A meshless method to compute pressure fields from image velocimetry

Pietro Sperotto$^{1,2}$, Sandra Pieraccini$^2$ and 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, Torino, Italy

E-mail: mendez@vki.ac.be

Received 23 December 2021, revised 19 April 2022
Accepted for publication 17 May 2022
Published 9 June 2022

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

(Some figures may appear in colour only in the online journal)

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 fluid 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. An extensive literature review and a comparative analysis between various methods can be found in Charonko et al (2010), de Kat and Oudheusden (2011), van Oudheusden (2013), Pan et al (2016), van Gent et al (2017).

$^*$ Author to whom any correspondence should be addressed.
compute instantaneous pressures (de Kat and Oudheusden 2011, Laskari et al 2016, der Kindere et al 2019). Within the class of Eulerian methods, more sophisticated methods include solvers based on pressure-velocity algorithms from CFD (Gunaydoglug and Kurtulus 2019, Felis-Carrasco et al 2021) 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, Schneider 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, Schneider 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 grid. 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, Sirignano and Spiliopoulos 2018, Raisi et al 2019, Li and Mei 2020), recently popularized as physics-informed neural networks (see Raisi 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 $y = f(u, w)$, linking some input field $u$ to some output field $y$, one seeks to identify the set of parameters $w$ such that $y$ solves a PDE (i.e. minimizes some properly defined residuals). The framework applies equally well to ANN 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, 1990b) and has grown into a mature approach with an extensive literature (see Chen and Tanaka 2002, Chen 2003, Sarler 2005, 2007, Fornberg and Flyer 2015). 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 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 RBFs as tools for approximating a function \( f: \mathbb{R}^d \rightarrow \mathbb{R} \), 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 sections 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:

\[
\varphi_k(x|x_k^*, c_k) = \exp\left(-c_k^2 \|x - x_k^*\|^2\right)
\]

(1)

where \( c_k > 0 \) is the shape parameter and \( x_k^* \in \mathbb{R}^d \) is the collocation point. In this section, we consider both \( c_k \) and \( x_k^* \) as given: their identification is discussed in section 2.4. The basis element \( \varphi_k \) is thus solely function of \( x \) (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:

\[
f(x) \approx \tilde{f}(x) = \sum_{k=1}^{n_c} w_k \varphi_k(x|x_k^*, c_k),
\]

(2)

with the weights \( w_k \) to be identified from data.

In the machine learning literature, equation (2) can be seen as a linear ANN (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 (PINNS) (Raissi et al 2019, Li and Mei 2020). 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 set. In this work, this is a set of \( n_p \) pairs \( \{(x_i, f_i)\}_{i=1,...,n_p} \) with \( x_i = (x_i, y_i, z_i) \) and \( f_i = f(x_i) \), produced, for example, by 3D PTV measurements. The coordinates of the sampling points can be arranged into a matrix \( X \in \mathbb{R}^{n_p \times 3} \) 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 \( \gamma_k(x) \) is added to the basis so that (2) becomes:

\[
f(x) = \sum_{k=1}^{n_c} w_k \gamma_k(x) + \sum_{k=1}^{n_{\gamma}} w_{n_c+k} \varphi_k(x|x_k^*, c_k).
\]

(3)

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

\[
f \approx \tilde{f} = \Phi(X)w,
\]

(4)

where \( f = (f_1, f_2, \ldots, f_{n_p})^T \) is the vector collecting all the training targets, \( w = (w_1, w_2, \ldots, w_{n_c})^T \) is the vector containing the set of weights (regardless of whether these are associated to RBFs or polynomial terms) and we let \( \Phi(X) \in \mathbb{R}^{n_p \times n_b} \) denote the matrix obtained by evaluating the \( n_b \) basis functions at the \( n_p \) training points:

\[
\Phi(X) = 
\begin{pmatrix}
\gamma_1(x_1) & \cdots & \gamma_{n_c}(x_1) & \varphi_1(x_1) & \cdots & \varphi_{n_c}(x_1) \\
\gamma_1(x_2) & \cdots & \gamma_{n_c}(x_2) & \varphi_1(x_2) & \cdots & \varphi_{n_c}(x_2) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\gamma_1(x_{n_p}) & \cdots & \gamma_{n_c}(x_{n_p}) & \varphi_1(x_{n_p}) & \cdots & \varphi_{n_c}(x_{n_p})
\end{pmatrix}.
\]

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

\[
J(w) = \|f - \Phi(X)w\|_2^2,
\]

(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 \( \alpha \|w\|_2^2 \), with penalty \( \alpha \) computed as described in section 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 \( C_1w = 0 \), with \( C_1 \in \mathbb{R}^{n_l \times n_b} \), acting on \( n_l \) points. Thus, an additional regularization term \( \alpha_1 \|C_1w\|_2^2 \) 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 \( C_1w = c_1 \) and the data used at the scope is referred to as training set. In this work, this is a set of \( n_p \) pairs \( \{(x_i, f_i)\}_{i=1,...,n_p} \) with \( x_i = (x_i, y_i, z_i) \) and \( f_i = f(x_i) \), produced, for example, by 3D PTV measurements. The coordinates of the sampling points can be arranged into a matrix \( X \in \mathbb{R}^{n_p \times 3} \) and the treatment that follows holds regardless of whether these points are randomly scattered as in PTV or over a grid as in PIV.
$C_2 w = c_2$, with $C_1 \in \mathbb{R}^{n \times n}$ and $C_2 \in \mathbb{R}^{n \times n}$ linear operators acting on $n_\lambda$ and $n_\lambda$ points respectively, and $c_1 \in \mathbb{R}^{n_\lambda}$ and $c_2 \in \mathbb{R}^{n_\lambda}$. 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:

$$\min J(w)$$

s.t. $C_1 w = c_1$, $C_2 w = c_2$. (7)

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

$$J^*(w, \lambda) = \|f - \Phi(X) w\|_2^2 + \lambda^T (C_1 w - c_1) + \lambda^T (C_2 w - c_2) + \alpha \|C w\|_2^2,$$

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

$$2 (\Phi^T \Phi + \alpha_s C_1^T C_1) w + C_1^T \lambda_1 + C_2^T \lambda_2 = 2 \Phi^T f,$$

$$C_1 w = c_1,$$

$$C_2 w = c_2,$$

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

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} w \\ \lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

with $A = 2 \Phi^T \Phi + 2 \alpha_s C_1^T C_1 \in \mathbb{R}^{n \times n}$, $B = (C_1^T, C_2^T) \in \mathbb{R}^{n \times n}$, $b_1 = 2 \Phi^T f \in \mathbb{R}^n$ and $b_2 = (c_1, c_2)^T \in \mathbb{R}^n$.

The solution of (9) leads to a minimum of $J(w)$ as long as its Hessian $\nabla_w^2 J(w) = A$ is positive semi-definite. Both the problems of velocity fields approximation and meshless pressure computation leads to constrained least squares 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 $x^2 = (x^2, y^2)$. Equation (2) and its matrix form in (4) can be extended to vector fields if the data is reshaped into column vectors. Namely, let $u, v \in \mathbb{R}^{n_p}$ be the vectors collecting all the sampled values of $u_i$ and $v_i$ in the grid points $x_i$ stored in $X \in \mathbb{R}^{n_p \times 2}$. Let $U = (u^\top, v^\top) \in \mathbb{R}^{n_p \times 2}$ be a column vector with the semicolon; denoting vertical concatenation. Similarly, let $w_U = (w_u, w_v) \in \mathbb{R}^{n_p}$ be the column vector concatenating the weights $w_u, w_v \in \mathbb{R}^{n_p}$ for the RBFs expansion of the velocity components $u$ and $v$. Then, the approximation of the velocity field in the available points is:

$$U = \begin{pmatrix} u \\ v \end{pmatrix} \approx U = \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \Phi(X) & 0 \\ 0 & \Phi(X) \end{pmatrix} \times \begin{pmatrix} w_u \\ w_v \end{pmatrix} = \Phi_U(X)w_U,$$

where the same matrix of basis function $\Phi(X)$ is used for both components, assuming that these are available at the same points in $X$. To ease the connection with the previous section, we introduce the basis matrix $\Phi_U(X) \in \mathbb{R}^{2n_p \times 2n_p}$. 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:

$$J(w_U) = \|U - \Phi_U(X)w_U\|_2^2.$$ (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 $x$ ($\partial_x$) and along $y$ ($\partial_y$):

$$\partial_x \varphi_k(x|x^*_k, c_k) = -2c^2_k(x - x^*_k) \varphi_k(x|x^*_k, c_k),$$

$$\partial_y \varphi_k(x|x^*_k, c_k) = -2c^2_k(y - y^*_k) \varphi_k(x|x^*_k, c_k).$$

Therefore, the derivatives of the approximation are:

$$\partial_x u(x_i) \approx \sum_{k=0}^{n_p} w_{u_k} \partial_x \varphi_k(x|x^*_k, c_k) \rightarrow \Phi_x(X)w_u,$$ (13a)

$$\partial_y v(x_i) \approx \sum_{k=0}^{n_p} w_{v_k} \partial_y \varphi_k(x|x^*_k, c_k) \rightarrow \Phi_y(X)w_v,$$ (13b)
where the matrices $\Phi_1(X), \Phi_2(X) \in \mathbb{R}^{n_U \times n_U}$ collect basis functions’ derivatives along their columns:
\[
\Phi_1(X) = [\partial_x \gamma_1, \ldots, \partial_x \gamma_{n_\lambda}, \partial_y \phi_1, \ldots, \partial_y \phi_{n_\lambda}],
\]
\[
\Phi_2(X) = [\partial_x \gamma_1, \ldots, \partial_x \gamma_{n_\lambda}, \partial_y \phi_1, \ldots, \partial_y \phi_{n_\lambda}] .
\tag{14a}
\]

The divergence of the velocity field $\nabla \cdot U(X_V)$ in a set of points $X_V \in \mathbb{R}^{n_V \times 2}$ is approximated by $D_V(X_V) \in \mathbb{R}^{n_V \times 2 \times n_U}$. The divergence free condition applied to the RBF is:
\[
\nabla \cdot (u(X_V) \ \nu(X_V)) \approx (\Phi_1(X_V) \ \Phi_2(X_V)) \times (w_U \ w_v) = D_V(X_V) w_U = 0 ,
\tag{15}
\]
where $\Phi_1(X_V), \Phi_2(X_V) \in \mathbb{R}^{n_V \times n_U}$ are the matrices of basis function derivatives at the points $X_V \in \mathbb{R}^{n_V \times 2}$, and $\theta \in \mathbb{R}^{n_V}$ is the zero vector of appropriate size.

Equation (15) is introduced in (8) as both a penalty ($C_2$) and a constraint ($C_1$), although generally at different points. We use $X_V \in \mathbb{R}^{n_V \times 2}$ 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 $X_D \in \mathbb{R}^{n_D \times 2}$ collect the set of points over which $n_D$ Dirichlet conditions must be enforced, and let $c_D \in \mathbb{R}^{2n_D}$ collect all the corresponding values for both $u$ and $v$ components, vertically concatenated. The associated linear operator $D(X_D) \in \mathbb{R}^{n_D \times 2 \times n_U}$ is:
\[
\begin{pmatrix}
\Phi(X_D) \\
0
\end{pmatrix}
\begin{pmatrix}
w_u \\
w_v
\end{pmatrix} = D(X_D) w_U = c_D .
\tag{16}
\]

Similarly, let $X_N \in \mathbb{R}^{n_N \times 2}$ collect the set of points over which Neumann conditions must be enforced, and let $c_N = (c_{n_U}, c_{n_N}) \in \mathbb{R}^{2n_N}$ collect all the corresponding values. The associated linear operator $N_N(X_N) \in \mathbb{R}^{2 n_N \times n_U}$ is:
\[
\begin{pmatrix}
\Phi_n(X_N) \\
0
\end{pmatrix}
\begin{pmatrix}
w_u \\
w_v
\end{pmatrix} = N_N(X_N) w_U = c_N ,
\tag{17}
\]
where $\Phi_n(X_N) \in \mathbb{R}^{n_N \times n_U}$ 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 $n = (n_u, n_v)$ the normal vector at a given location and $n_i(X_N), n_j(X_N) \in \mathbb{R}^{n_N}$ the collections of normal vectors at the points where the condition is to be imposed, we have:
\[
\Phi_n(X_N) = \Phi_1(X_N) \odot n_i(X_N) + \Phi_2(X_N) \odot n_j(X_N) ,
\tag{18}
\]
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:
\[
\begin{pmatrix}
A_U \\
B_U
\end{pmatrix}
\begin{pmatrix}
w_U \\
\lambda_U
\end{pmatrix} = \begin{pmatrix} b_{U1} \\
b_{U2}
\end{pmatrix} ,
\tag{19}
\]
where $\lambda_U = (\lambda \gamma, \lambda \delta, \lambda \xi)^T \in \mathbb{R}^{n_\lambda}$ is the vector of the Lagrange multipliers for all constraints, i.e. $n_\lambda = n_\gamma + 2n_D + 2n_N$, and:
\[
A_U = 2\Phi_1^T \Phi_U + 2\alpha \nabla \Phi_1^T D_V D_V \in \mathbb{R}^{2 n_U \times 2 n_U},
\tag{20a}
\]
\[
B_U = (D_V^T, D_V^T, N_N^T) \in \mathbb{R}^{2 n_U \times n_N}
\tag{20b}
\]
\[
b_{U1} = 2\Phi_1^T U \in \mathbb{R}^{2 n_U},
\tag{20c}
\]
\[
b_{U2} = (0, c_D, c_N)^T \in \mathbb{R}^{n_N}
\tag{20d}
\]
and having used the short hand notation $\Phi$ for $\Phi(X), N_N$ for $N_N(X_N)$, etc. It is worth highlighting that one has $D_V(X)$ in (20a) but $D_V(X_V)$ 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 $X_N$ and $X_V$. However, it is possible to have both Dirichlet and divergence-free conditions on the same points. In this case, the common $n_\lambda$ points will be included in both $X_D$ and $X_V$ and thus in the counting of both conditions (i.e. to the counting $n_\nu$ and $n_\delta$).

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:
\[
\Delta p = -\rho \nabla \cdot (U \cdot \nabla U) ,
\tag{21}
\]
where $\Delta p$ is the Laplacian of the pressure field $p$, $\rho$ is the fluid density, $U$ is the velocity field and $\nabla U$ 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 (Gurka et al. 1999, van Oudheusden 2013, der Kindere et al. 2019). 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 section 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 $U = (u,v)$, the right hand side of (21) reads:
\[
\nabla \cdot (U \cdot \nabla U) = (\partial_x u)^2 + 2\partial_x v \partial_y u + (\partial_y v)^2 .
\tag{22}
\]
Once the RBF approximation of the velocity field is computed as described in section 2, the evaluation of (22) on a set of $X \in \mathbb{R}^{n_X \times 2}$ points can be approximated as:

$$\nabla \cdot (U(X) \cdot \nabla U(X)) \approx (\Phi^T w_u)^2 + 2 \Phi^T w_u \odot \Phi^T w_r + (\Phi^T w_r)^2,$$

where the gradients 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 $s \in \mathbb{R}^{n_p}$.

Writing the pressure field at every location $X$ as in (4):

$$p \approx \tilde{p} = \Phi(X) w_p$$

with $w_p \in \mathbb{R}^{n_p}$ the weights for the pressure, the approximation of the Laplace operator is:

$$\Delta p \approx L(X) w_p,$$

where $L(X) = [\Delta \gamma_1(X), \ldots, \Delta \gamma_{n_r}(X), \Delta \varphi_1(X), \ldots, \Delta \varphi_{n_r}(X)] \in \mathbb{R}^{n_p \times n_r}$ is the matrix collecting the Laplacian of every element of the basis functions along its columns. For the Gaussian RBFs, at every point $x_i \in X$ this reads:

$$\Delta \varphi_k(x_i) = 2\sigma_k^2 [2\sigma_k^2 (x_i - x_i)^2 - 1] \varphi_k(x_i),$$

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

The RBF approximation of (21) is thus $L(X) w_p = s$, and the cost function to minimize is:

$$J(w_p) = ||L(X) w_p - s||_2^2.$$

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

$$\Phi(X_D) w_p = c_D$$

$$\Phi_n(X_N) w_p = c_N.$$

In the pressure integration problem for a steady flow, the pressure gradient required to impose Neumann conditions ($c_N$ in (28b)) is computed from Navier–Stokes equation (Gurka et al 1999):

$$\nabla p \cdot n = \partial_n p = (U \cdot \nabla U + \mu \Delta U + \rho g) \cdot n,$$

where $n$ is the vector normal to the surface at hand, $\mu$ is the dynamic viscosity and $g$ 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 incompressible 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 section 2.1 reads:

$$\begin{pmatrix} A_p & B_p \\ B_p^T & 0 \end{pmatrix} \begin{pmatrix} w_p \\ \lambda_p \end{pmatrix} = \begin{pmatrix} b_p1 \\ b_p2 \end{pmatrix},$$

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

$$A_p = 2L^T L \in \mathbb{R}^{n_p \times n_p};$$

$$B_p = (D^T N^T) \in \mathbb{R}^{n_p \times n_N};$$

$$b_p1 = 2L^T s \in \mathbb{R}^{n_p};$$

$$b_p2 = (0, c_D, c_N)^T \mathcal{R}^{n_N}.$$

The notation from the previous sections is used.

2.4. Clustering to collocate

The optimal choice of the collocation points $x_i^*$ and shape parameters $c_k$ has been investigated by several authors (see Hardy 1971, Franke 1982, 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 $A$. 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 Hardy 1971, Franke 1982). A more complex formulation, proposed by Rippa (1999) and extended by Karri et al (2009), consists in minimizing a cost function $J(x_i^*, c_k)$ that measures the accuracy of the RBF approximation. Although this leads to a nonlinear optimization problem, the availability of analytic gradients $\partial_k J$ and $\partial_k J$ 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 $\Omega_t$ the area within which a RBF is $\varphi_k (x_i^*, c_k) > \varepsilon_i$, with $\varepsilon_i$ a user-defined threshold, we compute $x_i^*$ and $c_k$ such that an average of $n_K$ particles fall inside $\Omega_t$. 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 $x_i$ (here the coordinates at which the velocity vectors are available) into $K$ clusters $C_k$ with $k = 1, \ldots, K$. Each
cluster contains $|C_k|$ particles and is such that $\sum_k |C_k| = n_p$ and $C_k \cap C_l = \emptyset \forall i \neq j$. 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 $\mu_k$ denote the centroid of cluster $C_k$, the clustering problem aims at minimizing the intra-cluster variance:

$$f(\mu_1, \ldots, \mu_n) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2 \quad \text{with} \quad \mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i.$$  

(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 $n_1 = 10$, 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 $n_j$ the number of levels and with $K_j$ the number of clusters at the $j$-th level, with $j = 1, \ldots, n_j$, one has:

$$K_j = \frac{n_p}{\prod_{a=1}^{n_j} n_k^{(j)}},$$  

(33)

where $n_k^{(j)}$ is the expected number of points per cluster in the $n$th level. In a compact notation, we use the vector $\mathbf{n_k} = [n_k^{(1)}, \ldots, n_k^{(n_j)}] \in \mathbb{R}$. All the centroids are taken as collocation points regardless of their level, i.e. $x_k^j = \mu_k$. The shape factors are computed at each level in such a way that $\varphi_k(\mu_k^{(j)}, \epsilon)$, where $\mu_k^{(j)}$ is the collocation point of interest at level $j$ and $\mu_k^{(j)}$ is the nearest collocation point at the same level. Therefore, one has $c_k^{(j)} = \sqrt{-\ln(\epsilon)/\min\left(\|\mu_i^{(j)} - \mu_k^{(j)}\|\right)}$ with $k \neq j$.

Figure 2 illustrates the result of the clustering for the second test case analyzed in this work (see section 3.3). The figure on the top shows the result of the clustering considering $n_p = 18755$ points and $\mathbf{n_k} = [4, 8, 8]$, i.e. having two sets with $K_1 = n_p/8 \approx 2417$ and $K_2 = n_p/(8 \cdot 4) = 604$ clusters. The top figures show 300 randomly selected RBFs, with the green circles denoting the regions $\Omega_k$ such that $c_k > 0.85$. The bottom figure shows a closed view near the cylinder, together with the green circle denoting the region $\Omega_k$ of about 30 randomly chosen RBFs. 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 $c_i$ from the nearest neighbor 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 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 $B \in \mathbb{R}^{m \times n}$ is full rank and the reduced Hessian $Z^T AZ$, with $Z \in \mathbb{R}^{n \times (n_r - n_k)}$, a basis for the nullspace of $B^T$ (i.e. $B^T Z = 0$), 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 $B$ to remove redundant constraints, but this is generally expensive in 3D problems where $B$ is large.

Therefore, the proposed approach relies on a direct heuristic method that has the guarantee 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 $A$, from the first equation in (9) one gets:

$$ w = A^{-1} (b_1 - B \lambda) . $$

Inserting this in the second equation, rearranging terms and defining $M = B^T A^{-1} B$, $b_2^* = B^T A^{-1} b_1 - b_2$ and $b_1^* = b_1 - B \lambda$, the linear system in (9) is decoupled in the two lower-dimensional systems:

$$ M \lambda = b_2^* $$

(35a)

$$ A w = b_1^* , $$

(35b)

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

Let $\alpha_A$ and $\alpha_M$ denote the regularization parameters for $A$ and $M$, such that the regularization of these matrices reads $A \leftarrow A + \alpha_A I$ and $M \leftarrow M + \alpha_M I$, with $I$ 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 these 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 $A$, for instance, letting $\kappa(A, 2)$ denote the spectral condition number, $\lambda_m(A)$ the smallest eigenvalue of $A$, $\lambda_M(A)$ the largest eigenvalue of $A$, and assuming $\alpha_A \approx \lambda_m(A)$, one has (see, for example Rannacher 2018)

$$ \kappa(A, 2) = \frac{\lambda_M(A)}{\lambda_m(A)} \approx \frac{\lambda_M(A)}{\alpha_A} \leq \sqrt{\pi(n)} |A|_\infty \rightarrow \alpha_A $$

(36)

$$ \leq \sqrt{\pi(n)} |A|_\infty = \text{tol} \sqrt{\pi(n)} |A|_\infty $$

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

$$ M = B^T A^{-1} B = B^T (L_A^{-1})^T L_A^{-1} B = R^T R. $$

(37)

We remark that the inversion of the lower triangular matrix $L_A$ is only formal as $R$ is computed by solving triangular systems, each one requiring $\sim \frac{n^3}{6}$ operations.

While the Cholesky decomposition in (37) is the most expensive operation (with an operation count of $1/6n^3$ of the numerical approach, the computed factors serve three purposes. Indeed, besides allowing for the rapid computation of $M = B^T A^{-1} B$ following equation (37), it can also be exploited in the computation of the right-hand-side $b_2^*$:

$$ b_2^* = B^T A^{-1} b_1 - b_2 = B^T (L_A^{-1})^T L_A^{-1} b_1 - b_2 = R^T L_A^{-1} b_1 - b_2, $$

(38)

thus only requiring one triangular solve and a matrix-vector multiplication. Furthermore, once the solution of the system $M \lambda = b_2^*$ 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.

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 de Kat and Oudheusden 2011, Azijli et al. 2016, McClure and Yarusevych 2017a), characterized by smooth gradients in both velocity and pressure fields.
is clearly unrealistic, but it allows to showcase the robustness of the algorithm against errors. 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.

The Navier–Stokes equations in polar coordinate give
\[
p(r) = -\frac{1}{2\rho^2 u_0^2} - \frac{\rho \Gamma^2}{4 \pi^2 e_0} \left[ E_1 \left( \frac{r^2}{2e_0} \right) - E_1 \left( \frac{2r^2}{c_0} \right) \right],
\]
where \( E_1(x) \) is the exponential integral:
\[
E_1(x) = \int_{x}^{\infty} \frac{e^{-t}}{t} dt.
\]

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 (40). Most of the particles are in a region of nearly uniform pressure and most of the pressure variation is located at approximately \( r < 0.2 \).

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 Schäfer et al 1996, John 2002) 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 PINNS, a valid alternative to the

\[ U_r = 0 \quad \text{and} \quad U_\theta = \frac{\Gamma}{2\pi r} \left( 1 - e^{-r^2/c_0} \right), \tag{39} \]

where \( \Gamma \) is the circulation, \( c_0 = r_\gamma^2/\gamma \), with \( r_\gamma \) the radial distance from the vortex center where the largest velocity is reached, and \( \gamma = 1.25643 \). This is a Lamb-Oseen vortex with \( 4\nu t = c_0 \).

The domain of interest is \( x \in [-0.5, 0.5] \times [-0.5, 0.5] \) and is taken as dimensionless. Assuming that this is mapped onto a sensor with 1024 \( \times \) 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 (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 \( U_c(x_i) = U(x_i) + \rho \lambda [\delta(x_i)] \), where \( U(x_i) \) is the ideal velocity field from (39), \( \rho \) is the noise level and \( \lambda \) 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.
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.

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.
Therefore, the dimensionless stagnation pressure becomes to their dimensionless counterpart as reference in the far field, is

derivatives along the (coordinate in the (plane, hence the symmetry of the flow is such that all derivatives along \( \phi \) vanish. The pressure field, taking \( p_\infty = 0 \) as reference in the far field, is

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

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

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 \( U_\infty \) past a sphere of radius \( R \). In spherical coordinates \( (r = (r, \theta, \phi)) \), the relevant components of the velocity field \( \hat{v}(v_r, v_\theta, 0) \) reads (see for example Bird et al. 2006):

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

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

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

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

Figure 8(a) shows the contour-plot of the dimensionless pressure field \( \hat{p} \) (on the half side) and the dimensionless velocity magnitude \( \hat{v} \) (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 \( \phi \in [0, \pi/2] \) 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 \( \sim 3 n_\theta \times 3 n_\theta \), it is easy to see that the memory demands grow considerably even if the proposed clustering algorithm is such that \( n_\theta \approx n_p/2 \).

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 simplify 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
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 \(\theta \in [-\pi, \pi]\).

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 \(n_\lambda\). The letter V denotes the velocity regression problem, the letter P denotes the pressure integration problem.

|             | \(n_p\) \(n_b\) | \(n_\lambda\) |
|-------------|------------------|--------------|
| V/P         | V/P              | V/P          |
| Case 1: Vortex flow | 5242 | 1158 | 196 |
| Case 2: 2D cylinder wake | 18755 | 4203 | 1348 |
| Case 3: 3D stokes flow | 18297 | 10381 | 13323 |

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.

|             | Time for clustering | Time to prepare A | Timing to prepare M (lines 4 and 5) | Timing to compute \(\lambda\) (line 7) | Timing to compute \(w\) (Line 8) |
|-------------|---------------------|-------------------|------------------------------------|-------------------------------------|---------------------------------|
|             | V/P                 | V/P               | V/P                                | V/P                                 | V/P                             |
| Case 1      | 2.2                 | 0.86              | 0.66                               | 0.24                                | 0.002                           |
| Case 2      | 4.3                 | 2.61              | 1.94                               | 2.19                                | 0.16                            |
| Case 3      | 12.9                | 320               | 90.3                               | 633                                 | 16.7                            |

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.

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 (3.6GHz). The implementations were coded in Python 3.7. It is important to note that the current implementation is not yet fully 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 problem.
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 s for the velocity regression and 2.2 s for the pressure computation while for the most expensive 3D test case considered in this work (with \( N_p = 18297 \) particles with \( N_p = 10381 \) RBFs in 3D) the timing goes to approximately 15 min for the velocity regression and 2 min for the pressure integration. The same 3D case (including both velocity regression and pressure computation) runs in about 4 min 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 \( n_\partial = n_\gamma = 0 \) 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 \( n_q = 200 \) and \( n_q = 1 \) in (31). The cluster-level vector is \( \epsilon_M = [6, 10] \) with \( \epsilon_1 = 0.88 \) and the maximum capping taken as \( c_2 = 0.7 \) (see section 2.4).

To explore the impact of seeding concentration and noise, a grid with \( N_p = [0.003, 0.005] \) and \( q = [0.2, 0.3] \) (see 3.1 for definitions) with \( 20 \times 20 \) elements was analyzed. Figure 9 shows the \( l_2 \) error for the velocity \( (E_U) \) and the pressure \( (E_P) \) computations, defined as:

\[
E_U = \frac{||u(X) - \Phi(X)w_u||_2 + ||v(X) - \Phi(X)w_v||_2}{||u(X)||_2 + ||v(X)||_2}
\]

and

\[
E_P = \frac{||p(X) - \Phi(X)w_p||_2}{||p(X)||_2}
\]

(45)

where \( || \cdot ||_2 \) is the \( l_2 \) norm of a vector, \((u(X), v(X))\) is the theoretical velocity field and \( p(X) \) is the theoretical pressure field on the scattered points \( X \), 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 10 illustrates the robustness of the method in the gradient reconstruction. Figure 10(a) shows the pressure gradient along the \( x \) direction by differentiating (40); figure 10(b) shows the same quantity computed analytically from the RBF reconstruction. This test case considers \( N_p = 0.004 \) with a noise level of \( q = 25\% \). The agreement is remarkable, as also shown in figure 10(c), which shows the profile of the pressure gradient from the analytic formula (blue dash-dotted line),
Figure 10. Comparison of the pressure gradient reconstruction with the background truth. Figure (a) shows the normalized pressure gradient $\partial_x pL/p_{peak}$, scaled with respect to $p_{peak}/L$, while figure (b) shows the results obtained by differentiating the RBF solution. Figure (c) shows the pressure gradient profile along $y = 0$. 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 appear 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).

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 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 and at the inlet. Therefore, we have $n_D = 600$ points with both conditions and a total of $n_{\lambda} = n_{\nabla} + 2 n_D = 1948$ 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...
Meas. Sci. Technol. 33 (2022) 094005
P Sperotto et al

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 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. 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.

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 section 2.4.). Figures 13(a) and (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 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 equation...
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 \((q = 0.1)\). 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 \(\theta\), velocity and pressure values are taken as the ones lying within the two planes, parallel to the radial direction and \(\pm 2\) mm 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 show 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 show 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 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. 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 coordinates, comparing the results of the pressure integration (black continuous line) with the 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 \(n_K = [6, 10, 20]\) results in \(n_p = 10381\) 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 \(U = 0\) at \(\hat{r} = 0.5\). This results in \(n_D = 2111\) and \(n_V = 6990\), thus a total of \(n_\lambda = 3 n_p + n_V = 13323\) constraints for the velocity regression (cf table 1). In addition to these, a large...
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 $n_p = 18755$).

penalty of $\alpha_c = 25$ 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 (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 $\theta = 0^\circ$ and $\varphi = 0^\circ$, i.e. on an equatorial plane (taking the south pole on the 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 Pan et al (2016, 2018), Faiella et al (2021), 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
Figure 15. Test case 2. Velocity profiles (second row) and pressure profiles (third row) extracted at three planes with $\theta = 0^\circ$, $90^\circ$, $180^\circ$. 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.

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).

Figure 18. Test case 3. Local absolute error distribution for the velocity regression. Figure (a) shows the results for the noiseless ($q = 0$) test case, figure (b) for the noisy ($q = 0.05$) test case. Figures (c) and (d) show the corresponding distribution of error for the flow divergence.
Figure 19. Test case 3. Velocity profile at $\theta = 0^\circ$ and $\varphi = 0^\circ$, from $\hat{r} = 0.5$, for the noiseless (a) and the noisy (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. Figure (a) shows the results for the noiseless ($q = 0$) test case, figure (b) for the noisy ($q = 0.05$) test case.

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.

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 figure 18), namely on the surface wall for the noiseless test case (figure 20(a)) and on the outer surface for the noisy test case (figure 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 $\theta$ because of the perfect axial symmetry of the regression. The impact of the noise appears negligible.

Finally, figures 21(c) and (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 21. Test case 3. Figures (a) and (b) show the pressure distribution along the sphere’s wall for the noiseless and the noisy case respectively. Similarly, (c) and (d) show the wall-normal pressure gradient.

5. Conclusions and perspectives

We presented a constrained 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 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’ regression, similar to what is done in ‘physics-informed’ 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.

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.

**Data availability statement**

All data that support the findings of this study are included within the article (and any supplementary files).

**Appendix A. Velocity approximations in 3D**

In a 3D problem, the collected velocity field can be arranged in the column vector \( \mathbf{U} = (u; v; w) \in \mathbb{R}^{3n} \), where \( u, v, w \in \mathbb{R}^{n} \) collects the entries of the three velocity components \( U_i = (u_i, v_i, w_i) \). In this section, the symbol \( w \) 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 2D case to \( x_i^2 = (x_i^x, y_i^x, z_i^x) \). Equation (10) is extended to:

\[
\bar{\mathbf{U}} = \begin{pmatrix}
\bar{u} \\
\bar{v} \\
\bar{w}
\end{pmatrix} = \begin{pmatrix}
\Phi(X) & 0 & 0 \\
0 & \Phi(X) & 0 \\
0 & 0 & \Phi(X)
\end{pmatrix}
\begin{pmatrix}
w_u \\
w_v \\
w_w
\end{pmatrix} = \Phi_u(X)w_U, \tag{46}
\]

where \( w_u \in \mathbb{R}^{n} \) are the weights for the third velocity component. The involved matrices and arrays increase their size accordingly, and are thus \( \Phi_u(x_i) \in \mathbb{R}^{3n \times 3n} \) and \( \mathbf{w}_U = (w_u; w_v; w_w) \in \mathbb{R}^{3n} \). The RBF derivatives in the \( z \)-direction can be added to the two dimensional set (12):

\[
\partial_3 \varphi_k(x | x_i^x, c_k) = -2 c_k^2 (z - z_k^i) \varphi_k(x | x_i^x, c_k). \tag{47}
\]

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

\[
\partial_3 w(X) \equiv \sum_{k=0}^{n_i} w_{uk} \partial_3 \varphi_k(X | x_i^x, c_k) = \Phi_u(X)w_u, \tag{48}
\]

where \( \Phi_u(X) \in \mathbb{R}^{3n \times 3n} \) is defined as in (14). In line with the notation in section 2, the divergence operator becomes:

\[
\nabla \cdot \begin{pmatrix}
u(X) \\
w(X)
\end{pmatrix} \approx \begin{pmatrix}
\Phi_x(X) & \Phi_y(X) & \Phi_z(X)
\end{pmatrix} \begin{pmatrix}
w_u \\
w_v \\
w_w
\end{pmatrix} = \mathbf{D}_\nabla(X)w_U = 0, \tag{49}
\]

where \( X \in \mathbb{R}^{3n \times 3n} \) and \( \mathbf{D}_\nabla \in \mathbb{R}^{n \times 3n} \). Denoting \( \mathbf{n} = (n_x, n_y, n_z) \) as the normal vector to a surface and \( \mathbf{n}_\lambda(X_N) \in \mathbb{R}^{n} \) with \( X_N \in \mathbb{R}^{n \times 3n} \), the directional derivative is:

\[
\Phi_n(X_N) = \Phi_x(X_N) \odot n_x(X_N) + \Phi_y(X_N) \odot n_y(X_N) + \Phi_z(X_N) \odot n_z(X_N). \tag{50}
\]

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

\[
\begin{pmatrix}
\Phi_u(X_N) & 0 & 0 \\
0 & \Phi_n(X_N) & 0 \\
0 & 0 & \Phi_u(X_N)
\end{pmatrix}
\begin{pmatrix}
w_u \\
w_v \\
w_w
\end{pmatrix} = N_u(X_N)w_U = c_N,
\]

while the linear constraints on the velocity field (cf equation (16) for the 2D) become:

\[
\begin{pmatrix}
\Phi(D) & 0 & 0 \\
0 & \Phi(X) & 0 \\
0 & 0 & \Phi(D)
\end{pmatrix}
\begin{pmatrix}
w_u \\
w_v \\
w_w
\end{pmatrix} = D(D)w_U = c_D, \tag{51}
\]

where \( c_D = (c_{Dx}; c_Dy; c_Dz) \in \mathbb{R}^{n} \), with \( c_{Dx} \in \mathbb{R}^{n} \) the values constraining the third velocity component and \( X_D \in \mathbb{R}^{n \times 3n} \) 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:

\[
\mathbf{A}_U = 2\Phi^T_u \Phi_u + 2\alpha \mathbf{D}^T \mathbf{D} \in \mathbb{R}^{3n \times 3n}, \tag{52a}
\]

\[
\mathbf{B}_U = (D^T \mathbf{D}^T \mathbf{N}_u^T) \in \mathbb{R}^{3n \times n_\lambda}, \tag{52b}
\]

\[
\mathbf{b}_{U1} = 2\Phi^T_u \mathbf{U} \in \mathbb{R}^{3n}, \tag{52c}
\]

\[
\mathbf{b}_{U2} = (0, c_D, c_N^T)^T \in \mathbb{R}^{n_\lambda}, \tag{52d}
\]

where \( n_\lambda = n_\tau + 3n_D + 3n_N \).

**Appendix 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:

\[
\nabla \cdot (U \cdot \mathbf{n}) = \nabla u^2 + \nabla v^2 + \nabla w^2 + 2\nabla u \nabla v + 2\nabla w \nabla u + 2\nabla v \nabla w + 2\nabla u \nabla v + 2\nabla v \nabla w, \tag{53}
\]

which can then be written with the same formalism as (23). Equation (29) requires particular attention as in 3D \( \mathbf{n} = (n_x, n_y, n_z) \) 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.

**ORCID ID**

Miguel A Mendez  \textcopyright https://orcid.org/0000-0002-1115-2187
McClure J and Yarusevych S 2017b Optimization of planar PIV-based pressure estimates in laminar and turbulent wakes Exp. Fluids 58 62
McClure J and Yarusevych S 2017c Instantaneous PIV/PTV-based pressure gradient estimation: a framework for error analysis and correction Exp. Fluids 58 92
Neeteson N J and Rival D E 2015 Pressure-field extraction on unstructured flow data using a voronoi tessellation-based networking algorithm: a proof-of-principle study Exp. Fluids 54 44
Nielsen F 2016 Introduction to HPC With MPI for Data Science (Springer International Publishing)
Nocedal J and Wright S 2006 Numerical Optimization 2nd edn (New York: Springer) (https://doi.org/10.1007/978-0-387-40065-5)
Novara M and Scarano F 2013 A particle-tracking approach for accurate material derivative measurements with tomographic PIV Exp. Fluids 54 1584
Pan J P Z Whitehead G R, Truscott T T and Smith B L 2018 Error propagation dynamics of piv-based pressure field calculation (3): what is the minimum resolvable pressure in a reconstructed field? Meas. Sci. Technol. 27 084012
Pirnia A, McClure J, Peterson S D, Helenbrook B T and Erath B D 2020 Estimating pressure fields from planar velocity data around immersed bodies: a finite element approach Exp. Fluids 61 55
Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations J. Comput. Phys. 378 686–707
Rannacher R 2018 Numerical Linear Algebra (Heidelberg University Publishing) (available at: https://heip.uni-heidelberg.de/catalog/book/407?lang=en)
Rao C, Sun H and Liu Y 2020 Physics-informed deep learning for incompressible laminar flows Theor. Appl. Mech. Lett. 10 207–12
Ratz M, Fiorini D, Simonini A, Cierpka C and Mendez M A 2022 Analysis of an unsteady quasi-capillary channel flow with time resolved piv and rbf-based super resolution
Rippa S 1999 An algorithm for selecting a good value for the parameter c in radial basis function interpolation Adv. Comput. Math. 11 193–210
Šarler B 2007 From global to local radial basis function collocation method for transport phenomena Advances in Meshfree Techniques (Netherlands: Springer) pp 257–82
Sarra S A and Cogar S 2017 An examination of evaluation algorithms for the RBF method Eng. Anal. Bound. Elem. 75 36–45
Schäfer M, Turek S, Durst F, Krause E and Rannacher R 1996 Benchmark computations of laminar flow around a cylinder Notes on Numerical Fluid Mechanics (NNFM) (Berlin: Vieweg+Teubner Verlag) pp 547–66
Schanz D, Gesemann S and Schröder A 2016 Shake-the-box: Lagrangian particle tracking at high particle image densities Exp. Fluids 57
Schneider J F G, Pröbsting S, Dwight R P, van Oudheusden B W and Scarano F 2016 Pressure estimation from single-snapshot tomographic PIV in a turbulent boundary layer Exp. Fluids 57 53
Schneider J F G and Scarano F 2016 Dense velocity reconstruction from tomographic PTV with material derivatives Exp. Fluids 57 139
Schwenker F, Kestler H A and Palm G 2001 Three learning phases for radial-basis-function networks Neural Netw. 14 439–58
Sculley D 2010 Web-scale k-means clustering Proc. 19th International Conf. on World Wide Web—WWW ’10 (ACM Press)
Sirignano J and Spiliopoulos K 2018 DGM: a deep learning algorithm for solving partial differential equations J. Comput. Phys. 375 1339–64
Solis-Oba R 2006 Approximation algorithms for the k-median problem Lecture Notes in Computer Science (Berlin: Springer) pp 292–320
Van der Kindere J W, Laskari A, Ganapathisubramani B and de Kat R 2019 Pressure from 2D snapshot PIV Exp. Fluids 58 60 2
van Gent P L et al 2017 Comparative assessment of pressure field reconstructions from particle image velocimetry measurements and lagrangian particle tracking Exp. Fluids 58 33
van Oudheusden B W 2013 PIV-based pressure measurement Meas. Sci. Technol. 24 032001
van Oudheusden B W, Scarano F, Roosenboom E W M, Casimir E W F and Souverein L J 2007 Evaluation of integral forces and pressure fields from planar velocimetry data for incompressible and compressible flows Exp. Fluids 43 153–62
Wang J, Zhang C and Katz J 2019 GPU-based, parallel-line, omni-directional integration of measured pressure gradient field to obtain the 3D pressure distribution Exp. Fluids 60 58