International Journal for Numerical Methods in Fluids

WILEY

RESEARCH ARTICLE OPEN ACCESS

Incremental Model Order Reduction of Smoothed-Particle Hydrodynamic Simulations

Eduardo Di Costanzo[1,2] | Niklas Kühl[3] | Jean-Christophe Marongiu[1] | Thomas Rung[2]

[1]R&D Department, ANDRITZ Hydro, Vevey, Switzerland | [2]Institute for Fluid Dynamics and Ship Theory, Hamburg University of Technology, Hamburg, Germany | [3]Hamburg Ship Model Basin, Hamburg, Germany

Correspondence: Eduardo Di Costanzo (eduardo.dicostanzo@andritz.com)

Received: 27 January 2025 | **Revised:** 5 July 2025 | **Accepted:** 7 August 2025

Funding: The authors received no specific funding for this work.

Keywords: computational fluid dynamics | incremental singular value decomposition | irregular snapshot matrix | Pelton turbine runner | reduced order modeling | smoothed particle hydrodynamics

ABSTRACT

Engineering simulations are usually based on complex, grid-based, or mesh-free methods for solving partial differential equations. The results of these methods cover large fields of physical quantities at very many discrete spatial locations and temporal points. Efficient compression methods can be helpful for processing and reusing such large amounts of data. A compression technique is attractive if it causes only a small additional effort and the loss of information with strong compression is low. The paper presents the development of an incremental singular value decomposition (SVD) strategy for compressing time-dependent particle simulation results. The approach is based on an algorithm that was previously developed for grid-based, regular snapshot data matrices. It is further developed here to process highly irregular data matrices generated by particle simulation methods during simulation. Various aspects important for information loss, computational effort, and storage requirements are discussed, and corresponding solution techniques are investigated. These include the development of an adaptive rank truncation approach, the assessment of imputation strategies to close snapshot matrix gaps caused by temporarily inactive particles, a suggestion for sequencing the data history into temporal windows as well as bundling the SVD updates. The simulation-accompanying method is embedded in a parallel, industrialized smoothed-particle hydrodynamics software and applied to several 2D and 3D test cases. The proposed approach reduces the memory requirement by about 90% and increases the computational effort by about 10%, while preserving the required accuracy. For the final application of a water turbine, the temporal evolution of the force and torque values for the compressed and simulated data is in excellent agreement.

1 | Introduction

Current and future computational engineering must support more than just the mere analysis of a given design. Instead, the procedures should also optimize the design, enhance its robustness against changing operating conditions, and facilitate the reuse of computed (field) data as a database for machine learning of various design-relevant input/output relationships.

The analysis of an engineering design is usually based on complex, grid-based, or mesh-free first-principle simulation methods, the results of which cover large fields of physical quantities in many million (discrete) spatial locations and many thousand temporal points. Processing, that is, reusing such large amounts of data, which typically result from the numerical solution of partial differential equations (PDEs), is a challenge. In this case, data compression of the results, using methods such as proper

orthogonal decomposition (POD), can effectively save storage space for large-scale but low-rank datasets. The POD technique [1] is particularly attractive when it incurs only a small computational overhead and the loss of information for strong compression is minor.

POD is generally considered a model order reduction approach that approximates a given dataset optimally on a low-dimensional basis [2]. Aiming at data compression, the strategy usually involves a singular value decomposition (SVD) of a snapshot data matrix to derive a reduced basis spanned by the POD modes. Due to the often very good approximation by a small number of consecutive initial POD modes, only a few are usually considered at the expense of a small loss of information. Linearity assumptions limit the use of POD strategies. Nevertheless, the solution manifold is often sufficiently well approximated by a low-rank subspace, which recommends the use of the POD for model order reduction of nonlinear systems [3–9]. This is especially true for unsteady systems where the data is generated progressively in time, and the snapshot matrix usually consists of many more discrete spatial points (defining the length of the matrix column) than temporal points (defining the number of matrix columns). SVD procedures are traditionally offline approaches and thus require storing the entire snapshot matrix before performing the SVD, which still demands prohibitive computer memory. To solve this issue, efficient algorithmic alternatives dedicated to the incremental computation of the SVD (iSVD) that accompanies the data growth over time have been developed [10–15].

Focusing on unsteady gradient- and thus adjoint-based optimization methods raises related issues due to the oppositely directed temporal information transport of the primal and adjoint PDEs involved. To obtain the gradient of the objective function with respect to the optimization parameters, one usually has to solve the forward time-dependent primal problem PDEs and store all forward solutions for later use when integrating the backward-directed adjoint PDEs. Naively storing and subsequently reading the forward solutions to and from the disk at runtime is prohibitive in large-scale industrial applications. It calls for an incrementally processed model order reduction during the integration of the primal flow.

This paper investigates iSVD as a data compression technique for particle simulation methods. The goal is to reduce the storage requirements for future time-dependent, PDE-constrained optimization as much as possible while minimizing the required computational overhead. Though incrementally constructed ROMs have recently been used to handle memory limitations for optimization studies [15–20], the iSVD has never been utilized for dynamic meshes and meshless methods such as smoothed particle hydrodynamics (SPH). The challenges and innovations are considerable and relate to applying an iSVD to particle data obtained from massively parallel SPH simulations. Three aspects are particularly important. In contrast to grid-based methods, (a) the spatial positions of the data vary over time; (b) the snapshot matrix is irregularly populated because the number of particles within the domain typically varies over time; and (c) all particles are only active during a partial period when simulating open boundary domains.

The paper is organized as follows: Sections 2 and 3 are devoted to the full-order model (FOM) and the numerical method used to solve the governing equations. Section 4 briefly outlines the model reduction strategy and the iSVD. Section 5 addresses the specifics and challenges of the iSVD in conjunction with an SPH-based snapshot matrix. Verification and validation examples are reported in Section 6, emphasizing accuracy and efficiency aspects for 2D and 3D benchmark flows. They refer to (a) a simple 2D sloshing in a closed box, (b) a 2D impinging jet featuring open boundaries, and (c) an unsteady 3D Pelton turbine runner. Note that although only hydrodynamic applications are shown, the strategy presented here can also be used for other particle simulation applications. Conclusions are drawn in Section 7.

In the remainder of the paper, reference properties are usually marked by an index "0". Field quantities are defined regarding Cartesian coordinates denoted by Greek superscripts, and Latin subscripts distinguish the particles and their spatial position. Moreover, we employ lower- and uppercase bold letters to denote vectors and tensors, respectively.

2 | Governing Equations

The section briefly recalls the governing equations for a weakly compressible, inviscid fluid. The chosen flow model is best suited for the application of particle methods to the pressure-driven flow around a Pelton turbine at high Reynolds numbers, whereby friction losses on the bucket's surfaces, which are responsible for a reduction in the machine's efficiency, are deliberately not taken into account.

A standard compressible formulation for conserving mass, volume, and momentum is employed for an inviscid single-phase flow. With regard to a numerical particle approach, the density ρ_i, velocity $\mathbf{v_i}$, and volume ω_i of a fluid particle currently located in the position $\mathbf{x}_i$ and transported with the velocity $\mathbf{v}_i^0$ follow from the system of advection, space conservation, mass and momentum equations, viz.

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i^0 \tag{1}$$

$$\frac{d\omega_i}{dt} = \omega_i \, \nabla \cdot \mathbf{v}_i^0 \tag{2}$$

$$\frac{d\rho_i}{dt} = -\rho_i \nabla \cdot \mathbf{v}_i \quad \rightarrow \quad \frac{d(\omega_i \rho_i)}{dt} = -\rho_i \omega_i \nabla \cdot (\mathbf{v}_i - \mathbf{v}_i^0) \tag{3}$$

$$\frac{d\mathbf{v}_i}{dt} = -(\mathbf{v}_i - \mathbf{v}_i^0) \cdot \left[\nabla \mathbf{v}_i\right] - \left(\frac{\nabla p}{\rho}\right)_i + \mathbf{g} \quad \rightarrow$$

$$\frac{d(\rho_i \omega_i \mathbf{v_i})}{dt} = -\rho_i \omega_i \nabla \cdot \left[(\mathbf{v}_i - \mathbf{v}_i^0)\mathbf{v}_i\right] - \omega_i \nabla p_i + \rho_i \omega_i \mathbf{g} \tag{4}$$

Here p is the pressure, $\mathbf{g}$ denotes a gravitational body force per unit mass, ∇ is the (left) spatial gradient, and $d\phi/dt$ indicates the temporal changes of the focal particle properties. Equations (1–4) refer to an *Arbitrary Langragian-Eulerian* (ALE) form. For $\mathbf{v}^0 = 0$, the particle does not move, and one obtains the classical Eulerian description with $d/dt \rightarrow \partial/\partial t$. Moving the particle with the flow, that is, $\mathbf{v}^0 = \mathbf{v}$, a Lagrangian description is recovered with $d/dt \rightarrow D/Dt$.

2.1 | Constitutive Equation

Using a weakly compressible fluid model, the pressure is derived from a heuristic polytropic pressure–density relation (cf [21])., aka. Tait's relation,

$$p = p_0\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \quad \text{with} \quad \gamma = 7 \tag{5}$$

where ρ_0 is a reference density, and p_0 is linked to an artificial reference speed of sound using $p_0 = \rho_0 c_0^2/\gamma$. To keep the density variations below 1%, the speed of sound value is usually obtained from an estimated maximum flow speed using $c_0 \geq 10|\mathbf{v}|_{\max}$, which ensures a Mach number $M \leq 0.1$. The fluid considered in this work has a reference density of $\rho_0 = 1000\ \mathrm{kg/m^3}$.

3 | Computational Model

The Lagrangian SPH method displays advantages over classical Eulerian flow simulation methods for problems with substantial temporal changes of the wetted domain. An example is violent-free surface flows, such as those encountered in hydraulic machinery simulations. In these cases, grid-based methods often require dynamic grid refinement and coarsening, which is not trivial in 3D parallel frameworks. On the contrary, SPH approaches are mesh-free and discretize the material, that is, it does not require the use or adaption of the computational grid to a free surface. Nowadays, SPH strategies are successfully applied to a broad range of engineering investigations, including—for example—free-surface flows [22–24], multi-continua applications [25], fluid–structure interaction [26–28], geotechnical [29, 30], energy harvesting from water waves and wind offshore [31, 32], and turbine applications [33, 34]. Recent advancements include addressing volume conservation in sloshing flows and free-surface conditions [35], improving consistency and transport-velocity formulations via reverse kernel gradient correction [36], implicit shifting methods [37], Quasi-Lagrangian formulations, and energy balance techniques with Riemann solvers have also been introduced [38]. Results of the FOM employed in this work were obtained from the in-house Andritz solver, ASPHODEL. This section provides a brief overview of the implemented methods. A more detailed introduction can be found in [39–42].

3.1 | Numerical Method

Equations (1–4) are discretized in space and time and subsequently explicitly integrated in time. For this purpose, the flow field is first discretized spatially by N particles of equal mass m and initially equal density $\rho_i(t_0) = \rho_0$ and volume $\omega_i(t_0) = m/\rho_i(t_0)$.

3.1.1 | Smoothed-Particle Hydrodynamics Approximation

Smooth particle hydrodynamics rely on two fundamental approximation steps: kernel-based mollification and discrete particle approximation of integrals. The starting point is the identity

$$f(\mathbf{x}_i) = \int f(\mathbf{x}_j)\delta_D(\mathbf{x}_i - \mathbf{x}_j)dx_j \tag{6}$$

with $\delta_D(\mathbf{x}_i - \mathbf{x}_j)$ being the Dirac function that vanishes everywhere but for the case $\mathbf{x}_i = \mathbf{x}_j$. Any physical quantity $f(\mathbf{x})$, such as the density or the velocity, is subsequently approximated by the convolution of $f(\mathbf{x})$ with a smooth, kernel function $W(\|\mathbf{x}_i - \mathbf{x}_j\|, h) = W_{ij}$ that mollifies the Dirac function and is numerically integrated with the help of neighboring particles j, that is,

$$\int f(\mathbf{x}_j)W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)dx_j \approx \sum_{j\in D_i} \omega_j f_j W_{ij} \tag{7}$$

The isotropic bell-shaped kernel function W is usually assumed to be positive and symmetric to mimic the properties of the Dirac function. The most crucial feature of W is its compact support D, which is characterized by the (homogeneous) length h. Similarly, the SPH approximation of the gradient $\nabla f(\mathbf{x}_i)$ is derived using integration by parts and the symmetry relation $\nabla_j W_{ij} = -\nabla_i W_{ij}$, which yields

$$\nabla f(\mathbf{x}_i) \approx \sum_{j\in D_i} \omega_j f(\mathbf{x}_j)\nabla_i W_{ij} + \sum_{k\in\partial D_i} \partial\omega_k f(\mathbf{x}_j)\mathbf{n}_k W_{ij} \tag{8}$$

The normal vector and the area of the discretized boundary are denoted by $\mathbf{n}_k$ and $\partial\omega_k$. Due to the compact support of W, the second term in (8) is only nonzero if the kernel support intersects the domain's boundary. Since the kernel function is a smooth analytical function, its gradient can be computed analytically, greatly simplifying the procedure.

The paper adopts an ALE formulation of the SPH method as initially suggested in [43] and described in [42]. The corresponding spatially discretized version of Equations (1–4) reads

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i^0 \tag{9}$$

$$\frac{d\omega_i}{dt} = \omega_i \sum_{j\in D_i} \omega_j \left[\mathbf{v}_j^0 - \mathbf{v}_i^0\right] \cdot \boldsymbol{\nabla}_i W_{ij} \tag{10}$$

$$\frac{d(\omega_i\rho_i)}{dt} = -\omega_i \sum_{j\in D_i} 2\omega_j \rho_{ij}^E \left[\mathbf{v}_{ij}^E - \mathbf{v}_{ij}^0\right] \cdot \boldsymbol{\nabla}_i W_{ij} \tag{11}$$

$$\frac{d(\omega_i\rho_i\mathbf{v}_i)}{dt} = \omega_i\rho_i\mathbf{g} - \omega_i \sum_{j\in D_i} 2\omega_j \left\{\rho_{ij}^E \mathbf{v}_{ij}^E \otimes [\mathbf{v}_{ij}^E - \mathbf{v}_{ij}^0] + p_{ij}^E \mathbf{I}\right\} \cdot \boldsymbol{\nabla}_i W_{ij} \tag{12}$$

For simplicity, the system of Equations (9–12) is presented without the inclusion of boundary terms in Equation (8). Mind that the space conservation law (10) has been augmented by the subtraction of a term proportional to $\sum \boldsymbol{\nabla}_i W_{ij}$ on the right-hand side (r.h.s.), which vanishes in continuous space, to improve the consistency of the discrete model. Similarly, the continuity Equation (11) and the momentum Equation (12) have been augmented by the addition of a term proportional to $\sum \boldsymbol{\nabla}_i W_{ij}$ on the r.h.s., to support the conservation on the particle level. The superscript E indicates interaction values between pairs of neighboring particles obtained by the solution of a Riemann problem, viz.

$$\mathbf{v}_{ij}^E = \frac{\mathbf{v}_i + \mathbf{v}_j}{2} + \frac{p_j - p_i}{2\overline{\rho}_{ij}\overline{c}_{ij}} \tag{13}$$

$$p_{ij}^E = \frac{p_i + p_j}{2} + \frac{\overline{\rho}_{ij}\overline{c}_{ij}(\mathbf{v}_j - \mathbf{v}_i)}{2} \tag{14}$$

where the double subscripts in the expressions $\overline{\rho}_{ij}$ and $\overline{c}_{ij}$ refer to averaged values of the density and speed of sound of the particles i and j. The interface velocity between these particles reads $\mathbf{v}_{ij}^0 = 0.5(\mathbf{v}_i^0 + \mathbf{v}_j^0)$. The present study employs a constant speed of sound and a Wendland C^4 kernel [44], viz.

$$W(\mathbf{x}, h) = \frac{\sigma}{h^d}\theta\left(\frac{\|\mathbf{x}\|}{2h}\right) \tag{15}$$

$$\theta(q) = (1-q)_+^5(8q^2 + 5q + 1) \tag{16}$$

$$x_+^n = \begin{cases} x^n & \text{if } x > 0, \\ 0 & \text{if } x \leq 0 \end{cases} \tag{17}$$

Here, $W(\mathbf{x}, h)$ represents the smoothing kernel function, $\mathbf{x}$ is the distance vector between two particles, and h is the smoothing length. The parameter d represents the number of spatial dimensions in the simulation, $\|\mathbf{x}\|$ denotes the Euclidean norm of the distance vector $\mathbf{x}$. The function $\theta(q)$ defines the shape of the kernel as a function of the normalized distance $q = \|\mathbf{x}\|/(2h)$. The subscript $()_+$ represents the positive part, meaning that the function goes to zero if the input is negative. The normalization factor σ is given by $\sigma = \frac{3}{4\pi}$ in 2D and $\sigma = \frac{165}{256\pi}$ in 3D.

3.1.2 | Temporal Discretization

The system (9–12) is discretized by adaptive time steps and explicitly integrated in time. In the scope of the present paper, we employ a third-order Runge–Kutta scheme and adjust the time step to comply with

$$\Delta t = K_{\text{CFL}} \min_{i \in D} \frac{h_i}{\|\mathbf{v}_i\| + c_i} \tag{18}$$

where K_{CFL} denotes the maximum CFL number assigned to $K_{\text{CFL}} = 0.5$ in the current 2D studies and $K_{\text{CFL}} = 0.3$ in the 3D application.

3.1.3 | Initial and Boundary Conditions

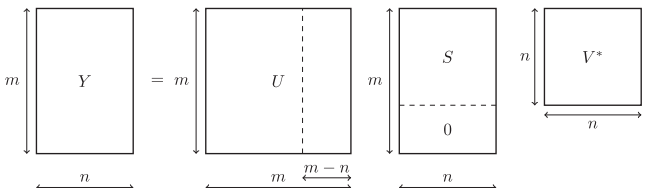
The initially realized particle distance reads Δx and is usually related to the smoothing length of a kernel function h. The present studies are based upon $h/\Delta x = 1.2$, which works well in practice, cf. [41]. The boundary fluxes are computed with partial Riemann solvers for wall boundaries, while open boundaries are managed according to a nonreflecting characteristic boundary condition derived from [45].

4 | Model Reduction Strategy

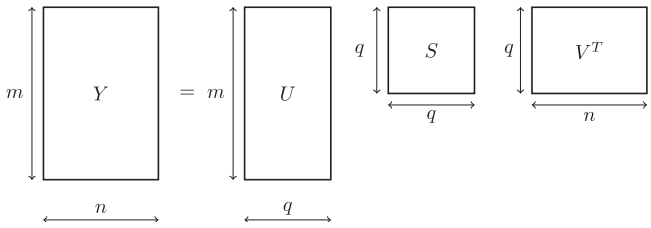
4.1 | Singular Value Decomposition

The *singular value decomposition* (SVD) is a unique and guaranteed to exist matrix factorization, which can provide the best low-rank approximation of a matrix in the least-squares sense [46]. A matrix $Y \in \mathbb{C}^{m \times n}$ is decomposed into two orthogonal matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$, along with a diagonal matrix $S = diag(s_i) \in \mathbb{R}^{m \times n}$, such that $Y = USV^*$, where $*$ is used to indicate the conjugate transpose. The columns of U are called left-singular vectors or modes and describe patterns in the original data. Additionally, these columns are arranged based on their ability to capture the variance in the columns of Y. The matrix S contains the singular values of Y, which are nonnegative and arranged in decreasing order of magnitude. The magnitude of these singular values provides a measure of the relative importance of each mode. The columns of V are called right-singular vectors. These vectors represent how the modes combine, scaled by the corresponding singular values, to reconstruct the columns of Y. The singular values of Y are also the square roots of the eigenvalues of the correlation matrix Y^*Y (or YY^*), while the singular vectors U (or V) are its eigenvectors. This explains why the columns of U (V) are ordered by how much correlation they capture in the columns (rows) of Y [47, 48].

There can be at most $rank(Y) = r \leq min(m, n)$ nonzero singular values. In the case where $m > n$ and Y is full rank (i.e., $\text{rank}(Y) = n$), the decomposition can be shown as



By excluding the rows of S that contain only zeroes and the corresponding columns of U, one obtains the reduced SVD (also called thin SVD or economy-size SVD). For the rest of this work, it is assumed that data are real numbers arranged in matrices where $m > n$. Only the necessary components of the SVD are kept, hence: $Y = USV^T$, $Y \in \mathbb{R}^{m \times n}$, $U \in \mathbb{R}^{m \times n}$, $S \in \mathbb{R}^{n \times n}$, and $V \in \mathbb{R}^{n \times n}$. Thanks to the optimal truncation properties of the SVD, further reduction is possible by neglecting the smallest nonzero singular values according to a desired criterion and only keeping the q most significant singular values. In this case, the SVD is known as truncated SVD: $U \in \mathbb{R}^{m \times q}$, $S \in \mathbb{R}^{q \times q}$, and $V \in \mathbb{R}^{n \times q}$.



The truncated SVD is the starting point of many algorithms utilized for dimensionality reduction, data compression, noise reduction, solving systems of algebraic equations, inverting rank-deficient matrices, and more [47–51]. Nevertheless, since the degree of achievable compression depends on the specific data, no a-priori criteria are usually available to decide how many modes are necessary to keep. Commonly used methods for truncation include setting a hard threshold for singular values, retaining a specified percentage of the total energy, or analyzing the singular values to identify an "elbow" or "knee" in their distribution [47, 52, 53]. It was suggested in [15] that ignoring the first o singular values might be beneficial while computing the retained energy to avoid machine accuracy issues.

4.2 | Incremental Singular Value Decomposition

Typical SVD reduction methods require the complete flow field upfront, leading to significant memory consumption. Incremental SVD approaches have been developed, addressing this challenge and allowing the SVD to be updated incrementally with rank-b modifications. In this work, only the additive modification to add columns to Y is considered. A brief summary of this specific incremental SVD approach is provided. For further details, the reader is referred to [10, 11, 15].

Let us assume that $Y = USV^T$ is known, and that new data becomes available. The original matrix Y and the updated matrix $\tilde{Y}$ are given by Equations (19) and (20), respectively. The objective is to compute the SVD of $\tilde{Y}$, without having to reconstruct Y first.

$$Y = \begin{bmatrix} \mathbf{y_{1,1}} & \cdots & \mathbf{y_{1,n}} \\ \vdots & \ddots & \vdots \\ \mathbf{y_{m,1}} & \cdots & \mathbf{y_{m,n}} \end{bmatrix} \tag{19}$$

$$\tilde{Y} = \begin{bmatrix} \mathbf{y_{1,1}} & \cdots & \mathbf{y_{1,n}} & \mathbf{y_{1,n+1}} & \cdots & \mathbf{y_{1,n+b}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{y_{m,1}} & \cdots & \mathbf{y_{m,n}} & \mathbf{y_{m,n+1}} & \cdots & \mathbf{y_{m,n+b}} \end{bmatrix} \tag{20}$$

Let the new data be represented by B, the so called bunch-matrix, of width b Equation (28).

$$B = \begin{bmatrix} \mathbf{y_{1,n+1}} & \cdots & \mathbf{y_{1,n+b}} \\ \vdots & \ddots & \vdots \\ \mathbf{y_{m,n+1}} & \cdots & \mathbf{y_{m,n+b}} \end{bmatrix} \tag{21}$$

The SVD of $\tilde{Y}$ is obtained from the following steps, as derived in [15] from [10, 11]:

1. The QR decomposition of the matrix $(I - UU^T)B$ is computed, where I is the identity matrix.

$$Q_B R_B = (I - UU^T)B \tag{22}$$

2. The matrix K is built:

$$K = \begin{bmatrix} S & U^T B \\ 0 & R_B \end{bmatrix} \tag{23}$$

3. The SVD of K is computed:

$$K = U' S' V'^T \tag{24}$$

4. Finally, the existing SVD is updated:

$$\tilde{V}_q = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix} V'_q \tag{25}$$

$$\tilde{S}_q = S'_q \tag{26}$$

$$\tilde{U}_q = [U \; Q_B] U'_q \tag{27}$$

The index q in equations (25), (26), and (27) refers to the truncation rank of the SVD.

5 | Model Reduction of SPH-Data

In the SPH framework, unique complexities arise from potential particle (a) deletion, (b) domain entering, or (c) exiting. In the latter case, the position of the calculation points remains unknown, resulting in additional fields $-\{x, y, z\}-$ to reduce and store via SVD. Similarly, particles may lack data prior to their injection or after their deletion, leading to bunch matrices of irregular row and column size.

Let $Y \in \mathbb{R}^{m_1 \times T}$ be the matrix containing the values $y_{i,j}$ of the field y for particle $i \in \{1, \ldots, m_1\}$ and time $j \in \{0, \ldots, T\}$. The bunch matrix is $B \in \mathbb{R}^{m \times b}$ with $m = m_1 + m_2$, where m_2 is the number of particles injected since T and b refers to the bunch size. B might present missing (non-existing) values, here indicated by **null**, due to the particle not being in the domain at the respective time, as shown in the following

$$B = \begin{bmatrix} \mathbf{y_{1,T+1}} & \mathbf{y_{1,T+2}} & \cdots & \mathbf{null} & \mathbf{null} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{y_{m_1,T+1}} & \mathbf{y_{m_1,T+2}} & \cdots & \mathbf{y_{m_1,T+b-1}} & \mathbf{y_{m_1,T+b}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{null} & \mathbf{null} & \cdots & \mathbf{y_{m,T+b-1}} & \mathbf{y_{m,T+b}} \end{bmatrix} \tag{28}$$

here, the entries $\mathbf{y}_{p,t}$ refer to time instants t when a particle p was present in the domain and are referred to herein as *available data*. We address missing data within **B** using one of the following approaches: if at least one non-null value exists for particle p within **B**, the initial guess for its value at time t is computed as the row mean of all available non-null entries for particle p in **B**; if no non-null values are available for particle p in the current **B**, then for each column t, the initial guess is set to the mean of the non-null values in that column (referred to here as *Mean Imputation*), imputation using the mean of subsets of the matrix defined by blocks of a given number of columns (referred to as *Block-Mean Imputation*), Gappy proper orthogonal decomposition (GPOD), or a combination of mean and block-mean imputation. The GPOD is typically used to provide approximations for missing or corrupted data, cf. exemplary applications regarding noisy PIV measurement [54], inverse design [55], unsteady flow estimation [56], design optimization [57], or flows with deforming meshes [58]. However, since the missing values in SPH data refer to time instants when a particle is not in the considered domain, the approximated value has no physical meaning and only completes the matrix B to compute the SVD. The only desired property of the imputed values is that they should not raise the rank of the matrix so that a sound reduction can be achieved. The basic idea behind the GPOD is to use the SVD to improve the guessed data for the missing values, initially applied by [59] and later modified by [60, 61]. According to Gunes algorithm for GPOD [54, 60, 61]:

1. The temporal mean of the available data within **B** is used as an initial guess to replace the gaps.
2. The truncated SVD of this new full bunch matrix is computed.
3. The gaps within the original **B** are filled using values from reconstructing the truncated SVD of **B**; the original data remains unchanged.

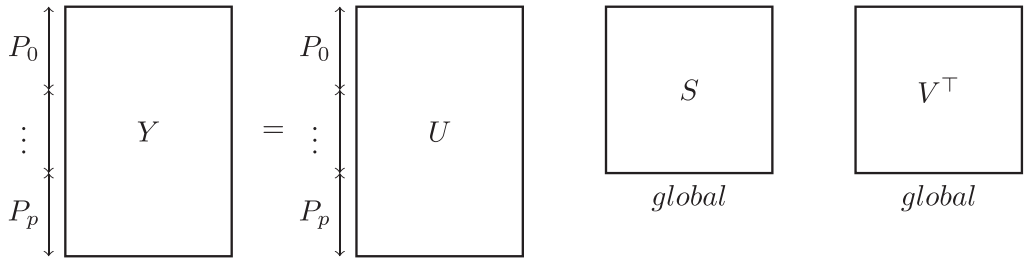


FIGURE 1 | Matrices distribution for spatially parallel but temporally serial data in processes P_i with $i \in [0, p]$.

4. Repeat these steps until convergence occurs, for example, the difference between the reconstructed values of two iterations is within a desired tolerance, the rank of the modified matrix remains unchanged, or a maximum number of iterations is met.

Particles injected past the last update will also miss left-singular vectors. One approach could be to divide the bunch-matrix into $B \in \mathbb{R}^{m_1 \times b}$ to add the new columns and $C \in \mathbb{R}^{m_2 \times (T+b)}$ to add the new rows, which is compatible with the algorithm in Section 4.2. Adding rows, which corresponds to adding C to the SVD of Y, is analogous to adding the columns of $C^\top$ to $Y^\top = VSU^\top$. However, in addition to computing the SVD of $K_B \in \mathbb{R}^{(q+b)\times(q+b)}$, this requires a second SVD on $K_C \in \mathbb{R}^{(q+m_2)\times(q+m_2)}$. The number of particles injected between updates is often immense. Considering that SVD's need to be computed for each b time steps and their computational effort usually scales with $O(mn\min(m,n))$, the additional efforts of K_C can become particularly costly. We decided to substitute the missing U-rows for the m_2 new particles with existing U-rows. This does not affect the singular values as well as the singular vectors of the existing SVD. One can use any row of U. For simplicity, when $m_2 < m_1$ —which is the most common case— the bottom m_2 rows of U are utilized:

$$\begin{bmatrix} u_{1,1} & \cdots & u_{1,q} \\ \vdots & \ddots & \vdots \\ \mathbf{u_{m_1-m_2,1}} & \cdots & \mathbf{u_{m_1-m_2,q}} \\ \vdots & \ddots & \vdots \\ \mathbf{u_{m_1,1}} & \cdots & \mathbf{u_{m_1,q}} \end{bmatrix} \rightarrow \begin{bmatrix} u_{1,1} & \cdots & u_{1,q} \\ \vdots & \ddots & \vdots \\ \mathbf{u_{m_1-m_2,1}} & \cdots & \mathbf{u_{m_1-m_2,q}} \\ \vdots & \ddots & \vdots \\ \mathbf{u_{m_1,1}} & \cdots & \mathbf{u_{m_1,q}} \\ \mathbf{u_{m_1-m_2,1}} & \cdots & \mathbf{u_{m_1-m_2,q}} \\ \vdots & \ddots & \vdots \\ \mathbf{u_{m_1,1}} & \cdots & \mathbf{u_{m_1,q}} \end{bmatrix} \tag{29}$$

Mind that adding rows to $\mathbf{U}$ is analogous to adding rows to $\mathbf{Y}$. Opting to replace the missing rows with existing rows, the data prior to a particle's injection are copies of data related to particles that were already active at that time instant of the simulation. Mind that these values are not physically meaningful and are discarded during post-processing since the new particle was not active yet. If the data matrix Y contains more than one field, the rows to be used depend on how the data has been organized when the snapshots were collected. However, for the remainder of this work, each field undergoes its own SVD and all fields are treated independently.

5.1 | Parallelization

When dealing with spatially parallel but temporally serial data, the data matrix Y and left-singular vectors U are distributed among the CPU/GPU processors. However, the singular values S and the right-singular vectors V coincide on all processors, cf. Figure 1.

Suppose the domain decomposition remains constant in time. In that case, all processors see data from the same amount of particles during run time, and the left-singular values, as well as the respective rows in the bunch matrix, are already placed correctly. However, if the domain decomposition changes, one must carefully handle the parallel communications so that U and B remain correctly aligned. Data from the employed SPH solver exacerbate this issue, as particles might travel between processes even when the domain decomposition does not change. The two following strategies could be adopted to circumvent these issues:

1. Particle-Based Row Transfer: As a particle travels, it carries its respective U and B rows. In this case, the changes in the particle's owner can be seen as a permutation via an elementary matrix E that permutes the rows of Y and U, that is,

$$EY = (EU)SV^\top \tag{30}$$

2. Fixed Left-Singular Vectors: The left-singular vectors remain on the process used for the first update. As a particle travels, it carries its row of B. Before the next SVD update, the rows of B that are misaligned are sent to their owner process.

The first strategy risks having communication overhead, as a particle might travel from one process to another—and even back before the next update. Furthermore, due to load balancing issues, new processes are activated if a minimum number of new particles are added. In this scenario, a large volume of data that scales with the number of particles, fields, modes, etc., needs to be sent to the new process. However, with the second strategy, once a particle is on the correct process, its values are already placed correctly in the bunch matrix and require no further manipulation. Additionally, large data transfers to new processes are avoided. In this work, the fixed left-singular vectors strategy is chosen.

The proposed incremental SVD-SPH strategy is given in Algorithm 1. Therein, parts related to collecting the data in B while ensuring alignment between B and U are not included, as they depend on the specific parallelization of the flow solver. The first step is to determine if there are missing data in the

ALGORITHM 1 | General parallel incremental singular value decomposition process.

```
if missing data
    B = imputeData(B)                          // Replace missing data, cf. Section 5
if new SVD
    U, S, V = initialSVD(U, S, V, B)           // Compute the initial SVD, cf. Algorithm 2
else
    U, S, V = updateSVD(U, S, V, B)            // Update the existing SVD, cf. Algorithm 4
emptyBunchMatrix(B)                            // Clear the bunch-matrix
```

ALGORITHM 2 | Initial parallel singular value decomposition.

```
if parallel
    Q, R = TSQR(B)                             // Parallel QR decomposition (Algorithm 3)
    U_R, S, V = serialSVD(R)                   // Serial SV decomposition
    U = Q U_R
else
    U, S, V = serialSVD(B)
```

ALGORITHM 3 | Parallel tall and skinny QR decomposition.

```
if parallel
    Q_1, R_1 = serialQR(B)                     // Serial QR decomposition of local B block
    R̃ = AllGather(R_1)                         // Gather all parts of R
    Q_2, R = serialQR(R̃)                       // Serial QR decomposition on gathered R
    Q̃ = Q_2[procId · A.cols() : (procId + 1) · A.cols(), :]   // Extract process specific data
    Q = Q_1 Q̃
else
    Q, R = serialQR(B)                         // Serial QR decomposition of B
```

bunch matrix. If this condition is met, an imputation strategy is performed as described in Section 5. Subsequently, a new SVD construction or an update is performed. Finally, the bunch matrix is emptied.

In general, serial and parallel SVD and QR decompositions can be computed using algorithms from prominent linear algebra libraries and packages, for example, [62, 63]. However, due to the specific, distributed matrix structures and the parallelization as well as communication processes involved, we opt for the parallel implementation of SV and QR decompositions as described in Algorithms 2 and 3.

Parallel QR decompositions follow strategies of [64–66] for tall and skinny (TS) matrices, frequently labeled TSQR. The implemented parallel TSQR decomposition is built upon the following three steps: (1) A serial QR decomposition is performed on the matrix's B local blocks, resulting in local R_1 matrices that are subsequently gathered toward a temporary, global $\tilde{R}$ matrix, on which a second (2) serial QR decomposition is performed, offering the global R. Finally, local Q matrices from the first serial decompositions are (3) multiplied by the local components of the second serial QR decomposition to obtain the final global Q. The procedure minimizes the parallel communication effort and outperforms, for example, (modified) Gram–Schmidt procedures. The computation of the parallel SVD involves an initial parallel TSQR on B, followed by a serial SVD on the resulting R that provides the global S and V. Finally, local U matrices are obtained by multiplying the local Q matrices from the initial TSQR decomposition with the left-singular vectors of R.

The procedure to update the SVD is presented in Algorithm 4 and follows [15] with some key differences: (a) Handling missing rows in U, (b) performing the initial SVD on a complete bunch-matrix rather than a single column vector, (c) neglecting the specification of a truncation criterion, and (d) using the TSQR decomposition instead of a modified Gram–Schmidt procedure. The algorithm starts by checking for a possibly empty local matrix U —which would happen if enough particles were injected since the last update to activate a new process. In that case, the local U is initialized as a zero matrix instead of copying left-singular vectors from other processes, which would require additional communication effort. The remaining cases with already populated local left-singular vectors extend U depending on whether the number of additional rows exceeds the number of existing rows. The global product $U^T B$ is computed in serial mode and then made global by parallel summation, cf. [15]. The residual matrix $P = B - UM$ is orthogonalized using the proposed TSQR decomposition strategy from Alg 3. An additional orthonormalization step is applied to Q, which accounts for a potential numerical loss of orthogonality during SVD updates [11–13, 15, 67, 68]. Following the enhanced itSVD algorithm outlined in [15], this step precedes the construction of K. The global matrix K is constructed and decomposed on one specific process to avoid rounding errors in the singular values. Truncated SVD results are

ALGORITHM 4 | Update of the incremental singular value decomposition.

if $U.rows() == 0$
  $U = \text{Zeros(m, } r)$ // New process without existing U
  // m current number of rows of B assigned to this process
  // r current global truncation rank of U
else
  $m_1 = U.rows()$ // Number of rows of U assigned to this process at last update
  $m_2 = B.rows() - U.rows()$ // Number of rows to add for this update
  if $m_2 \leq m_1$
    U = AppendBottomRows(U, m2) // Append bottom m2 rows of U to itself
  else
    $i = \text{round}\left(\frac{m_2}{m_1}\right)$ // Full copies needed
    $j = m_2 - i \times m_1$ // Remaining rows needed
    U = AppendMultipleTimes(U, i) //Append U to itself i times
    U = AppendBottomRows(U, j) //Append j bottom rows of U to itself
$M = U^\top B$ // Compute the local matrix M
if parallel
  $M = \text{parallelSum}(M)$ // Sum M from all processes to make it global
$P = B - UM$
$Q_p, R_p = \text{TSQR}(P)$
$Q = \begin{bmatrix} U & Q_p \end{bmatrix}$
$Q_q, R_q = \text{TSQR}(Q)$
if $procId == 0$
  $K = R_q \begin{bmatrix} S & U^T B \\ 0 & R_p \end{bmatrix}$
  $U', S', V' = \text{serialSVD}(K)$
  if adaptive truncation
    q = computeTruncationRank(S) //Criteria-dependent truncation
  else
    q = constant // Truncation by fixed value
  $U' = U'(:, 1 : q)$
  $S = S'(1 : q)$
  $V' = V'(:, 1 : q)$
$Bcast(U', S, V')$ // Broadcast truncated results to all processes
$R = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}$
$U = Q_q U'$
$V = RV'$

then made global through broadcasting, and the SVD matrices are updated.

All multi-CPU operations are performed with the message passing interface (MPI) protocol, while the GPU operations are performed with the compute unified device architecture (CUDA) along with the *cuSOLVER* and *cuBLAS* libraries. Serial matrix operations and decompositions utilize the *Eigen* library [69].

6 | Validation

The subsequent validation displays different aspects of the reconstruction. First, the quality of the reconstruction for different ranks is discussed. To this end, we use the relative reconstruction error

$$\varepsilon(\phi(\tau)) = 100 \cdot \max\left(\left\| \frac{\phi_{\text{FOM}}(\tau) - \phi_{\text{ROM}}(\tau)}{\phi_{\text{ref}}} \right\|\right) \tag{31}$$

which observes the maximum difference between *Full Order Model* (SPH) and *Reduced Order Model* (SVD) for a field ϕ at time τ, with respect to the maximum absolute value during the complete simulation ϕ_{ref}, where $\|\cdot\|$ represents the absolute value. The SVD is either adaptively truncated by neglecting all singular values that are five orders of magnitude smaller than the first singular value or always truncated at a fixed rank. Second, the overhead of the procedures is assessed, which involves both the computational time and the associated storage/memory efforts. For short-term dynamics with low periodicity and limited temporal correlations, a segmentation of the time window under consideration for the SVD could be advantageous. The latter is associated with separate ROMs for each temporal segment combined with adaptive truncation criteria. Related computational advantages are, therefore, not guaranteed and will be investigated in the first case of this study. In this context, we will also examine the changes in the reconstruction error when moving from one segment to another.

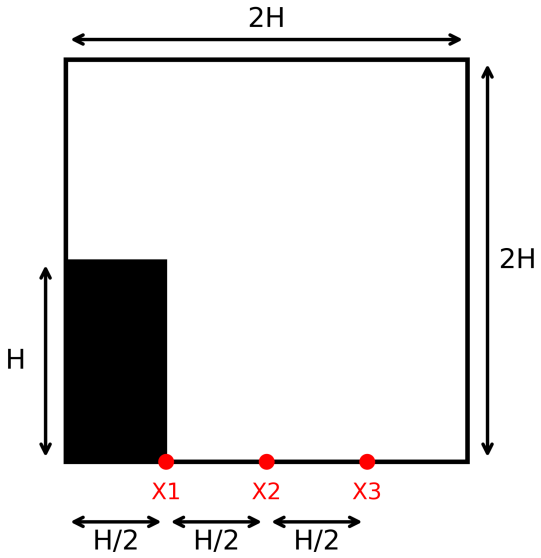


FIGURE 2 | 2D Dam Break: Illustration of the initial configuration. Three height measurement probes (X1, X2, and X3) are highlighted in red. [Colour figure can be viewed at wileyonlinelibrary.com]

All simulations were conducted on an AMD EPYC 7573X 32-Core Processor (2 sockets), 512GiB of RAM, and an NVIDIA RTX A6000 GPU (48GiB memory) for the 3D application.

6.1 | 2D Dam Break

The 2D dam break simulation models the temporal evolution of an initially rectangular column of water ($\rho_0 = 1000$ kg/m^3) under the influence of pressure and gravity. The complex sloshing motion involves different wave patterns and splashing as the water interacts with the walls. The test case is, therefore, frequently used to scrutinize SPH methods, see [70, 71]. The considered domain refers to a closed box, cf. Figure 2, that is, no particle can enter and leave the domain. This allows for an assessment of the incremental SVD, particularly its ability to capture the particle positions without entering missing data.

The square domain spans $2H \times 2H$, where $H = 2m$ denotes the initial water column height. The initial width of the water column refers to $H/2$, and the initial particle spacing is assigned to $\Delta x = 0.02m$, which amounts to 5000 fluid particles. The simulation is performed over 20,000 time steps and terminates before the flow comes to rest, that is, after the waterfront hits the downstream wall, goes up, and then comes back down. The Courant number criterion (18) controls the time step. To preserve a small Mach number $M \leq 0.1$, the numerical speed of sound is assigned to $c_0 = 100$ m/s which is well above an estimated Torricelli speed of $\sqrt{2\ g\ H} \approx 6.3m/s$, with $\mathbf{g} = -g\mathbf{e}_z$ and $g = 9.81m/s^2$. Mind that the bunch size used for the SVD updates is assigned to $b = 400$ in this case.

Figure 3 displays the water body at the final time of the simulation. An FOM solution (blue) and an ROM solution (red) with varied accuracy are compared in all six graphs. As indicated by the figure, an insufficient number of considered SVD modes yields poor reconstructions of the particle positions and the shape of the water body, and reconstructed particles might even leave the domain (cf. Figure 3a). Conversely, using enough modes yields a reconstruction that closely matches the original data. However, results indicate that a sufficient reconstruction of the body of water can be obtained with approximately 50 modes, which refers to 0.25% of the theoretically available modes for the nonsegmented, single-window approach.

A more quantitative assessment of the reconstruction by the reduced order model is shown in Figures 4–8. The figures compare the temporal evolution of the reconstruction error for the adaptive rank truncation (orange line) with results obtained in combination with four different fixed truncation ranks $q \in [50, 100, 200, 400]$, that would be considered sufficient based on the visual inspection of the reconstructed water body. Attention is confined to the error obtained for the density, velocity, and position reconstruction. In each figure, two different approaches are presented. The respective left panels depict the results for a single window SVD, and the right panels refer to a subdivision into ten distinct, consecutive time windows that cover an equal amount of time steps with ten respective SVDs.

In order to ensure efficiency, the adaptive truncation variant is preferred. When attention is directed to the reconstruction of the density and positions, cf. Figures 4–6, it appears that the adaptive truncation essentially corresponds to a truncation at $q = 50$, essentially confirming the results of Figure 3. Except for very few outliers, maximum relative error magnitudes below one per cent are achievable with the adaptive scheme. The picture changes for the reconstruction of the velocities, displayed in Figures 7 and 8, where the adaptive truncation agrees with the fixed-rank truncation at $q = 200$ or $q = 400$ for a similar error level, which reveals more complex dynamics of the velocities and a smaller degree of compression. The large discrepancy of the required rank recommends separate model reductions for each field considered to increase the efficiency. Figure 9 depicts the temporal evolution of the computed water level (height) at the locations X1, X2, and X3, cf. Figure 2. The figure shows excellent agreement between the FOM and the ROM results. Note that isolated splashes and droplets were not taken into account when measuring the height.

Due to the nonperiodic behavior of the dam break flow, more information is continuously added to the SVD as the simulation progresses, and the truncation ranks for each field keep growing, cf. Figure 10a. Having multiple SVDs that handle separate time windows might help mitigate this problem. Let us consider two exemplary setups: one option in which a single SVD covers all 20,000 time steps, and a second option in which 10 separate SVDs each cover 2000 non-overlapping time steps. Both setups feature the same bunch width, that is, $b = 400$. Although the segmented SVD approach, in combination with adaptive truncation, always shows a better agreement between FOM and ROM, there may be doubts about the associated data effort. For the example employed above, that is, ten vs. one window, Figure 10b reveals an average maximum rank for the vertical velocity w above 100. When multiplied by ten windows, this is more than the maximum rank of the single-window setup indicated in Figure 10a, which is slightly below 800 and would give the impression that the 10-window setup is less efficient than the single-window setup. However, the associated data effort for multiple windows is estimated by more than simply adding up the maximum rank for all windows. Considering that the size of the left singular vector is

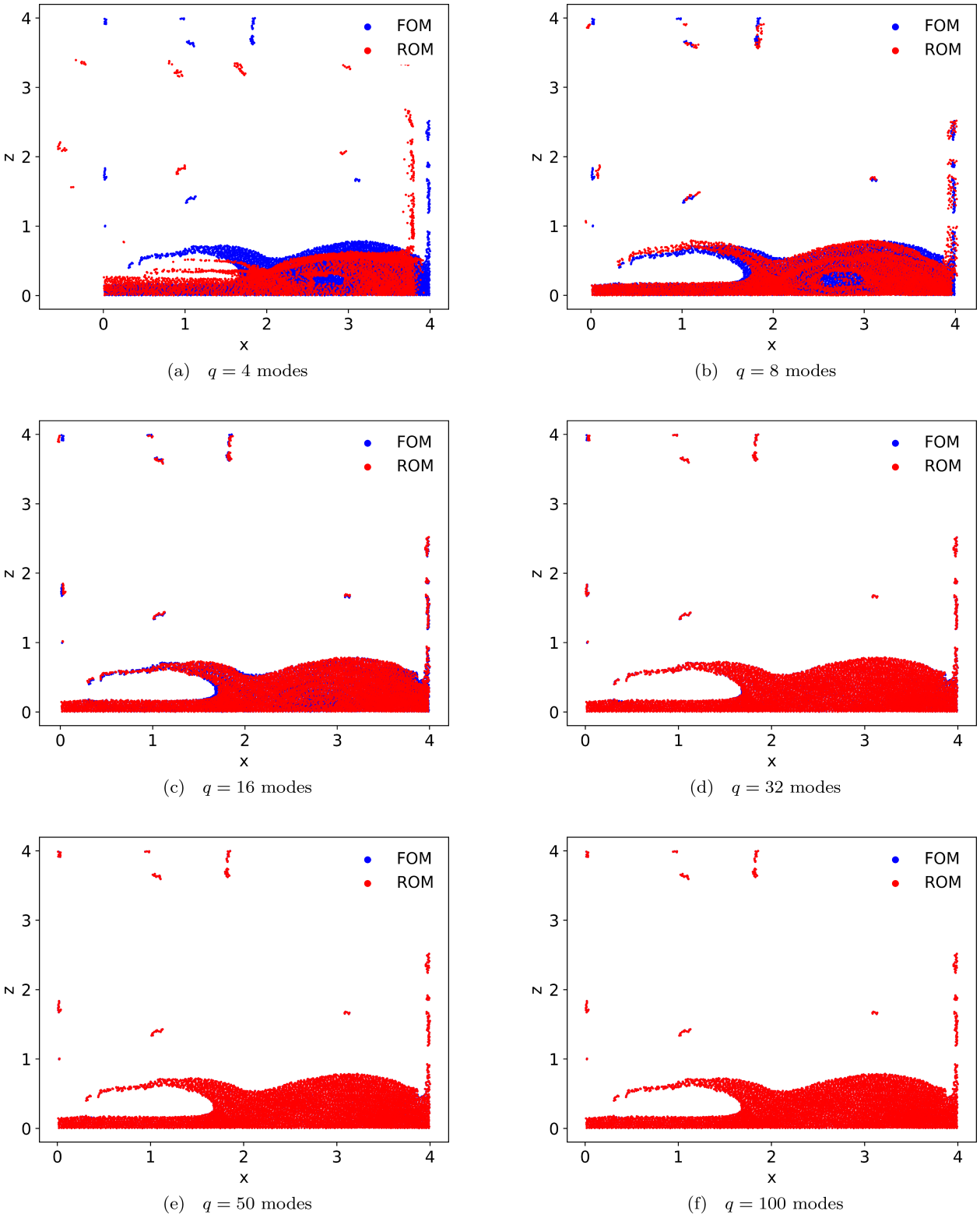


FIGURE 3 | 2D Dam Break: Snapshot of the particle positions at the last simulated time step ($t = 2.26$s) obtained from the original FOM (blue; SPH) and the SVD-based reconstruction (red) for six different numbers of considered modes. [Colour figure can be viewed at wileyonlinelibrary.com]

determined by the product of the rank and the number of particles, the size of the singular values corresponds to the rank. The product of the number of time steps and the rank determines the size of the right singular vector. The following equation therefore gives the compression ratio (CR) for the windowing strategy CR_w:

$$CR_w = \frac{\sum_{\text{Fields}} q^1}{\sum_{i=1}^{N_w} \left(\sum_{\text{Fields}} q^{N_w}(i) \right)} \left(\frac{N_p + 1 + N_s}{N_p + 1 + \frac{N_s}{N_w}} \right) \tag{32}$$

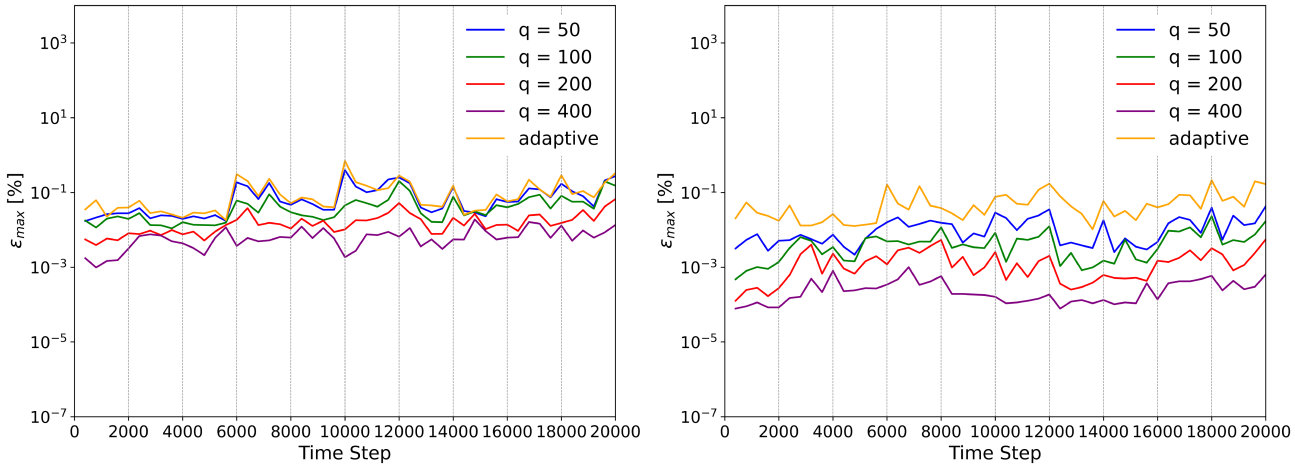


FIGURE 4 | 2D Dam Break: Temporal evolution of the maximum density reconstruction error in percent obtained with different truncation ranks q. Left graph shows single window/SVD results and right graph displays results of segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

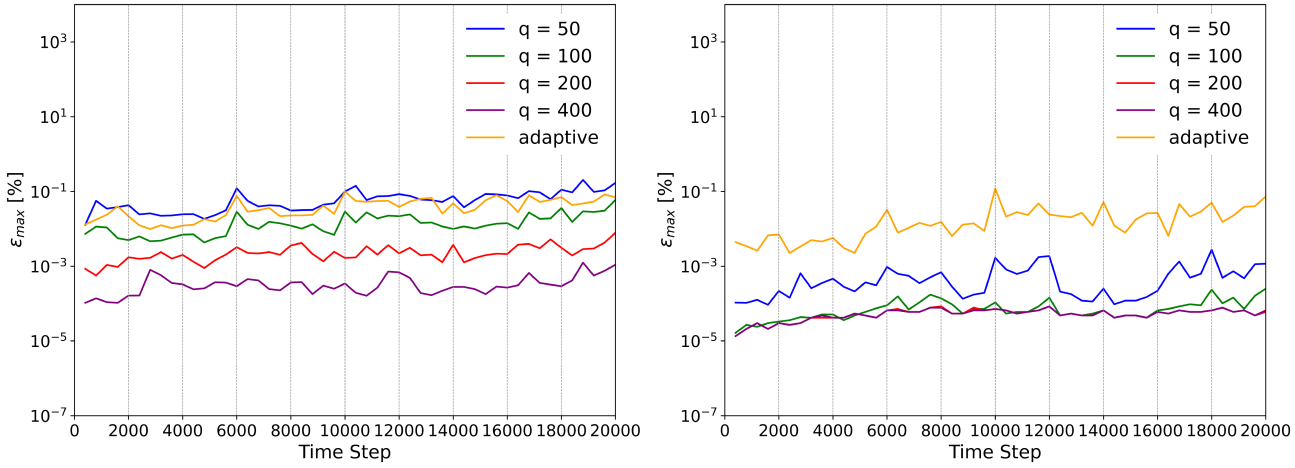


FIGURE 5 | 2D Dam Break: Temporal evolution of the maximum horizontal position reconstruction error in percent obtained with different truncation ranks q. Left graph shows single window/SVD results and right graph displays results of segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

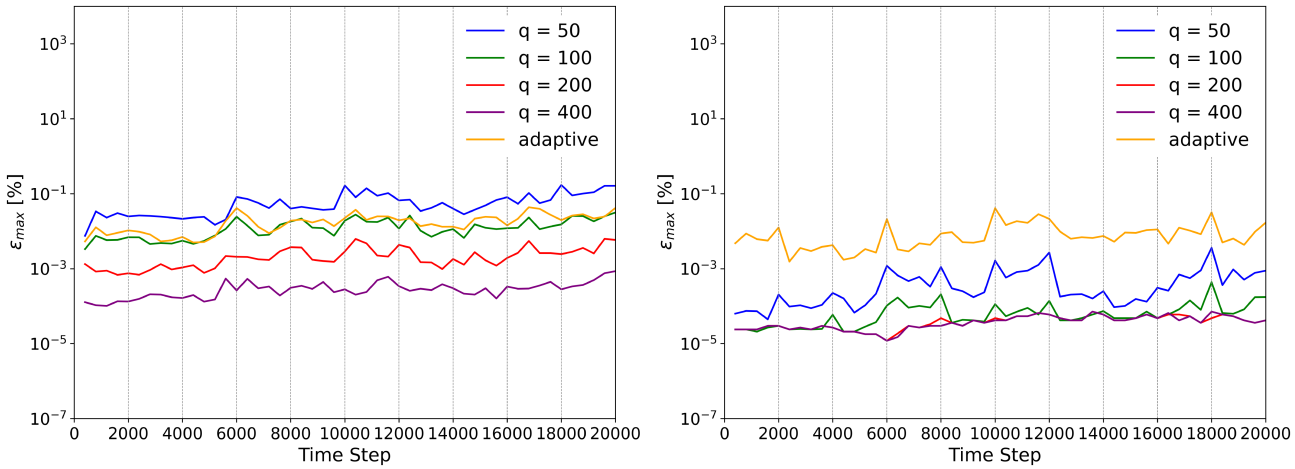


FIGURE 6 | 2D Dam Break: Temporal evolution of the maximum vertical position reconstruction error in percent obtained with different truncation ranks q. Left graph shows single window/SVD results and right graph displays results of segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

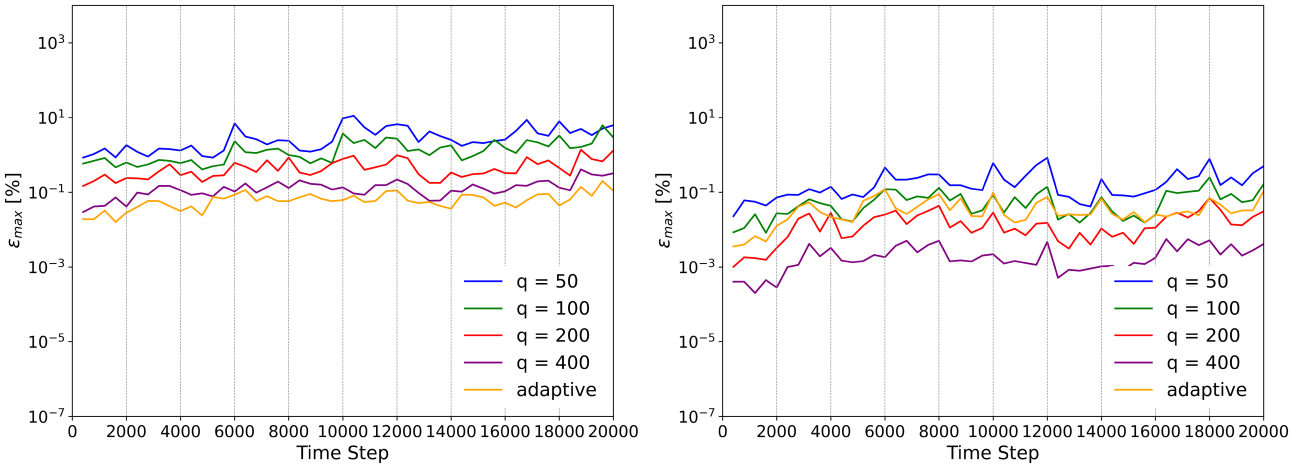


FIGURE 7 | 2D Dam Break: Temporal evolution of the maximum horizontal velocity reconstruction error in percent obtained with different truncation ranks q. Left graph shows single window/SVD results and right graph displays results of segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

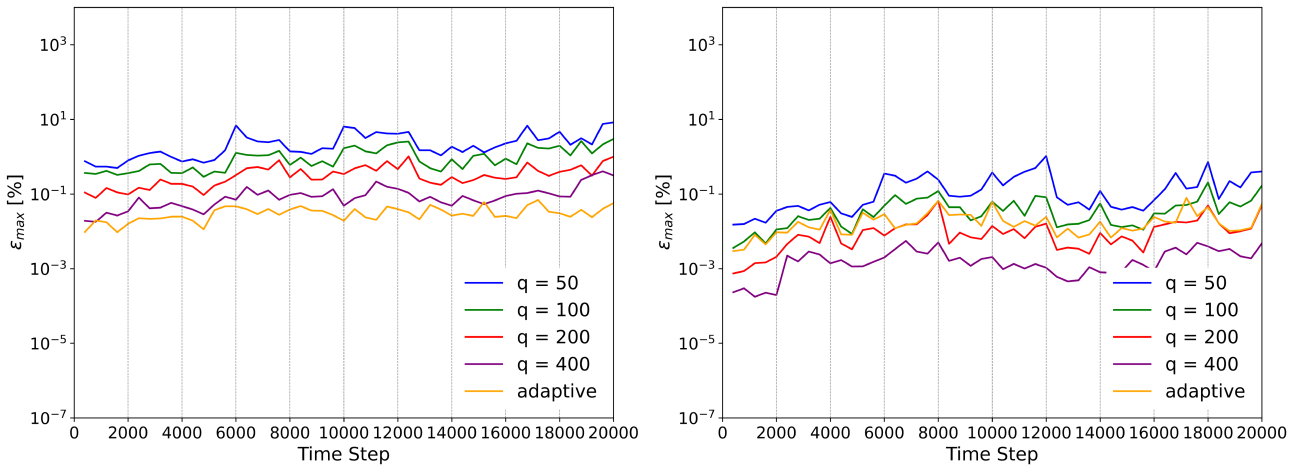


FIGURE 8 | 2D Dam Break: Temporal evolution of the maximum vertical velocity reconstruction error in percent obtained with different truncation ranks q. Left graph shows single window/SVD results and right graph displays results of segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]

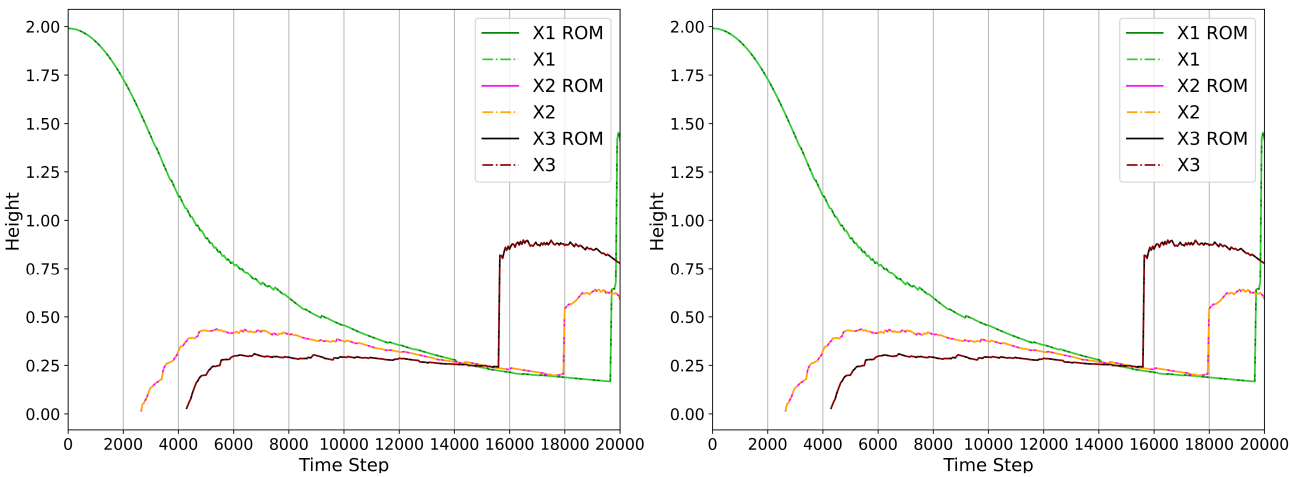
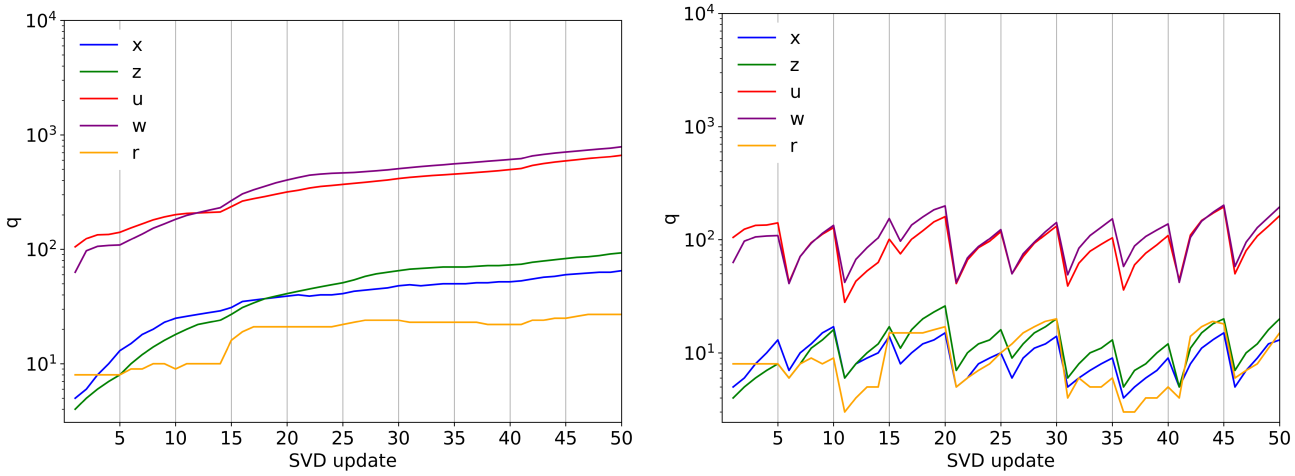


FIGURE 9 | 2D Dam Break: Temporal evolution of the water level (height) at the horizontal locations X1, X2, and X3, obtained from the ROM with adaptive truncation. Left graph shows single window/SVD results and right graph displays results of a segmentation into 10 (equally sized) windows/SVDs, both in combination with a bunch matrix width $b = 400$. [Colour figure can be viewed at wileyonlinelibrary.com]



(a) 1 window (b) 10 windows

FIGURE 10 | 2D Dam Break: Evolution of the truncation rank over SVD update for (a) one and (b) ten windows. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 1 | 2D Dam Break: Compression ratio for the windowing strategy CR_w obtained with different number of windows N_w using 5000 particles and 20,000 time steps ($\sum_{\text{Fields}} q^1 = 1637$).

N_w	2	5	10	25	50
CR_w	1.31	1.69	1.76	1.56	1.30
$\sum_i (\sum_{\text{Fields}} q^{N_w}(i))$	2082	2697	3320	4526	5824

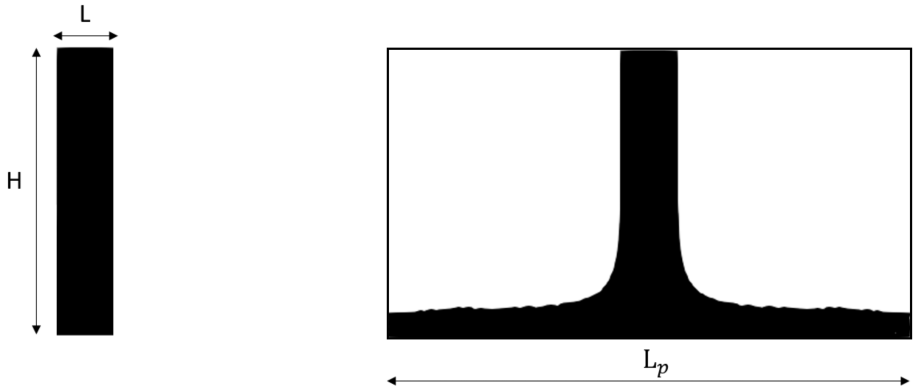
TABLE 2 | 2D Dam Break: Computational overhead $\frac{t^{SVD}}{t^{SPH}}$ in percent for the windowing strategy obtained with different number of windows N_w and truncation ranks q using 5000 particles and 20,000 time steps.

$t^{SVD}/t^{SPH}[\%]$	$N_w = 1$	$N_w = 2$	$N_w = 5$	$N_w = 10$	$N_w = 25$	$N_w = 50$
$q = 50$	14.92	14.27	13.7	13.14	11.72	9.31
$q = 100$	17.53	16.83	16.14	15.21	12.97	9.31
$q = 200$	24.90	23.13	21.70	20.14	16.08	9.31
$q = 400$	43.95	39.85	36.45	33.08	24.03	9.31
adaptive q	27.84	19.05	14.54	12.85	11.07	9.31

In Equation (32), the superscript indicates the total number of windows: q^1 represents the final rank of a particular field for a single window, while $q^{N_w}(i)$ denotes the corresponding final rank for the i-th window. N_s is the total number of time steps, N_p is the number of particles, and N_w is the number of windows. For huge particle numbers that significantly exceed the number of time steps, that is, $N_s/N_p << 1$, the second ratio in (32) is close to unity. Moreover, the factor in brackets in equation (32) tends to a $(N_s/N_p) \to 1$ for large numbers of windows N_W, while the total number of involved ranks in the denominator of the first factor increases to much larger values. Note that in cases featuring open boundaries, that is inlets and outlets, N_p depends on the respective window and the definition is altered towards

$$CR_w = \frac{\left[\sum_{\text{Fields}} q^1\right] (N_p^1 + 1 + N_s)}{\sum_{i=1}^{N_w} \left[\left(\sum_{\text{Fields}} q^{N_w}(i)\right) \left(N_p^{N_w}(i) + 1 + \frac{N_s}{N_w} \right) \right]} \tag{33}$$

Table 1 displays the resulting CR for different numbers of windows $N_w = 2 - 50$ and indicates that the segmented, piece-wise approach indeed achieves the best degree of compression due to the small amount of particles and the more significant number of time steps for $N_W = 10$. Mind that $CR_w = 1$ refers to the single window approach. Table 2 depicts the computational surplus (wall-clock time) of the ROM, normalized with the respective SPH simulation time. The table shows the development of the relative computing effort as a function of the number of windows N_W and the truncation rank q. It is seen that the effort scales almost linearly with the truncation rank for a single window. Furthermore, a strong positive influence of increasing window numbers for high truncation ranks is noticeable. While each new SVD has a constant computational cost relative to the truncation, the cost of an update does not scale linearly with the number of retained modes, making the influence of the number of windows dependent on the rank. As the number of windows increases, the number of new SVD computations grows, which



(a) Initial state (b) Fully developed jet

FIGURE 11 | 2D Impinging Jet: Illustration of (a) the initial configuration/domain and (b) the fully developed state for the investigated.

causes the overall cost of the algorithm to converge toward a limit where only new SVDs are computed, rather than updates being applied to existing ones. In the limit case, that is, for $N_w = N_s/b = 50$ windows for the considered case, only new SVDs are computed, leading to uniform computational cost across all truncations in the last column of Table 2. Note that the computational overhead with adaptive tuncation for $N_w = 10$ windows is less than the overhead with $q = 50$ for a single window.

The incremental SVD accurately reproduces meshless particle flow data in agreement with the FOM. While segmented, piece-wise SVD methods reduce the computational overhead, it only saves memory in combination with an adaptive truncation. In the remainder of this paper, truncation is assumed to be adaptive, and the application of the windowing strategy is always specified.

6.2 | 2D Impinging Jet

The second test case concerns the simulation of an impinging jet. The rectangular domain has a horizontal bottom boundary corresponding to a flat plate with a length $L_p = 0.18$m, and the height of the domain reads $H = 0.1$m. Particles are injected vertically through a confined, centered area of width $L = 0.02$m from the top with a velocity $w_{in} = -20$ m/s and flow towards the plate along its normal. Figure 11a illustrates the initial particle configuration that employs a regular Cartesian lattice distance $\Delta x = 0.0005$ and amounts to 8000 particles. The speed of sound is assigned to $c_0 = 200$ m/s, that is, $w_{in}/c_0 = 0.1$, and the influence of gravity is neglected.

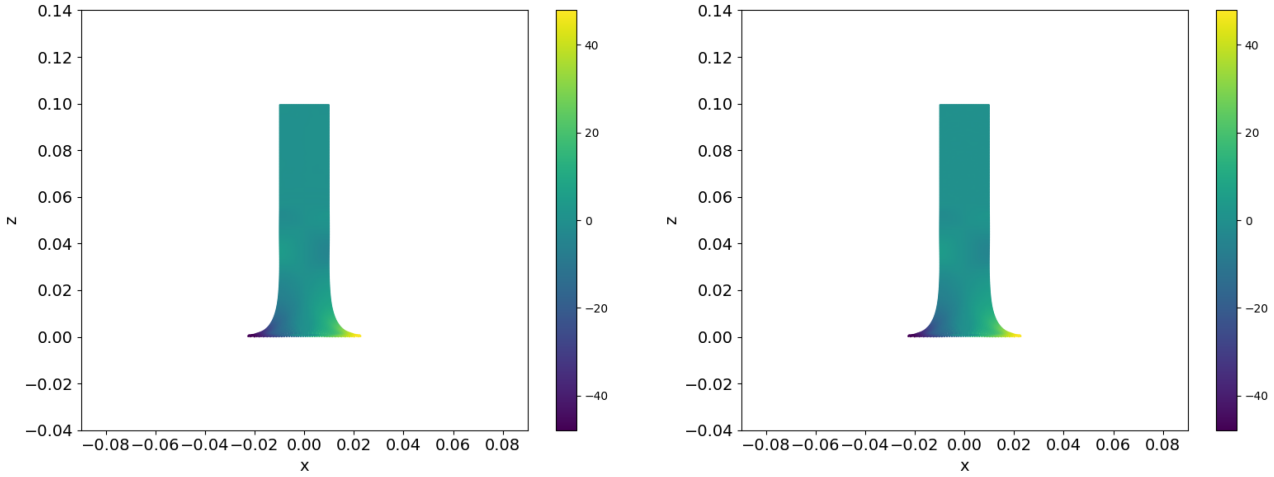
The simulation is performed over $N_s = 5000$ time steps and terminates in a fully developed state illustrated in Figure 11. The total number of active particles in the fully-developed (pseudo-steady) state oscillates around $N_p = 15500$.

Following the impact, the fluid is horizontally displaced, eventually reaching the domain's vertical borders and leaving it. Any particle outside the rectangular computational domain ($L_p \times H$) is removed from the SPH calculation. The presence of inlet and outlet sections distinguishes this case from the dam break case investigate in the previous section. At the same time, it makes this case interesting for assessing how imputing/assigning missing values for the empty positions of the data matrices affects the accuracy of the reconstruction, the achievable compression and the computational effort. The section therefore focuses on discussions on the influence of (a) the imputation as well as (b) the windowing strategy on the accuracy, the compression rate and the related CPU effort.

All four imputation strategies described in Section 5, that is, mean (MI), block-mean (BMI), hybrid MI-BMI and gappy POD (GPOD) are assessed here. The hybrid mean and block-mean imputation, denoted as h_{MI-BMI}, refers to a situation, where a bunch matrix with a standard deviation below unity after the MI is replaced by BMI. Furthermore, the GPOD approach is assessed for three different (fixed) iteration numbers, that is, GPOD(i), where $i \in [1, 5, 10]$.

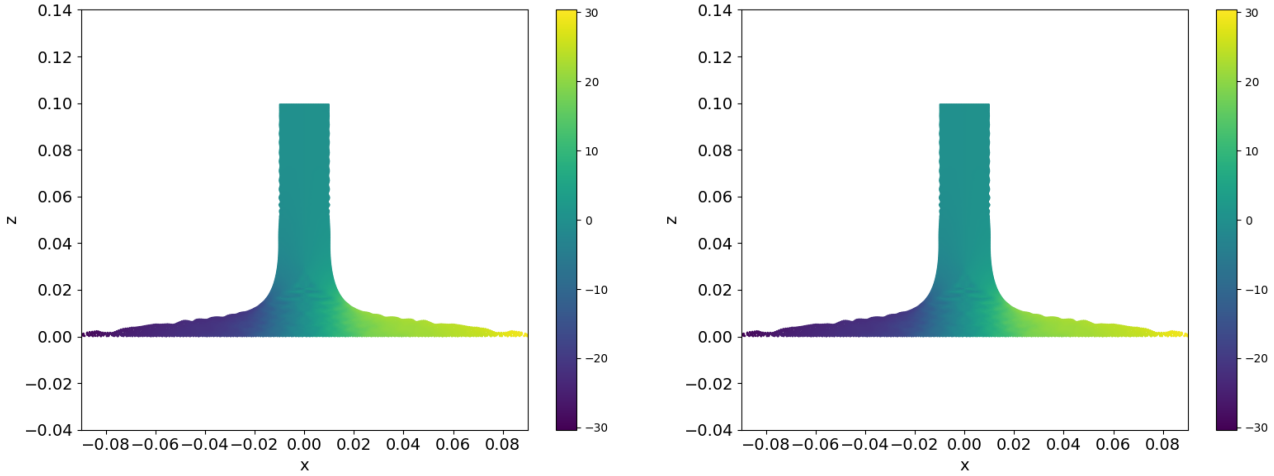
Figures 12 and 13 display two qualitative comparison of the reconstructed jet and the full-order SPH data extracted at three different time instants, with $b = 250$, for the u-velocity field and dynamic pressure, respectively.

In line with Equation (31), we build an error matrix $E \in \mathbb{R}^{5\times5000}$, where each row is assigned to one of the five physical fields stored—here in particular the two components of position, the two components of velocity, and the density—and each column corresponds to a particular time step. This matrix's mean and maximum values are indicated by ϵ_{mean} and ϵ_{max}, respectively. Table 3 shows the final rank q_ϕ of a field ϕ, the sum of all field ranks $\sum q$, the error measures ϵ_{mean} and ϵ_{max}, as well as the relative computational overhead $t^{SVD}/t^{SPH} \cdot 100\%$ for all utilized imputation strategies. Therein, t^{SVD} and t^{SPH} refer to the measured wall clock time of the ROM construction and the flow solver integration, respectively. The simulations are conducted thrice to underline the credibility of the measured wall clock time, and their mean value is reported. Investigations of this first step are performed for a fixed bunch width of $b = 250$ and no windowing was employed. While the different imputation strategies do not significantly affect the accuracy, the associated number of modes and computational effort vary significantly. Considering that the imputation strategies were tested on the same simulation, the number of retained modes directly relates to the compression rate. As expected, the GPOD with 10 iterations achieves



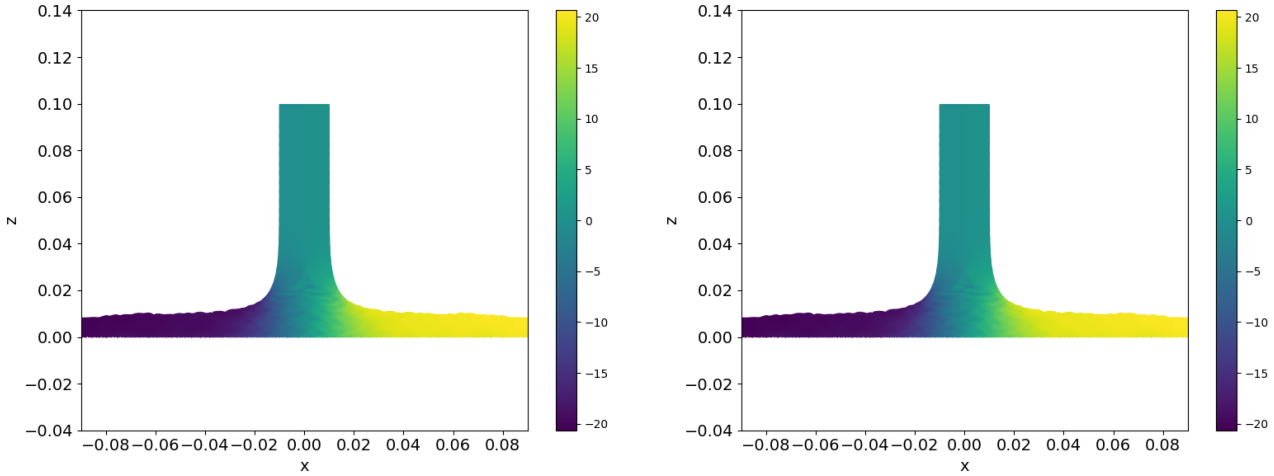
(a) FOM time step = 250

(b) ROM time step = 250



(c) FOM time step = 2500

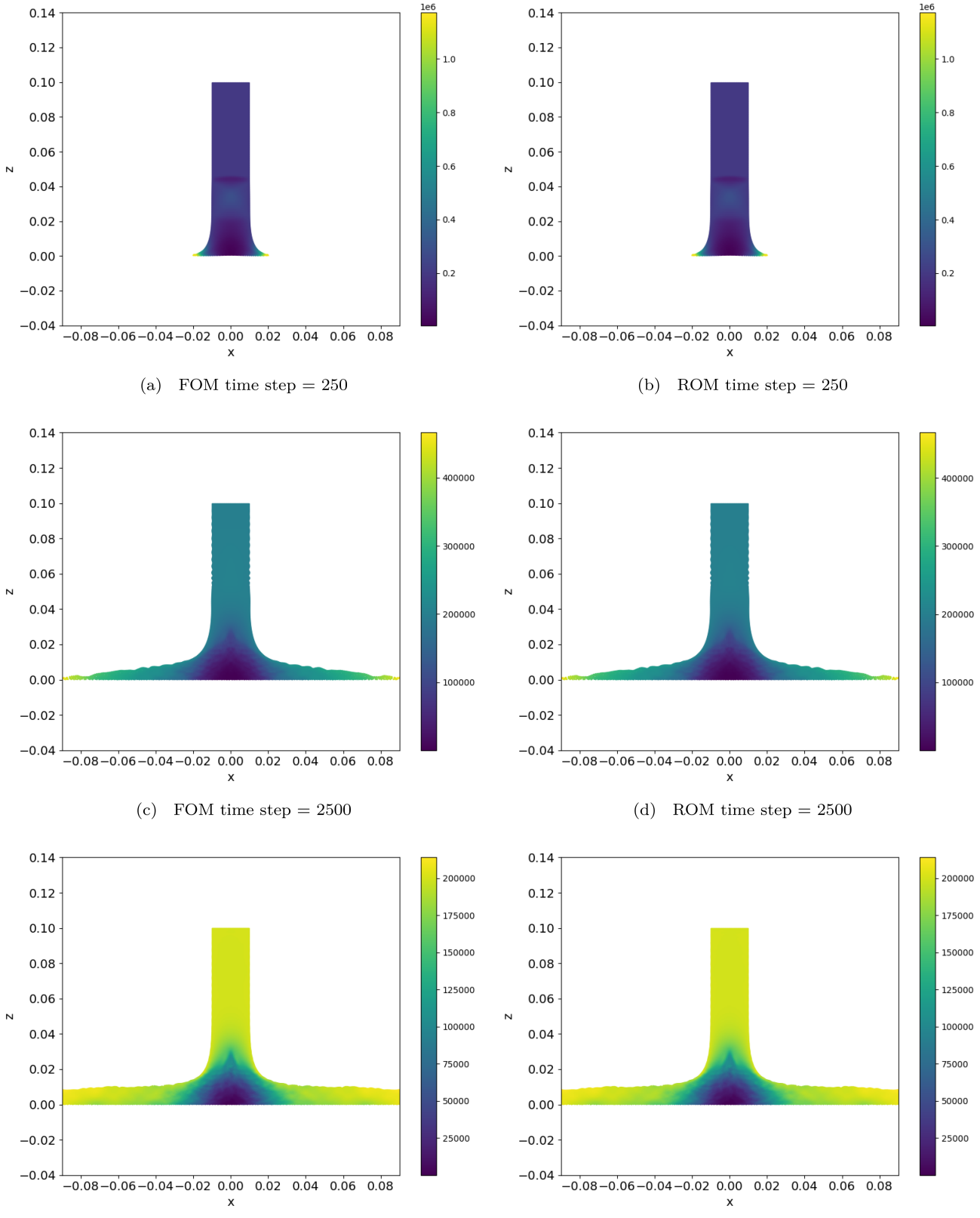
(d) ROM time step = 2500



(e) FOM time step = 5000

(f) ROM time step = 5000

FIGURE 12 | 2D Impinging Jet: Snap-shot of the particle positions colored by u-velocity field obtained from the original FOM (left; SPH) and the SVD-based reconstruction (right) at three different time instants. [Colour figure can be viewed at wileyonlinelibrary.com]



(a) FOM time step = 250

(b) ROM time step = 250

(c) FOM time step = 2500

(d) ROM time step = 2500

(e) FOM time step = 5000

(f) ROM time step = 5000

FIGURE 13 | 2D Impinging Jet: Snap-shot of the particle positions colored by dynamic pressure field obtained from the original FOM (left; SPH) and the SVD-based reconstruction (right) at three different time instants. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 3 | Reconstruction of the 2D impinging jet results for 5000 time steps using 6 different imputation strategies without windowing: Comparison of mode counts q_ϕ, mean and maximum reconstruction error ε, as well as relative computational overhead t^{SVD}/t^{SPH} for the mean imputation (MI), block-mean imputation (BMI), three Gappy POD methods and the hybrid mean/block-mean h_{MI-BMI} imputation strategy, from top to bottom, respectively.

	q_x	q_z	q_u	q_w	q_ρ	$\sum q_\phi$	$\varepsilon_{mean}[\%]$	$\varepsilon_{max}[\%]$	$t^{SVD}/t^{SPH} \cdot 100\%$
MI	856	254	523	480	160	2391	0.071	0.377	31.34
BMI	301	236	593	598	176	1904	0.070	0.235	29.49
GPOD(1)	663	253	515	472	162	2191	0.081	0.373	37.64
GPOD(5)	472	204	511	466	163	1948	0.077	0.376	62.89
GPOD(10)	364	179	509	463	163	1813	0.075	0.374	95.60
h_{MI-BMI}	301	236	523	480	160	1818	0.071	0.377	27.37

TABLE 4 | Reconstruction of the 2D impinging jet results for 5000 time steps using the h_{MI-BMI} imputation strategy: CR of the reduced-order mdels measured against a Full-Order Full-Storage approach for different windows N_w and bunch-widths b sizes.

CR	$b = 50$	$b = 125$	$b = 250$	$b = 625$
$N_w = 1$	7.72	7.74	7.80	7.79
$N_w = 2$	8.78	8.88	8.96	9.03
$N_w = 4$	8.26	8.38	8.46	8.52

TABLE 5 | Reconstruction of the 2D impinging jet results for 5000 time steps using the h_{MI-BMI} imputation strategy: Relative computational overhead $t^{SVD}/t^{SPH} \cdot 100\%$ of the Singular Value Decomposition against the flow solver's simulation time for different windows N_w and bunch-widths b sizes.

$t^{SVD}/t^{SPH} \cdot 100\%$	$b = 50$	$b = 125$	$b = 250$	$b = 625$
$N_w = 1$	50.0	30.4	27.37	37.6
$N_w = 2$	17.6	13.1	14.8	26.9
$N_w = 4$	7.60	7.60	10.0	22.1

the best compression and requires the least number of modes. On the other hand, this example has a CPU overhead that is almost as expensive as the SPH simulation. The BMI method is algorithmically very simple and requires only a moderate computational effort in the range of 30%. However, it still delivers competitive compression rates. The strategy can be further optimized slightly in combination with the hybrid h_{MI-BMI}. Hence, the strategy we employ for the remaining investigations is the hybrid h_{MI-BMI} approach.

In a second step, the windowing strategy is assessed for the h_{MI-BMI}. The number of particles included in a given SVD and the number of time steps assigned to the SVD are influencing parameters on the CR (cf. Equation 33) and the CPU effort. However, since we also store values for inactive particles, the CR is measured against a Full-Order Full-Storage (FOFS) approach. For each time step (n_t), one needs to store one numerical value per field (n_f) per active particle (n_p), allowing for a hypothetical FOFS approach's storing effort, proportional to approximately $c\ \mathrm{n_f} \sum_{i=1}^{i=n_t} \mathrm{n_p(i)}$, where c denotes the required size for storing a number, that depends, for example, on the underlying single- or double precision metric. Table 4 presents the results for three different windowing configurations and four different bunch-widths. The CR moderately varies between $7.72 \leq CR \leq 9.03$ and reveals minor benefits for the single window approach. The relative computational overhead $t^{SVD}/t^{SPH} \cdot 100\%$ is provided in Table 5. In contrast to the CR, the computational overhead varies substantially, between $7.6 \leq t^{SVD}/t^{SPH} \cdot 100\% \leq 50.0$. The overhead lowers as the number of windows increases for all considered bunch-widths. Additionally, a larger number of windows seems to favor smaller bunch widths, as minimal efforts are continuously shifted towards smaller bunch-widths when the amount of windows increases.

6.3 | 3D Pelton Runner

The final application examines a Pelton turbine, that is, a hydraulic impulse machine adapted for high head and low discharge. Its main components are the distributor, the injectors, the runner, and the housing. Mesh-based methods are well-suited to deal with the confined flow in the distributor and injectors. However, the runner and housing are characterized by complex free surface flow, where SPH is an attractive alternative. The simulated configuration adopts a set-up similar to that described in a previous study [42]. As illustrated in Figure 14, it refers to a runner sector comprising five buckets fed with a single water jet. It employs a symmetry condition in the mid-plane of the runner.

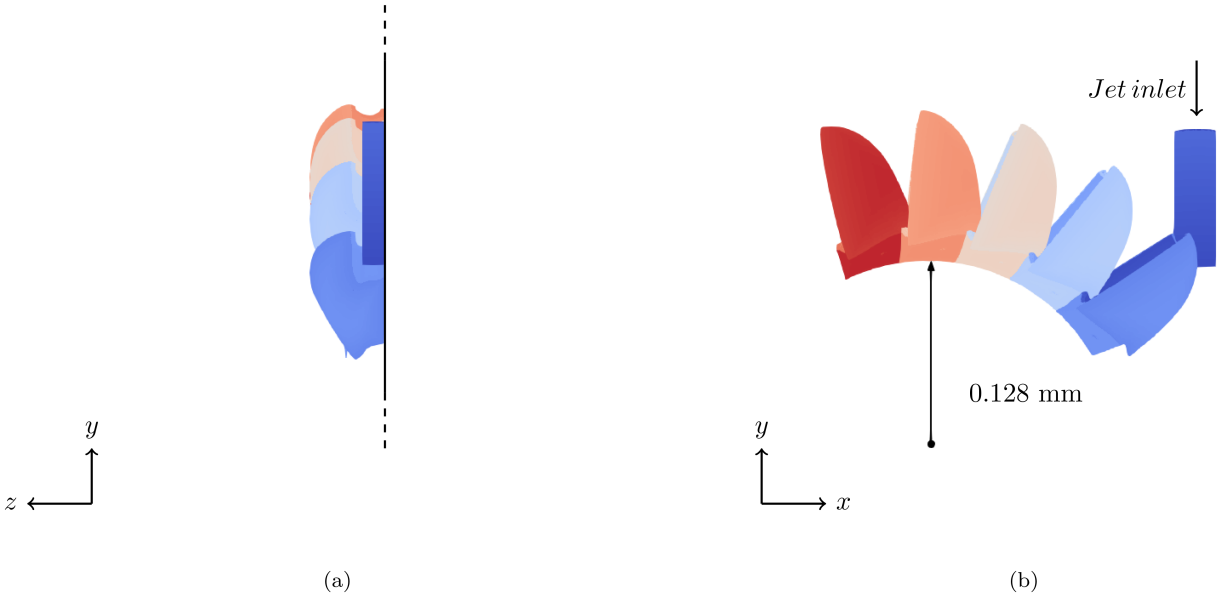


FIGURE 14 | Schematic of the initial 3D Pelton runner configuration which consist of the jet and five buckets. The position of the nozzle and the incoming jet are displayed on the right figure (b). The left figure (a) shows the plane of symmetry. [Colour figure can be viewed at wileyonlinelibrary.com]

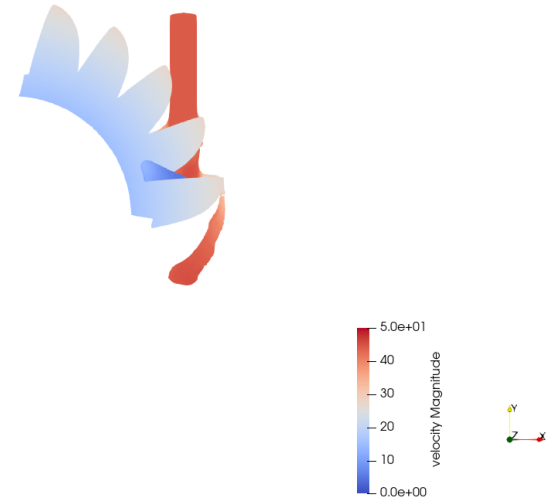
Following the approach outlined in [42], the simulation includes all possibly wetted surfaces and is carried out in a pure Lagrangian mode, discretizing solely the liquid fluid phase and the solid bucket model, excluding the air phase. The box-shaped physical domain has a height of 0.55 m, a width of 0.55 m, and a depth of 0.068 m. The runner is centered in the x-y plane, and the runner center is positioned at $(0, 0, 0)$. The buckets are modeled by (solid) particles with prescribed motion. Fluid particles are injected with a speed of 44.30 m/s, and the initial particle distance reads $\Delta x = 0.001$ m. The speed of sound is assigned to $c = 443.00$ m/s, that is, $v_{in}/c = 0.1$, and the influence of gravity is neglected. The simulation is performed over $N_s = 30,000$ time steps and terminates when all buckets have interacted with the jet. Based on the results obtained for the previous test case in Section 6.2, this application is simulated using the hybrid mean/block-mean imputation h_{MI-BMI} using $N_w = 15$ windows and a bunch width of $b = 250$.

Regarding SVD, the case shares many properties with the example discussed in Section 6.2, particularly the presence of input and output boundaries where particles can enter and exit the domain. However, unlike the previous scenario, no fully developed state is reached where the number of active liquid particles remains virtually constant, but the total number of simultaneously active particles at the end of the simulation grows to over 620,000. As a result, the reduced system exhibits a fully unsteady behavior. The 3D test case is also significantly more computationally intensive and is therefore computed using a thread-parallel approach on one GPU.

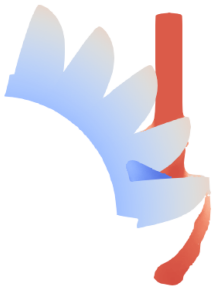
Figure 15 displays a visual comparison of the full-order SPH data (left) and the reconstructed jet (right) extracted at three different exemplary time instants. This qualitative comparison shows an excellent agreement. Note that only the fluid particles are reconstructed from the SVD and no penetration by the reconstructed fluid particles into the non-reconstructed/prescribed bucket region is observed.

The quality of the reconstruction by the ROM is assessed according to Equation (31) and shown in Figure 16. Attention is confined to the error obtained for reconstructing the pressure, velocity, and position fields. Since the pressure and the density are related according to the algebraic equation of state (5), storing both fields in the snapshot matrix is strictly speaking not necessary. Figure 16 reveals an excellent agreement between FOM and ROM, as expected from the visual comparison of the reconstructed and the simulated water body in Figure 15. With only a few exceptions, maximum relative error magnitudes inside the fluid remain below one per cent.

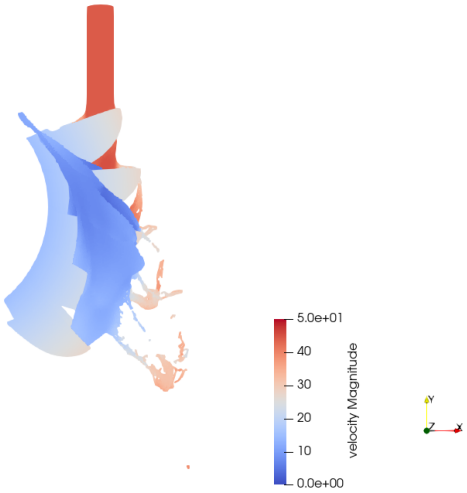
With regard to pressure, however, another detail comes to the fore, which concerns the solid pressures and their reconstruction. The solid pressures in SPH are usually determined from the fluid properties using boundary-specific formulae and the algebraic equation of state (5). The pressure levels can quickly be in the range of several million Pascals in wetted regions, but fall back to zero Pascal in (temporarily) unwetted solid areas. The reconstruction of the pressure field by the SVD is therefore more challenging for the solid than in the fluid particles. In essence, the snapshot matrix for the pressure of a wall particle resembles a random matrix with many zero entries, and even when maintaining a large number of modes, it is difficult to maintain the same accuracy as for the fluid particles. For this reason, the adaptive truncation approach would lead to storing almost all the available modes. Therefore, we decided to use a fixed truncation rank for the pressure field, which allows assigning the degree of compression but not the resulting accuracy. The accuracy of the reconstruction is evaluated based on the maximum and mean error, as well as for the integral quantities force and moment. Alternatively, analogous to the procedure for the FOM SPH method, the reconstructed fluid pressure can be determined in a post-processing step, which appears to be much simpler. Nonetheless, we obtain the ROM pressure field for the solid from the SVD of the FOM solid pressures in the present study, to evaluate the forces and torques on the turbine. Mind that only a subset



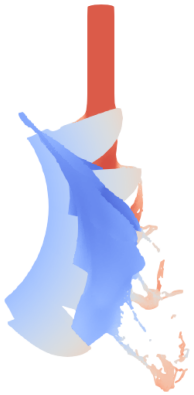
(a) FOM time step = 6000



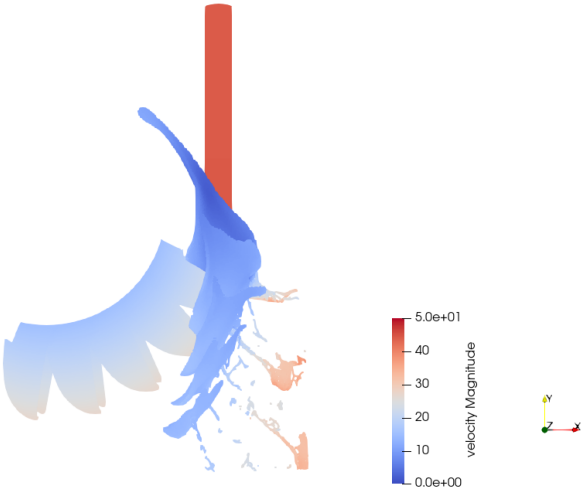
(b) ROM time step = 6000



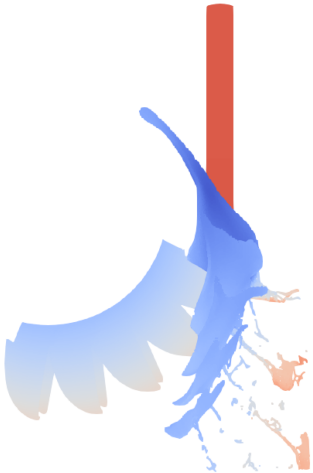
(c) FOM time step = 16000



(d) ROM time step = 16000



(e) FOM time step = 26000



(f) ROM time step = 26000

FIGURE 15 | Snap-shots of the particle positions colored by velocity magnitude obtained from the original FOM (left; SPH) and the SVD-based reconstruction (right) of the 3D Pelton runner flow at three different time instants. [Colour figure can be viewed at wileyonlinelibrary.com]

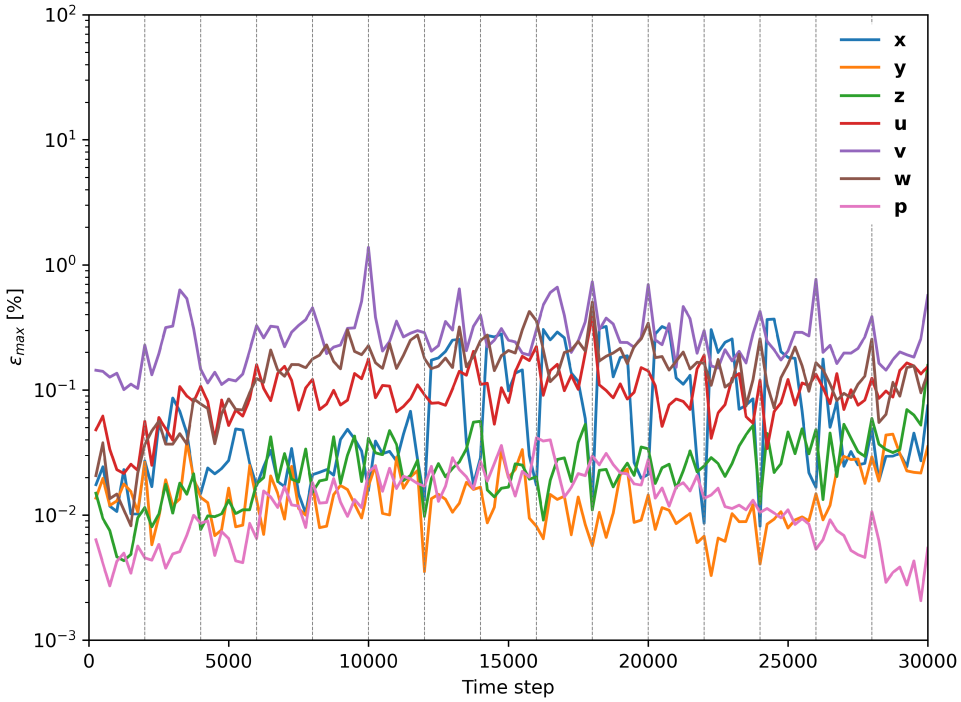


FIGURE 16 | 3D Pelton runner: Temporal evolution of the maximum reconstruction error of fluid field data in percent obtained with adaptive truncation, h_{MI-BMI} imputation, employing a bunch width of $b = 250$ and $N_w = 15$ (equally sized) time windows/SVDs indicated by the dashed vertical lines. [Colour figure can be viewed at wileyonlinelibrary.com]

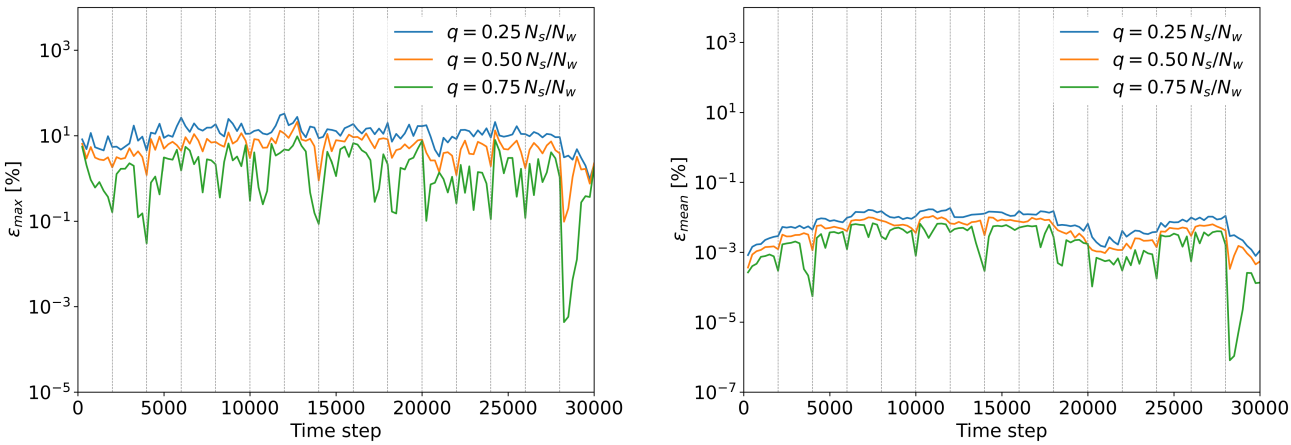


FIGURE 17 | 3D Pelton runner: Temporal evolution of the maximum (left) and mean (right) reconstruction error of the solid particle pressures in percent obtained with three fixed truncation ranks q, employing a bunch width of $b = 250$ and $N_w = 15$ (equally sized) time windows/SVDs. [Colour figure can be viewed at wileyonlinelibrary.com]

of solid particles interacts with water during the simulation; for those that do not interact, the pressure value remains zero. Solid particles are included in their corresponding bunch matrix only if they have interacted with fluid particles at least once. This avoids storing irrelevant data. Solid particles are then handled similarly to newly injected fluid particles, with the distinction that imputation is unnecessary since no data is missing.

Figure 17 depicts the maximum (left) and mean (right) reconstruction errors obtained for the solid pressure using three different (fixed) truncation ranks in the range of 25%–75% of the maximum possible rank. The irregularity of the snapshot matrix of the solid pressures mainly concerns the maximum error. The mean error, however, remains relatively small, which suggests that the rank truncation primarily impairs the local accuracy of the wall pressure. The latter is confirmed by evaluating the reconstruction of integral quantities, that is, forces and torques, as shown in Figures 18 and 19, revealing an error is in the order of per mille.

The attained CR for the Pelton runner case is evaluated against the full-order full-storage effort in Table 6. This assessment is performed separately for the fluid and solid particles and subsequently collectively. The tabulated values for the fluid data are of the same order of magnitude as those in the 2D cases,

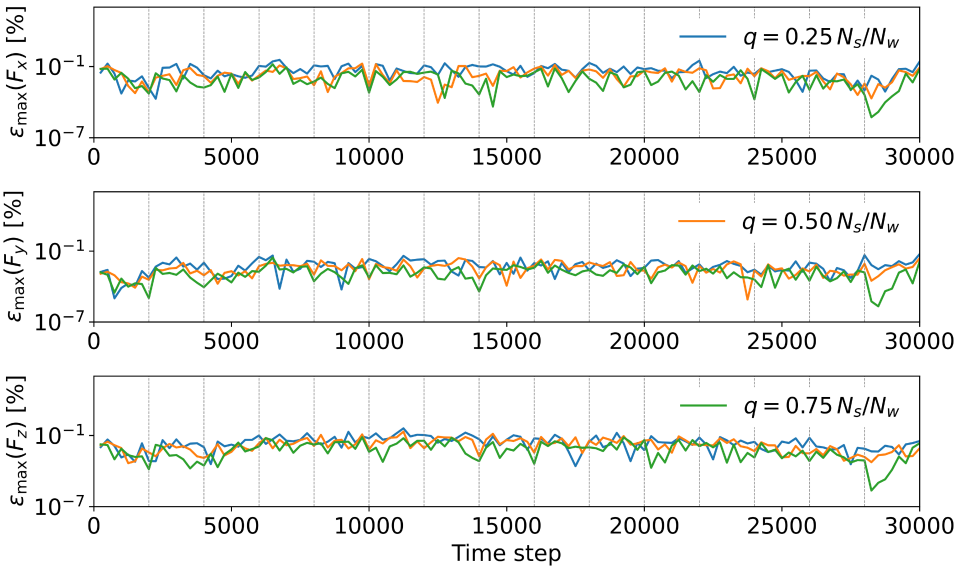


FIGURE 18 | 3D Pelton runner: Temporal evolution of the reconstruction error of the force in percent obtained with three fixed truncation ranks q, employing a bunch width of $b = 250$ and $N_w = 15$ (equally sized) time windows/SVDs. [Colour figure can be viewed at wileyonlinelibrary.com]

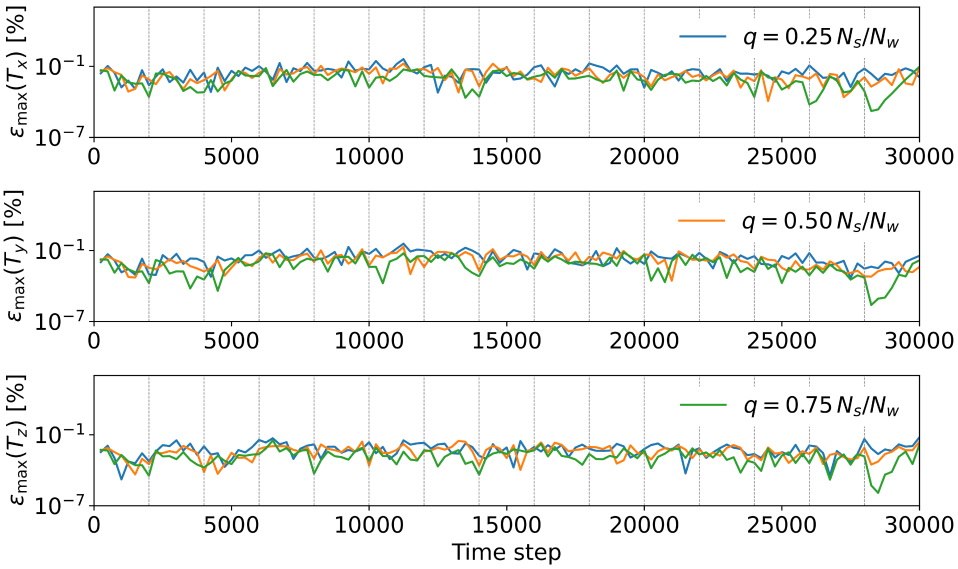


FIGURE 19 | 3D Pelton runner: Temporal evolution of the reconstruction error of the torque in percent obtained with three fixed truncation ranks q, employing a bunch width of $b = 250$ and $N_w = 15$ (equally sized) time windows/SVDs indicated by the dashed vertical lines. [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 6 | Compression ratio (CR) of the reduced-order model obtained for the 3D Pelton runner case normalized with the full-order full-storage effort. Result were obtained with adaptive truncation and h_{MI-BMI} imputation for the fluid, fixed truncation $q = 0.25N_s/N_w$ for the solid, employing a bunch width of $b = 250$ and $N_w = 15$ equally sized time windowed SVDs.

	Fluid	Solid	Total
CR	9.82	3.42	7.55

cf. for example Table 4, and the experienced CR reduction can be attributed the ROM of the solid pressure. Mind that post-processing the ROM for the solid pressures from the ROM fluid data in line with the employed boundary condition could represent a more efficient strategy. The computational overhead is $t^{SVD}/t^{SPH} \approx 7.26\%$ for the considered simulation which agrees with best practice results obtained for the 2D impinging jet case depicted in Table 5.

7 | Conclusions

An incremental SVD-based strategy has been developed to reduce transient SPH results. The approach is based on an existing incremental SVD algorithm that was originally devised for mesh-based regular snapshot data matrices, and is further developed here for processing highly irregular data matrices associated with Lagrangian particle simulation methods.

The system of equations from the FOM does not influence the development of the Reduced-Order Model (ROM) itself and therefore allows its application to other numerical methods characterized by irregular spatio/temporal data. The method has been embedded into a parallel, industrialized in-house SPH software (*Asphodel*) and was successfully validated for a sequence of relevant 2D and 3D test cases. The suggested approach reduces storage requirements by $\mathcal{O}(90\%)$ with good agreement between ROM and FOM data while maintaining reasonable computational overhead in the order of $\mathcal{O}(10\%)$. For the final water turbine application, the temporal evolution of force and torque values returned by the ROM is in excellent agreement with FOM recordings.

Various aspects important for accuracy, computational effort, and memory requirements were addressed, such as an adaptive rank truncation approach, the imputation strategy for gaps in the snapshot matrix caused by temporarily inactive particles and the sequencing of the data history into temporal windows as well as the bunching of the SVD updates. To ensure accuracy, an appropriate threshold was identified from the decrease in singular values, where the SVD was truncated when the values decreased by five orders of magnitude compared to the dominant singular value. Different imputation strategies have been tested for missing data in the data matrices, and a hybrid mean/block-mean method has been found to be the most effective approach that outperforms the gappy POD method. The suggested windowing strategy supports the effective management of scenarios with variable dynamics.

Author Contributions

Eduardo Di Costanzo: conceptualization, methodology, software, validation, formal analysis, investigation, writing – original draft, visualization. **Niklas Kühl:** conceptualization, software, resources, writing – review and editing, supervision, project administration. **Jean-Christophe Marongiu:** resources, writing – review, project administration, funding acquisition. **Thomas Rung:** conceptualization, resources, writing – review and editing, supervision, project administration, funding acquisition.

Acknowledgments

The first author was supported by the State Secretariat for Education, Research and Innovation (SERI) of Switzerland and Co-Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them. Open Access funding enabled and organized by Projekt DEAL.

Data Availability Statement

Data sharing not applicable, since no new data was generated, but only standard 2D testcases were used to scrutinze the method. For the final 3D testcase, the employed geometry is confidential, but the method essentially works for any irregular data matrix.

References

1. L. Sirovich, "Turbulence and the Dynamics of Coherent Structures. I. Coherent Structures," *Quarterly of Applied Mathematics* 45, no. 3 (1987): 561–571, https://doi.org/10.1090/qam/910462.

2. P. Benner, S. Gugercin, and K. Willcox, "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems," *SIAM Review* 57, no. 4 (2015): 483–531, https://doi.org/10.1137/130932715.

3. M. Abbaszadeh, M. Dehghan, and I. Navon, "A Proper Orthogonal Decomposition Variational Multiscale Meshless Interpolating Element Free Galerkin Method for Incompressible Magnetohydrodynamics Flow," *International Journal for Numerical Methods in Fluids* 92, no. 4 (2020): 1415–1436, https://doi.org/10.1002/fld.4834.

4. F. Ballarin and G. Rozza, "Pod-Galerkin Monolithic Reduced Order Models for Parametrized Fluid-Structure Interaction Problems: Pod-Galerkin Monolithic Rom for Parametrized FSI Problems," *International Journal for Numerical Methods in Fluids* 82, no. 12 (2016): 1010–1034, https://doi.org/10.1002/fld.4252.

5. K. Kunisch and S. Volkwein, "Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics," *SIAM Journal on Numerical Analysis* 40, no. 2 (2002): 492–515, https://doi.org/10.1137/S0036142900382612.

6. T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza, "Model Order Reduction in Fluid Dynamics: Challenges and Perspectives," in *Reduced Order Methods for Modeling and Computational Reduction* (Springer International Publishing, 2014), 235–273, https://doi.org/10.1007/978-3-319-02090-7_9.

7. Y. Nakamura, S. Sato, and N. Ohnishi, "Application of Proper Orthogonal Decomposition to Flow Fields Around Various Geometries and Reduced-Order Modeling," *Computer Methods in Applied Mechanics and Engineering* 432 (2024): 117340, https://doi.org/10.1016/j.cma.2024.117340.

8. K. Willcox and J. Peraire, "Balanced Model Reduction via the Proper Orthogonal Decomposition," *AIAA Journal* 40, no. 11 (2002): 2323–2330, https://doi.org/10.2514/2.1570.

9. M. Xiao, J. Ma, X. Gao, et al., "Primal–Dual On-The-Fly Reduced-Order Modeling for Large-Scale Transient Dynamic Topology Optimization," *Computer Methods in Applied Mechanics and Engineering* 428 (2024): 117099, https://doi.org/10.1016/j.cma.2024.117099.

10. M. Brand, "Incremental Singular Value Decomposition of Uncertain Data With Missing Values," in *Proceedings of the 7th European Conference on Computer Vision-Part I, ECCV '02* (Springer-Verlag, 2002), 707–720.

11. M. Brand, "Fast Low-Rank Modifications of the Thin Singular Value Decomposition," *Linear Algebra and Its Applications* 415, no. 1 (2006): 20–30, https://doi.org/10.1016/j.laa.2005.07.021.

12. H. Fareed and J. R. Singler, "A Note on Incremental Pod Algorithms for Continuous Time Data," *Applied Numerical Mathematics* 144 (2019): 223–233, https://doi.org/10.1016/j.apnum.2019.04.020.

13. H. Fareed, J. R. Singler, Y. Zhang, and J. Shen, "Incremental Proper Orthogonal Decomposition for PDE Simulation Data," *Computers & Mathematics With Applications* 75, no. 6 (2018): 1942–1960, https://doi.org/10.1016/j.camwa.2017.09.012.

14. N. Kühl, "Incremental Singular Value Decomposition Based Model Order Reduction of Scale Resolving Fluid Dynamic Simulations," *Computers & Fluids* 291 (2025): 106579, https://doi.org/10.1016/j.compfluid.2025.106579.

15. N. Kühl, H. Fischer, M. Hinze, and T. Rung, "An Incremental Singular Value Decomposition Approach for Large-Scale Spatially Parallel & Distributed but Temporally Serial Data – Applied to Technical Flows," *Computer Physics Communications* 296 (2024): 109022, https://doi.org/10.1016/j.cpc.2023.109022.

16. X. Li, S. Hulshoff, and S. Hickel, "Towards Adjoint-Based Mesh Refinement for Large Eddy Simulation Using Reduced-Order Primal Solutions: Preliminary 1D Burgers Study," *Computer Methods in Applied Mechanics and Engineering* 379 (2021): 113733, https://doi.org/10.1016/j.cma.2021.113733.

17. X. Li, J. R. Singler, and X. He, "Incremental Data Compression for PDE-Constrained Optimization With a Data Assimilation Application," arXiv Preprint arXiv:2404.09323, (2024), https://doi.org/10.48550/arXiv.2404.09323.

18. A.-S. I. Margetis, E. M. Papoutsis-Kiachagias, and K. C. Giannakoglou, "Lossy Compression Techniques Supporting Unsteady Adjoint on 2d/3d Unstructured Grids," *Computer Methods in Applied Mechanics and Engineering* 387 (2021): 114152, https://doi.org/10.1016/j.cma.2021.114152.

19. A.-S. I. Margetis, E. M. Papoutsis-Kiachagias, and K. C. Giannakoglou, "Reducing Memory Requirements of Unsteady Adjoint by Synergistically Using Check-Pointing and Compression," *International Journal for Numerical Methods in Fluids* 95 (2022): 23–43, https://doi.org/10.1002/fld.5136.

20. C. Vezyris, E. M. Papoutsis-Kiachagias, and K. C. Giannakoglou, "On the Incremental Singular Value Decomposition Method to Support Unsteady Adjoint-Based Optimization," *International Journal for Numerical Methods in Fluids* 91, no. 7 (2019): 315–331, https://doi.org/10.1002/fld.4755.

21. J. J. Monaghan, "Simulating Free Surface Flows With SPH," *Journal of Computational Physics* 110, no. 2 (1994): 399–406, https://doi.org/10.1006/jcph.1994.1034.

22. A. Colagrossi, M. Antuono, and D. Le Touzé, "Theoretical Considerations on the Free-Surface Role in the Smoothed-Particle-Hydrodynamics Model," *Physical Review E* 79 (2009): 1–13, https://doi.org/10.1103/PhysRevE.79.056701.

23. E. Nowoghomwenma, S. Maxutov, and Y. C. Lee, "Modelling Surface Tension of Two-Dimensional Droplet Using Smoothed Particle Hydrodynamics: Modelling Surface Tension of Two-Dimensional Droplet Using SPH," *International Journal for Numerical Methods in Fluids* 88, no. 6 (2018): 334–346, https://doi.org/10.1002/fld.4663.

24. B. Zheng, L. Sun, Z. Chen, C. Cheng, and C. Liu, "Multiphase SPH Modeling of Forced Liquid Sloshing," *International Journal for Numerical Methods in Fluids* 93, no. 7 (2020): 411–428, https://doi.org/10.1002/fld.4889.

25. C. Ulrich, M. Leonardi, and T. Rung, "Multi-Physics SPH Simulation of Complex Marine-Engineering Hydrodynamic Problems," *Ocean Engineering* 64 (2013): 109–121, https://doi.org/10.1016/j.oceaneng.2013.02.007.

26. S. Cummins, T. Silvester, and P. Cleary, "Three-Dimensional Wave Impact on a Rigid Structure Using Smoothed Particle Hydrodynamics," *International Journal for Numerical Methods in Fluids* 68, no. 7 (2012): 1471–1496, https://doi.org/10.1002/fld.2539.

27. N. Salis, X. Hu, M. Luo, A. Reali, and S. Manenti, "3D SPH Analysis of Focused Waves Interacting With a Floating Structure," *Applied Ocean Research* 144 (2024): 103885, https://doi.org/10.1016/j.apor.2024.103885.

28. A. Zhang, P. Sun, F. Ming, and A. Colagrossi, "Smoothed Particle Hydrodynamics and Its Applications in Fluid-Structure Interactions," *Journal of Hydrodynamics* 29 (2017): 187–216, https://doi.org/10.1016/S1001-6058(16)60730-8.

29. T. N. Hoang, T. T. Nguyen, T. V. Nguyen, G. D. Nguyen, and H. H. Bui, "SPH Simulation of Earthquake-Induced Liquefaction and Large Deformation Behaviour of Granular Materials Using Sanisand Constitutive Model," *Computers and Geotechnics* 174 (2024): 106617, https://doi.org/10.1016/j.compgeo.2024.106617.

30. M. Sauro, S. Nicolò, and T. Sara, "3D Wcsph Modelling of Landslide-Water Dynamics During 1963 Vajont Disaster," *Scientific Reports* 14, no. 11 (2024): 27504, https://doi.org/10.1038/s41598-024-77922-5.

31. S. Capasso, B. Tagliafierro, I. Martínez-Estévez, et al., "Development of an Sph-Based Numerical Wave–Current Tank and Application to Wave Energy Converters," *Applied Energy* 377 (2025): 124508, https://doi.org/10.1016/j.apenergy.2024.124508.

32. Y. Yang, A. English, B. D. Rogers, et al., "Numerical Modelling of a Vertical Cylinder With Dynamic Response in Steep and Breaking Waves Using Smoothed Particle Hydrodynamics," *Journal of Fluids and Structures* 125 (2024): 104049, https://doi.org/10.1016/j.jfluidstructs.2023.104049.

33. J.-C. Marongiu, F. Leboeuf, J. Caro, and E. Parkinson, "Free Surface Flows Simulations in Pelton Turbines Using an Hybrid Sph-Ale Method," *Journal of Hydraulic Research* 48, no. 1 (2010): 40–49, https://doi.org/10.3826/jhr.2010.0002.

34. M. Rentschler, M. Neuhauser, J.-C. Marongiu, and E. Parkinson, "Understanding Casing Flow in Pelton Turbines by Numerical Simulation," *IOP Conference Series: Earth and Environmental Science* 49, no. 11 (2016): 022004, https://doi.org/10.1088/1755-1315/49/2/022004.

35. C. Pilloton, P. N. Sun, X. Zhang, and A. Colagrossi, "Volume Conservation Issue Within SPH Models for Long-Time Simulations of Violent Free-Surface Flows," *Computer Methods in Applied Mechanics and Engineering* 419 (2024): 116640, https://doi.org/10.1016/j.cma.2023.116640.

36. B. Zhang, N. Adams, and X. Hu, "Towards High-Order Consistency and Convergence of Conservative SPH Approximations," *Computer Methods in Applied Mechanics and Engineering* 433 (2025): 117484, https://doi.org/10.1016/j.cma.2024.117484.

37. P. Rastelli, R. Vacondio, and J. C. Marongiu, "An Arbitrarily Lagrangian–Eulerian SPH Scheme With Implicit Iterative Particle Shifting Procedure," *Computer Methods in Applied Mechanics and Engineering* 414 (2023): 116159, https://doi.org/10.1016/j.cma.2023.116159.

38. J. Michel, M. Antuono, G. Oger, and S. Marrone, "Energy Balance in Quasi-Lagrangian Riemann-Based SPH Schemes," *Computer Methods in Applied Mechanics and Engineering* 410 (2023): 116015, https://doi.org/10.1016/j.cma.2023.116015.

39. J. Leduc, "Etude physique et numérique de l'écoulement dans un dispositif d'injection de turbine Pelton," PhD Thesis, 12, ((2010)).

40. J.-C. Marongiu, "Méthode numérique lagrangienne pour la simulation d'écoulements à surface libre: application aux turbines Pelton," PhD Thesis, (2007), http://www.theses.fr/2007ECDL0046.

41. M. Neuhauser, "Development of a Coupled SPH-ALE/Finite Volume Method for the Simulation of Transient Flows in Hydraulic Machines," Theses, Ecole Centrale de Lyon, December, (2014), https://theses.hal.science/tel-01184761.

42. M. Rentschler, J.-C. Marongiu, M. Neuhauser, and E. Parkinson, "Overview of Sph-Ale Applications for Hydraulic Turbines in Andritz Hydro," *Journal of Hydrodynamics* 30, no. 2 (2018): 114–121, https://doi.org/10.1007/s42241-018-0012-y.

43. J. P. Vila, "On Particle Weighted Methods and Smoothed Particle Hydrodynamics," *Mathematical Models & Methods in Applied Sciences - M3AS* 09, no. 3 (1999): 161–209, https://doi.org/10.1142/S0218202599000117.

44. H. Wendland, "Piecewise Polynomial, Positive Definite and Compactly Supported Radial Functions of Minimal Degree," *Advances in Computational Mathematics* 4, no. 1 (1995): 389–396, https://doi.org/10.1007/BF02123482.

45. L. Selle, F. Nicoud, and T. Poinsot, "Actual Impedance of Nonreflecting Boundary Conditions: Implications for Computation of Resonators," *AIAA Journal* 42, no. 5 (2004): 958–964, https://doi.org/10.2514/1.1883.

46. C. Eckart and G. Young, "The Approximation of One Matrix by Another of Lower Rank," *Psychometrika* 1, no. 3 (1936): 211–218, https://doi.org/10.1007/BF02288367.

47. S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2019), https://doi.org/10.1017/9781108380690.

48. K. Taira, S. L. Brunton, S. T. M. Dawson, et al., "Modal Analysis of Fluid Flows: An Overview," *AIAA Journal* 55, no. 12 (2017): 4013–4041, https://doi.org/10.2514/1.J056060.

49. A. Kaur, s. Martha, and A. Chakrabarti, "Linear Algebraic Method of Solution for the Problem of Mitigation of Wave Energy Near Seashore by Trench-Type Bottom Topography," *Journal of Engineering Mechanics* 146, no. 9 (2020): 04020125, https://doi.org/10.1061/(ASCE)EM.1943-7889.0001856.

50. A. Kaur, S. Martha, and A. Chakrabarti, "An Algebraic Method of Solution of a Water Wave Scattering Problem Involving an Asymmetrical Trench," *Computational and Applied Mathematics* 39 (2020): 229–248, https://doi.org/10.1007/s40314-020-01255-y.

51. K. Tenekedjiev, N. Abdussamie, H. An, and N. Nikolova, "Regression Diagnostics With Predicted Residuals of Linear Model With Improved Singular Value Classification Applied to Forecast the Hydrodynamic Efficiency of Wave Energy Converters," *Applied Sciences* 11, no. 7 (2021): 2990, https://doi.org/10.3390/app11072990.

52. C. Gräßle and M. Hinze, "Pod Reduced Order Modeling for Evolution Equations Utilizing Arbitrary Finite Element Discretizations," *Advances in Computational Mathematics* 44, no. 12 (2018): 1941–1978, https://doi.org/10.1007/s10444-018-9620-x.

53. M. Gubisch and S. Volkwein, *Chapter 1: Proper Orthogonal Decomposition for Linear-Quadratic Optimal Control* (Society for Industrial & Applied Mathematics, 2017), 3–63, https://doi.org/10.1137/1.9781611974829.ch1.

54. P. Saini, C. Arndt, and A. Steinberg, "Development and Evaluation of Gappy-POD as a Data Reconstruction Technique for Noisy PIV Measurements in Gas Turbine Combustors," *Experiments in Fluids* 57 (2016): 122, https://doi.org/10.1007/s00348-016-2208-7.

55. B. Tan Bui-Thanh, M. Damodaran, and K. Willcox, "Aerodynamic Data Reconstruction and Inverse Design Using Proper Orthogonal Decomposition," *AIAA Journal* 42, no. 8 (2004): 1505–1516, https://doi.org/10.2514/1.2159.

56. K. Willcox, "Unsteady Flow Sensing and Estimation via the Gappy Proper Orthogonal Decomposition," *Computers & Fluids* 35, no. 2 (2006): 208–226, https://doi.org/10.1016/j.compfluid.2004.11.006.

57. Y. Duan, J. Cai, and Y. Li, "Gappy Proper Orthogonal Decomposition-Based Two-Step Optimization for Airfoil Design," *AIAA Journal* 50, no. 4 (2012): 968–971, https://doi.org/10.2514/1.J050997.

58. B. A. Freno and P. G. A. Cizmas, "A Proper Orthogonal Decomposition Method for Nonlinear Flows With Deforming Meshes," *International Journal of Heat and Fluid Flow* 50 (2014): 145–159, https://doi.org/10.1016/j.ijheatfluidflow.2014.07.001.

59. R. Everson and L. Sirovich, "Karhunen–Loève Procedure for Gappy Data," *Journal of the Optical Society of America. A, Optics, Image Science, and Vision* 12, no. 8 (1995): 1657–1664, https://doi.org/10.1364/JOSAA.12.001657.

60. H. Gunes, S. Sirisup, and G. Karniadakis, "Gappy Data: To Krig or not to Krig?," *Journal of Computational Physics* 212 (2006): 358–382, https://doi.org/10.1016/j.jcp.2005.06.023.

61. S. Raben, J. Charonko, and P. Vlachos, "Adaptive Gappy Proper Orthogonal Decomposition for Particle Image Velocimetry Data Reconstruction," *Measurement Science and Technology* 23 (2012): 025303, https://doi.org/10.1088/0957-0233/23/2/025303.

62. L. Blackford, J. Choi, A. Cleary, et al., *ScaLAPACK Users' Guide* (SIAM, 1997).

63. V. Hernandez, J. E. Roman, and V. Vidal, "SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems," *ACM Transactions on Mathematical Software* 31, no. 3 (2005): 351–362, https://doi.org/10.1145/1089014.1089019.

64. A. R. Benson, D. F. Gleich, and J. Demmel, "Direct QR Factorizations for Tall-And-Skinny Matrices in MapReduce Architectures," in *Proceedings of the 2013 IEEE International Conference on Big Data* (IEEE, 2013), 264–272, https://doi.org/10.1109/BigData.2013.6691583.

65. J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-Optimal Parallel and Sequential QR and LU Factorizations," *SIAM Journal on Scientific Computing* 34, no. 1 (2012): A206–A239, https://doi.org/10.1137/080731992.

66. T. Sayadi and P. Schmid, "Parallel Data-Driven Decomposition Algorithm for Large-Scale Datasets: With Application to Transitional Boundary Layers," *Theoretical and Computational Fluid Dynamics* 30, no. 10 (2016): 415–428, https://doi.org/10.1007/s00162-016-0385-x.

67. C. Bach, D. Ceglia, L. Song, and F. Duddeck, "Randomized Low-Rank Approximation Methods for Projection-Based Model Order Reduction of Large Nonlinear Dynamical Problems," *International Journal for Numerical Methods in Engineering* 118, no. 12 (2018): 209–241, https://doi.org/10.1002/nme.6009.

68. H. Fareed and J. Singler, "Error Analysis of an Incremental Proper Orthogonal Decomposition Algorithm for PDE Simulation Data," *Journal of Computational and Applied Mathematics* 368 (2019): 112525, https://doi.org/10.1016/j.cam.2019.112525.

69. G. Guennebaud and B. Jacob, "Eigen v3," (2010), http://eigen.tuxfamily.org.

70. M. Antuono, P. N. Sun, S. Marrone, and A. Colagrossi, "The δ-Ale-Sph Model: An Arbitrary Lagrangian-Eulerian Framework for the δ-Sph Model With Particle Shifting Technique," *Computers & Fluids* 216 (2021): 104806, https://doi.org/10.1016/j.compfluid.2020.104806.

71. H. S. Yoo, Y. B. Jo, and E. S. Kim, "Comparative Study of Wcsph, Eisph and Explicit Incompressible-Compressible SPH (Eicsph) for Multi-Phase Flow With High Density Difference," *Journal of Computational Physics* 506 (2024): 112930, https://doi.org/10.1016/j.jcp.2024.112930.