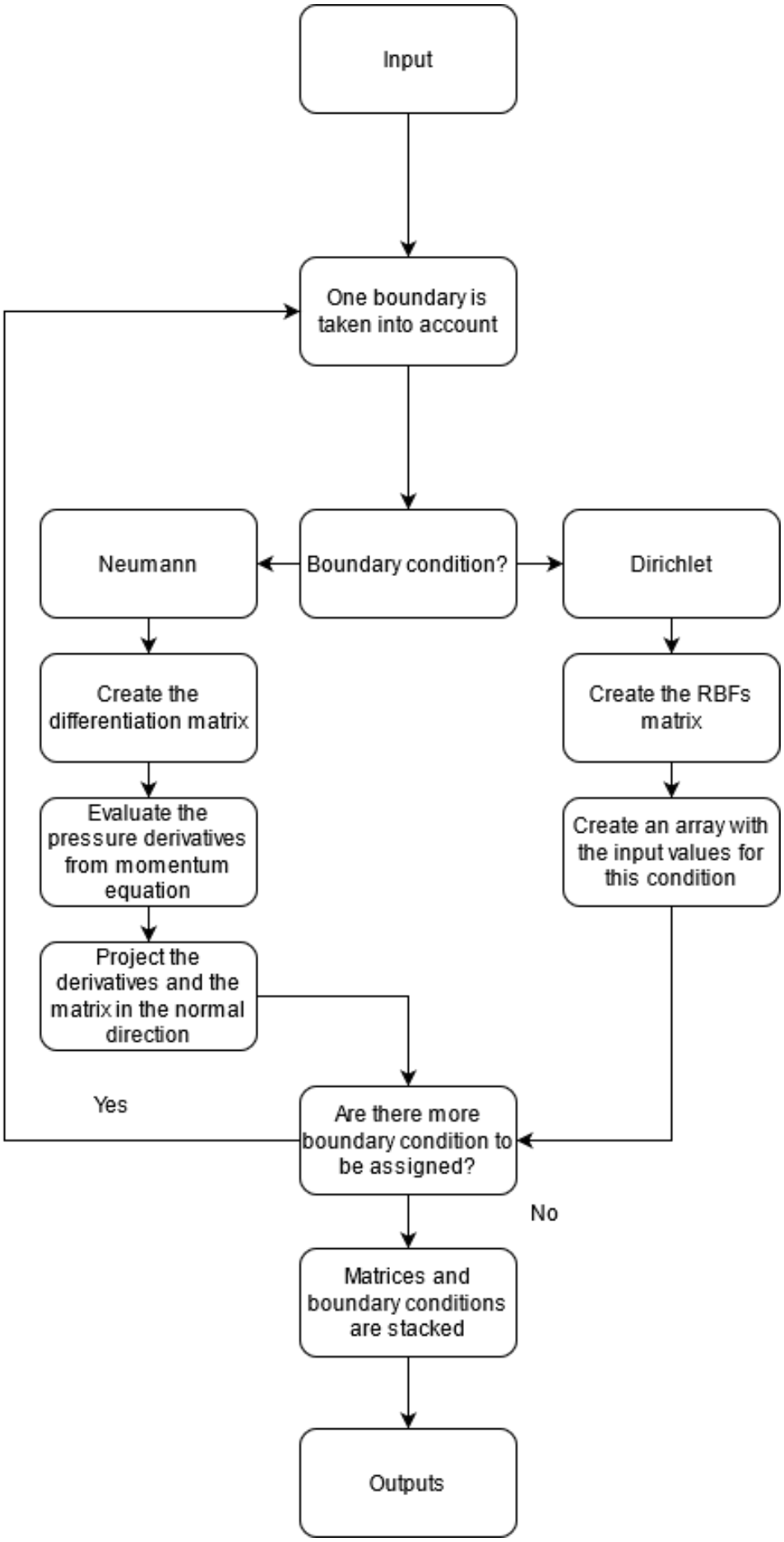


Figure 4.3: Flow chart of the boundary condition function.

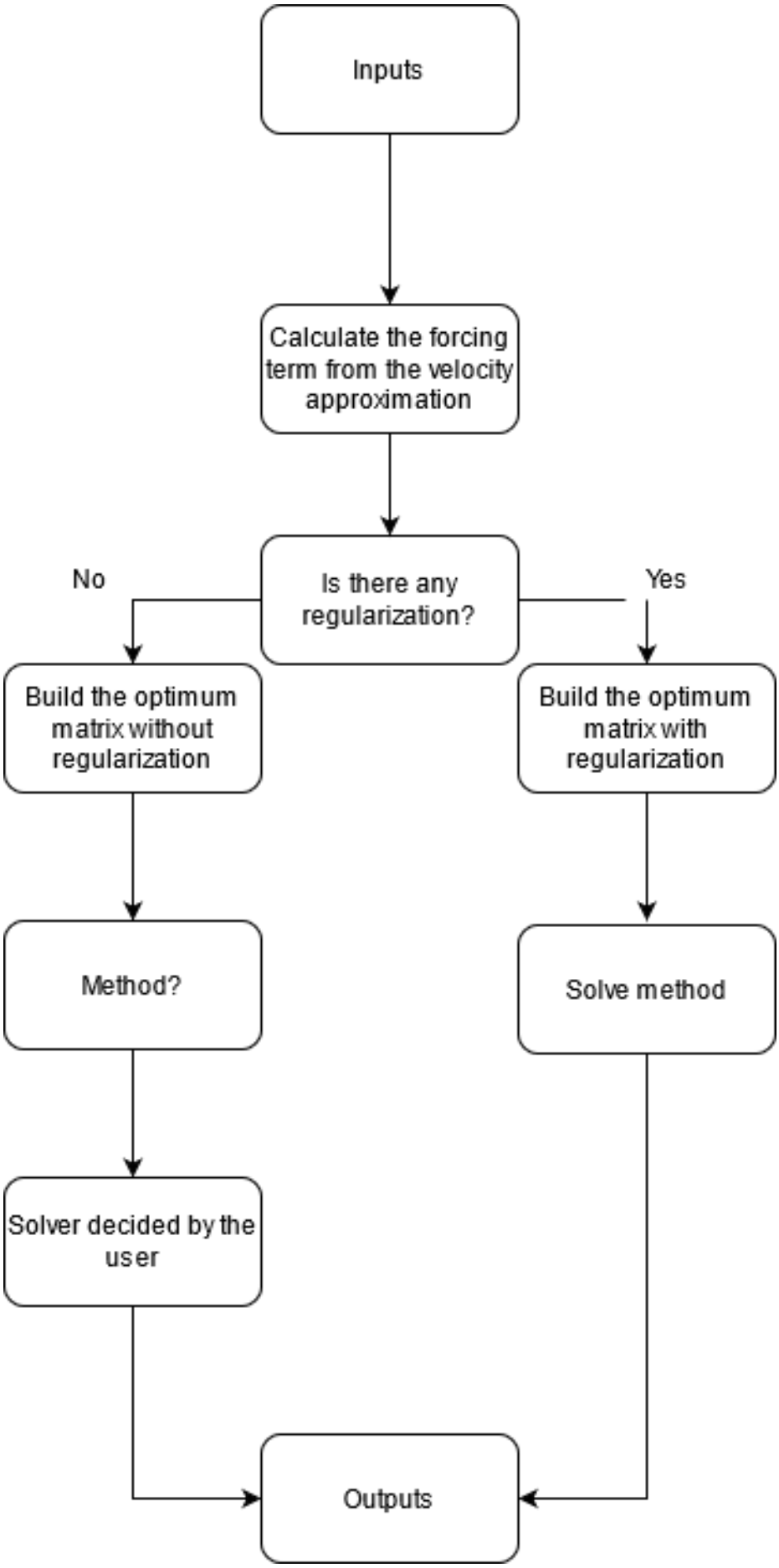


Figure 4.4: Flow chart of the Poisson solver function.

Chapter 5

Test Cases

This chapter introduce the test cases used to validate the whole algorithm and the Python implementation. Two test cases of increasing complexity were carried out. This difference in complexity opened the opportunity to exploit the simpleness of the first test case and achieve a wider study on the effect of the different parameters. On the other hand, the complexity of the second demonstrates that the proposed method holds its validity even if complex problems are analyzed. Unfortunately, these test case are still relatively simple. Indeed they are:

- stationary;
- inviscid /laminar;
- bi-dimensional.

5.1 Gaussian Vortex

The first test case is the Gaussian vortex, mimicking to some extent the work of De Kat and Van Oudheusden (2012). In this first test, the velocity field and the pressure are analytically known.

The tangent velocity distribution is defined as follows:

$$V_\theta = \frac{\Gamma}{2\pi r}\left(1 - e^{-\frac{r^2}{c_\theta}}\right) \tag{5.1}$$

where Γ is the circulation and is set to 10, $c_\theta = r_C^2/\gamma$ with $\gamma = 1.256431$, the core radius r_C is set to 0.1 $[m]$. Finally the density is set to $1[kg/m^3]$. The tangent velocity could be evaluated in some arbitrary point. Indeed, this formula has been used to create synthetic velocities over scattered grids. The pressure field is known as well:

$$p = -\frac{1}{2}\rho V_\theta^2 - \frac{\rho\Gamma^2}{4\pi^2 c_\theta}\left(E_1\left(\frac{r^2}{c_\theta}\right) - E_1\left(\frac{2r^2}{c_\theta}\right)\right). \tag{5.2}$$

where E_1 is the exponential integral, defined as:

$$E_1(x) = \int_x^\infty \frac{e^{-t}}{t}\,\mathrm{d}t.$$

Hence, the evaluation of the pressure error can be easily achieved by calculating the difference between the numerical and analytical solution. The domain of the Poisson equation is a $[-0.5, 0.5] \times [-0.5, 0.5]$ square. All the edges are subject to the Neumann conditions derived from momentuum equation, with the exception of the bottom edge that has a Dirichlet condition as proposed by De Kat and Van Oudheusden (2012):

$$p(x, -0.5) = -\frac{1}{2}\rho V_\theta(x, -0.5)^2,$$

where $p_\infty = 0$ is supposed.

5.2 Cylinder at Re=5

The second test case is a laminar two-dimensional flow around a cylinder. The reference data are from Rao et al. (2020), which are freely downloadable[1]. These data were obtained with `ANSYS Fluent 18.1`.

In figure 5.1 the dimensions of the flow are presented. The density of the fluid is 1 $[kg/m^3]$ and the viscosity is $2 \cdot 10^{-2}$ $[Pa \cdot s]$. The inlet velocity profile is parabolic: $4(H-y)y/H^2$.

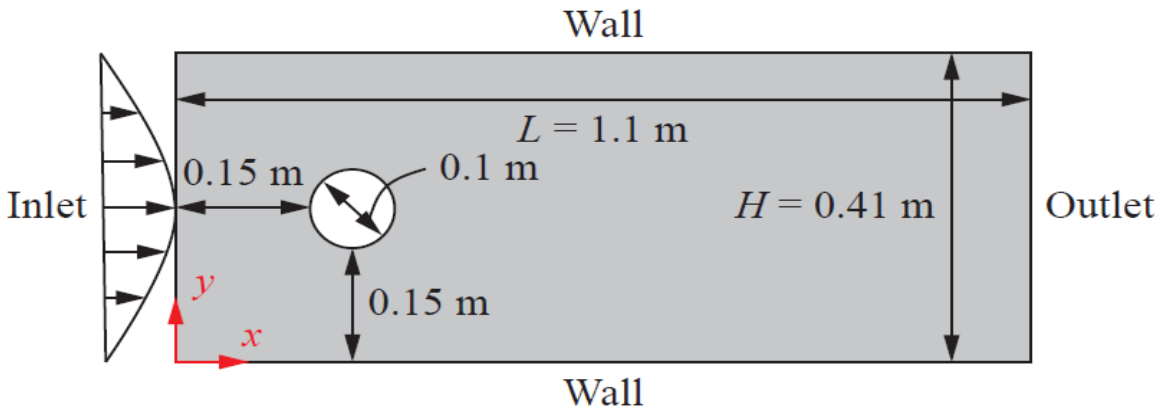


Figure 5.1: Geometries of the flow, Rao et al. (2020).

Exactly 19340 scattered points define the pressure and the velocity field. To simulate a realistic situation, the velocity data at the borders were deleted, so the total points drop to 18755.

The imposed boundary conditions are of Neumann type and are calculated as usual with the momentum equations. The only exception is the outlet, where pressure is forced to reach 0 (Dirichlet condition).

[1] `https://github.com/Raocp/PINN-laminar-flow`

Chapter 6

Results

This chapter presents the results of the test cases described in chapter 5. As previously mentioned, the Vortex test case is more suitable for an extensive test where parameters such as:

- regularization
- penalties
- noise
- number of particle

are studied in depth. Instead, the cylinder works as code validation on a complex problem. It must be pointed out that these results were obtained with the method explained in this section 4.3.2, as this is the best solver in terms of accuracy. The only exception to this rule is the cylinder with the clustering. For this case, the behaviour within the noise was too poor. Therefore a Mixed approach (4.3.4) is instead preferred.

6.1 Vortex

Concerning configuration for the vortex, 50 constraints points are prescribed. At this point, all constraints are applied: divergence-free, velocity inlet, Dirichlet conditions, *etc.* The constraint points would be the same for pressure and velocities.

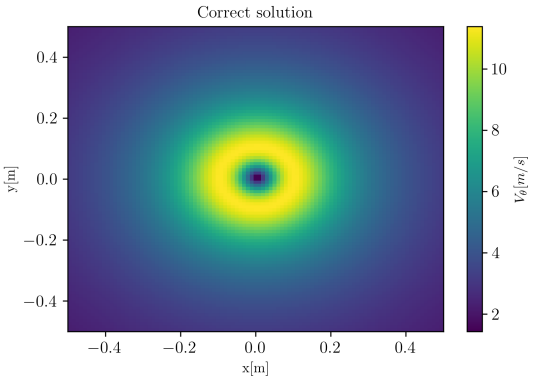
The field for the vortex is computed in two manners:

- the first, uses 3025 RBF on a uniform grid with a shape parameter of 15;
- the second, uses clustering technique.

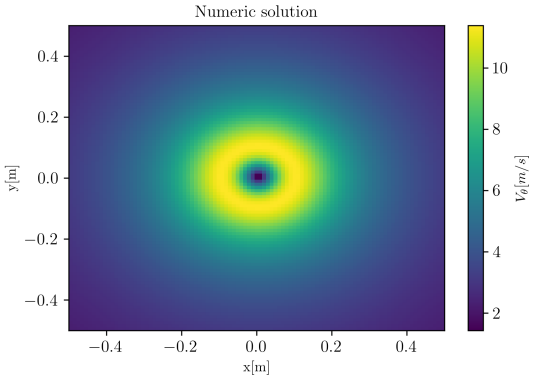
6.1.1 Vortex with standard collocation

The best solution is chosen to be compared to the correct values. The setup for this solution is proposed as follows:

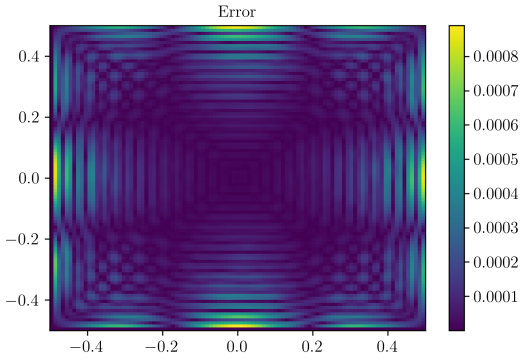
- zero noise
- number of point N=20000
- divergence-free penalty
- $\alpha = 10^{-5}$, that is the regularization parameter for both velocity and pressure



(a) Exact solution (5.1).



(b) Function approximation.



(c) Error between a and b.

Figure 6.1: Velocity approximation.

In figure 6.1 and 6.2, the solution obtained for this scenario are shown.

This test case also represents a great opportunity to demonstrate that this function approximator acts like a low-pass filter. In figure 6.4 a glimpse of this effect is shown. In particular, two approximations are proposed with the same number of RBFs but with different c's that determine the cut-off frequency. This means that even an infinite number of RBFs cannot perfectly represent the function if c's are finite.

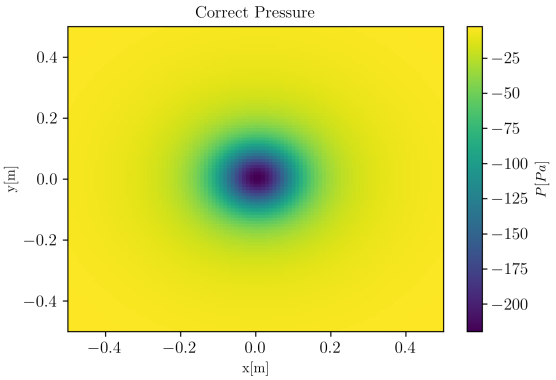
The following paragraphs focus on study of the effects of the setup parameters on the solution. In particular, some setups are considered:

- $\alpha = 10^{-2}$ without divergence-free penalty;
- $\alpha = 10^{-5}$ without divergence-free penalty;
- $\alpha = 10^{-5}$ with divergence-free penalty.

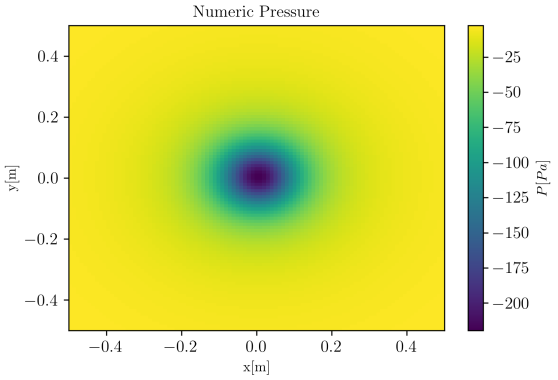
Figure 6.3 illustrates the behaviour of the different setups with a variable number of particles, and it clearly shows that a high value of α deteriorates the solution. Meanwhile small alpha are giving a better result. The error is generally decreasing as the number of particles increases. The trend seemingly shows a plateau. The third image exhibits an exception.

Having the velocity given as a finite number of points means that somehow the original velocity has been already subjected to a low pass filter. Therefore, there must exist a maximum frequency that can be represented for any particles distribution f_N. Furthermore, if N increase also does f_N.

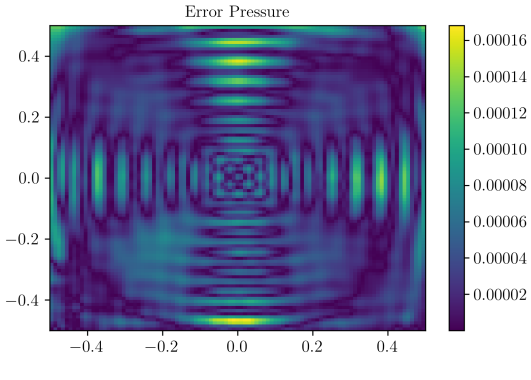
These data are approximated with fixed RBF. Therefore, the results are filtered with a cut-off frequency f_{RBF}. The collocation grid is fixed, so $f_{RBF} = const$. Hence, until $f_{RBF} > f_N$ the error improves by increasing



(a) Exact solution (5.2).



(b) Resulting pressure solution.



(c) Error between a and b.

Figure 6.2: Pressure comparison.

N as more frequencies can be represented. The plateau occurs because $f_{RBF} \approx f_{7000}$, then for $N > 7000$ new particles add information that would be deleted from the RBF, and no improvement to the error is achieved.

For the third case in Figure 6.3 a distinct discussion has to be done. In this case the plateau does not form. This is probably due to the high regularization that, as Figure 6.5 demonstrates, increases the low-pass filter effect. Additionally it is possible that since points are given randomly, certain regions might have a higher density of points, even when N is relatively small. As a consequence, such regions suffer more from this low-pass filter effect. Similarly, the effect of noise is investigated in figure 6.6. The best response to noise is obtained in example (a) followed by (c) and then (b). By analysing these figures it is clear that the divergence-free penalty acts as a physical regularization. If noisy data are present, the divergence-free is not automatically verified. Hence the system admits some difference on the velocities to reduce the error given by the divergence-free penalty; this difference is noise because it does not satisfy the divergence-free. This effect of the divergence-free penalty reduces the presence of α to just one purpose, that is enhancing the conditioning of the matrix. Hence it has to be taken as small as possible to make the system work.

An example of standard algorithm is compared with the new method proposed in this thesis. In particular, the forcing term of the Poisson equation is calculated with both methods with these parameters: N=20000 and 5% noise. Before digging any deeper it is important to explain how the standard method was implemented. Firstly the data are interpolated with `scipy.interpolate.griddata`[1] onto a new 100×100 uniform grid; secondly, gradients are evaluated with `numpy.gradient`[2], and eventually since all the derivatives are known, the forcing term is calculated. The correct forcing term has been calculated by using `numpy.gradient` directly on the true velocity.

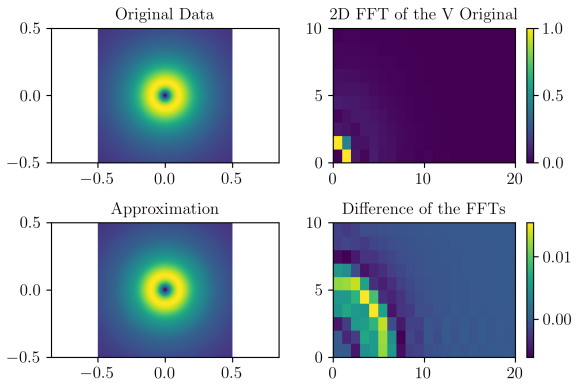
[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html`
[2] `https://numpy.org/doc/stable/reference/generated/numpy.gradient.html`

(a) $\alpha = 10^{-5}$ with divergence-free penalty

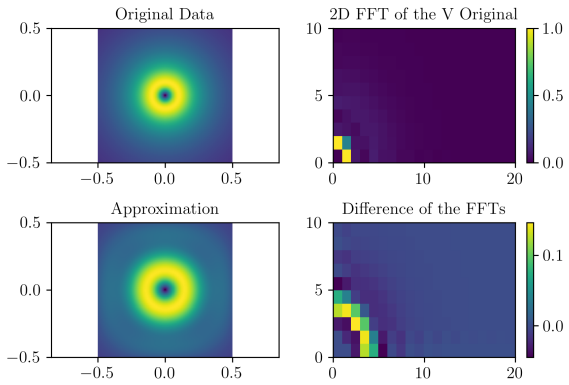
(b) $\alpha = 10^{-5}$ no divergence-free penalty

(c) $\alpha = 10^{-2}$ no divergence-free penalty

Figure 6.3: Error with varying particle N=[1000, 2500, 5000, 7500, 10000, 15000, 20000], noise=0%.



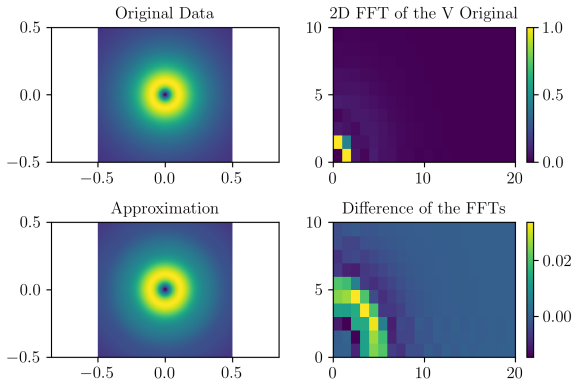
(a) c=5



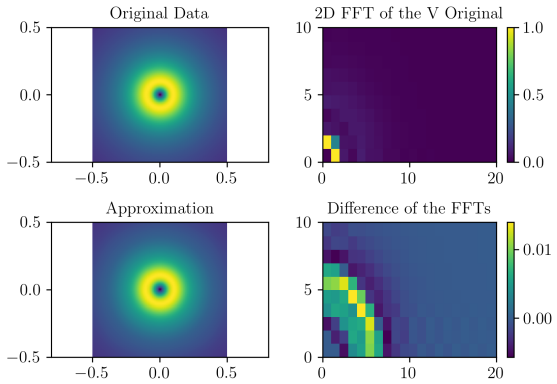
(b) c=3

Figure 6.4: Fourier transform of the approximation and the original data with different c's.

In figure 6.7 the results show evidence that the RBF results is better. RBF solution cannot actually be distinguished from the correct one. Indeed, the relative L2 error is 0.025. The standard results, instead, have points dropping below zero and do not appear as smooth as they should be. In this case, the relative L2 error is 0.31.

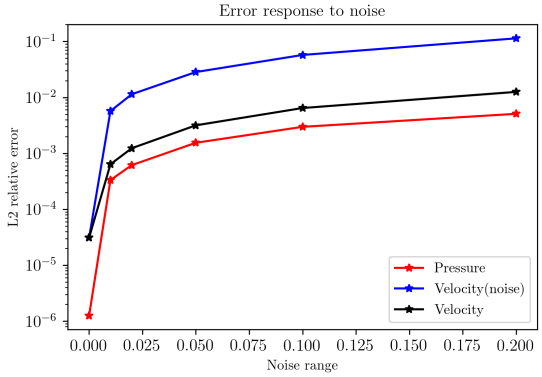


(a) c=5 and $\alpha = 10^{-2}$

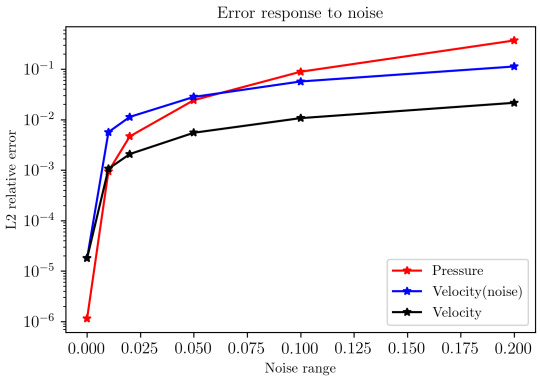


(b) c=5, $\alpha = 10^{-5}$ and divergence-free penalty

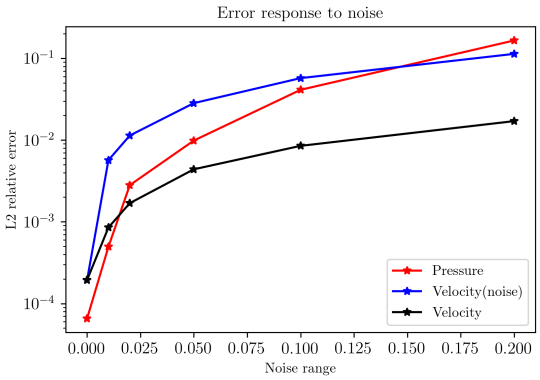
Figure 6.5: Effect of regularization on the spectra.



(a) $\alpha = 10^{-5}$ with divergence-free penalty



(b) $\alpha = 10^{-5}$ no divergence-free penalty

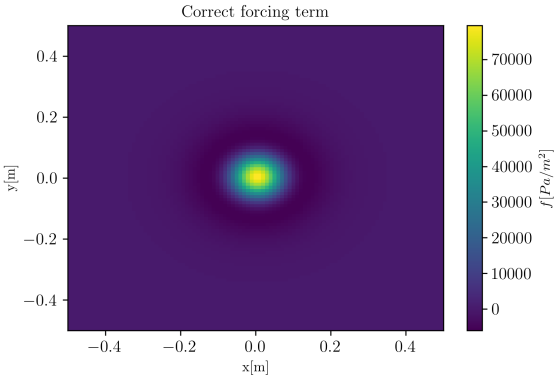


(c) $\alpha = 10^{-2}$ no divergence-free penalty

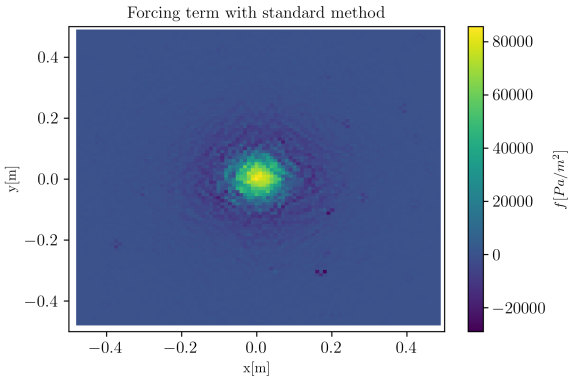
Figure 6.6: Error with varying Noise=[0%,1%,2%,5%,10%,20%], N=20000.

6.1.2 Clustering on Vortex

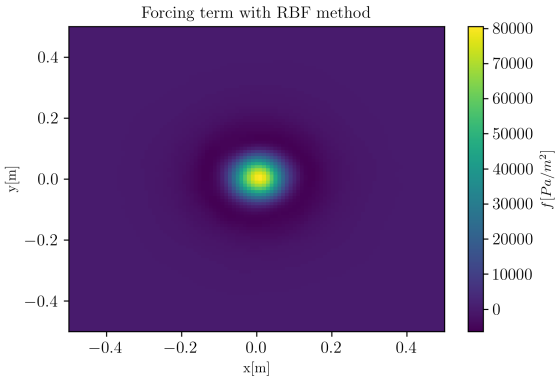
Next, the clustering is applied (Fig.6.8) by considering clustering layers, with the first layer of Gaussian containing 7 particles and the second containing 4 collocation point of the first layer. The results are similar to the ones previously obtained. However, the plateau does not form because the Gaussians are evaluated with respect to the data, then $f_{RBF} \neq const$. These results suggest that $f_{RBF} < f_N$ for any N studied. Velocity shows an error different from before. This could be probably linked with the changing number of RBF. In addition, all



(a) Correct f

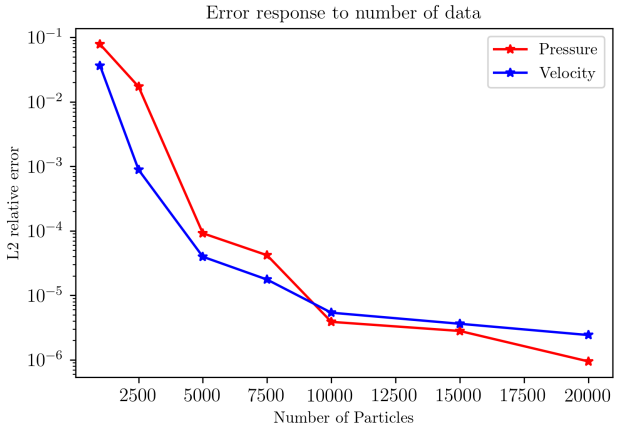


(b) Standard f

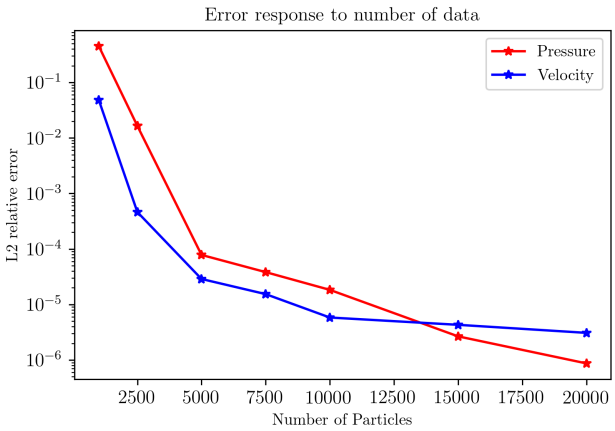


(c) RBF method forcing term

Figure 6.7: Forcing Terms.



(a) $\alpha = 10^{-5}$ with divergence-free penalty



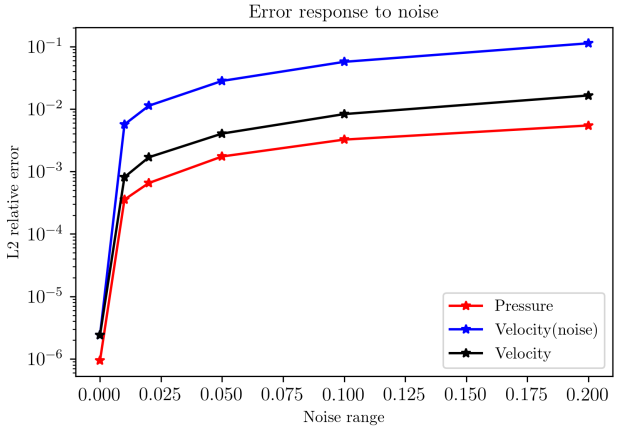
(b) $\alpha = 10^{-5}$ no divergence-free penalty

(c) $\alpha = 10^{-2}$ no divergence-free penalty

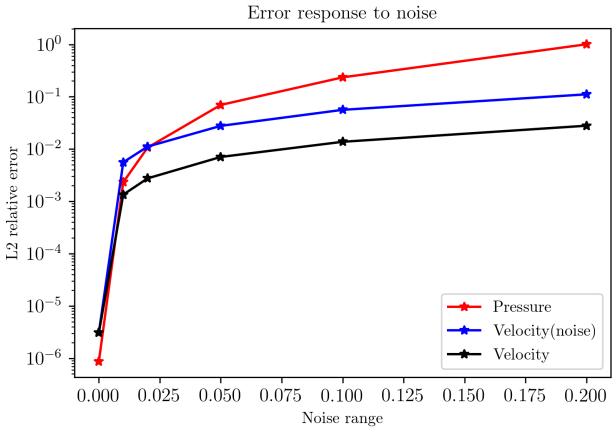
Figure 6.8: Error(Clustering) with varying particle N=[1000, 2500, 5000, 7500, 10000, 15000, 20000], noise=0%.

computations have nearly a monotonic decrease, especially for the case with $\alpha = 10^{-2}$, which is significantly different from the one of figure 6.3. This effect happens because c is locally chosen. Then the high-density regions are represented with higher c's and therefore avoid the flattening of those local frequencies as before. Finally, also the effect of the noise (Fig.6.9) is evaluated.

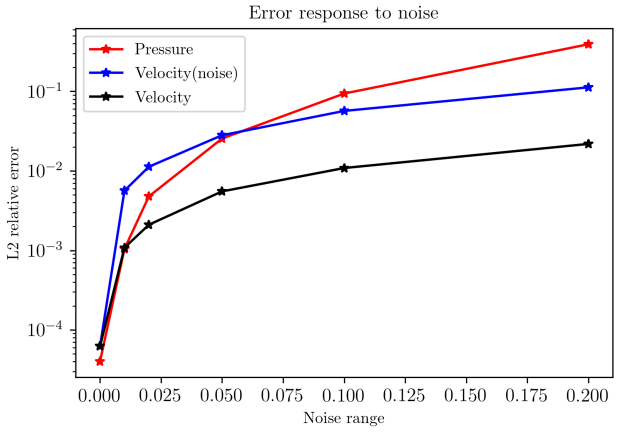
As a final consideration for this case, the error with the clustering is higher for lower N, this is due to the fact that the number of used RBFs are somehow dependent on N; therefore smaller N means less RBFs, and so a higher error. Nevertheless, the objective of the clustering method is not to improve the solution, but to automatically identify the Gaussians instead.



(a) $\alpha = 10^{-5}$ with divergence-free penalty



(b) $\alpha = 10^{-5}$ no divergence-free penalty



(c) $\alpha = 10^{-2}$ no divergence-free penalty

Figure 6.9: Error(Clustering) with varying Noise=[0%,1%,2%,5%,10%,20%], N=20000.

6.2 Cylinder at Re=5

The cylinder would have the same scheme as the vortex: first a standard collocation point, and then with the clustering method. In Figure 6.10, 6.11 and 6.12 the two velocities and pressure are presented as they are shown in Rao et al. (2020).

6.2.1 Cylinder with standard collocation

In this case, the correct collocation point and shape parameter has been found by trial and error. To obtain the solutions, two different but equally-spaced grids of collocation points were used; one is the overall domain

whereas the second is the annulus around the cylinder with a radius of $0.07[m]$.

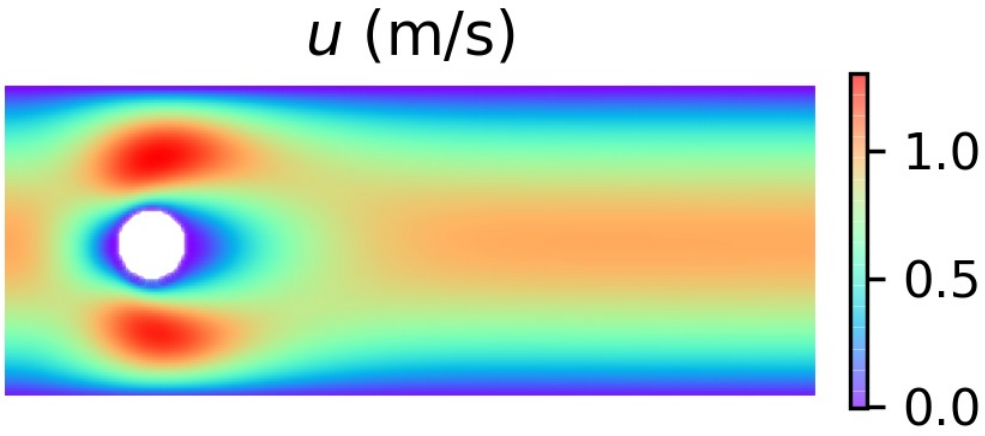


Figure 6.10: Velocity in x direction, Rao et al. (2020).

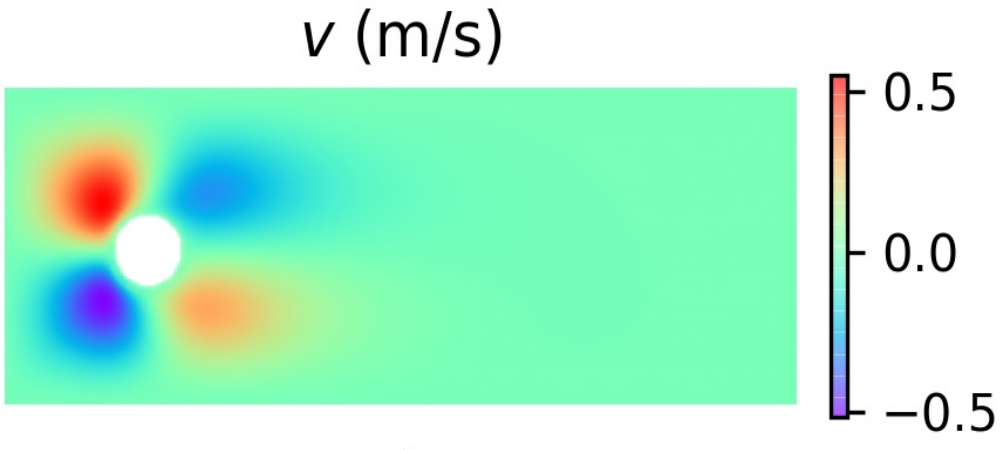


Figure 6.11: Velocity in y direction, Rao et al. (2020).

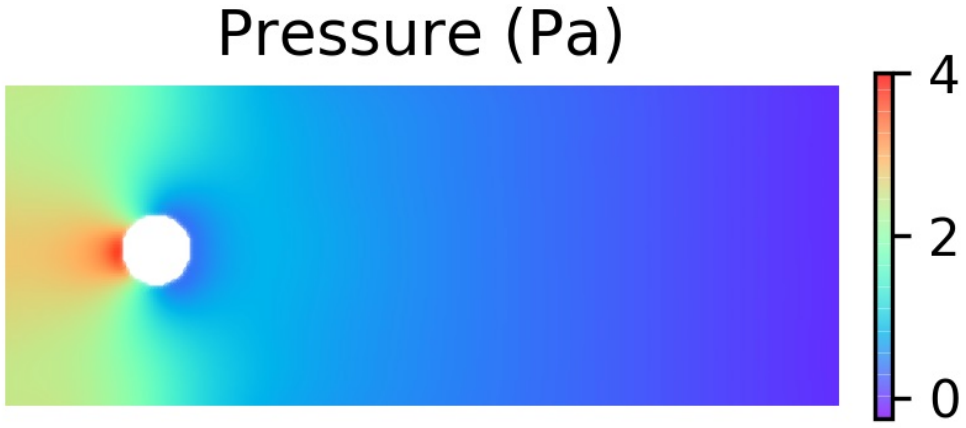


Figure 6.12: Pressure, Rao et al. (2020).

In table 6.1 the parameters for both the velocity interpolation and the PDE solver are reported. Every edge has an assigned label ,B1 to B4 refer to edges starting from the inlet and the rotating counterclockwise. B5 is

the cylinder edge. In table 6.1 all the edges are associated with some constraints for the function approximation or boundary conditions for the PDE solver.

	MESH1		MESH2(Annulus)			Constraints/Boundary conditions					
	Number of RBF	c	Number of RBF	c	α	B1	B2	B3	B4	B5	NC
Function approximation	1320	30	1978	52.5	0.8	df+inlet velocity	df+ns	df	df+ns	df+ns	80
PDE solver	5520	30	2970	52.5	0.5	N	N	D	N	N	50

Table 6.1: Parameters of interpolation and PDE solver where NC is the number of constraints per boundary. (df= divergence-free ,ns=no-slip, N=Neumann, D=Dirichlet)

The parameter fixed before are used to obtain the velocity field in Figure 6.14, 6.13, 6.16. The L2 norm relative error for both velocities u and v are $5.5 \cdot 10^{-3}$ and $1.43 \cdot 10^{-2}$ respectively. A complete overview of the comparison between data and results is given in figure 6.15.

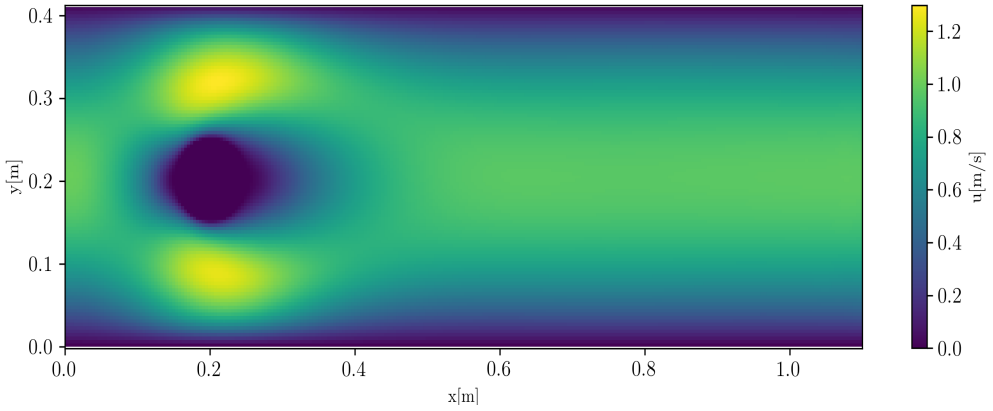


Figure 6.13: Velocity in x direction.

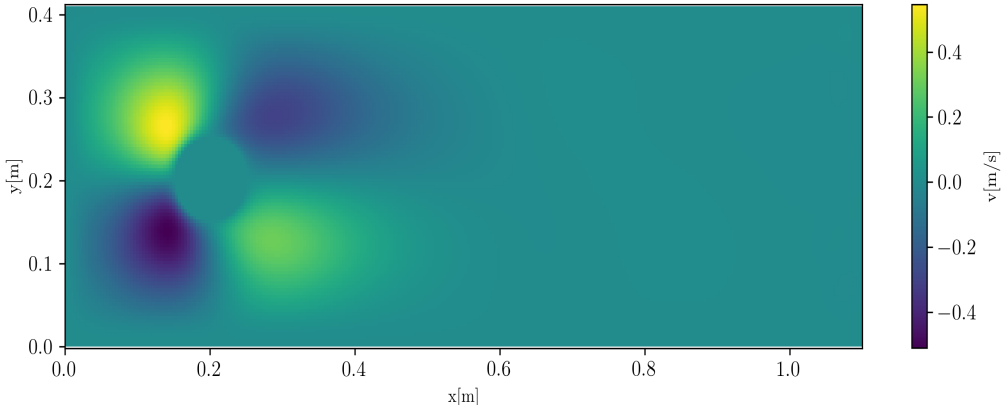


Figure 6.14: Velocity in y direction.

Pressure is computed in figure 6.17 with an L2-norm relative error of $5.6 \cdot 10^{-2}$; the computed values versus the data are then plotted in 6.18.

Finally, pressure is estimated around the cylinder, as is shown in Figure 6.19. The result matches the expected values, especially on the peak. On the other hand, precision decreases in correspondence of the opposite stagnation point. In this test case is possible to demonstrate the importance of the constraints, to achieve this the Neumann boundary conditions on the cylinder are computed in three different ways:

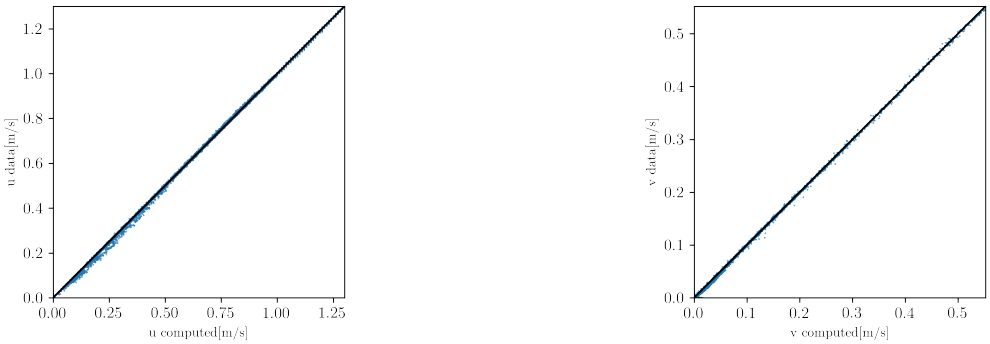


Figure 6.15: Comparisons between computed and real velocities. The black line has a 45^o slope and represent the approximation with zero error. The further the blue points are from the black line worse is the computed value.

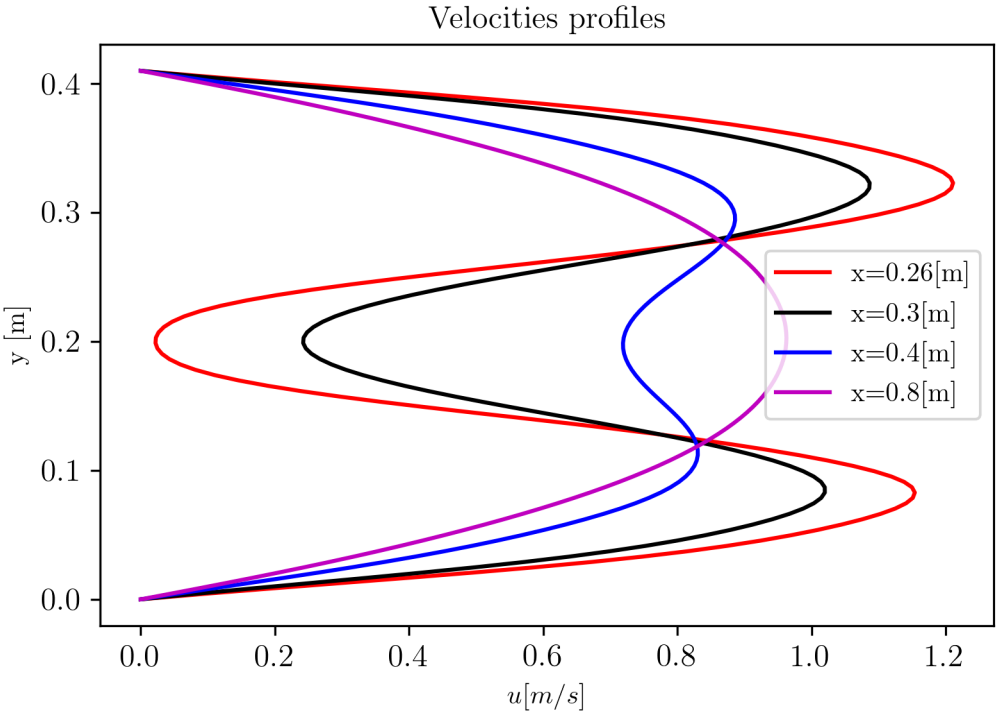


Figure 6.16: u at different y behind the cylinder.

- normally with no-slip condition and divergence-free;
- only divergence-free;
- no constraint.

Obviously, if the no-slip constraint is not applied in the function approximation, then the point subjected to that condition are added to the error. For all calculations the parameters were the same as in Table 6.1. In figure 6.20 the difference in the quality of this boundary condition is shown. The case with both constraints is smooth,symmetric and surely correct as these conditions were used to calculate pressure. On the other hand, the remaining two cases show deteriorating quality and are not suitable to compute pressure.

A careful reader would not miss that in this case the divergence penalties has not be applied. Indeed, this case with standard collocation point is not tested with noise, that would make this penalty necessary.

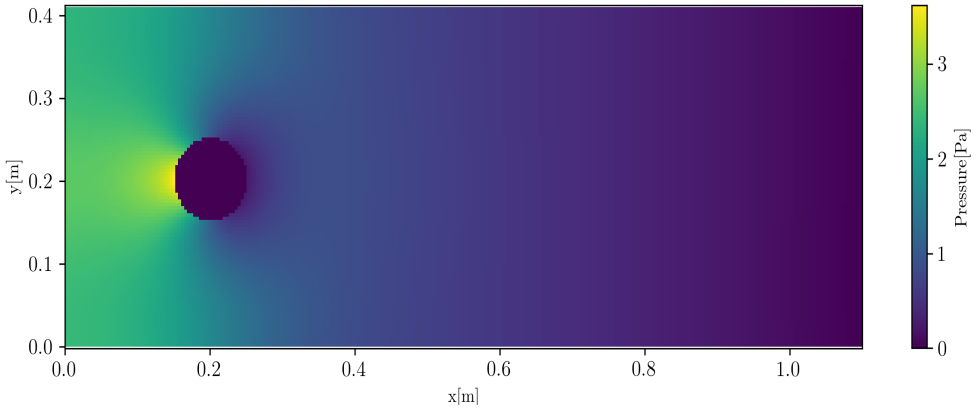


Figure 6.17: Pressure field of the cylinder.

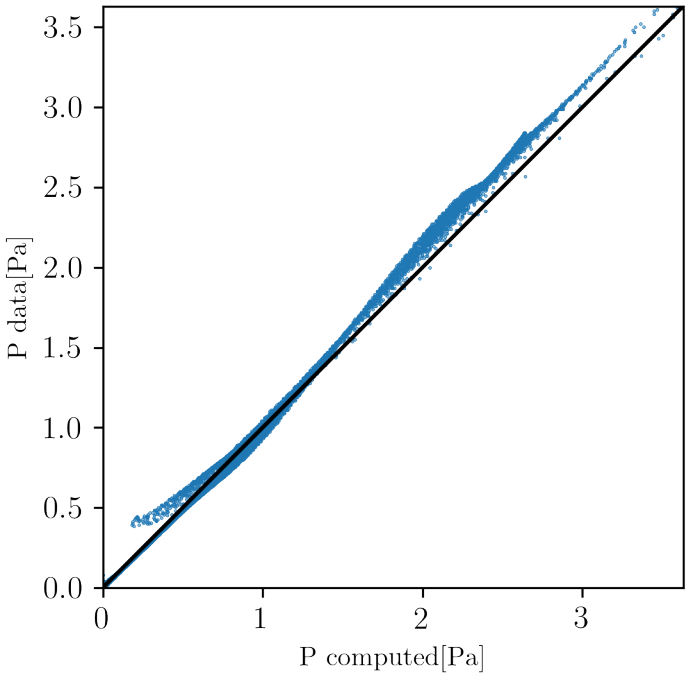


Figure 6.18: Pressure comparison.

6.2.2 Cylinder with clustering method

The collocation points are built on three level of clustering. Having in the first level 10 particle per Gaussian and then 2 level with 5 Gaussian per collocation point. The resulting number of RBFs is 2797, which is the same for both velocity and pressure. The boundary conditions are the same reported in 6.1, but in this case the divergence-free penalty is added; therefore the regularization changes as imposed the mixed solver.

The errors are reported in Table 6.2. Some improvements with respect to the previous case can be spotted.

	u	v	p
L2 norm relative error	$3.86 \cdot 10^{-3}$	$1.67 \cdot 10^{-2}$	$2.72 \cdot 10^{-2}$

Table 6.2: Errors with clustering method.

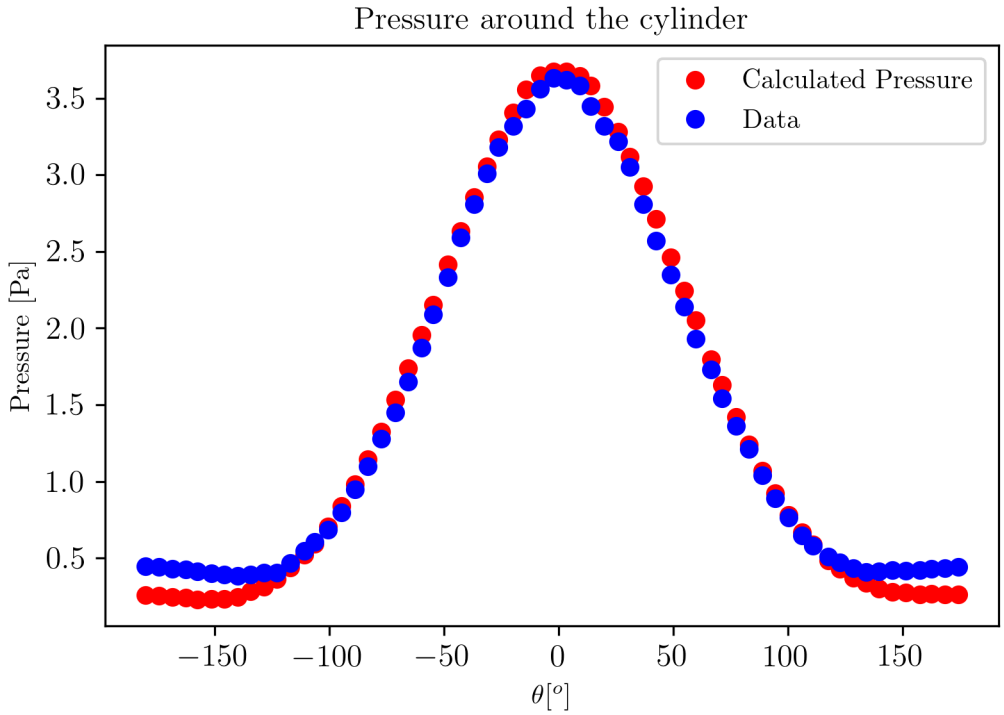
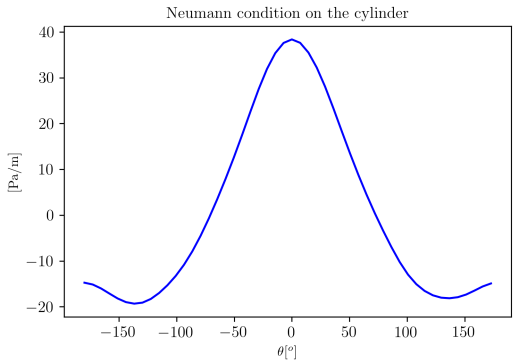


Figure 6.19: Pressure around the cylinder.

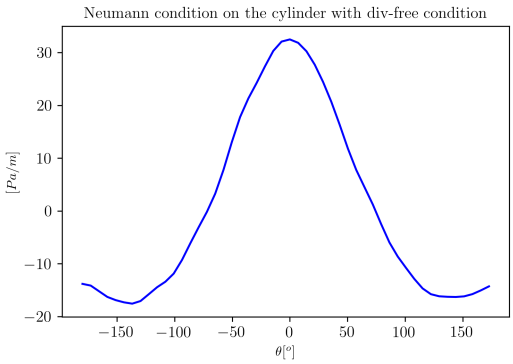
Nearly all plots are indiscernible between the two cases, so it is pointless to plot it again. However, pressure around the cylinder improves consistently (Figure 6.21). Finally, different cases with varying noise is presented in Table 6.3.

Noise	u	v	p
1 %	$3.9 \cdot 10^{-3}$	$1.69 \cdot 10^{-2}$	$2.73 \cdot 10^{-2}$
2 %	$4.01 \cdot 10^{-3}$	$1.71 \cdot 10^{-2}$	$2.74 \cdot 10^{-2}$
5 %	$4.74 \cdot 10^{-3}$	$1.86 \cdot 10^{-2}$	$2.81 \cdot 10^{-2}$
10 %	$6.74 \cdot 10^{-3}$	$2.3 \cdot 10^{-2}$	$2.92 \cdot 10^{-2}$
20 %	$1.16 \cdot 10^{-2}$	$3.51 \cdot 10^{-2}$	$3.16 \cdot 10^{-2}$

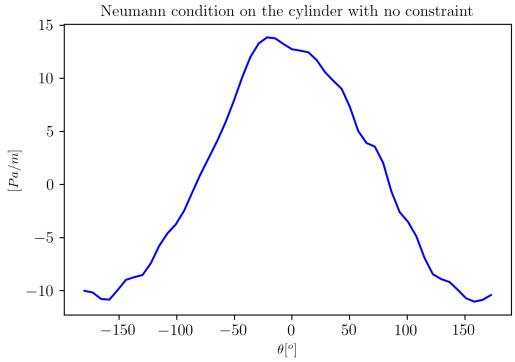
Table 6.3: Relative L2-norm errors.



(a) No-slip and divergence-free



(b) Only divergence-free constraint



(c) No constraint

Figure 6.20: Boundary condition around the circle, 0^o is the stagnation point which faces the inlet.

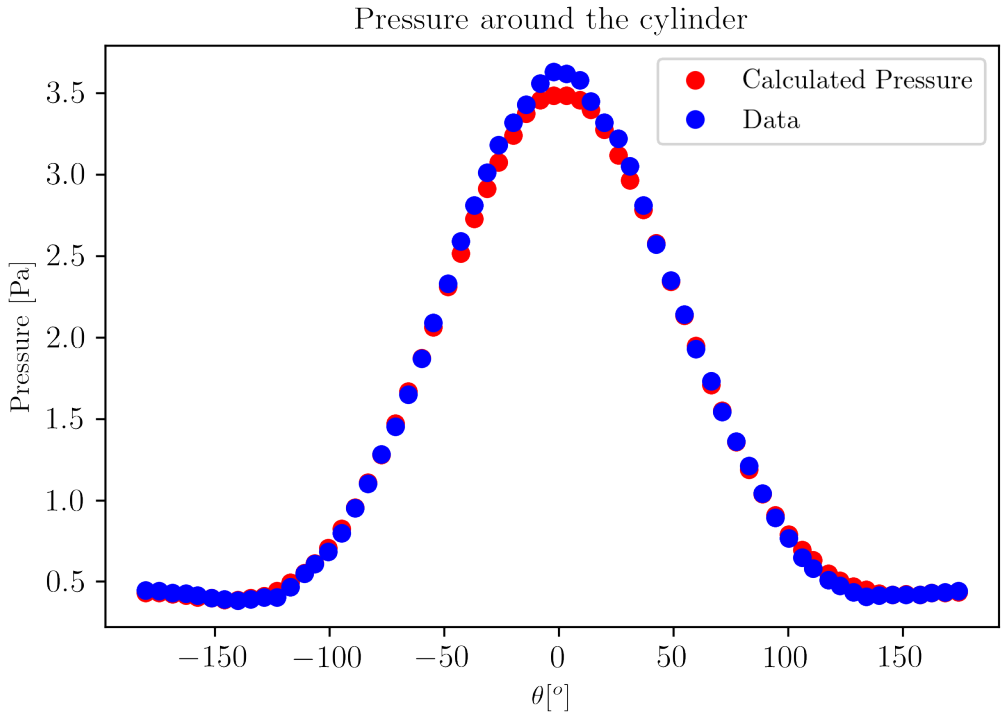


Figure 6.21: Pressure around the cylinder.

Chapter 7

Conclusion and perspectives

The aim of this work was to achieve a meshless method to compute pressure. However, this was not the only target the method could reach. Indeed, the function approximation of the velocity already works perfectly to enhance data, trigger super-resolution, replace outliers and filter noise. Therefore, even if the computational cost of the proposed method would be higher than each separate step, it could be competitive when compared to the whole procedure, which usually comprehends image pre-processing, filtering, *etc.*

Another advantage of this method is that, at the end of the calculation, an analytical formulation of the field is derived. As a result no further investigation is necessary to evaluate pressure at points different from the original grid, or to compute the derivatives if required.

Many disadvantages are still evident; to solve complex problems the matrices are huge and full, requiring plenty of processing memory to handle them. Furthermore, the matrices are ill-conditioned, thus requiring expensive tools to treat them. Such issues are not dramatic at the moment, but they will probably be in turbulent 3D cases where the number of particles and therefore required RBFs increase.

Some improvements are probably needed in terms of efficiency of the implementation itself, for example:

- improve the memory allocation of the matrix by neglecting small values inside them to obtain a small level of sparsity;
- avoid some matrices multiplication by directly defining the resulting matrix;
- avoid loops to build the matrices.

Moreover, the numerical part needs to be developed further:

- if some level of sparsity is achieved, iterative methods as conjugate gradient could be used;
- implement a stochastic gradient descent to treat the problem.

Eventually, the last problem to deal with involves the deciding criterion for the number of constraints that are to be imposed to make the method works finely.

Appendix A

Theorems

Theorem 1 Let be $\mathbf{B} \in \mathbb{R}^{M\times N}$ with $N < M$ with $dim(Ker(\mathbf{B})) = 0$, hence the matrix $\mathbf{B^T} \cdot \mathbf{B} \in \mathbb{R}^{N\times N}$ is positive definite.

Proof $\mathbf{B^T} \cdot \mathbf{B}$ is semidefinite positive, therefore:

$$\mathbf{x^T} \cdot \mathbf{B^T} \cdot \mathbf{B} \cdot \mathbf{x} \geq 0 \qquad \forall \mathbf{x} \in \mathbb{R}^N.$$

However for the assumptions $dim(Ker(B)) = 0$, hence $\nexists\, \mathbf{x} \in \mathbb{R}^N |$

$$\mathbf{B} \cdot \mathbf{x} = \mathbf{0},$$

except for the trivial solution. Hence $\nexists\, \mathbf{x} \in \mathbb{R}^N |$

$$(\mathbf{B} \cdot \mathbf{x})^{\mathbf{T}} \cdot \mathbf{B} \cdot \mathbf{x} = 0,$$

therefore $\mathbf{B^T} \cdot \mathbf{B}$ is definite positive □

Theorem 2 Let be $A, B \in \mathbb{R}^{M\times N}$, then the matrix:

$$\mathbf{C} = \begin{pmatrix} \mathbf{A^T} \cdot \mathbf{A} & \mathbf{A^T} \cdot \mathbf{B} \\ \mathbf{B^T} \cdot \mathbf{A} & \mathbf{B^T} \cdot \mathbf{B} \end{pmatrix}$$

is semi defined positive.

Proof Apply the semi definite positive definition to the matrix $\mathbf{C}$:

$$\mathbf{w^T} \cdot \mathbf{C} \cdot \mathbf{w} \geq 0$$

Let be $\mathbf{w_1}, \mathbf{w_2} \in \mathbb{R}^N |$

$$\mathbf{x} = \begin{pmatrix} \mathbf{w_1} \\ \mathbf{w_2} \end{pmatrix},$$

Substitute it in the first equation and with the definition given above this is obtained:

$$\mathbf{w_1^T} \cdot \mathbf{A^T} \cdot \mathbf{A} \cdot \mathbf{w_1} + \mathbf{w_2^T} \cdot \mathbf{B^T} \cdot \mathbf{B} \cdot \mathbf{w_2} + \mathbf{w_1^T} \cdot \mathbf{A^T} \cdot \mathbf{B} \cdot \mathbf{w_2} + \mathbf{w_2^T} \cdot \mathbf{B^T} \cdot \mathbf{A} \cdot \mathbf{w_1} \geq 0.$$

The last term could be manipulated as follow:

$$\mathbf{w_2^T} \cdot \mathbf{B^T} \cdot \mathbf{A} \cdot \mathbf{w_1} = \left(\mathbf{w_2^T} \cdot \mathbf{B^T} \cdot \mathbf{A} \cdot \mathbf{w_1}\right)^{\mathbf{T}} = \mathbf{w_1^T} \cdot \mathbf{A^T} \cdot \mathbf{B} \cdot \mathbf{w_2},$$

therefore the condition change again to:

$$\mathbf{w_1^T} \cdot \mathbf{A^T} \cdot \mathbf{A} \cdot \mathbf{w_1} + \mathbf{w_2^T} \cdot \mathbf{B^T} \cdot \mathbf{B} \cdot \mathbf{w_2} + \mathbf{2w_1^T} \cdot \mathbf{A^T} \cdot \mathbf{B} \cdot \mathbf{w_2} \geq 0.$$

Let be $\mathbf{a}, \mathbf{b} \in \mathbb{R}^M|$

$$\mathbf{a} = \mathbf{A} \cdot \mathbf{w_1}$$

$$\mathbf{b} = \mathbf{B} \cdot \mathbf{w_2}$$

These two are substituted in the condition:

$$\mathbf{a^T} \cdot \mathbf{a} + \mathbf{b^T} \cdot \mathbf{b} + \mathbf{2a^T} \cdot \mathbf{b} \geq 0 \Rightarrow \left(\mathbf{a^T} + \mathbf{b^T}\right) \cdot (\mathbf{a} + \mathbf{b}) \geq 0,$$

that is verified $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^M \square$

Bibliography

Cheng, A.-D., Golberg, M., Kansa, E., and Zammito, G. (2003). Exponential convergence and h-c multiquadric collocation method for partial differential equations. *Numerical Methods for Partial Differential Equations: An International Journal*, 19(5):571–594.

De Kat, R. and Ganapathisubramani, B. (2012). Pressure from particle image velocimetry for convective flows: a taylor's hypothesis approach. *Measurement Science and Technology*, 24(2):024002.

De Kat, R. and Van Oudheusden, B. (2012). Instantaneous planar pressure determination from piv in turbulent flow. *Experiments in fluids*, 52(5):1089–1106.

De Kat, R., Van Oudheusden, B. W., and Scarano, F. (2008). Instantaneous planar pressure field determination around a square-section cylinder based on time-resolved stereo-piv. In *14th international symposium on applications of laser techniques to fluid mechanics. Lisbon, Portugal*.

Discetti, S. and Ianiro, A. (2017). *Experimental aerodynamics*. CRC Press.

Elsinga, G. E., Scarano, F., Wieneke, B., and van Oudheusden, B. W. (2006). Tomographic particle image velocimetry. *Experiments in fluids*, 41(6):933–947.

Fasshauer, G. E. (1996). Solving partial differential equations by collocation with radial basis functions. In *Proceedings of Chamonix*, volume 1997, pages 1–8. Citeseer.

Fasshauer, G. E. (1999). Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in computational mathematics*, 11(2):139–159.

Gesemann, S., Hunh, F., Schanz, D., and Schröder, A. (2016). From noisy particle tracks to velocity, acceleration and pressure fields using b-splines and penalties. In *Int. Symp. on Applications of Laser Techniques to Fluid Mechanics*.

Ghaemi, S., Ragni, D., and Scarano, F. (2012). PIV-based pressure fluctuations in the turbulent boundary layer. *Experiments in Fluids*, 53(6):1823–1840.

Gresho, P. M. and Sani, R. L. (1987). On pressure boundary conditions for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1111–1145.

Gurka, R., Liberzon, A., Hefetz, D., Rubinstein, D., and Shavit, U. (1999). Computation of pressure distribution using piv velocity data. In *Workshop on particle image velocimetry*, volume 2, pages 1–6.

Hon, Y. and Schaback, R. (2001). On unsymmetric collocation by radial basis functions. *Applied Mathematics and Computation*, 119(2-3):177–186.

Hosseini, B. and Hashemi, R. (2011). Solution of burgers' equation using a local-rbf meshless method. *International Journal for Computational Methods in Engineering Science and Mechanics*, 12(1):44–58.

Kähler, C. J. and Kompenhans, J. (2000). Fundamentals of multiple plane stereo particle image velocimetry. *Experiments in Fluids*, 29(1):S070–S077.

Kansa, E. J. (1990). Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—i surface approximations and partial derivative estimates. *Computers & Mathematics with applications*, 19(8-9):127–145.

Kim, B., Lee, Y., and Choi, D.-H. (2009). Construction of the radial basis function based on a sequential sampling approach using cross-validation. *Journal of Mechanical Science and Technology - J MECH SCI TECHNOL*, 23:3357–3365.

Koschatzky, V., Overmars, E., Boersma, B., and Westerweel, J. (2012). Comparison of planar piv and tomographic piv for aeroacoustics. In *6th Int. Symp. on Applications of Laser Techniques to Fluids Mechanics, Lisbon*.

Laskari, A., de Kat, R., and Ganapathisubramani, B. (2016). Full-field pressure from snapshot and time-resolved volumetric piv. *Experiments in Fluids*, 57(3):44.

Liu, X. and Katz, J. (2006). Instantaneous pressure and material acceleration measurements using a four-exposure piv system. *Experiments in fluids*, 41(2):227–240.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Pan, Z., Whitehead, J., Thomson, S., and Truscott, T. (2016). Error propagation dynamics of PIV-based pressure field calculations: How well does the pressure poisson solver perform inherently? *Measurement Science and Technology*, 27(8):084012.

Raffel, M., Willert, C. E., Scarano, F., Kähler, C. J., Wereley, S. T., and Kompenhans, J. (2018). *Particle image velocimetry: a practical guide*. Springer.

Rao, C., Sun, H., and Liu, Y. (2020). Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212.

Rippa, S. (1999). An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2):193–210.

Scarano, F. (2012). Tomographic PIV: principles and practice. *Measurement Science and Technology*, 24(1):012001.

Van der Kindere, J., Laskari, A., Ganapathisubramani, B., and De Kat, R. (2019). Pressure from 2d snapshot piv. *Experiments in fluids*, 60(2):32.

van Oudheusden, B. W., Scarano, F., Roosenboom, E. W. M., Casimiri, E. W. F., and Souverein, L. J. (2007). Evaluation of integral forces and pressure fields from planar velocimetry data for incompressible and compressible flows. *Experiments in Fluids*, 43(2-3):153–162.

Violato, D., Moore, P., and Scarano, F. (2011). Lagrangian and eulerian pressure field evaluation of rod-airfoil flow from time-resolved tomographic piv. *Experiments in fluids*, 50(4):1057–1070.

Wang, J. and Liu, G. (2002). On the optimal shape parameters of radial basis functions used for 2-d meshless methods. *Computer Methods in Applied Mechanics and Engineering*, 191(23):2611–2630.