A Meshless Method to Compute Pressure Fields from Image Velocimetry

A Meshless Method to Compute Pressure Fields from Image Velocimetry

To cite this article before publication: Pietro Sperotto *et al* 2022 *Meas. Sci. Technol.* in press https://doi.org/10.1088/1361-6501/ac70a9

A Meshless Method to Compute Pressure Fields from Image Velocimetry

Pietro Sperotto[1,2] Sandra Pieraccini[2] Miguel A. Mendez[1]

[1]von Karman Institute for Fluid Dynamics, Sint-Genesius-Rode, Belgium
[2]Dipartimento di Ingegneria Meccanica e Aerospaziale, Politecnico di Torino, Italy

Abstract

We propose a meshless method to compute pressure fields from image velocimetry data, regardless of whether this is available on a regular grid as in cross-correlation based velocimetry or on scattered points as in tracking velocimetry. The proposed approach is based on Radial Basis Functions (RBFs) regression and relies on the solution of two constrained least square problems. The first one is the regression of the measurements to create an analytic representation of the velocity field. This regression can be constrained to impose boundary conditions (e.g. no-slip velocity on a wall or inlet conditions) or differential constraints (e.g. the solenoidal condition for an incompressible flow). The second one is the meshless integration of the pressure Poisson equation, achieved by seeking a solution in the form of a RBF expansion and using constraints to impose boundary conditions.

We first illustrate the derivation of the two least square problems and the numerical techniques implemented for their solution. Then, we showcase the method with three numerical test cases of growing complexity. These are a 2D Gaussian Vortex, a 2D flow past a cylinder from CFD and a 3D Stokes flow past a sphere. For each case, we consider randomly sampled vector fields simulating particle tracking measurements and analyze the sensitivity to noise and seeding density.

Keywords: Pressure from PIV and PTV, Radial Basis Functions, Meshless integration of PDEs.

Contents

1 Introduction

Modern image-based velocimetry can provide velocity fields with sufficient resolution to allow for computing pressure fields. This enables non-intrusive measurements of aerodynamic loading or sound fields, and considerable experimental insights into fluids dynamics.

Many methods have been developed to this end, initially mostly based on Particle Image Velocimetry (PIV) and more recently adapted and enhanced by advances in Lagrangian Particle Tracking (LPT). An extensive literature review and a comparative analysis between various methods can be found in Charonko et al. (2010); de Kat and Oudheusden (2011); Liu and Moreto (2020); McClure and Yarusevych (2017a); Pan et al. (2016); van Gent et al. (2017); van Oudheusden (2013). Broadly, pressure integration methods can be classified into Eulerian and Lagrangian, depending on how the material acceleration is computed.

Eulerian methods use the local time derivatives and velocity gradients and can be further subdivided into directional approaches integrating the pressure gradient from the Navier Stokes Equation (e.g., Jakobsen et al. (1997); Köngeter (1999); Liu and Katz (2006); Wang et al. (2019)) or global approaches integrating the Poisson equation (e.g., Ghaemi et al. (2012); Gurka et al. (1999); Pan et al. (2016)). These methods have been implemented in many variants. For example, one could include turbulence modelling via Reynolds Averaged Navier Stokes (RANS) formulations to compute averaged pressure fields (e.g., Gurka et al. (1999); van Oudheusden et al. (2007)) or leverage Taylor's frozen turbulence hypothesis to compute instantaneous pressures (de Kat and Oudheusden, 2011; der Kindere et al., 2019; Laskari et al., 2016). Within the class of Eulerian methods, more sophisticated methods include solvers based on pressure-velocity algorithms from CFD (Felis-Carrasco et al., 2021; Gunaydinoglu and Kurtulus, 2019) or immersed boundary techniques (Pirnia et al., 2020).

Lagrangian or pseudo-Lagrangian methods compute the material acceleration along the trajectories of fluid particles. Following de Kat and Oudheusden (2011), pseudo-LAgrangian (pLA) approaches use pseudo-tracking, in the sense that the particle trajectories are reconstructed from (Eulerian) velocity fields (e.g., Novara and Scarano (2013); Schneiders et al. (2016)). On the other hand, fully Lagrangian methods are based on the direct determination of particle trajectories. These have been recently enabled by advances in tracking techniques (Schanz et al., 2016), nowadays capable of providing accurate trajectories with high seeding densities. Lagrangian approaches can be further distinguished in techniques that interpolate the particle acceleration onto a Cartesian grid (Gesemann et al., 2016; Huhn et al., 2016, 2018; Schneiders and Scarano, 2016) and techniques that integrate the pressure on scattered data such as the Voronoi integration proposed by Neeteson and Rival (2015).

Once the material acceleration is computed, most of the aforementioned approaches are based on "classic" numerical techniques to integrate the relevant equations. By "classic", we here consider those methods based on Finite Differences, Finite Volumes or Finite Elements that require building a computational mesh. This is a difficult task, especially in the presence of curved boundaries and/or laser reflections, and requires advanced interpolation methods (Agarwal et al., 2021) to map the scattered data onto the computational grid.

Recent advances in data assimilation and machine learning are currently opening new perspectives for meshless methods to solve Partial Differential Equations (PDEs). Among the most notable examples, we mention the use of Artificial Neural Networks (ANN) to discover and solve PDEs (Lagaris et al., 1998; Li and Mei, 2020; Raissi et al., 2019; Sirignano and Spiliopoulos, 2018), recently popularized as *physics-informed* neural networks (see Raissi et al. (2019)).

Promoting this paradigm shift is the fact that many parametric models from the machine learning literature can be easily differentiated with respect to their inputs. Therefore, their prediction can be easily constrained to respect PDEs and related initial and boundary conditions, as well as other differential conditions (e.g. solenoid or potential fields). Thus, the problem of solving a PDE can be converted into a constrained optimization problem: given a parametric

representation of the form $\boldsymbol{y} = f(\boldsymbol{u}, \boldsymbol{w})$, linking some input field $\boldsymbol{u}$ to some output field $\boldsymbol{y}$, one seeks to identify the set of parameters $\boldsymbol{w}$ such that $\boldsymbol{y}$ solves a PDE (i.e. minimizes some properly defined residuals). The framework applies equally well to Artificial Neural Networks or to Radial Basis Functions (RBFs). Compared to the ANN, the RBF leads to least square that are considerably easier to handle because of their linearity with respect to the model parameters. It is thus not surprising that the meshless integration of PDEs via RBFs has a long history in computational engineering. The idea was introduced by Kansa (1990a,b) and has grown into a mature approach with an extensive literature (see Chen (2003); Chen and Tanaka (2002); Fornberg and Flyer (2015); Šarler (2007); Šarler (2005)). RBF-based meshless integration extend classic pseudo-spectral methods (Fornberg, 1996), in which the parametrization is usually based on Fourier or Chebyshev expansions, and can be seen as a special class of collocation methods.

In the literature of image velocimetry, RBFs have been used for their robust interpolation (Casa and Krueger, 2013), to compute derivatives (Karri et al., 2009), as regression tools for super-resolution (Ratz et al., 2022) and to support the physics-informed interpolation from scattered to uniform grids (Schneiders and Scarano, 2016), also enabling super-resolution. Nevertheless, to the author's knowledge, their implementation for the mesh-less integration of PDEs has not yet been fully exploited. This work proposes a meshless method based on RBFs to compute pressure fields from scattered (and noisy) velocity fields. The input velocity fields can be 2D or 3D and can result from PTV or PIV measurements. We focus on integrating Poisson equation in an Eulerian formalism, but we note that the proposed algorithm is essentially a tool to solve PDEs and can be generalized to a Lagrangian formalism. Similarly, the integration could be used to compute instantaneous pressure fields if time-resolved data is available (or Taylor's hypothesis invoked) or include RANS modelling for turbulence. These extensions are currently under investigation and will be presented in a dedicated contribution.

The proposed method differs from the "track-based" approaches proposed by Gesemann (2015) and Bobrov et al. (2021) in that our approach is purely Eulerian, i.e. it requires velocity information in various points but no trajectories. It it thus conceptually similar to the "second-generation" flowfit by Gesemann et al. (2016), in that it also formulates the pressure integration as a sparse least square problem. However, our method differ in the basis selection and collocation, in the cost function formulation and in definition of penalties and constraints.

The second generation flow fit Gesemann et al. (2016) uses uniformly collocated B-splines and constructs a single cost function including the accuracy of the velocity regression and the pressure integration. The result is a nonlinear regression problem. Moreover, physical constraints (e.g. boundary conditions or solenoidal condition) are included as penalties (regularizations) and not as *hard* constraints. Our approach uses scattered truncated Gaussian RBFs, collocated by a clustering algorithm, and splits the velocity regression and the pressure integration into two different problems. This produces two *linear* least square problems. Moreover, physical constraints are included both as penalties and as hard constraints, by using Lagrange multiplies.

We illustrate the method on three synthetic test cases of growing complexity, for which the background truth is available. This let us test its robustness against measurement noise and seeding concentration (sparsity) of the measured velocity field. The mathematical background of the RBF integration is described in Section 2, while Section 3 briefly introduces the selected test cases. Section 4 collects the results; conclusions and perspectives are given in Section 5.

2 Mathematical Framework

We begin by introducing Radial Basis Functions (RBFs) as tools for approximating a function $f : \mathbb{R}^d \mapsto \mathbb{R}$, focusing on the cases $d = 2$ and $d = 3$. The function approximation is built from data and can be constrained via standard tools from constrained optimization.

The general formulation is described in Section 2.1. We introduce in Section 2.2 the problem

of deriving an approximation of velocity fields using constraints (or penalties) to impose (or to promote) boundary conditions and physical priors (e.g. divergence-free). The same framework is then used in Section 2.3 for the meshless integration of the Poisson equation. Section 2.4 completes the presentation of the method illustrating the clustering algorithm used to collocate the RBFs. Finally, Section 2.5 presents our current approach to solve the large systems of equations produced in 2.2 and 2.3.

2.1 Constrained Regression via RBFs

Among the many possible RBFs (see Fornberg and Flyer (2015) for more background), we here consider Gaussians of the form

$$\varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) = \exp\left(-c_k^2 \left\|\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{k}}^*\right\|^2\right) \tag{1}$$

where $c_k > 0$ is the *shape parameter* and $\boldsymbol{x}_k^* \in \mathbb{R}^d$ is the *collocation point*. In this section, we consider both c_k and $\boldsymbol{x}_k^*$ as given: their identification is discussed in Sec. 2.4. The basis element φ_k is thus solely function of $\boldsymbol{x}$ (the symbol | separates variables from parameters) and is a *radial* function because it only depends on the distance from the collocation points. Note that we restrict the treatment to *isotropic* bases, as they depend on one shape parameter only.

The function approximation is a linear combination of n_c RBFs:

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x}) = \sum_{k=1}^{n_c} w_k \, \varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k)\,, \tag{2}$$

with the *weights* w_k to be identified from data.

In the machine learning literature, eq. (2) can be seen as a linear artificial neural network (Broomhead and Lowe, 1988; Schwenker et al., 2001) with a single hidden layer, having the RBFs in (1) as activation functions and a single output with linear activation. This parallelism opens the path towards an arsenal of stochastic optimization techniques and bridges with the recent advances in physics informed neural networks (Li and Mei, 2020; Raissi et al., 2019). However, the linearity of the model (2) with respect to the weights makes the regression problem considerably simpler than in ANN-based formulation.

Following the machine learning literature, the identification of the weights is hereinafter referred to as *training*, and the data used at the scope is referred to as *training set*. In this work, this is a set of n_p pairs $\{(\boldsymbol{x}_i, f_i)\}_{i=1,\dots,n_p}$ with $\boldsymbol{x}_i = (x_i, y_i, z_i)$ and $f_i = f(\boldsymbol{x}_i)$, produced, for example, by 3D PTV measurements. The coordinates of the sampling points can be arranged into a matrix $\boldsymbol{X} \in \mathbb{R}^{n_p \times 3}$ and the treatment that follows holds regardless of whether these points are randomly scattered as in PTV or over a grid as in PIV.

The stability of the learning is greatly enhanced if a set of $n_\gamma \ll n_c$ polynomial functions $\gamma_k(\boldsymbol{x})$ is added to the basis so that (2) becomes

$$\tilde{f}(\boldsymbol{x}) = \sum_{k=1}^{n_\gamma} w_k \gamma_k(\boldsymbol{x}) + \sum_{k=1}^{n_c} w_{n_\gamma+k} \varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k)\,. \tag{3}$$

Nevertheless, in the interest of conciseness in the notation, we treat these additional terms as elements of the same basis of $n_b = n_\gamma + n_c$ functions. Accordingly, we write the function approximation (3) on the training set as

$$\boldsymbol{f} \simeq \tilde{\boldsymbol{f}} = \boldsymbol{\Phi}(\boldsymbol{X})\, \boldsymbol{w}, \tag{4}$$

where $\boldsymbol{f} = (f_1, f_2 \dots f_{n_p})^T$ is the vector collecting all the training *targets*, $\boldsymbol{w} = (w_1, w_2 \dots w_{n_b})^T$ is the vector containing the set of weights (regardless of whether these are associated to RBFs

or polynomial terms) and we let $\boldsymbol{\Phi}(\boldsymbol{X}) \in \mathbb{R}^{n_b \times n_p}$ denote the matrix obtained by evaluating the n_b basis functions at the n_p training points:

$$\boldsymbol{\Phi}(\boldsymbol{X}) = \begin{pmatrix} \gamma_1(\boldsymbol{x}_1) & \cdots & \gamma_{n_\gamma}(\boldsymbol{x}_1) & \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_{n_c}(\boldsymbol{x}_1) \\ \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_1(\boldsymbol{x}_{n_p}) & \cdots & \gamma_{n_\gamma}(\boldsymbol{x}_{n_p}) & \varphi_1(\boldsymbol{x}_{n_p}) & \cdots & \varphi_{n_c}(\boldsymbol{x}_{n_p}) \end{pmatrix}.$$

Training the RBF model (3) corresponds to solving the linear system (4), in a least square sense, for the weights $\boldsymbol{w} \in \mathbb{R}^{n_b}$. This is a classic problem involving the minimization of a quadratic cost function:

$$J(\boldsymbol{w}) = ||\boldsymbol{f} - \boldsymbol{\Phi}(\boldsymbol{X})\,\boldsymbol{w}||_2^2\,, \tag{5}$$

where $||\cdot||_2$ denotes the l_2 norm. Since such a problem is notoriously ill-posed (Bishop, 2006), it is common practice adding a regularization, namely a penalty on the magnitude of the weights. We consider a Tikhonov (Kress, 1998) regularization of the form $\alpha||\boldsymbol{w}||_2^2$, with penalty α computed as described in Sec. 2.5. However, since this regularization is only used to ensure numerical stability, the introduction of this term is deferred to the next sections and we here focus on the general problem formulation. Moreover, we leave the treatment open to a second penalty to promote certain conditions (e.g. divergence-free) for the resulting function. We include such a condition in the form of a linear operator $\boldsymbol{C}_s\boldsymbol{w} = 0$, with $\boldsymbol{C}_s \in \mathbb{R}^{n_s \times n_b}$, acting on n_s points. Thus, an additional regularization term $\alpha_s||\boldsymbol{C}_s\boldsymbol{w}||_2^2$ is added.

Finally, we further extend the treatment to include constraints to the approximation. In particular, we consider two sets of linear equality constraints written as $\boldsymbol{C}_1\boldsymbol{w} = \boldsymbol{c}_1$ and $\boldsymbol{C}_2\boldsymbol{w} = \boldsymbol{c}_2$, with $\boldsymbol{C}_1 \in \mathbb{R}^{n_{\lambda_1} \times n_b}$ and $\boldsymbol{C}_2 \in \mathbb{R}^{n_{\lambda_2} \times n_b}$ linear operators acting on n_{λ_1} and n_{λ_2} points respectively, and $\boldsymbol{c}_1 \in \mathbb{R}^{n_{\lambda_1}}$ and $\boldsymbol{c}_2 \in \mathbb{R}^{n_{\lambda_2}}$. As detailed in the next sections, we use these constraints to impose Dirichlet, Neumann or mixed boundary conditions in the pressure integration, or to impose physical constraints to the regression of the velocity field.

In other words, the problem is formulated as

$$\min \quad J(\boldsymbol{w}) \tag{6}$$

$$\text{s.t.} \quad \boldsymbol{C}_1\boldsymbol{w} = \boldsymbol{c}_1,\, \boldsymbol{C}_2\mathbf{w} = \boldsymbol{c}_2 \tag{7}$$

The solution of problem (6) can be pursued by solving the Karush-Kuhn-Tucker optimality conditions (Chong and Zak, 2013; Nocedal and Wright, 2006). Let $J^*(\boldsymbol{w}, \boldsymbol{\lambda})$ denote the Lagrangian

$$J^*(\boldsymbol{w}, \boldsymbol{\lambda}) = ||\boldsymbol{f} - \boldsymbol{\Phi}(\boldsymbol{X})\,\boldsymbol{w}||_2^2 + \boldsymbol{\lambda}_1^T(\boldsymbol{C}_1\boldsymbol{w} - \boldsymbol{c}_1) + \boldsymbol{\lambda}_2^T(\boldsymbol{C}_2\boldsymbol{w} - \boldsymbol{c}_2) + \alpha_s||\boldsymbol{C}_s\boldsymbol{w}||_2^2\,, \tag{8}$$

where $\boldsymbol{\lambda}_1 \in \mathbb{R}^{n_{\lambda 1}}$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^{n_{\lambda_2}}$ are Lagrange multipliers associated to the linear equality constraints and $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)^T \in \mathbb{R}^{n_\lambda}$ with $n_\lambda = n_{\lambda_1} + n_{\lambda_2}$. The KKT optimality conditions for this problem are obtained canceling the gradients with respect both to weights ($\nabla_{\boldsymbol{w}} J^*$), and to the multipliers ($\nabla_{\boldsymbol{\lambda}} J^*$). This leads to the following system of equations:

$$\begin{aligned} 2\big(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \alpha_s\boldsymbol{C}_s^T\boldsymbol{C}_s\big)\boldsymbol{w} + \boldsymbol{C}_1^T\boldsymbol{\lambda}_1 + \boldsymbol{C}_2^T\boldsymbol{\lambda}_2 &= 2\boldsymbol{\Phi}^T\boldsymbol{f}_i \\ \boldsymbol{C}_1\boldsymbol{w} &= \boldsymbol{c}_1 \\ \boldsymbol{C}_2\boldsymbol{w} &= \boldsymbol{c}_2\,, \end{aligned}$$

having used the short-hand notation $\boldsymbol{\Phi}$ for $\boldsymbol{\Phi}(\boldsymbol{X})$. This system can be conveniently written as

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B^T} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}, \tag{9}$$

with $\boldsymbol{A} = 2\boldsymbol{\Phi}^T\boldsymbol{\Phi} + 2\alpha_s\boldsymbol{C}_s^T\boldsymbol{C}_s \in \mathbb{R}^{n_b\times n_b}$, $\boldsymbol{B} = \left(\boldsymbol{C}_1^T, \boldsymbol{C}_2^T\right) \in \mathbb{R}^{n_b\times n_\lambda}$, $\boldsymbol{b_1} = 2\boldsymbol{\Phi}^T\boldsymbol{f}_i \in \mathbb{R}^{n_b}$ and $\boldsymbol{b}_2 = (\boldsymbol{c}_1, \boldsymbol{c}_2)^T \in \mathbb{R}^{n_\lambda}$.

The solution of (9) leads to a minimum of $J(\boldsymbol{w})$ as long as its Hessian $\nabla^2_{\boldsymbol{w}}J(\boldsymbol{w}) = \boldsymbol{A}$ is positive semi-definite. Both the problems of velocity fields approximation and meshless pressure computation leads to constrained least square problems of the form in (9). It is worth pointing out that once the training is complete, the approximation in (4) is analytical: it can be evaluated in *any* new set of points and its derivatives are analytically available. This enables the meshless integration of PDEs.

2.2 Approximating Velocity Fields

We now set the problem of approximating velocity fields in the framework introduced in the previous section. We present the derivation for the case of a 2D field and refer the reader to Appendix A for the problem set in 3D. The dataset now consists of two scalar fields for the velocity components, i.e. $U = (u_i, v_i)$, sampled over a set of n_p points. The collocation points for the Gaussian RBFs are now on the plane $\boldsymbol{x}_k^* = (x_k^*, y_k^*)$. Eq. (2) and its matrix form in (4) can be conveniently extended to vector fields if the data is reshaped into column vectors. Namely, let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{n_p}$ be the vectors collecting all the sampled values of u_i and v_i in the grid points $\boldsymbol{x}_i$ stored in $\boldsymbol{X} \in \mathbb{R}^{n_p\times 2}$. Let $\boldsymbol{U} = (\boldsymbol{u}; \boldsymbol{v}) \in \mathbb{R}^{2n_p}$ be a column vector with the semicolon ; denoting vertical concatenation. Similarly, let $\boldsymbol{w}_U = (\boldsymbol{w_u}; \boldsymbol{w_v}) \in \mathbb{R}^{2n_b}$ be the colum vector concatenating the weights $\boldsymbol{w_u}, \boldsymbol{w_v} \in \mathbb{R}^{n_b}$ for the RBFs expansion of the velocity components u and v. Then, the approximation of the velocity field in the available points is:

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \approx \tilde{\boldsymbol{U}} = \begin{pmatrix} \tilde{\boldsymbol{u}} \\ \tilde{\boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{X}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}(\boldsymbol{X}) \end{pmatrix} \begin{pmatrix} \boldsymbol{w_u} \\ \boldsymbol{w_v} \end{pmatrix} = \boldsymbol{\Phi}_U(\boldsymbol{X})\boldsymbol{w}_U\,, \tag{10}$$

where the same matrix of basis function $\boldsymbol{\Phi}(\boldsymbol{X})$ is used for both components, assuming that these are available at the *same* points in $\boldsymbol{X}$. To ease the connection with the previous section, we introduce the basis matrix $\boldsymbol{\Phi}_U(\boldsymbol{X}) \in \mathbb{R}^{2n_p\times 2n_b}$. Note that the regression could be carried out for both fields independently, but this would prevent using constraints and penalties that links them. The general cost function is maintained as a single objective and quadratic function:

$$J(\boldsymbol{w}_U) = ||\boldsymbol{U} - \boldsymbol{\Phi}_U(\boldsymbol{X})\boldsymbol{w}_U||_2^2\,. \tag{11}$$

Concerning the penalties and constraints for this problem, besides the Tikhonov regularization, we seek to impose boundary conditions and physical constraints. Focusing on incompressible flows, the main physical constraint is that of a divergence-free flow. Other conditions can be considered as long as the corresponding operator is linear.

The divergence-free condition is set both as a constraint and as a penalty in different domain points. This choice offers a trade-off between the accuracy of the approximation and the amount of data required: large amounts of constraints increase the problem's size and require large sets of RBFs; penalties, albeit less restrictive, do not increase the problem size. To define the divergence operator, note that the derivatives of the approximated field are readily available from the derivatives of the basis functions. For the RBF, for example, one has the following partial derivatives along x (∂_x) and along y (∂_y):

$$\partial_x\varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) = -2c_k^2(x - x_k^*)\varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) \tag{12a}$$

$$\partial_y\varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) = -2c_k^2(y - y_k^*)\varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) \tag{12b}$$

Therefore, the derivatives of the approximation are:

$$\partial_x u(\boldsymbol{x}_i) \approx \sum_{k=0}^{n_c} w_{u_k}\, \partial_x\, \varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) \rightarrow \boldsymbol{\Phi}_x(\boldsymbol{X})\boldsymbol{w}_u \tag{13a}$$

$$\partial_y v(\boldsymbol{x}_i) \approx \sum_{k=0}^{n_c} w_{v_k}\, \partial_y\, \varphi_k(\boldsymbol{x}|\boldsymbol{x}_k^*, c_k) \rightarrow \boldsymbol{\Phi}_y(\boldsymbol{X})\boldsymbol{w}_v\,, \tag{13b}$$

where the matrices $\boldsymbol{\Phi}_x(\boldsymbol{X}), \boldsymbol{\Phi}_y(\boldsymbol{X}) \in \mathbb{R}^{n_p \times n_b}$ collect basis functions' derivatives along their columns:

$$\boldsymbol{\Phi}_x(\boldsymbol{X}) = [\partial_x\gamma_1, \dots \partial_x\gamma_{n_\gamma}, \partial_x\phi_1 \dots \partial_x\phi_{n_c}] \tag{14a}$$

$$\boldsymbol{\Phi}_y(\boldsymbol{X}) = [\partial_y\gamma_1, \dots \partial_y\gamma_{n_\gamma}, \partial_y\phi_1 \dots \partial_y\phi_{n_c}]\,. \tag{14b}$$

The divergence of the velocity field $\nabla{\cdot}\boldsymbol{U}(\boldsymbol{X}_\nabla)$ in a set of points $\boldsymbol{X}_\nabla \in \mathbb{R}^{n_\nabla \times 2}$ is approximated by $\boldsymbol{D}_\nabla(\boldsymbol{X}_\nabla) \in \mathbb{R}^{n_\nabla \times 2n_b}$. The divergence free condition applied to the RBF is:

$$\nabla \cdot \begin{pmatrix} \boldsymbol{u}(\boldsymbol{X}_\nabla) \\ \boldsymbol{v}(\boldsymbol{X}_\nabla) \end{pmatrix} \approx \begin{pmatrix} \boldsymbol{\Phi}_x(\boldsymbol{X}_\nabla) & \boldsymbol{\Phi}_y(\boldsymbol{X}_\nabla) \end{pmatrix} \begin{pmatrix} \boldsymbol{w_u} \\ \boldsymbol{w_v} \end{pmatrix} = \boldsymbol{D}_\nabla(\boldsymbol{X}_\nabla)\, \boldsymbol{w}_U = \boldsymbol{0}\,, \tag{15}$$

where $\boldsymbol{\Phi}_x(\boldsymbol{X}_\nabla), \boldsymbol{\Phi}_y(\boldsymbol{X}_\nabla) \in \mathbb{R}^{n_\nabla \times n_b}$ are the matrices of basis function derivatives at the points $\boldsymbol{X}_\nabla \in \mathbb{R}^{n_\nabla \times 2}$, and $\boldsymbol{0} \in \mathbb{R}^{2n_\nabla}$ is the zero vector of appropriate size.

Equation (15) is introduced in (8) as both a penalty ($\boldsymbol{C}_s$) and a constraint ($\boldsymbol{C}_1$), although generally at different points. We use $\boldsymbol{X}_\nabla \in \mathbb{R}^{n_\nabla \times 2}$ to denote matrix collecting the coordinates of the points in which the divergence-free is a constraint.

Finally, other constraints are added to impose boundary conditions. Let $\boldsymbol{X}_D \in \mathbb{R}^{n_D \times 2}$ collect the set of points over which n_D Dirichlet conditions must be enforced, and let $\boldsymbol{c}_D \in \mathbb{R}^{2n_D}$ collect all the corresponding values for both u and v components, vertically concatenated. The associated linear operator $\boldsymbol{D}(\boldsymbol{X}_D) \in \mathbb{R}^{n_D \times 2n_b}$ is:

$$\begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{X}_D) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}(\boldsymbol{X}_D) \end{pmatrix} \begin{pmatrix} \boldsymbol{w_u} \\ \boldsymbol{w_v} \end{pmatrix} = \boldsymbol{D}(\boldsymbol{X}_D)\, \boldsymbol{w}_U = \boldsymbol{c}_D\,. \tag{16}$$

Similarly, let $\boldsymbol{X}_N \in \mathbb{R}^{n_N \times 2}$ collect the set of points over which Neumann conditions must be enforced, and let $\boldsymbol{c}_N = (\boldsymbol{c}_{Nu}; \boldsymbol{c}_{Nv}) \in \mathbb{R}^{2n_N}$ collect all the corresponding values.

The associated linear operator $\boldsymbol{N}_n(\boldsymbol{X}_N) \in \mathbb{R}^{2n_N \times 2n_b}$ is:

$$\begin{pmatrix} \boldsymbol{\Phi_n}(\boldsymbol{X}_N) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}_n(\boldsymbol{X}_N) \end{pmatrix} \begin{pmatrix} \boldsymbol{w_u} \\ \boldsymbol{w_v} \end{pmatrix} = \boldsymbol{N_n}(\boldsymbol{X}_N)\, \boldsymbol{w}_U = \boldsymbol{c}_N\,, \tag{17}$$

where $\boldsymbol{\Phi_n}(\boldsymbol{X}_N) \in \mathbb{R}^{n_N \times n_b}$ is the matrix collecting the projection of the gradient along the normal to the surface for which the boundary condition must be set. Denoting as $\boldsymbol{n} = (n_x, n_y)$ the normal vector at a given location and $\boldsymbol{n}_x(\boldsymbol{X}_N), \boldsymbol{n}_y(\boldsymbol{X}_N) \in \mathbb{R}^{n_N}$ the collections of normal vectors at the points where the condition is to be imposed, we have

$$\boldsymbol{\Phi}_n(\boldsymbol{X}_N) = \boldsymbol{\Phi}_x(\boldsymbol{X}_N) \odot \boldsymbol{n}_x(\boldsymbol{X}_N) + \boldsymbol{\Phi}_y(\boldsymbol{X}_N) \odot \boldsymbol{n}_y(\boldsymbol{X}_N) \tag{18}$$

where $\odot$ denote the entry by entry (Shur) product acting along the columns.

Finally, assembling the problem in the form of (9), the full problem for the velocity approximation becomes:

$$\begin{pmatrix} \boldsymbol{A}_U & \boldsymbol{B}_U \\ \boldsymbol{B}_U^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_U \\ \boldsymbol{\lambda}_U \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{U1} \\ \boldsymbol{b}_{U2} \end{pmatrix}, \tag{19}$$

where $\boldsymbol{\lambda}_U = (\lambda_\nabla, \lambda_D, \lambda_N)^T \in \mathbb{R}^{n_\lambda}$ is the vector of the Lagrange multipliers for all constraints, i.e. $n_\lambda = n_\nabla + 2n_D + 2n_N$, and:

$$\boldsymbol{A}_U = 2\boldsymbol{\Phi}_U^T \boldsymbol{\Phi}_U + 2\alpha_\nabla \boldsymbol{D}_\nabla^T \boldsymbol{D}_\nabla \in \mathbb{R}^{2n_b \times 2n_b} \tag{20a}$$

$$\boldsymbol{B}_U = \left(\boldsymbol{D}_\nabla^T, \boldsymbol{D}^T, \boldsymbol{N}_{\boldsymbol{n}}^T\right) \in \mathbb{R}^{2n_b \times n_\lambda} \tag{20b}$$

$$\boldsymbol{b}_{U1} = 2\boldsymbol{\Phi}_U^T \boldsymbol{U} \in \mathbb{R}^{2n_b} \tag{20c}$$

$$\boldsymbol{b}_{U2} = (\boldsymbol{0}, \boldsymbol{c}_D\,, \boldsymbol{c}_N)^T \in \mathbb{R}^{n_\lambda}, \tag{20d}$$

and having used the short hand notation $\boldsymbol{\Phi}$ for $\boldsymbol{\Phi}(\boldsymbol{X})$, $\boldsymbol{N}_{\boldsymbol{n}}$ for $\boldsymbol{N}_{\boldsymbol{n}}(\boldsymbol{X}_N)$, etc. It is worth highlighting that one has $\boldsymbol{D}_\nabla(\boldsymbol{X})$ in (20a) but $\boldsymbol{D}_\nabla(\boldsymbol{X}_\nabla)$ in (20b). Moreover, Neumann-like conditions can be redundant or conflicting with the divergence-free constraints if the latter are applied on boundaries. One should not have common points between the sets represented by $\boldsymbol{X}_N$ and $\boldsymbol{X}_\nabla$. However, it is possible to have both Dirichlet and divergence-free conditions on the same points. In this case, the common n_λ points will be included in both $\boldsymbol{X}_D$ and $\boldsymbol{X}_\nabla$ and thus in the counting of both conditions (i.e to the counting n_∇ and n_D).

2.3 Computing Pressure Fields

We now consider the pressure computation by integrating the Poisson equation for incompressible and stationary flows. This equation is derived by taking the divergence of the momentum equation and reads

$$\Delta p = -\rho \nabla \cdot (\boldsymbol{U} \cdot \nabla \boldsymbol{U})\,, \tag{21}$$

where Δp is the Laplacian of the pressure field p, ρ is the fluid density, $\boldsymbol{U}$ is the velocity field and $\nabla \boldsymbol{U}$ is the velocity gradient tensor. The divergence-free assumption makes the equation independent from the viscous and unsteady terms; these nevertheless play a role at the boundary condition.

For unsteady flows, the material derivative of the velocity field is required to compute instantaneous pressure fields (entering through the Neumann boundary conditions) while Reynolds stresses could be included to compute average pressure fields from turbulent flows (der Kindere et al., 2019; Gurka et al., 1999; van Oudheusden, 2013). None of these generalization poses additional difficulties to the proposed meshless integration and hence their implementation is postponed to future work.

We here illustrate the RBF integration of (21) and its formulation in the framework of Sec. 2.1 in the case of a 2D flow. The generalization to the 3D case is reported in Appendix B. In 2D, with the velocity field denoted as $\boldsymbol{U} = (u, v)$, the right hand side of (21) reads:

$$\nabla \cdot (\boldsymbol{U} \cdot \nabla \boldsymbol{U}) = (\partial_x u)^2 + 2\partial_x v\, \partial_y u + (\partial_y v)^2\,. \tag{22}$$

Once the RBF approximation of the velocity field is computed as described in Sec.2, the evaluation of (22) on a set of $\boldsymbol{X} \in \mathbb{R}^{n_p \times 2}$ points can be approximated as

$$\nabla \cdot (\boldsymbol{U}(\boldsymbol{X}) \cdot \nabla \boldsymbol{U}(\boldsymbol{X})) \approx (\boldsymbol{\Phi}_x \boldsymbol{w}_u)^2 + \boldsymbol{\Phi}_y \boldsymbol{w}_u \odot \boldsymbol{\Phi}_x \boldsymbol{w}_v + (\boldsymbol{\Phi}_y \boldsymbol{w}_v)^2\,, \tag{23}$$

where the squares are to be computed entry by entry. We compute the source term (23) on the available n_p points and store the results into a column vector $\boldsymbol{s} \in \mathbb{R}^{n_p}$.

Writing the pressure field at every location $\boldsymbol{X}$ as in (4):

$$\boldsymbol{p} \approx \tilde{\boldsymbol{p}} = \boldsymbol{\Phi}(\boldsymbol{X}) \boldsymbol{w}_p \tag{24}$$

with $\boldsymbol{w}_p \in \mathbb{R}^{n_b}$ the weights for the pressure, the approximation of the Laplace operator is:

$$\Delta \boldsymbol{p} \approx \boldsymbol{L}(\boldsymbol{X}) \boldsymbol{w}_p\,, \tag{25}$$

where $\boldsymbol{L}(\boldsymbol{X}) = [\Delta\gamma_1(\boldsymbol{X}), \dots \Delta\gamma_{n_\gamma}(\boldsymbol{X}), \Delta\varphi_1(\boldsymbol{X}) \dots \Delta\varphi_{n_c}(\boldsymbol{X})] \in \mathbb{R}^{n_p \times n_b}$ is the matrix collecting the Laplacian of every element of the basis functions along its columns. For the Gaussian RBFs, at every point $\mathbf{x}_i \in \mathbf{X}$ this reads

$$\Delta\varphi_k(\boldsymbol{x}_i) = 4c_k^2 \left[(\boldsymbol{x}_i - \boldsymbol{x}_k)^2 - 1\right] \varphi_k(\boldsymbol{X}) \,, \tag{26}$$

and one recovers $\Delta\varphi_k(\boldsymbol{X}) \in \mathbb{R}^{n_p}$ when (26) is applied to all the set of points.

The RBF approximation of (21) is thus $\boldsymbol{L}(\boldsymbol{X})\boldsymbol{w}_p = \boldsymbol{s}$, and the cost function to minimize is:

$$J(\boldsymbol{w}_p) = ||\boldsymbol{L}(\boldsymbol{X})\boldsymbol{w}_p - \boldsymbol{s}||_2^2 \,. \tag{27}$$

Among the constraints for the pressure integration, we consider Dirichlet and Neumann conditions. Using the same notation as in the previous section, these read:

$$\boldsymbol{\Phi}(\boldsymbol{X}_D)\boldsymbol{w}_p = \boldsymbol{c}_D \tag{28a}$$
$$\boldsymbol{\Phi}_{\boldsymbol{n}}(\boldsymbol{X}_N)\boldsymbol{w}_p = \boldsymbol{c}_N \,. \tag{28b}$$

In the pressure integration problem for a steady flow, the pressure gradient required to impose Neumann conditions ($\boldsymbol{c}_N$ in (28b)) is computed from Navier-Stokes equation (Gurka et al., 1999):

$$\nabla p \cdot \boldsymbol{n} = \partial_n p = (-\boldsymbol{U} \cdot \nabla\boldsymbol{U} + \mu\Delta\boldsymbol{U} + \rho\boldsymbol{g}) \cdot \boldsymbol{n} \tag{29}$$

where $\boldsymbol{n}$ is the vector normal to the surface at hand, μ is the dynamic viscosity and $\boldsymbol{g}$ is the gravity acceleration. This equation can be easily computed from the vector field approximation and then set in form of (28). The reader is referred to Gresho and Sani (1987) for a discussion on the relevant boundary conditions for the pressure computation in an incompressibe flow.

It is worth noticing that the points used to integrate the pressure field in (25) and the points selected to impose the boundary conditions in (28) need not be the same as the ones chosen for the velocity field approximation. Nevertheless, for the first implementation proposed in this work, the same points and the same bases are used.

To conclude, the pressure integration problem in the template from Sec. 2.1 reads:

$$\begin{pmatrix} \boldsymbol{A}_P & \boldsymbol{B}_P \\ \boldsymbol{B}_P^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_p \\ \boldsymbol{\lambda}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{P1} \\ \boldsymbol{b}_{P2} \end{pmatrix} , \tag{30}$$

where $\boldsymbol{\lambda}_p = (\lambda_D, \lambda_N)^T \in \mathbb{R}^{n_\lambda}$ is the vector of the Lagrange multipliers for all constraints, i.e. $n_\lambda = n_D + n_N$ and:

$$\boldsymbol{A}_P = 2\boldsymbol{L}^T\boldsymbol{L} \in \mathbb{R}^{n_b \times n_b} \tag{31a}$$
$$\boldsymbol{B}_P = \left(\boldsymbol{D}^T, \boldsymbol{N}_{\boldsymbol{n}}^T\right) \in \mathbb{R}^{n_b \times n_\lambda} \tag{31b}$$
$$\boldsymbol{b}_{P1} = 2\boldsymbol{L}^T\boldsymbol{s} \in \mathbb{R}^{n_b} \tag{31c}$$
$$\boldsymbol{b}_{P2} = (\boldsymbol{0}, \boldsymbol{c}_D \,, \boldsymbol{c}_N)^T \mathbb{R}^{n_\lambda} . \tag{31d}$$

The notation from the previous sections is used.

2.4 Clustering to Collocate

The optimal choice of the collocation points $\boldsymbol{x}_k^*$ and shape parameters c_k has been investigated by several authors (see Franke (1982); Hardy (1971); Kansa and Carlson (1992); Rippa (1999); Sarra and Cogar (2017)) but no universally accepted method is available.

It is rather intuitive that RBFs should well cover the domain of interest and that large overlapping between basis elements hurts the conditioning of the matrix $\boldsymbol{A}$. A common practice is thus that of randomly placing the RBFs in the domain (see Kansa and Carlson (1992); Sarra and Cogar (2017)) and selecting the shape factor proportionally to the average distance to the k nearest points (see Franke (1982); Hardy (1971)). A more complex formulation, proposed by Rippa (1999) and extended by Karri et al. (2009), consists in minimizing a cost function $J(\boldsymbol{x}_k^*, c_k)$ that measures the accuracy of the RBF approximation. Although this leads to a non-linear optimization problem, the availability of analytic gradients $\partial_{\boldsymbol{x}_k} J$ and $\partial_{c_k} J$ allows for the effective implementation of powerful gradient-based optimizers. Nevertheless, this optimization requires considerable computational resources and time for large problems (e.g., in 3D tracking velocimetry).

In this work, we propose a much simpler and computationally cheaper approach. The main idea is to ensure that all RBFs are "supported" by approximately the same number of data points. In other words, defining as Ω_k the area within which a RBF is $\varphi_k(\boldsymbol{x}_i | x_k^*, c_k) > \varepsilon_t$, with ε_t a user-defined threshold, we compute x_k^* and c_k such that an average of n_K particles fall inside Ω_k. This problem can be solved using clustering techniques (Bishop, 2006; Nielsen, 2016), which we implement using the fast mini-batch version of the K-means algorithm by Sculley (2010).

Briefly, the K-means clustering aims at partitioning a set of n_p vectors $\boldsymbol{x}_i$ (here the coordinates at which the velocity vectors are available) into K clusters $\mathcal{C}_k$ with $k = 1, \dots, K$. Each cluster contains $|\mathcal{C}_k|$ particles and is such that $\sum_k |\mathcal{C}_k| = n_p$ and $\mathcal{C}_i \cap \mathcal{C}_j = 0 \, \forall i \neq j$. Each cluster identify vectors with certain degree of similarity, defined in terms of some distance metrics. In this work, we consider the Euclidean distance. Let $\boldsymbol{\mu}_k$ denote the centroid of cluster $\mathcal{C}_k$, the clustering problem aims at minimizing the intra-cluster variance

$$J(\boldsymbol{\mu}_1, \dots \boldsymbol{\mu}_N) = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2 \quad \text{with} \quad \boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{j \in \mathcal{C}_k} \boldsymbol{x}_j \, . \tag{32}$$

The classic iterative algorithm by Lloyd (1982) solves this minimization starting with a random set of cluster centroids. At each iteration, the cluster assignment is followed by a new computation of centroids until convergence. Because the function (32) is usually not convex, the algorithm is repeated a number of times, and various initialization techniques have been proposed to escape local minima (see Solis-Oba (2006)). The mini-batch version of the K-means algorithm implemented in this work uses stochastic gradient descent and allows for considerable saving in the computational cost (Sculley, 2010).

The proposed collocation technique is essentially an agglomerative clustering approach (Nielsen, 2016) using the K-means algorithm iteratively and at multiple levels. The procedure is illustrated in Figure 1 for the RBF collocation in $n_p = 100$ points using two levels. At the first level (figure on the left), we set $n_k = 10$, resulting in $K = 10$ clusters. The centroids are shown with square markers. At the second level (figure on the right), the clustering is repeated on the centroids of the previous level. If one expect approximately $n_k = 3$ centroids per cluster, then $K = \text{floor}(100/(3 \cdot 10)) = 3$ clusters are used (with floor the rounding down operator). The centroids belonging to the second cluster are shown with blue diamond markers.

In general, denoting as n_l the number of levels and with K_j the number of clusters at the j-th level, with $j = 1, \dots, n_l$, one has

$$K_j = \frac{n_p}{\prod_{n=1}^{j} n_K^{(n)}} \, , \tag{33}$$

where $n_K^{(n)}$ is the expected number of points per cluster in the n-th level. In a compact notation, we use the vector $\boldsymbol{n}_K = [n_K^{(1)}, \dots, n_K^{(n_l)}] \in \mathbb{R}^{n_l}$.

All the centroids are taken as collocation points regardless of their level, i.e. $\boldsymbol{x}_k^* = \boldsymbol{\mu}_k$. The shape factors are computed at each level in such a way that $\varphi_k(\boldsymbol{\mu}_i^{(j)} | \boldsymbol{\mu}_k^{(j)}, c_k) = \varepsilon_t$, where $\boldsymbol{\mu}_k^{(j)}$ is

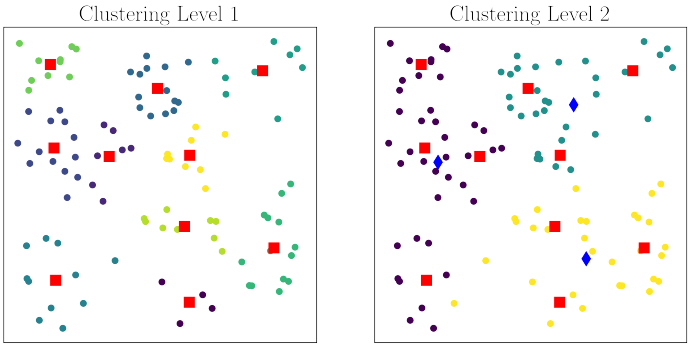


Figure 1: Illustration of the proposed hierarchical clustering to collocate RBFs and to compute their shape parameter. The ilustration takes $n_p = 100$ particles with $\boldsymbol{n}_K = [10, 30]$. The red square markers are the centroids produced in the first level; the blue diamond markers are the ones at the second level. In both figures, the particles are coloured based on the cluster they belong to.

the collocation point of interest at level j and $\boldsymbol{\mu}_i^{(j)}$ is the nearest collocation point at the same level. Therefore, one has $c_k^{(j)} = \sqrt{-\ln(\varepsilon_t)}/\min\left(|\boldsymbol{\mu}_i^{(j)} - \boldsymbol{\mu}_k^{(j)}|\right)$ with $k \neq j$.

Figure 2 illustrates the result of the clustering for the second test case analyzed in this work (see Sec. 3.3). The figure on the top shows the result of the clustering considering $n_p = 18755$ points and $\boldsymbol{n}_K = [4, 8]$, i.e. having two sets with $K_1 = n_p/8 \approx 2417$ and $K_2 = n_p/(8 \cdot 4) \approx 604$ clusters. The top figures show 300 randomly selected RBFs, with the green circles denoting the regions Ω_k such that $\varphi_k > 0.85$. The bottom figure shows a closed view near the cylinder, together with the region Ω_k of about 30 randomly chosen radial basis functions. The large variety of shape factors is evident.

Some conservative conditions are added to avoid overly large c_k (resulting in too small RBFs), as these make the problem ill-posed (Cheng et al., 2003). Firstly, if a cluster contains only one point, its shape factor is set to the smallest of the same level, i.e. $min(c_k^{(j)})$. The same limiting factor is used if the points within a RBF are less than $n_K^{(j)}$. Secondly, an upper limit is set to the value of c_k. This is provided by the user, and all shape factors above this value are set equal to this value. Finally, because constraints remove degrees of freedom to the regression problem, a basis function is given to each constraint (i.e. every constraint point is also a collocation point). The same criteria limiting the shape factors are used for these basis elements, but computing the c_k from the nearest neighbour regardless of the clustering level. Figure 2 illustrates several basis functions located on the cylinder walls.

2.5 Numerical Methods

The previous sections have shown that both the velocity regression and the pressure integration require solving a linear system of the form in (9). This kind of system arises in quadratic problems with linear equality constraints, and it is known as the Karush-Kuhn-Tucker (KKT) system. The reader is referred to Nocedal and Wright (2006) for a vast literature on both direct and iterative solution methods.

It is possible to show that the KKT system admits a unique solution if $\boldsymbol{B} \in \mathbb{R}^{n_b \times n_\lambda}$ is full rank and the reduced Hessian $\boldsymbol{Z}^T \boldsymbol{A} \boldsymbol{Z}$, with $\boldsymbol{Z} \in \mathbb{R}^{n_b \times (n_b - n_\lambda)}$ a basis for the nullspace of $\boldsymbol{B}^T$ (i.e. $\boldsymbol{B}^T \boldsymbol{Z} = \boldsymbol{0}$), is positive definite. This condition is nevertheless hardly met in practice, notably because it is difficult to avoid some redundancy in the constraints. A natural solution is thus the use of SVD or a rank revealing QR factorization of $\boldsymbol{B}$ to remove redundant constraints, but this is generally expensive in 3D problems where $\boldsymbol{B}$ is large.

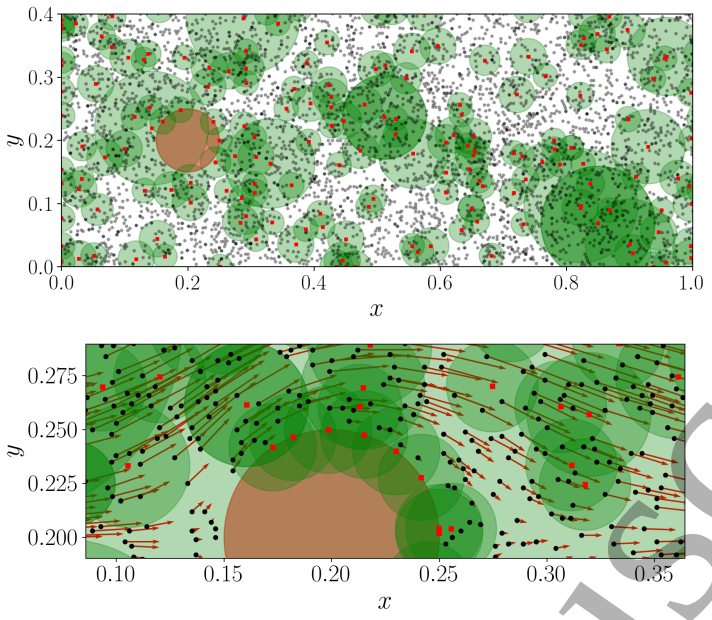


Figure 2: Test case 2. Results from the clustering technique for collocation and shape factor selection; the cylinder is identified by red color. Top: 300 randomly selected collocation points (red square markers) together with their region Ω_k such that $\varphi_k > 0.85$. The black markers identify the seeding particles. Bottom: a zoomed view with quiver plot of the available data for the velocity field.

Therefore, the proposed approach relies on a direct heuristic method that has no guarantees of uniqueness but has so far provided the best compromise between solution accuracy, robustness, and memory requirements. Assuming that the regularization parameter ensures the positive definiteness of $\boldsymbol{A}$, from the first equation in (9) one gets

$$\boldsymbol{w} = \boldsymbol{A}^{-1}(\boldsymbol{b}_1 - \boldsymbol{B}\boldsymbol{\lambda})\,. \tag{34}$$

Inserting this in the second equation, rearranging terms and defining $\boldsymbol{M} = \boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}$, $\boldsymbol{b}_2^* = \boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{b}_1 - \boldsymbol{b}_2$ and $\boldsymbol{b}_1^* = \boldsymbol{b}_1 - \boldsymbol{B}\boldsymbol{\lambda}$, the linear system in (9) is decoupled in the two lower-dimensional systems

$$\boldsymbol{M}\boldsymbol{\lambda} = \boldsymbol{b}_2^* \tag{35a}$$
$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}_1^* \tag{35b}$$

to be solved sequentially. The solution method now hinges on the symmetry and the positive definitiveness of the matrices $\boldsymbol{A} \in \mathbb{R}^{n_b \times n_b}$ and $\boldsymbol{M} \in \mathbb{R}^{n_\lambda \times n_\lambda}$, ensured by an appropriate regularization.

Let α_A and α_M denote the regularization parameters for $\boldsymbol{A}$ and $\boldsymbol{M}$, such that the regularization of these matrices reads $\boldsymbol{A} \leftarrow \boldsymbol{A} + \alpha_A\boldsymbol{I}$ and $\boldsymbol{M} \leftarrow \boldsymbol{M} + \alpha_M\boldsymbol{I}$, with $\boldsymbol{I}$ the identity matrix of appropriate size. Ideally, one would set these parameters as the smallest positive values guaranteeing positive definiteness of the corresponding matrix. However, since estimating this parameters would be a too expensive task for large datasets, we rely on a cheap estimation of the condition number using the infinity norm. For the matrix $\boldsymbol{A}$, for instance, letting $\kappa(\boldsymbol{A}, 2)$ denote the spectral condition number, $\lambda_m(\boldsymbol{A})$ the smallest eigenvalue of $\boldsymbol{A}$, $\lambda_M(\boldsymbol{A})$ is the largest eigenvalue of $\boldsymbol{A}$, and assuming $\alpha_A \approx \lambda_m(\boldsymbol{A})$, one has (see, for example Rannacher (2018))

$$\kappa(\boldsymbol{A},2)=\frac{\lambda_M(\boldsymbol{A})}{\lambda_m(\boldsymbol{A})}\approx\frac{\lambda_M(\boldsymbol{A})}{\alpha_A}\leq\frac{\sqrt{n_b}||\boldsymbol{A}||_\infty}{\alpha_A}\rightarrow\alpha_A\lesssim\frac{\sqrt{n_b}||\boldsymbol{A}||_\infty}{\kappa(\boldsymbol{A},2)}=\mathrm{tol}\sqrt{n_b}||\boldsymbol{A}||_\infty \tag{36}$$

where tol is a user defined estimate of the tolerated $1/\kappa(\boldsymbol{A},2)$ (usually $\mathrm{tol}=10^{-12}$ when working in float64 and $\mathrm{tol}=10^{-5}$ when working in float32). The same procedure is used for estimating α_M to regularize the matrix $\boldsymbol{M}$ in (35).

The successful regularization enables solving both systems in (35) using the Cholesky decomposition. Let $\boldsymbol{A}=\boldsymbol{L}_A\boldsymbol{L}_A^T$ be the Cholesky decomposition of $\boldsymbol{A}$. Then, letting $\boldsymbol{R}$ denote the product $\boldsymbol{R}=\boldsymbol{L}_A^{-1}\boldsymbol{B}$ we have

$$\boldsymbol{M}=\boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}=\boldsymbol{B}^T(\boldsymbol{L}_A^{-1})^T\boldsymbol{L}_A^{-1}\boldsymbol{B}=\boldsymbol{R}^T\boldsymbol{R}, \tag{37}$$

We remark that the inversion of the lower triangular matrix $\boldsymbol{L}_A$ is only formal as $\boldsymbol{R}$ is computed by solving triangular systems, each one requiring $\simeq\frac{1}{2}n_b^2$ operations.

While the Cholesky decomposition in (37) is the most expensive operation (with an operation count of $1/6n_b^3$) of the numerical approach, the computed factors serve three purposes. Indeed, besides allowing for the rapid computation of $\boldsymbol{M}=\boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}$ following equation (37), it can also be exploited in the computation of the right-hand-side $\boldsymbol{b}_2^*$:

$$\boldsymbol{b}_2^*=\boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{b}_1-\boldsymbol{b}_2=\boldsymbol{B}^T(\boldsymbol{L}_A^{-1})^T\boldsymbol{L}_A^{-1}\boldsymbol{b}_1-\boldsymbol{b}_2=\boldsymbol{R}^T\boldsymbol{L}_A^{-1}\boldsymbol{b}_1-\boldsymbol{b}_2 \tag{38}$$

thus only requiring one triangular solve and a matrix-vector multiplication. Furthermore, once the solution of the system $\boldsymbol{M}\boldsymbol{\lambda}=\boldsymbol{b}_2^*$ is found, the factors allow to compute the weights in (34) cheaply, by solving again two triangular systems. The complete algorithm is summarized in the listing 1.

Algorithm 1 Direct Method for the systems in Eq. (35)

1: Assembly $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ as in (9).
2: Compute $||\boldsymbol{A}||_\infty$ and α_A from (36). Then regularize $\boldsymbol{A}\leftarrow\boldsymbol{A}+\alpha\boldsymbol{I}$
3: Compute the Cholesky factorization $\boldsymbol{A}=\boldsymbol{L}_A\boldsymbol{L}_A^T$
4: Compute $\boldsymbol{R}$ solving a triangular system for each column of $\boldsymbol{B}$
5: Compute $\boldsymbol{M}=\boldsymbol{R}^T\boldsymbol{R}$, $||\boldsymbol{M}||_\infty$ and α_M as in (36). Then regularize $\boldsymbol{M}\leftarrow\boldsymbol{M}+\alpha_M\boldsymbol{I}$
6: Compute the Cholesky decomposition $\boldsymbol{M}=\boldsymbol{L}_M\boldsymbol{L}_M^T$
7: Compute the r.h.s $\boldsymbol{b}_2^*$ as in (38)
8: Solve two triangular systems for $\boldsymbol{\lambda}=(\boldsymbol{L}_M^{-1})^T\boldsymbol{L}_M^{-1}\boldsymbol{b}_2^*$
9: Solve two triangular systems for $\boldsymbol{w}=(\boldsymbol{L}_A^{-1})^T\boldsymbol{L}_A^{-1}\big(\boldsymbol{b_1}-\boldsymbol{B}\boldsymbol{\lambda}\big)$

3 Selected Test Cases

3.1 Test case 1: a Gaussian Vortex in 2D

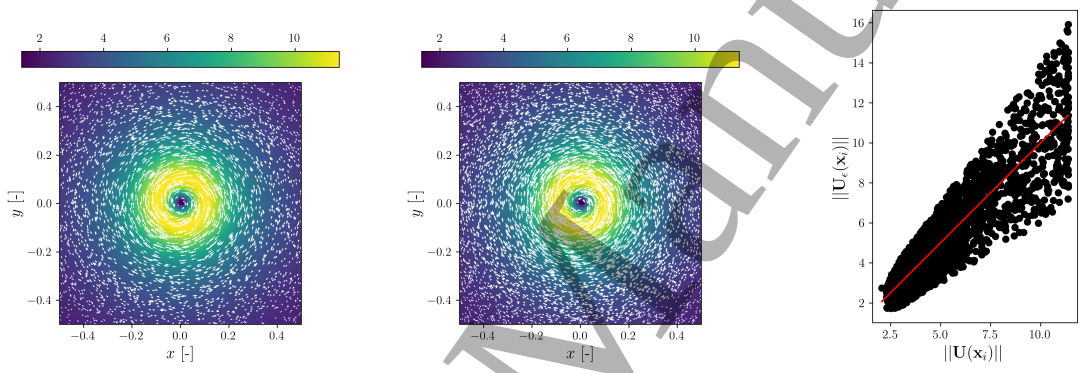
As a first test case we consider a Gaussian vortex field. This is a classic benchmark problem (see Azijli et al. (2016); de Kat and Oudheusden (2011); McClure and Yarusevych (2017a)), characterized by smooth gradients in both velocity and pressure fields. The conditions are those in de Kat and Oudheusden (2011), with a velocity field in polar coordinate $U=(u_r,u_\theta)$:

$$u_r=0\quad\text{and}\quad u_\theta=\frac{\Gamma}{2\pi r}\left(1-e^{-r^2/c_\theta}\right), \tag{39}$$

where Γ is the circulation, $c_\theta=r_c^2/\gamma$, with r_c the radial distance from the vortex center where the largest velocity is reached, and $\gamma=1.25643$. This is a Lamb-Oseen vortex with $4\nu t=c_\theta$.

The domain of interest is $\boldsymbol{x} \in [-0.5, 0.5] \times [-0.5, 0.5]$ and is taken as dimensionless. Assuming that this is mapped onto a sensor with 1024×1024 pixels allows for analyzing the impact of the seeding concentration in terms of the familiar source density N_p (in particles per pixel, ppp). In this test case, we consider a seeding concentration in the range $N_p = [0.003, 0.005]$, corresponding to a number of particles in the range $n_p = [3145, 5242]$. Particles are randomly distributed within the domain, sampled with a uniform distribution, and their velocity is taken according to (39). Figures 3(a) and 3(b) show the cases with the smallest and the largest seeding density. The contour map in the background is the velocity magnitude (dimensionless).

In addition to the impact of the seeding density, we consider the impact of random noise. This is introduced as $\boldsymbol{U}_\varepsilon(\boldsymbol{x}_i) = \boldsymbol{U}(\boldsymbol{x}_i)(1 + q\,\boldsymbol{w}(\boldsymbol{x}_i))$, where $\boldsymbol{U}(\boldsymbol{x}_i)$ is the ideal velocity field from (39), q is the noise level and $\boldsymbol{w}(\boldsymbol{x}_i) \in [-1, 1] \times [-1, 1]$ is a bi-dimensional field of uniformly distributed noise. For this specific test case, characterized by a smooth velocity field and no solid surfaces, the algorithm proved capable of handling errors as large as $q = 0.4$. Figure 3(c) plots the distribution of noisy fields versus the expected field, showing the large impact of the noise for a case with $N_p = 0.005$ and $q = 0.4$. This noise level is clearly unrealistic, but it allows to showcase the robustness of the approach for problems with smooth velocity and pressure fields. The results are presented in section 4.1.



(a) (b) (c)

Figure 3: Test case 1. Velocity fields for the lowest (a) and the highest (b) seeding densities used to test the pressure integration. The number of particles is $n_p = 3145$ and $n_p = 5242$, respectively; velocity is randomly sampled from (39). Figure (c): for the highest density case, velocity distributions of the noisy field versus the original field, with the highest noise level tested ($p = 0.4$).

The Navier-Stokes equations in polar coordinate give $\partial_r p = \rho / r u_\theta^2$ and $\partial_\theta p = 0$. The integration of the pressure gradient from $-\infty$ to r gives

$$p(r) = -\frac{1}{2}\rho u_\theta^2 - \frac{\rho \Gamma^2}{4\pi^2 c_\theta}\left[E_1\left(\frac{r^2}{c_\theta}\right) - E_1\left(\frac{2r^2}{c_\theta}\right)\right], \tag{40}$$

where $E_1(x)$ is the exponential integral

$$E_1(x) = \int_x^\infty \frac{e^{-t}}{t} dt\,. \tag{41}$$

The pressure field is shown in Figure 4 over the $n_p = 5242$ scattered points which corresponds to $N_p = 0.005$, i.e. the highest seeding density considered for this example. The contour of the pressure field is made visible by coloring the markers, located at the particle positions, with the pressure values in equation (4). Most of the particles are in a region of nearly uniform pressure and most of the pressure variation is located at approximately $r < 0.2$.

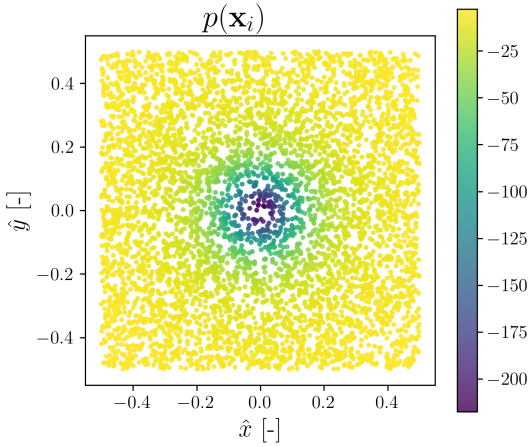


Figure 4: Test case 1. Pressure field from (40), made visible by coloring the markers of a scatter plot. The seeding density in the plot corresponds to $N_p = 0.005$, i.e. $n_p = 5242$ particles.

3.2 Test case 2: a 2D Flow Past a Cylinder from CFD

The second selected test case is a 2D problem featuring curved walls and demanding for boundary conditions in both the velocity regression and the pressure integration. This is the laminar and incompressible flow past a cylinder in a 2D channel. The flow configuration and the relevant dimensions and boundaries are shown in Figure 5. Figure 6 shows a snapshot for the velocity field, with a zoomed view around the cylinder wall, while Figure 7 shows the associated pressure field.

This is a classic benchmark test case (see John (2002); Schäfer et al. (1996)) presented in various tutorials (e.g., Langtangen and Logg (2017)). Nevertheless, in this work we use the dataset released by Rao et al. (2020). This dataset was used to train a mixed-variable scheme for physics informed neural networks (PINNS), a valid alternative to the meshless approach proposed in this work and to which we compare our regression results.

We here recall the main parameters for this dataset and refer the reader to the original publication for more details. The velocity profile on the inlet is set as

$$u(0, y) = 4\frac{U_M}{H^2}\Big(H - y\Big)y \tag{42}$$

with $U_M = 1\text{m/s}$ and $H = 0.41\text{m}$. The fluid density is taken as $\rho = 1\text{kg/m}^3$ and the dynamic viscosity is taken as $\mu = 2 \cdot 10^{-2}\text{kg/m}^3$. The velocity vector is set to zero at all walls and the pressure is set to $p = 0$ at the outlet.

3.3 Test case 3: the 3D Stokes Flow past a Sphere

The third selected test case is the classic 3D axisymmetric Stokes flow with uniform free stream velocity U_∞ past a sphere of radius R. In spherical coordinates ($\boldsymbol{r} = (r, \theta, \varphi)$), the relevant components of the velocity field ($\boldsymbol{v} = (v_r, v_\theta, 0)$) reads (see for example Bird et al. (2006))

$$v_r = U_\infty\left[1 - \frac{3}{2}\left(\frac{R}{r}\right) + \frac{1}{2}\left(\frac{R}{r}\right)^3\right]\cos(\theta) \tag{43a}$$

$$v_\theta = U_\infty\left[1 - \frac{3}{4}\left(\frac{R}{r}\right) - \frac{1}{4}\left(\frac{R}{r}\right)^3\right]\sin(\theta)\ , \tag{43b}$$

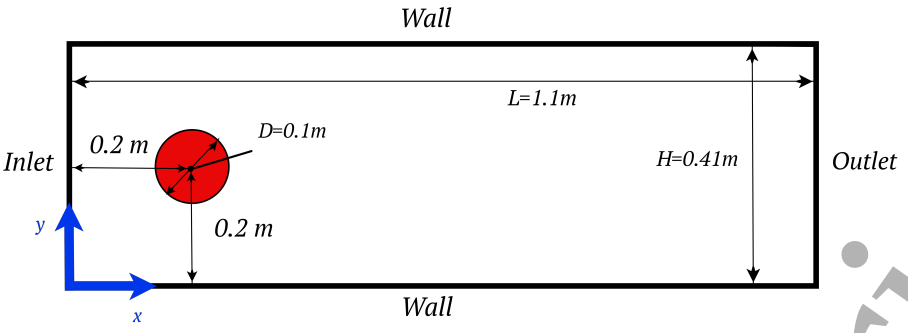


Figure 5: Test case 2. Inlet Schematic, recalling the relevant dimensions and boundary conditions. The dataset is taken from Rao et al. (2020), who share it at https://github.com/Raocp/PINN-laminar-flow.

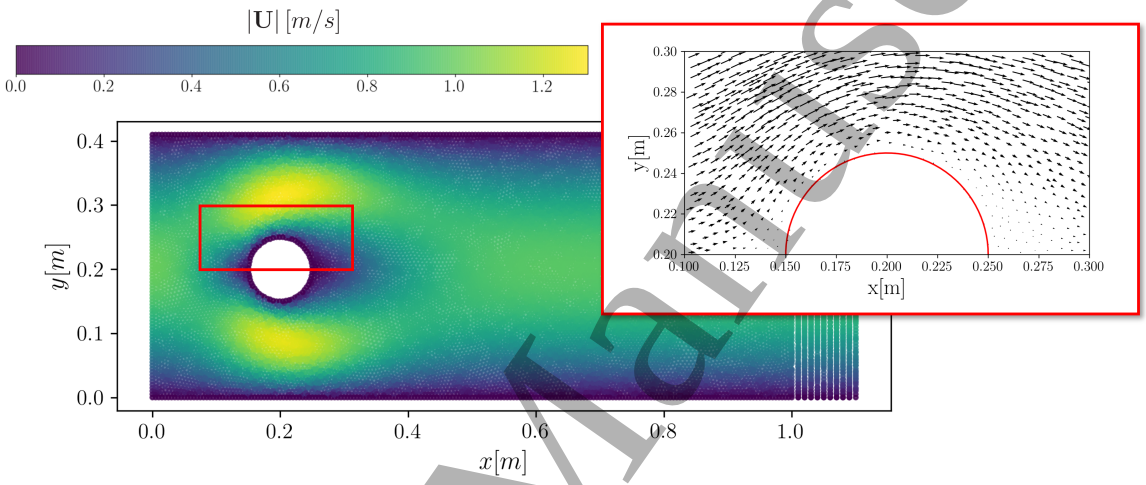


Figure 6: Test case 2. Overview of the flow field with $n_p = 18755$. The velocity magnitude is plotted in all the sampling location in the form of a scatter plot; the high particle density makes the plot appear continuous. The zoom shows a quiver plot around the cylinder.

where the free stream flows along the z direction, θ is the polar coordinate in the (x, y) plane, and φ is the polar coordinate in the (y, z) plane, hence the symmetry of the flow is such that all derivatives along φ vanish. The pressure field, taking $p_\infty = 0$ as reference in the far field, is

$$p(r, \theta) = -\frac{3}{2}\left(\frac{\mu U_\infty}{R}\right)\left(\frac{R}{r}\right)^2 \cos(\theta)\,. \tag{44}$$

Both the velocity and the pressure fields are here scaled to their dimensionless counterpart $\hat{p} = pD/(\mu U_\infty)$ and $\hat{\mathbf{v}} = (v_r/U_\infty, v_\theta/U_\infty)$ and the radial coordinate becomes $\hat{r} = r/D$. Therefore, the dimensionless stagnation pressure becomes $\hat{p}(0.5, 0) = 3$.

Figure 8(a) shows the contour-plot of the dimensionless pressure field $\hat{p}$ (on the half side) and the dimensionless velocity magnitude $\hat{\mathbf{v}}$ (on the right side) on the (y, z) plane. Both fields are shown in a domain with $\hat{r} \in [0.5, 1]$. Within the same domain, Figure 8(b) shows a quiver plot of the velocity field using $n_p = 450$ particles within $\varphi \in [0, \pi/2]$ and $\theta \in [-\pi, \pi]$.

This test case was analyzed with a number of particles ranging from $n_p = 10000$ to $n_p = 20000$. This is thus a relatively small 3D problem if compared to the experiments performed by Huhn et al. (2018), but in line with other authors working on the regression of 3D Lagrangian tracking on regular grids (see for examples Agarwal et al. (2021)). Recalling that a 3D problem leads to matrices of the order of $\sim 3n_b \times 3n_b$, it is easy to see that the memory demands grow

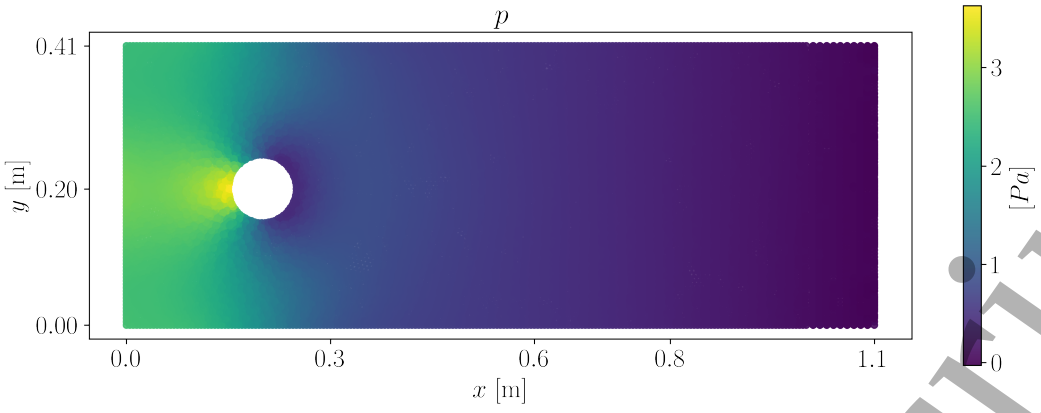


Figure 7: Test case 2. Overview of the pressure field with $n_p = 18755$.

considerably even if the proposed clustering algorithm is such that $n_b \approx n_p/2$.

A particle count of the order of $n_p \approx 20000$ was considered appropriate to the scope of this work, namely providing a first proof of concept to the constrained RBF for regression and pressure computation on a 3D problem of realistic size. Accordingly, we present a test case with $n_p = 18300$, taking the volume spherical shell within $\hat{r} = [0.5, 1]$ as the integration volume. The choice of such a small volume makes the pressure integration particularly challenging because the boundary conditions at the open boundary ($\hat{r} = 1$) requires an accurate evaluation of all the gradients of the velocity field. A larger domain or a much less 'viscous' flow would allow the velocity field to become almost potential sufficiently far from the wall and would considerably simply the setting of the boundary conditions. The results for the velocity regression and the pressure integration for this test case are presented in section 4.3.

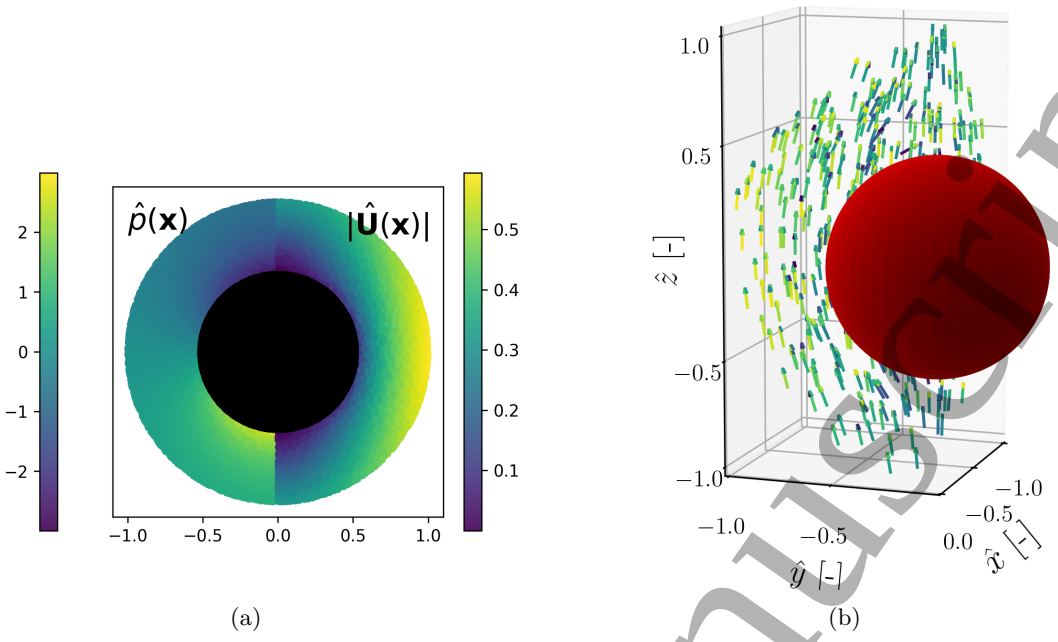
4 Results

The three analyzed test cases represent problems of vastly different scales in terms of computational cost and complexity of the associated least-square problems. The first test case needs only a few constraints. The second has curved walls requiring more constraints and the third, being 3D, leads to a fairly large least square problem. Table 1 recalls the main parameters controlling the size of the problem in terms of number of particles, number of constraints and number of employed RBFs. Because these test cases were analyzed with various conditions in terms of seeding density and clustering approach, these numbers refer to one condition per case, namely the one considered the most representative. The implementation details and the results for each case are discussed in the following subsections.

	N. of particles (n_p)	Size of the basis (n_b)	N of Constraints (n_λ)	
	V/P	V/P	V	P
Case 1: Vortex Flow	5242	1158	196	196
Case 2: 2D Cylinder Wake	18755	4203	1348	748
Case 3: 3D Stokes Flow	18297	10381	13323	6990

Table 1: Size of the test cases analyzed in this work in terms of number of particles n_p, number of RBFs n_b and constraints n_λ. The letter V denotes the velocity regression problem, the letter P denotes the pressure integration problem.

All computations were performed on a modest desktop computer with 32 GB of installed RAM, running with an Intel(R) Core(TM)i7-3770 CPU (3.GHz). The implementations were coded in Python 3.7. It is important to note that the current implementation is not yet fully



(a) (b)

Figure 8: Test case 3. (a) Contour-plot of the dimensionless pressure (left) and dimensionless velocity magnitude (right) on the (y, z) plane. (b) View of the flow seeded with $n_p = 450$ particles in $\varphi \in [0, \pi/2]$ and $\theta \in [-\pi, \pi]$.

optimized, and significant performance gains are possible using low-level programming languages or code parallelization. We nevertheless share the timing results in table 2 for the critical operations from Algorithm 1. These numbers are given only to have an order of magnitude of the computational costs involved in the velocity regression and pressure computation. While the main computational concern of the proposed method in its current form is the memory demand for large scale problems, it is interesting to note that the computing time for a 2D problem like the one in case 2 is of the order of 9 seconds for the velocity regression and 2.2 seconds for the pressure computation while for the most expensive 3D test case considered in this work (with $n_p = 18297$ particles with $n_b = 10381$ RBFs in 3D) the timing goes to approximately 15 minutes for the velocity regression and 2 minutes for the pressure integration. The same 3D case (including both velocity regression and pressure computation) runs in about 4 minutes on a single AMD EPYC 7742 64-core processors and 528 GB of RAM.

4.1 Test Case 1: Gaussian Vortex

A total of $n_\lambda = 196$ constraints is chosen for both the regression of the velocity fields and the pressure integration. These constraints (50 on each side) are equally spaced along the boundaries of the square domain. On these points, the divergence-free conditions is imposed as a constraints of the velocity regression. Moreover the implementation find automatically the repeated point to ease the input procedure. We thus have $n_\Delta = 196$, and $n_D = n_N = 0$ in (20). A divergence-free penalty is set in the whole domain with $\alpha_\nabla = 1$.

The same boundary points are used to impose Neumann boundary conditions for the pressure integration (cf equation (29)). Moreover, a Dirichlet boundary condition is set in one point, on the top left corner. This simulates a local pressure probe and enables a unique solution to the Poisson equation. We thus have $n_N = 200$ and $n_D = 1$ in (31). The cluster-level vector is

	Time for Clustering	Time to Prepare $\mathbf{A}$		Timing to prepare $\mathbf{M}$ (lines 4 and 5)		Timing to compute $\boldsymbol{\lambda}$ (line 7)		Timing to compute $\boldsymbol{w}$ (Line 8)	
	V/P	V	P	V	P	V	P	V	P
Case 1	2.2	0.86	0.66	0.24	0.04	0.002	0.001	0.01	0.002
Case 2	4.3	2.61	1.94	2.19	0.27	0.16	0.028	0.035	0.012
Case 3	12.9	320	90.3	633	42.8	16.7	3.6	10.9	0.078

Table 2: Timing (in seconds) for some of the key steps in the algorithm (cf. Algorithm 1) for the three test cases considered in this work. The lines involved in the timing are recalled in the relevant columns. As for Table 1, V denotes the velocity regression and P denotes the pressure integration.

$\boldsymbol{n}_M = [6, 10]$ with $\varepsilon_t = 0.88$ and the maximum capping taken as $c_k = 0.7$ (see Sec. 2.4).

To explore the impact of seeding concentration and noise, a grid with $N_p = [0.003, 0.005]$ and $q = [0, 0.3]$ (see 3.1 for definitions) with 20×20 elements was analyzed. Figure 10 shows the l_2 error for the velocity (E_U) and the pressure (E_P) computations, defined as

$$E_U = \frac{||\boldsymbol{u}(\boldsymbol{X}) - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}_u||_2 + ||\boldsymbol{v}(\boldsymbol{X}) - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}_v||_2}{||\boldsymbol{u}(\boldsymbol{X})||_2 + ||\boldsymbol{v}(\boldsymbol{X})||_2} \quad \text{and} \quad E_P = \frac{||\boldsymbol{p}(\boldsymbol{X}) - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}_p||_2}{||\boldsymbol{p}(\boldsymbol{X})||_2} \tag{45}$$

where $||\cdot||_2$ is the l_2 norm of a vector, $(\boldsymbol{u}(\boldsymbol{X}), \boldsymbol{v}(\boldsymbol{X}))$ is the theoretical velocity field and $\boldsymbol{p}(\boldsymbol{X})$ is the theoretical pressure field on the scattered points $\boldsymbol{X}$, arranged as column vectors.

It is worth noticing that the error is computed with respect to the theoretical (noiseless) data, thus also evaluating the RBF expansion's filtering capabilities.

Overall, the (global) l_2 error is below 2% for both velocity and pressure reconstruction even at noise levels as high as $q = 30\%$ and seeding densities as low as $N_p = 0.003$ (corresponding to $n_p = 3150$ particles). Moreover, as long as a sufficient number of particle is available, the accuracy of the velocity reconstruction is found to be independent from the seeding concentration and approximately linearly dependent on the noise level.

Figure 9 illustrates the robustness of the method in the gradient reconstruction. Figure 9(a) shows the pressure gradient along the x direction by differentiating (40); Figure 9(b) shows the same quantity computed analytically from the RBF reconstruction. This test case considers $N_{pp} = 0.004$ with a noise level of $q = 25\%$. The agreement is remarkable, as also shown in Figure 9(c), which shows the profile of the pressure gradient from the analytic formula (blue dash-dotted line), from the RBF computation (continuous black line) and the projection of the NS equation evaluated via the RBF expansion (red dashed lines). The curves are nearly indistinguishable. Compared to the results presented in the literature for this test case (see e.g. McClure and Yarusevych (2017c), who showed results with comparable noise levels), the results appears much smoother and more accurate. This appears remarkable if one consider that this test case is usually solved with at least one edge of the domain implementing Dirichlet boundary conditions (see de Kat and Ganapathisubramani (2012), while McClure and Yarusevych (2017c) uses Dirichlet boundary conditions on the four sides).

To conclude, considering an intermediate test case with $N_{pp} = 0.004$ and $q = 0.15$, Figure 11(a) shows the distribution of absolute error for the velocity field $(\boldsymbol{u}_i - \boldsymbol{\Phi}(\mathbf{x}_i)\boldsymbol{w}_u)^2 + (\boldsymbol{v}_i - \boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{w}_v)^2$ while Figure 11(b) shows the distribution of absolute error for the pressure field $\boldsymbol{p}_i - \boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{w}_p$ (on the right). The error is mostly present in the region of high shear, but it is overall negligible for both the velocity regression and the pressure computation.

4.2 Test Case 2: 2D Flow Past a Cylinder

This test case involves all the set of boundary conditions implemented in this work. In each of the solid walls, 150 (equally spaced) constraints are taken for both the regression of the velocity

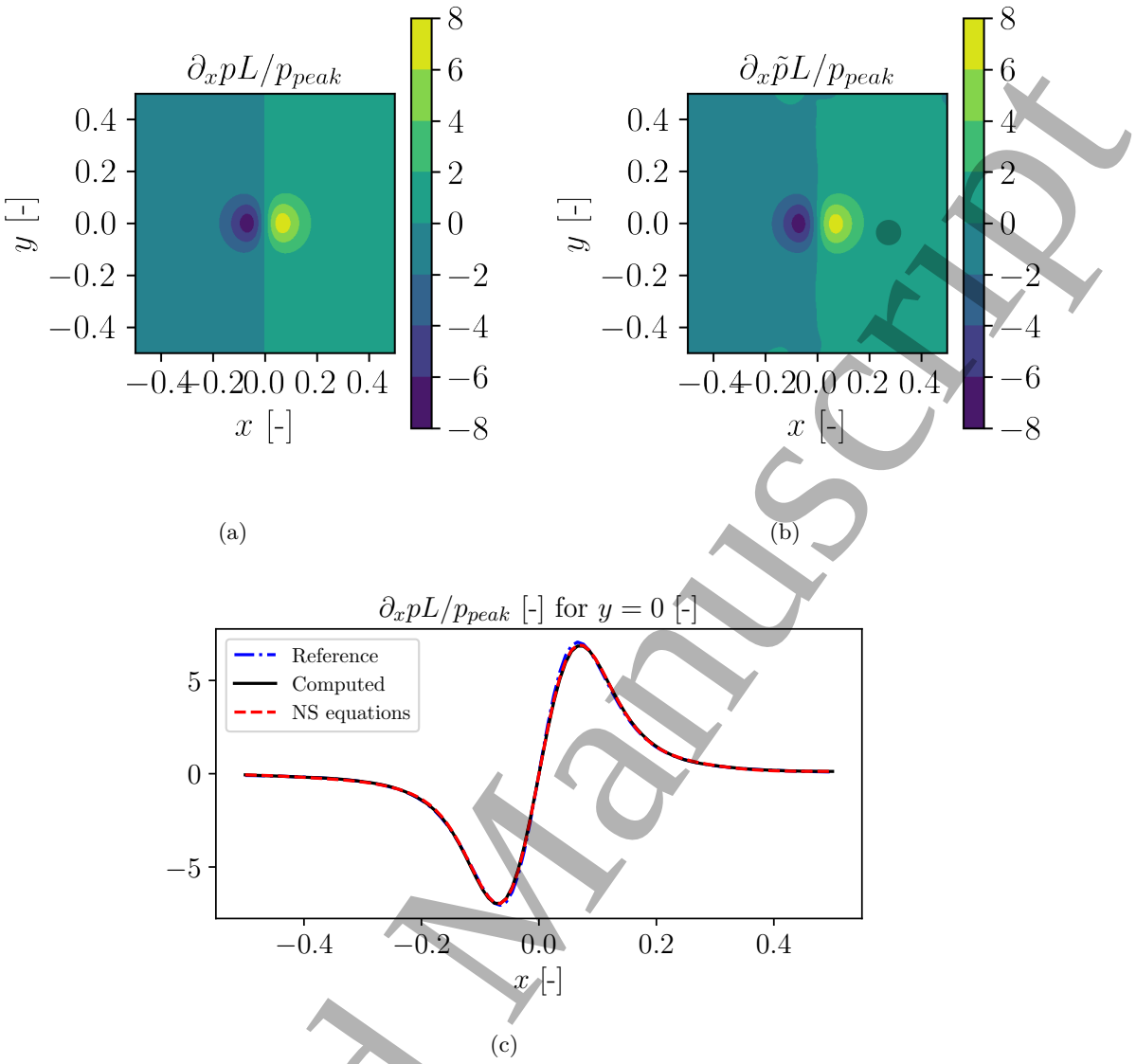
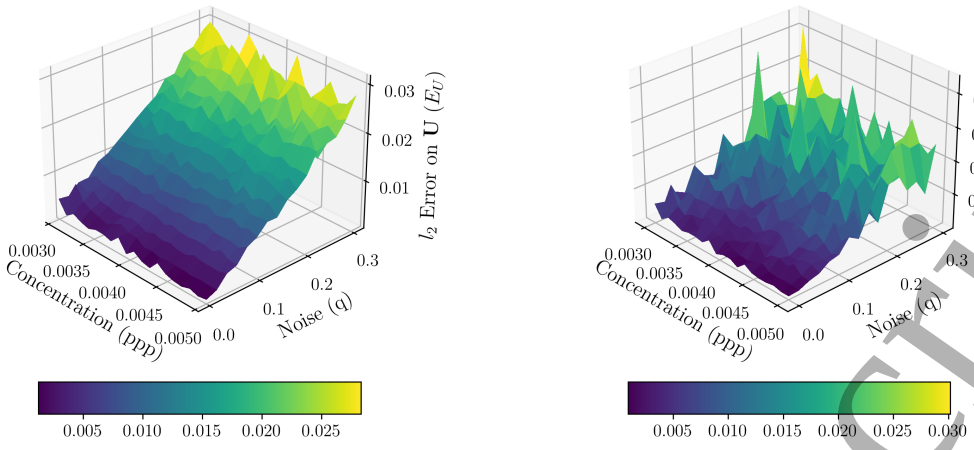


Figure 9: Comparison of the pressure gradient reconstruction with the background truth. Figure 9(a) shows the normalized pressure gradient $\partial_x p$, scaled with respect to p_{peak}/L, while Figure 9(b) shows the results obtained by differentiating the RBF solution. Figure 9(c) show the pressure gradient profile along $y = 0$.

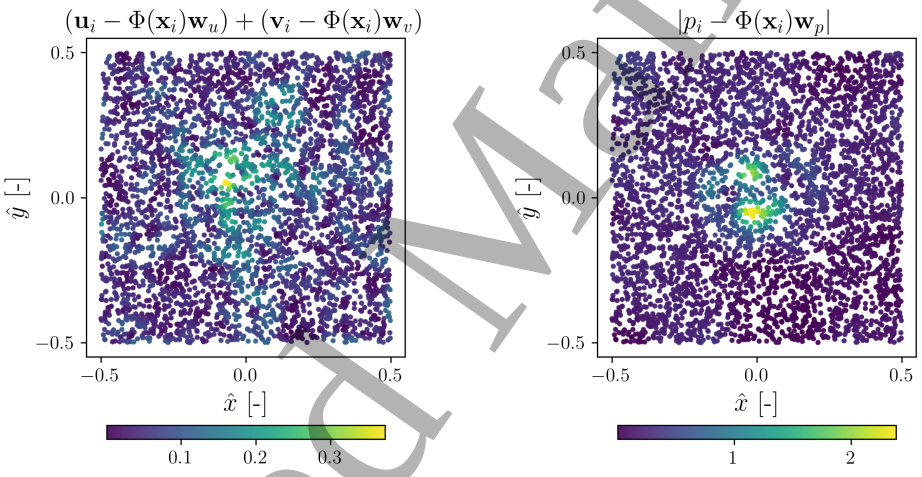
and the pressure integration. In the velocity regression, the divergence-free condition is imposed in all these points (leading to $n_\nabla = 748$ once common points are removed), while Dirichlet conditions are imposed only at walls walls and at the inlet. Therefore, we have $n_D = 600$ points with both conditions and a total of $n_\lambda = n_\nabla + 2n_D = 1948$ constraints (cf. Table 1).

For the pressure integration, Neumann conditions from the momentum equation (cf. equation (29)) in all of these points. Moreover, a Dirichlet condition is set on the top corner, at the inlet. This test case was analyzed with a number of particles ranging from $n_p = 6000$ to $n_p = 18755$ points and two noise levels, namely $q = 0$ (no noise) and $q = 0.1$. Figure 12 shows a close-up near the cylinder for the cases with the lowest and the highest concentration; it is worth pointing out that the lowest seeding only leaves a few dozens of particles in the region $r \in [R, 1.2R]$, which is the one where most of the velocity gradients are located. It is thus somewhat surprising that the velocity field is reasonably well reconstructed even in these conditions.



(a) (b)

Figure 10: Test case 1. l_2 errors in the velocity field reconstruction (left) and pressure integration (right) as defined in 45 as a function of the particle concentration N_p (in particles per pixels) and noise level q (see definitions in Sec. 3.1) .



(a) (b)

Figure 11: Test case 1. Local absolute error distributions for the velocity regression (a) and the pressure computation (b) with $N_{pp} = 0.004$ and $q = 0.15$ (cf. Section 3.1). The local error is everywhere below 1% for the pressure reconstruction.

Figure 13 shows the behaviour of the global convergence error as a function of the image density for both the velocity regression and the pressure computation. Three clustering approaches are compared, namely $\mathbf{n}_K = [4, 10, 20]$, $\mathbf{n}_K = [6, 10, 20]$ and $\mathbf{n}_K = [10, 10, 20]$ (see Sec.2.4). everFigs. 13(a) and 13(b) show the results in absence of noise. Although the coarser clustering (blue triangles markers) suffers at the lowest seeding densities (because at low seeding produce overly large RBFs),it is worth noticing that the global error for the velocity in noise free conditions (Figure 13(a)) is never larger than 2% and settles at 0.5% if sufficient particles are available. In the presence of noise (Figure 13(c)), the impact of the clustering method is more important and, for a sufficiently large number of particles, the error stabilizes in the range 0.5-1.2% depending on the collocation points. These results confirm the robustness of the

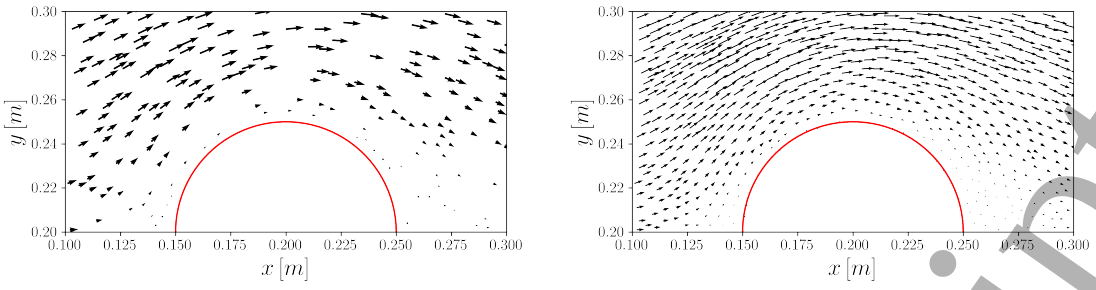


Figure 12: Test case 2. Zoomed view of the scattered velocity data used at the lowest (left, with $n_p = 6000$) and the highest (right, with $n_p = 18755$) seeding densities.

proposed approach for physics-informed (constrained) regression.

On the other hand, the pressure integration is more vulnerable and fails at the lowest seeding density for the coarser clustering (with global error up to 40%). Nevertheless, the global error drops to about 2% with the intermediate clustering if a reasonable amount of particles is available. As expected, the pressure integration appears to be more sensitive to noise and the clustering strategy versus the number of particles available. This is evident as the finest clustering (red circular markers) leads to failure over a wide range of concentrations. Among the tested clustering approaches, the intermediate one with $\mathbf{n}_K = [6, 10, 20]$ (green diamond markers) appears to be the most robust and is the one kept for the remainder of this section.

Considering a case with a significant amount of noise ($q = 0.1$) and the largest seeding density ($N_p = 0.089$, corresponding to $n_p = 18755$), Figures 14 shows the distribution of the error for the velocity regression (top) and the pressure integration (bottom). Both provide excellent results in the entire domain, with errors (defined as in eq. 45) of the order of 1%. For the same seeding and noise conditions, Figure 15 offers a closer view of the velocity regression and the pressure computation in the most challenging region, namely close to the cylinder walls. Three profiles are extracted at different angles for the velocity magnitude and the pressure. For each of the lines identified by the polar angle θ, velocity and pressure values are taken as the ones lying within the two planes, parallel to the radial direction and $\pm 2mm$ apart from the original line. The velocity and pressure available from the RBFs expansion are analytical and computed on the associated planes. The comparison is thus primarily qualitative, as the reference data is not taken precisely at the sampled planes.

For the velocity regression, the figures shows both the available CFD data (blue circular markers) and the data used for the regression (green diamonds markers), which was polluted by noise. The results shows that the regression removes the noise and provide an analytical solution (black continuous line) that satisfies the boundary conditions. This is in excellent agreement with the CFD data, where available, and shows a realistic fit in the remaining portions.

The vertical velocity profiles far from the cylinder walls are shown in Figure 16 for three positions, namely $x = 0.25, 0.5, 0.75$. The matching with the reference data is remarkable. To evaluate the accuracy of the proposed approach, it is worth comparing the errors obtained in this work with the errors obtained via Physics Informed Neural networks (PINNS) by Rao et al. (2020). Considering the case in which all the dataset is used, the number of training data used by the proposed RBF regression is equal to $n_p = 18755$ while Rao et al. (2020) uses $n_b = 50000$ points were used in Rao et al. (2020) using a Latin hypercube sampling (LHS). Yet, the global error obtained by the RBF regression of the velocity reconstruction is 10 times smaller ($E_U = 0.014$ versus the $E_U = 0.14$) than the error obtained via ANN-regression in Rao et al. (2020)). No information was found concerning the global error of the pressure field.

Finally, Figure 17 shows the pressure distribution around the cylinder walls in polar co-ordinates, comparing the results of the pressure integration (black continuous line) with the

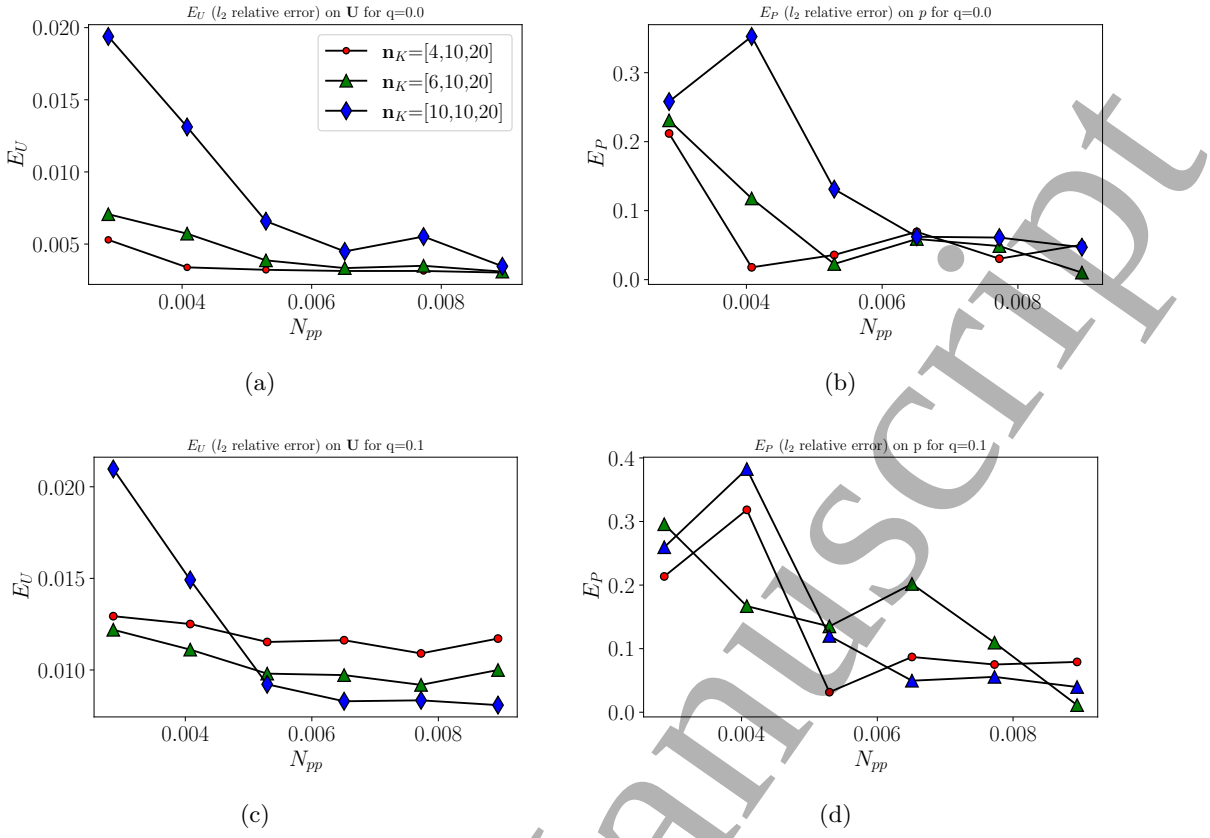


Figure 13: Test case 2. Global error for the velocity regression (left) and for the pressure integration (right). The figures on the top collects results for the noise-free cases ($q = 0$) while the figures on the bottom are related to the noisy cases ($q = 0.1$).

available CFD data. The results are in good agreement.

4.3 Test Case 3: The 3D Stokes Flow past a Sphere

We consider a case with $n_p = 18300$ in the domain $\hat{r} \in [0.5, 1]$, i.e. a thick spherical shell bounded on one side by the solid sphere. Using a clustering scheme with $\boldsymbol{n}_K = [6, 10, 20]$ results in $n_b = 10381$ RBFs. A total of 2111 constraints are placed on the sphere's surface at $\hat{r} = 0.5$ and 4879 on the outer surface at $\hat{r} = 1$. These points are used to impose the divergence-free conditions and the points at the wall are also used to impose $\boldsymbol{U} = 0$ at $\hat{r} = 0.5$. This results in $n_D = 2111$ and $n_\nabla = 6990$, thus a total of $n_\lambda = 3n_D + n_\nabla = 13323$ constraints for the velocity regression (cf. Table 1). In addition to these, a large penalty of $\alpha_\nabla = 25$ is used to promote the divergence-free condition over the entire domain. This results in a sub-optimal reconstruction of the velocity field but helps in the pressure integration.

The test case is analyzed with $q = 0$ (no noise) and $q = 0.05$. The local absolute error distribution is shown in Figure 18 for both ($q = 0$ on the left and $q = 0.05$ on the right). The global error for these test cases are $E_U = 0.1\%$ and $E_U = 0.6\%$ respectively. While in the first case the absolute errors are mostly produced next to the sphere's wall, in the second case these are mostly occurring in the outer surface. Nevertheless, the error is everywhere negligible in absence of noise and acceptable in the presence of noise. Figures 18(c) and 18(d) show the corresponding absolute error for the divergence of the flow. These are everywhere negligible, confirming the quality of the regression over the full domain.

To further illustrate the quality of the regression, Figure 19 shows the stream-wise velocity component at $\theta = 0^o$ and $\varphi = 0^o$, i.e. on an equatorial plane (taking the south pole on the

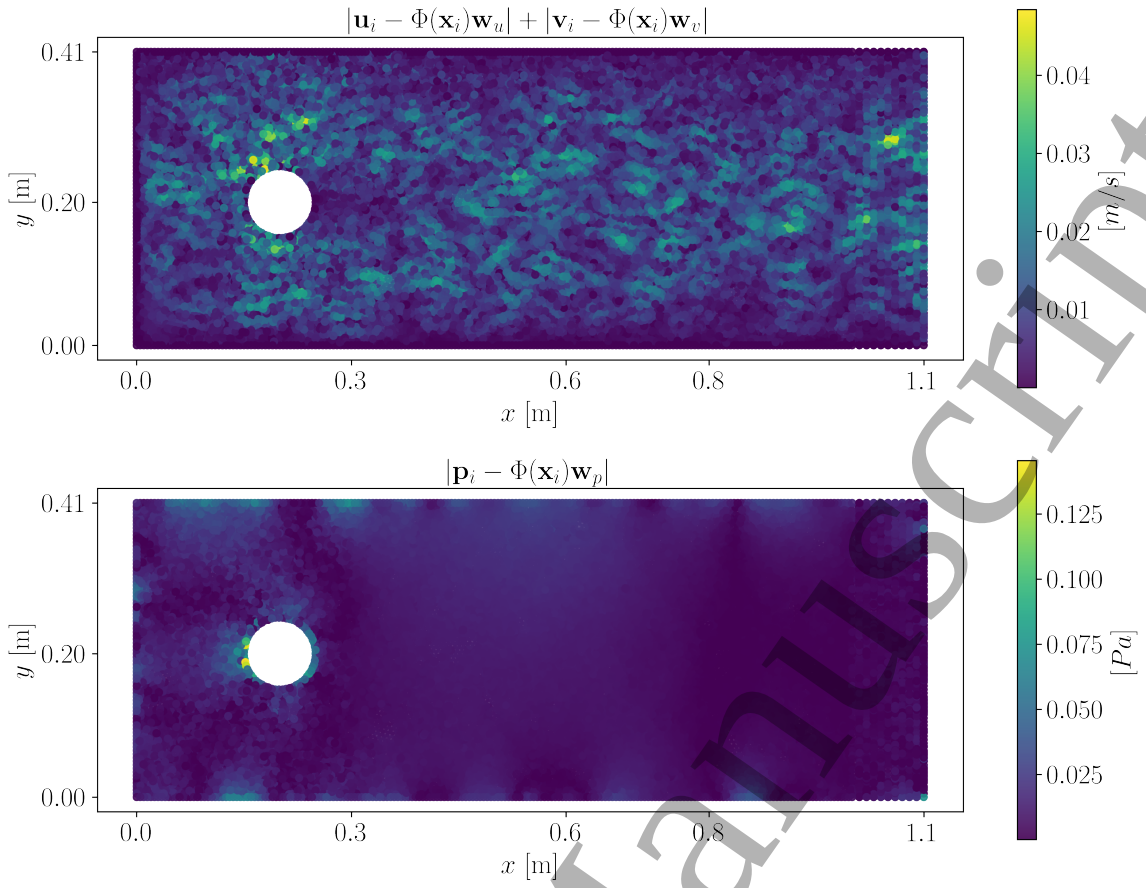


Figure 14: Test case 2. Local absolute error distribution for the velocity regression (top) and the pressure integration (bottom) for the 2D cylinder wake with a very noisy field $q = 0.1$ and the largest seeding concentration ($N_p = 0.089$, i.e $n_p = 18755$).

stagnation point). The RBF regression (black continuous line) is indistinguishable from the analytical data (blue dashed-dotted line).

The resulting fields were used to compute the pressure field, as usual, using the same RBF basis used for the velocity field. The same points used to constrain the velocity regression were used to impose Neumann conditions (cf. equation (29)) on the sphere's wall and on the outer surface. In addition to those, 6 points were used to impose Dirichlet conditions on the sphere's surface. These points mimic pressure taps located on the sphere's wall and greatly enhance the stability of the pressure computation. The importance of having Dirichlet conditions on some of the boundaries has been highlighted by Faiella et al. (2021); Pan et al. (2016, 2018), who have thoroughly analyzed the error propagation in the pressure integration near boundaries with Neumann conditions. This explains why the natural approach for integrating the pressure in the flow past a blunt body consists in setting Dirichlet boundary conditions in the far-field (see for example McClure and Yarusevych (2017b) and the works cited in table 1 from Pan et al. (2016)), where the flow irrotationality allows for using the Bernoulli theorem; this approach is however not possible in this test case.

Reducing the number of pressure taps to 1 has a detrimental effect on the pressure integration unless a much larger number of constraints is added on the velocity regression (in line with the observations by Faiella et al. (2021)). We leave a detailed analysis on the impact of number and location of Dirichlet boundary conditions (i.e. pressure taps) to future work and we encourage the reader considering this test case in future studies on the pressure integration from image velocimetry.

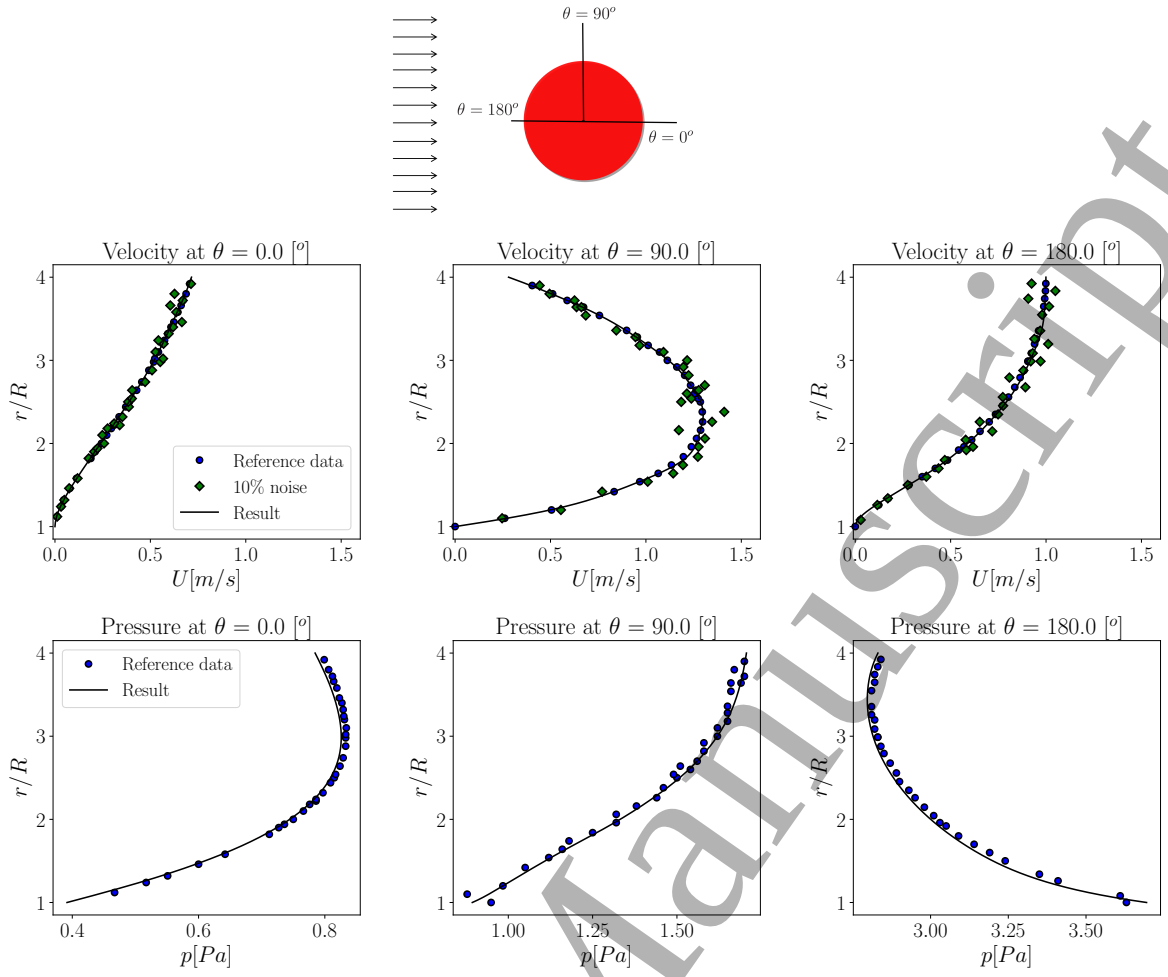


Figure 15: Test case 2. Velocity profiles (second row) and pressure profiles (third row) extracted at three planes with $\theta = 0^o, 90^o, 180^o$. The top panel recalls the flow orientation. Round blue markers are used for the CFD data, while green diamonds are used for the data employed in the (noisy) regression. The (analytic) result of the regression is shown with a continuous line.

The local absolute error for the pressure field is shown in Figure 20. These are characterized by a global error of $E_P = 3.2\%$ and $E_P = 8.2\%$ respectively. Although the differences in the velocity fields between the two cases are minor, the impact on the pressure integration is considerable. In both cases, the pressure error is located where the highest velocity error is produced (cf. Fig 18), namely on the surface wall for the noiseless test case (Fig. 20(a)) and on the outer surface for the noisy test case (Fig. 20(b)). While for the noiseless test case the impact is negligible, in the noisy test case the error is important far from the sphere's wall.

Nevertheless, it is worth highlighting that the pressure reconstruction is satisfactory in both cases in the proximity of the sphere, thanks to the contribution of the Dirichlet boundary conditions. Figure 21 shows the pressure distribution along the sphere for the noiseless and the noisy test cases. These are plotted as a function of θ because of the perfect axial symmetry of the regression. The impact of the noise appears negligible.

Finally, Figure 21(c) and 21(d) show the wall normal pressure gradient along the sphere's wall. In addition to the reference data (dashed-dotted blue line) and the RBF regression (black continuous line), the continuous red line shows the projection of the RBF velocity field according to equation (29). The excellent agreement between these two shows that the (minor) discrepancy with respect to the reference data is due to the (minor) discrepancy in the reconstruction of the velocity field and not the pressure integration itself.

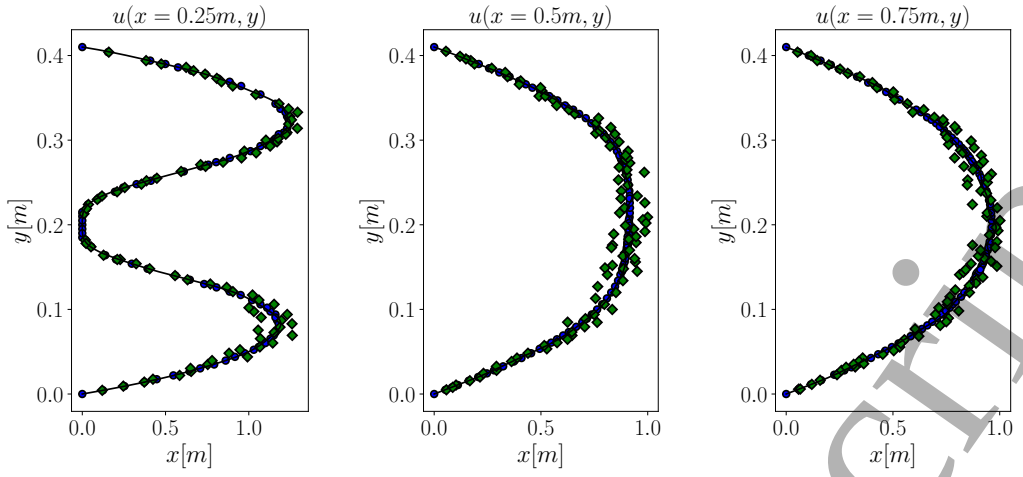


Figure 16: Test case 2. Vertical velocity profiles (u component only) taken at $x = 0.25, 0.5, 0.75$[mm]. The same legend used in Figure 15 is used.

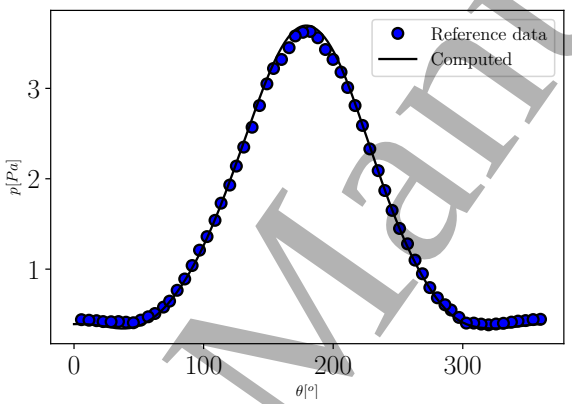


Figure 17: Test case 2. Pressure distribution around the cylinder wall, comparing the available CFD data (blue round markers) with the results of the mesh-less integration (black continuous line).

5 Conclusions and Perspectives

We presented a constrained Radial Basis Function (RBF) method for the regression of velocity fields and for the meshless computation of pressure fields in incompressible flows. While the derivation focuses on steady laminar flows, its extension to unsteady and turbulent flows is relatively straightforward and is currently under development. Moreover, the proposed methodology applies identically if the velocity data is available on Cartesian grids as in cross-correlation based velocimetry or on scattered points as in tracking-based velocimetry.

We presented all the details of the mathematical framework and showed that both the velocity regression and the pressure integration can be cast as quadratic problems with linear constraint, resulting in two classic Karush-Kuhn-Tucker (KKT) systems. A simple direct solution method, based on Schur complements and Cholesky factorizations, was introduced together with a hierarchical clustering technique to automatically compute RBFs' collocation points and shape factors.

The constraints are used to impose boundary conditions (e.g. no-slip) and physical priors (e.g. divergence-free or compliance with the Navier-Stokes equations). The result is an analytic expression for both velocity and pressure fields. Constraints allow a "physics-informed"

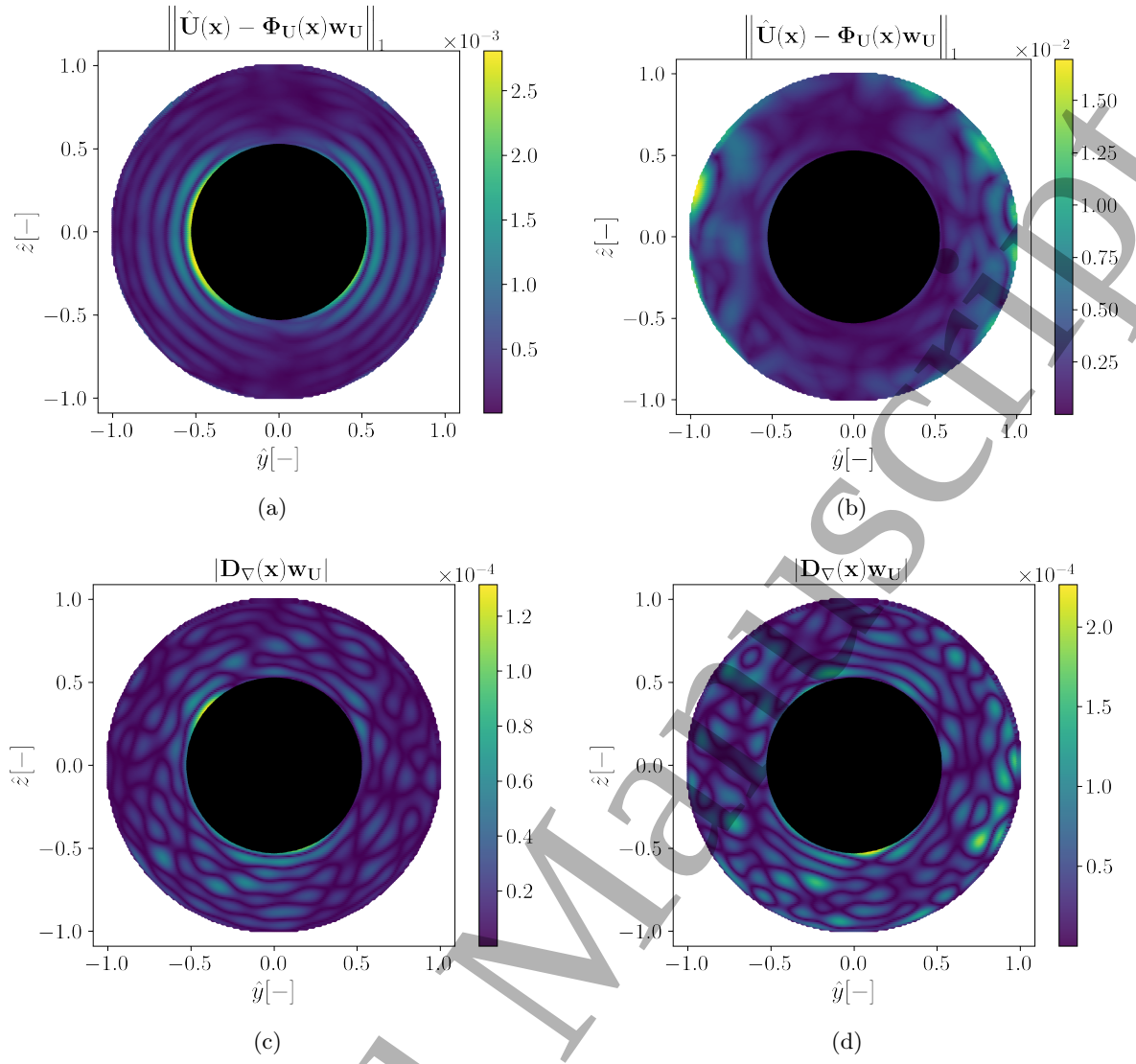


Figure 18: Test case 3. Local absolute error distribution for the velocity regression. Fig. 18(a) shows the results for the noiseless ($q = 0$) test case, Fig. 18(b) for the noisy ($q = 0.05$) test case. Fig. 18(c) and 18(d) show the corresponding distribution of error for the flow divergence.

regression, similar to what is done in "physics-informed" artificial neural networks (ANN). However, the RBF-based regression is considerably simpler than the ANN-based regression, and the conditions are introduced as hard constraints (with Lagrange multipliers) and not penalties.

We illustrate the RBF approach on three test cases of growing complexity, from a small 2D problem to a fairly large 3D problem. All the selected test cases are numerical and therefore offer the possibility of comparing the result with reference data, validating the method and testing its robustness against noise and seeding concentration. The application to experimental data is currently under development and will be presented in a dedicated article.

The results of the velocity regression proved to be remarkably robust in all test cases and all investigated conditions. On the other hand, the pressure integration proved to be strongly sensitive to small errors in the velocity regression if these occur in the proximity of boundaries at which Neumann boundary conditions are to be imposed from the Navier-Stokes equation. Nevertheless, even in the worst investigated test case (with 8% of global error), the pressure reconstruction near solid boundaries (where some Dirichlet boundary conditions might be applied) is excellent.

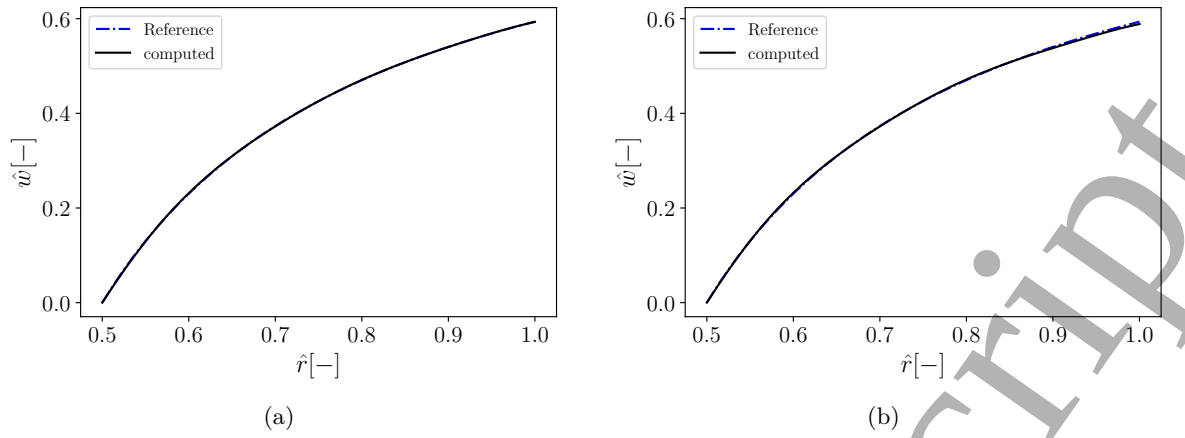


Figure 19: Test case 3. Velocity profile at $\theta = 0^o$ and $\varphi = 0^o$, from $\hat{r} = 0.5$, for the noiseless (Fig. 19(a)) and the noisy (Fig. 19(b)) test cases. The reference data (blue dashed-dotted line) and the RBF regression (black continuous line) are nearly indistinguishable.

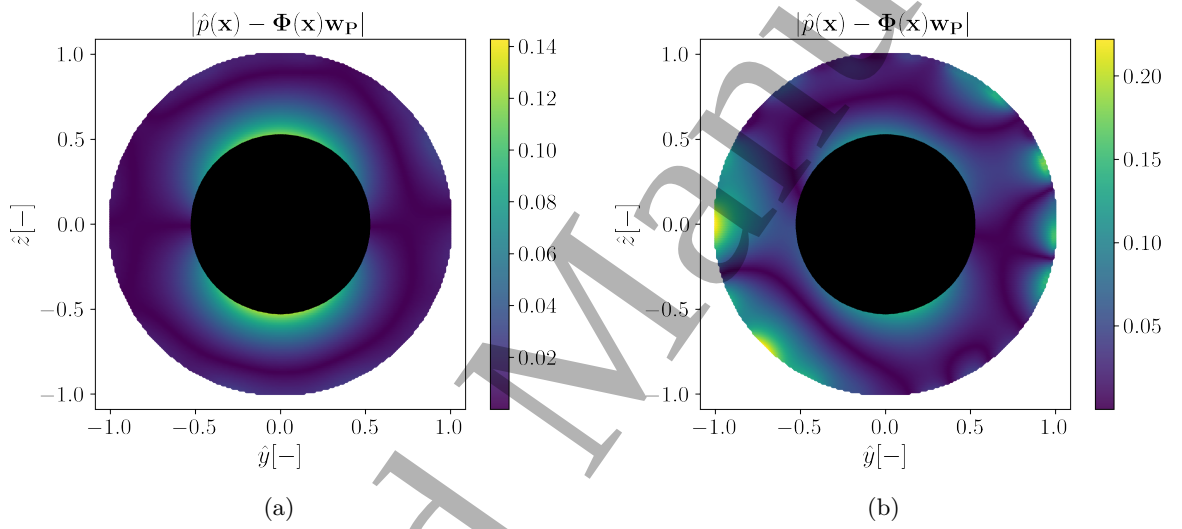


Figure 20: Test case 3. Local absolute error distribution for the pressure field. Fig. 20(a) shows the results for the noiseless ($q = 0$) test case, Fig. 20(b) for the noisy ($q = 0.05$) test case.

Besides the aforementioned extension to unsteady and turbulent flows and the implementation of experimental data, ongoing work is focused on reducing the memory requirements following two research paths. On the one hand, by implementing iterative methods to solve the least square problems. On the other hand, developing techniques to promote sparsity of the basis matrix, combining compact supports RBFs with the Partition of Unity Method (PUM).

A Velocity Approximations in 3D

In a 3D problem, the collected velocity field can be arranged in the column vector $\boldsymbol{U} = (\boldsymbol{u}; \boldsymbol{v}; \boldsymbol{w}) \in \mathbb{R}^{3n_p}$, where $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^{n_p}$ collects the entries of the three velocity components $U_i = (u_i, v_i, w_i)$. In this section, the symbol $\boldsymbol{w}$ refers to the velocity in the z direction and not the vector of weights as in (4). The definition of the collocation points are adapted from the

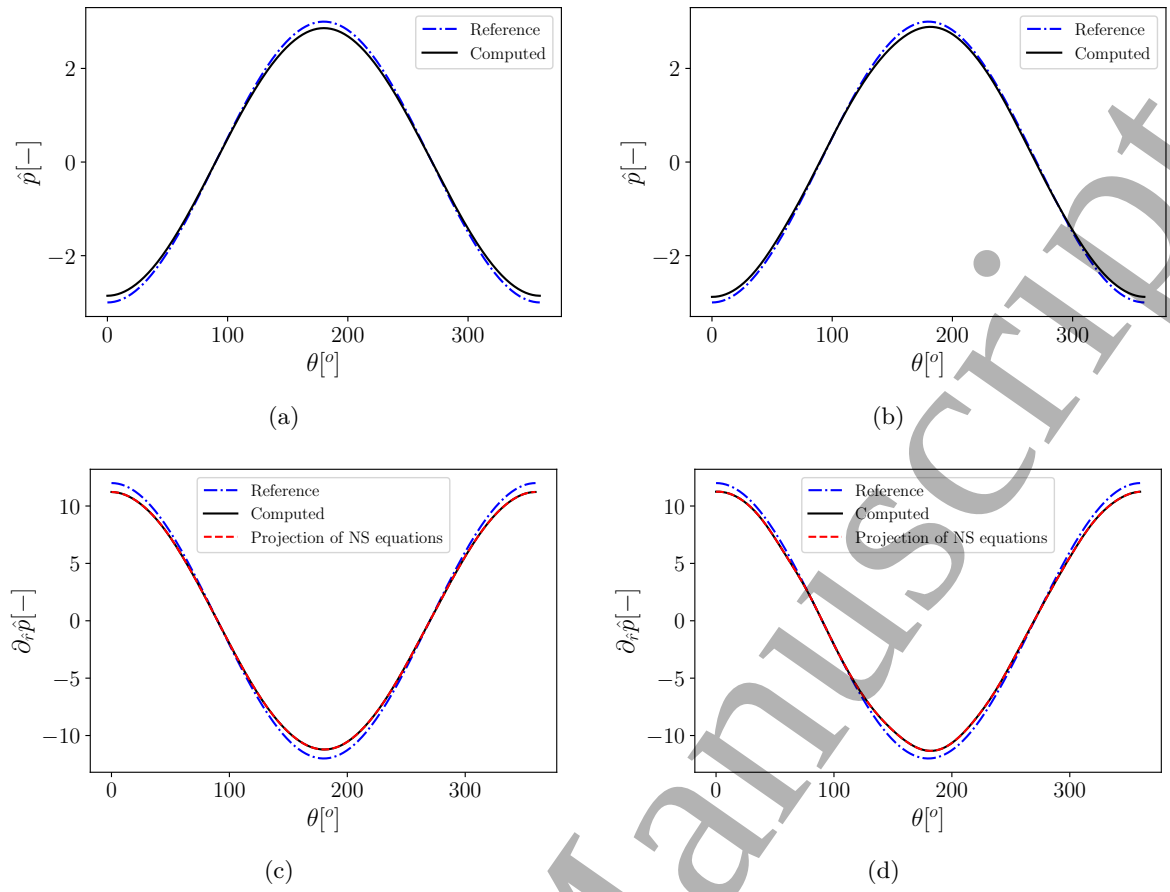


Figure 21: Test case 3. Figs. 21(a) and 21(b) show the pressure distribution along the sphere's wall for the noiseless and the noisy case respectively. Similarly, 21(c) and 21(d) show the wall-normal pressure gradient.

2D case to $\boldsymbol{x}_k^* = (x_k^*, y_k^*, z_k^*)$. Equation (10) is extended to:

$$\tilde{\boldsymbol{U}} = \begin{pmatrix} \tilde{\boldsymbol{u}} \\ \tilde{\boldsymbol{v}} \\ \tilde{\boldsymbol{w}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{X}) & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}(\boldsymbol{X}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Phi}(\boldsymbol{X}) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_{\boldsymbol{u}} \\ \boldsymbol{w}_{\boldsymbol{v}} \\ \boldsymbol{w}_{\boldsymbol{w}} \end{pmatrix} = \boldsymbol{\Phi}_U(\boldsymbol{X})\boldsymbol{w}_U \,, \tag{46}$$

where $\boldsymbol{w}_{\boldsymbol{w}} \in \mathbb{R}^{n_b}$ are the weights for the third velocity component. The involved matrices and arrays increase their size accordingly, and are thus $\boldsymbol{\Phi}_U(\boldsymbol{x}_i) \in \mathbb{R}^{3n_p \times 3n_b}$ and $\boldsymbol{w}_U = (\boldsymbol{w}_{\boldsymbol{u}}; \boldsymbol{w}_{\boldsymbol{v}}; \boldsymbol{w}_{\boldsymbol{w}}) \in \mathbb{R}^{3n_b}$. The RBF derivatives in the z-direction can be added to the two dimensional set (12) :

$$\partial_z \varphi_k \left(\boldsymbol{X} \mid \boldsymbol{x}_k^*, c_k \right) = -2c_k^2 \left(z - z_k^* \right) \varphi_k \left(\boldsymbol{X} \mid \boldsymbol{x}_k^*, c_k \right). \tag{47}$$

Analogously to the other velocities component (13), it is possible to differentiate the third velocity:

$$\partial_z w \left(\boldsymbol{X} \right) \approx \sum_{k=0}^{n_c} w_{wk} \partial_z \varphi_k \left(\boldsymbol{X} \mid \boldsymbol{x}_k^*, c_k \right) = \boldsymbol{\Phi}_z \left(\boldsymbol{X} \right) \boldsymbol{w}_w, \tag{48}$$

where $\boldsymbol{\Phi}_z(\boldsymbol{X}) \in \mathbb{R}^{n_p \times n_b}$ is defined as in (14). In line with the notation in Section 2, the divergence operator becomes:

$$\nabla \cdot \begin{pmatrix} \boldsymbol{u}\left(\boldsymbol{X}_\nabla\right) \\ \boldsymbol{v}\left(\boldsymbol{X}_\nabla\right) \\ \boldsymbol{w}\left(\boldsymbol{X}_\nabla\right) \end{pmatrix} \approx \begin{pmatrix} \boldsymbol{\Phi}_x\left(\boldsymbol{X}_\nabla\right) & \boldsymbol{\Phi}_y\left(\boldsymbol{X}_\nabla\right) & \boldsymbol{\Phi}_z\left(\boldsymbol{X}_\nabla\right) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_{\boldsymbol{u}} \\ \boldsymbol{w}_{\boldsymbol{v}} \\ \boldsymbol{w}_{\boldsymbol{w}} \end{pmatrix} = \boldsymbol{D}_\nabla\left(\boldsymbol{X}_\nabla\right) \boldsymbol{w}_U = \boldsymbol{0} \tag{49}$$

where $\boldsymbol{X}_\nabla \in \mathbb{R}^{n_\nabla \times 3}$ and $\boldsymbol{D}_\nabla \in \mathbb{R}^{n_\nabla \times 3n_b}$. Denoting $\boldsymbol{n} = (n_x, n_y, n_z)$ as the normal vector to a surface and $\boldsymbol{n}_z(\boldsymbol{X}_N) \in \mathbb{R}^{n_N}$ with $\boldsymbol{X}_N \in \mathbb{R}^{n_N \times 3}$, the directional derivative is:

$$\boldsymbol{\Phi}_{\boldsymbol{n}}(\boldsymbol{X}_N) = \boldsymbol{\Phi}_x(\boldsymbol{X}_N) \odot \boldsymbol{n}_x(\boldsymbol{X}_N) + \boldsymbol{\Phi}_y(\boldsymbol{X}_N) \odot \boldsymbol{n}_y(\boldsymbol{X}_N) + \boldsymbol{\Phi}_z(\boldsymbol{X}_N) \odot \boldsymbol{n}_z(\boldsymbol{X}_N). \tag{50}$$

Linear constraints on the normal derivatives on a surface with normal $\boldsymbol{n}$ (cf. equation (17) for the 2D) become:

$$\begin{pmatrix} \boldsymbol{\Phi}_{\boldsymbol{n}}(\boldsymbol{X}_N) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_n(\boldsymbol{X}_N) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_n(\boldsymbol{X}_N) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_u \\ \boldsymbol{w}_v \\ \boldsymbol{w}_w \end{pmatrix} = \boldsymbol{N}_{\boldsymbol{n}}(\boldsymbol{X}_N)\, \boldsymbol{w}_U = \boldsymbol{c}_N,$$

While the linear constraints on the velocity field (cf. equation (16) for the 2D) become

$$\begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{X}_D) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}(\boldsymbol{X}_D) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}(\boldsymbol{X}_D) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_u \\ \boldsymbol{w}_v \\ \boldsymbol{w}_w \end{pmatrix} = \boldsymbol{D}(\boldsymbol{X}_D)\, \boldsymbol{w}_U = \boldsymbol{c}_D, \tag{51}$$

where $\boldsymbol{c}_D = (\boldsymbol{c}_{Du}; \boldsymbol{c}_{Dv}; \boldsymbol{c}_{Dw}) \in \mathbb{R}^{3n_D}$, with $\boldsymbol{c}_{Dw} \in \mathbb{R}^{n_D}$ the values constraining the third velocity component and $\boldsymbol{X}_D \in \mathbb{R}^{n_D \times 3}$ collecting the points where the condition is set. Finally, the problem can be cast in the general form as in (19), here reported for completeness together with the related dimension:

$$\boldsymbol{A}_U = 2\boldsymbol{\Phi}_U^T \boldsymbol{\Phi}_U + 2\alpha_\nabla \boldsymbol{D}_\nabla^T \boldsymbol{D}_\nabla \in \mathbb{R}^{3n_b \times 3n_b} \tag{52a}$$

$$\boldsymbol{B}_U = \left(\boldsymbol{D}_\nabla^T, \boldsymbol{D}^T, \boldsymbol{N}_{\boldsymbol{n}}^T\right) \in \mathbb{R}^{3n_b \times n_\lambda} \tag{52b}$$

$$\boldsymbol{b}_{U1} = 2\boldsymbol{\Phi}_U^T \boldsymbol{U} \in \mathbb{R}^{3n_b} \tag{52c}$$

$$\boldsymbol{b}_{U2} = (\mathbf{0}, \boldsymbol{c}_D, \boldsymbol{c}_N)^T \in \mathbb{R}^{n_\lambda}, \tag{52d}$$

where $n_\lambda = n_\nabla + 3n_D + 3n_N$.

B Pressure Integration in 3D

The problem of pressure integration is less affected by the increased dimension of the problem when moving from 2D to 3D. The main difference concerns the forcing term:

$$\nabla \cdot (U \cdot \nabla U) = (\partial_x u)^2 + (\partial_y v)^2 + (\partial_z w)^2 + 2\partial_x v \partial_y u + 2\partial_x w \partial_z u + 2\partial_y w \partial_z v, \tag{53}$$

which can then be written with the same formalism as (23). Equation (29) requires particular attention as in 3D $\boldsymbol{n} = (n_x, n_y, n_z)$ and all three momentum equation must be projected. If the aforementioned changes are applied, the obtained matrices do not differ from the 2D and therefore equations (30) and (31) hold true.

References

K. Agarwal, O. Ram, J. Wang, Y. Lu, and J. Katz. Reconstructing velocity and pressure from noisy sparse particle tracks using constrained cost minimization. *Experiments in Fluids*, 62 (4), mar 2021. doi: 10.1007/s00348-021-03172-0.

I. Azijli, A. Sciacchitano, D. Ragni, A. Palha, and R. P. Dwight. A posteriori uncertainty quantification of PIV-based pressure data. *Experiments in Fluids*, 57(5), apr 2016. doi: 10.1007/s00348-016-2159-z.

R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. John Wiley & Sons, 2nd edition, 2006.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2006. ISBN 0387310738.

Maxim Bobrov, Mikhail Hrebtov, Vladislav Ivashchenko, Rustam Mullyadzhanov, Alexander Seredkin, Mikhail Tokarev, Dinar Zaripov, Vladimir Dulin, and Dmitriy Markovich. Pressure evaluation from lagrangian particle tracking data using a grid-free least-squares method. *Measurement Science and Technology*, 32(8):084014, may 2021. doi: 10.1088/1361-6501/abf95c.

D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2, 1988.

L. D. C. Casa and P. S. Krueger. Radial basis function interpolation of unstructured, three-dimensional, volumetric particle tracking velocimetry data. *Measurement Science and Technology*, 24(6):065304, may 2013. doi: 10.1088/0957-0233/24/6/065304.

J. J. Charonko, C. V. King, B. L. Smith, and P. P. Vlachos. Assessment of pressure field calculations from particle image velocimetry measurements. *Measurement Science and Technology*, 21(10):105401, aug 2010. doi: 10.1088/0957-0233/21/10/105401.

W. Chen. New RBF collocation methods and kernel RBF with applications. In *Lecture Notes in Computational Science and Engineering*, pages 75–86. Springer Berlin Heidelberg, 2003. doi: 10.1007/978-3-642-56103-0_6.

W. Chen and M. Tanaka. A meshless, integration-free, and boundary-only RBF technique. *Computers & Mathematics with Applications*, 43(3-5):379–391, feb 2002. doi: 10.1016/s0898-1221(01)00293-0.

A. H.-D. Cheng, M. A. Golberg, E. J. Kansa, and G. Zammito. Exponential convergence and h-c multiquadric collocation method for partial differential equations. *Numerical Methods for Partial Differential Equations*, 19(5):571–594, may 2003. doi: 10.1002/num.10062.

E. K. P. Chong and S. H. Zak. *An Introduction to Optimization*. John Wiley & Sons, 2013.

R. de Kat and B. Ganapathisubramani. Pressure from particle image velocimetry for convective flows: a taylor's hypothesis approach. *Measurement Science and Technology*, 24(2):024002, dec 2012. doi: 10.1088/0957-0233/24/2/024002.

R. de Kat and B. W. Van Oudheusden. Instantaneous planar pressure determination from PIV in turbulent flow. *Experiments in Fluids*, 52(5):1089–1106, dec 2011. doi: 10.1007/s00348-011-1237-5.

J. W. Van der Kindere, A. Laskari, B. Ganapathisubramani, and R. de Kat. Pressure from 2d snapshot PIV. *Experiments in Fluids*, 60(2), jan 2019. doi: 10.1007/s00348-019-2678-5.

Matthew Faiella, Corwin Grant Jeon Macmillan, Jared P Whitehead, and Zhao Pan. Error propagation dynamics of velocimetry-based pressure field calculations (2): on the error profile. *Measurement Science and Technology*, 32(8):084005, may 2021. doi: 10.1088/1361-6501/abf30d.

F. Felis-Carrasco, D. Hess, B. B. Watz, and M. A. Mendez. A study on piv-based pressure measurements using cfd techniques. In *15th International Symposium on Particle Image Velocimetry (ISPIV2021)*, 2021.

B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, jan 1996. doi: 10.1017/cbo9780511626357.

B. Fornberg and N. Flyer. Solving PDEs with radial basis functions. *Acta Numerica*, 24: 215–258, apr 2015. doi: 10.1017/s0962492914000130.

R. Franke. Scattered data interpolation: Tests of some method. *Mathematics of Computation*, 38(157):181, jan 1982. doi: 10.2307/2007474.

S. Gesemann, F. Hunh, D. Schanz, and A. Schröder. 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*, 2016.

Sebastian Gesemann. From particle tracks to velocity and acceleration fields using b-splines and penalties. October 2015.

S. Ghaemi, D. Ragni, and F. Scarano. PIV-based pressure fluctuations in the turbulent boundary layer. *Experiments in Fluids*, 53(6):1823–1840, oct 2012. doi: 10.1007/s00348-012-1391-4.

Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1111–1145, oct 1987. doi: 10.1002/fld.1650071008.

E. Gunaydinoglu and D. F. Kurtulus. Pressure–velocity coupling algorithm-based pressure reconstruction from PIV for laminar flows. *Experiments in Fluids*, 61(1), nov 2019. doi: 10.1007/s00348-019-2831-1.

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

R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, mar 1971. doi: 10.1029/jb076i008p01905.

F. Huhn, D. Schanz, S. Gesemann, and A. Schröder. FFT integration of instantaneous 3d pressure gradient fields measured by lagrangian particle tracking in turbulent flows. *Experiments in Fluids*, 57(9), sep 2016. doi: 10.1007/s00348-016-2236-3.

F. Huhn, D. Schanz, P. Manovski, S. Gesemann, and A. Schröder. Time-resolved large-scale volumetric pressure fields of an impinging jet from dense lagrangian particle tracking. *Experiments in Fluids*, 59(5), apr 2018. doi: 10.1007/s00348-018-2533-0.

M. L. Jakobsen, T. P. Dewhirst, and C. A. Greated. Particle image velocimetry for predictions of acceleration fields and force within fluid flows. *Measurement Science and Technology*, 8 (12):1502–1516, dec 1997. doi: 10.1088/0957-0233/8/12/013.

Volker John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3d navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 40(6):775–798, 2002. doi: 10.1002/fld.377.

E. J. Kansa and R. E. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Computers & Mathematics with Applications*, 24(12):99–120, dec 1992. doi: 10.1016/0898-1221(92)90174-g.

E.J. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—i surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, 19(8-9):127–145, 1990a. doi: 10.1016/0898-1221(90)90270-t.

E.J. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19(8-9):147–161, 1990b. doi: 10.1016/0898-1221(90)90271-k.

S. Karri, J. Charonko, and P. P. Vlachos. Robust wall gradient estimation using radial basis functions and proper orthogonal decomposition (POD) for particle image velocimetry (PIV) measured fields. *Measurement Science and Technology*, 20(4):045401, feb 2009. doi: 10.1088/0957-0233/20/4/045401.

J. Köngeter. Piv with high temporal resolution for the determination of local pressure reductions from coherent turbulence phenomena. *Experiments in Fluids*, 29, 1999.

R. Kress. *Numerical Analysis*. Springer New York, 1998. doi: 10.1007/978-1-4612-0599-9.

I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178.

Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python*. Springer, 2017. ISBN 978-3-319-52461-0. doi: 10.1007/978-3-319-52462-7.

A. Laskari, R. de Kat, and B. Ganapathisubramani. Full-field pressure from snapshot and time-resolved volumetric PIV. *Experiments in Fluids*, 57(3), mar 2016. doi: 10.1007/s00348-016-2129-5.

Y. Li and F. Mei. Deep learning-based method coupled with small sample learning for solving partial differential equations. *Multimedia Tools and Applications*, jul 2020. doi: 10.1007/s11042-020-09142-8.

X. Liu and J. Katz. Instantaneous pressure and material acceleration measurements using a four-exposure PIV system. *Experiments in Fluids*, 41(2):227–240, may 2006. doi: 10.1007/s00348-006-0152-7.

X. Liu and J. R. Moreto. Error propagation from the PIV-based pressure gradient to the integrated pressure by the omnidirectional integration method. *Measurement Science and Technology*, 31(5):055301, feb 2020. doi: 10.1088/1361-6501/ab6c28.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, mar 1982. doi: 10.1109/tit.1982.1056489.

J. McClure and S. Yarusevych. Instantaneous PIV/PTV-based pressure gradient estimation: a framework for error analysis and correction. *Experiments in Fluids*, 58(8), jul 2017a. doi: 10.1007/s00348-017-2369-z.

Jeffrey McClure and Serhiy Yarusevych. Optimization of planar PIV-based pressure estimates in laminar and turbulent wakes. *Experiments in Fluids*, 58(5), apr 2017b. doi: 10.1007/s00348-017-2337-7.

Jeffrey McClure and Serhiy Yarusevych. Instantaneous PIV/PTV-based pressure gradient estimation: a framework for error analysis and correction. *Experiments in Fluids*, 58(8), jul 2017c. doi: 10.1007/s00348-017-2369-z.

N. J. Neeteson and D. E. Rival. Pressure-field extraction on unstructured flow data using a voronoi tessellation-based networking algorithm: a proof-of-principle study. *Experiments in Fluids*, 56(2), feb 2015. doi: 10.1007/s00348-015-1911-0.

Frank Nielsen. *Introduction to HPC with MPI for Data Science*. Springer International Publishing, 2016. doi: 10.1007/978-3-319-21903-5.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag GmbH, 2006. ISBN 0387303030.

M. Novara and F. Scarano. A particle-tracking approach for accurate material derivative measurements with tomographic PIV. *Experiments in Fluids*, 54(8), jul 2013. doi: 10.1007/s00348-013-1584-5.

Z. Pan, J. Whitehead, S. Thomson, and T. Truscott. 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, jul 2016. doi: 10.1088/0957-0233/27/8/084012.

Zhao Pan, Jared P. Whitehead, Geordie Richards, Tadd T. Truscott, and Barton L. Smith. Error propagation dynamics of piv-based pressure field calculation (3): What is the minimum resolvable pressure in a reconstructed field? July 2018.

A. Pirnia, J. McClure, S. D. Peterson, B. T. Helenbrook, and B. D. Erath. Estimating pressure fields from planar velocity data around immersed bodies: a finite element approach. *Experiments in Fluids*, 61(2), jan 2020. doi: 10.1007/s00348-020-2886-z.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, feb 2019. doi: 10.1016/j.jcp.2018.10.045.

R. Rannacher. *Numerical Linear Algebra*. Heidelberg University Publishing, 2018. doi: 10.17885/HEIUP.407.

Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212, mar 2020. doi: 10.1016/j.taml.2020.01.039.

Manuel Ratz, Domenico Fiorini, Alessia Simonini, Christian Cierpka, and Miguel A. Mendez. Analysis of an unsteady quasi-capillary channel flow with time resolved piv and rbf-based super resolution. February 2022.

S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2/3):193–210, 1999.

B. Šarler. From global to local radial basis function collocation method for transport phenomena. In *Advances in Meshfree Techniques*, pages 257–282. Springer Netherlands, 2007. doi: 10.1007/978-1-4020-6095-3_14.

S. A. Sarra and S. Cogar. An examination of evaluation algorithms for the RBF method. *Engineering Analysis with Boundary Elements*, 75:36–45, feb 2017. doi: 10.1016/j.enganabound.2016.11.006.

D. Schanz, S. Gesemann, and A. Schröder. Shake-the-box: Lagrangian particle tracking at high particle image densities. *Experiments in Fluids*, 57(5), apr 2016. doi: 10.1007/s00348-016-2157-1.

J. F. G. Schneiders and F. Scarano. Dense velocity reconstruction from tomographic PTV with material derivatives. *Experiments in Fluids*, 57(9), aug 2016. doi: 10.1007/s00348-016-2225-6.

J. F. G. Schneiders, S. Pröbsting, R. P. Dwight, B. W. van Oudheusden, and F. Scarano. Pressure estimation from single-snapshot tomographic PIV in a turbulent boundary layer. *Experiments in Fluids*, 57(4), mar 2016. doi: 10.1007/s00348-016-2133-9.

F. Schwenker, Hans A. Kestler, and G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14(4-5):439–458, may 2001. doi: 10.1016/s0893-6080(01)00027-2.

M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark computations of laminar flow around a cylinder. In *Notes on Numerical Fluid Mechanics (NNFM)*, pages 547–566. Vieweg + Teubner Verlag, 1996. doi: 10.1007/978-3-322-89849-4_39.

D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web - WWW '10*. ACM Press, 2010. doi: 10.1145/1772690.1772862.

J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, dec 2018. doi: 10.1016/j.jcp.2018.08.029.

R. Solis-Oba. Approximation algorithms for the k-median problem. In *Lecture Notes in Computer Science*, pages 292–320. Springer Berlin Heidelberg, 2006. doi: 10.1007/11671541_10.

P. L. van Gent, D. Michaelis, B. W. van Oudheusden, P. É. Weiss, R. de Kat, A. Laskari, Y. J. Jeon, L. David, D. Schanz, F. Huhn, S. Gesemann, M. Novara, C. McPhaden, N. J. Neeteson, D. E. Rival, J. F. G. Schneiders, and F. F. J. Schrijer. Comparative assessment of pressure field reconstructions from particle image velocimetry measurements and lagrangian particle tracking. *Experiments in Fluids*, 58(4), mar 2017. doi: 10.1007/s00348-017-2324-z.

B. W. van Oudheusden. PIV-based pressure measurement. *Measurement Science and Technology*, 24(3):032001, jan 2013. doi: 10.1088/0957-0233/24/3/032001.

B. W. van Oudheusden, F. Scarano, E. W. M. Roosenboom, E. W. F. Casimiri, and L. J. Souverein. Evaluation of integral forces and pressure fields from planar velocimetry data for incompressible and compressible flows. *Experiments in Fluids*, 43(2-3):153–162, feb 2007. doi: 10.1007/s00348-007-0261-y.

J. Wang, C. Zhang, and J. Katz. GPU-based, parallel-line, omni-directional integration of measured pressure gradient field to obtain the 3d pressure distribution. *Experiments in Fluids*, 60(4), mar 2019. doi: 10.1007/s00348-019-2700-y.

B. Šarler. A radial basis function collocation approach in computational fluid dynamics. *Computer Modeling in Engineering & Sciences*, 7(2):185–194, 2005. ISSN 1526-1506. doi: 10.3970/cmes.2005.007.185.