**ORIGINAL ARTICLE**

# Dynamic mode decomposition in adaptive mesh refinement and coarsening simulations

Gabriel F. Barros[1] · Malú Grave[1] · Alex Viguerie[2] · Alessandro Reali[3] · Alvaro L. G. A. Coutinho[1]

## Abstract

Dynamic mode decomposition (DMD) is a powerful data-driven method used to extract spatio-temporal coherent structures that dictate a given dynamical system. The method consists of stacking collected temporal snapshots into a matrix and mapping the nonlinear dynamics using a linear operator. The classical procedure considers that snapshots possess the same dimensionality for all the observable data. However, this often does not occur in numerical simulations with adaptive mesh refinement/coarsening schemes (AMR/C). This paper proposes a strategy to enable DMD to extract features from observations with different mesh topologies and dimensions, such as those found in AMR/C simulations. For this purpose, the adaptive snapshots are projected onto the same reference function space, enabling the use of snapshot-based methods such as DMD. The present strategy is applied to challenging AMR/C simulations: a continuous diffusion–reaction epidemiological model for COVID-19, a density-driven gravity current simulation, and a bubble rising problem. We also evaluate the DMD efficiency to reconstruct the dynamics and some relevant quantities of interest. In particular, for the SEIRD model and the bubble rising problem, we evaluate DMD's ability to extrapolate in time (short-time future estimates).

✉ Alessandro Reali
alereali@unipv.it

Gabriel F. Barros
gabriel.barros@coc.ufrj.br

Malú Grave
malugrave@nacad.ufrj.br

Alex Viguerie
alexander.viguerie@gssi.it

Alvaro L. G. A. Coutinho
alvaro@nacad.ufrj.br

1 Department of Civil Engineering, COPPE/Federal University of Rio de Janeiro, P.O. Box 68506, 21945-970 Rio de Janeiro, RJ, Brazil

2 Department of Mathematics, Gran Sasso Science Institute, Viale Francesco Crispi 7, L'Aquila, AQ 67100, Italy

3 Dipartimento di Ingegneria Civile ed Architettura, Università di Pavia, Via Ferrata 3, Pavia, PV 27100, Italy

## 1 Introduction

Data-driven methods are currently revolutionizing the modeling, prediction, and control of complex systems. Increasingly, researchers are considering data-driven approaches for a diverse range of complex systems, such as turbulent flows, climate sciences, epidemiology, finance, robotics, and many other different applications [12]. Even with the availability of better hardware and advances in techniques and algorithms, numerical simulations of these systems are still resource-demanding: strong nonlinearities, multiple scales, and large dimensionalities are typical examples of complexities found in modern applications. With the assembly of modern mathematical methods, unprecedented data availability, and increasing computational resources, previously complex, challenging problems can now be tackled within the new research field entitled scientific machine learning (SciML).

SciML is a core component of artificial intelligence and computational technology that can be trained, with scientific data, to augment or automate human skills [6]. This emerging research area aims at the opportunities and challenges

in the context of complex applications across science and engineering, and other interdisciplinary fields. A wide range of SciML methods can be categorized regarding the type of information available, and their intended use [13]. For instance, many state-of-the-art methods and applications within the SciML scope can be found in [13, 36, 45, 46, 52, 60]. In this study, we focus on Dynamic Mode Decomposition (DMD), an unsupervised SciML method that can extract the most dynamically relevant low-rank structures from large-dimensional data observed in dynamical systems. DMD can be applied to both numerical [56] and experimental data [57].

The standard DMD procedure for numerical simulations consists of stacking the $m$ snapshots (discrete solutions in space for a given time step) in columns to create a matrix and map the dynamics using a linear operator. The DMD procedure assumes that the $m$ collected snapshots have spatial dimensionality $n$, where $n \gg m$, such that the snapshots matrix has dimension $n \times m$. This occurs when snapshots are obtained from experimental data with sensors in fixed positions or numerical simulations considering a fixed mesh. However, in many situations, this is not always achievable. For numerical simulations using Adaptive Mesh Refinement/ Coarsening (AMR/C), for instance, spatial adaptivity leads to solutions computed in meshes that constantly change in time. Adaptive meshes lead to a different number of nodes, nodal coordinates and numbering, and mesh topologies. Although there are strategies regarding missing spatial data or subsampling for the DMD [14, 41], specific treatments related to AMR/C approaches in the finite element context are not explored in the literature. In the present study, we develop a strategy to project all the snapshots of a given simulation with different dimensionality onto a reference target mesh with minor accuracy loss, enabling the use of any SVD-based data-driven technique such as DMD.

This paper is structured as follows: Sect. 2 describes the relation between the discretization of PDEs in space and time and a dynamical system. This section introduces Dynamic Mode Decomposition, our method of choice for short-time future estimates and extrapolation. Section 3 describes our strategy to deal with simulations that consider AMR/C in their evolution (e.g., in the case of dimensionality of the output vector, as well as of mesh topology and/or node numbering, varying in time). In Sect. 4 we describe the numerical applications in this study: the use of DMD on a continuous SEIRD model for COVID-19 and two fluid dynamics problems, a density-driven gravity current, and a bubble rising problem. We show efficiency and accuracy results for the signal reconstruction. Moreover, future time step predictions using DMD are evaluated for the SEIRD model and the bubble rising problem. In Sect. 5, we draw our final remarks and conclusions.

## 2 Numerical methods and dynamic mode decomposition

Solving partial differential equations (PDEs) using fast, accurate, reliable, and robust methods is crucial for many industrial and scientific applications. Several methods (such as finite elements, finite differences, and many others) are responsible for approximating the infinite-dimensional PDEs into finite-dimensional spaces. The discretization of these equations allows the process to be automated. In the present study, we focus on using the finite element method for spatial discretization of the PDEs. That is, consider a generic transient parametric PDE such as

$$\frac{\partial \mathbf{u}}{\partial t} + \mathcal{N}(\mathbf{u};\boldsymbol{\zeta}) = \mathbf{f}, \text{ in } \Omega \times (0, T], \tag{1}$$

where $\mathcal{N}$ is a nonlinear operator, $\boldsymbol{\zeta}$ is a vector of parameters (e.g., diffusion, density, viscosity, etc.), $\mathbf{f} = \mathbf{f}(t;\boldsymbol{\zeta})$ is a given function and the solution $\mathbf{u}$ is a function of spatial coordinates $\mathbf{x}$, temporal coordinates $t$, and parameters $\zeta$ such that $\mathbf{u} = \mathbf{u}(\mathbf{x}, t;\boldsymbol{\zeta})$. The equation is equipped with boundary and initial conditions

$$\begin{aligned}
\mathbf{u} &= \mathbf{g} \text{ on } \Gamma_D \times (0, T], \\
\nabla \mathbf{u} \cdot \mathbf{n} &= \mathbf{h} \text{ on } \Gamma_N \times (0, T], \\
\mathbf{u}(\mathbf{x}, 0;\zeta) &= \mathbf{u}_0(\mathbf{x};\zeta) \text{ on} \tilde{\Omega},
\end{aligned} \tag{2}$$

where $\mathbf{g}$ and $\mathbf{h}$ are the Dirichlet and Neumann boundary conditions, respectively, $T$ is the final time, and $\mathbf{u}_0$ is the initial condition for $\mathbf{u}$. The domain $\Omega \subset \mathbb{R}^{nsd}$ is bounded by Lipschitz continuous boundaries $\Gamma_N \cup \Gamma_D = \Gamma \subset \mathbb{R}^{nsd-1}$, and $\mathbf{n}$ is the unit outward normal to $\Gamma_N$. The union of boundaries and domain is represented as $\tilde{\Omega}$. The standard finite element method consists of discretizing $\tilde{\Omega}$ into a mesh composed of nodes and elements. Each element has its domain $\Omega^e \subset \mathbb{R}^{nsd}$ and boundary $\Gamma^e \subset \mathbb{R}^{nsd-1}$. The weak form of the system can be obtained by integrating Eq. (1) in its strong form against a weighting function $\mathbf{w} \in H^1(\Omega)$, where $H^1(\Omega)$ is the Sobolev space of the square-integrable functions with an integrable first weak derivative, and applying the divergence theorem. Being $P^k(\Omega^e)$ the space of polynomials of degree equal or less than $k$ over $\Omega^e$, the function spaces are defined as

$$S_t^h = \{\mathbf{u}^h(\cdot, t) \in H^1(\Omega) \mid \mathbf{u}^h(\cdot, t)|_{\Omega_e} \in P^k(\Omega^e), \forall e\}, \tag{3}$$

$$W^h = \{\mathbf{w}^h \in H^1(\Omega) \mid \mathbf{w}^h|_{\Omega_e} \in P^k(\Omega^e), \forall e\}. \tag{4}$$

Therefore, the semi-discrete finite element formulation for Eq. (1) is: find $\mathbf{u}^h(t) \in S_t^h$ such that $\forall \mathbf{w}^h \in W^h$:

$$\left(\frac{\partial \mathbf{u}^h}{\partial t}, \mathbf{w}^h\right) + (\mathcal{N}(\mathbf{u}^h;\boldsymbol{\zeta}), \mathbf{w}^h) = (\mathbf{f}^h, \mathbf{w}^h), \tag{5}$$

where the $\mathcal{L}^2$ inner product over the domain $\Omega$ is indicated by $(\cdot, \cdot)$. The weak form given by Eq. (5) naturally accommodates several finite element formulations, from Galerkin to Variational Multiscale methods [1, 8, 19, 39, 53]. After the temporal discretization of Eq. (5), the equation can be translated into a discrete-time dynamic system. In this system, the state vector $\mathbf{u}^h$ at the time instant $k + 1$ can be written such that:

$$\mathbf{u}^h_{k+1} = \mathcal{F}(\mathbf{u}^h_k), \tag{6}$$

where $\mathcal{F}$ represents the discrete-time flow map of the system and incorporates information regarding the parameters $\zeta$, mesh size, solver tolerances, etc. In the present study, we consider that the measurements of the system are the state vectors themselves, that is, $\mathbf{u}^h_k$. Analyzing the evolution in time of a discretized PDE as a dynamical system is a key concept for introducing Dynamic Mode Decomposition (DMD).

DMD is an equation-free, data-driven method that provides accurate assessments of the dominant structures in a given complex system [43]. DMD provides a decomposition of data into spatio-temporal modes that correlates the data across spatial features and also associates them to unique temporal Fourier modes. The main idea of the method is to efficiently compute the regression of linear/nonlinear terms to a least-square linear dynamics approximation from experimental or numerical observable data. Despite its first appearance in the fluid dynamics context [55, 56], DMD has been used in many other applications such as epidemiology [51], biomechanics [16], urban mobility [3], climate [44] and aeroelasticity [28], especially in structure extraction from data and control-oriented methods.

We can now apply DMD on the dynamical system described in Eq. (6). Consider a dataset $\mathbf{Y}^h$ containing the observations in time of the dynamical system $\mathbf{u}^h_k$ for $k = 0, 1, \ldots, m$, where $m + 1$ is the total number of observations. The dataset

$$\mathbf{Y}^h = \begin{bmatrix} | & | & & | \\ \mathbf{u}^h_0 & \mathbf{u}^h_1 & \cdots & \mathbf{u}^h_m \\ | & | & & | \end{bmatrix} \tag{7}$$

can be split into two datasets $\mathbf{Y}^h_1 = [\mathbf{u}^h_0 \ldots \mathbf{u}^h_{m-1}] \in \mathbb{R}^{n \times m}$ and $\mathbf{Y}^h_2 = [\mathbf{u}^h_1 \ldots \mathbf{u}^h_m] \in \mathbb{R}^{n \times m}$. DMD consists on finding the best fit approximation of the linear mapping $\mathbf{A}$ that transforms dataset $\mathbf{Y}^h_1$ into dataset $\mathbf{Y}^h_2$, that is,

$$\mathbf{Y}^h_2 = \mathbf{A}\mathbf{Y}^h_1. \tag{8}$$

The computation of $\mathbf{A}$ can be done as $\mathbf{A} = \mathbf{Y}^h_2 \mathbf{Y}^{h\dagger}_1$, where $\mathbf{Y}^{h\dagger}_1$ is the Moore–Penrose pseudoinverse of $\mathbf{Y}^h_1$. However, we avoid the computation of the full matrix $\mathbf{A}$ since $\mathbf{A}$ is a $n \times n$ matrix. Also, the computation of the full Moore–Penrose

pseudoinverse is not advisable due to its ill-conditioning. Instead, we can compute the SVD of $\mathbf{Y}^h_1$ as

$$\mathbf{Y}^h_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{9}$$

where $\mathbf{U} \in \mathbb{R}^{n \times m}$ and $\mathbf{V} \in \mathbb{R}^{m \times m}$ are the left and right singular vectors and $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$ is a diagonal matrix with real, non-negative, and decreasing entries named singular values. The singular values $\sigma_0 \geq \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{m-1}$ are hierarchical and can be interpreted in terms of how much the singular vectors influence the original matrix $\mathbf{Y}^h_1$. For the DMD procedure, considering the Eckart–Young Theorem [23], the optimal low-rank update approximation matrix $\mathbf{Y}^h_1$, when subjected to a truncation rank $r$, can be written as

$$\mathbf{Y}^h_1 \approx \tilde{\mathbf{Y}}^h_1 = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}^T_r, \tag{10}$$

where $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ is a matrix containing the first $r$ columns of $\mathbf{U}$, $\mathbf{V}_r \in \mathbb{R}^{m \times r}$ contains the first $r$ columns of $\mathbf{V}$, and $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$ is the diagonal matrix containing the first $r$ singular values. The pseudoinverse can be approximated as

$$\mathbf{Y}^{h\dagger}_1 \approx \tilde{\mathbf{Y}}^{h\dagger}_1 = \mathbf{V}_r\mathbf{\Sigma}^{-1}_r\mathbf{U}^T_r \tag{11}$$

and, instead of computing $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can obtain $\tilde{\mathbf{A}}$, a $r \times r$ projection of $\mathbf{A}$ as

$$\tilde{\mathbf{A}} = \mathbf{U}^T_r\mathbf{A}\mathbf{U}_r = \mathbf{U}^T_r\mathbf{Y}^h_2\mathbf{V}_r\mathbf{\Sigma}^{-1}_r. \tag{12}$$

Note that $\tilde{\mathbf{A}}$ is unitarily similar to $\mathbf{A}$. Further mathematical details regarding the optimization problem (the best-fitting matrix $\tilde{\mathbf{A}}$) and the influence of the Eckart–Young Theorem on constraints of the problem can be found in [40]. Now we can compute the eigendecomposition of $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}W = W\Lambda, \tag{13}$$

where $\Lambda$ is a diagonal matrix containing the discrete eigenvalues $\lambda_j$ and the matrix $W$ contains the eigenvectors $\boldsymbol{\phi}_j$ of $\tilde{\mathbf{A}}$. The DMD basis can be written as

$$\mathbf{\Psi} = \mathbf{Y}^h_2\mathbf{V}_r\mathbf{\Sigma}^{-1}_r W, \tag{14}$$

and the signal reconstruction as

$$\mathbf{u}^h(t) \approx \tilde{\mathbf{u}}^h(t) = \mathbf{\Psi}\exp(\mathbf{\Omega}_{eig}t)\mathbf{b}, \tag{15}$$

being $\mathbf{b}$ the vector containing the projected initial conditions such that $\mathbf{b} = \mathbf{\Psi}^\dagger\mathbf{u}^h_0$, and $\mathbf{\Omega}_{eig}$ is a diagonal matrix whose entries are the continuous eigenvalues $\omega_i = \ln(\lambda_i)/\Delta t_o$, where $\Delta t_o$ is the time step size between the outputs. In the present study, $\Delta t_o = j\Delta t$, where $j = 1, 2, \ldots, m$, being $\Delta t$ the time step size used in the temporal integration of the PDEs. For instance, if one chooses to output the solutions once every two time steps, the time step size between the two observations will be two times larger than the time step size

used to compute time integration, that is, $\Delta t_o = 2\Delta t$. It is important to mention that the snapshots sampling frequency affects directly the DMD's ability to capture the dynamics. For lower dominant frequencies, a larger $\Delta t_o$ is more adequate, while smaller values of $\Delta t_o$ are required for capturing rapid dynamics [56]. The form of (15) can be regarded as a generalization of the Sturm–Liouville expansion for a differential problem:

$$u(t) = \sum_{i=0}^{\infty} b_i \psi_i e^{\omega_i t}, \tag{16}$$

where $\psi_i$ and $\omega_i$ are the $i$th Sturm–Liouville eigenfunctions and eigenvalues for a given differential operator.

DMD can be seen as a dimensionality reduction method due to its inherent ability to extract the $r$ most relevant dynamical modes, where $r$ is often much smaller than the snapshot matrix rank $m$. However, a strategy to determine the number of relevant modes is not straightforward, and is an active topic in DMD research [59]. Even though the choice of $r$ for DMD may require some trial and error, some techniques can be used to find a good starting point. A hard threshold technique [43] consists of choosing $r$ such that

$$\kappa = 1 - \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{m} \sigma_i^2} \le \tau, \tag{17}$$

where $\tau$ is a tolerance threshold, set, e.g., to $10^{-6}$. This method implies that more than $100(1 - \tau)\%$ of the variance in the data is retained by the approximation. For the case where DMD is used on experimental (or numerical but noisy) data, more sophisticated solutions are presented in the literature [22, 29].

Another important consideration when using DMD is the SVD algorithm. The SVD can represent a significant part of the computational effort, meaning that improvements in the SVD performance lead to significant CPU time gains. For many SVD-based methods (such as DMD), there is no need to compute the SVD for the whole matrix, since the method aims at extracting the first $r$ dominant structures in the matrix. Many algorithms are designed in this direction to make this computation more efficient. One important contribution in this direction is seen in [58], where the method of snapshots was proposed, paving the way to more algorithms and techniques. In this paper, we employ the randomized SVD (rSVD) algorithm [25, 38], a non-deterministic algorithm able to compute the near-optimal low-rank approximation of a given large dataset with good efficiency.

## 3 DMD on adapted meshes

Regarding numerical methods to approximate PDEs, the use of finer meshes in finite element discretizations usually leads to more accurate solutions. On the other hand, reducing the number of equations $n_{eq}$ in the nonlinear system is crucial for efficiency, especially considering that the optimal computational complexity of a single physics transient finite element simulation is $\mathcal{O}(n_{eq}^{\frac{4}{3}})$ [15]. The duality between the two statements describes a well-known trade-off between accuracy and efficiency in the finite element context. Despite a considerable research effort in the past decades, strategies to generate tailored meshes to maximize the accuracy while minimizing the computational effort are still an open research topic. Milestones addressing this subject are finite element a posteriori error estimators/indicators [2], techniques such as adaptation, interpolation [17, 47] and projection [18, 27, 50], as well as libraries and frameworks containing automated versions of AMR/C techniques [4, 42].

The general structure of the AMR/C scheme is given in Algorithm 1 and illustrated in Fig. 1. Three criteria are fundamental in an AMR/C algorithm: remeshing, flagging, and stopping. The remeshing criterion defines whether the computed solution at a given time step requires remeshing driven by global a posteriori error estimators (or indicators) and/or by calling the refinement/coarsening procedure at every $j$ time steps. Next, all mesh elements are visited and flagged for refinement or coarsening. The element flagging criterion is often represented by local a posteriori error estimators or indicators, i.e. flux-jumps of the solution gradient. In possession of the flagged elements, the remeshing algorithm is invoked. Finally, the previous mesh solution must be projected or interpolated into the target mesh. This is done by the projection/interpolation algorithm. The whole process is repeated until the stopping criterion is achieved. This criterion could be error equidistribution (until a certain threshold), a given element size, the maximum number of
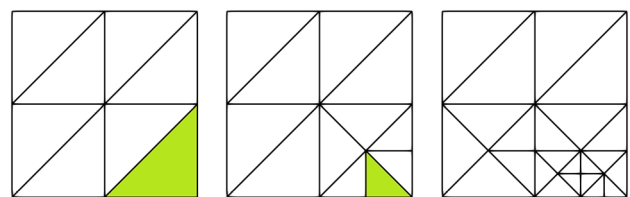


**Fig. 1** Illustration of a mesh refinement procedure. A local a posteriori error estimator or indicator flags an element for refinement (in green) using the solution computed in the mesh on the left. The mesh is refined (or coarsened) according to the flagged elements, and the process can be restarted until a given criterion is met (error level, element size, maximum number of elements, etc.). Note that the initial and the final mesh differ in the number of degrees of freedom and topology

elements in the mesh, or the number of refinement levels. Note that the meshes generated in Algorithm 1, guided by the error estimation procedure, have different dimensions (number of degrees of freedom) and different topologies, which poses difficulties for DMD (or, in fact, for any snapshot-based method).

The structure of the exact DMD algorithm relies on the fact that the measurements of the state vectors, i.e., $\mathbf{u}_0^h \dots \mathbf{u}_m^h$ have the same dimensionality. This structure could be exemplified in numerical experiments as fixed discretizations in space, i.e., fixed meshes or static sensors in experimental data. However, finite element simulations equipped with AMR/C strategies provide solutions in different function

---

**Algorithm 1** Adaptive Mesh Refinement/Coarsening Algorithm

INPUT: Finite element solution computed at a given time instant $t_i$.
OUTPUT: Adapted finite element solution.

---

- Definition of a remeshing criterion: global *a posteriori* error estimator/indicator, remeshing at every $j$ time steps, etc.
**for** each computed solution **do**
    **if** one or more remeshing criteria are met **then**
        **while** one or more stopping criteria are not met **do**
            - Compute local error estimators/indicators (or another strategy) to identify the elements requiring refinement or coarsening and flag them.
            - Call the refinement/coarsening procedure to generate a target mesh for the flagged elements. Different strategies for mesh refinement/coarsening lead to distinct meshes.
            - Project or interpolate the solution computed on instant $t_i$ onto the target mesh
        **end while**
    **end if**
**end for**

---

spaces, depending on the mesh used to compute the solution on a given time step. Spatial adaptivity on transient finite element simulations leads to meshes with a different number of nodes, numbering, and nodal coordinates. It can also lead to different mesh topologies and structures. For that reasons, snapshots obtained by AMR/C simulations cannot be stacked in columns to construct the snapshot matrix. Even if one considers an AMR/C strategy that restricts the adaptive meshes to preserve the dimensionality of the snapshots, the difference between the nodal coordinates of the various meshes will lead to misleading dynamics captured by DMD. In the present study, we circumvent this issue by considering the $\mathcal{L}^2$-projection [50] of the numerical simulation results for different meshes into a reference target mesh. Recent work on reducing these projection costs for related reduced-order modeling techniques (though not DMD) may be found in [26, 32, 37]. The projection or interpolation of numerical solutions between finite element meshes is a well-known computational mechanics subject. Many issues regarding boundary conditions, data visualization, or coupling arise from this kind of problem. The choice of a proper method to successfully project functions in different finite element spaces is not a trivial task since conservation may not be satisfied [18, 27, 50]. This issue is not addressed in the present study since the mesh projection occurs as a post-processing phase after computing the AMR/C solutions. Therefore, it

cannot lead to cumulative errors. Furthermore, we are also careful to choose reference target meshes with characteristic length equivalent with the existent in the AMR/C meshes to avoid any major accuracy losses. For the sake of generality, we consider the $\mathcal{L}^2$-projection as our method of choice, and it can be defined as follows [61]. Assuming that the solution on the donor mesh $\mathbf{u}^h(\mathbf{x}) = \sum_{j=1}^n \mathbf{u}_j \phi_j(\mathbf{x})$ to be projected onto the target mesh must satisfy the orthogonality condition,

$$(\mathbf{u}^h - \mathbf{u}^{\text{proj}}, \mathbf{v})_{\mathcal{L}^2} = 0 \quad \forall \mathbf{v} \in V^\psi \tag{18}$$

where $V^\psi$ is a finite-dimensional subspace of $\mathcal{L}^2(\Omega)$ defined by the target mesh and the interpolant $\mathbf{u}_{\text{proj}}$ is the optimal interpolant in the $\mathcal{L}^2$-norm for $V^\psi$. The orthogonal projection can be defined in terms of the following linear system

$$\mathbf{M}\mathbf{u}^{\text{proj}} = \mathbf{P}\mathbf{u}, \tag{19}$$

where $\mathbf{M}$ is the mass matrix and $\mathbf{P}$ is the projection matrix. The mass matrix is usual in finite element computations. The $\mathbf{P}$ matrix, however, can present technical difficulties (see "Appendix").

Therefore, our strategy consists of applying the $\mathcal{L}^2$-projection onto a tailored reference target mesh capable of representing the many scales in time and space of all the snapshots. This routine is inserted in the code and invoked after the adaptive procedure in Algorithm 1 at every output

time step $\Delta t_o$. This tailored reference mesh is described in this work as target reference mesh and should not be confused with the target mesh generated during the AMR/C procedure. The new solution (computed on the adaptive donor mesh) is projected onto the reference target mesh and exported as a simulation output. In this work, we export as output files for visualization purposes, although there is no restriction on stacking the snapshots on the snapshot matrix during the simulation runtime or dumping only $\mathbf{u}^{proj}$ on disk. We considered Gmsh [30], an open-source robust mesh generator, as our software of choice for defining and creating the target meshes for this study. The output for each time step is the snapshot with constant dimensionality $n$ such that all nodes in space are correctly mapped and capturing the dynamics existent in the system. The procedure is summarized in Algorithm 2. This strategy is relatively simple since the $\mathcal{L}^2$-projection consists of solving a linear system where the generated matrix is a mass matrix and requires no extra outputs for storing the projected solutions since the projection can be applied right after the AMR/C code. The mass matrix is generated in the finite element context by a self-adjoint operator, enabling more efficient solvers. In terms of versatility, the $\mathcal{L}^2$-projection method is flexible because a solution obtained for a given mesh can be naturally projected onto reference target meshes with different topologies and dimensionalities. Also, since the mesh projection is a vital part of AMR/C algorithms, finite element algorithms frequently present efficient implementations of interpolation or projection techniques. Figure 2 shows an example where

a solution obtained by an adaptive mesh simulation is projected onto two meshes with different topologies.

For this example, we consider the square domain $\Omega = [-1, 1] \times [-1, 1]$ and the function $u$ defined in $\Omega$ such that

$$u = \begin{cases} 1, \text{ if } (x, y) \in [-0.3, 0.3] \times [-0.3, 0.3], \\ 0, \text{ otherwise.} \end{cases} \tag{20}$$

This function is approximated on a structured finite element mesh discretized into $10 \times 10$ cells where each cell is divided into two triangular elements. An AMR/C procedure is invoked to refine three times the transition between $\mathbf{u} = 1$ and $\mathbf{u} = 0$, creating a new mesh containing 1672 elements and 857 nodes. Figure 2a shows the new mesh generated after the AMR/C procedure and the function approximated by the resulting function space. Two meshes—one structured and another unstructured—are considered for projection. The structured mesh contains $80 \times 80$ cells, resulting in 12,800 triangular elements and 6561 nodes. The element sizes of the structured mesh are similar to the smallest elements in the adaptive mesh. The unstructured mesh presents a smaller characteristic length than the structured mesh and contains 17,088 elements and 8705 nodes. Figure 2b, c shows the proposed meshes and the projections of the solution onto the new finite element spaces. Note that the projected solutions are fairly accurate since their infinity norm is in good agreement with the adaptive mesh solution's infinity norm.

---

**Algorithm 2** Snapshot projection

INPUT: Finite element solutions considering AMR/C
OUTPUT: Projected solutions onto a prescribed reference target mesh.

---

**for** each observation time step $\Delta t_o$ **do**
    1: Apply the $\mathcal{L}^2-$projection of the adaptive solution onto the function space of the reference target mesh.
    2: Stack the resulting snapshot into the snapshot matrix $\mathbf{Y}^h$ or output the projected solutions to disk.
**end for**

---

*Remark* In this study, the projection is carried out inside the finite element simulation since the projection computational cost is practically negligible in comparison with the overall time required for solving nonlinear systems of a complex numerical simulation. However, if one does not have access to the finite element simulation codes used, the reconstruction of the solution of each time step can still be done offline. Output files of various formats contain information regarding the mesh used (such as nodal coordinates and connectivities) for visualization purposes. By properly reading these files, the solutions can be reconstructed under a finite

element framework (such as FEniCS [4] or libMesh [42]) or on a code developed by the user. However, this totally non-intrusive approach can significantly increase the computational cost due to several I/O operations that are often extremely low compared to computational intensive operations. Since the mesh constantly varies in an AMR/C finite element simulation, each solution obtained in the simulation has to be imported, reconstructed under its original mesh, and projected. After the projection, the user can choose to dump the solution in the disk or stack the snapshots on the snapshot matrix. For the first case, another I/O operation
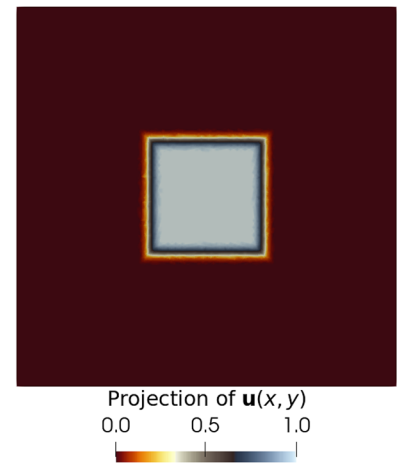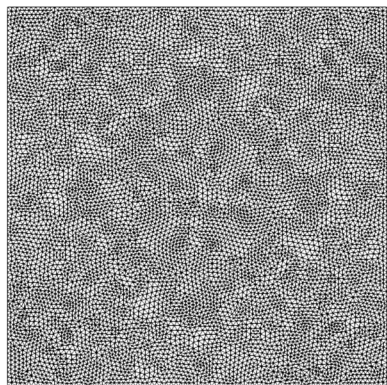
**Fig. 2** Comparison of different $\mathcal{L}^2$-projection examples on structured and unstructured meshes
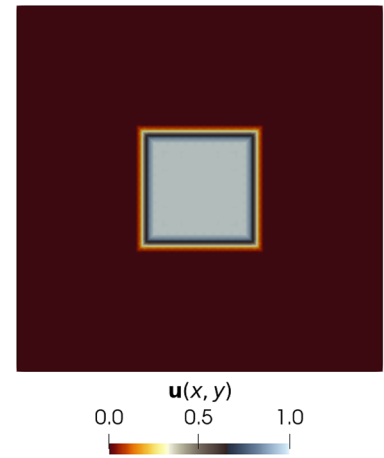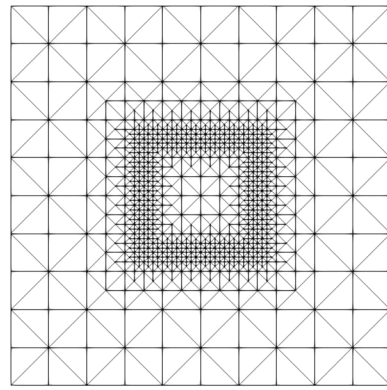


(a) Adaptive mesh (left) and solution (right), $||\mathbf{u}^h||_\infty = 1.000$.
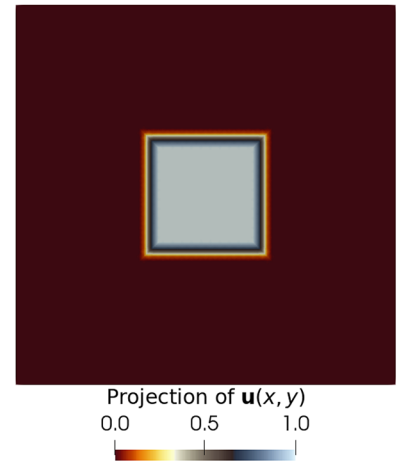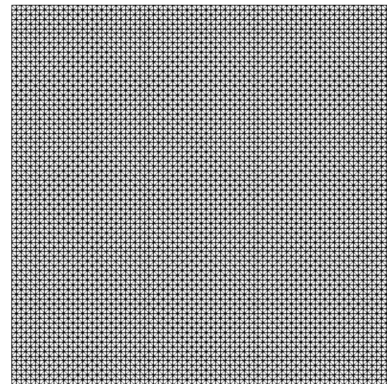


(b) Structured mesh (left) and projected solution (right), $||\mathbf{u}^{proj}||_\infty = 1.000$.



(c) Unstructured mesh (left) and projected solution (right), $||\mathbf{u}^{proj}||_\infty = 0.999$.

would be invoked. This non-intrusive strategy is especially suitable (and restricted) to be used with DMD, which is also a non-intrusive algorithm. For intrusive snapshot-based methods such as Proper Orthogonal Decomposition (POD), one needs to project finite element matrices onto the computed basis and, therefore, requires access to the code [62].

# 4 Numerical experiments

In this section, for the sake of generality, we apply our method in several applications, with different systems of equations, numerical formulations, mesh topologies, spatial dimensions, refinement criteria, and finite element libraries. We compare the results between the AMR/C solution, the fixed mesh solution, and the DMD results for all cases. First, we test the DMD short-time future prediction capabilities on a continuous SEIRD model for COVID-19, a nonlinear system of diffusion–reaction equations. The equations are considered using a Galerkin finite element discretization and are solved using `libMesh` [42], a high-performance C++ finite element library. The error estimators, refinement/ coarsening strategies built-in on `libMesh` can be seen on [34, 63]. Also, the $\mathcal{L}^2$-projection algorithm is embedded in `libMesh`. We explore the results in one and two spatial dimensions, where the 1D case is a hypothetical example, and the 2D case describes the COVID-19 evolution in the Lombardy region in Italy [34, 63, 64]. The simulation obtains the results for 44 and 60 days for the 1D and 2D cases, respectively. Since we want to predict 14 days in the future, we only feed DMD with snapshot matrices containing the first 30 and 46 days in the 1D and 2D cases, respectively, such that the predicted results can be compared with the results obtained in the simulations. To apply DMD to the SEIRD data obtained on an AMR/C simulation, we project the adaptive solution onto a mesh with characteristic length compatible with the obtained simulation results. The discretization of the reference mesh used for the projection is as fine as the final refinement level on the adaptive meshes. Next, we consider two fluid dynamics applications where the governing equations for both cases are advection-dominated. To circumvent the LBB condition and spurious oscillations regarding dominant advection, these equations consider the residual-based variational multiscale (RBVMS) formulation [1, 8, 19, 35, 39, 53] on a finite element discretization. We consider the use of DMD on a 2D density-driven gravity flow and a 3D bubble rising simulation. The density-driven gravity current is modeled by the coupling of the incompressible Navier–Stokes equation and the advection–diffusion equation. We consider a lock-exchange problem, where a tank is filled with two fluids of different densities, separated by a lock. The simulation starts when the lock is removed, and the difference in the densities of the fluids

generates the driving forces responsible for the motion of the fluids. Unlike the other examples in this work, the implementation of this numerical test is made on the `FEniCS` v.2019.1 framework [4], a high-performance Python/C++ finite element library. The refinement/coarsening algorithm and $\mathcal{L}^2$-projection algorithm for this example are part of the framework. We consider an interface-tracking error indicator for the AMR/C simulations, and the mesh is refined following the bisection method [54]. DMD is considered to reconstruct the solution, and the results are compared to the fixed mesh results and the results obtained by AMR/C simulations. Finally, we extend our analysis to a 3D bubble rising case [33], a two-phase incompressible flow problem where the interface is captured by the convected level-set method. This model is implemented on `libMesh`, taking advantage of the same refinement/coarsening strategies as well as the projection algorithm used in the SEIRD numerical tests. In this example, we test the projection of the adaptive solutions onto three different meshes containing the different scales existent on the AMR/C simulation and evaluate the results. We consider DMD to predict the bubble geometry and dynamics for a short time in the future.

For all the numerical tests proposed, we evaluate the results in terms of efficiency and accuracy. For efficiency purposes, we compute the ratio between the computational time required to run the finite element simulations and the time required to run DMD separately. We refer to this quotient as speedup. The finite element code is responsible for computing the snapshots and projecting the results onto the reference target meshes proposed, while the DMD code imports the output files, extract the snapshots, computes the approximation, and outputs the results. Also, we provide a table describing how the projection routine affects the overall computational time required for the simulations for all examples. In terms of accuracy, we evaluate the results in terms of overall relative error $\eta_F = \dfrac{||\mathbf{Y}^h - \mathbf{Y}^h_{\text{DMD}}||_F}{||\mathbf{Y}^h||_F}$ where $\mathbf{Y}^h$ is the snapshots matrix, $\mathbf{Y}^h_{\text{DMD}}$ is a matrix comprising the approximations obtained by DMD and $|| \cdot ||_F$ denotes the Frobenius norm. A more detailed analysis is also done in terms of relative error in time $\eta$. For that, we plot the curves of the relative errors (in terms of $\mathcal{L}^2$-norm) of each snapshot. We compute $\eta = \dfrac{||\mathbf{u}^h_k - \mathbf{u}^h_{k\text{DMD}}||_2}{||\mathbf{u}^h_k||_2}$ for $k = 0, 1, \ldots, m$ snapshots, where $|| \cdot ||_2$ denotes the $\mathcal{L}^2$-norm. Also, to avoid unphysical results, some quantities of interest are evaluated and compared using both simulation and approximation results. For the SEIRD model, we plot the results regarding the total population. This quantity should be constant in time according to the hypothesis of the model. For the lock-exchange simulation, we compute the mass during the simulation and the front position. Since the simulation is

considered on a closed tank, the mass must be kept constant during the simulation. For the 3D bubble rising problem, we plot the quantities of interest related to the geometry (volume and sphericity) and dynamics (center of mass and rise velocity) of the bubble. In this case, we test meshes with different minimum characteristic lengths. The results are shown and discussed below.

## 4.1 Continuous SEIRD model for COVID-19

The outbreak of COVID-19 in 2020 has led to a surge in interest in the mathematical modeling of infectious diseases. This new virus is responsible for infecting millions of people worldwide and impacting the economy in an unprecedented way. Therefore, numerical simulation of the virus' dynamics may help provide short-term prediction models for forecasting the number of future cases. In this perspective, it is possible to develop strategic planning in the public health system to avoid deaths and manage patients.

Disease transmission may be modeled as *compartmental models*, in which the population under study is divided into compartments and has assumptions about the nature and time rate of transfer from one compartment to another [11]. Here, we work with a spatio-temporal SEIRD model, presented in [34, 63, 64], and given by,

$$\frac{\partial s}{\partial t} + \beta_i \left(1 - \frac{A_e}{n_{\text{pop}}}\right) si + \beta_e \left(1 - \frac{A_e}{n_{\text{pop}}}\right) se$$
$$- \nabla \cdot (n_{\text{pop}} \nu_s \nabla s) = 0 \tag{21}$$

$$\frac{\partial e}{\partial t} - \beta_i \left(1 - \frac{A_e}{n_{\text{pop}}}\right) si - \beta_e \left(1 - \frac{A_e}{n_{\text{pop}}}\right) se$$
$$+ (\alpha + \gamma_e)e - \nabla \cdot (n_{\text{pop}} \nu_e \nabla e) = 0 \tag{22}$$

$$\frac{\partial i}{\partial t} - \alpha e + (\gamma_i + \delta)i - \nabla \cdot (n_{\text{pop}} \nu_i \nabla i) = 0 \tag{23}$$

$$\frac{\partial r}{\partial t} - \gamma_e e - \gamma_i i - \nabla \cdot (n_{\text{pop}} \nu_r \nabla r) = 0 \tag{24}$$

$$\frac{\partial d}{\partial t} - \delta i = 0, \tag{25}$$

where $s(\mathbf{x}, t)$, $e(\mathbf{x}, t)$, $i(\mathbf{x}, t)$, $r(\mathbf{x}, t)$, and $d(\mathbf{x}, t)$ denote the densities of the *susceptible*, *exposed*, *infected*, *recovered*, and *deceased* populations, respectively. The sum of all the compartments with the exception of $d(\mathbf{x}, t)$ is represented by $n_{\text{pop}}$ which is the total living population. $A_e$ characterizes the Allee effect (persons), that takes into account the tendency of outbreaks to cluster around large populations, $\beta_i$ and $\beta_e$ denote the transmission rates between symptomatic and susceptible individuals and asymptomatic and susceptible
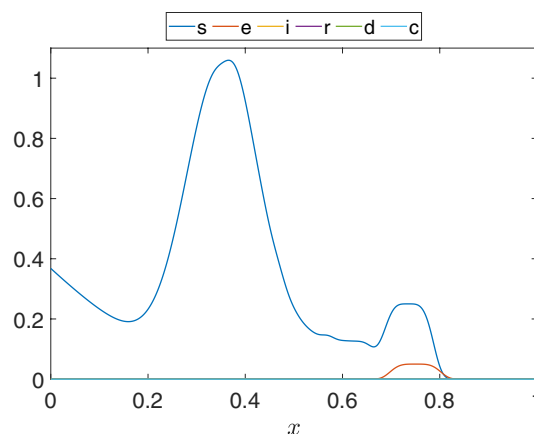


**Fig. 3** Initial conditions for the 1D model

individuals, respectively (units days$^{-1}$), $\alpha$ denotes the incubation period (units days$^{-1}$), $\gamma_e$ corresponds to the asymptomatic recovery rate (units days$^{-1}$), $\gamma_i$ the symptomatic recovery rate (units days$^{-1}$), $\delta$ represents the mortality rate (units days$^{-1}$), and $\nu_s$, $\nu_e$, $\nu_i$, $\nu_r$ are the diffusion parameters of the different population groups as denoted by the sub-scripted letters (units km$^2$ persons$^{-1}$ days$^{-1}$). Note that all these parameters can be considered time and space-dependent. We also compute the compartment $c$, the cumulative field of the $i$ compartment.

For the numerical solution of (21)–(25), we discretize in space using a Galerkin finite element variational formulation. The resulting systems of equations are stiff, leading us to employ implicit methods for time integration. We apply the Backward Differentiation Formula (BDF2), which offers second-order accuracy while remaining unconditionally stable. We implement the whole model in `libMesh` [42]. We additionally make use of AMR/C, allowing us to resolve multiple scales. One may find more details about the methods in [34, 63].

### 4.1.1 Reproducing a 1D model

First, we use a simple 1D continuous SEIRD model for COVID-19 with adaptive mesh refinement to validate the $\mathcal{L}^2$-projection and DMD. This example was introduced in [63] and reproduced in [34]. Basically, we consider a 1D region $\Omega = [0, 1]$ with initial conditions that represents a large population centered around $x = 0.35$ with no exposed persons and a small population centered around $x = 0.75$ with some exposed individuals, as shown in Fig. 3. Thus, we set $s = s_0$ and $e = e_0$ as follows:

**Fig. 4** Solution at $t = 30$ days for the fixed mesh solution, AMR solution and the respective projection onto a reference mesh for the 1D SEIRD example. The reference mesh was built with characteristic length similar to the smaller elements in the adaptive mesh
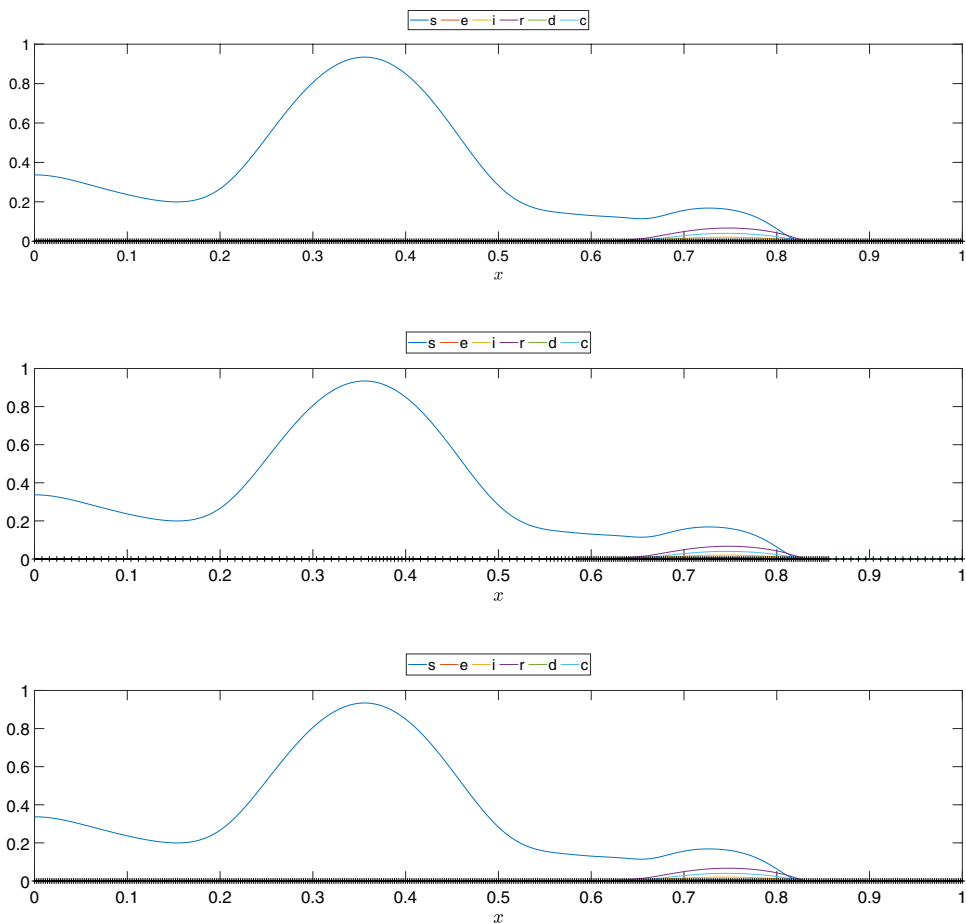


**Table 1** Absolute and relative time required for the projection routine in comparison with the adaptive finite element simulation code (AMR/C FEM) for the SEIRD model in the 1D case

| Code part | Absolute time (s) | Relative time (%) |
|---|---|---|
| AMR/C FEM | 1262.89 | 98.53 |
| Mesh projection | 18.88 | 1.47 |

$$s_0 = e^{-(x+1)^4} + e^{-\frac{(x-0.35)^2}{10^{-2}}}$$
$$+ \frac{1}{8}\left(e^{-\frac{(x-0.62)^4}{10^{-5}}} + e^{-\frac{(x-0.52)^4}{10^{-5}}} + e^{-\frac{(x-0.42)^4}{10^{-5}}}\right) \qquad (26)$$
$$+ \frac{1}{4}e^{-\frac{(x-0.735)^4}{10^{-5}}}$$

$$e_0 = \frac{1}{20}e^{-\frac{(x-0.75)^4}{10^{-5}}}. \qquad (27)$$

We further set $i_0 = 0$, $r_0 = 0$, and $d_0 = 0$. We also enforce homogeneous Neumann boundary conditions at $x = 0$ and a zero-population Dirichlet boundary condition at $x = 1$ for all model compartments. The latter represents a non-populated area at $x = 1$.

Following [34, 63] we set $\alpha = 0.09375$ days$^{-1}$, $\beta_i = \beta_e = 0.375$ days$^{-1}$persons$^{-1}$, $\delta = 0.0046875$ days$^{-1}$, $\gamma_i = 0.03125$ days$^{-1}$ and $\gamma_e = 0.125$ days$^{-1}$, $A = 0$, $\nu_s = 3.75 \times 10^{-5}$, $\nu_e = 0.75 \times 10^{-3}$, $\nu_i = 0.75 \times 10^{-10}$ and $\nu_r = 3.75 \times 10^{-5}$ km$^2$persons$^{-1}$days$^{-1}$. The time step size is defined as $\Delta t = 0.25$ days and we consider the mesh projection and outputs at every time step, that is, $\Delta t_o = \Delta t = 0.25$.

We use an adapted mesh with initially 125 elements, and after the refinement, the smallest element has a size 0.002. At the beginning of the simulation, we refine uniformly the whole domain into two levels and, after that, we apply the adaptive mesh refinement every 4 time steps. The idea is that the AMR/C strategy will keep this spatial resolution on more dynamically relevant regions while coarsening other regions in the domain. As a target reference mesh, we consider the uniformly refined mesh, such that all the domain contains the minimal spatial resolution obtained by the AMR/C simulation.

The results of this simulation are validated against the results from [34, 63]. Figure 4 show the solution of the 1D SEIRD example at $t = 30$ days for the fixed mesh simulation, the adaptive mesh simulation, and the projection of the adaptive solution onto the target reference mesh used in the fixed mesh simulation. We observe a good agreement between

**Table 2** Relative error between reconstructed (and predicted) data and the projected snapshots

| Compartments | Relative error | Speedup |
|---|---|---|
| $s$ | $1.590 \times 10^{-3}$ | $1.079 \times 10^{3}$ |
| $e$ | $2.574 \times 10^{-2}$ | $1.316 \times 10^{3}$ |
| $i$ | $1.162 \times 10^{-2}$ | $1.225 \times 10^{3}$ |
| $r$ | $1.439 \times 10^{-2}$ | $1.264 \times 10^{3}$ |
| $d$ | $2.001 \times 10^{-2}$ | $1.331 \times 10^{3}$ |
| $c$ | $1.286 \times 10^{-2}$ | $1.341 \times 10^{3}$ |

the solutions. This agreement is a positive indicator that the DMD can be used on the projected meshes with little to no error compared to the DMD on a fixed mesh simulation. In terms of efficiency, Table 1 shows the computational effort required for the reference mesh projection embedded on the adaptive finite element code in comparison with the time used for the simulation code itself (AMR/C FEM).

After some initial experiments, we set $r = 15$ for all compartments. In this example, the finite element simulation computes the results for 44 days, that is, 176 time steps. However, we only consider the first 30 days in the snapshot matrix for reconstruction. DMD is set to approximate the results for a further 14 days so the results can be compared with the 44 days results obtained in the finite element simulation. In other words, we want to evaluate the DMD ability to predict the COVID-19 scenario two weeks in the future given the data of the last 30 days. For numerical reasons, we considered an initial 3 days shift in the snapshots since some compartments are initialized with zeros, affecting how DMD captures the dynamics. The 44th day solution for the adaptive simulation and the 44th day prediction considering DMD are seen in Fig. 5. We observe good agreement between predictions and the numerical solutions of the simulations for most compartments. The speedup and overall relative error between the DMD approximation and the snapshots are seen in Table 2.

### 4.1.2 The Lombardy region

We extend our analysis by solving the continuous SEIRD model and applying DMD to a 2D real world domain that is the Lombardy region in Italy. The spread of the COVID-19 has been studied in this region using the continuous SEIRD model with accurate results [63, 64]. Here, we reproduce this simulation with the solver developed in [34] which invokes adaptive mesh refinement every 4 time steps. We use the same parameters as shown in [63, 64]. It is important to point that, in this simulation, the transmission rates and diffusion parameters vary with time in order to reproduce the effects of restrictions during the simulated period.

For this simulation, an unstructured mesh is considered due to the complex geometry imposed by the domain. The mesh is generated using Gmsh and is uniformly refined as the simulation starts. After refining the whole mesh in one level, the mesh presents a minimum spatial resolution of approximately 1 km. This procedure allows the solver to coarsen the regions where no significant dynamics are observed while preserving the scales of the regions of interest. Figure 6 shows the variation of the number of nodes in time for the AMR/C strategy. The coarsening approach improves the simulation performance significantly since the average number of nodes (and respectively, the number of equations $n_{eq}$) used in the adaptive simulation is approximately half with respect to the case of a fixed mesh with the same spatial resolution considered for the entire domain. In this example, the reference target mesh is the uniformly refined unstructured mesh considered in the early stages of the simulation, presenting 13,158 nodes and 25,340 elements. The simulation considers a time step size of $\Delta t = 0.25$ days for the numerical integration and $\Delta t_o = \Delta t = 0.25$ days for the observations. Initial conditions for the Lombardy domain are the same presented on [63, 64] and are seen in Fig. 7, while compartments $r$ and $d$ are initialized to zero.

We then proceed to run both adaptive and fixed mesh simulations and, to apply DMD in the adaptive mesh results, we consider the proposed projection scheme. Figure 8 shows the $s$ compartment solution at $t = 46$ days for both simulations and the projected adaptive solution onto the reference mesh, revealing that the results are in good agreement. In terms of efficiency, Table 3 shows results for the computational time required for the projection compared to the finite element code. That said, the adaptive snapshots can now be assembled into a snapshot matrix for the DMD reconstruction and prediction.

The DMD analysis is made in the same way as presented in the 1D case: the simulation outputs the projected snapshots for the first 60 days (240 snapshots). The snapshot matrix assembles the information regarding 46 days of simulations, while DMD approximates the results for 60 days. The idea is to predict two weeks in the future, given the data observed in the past 46 days. For this example, we consider the SVD truncation at $r = 20$ for all compartments. Again, an initial 3 days shift in the data is considered to avoid issues with the compartments initialized to zero. Results for the 60th day comparing the computed numerical solutions and the DMD predictions are seen in Figs. 9 and 10. We present the projected solutions, the DMD prediction, and the relative error in space between the two results from left to right. We can note that most compartments show results in agreement with the simulations, while the exposed compartment reveals more pronounced differences than the other compartments.

Figure 11 shows the relative error in time between the DMD results and the projected snapshots. The first thing we
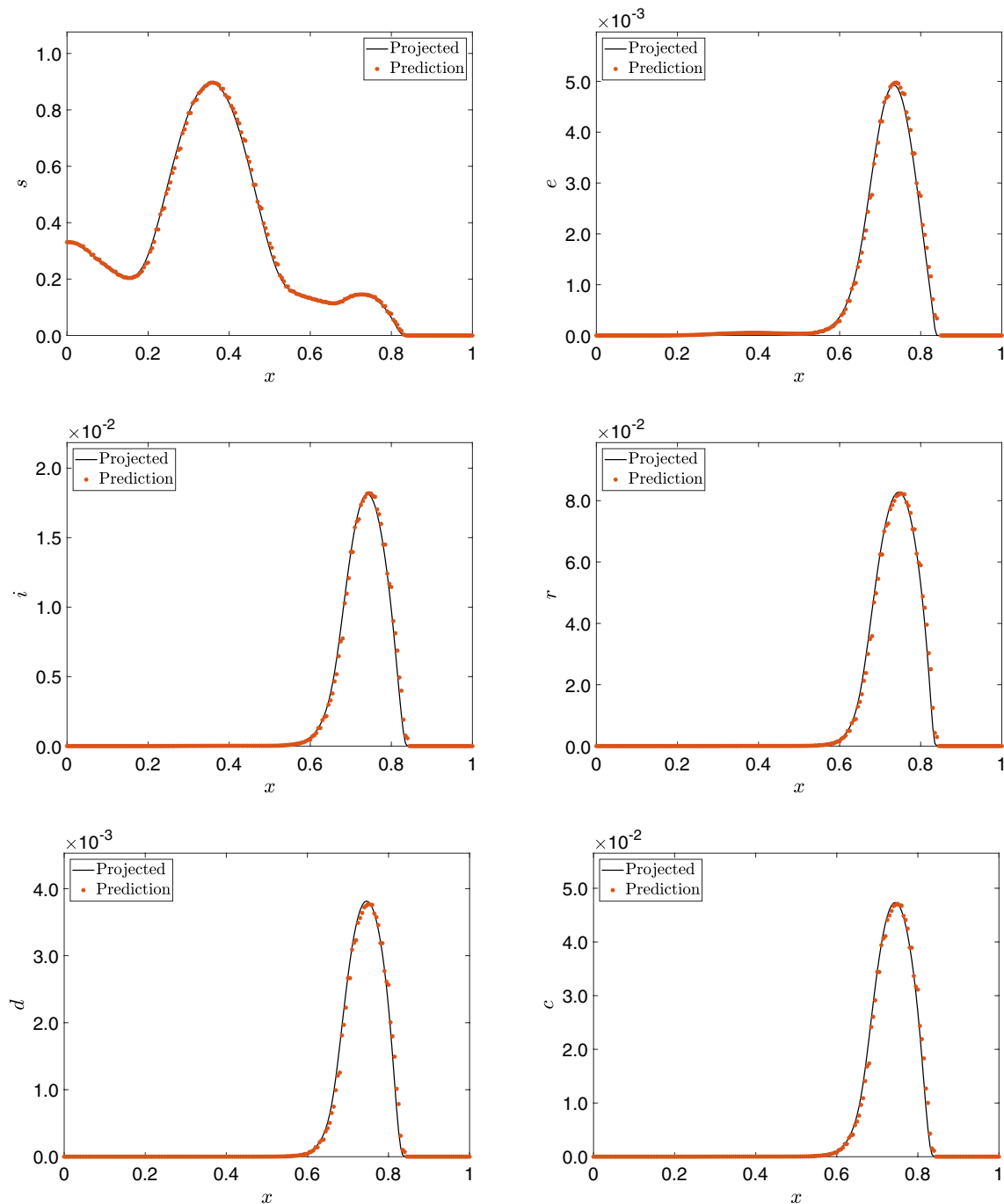
**Fig. 5** Solution at $t = 44$ days for the AMR simulation solution and the 14 days projection using DMD for the 1D SEIRD example

notice is that the curves are different for each compartment. This discrepancy occurs due to the different parameters for each equation in the SEIRD model, which largely affects the dynamics of the system. The dynamics for each compartment are different since each compartment presents different coupling, diffusion, and reaction parameters. Also, regarding this issue, since the parameters are time and space-dependent, sudden changes in their values can affect the dynamics

of the system as well as DMD's dynamics mapping ability. Some sudden changes in the $s$ and $e$ compartments related to stricter public policies considered to reduce the transmission rates (parameters $\beta_i$ and $\beta_e$) are incorporated into the model. Since the variation in the parameters is not introduced smoothly, DMD's ability to map sudden changes in the dynamics of the system is reflected by the existence of some spikes on the curves of the relative errors in time.
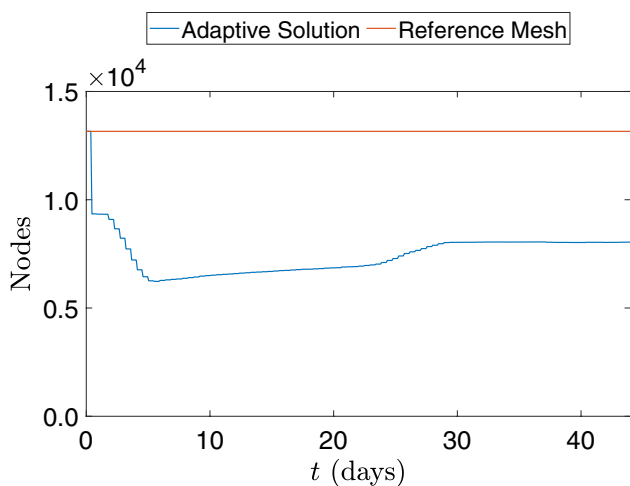
**Fig. 6** Number of mesh nodes in time for the adaptive solution and the proposed reference mesh for the 2D SEIRD example

Comparing the reconstruction and prediction stages, we observe that the errors tend to grow as soon as the prediction stage starts (dashed line). The exposed compartment, which yielded most of the oscillations due to parameter changing on the reconstruction stage, presented the same behavior on the prediction phase around day 49. We also note that the exposed compartment yields a larger relative error for the 60th day in comparison with the other compartments. Table 4 shows the overall relative error and the speedups for the six compartments approximations. Comparing these results with the results presented in Figs. 9, 10 and 11, we can conclude that the predictions are reasonably accurate in comparison with the numerical solutions, specially when considering the time required for calculation.

Another important analysis to be done is the conservation property of the continuous SEIRD model. As mentioned before, the standard $\mathcal{L}^2$-projection does not guarantee

conservation among the projections. Figure 12 shows the total population during the simulation, normalized by the total population modeled in the initial conditions. The total population is computed as the sum of the integral of the compartments (excluding $c$) divided by the sum of the integral of the elements of the mesh. Since the SEIRD model does not consider any population growth, the value must be theoretically constant for all the simulations. From the figure, we observe that the population is kept constant during all the adaptive simulation, and this was preserved by the projected solutions and the DMD reconstruction stage. That is, we can note that the $\mathcal{L}^2$-projection does not yield conservation issues in this example. For the prediction phase, DMD preserves the total population for several days in the future. However, it presents a slight increase (around 0.1%) for predictions over 10 days, which does not affect the results significantly. This increase can be explained by the relative errors behavior, observed in Fig. 11, as DMD computes future estimates.

## 4.2 Fluid dynamics

This section evaluates the DMD use on two cases involving AMR/C in computational fluid dynamics: the reconstruction of a 2D density-driven gravity current simulation and the temporal prediction on a 3D rising bubble. Different from the previous cases, the test cases presented in this section are advection-dominant. To approximate the governing equations, we use a finite element RBVMS formulation [1, 19, 35, 39, 53]. In the first case, we reconstruct the solution and evaluate important quantities of interest regarding density-driven gravity flows. On the bubble rising simulation, we show how our strategy works on a 3D mesh, and we evaluate the DMD ability to predict the quantities of interest of the rising bubble in time.
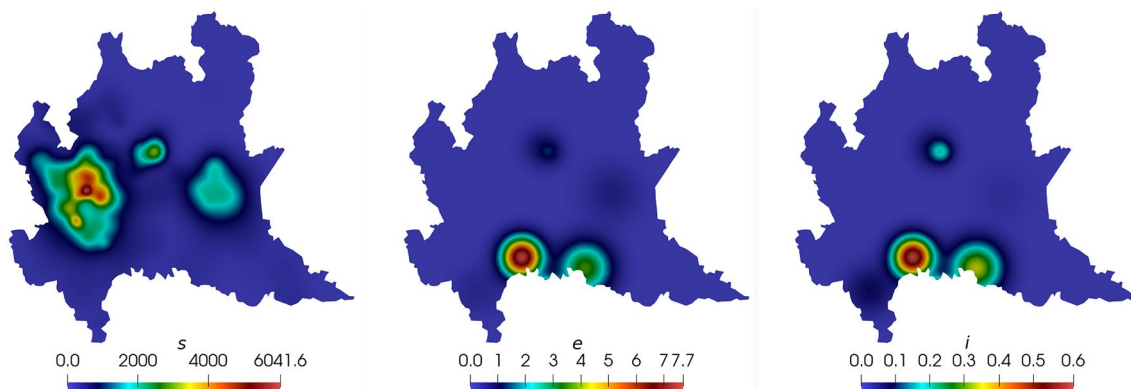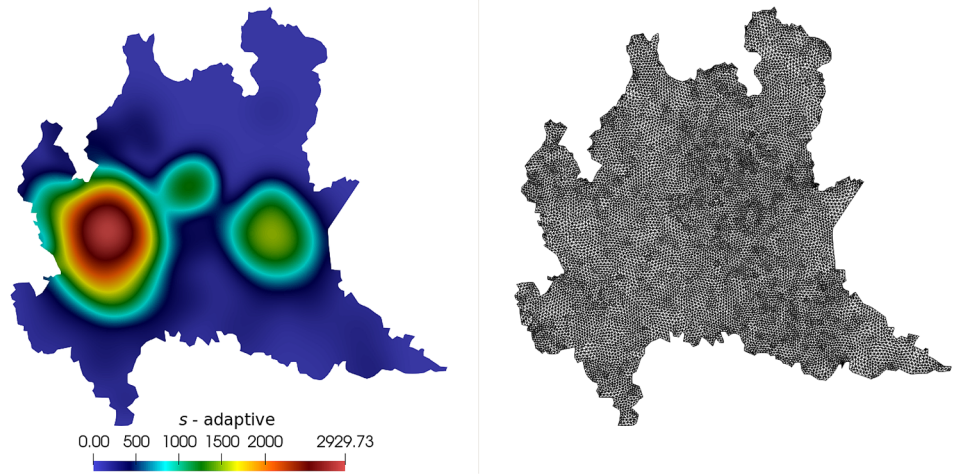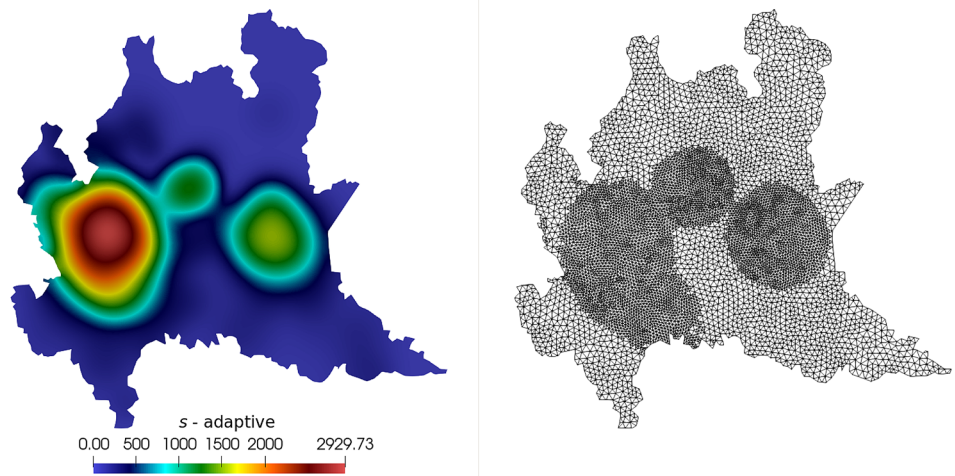


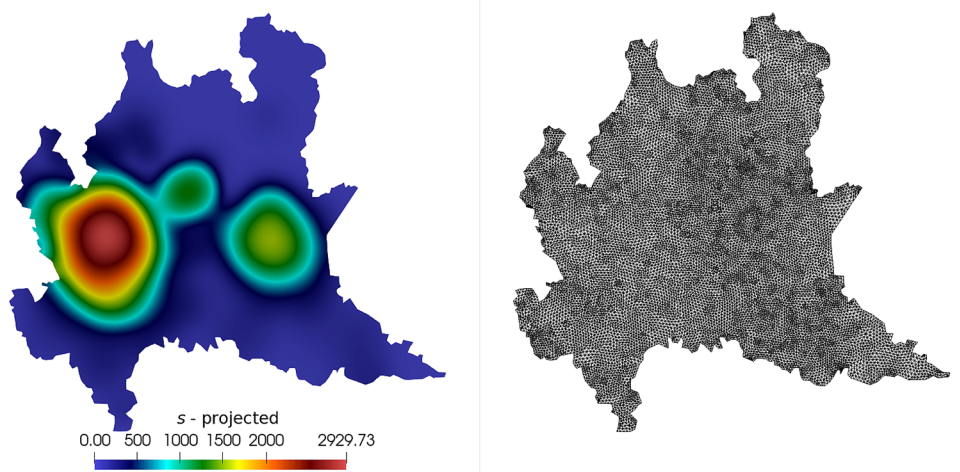**Fig. 7** Initial conditions for the SEIRD model in the Lombardy case

(a) Fixed mesh solution

(b) Adaptive solution

(c) Projection onto reference mesh

**Table 3** Absolute and relative time required for the projection routine in comparison with the adaptive finite element simulation code (AMR/C FEM) for the SEIRD model in the Lombardy case

| Code part | Absolute time (s) | Relative time (%) |
| --- | --- | --- |
| AMR/C FEM | 2107.24 | 98.52 |
| Mesh projection | 31.60 | 1.48 |

### 4.2.1 Density-driven gravity flow

In this section, we consider a long numerical simulation that consists of a lock-exchange between two fluids, the heavy fluid, A, and the lighter fluid, B, based on the numerical example in [49]. The difference between their densities is such that the Boussinesq hypothesis is considered valid. Moreover, particles in the heavy fluid have negligible inertia and are much smaller than the smallest length scales of
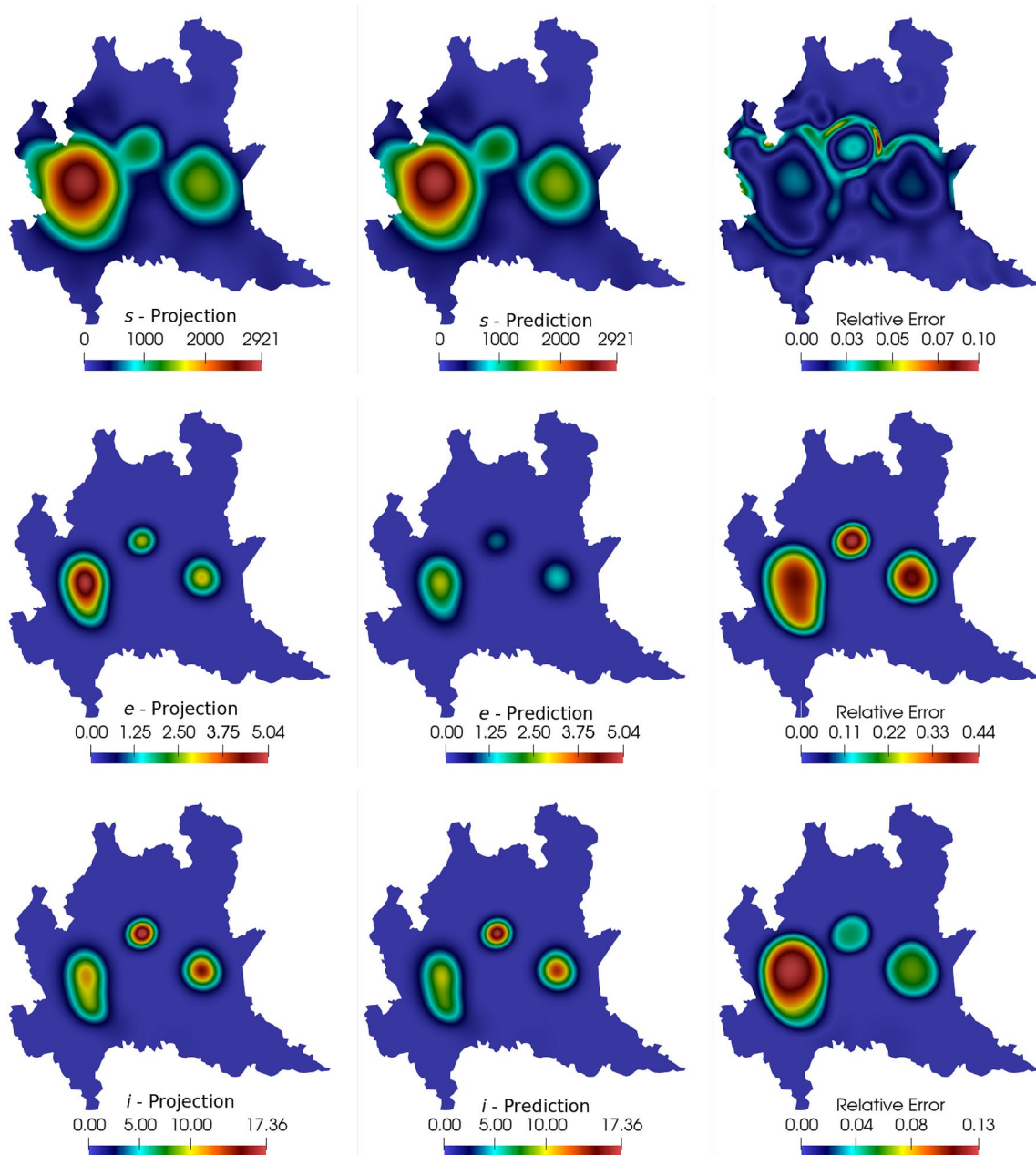


**Fig. 9** Comparison between computed and predicted solutions at $t = 60$ days for the susceptible, exposed, and infected compartments
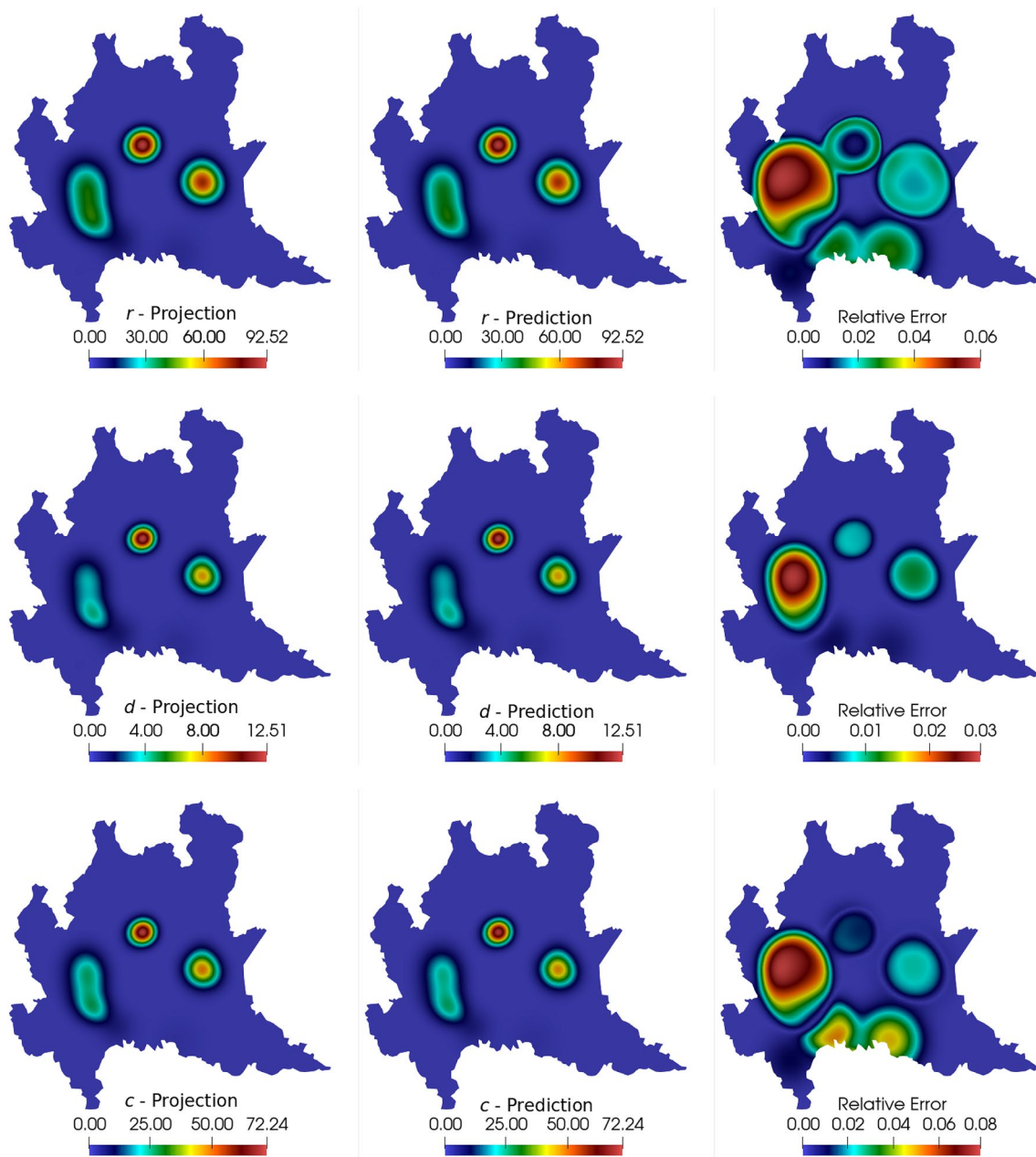
**Fig. 10** Comparison between computed and predicted solutions at $t = 60$ days for the recovered, deceased, and cumulative infected compartments

the buoyancy-induced fluid motion. Thus, the dimensionless governing equations are

$$\nabla \cdot \mathbf{u} = 0,$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \frac{1}{\sqrt{Gr}} \Delta \mathbf{u} - \phi \mathbf{e}^g = 0,$$

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi - \frac{1}{Sc\sqrt{Gr}} \Delta \phi = 0,$$

(28)

where $\mathbf{u}$ is the fluid velocity, $\phi$ is the concentration field, $p$ is the pressure, $\mathbf{e}^g = (0, -1)$ is the vector pointing in the direction of gravity, $Sc = 1.0$ is the Schmidt number and $Gr = 5 \times 10^6$ is the Grashof number, two dimensionless numbers that relate viscous effects with diffusion and buoyancy effects, respectively. A Grashof number of this magnitude indicates a turbulent flow. The field $\phi = \phi_A / \phi_B$ is the concentration and is responsible for mapping the evolution of fluid interactions. The time step size considered for this simulation consists on $\Delta t = 0.01$s for a total simulation time
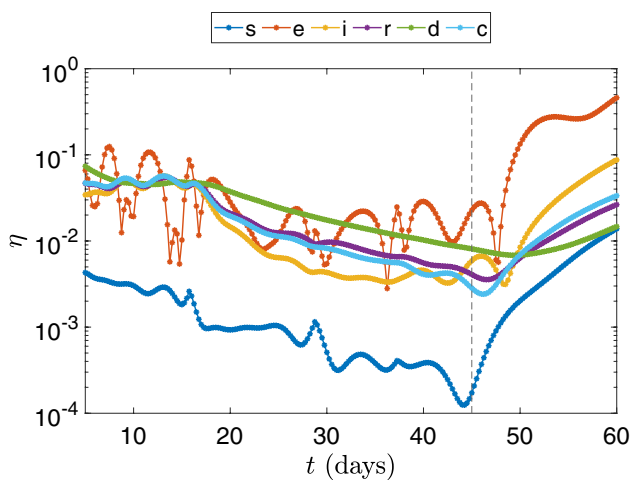
**Fig. 11** Relative error for all compartments between numerical simulation snapshots and DMD reconstruction and prediction. The dashed line represents the beginning of the DMD prediction stage

**Table 4** Relative error between reconstructed (and predicted) data and the computed snapshots and speedup between DMD and the numerical simulation

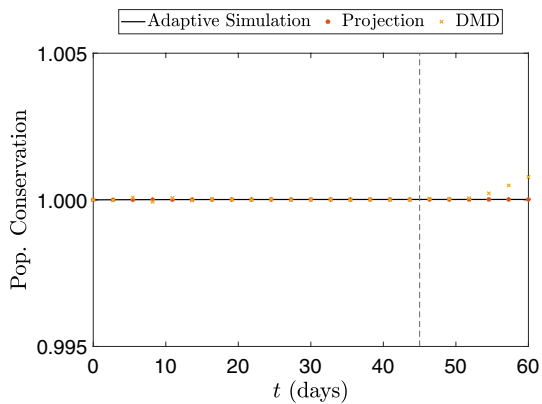| Compartments | Relative error ($\eta_F$) | Speedup |
| --- | --- | --- |
| $s$ | $1.345 \times 10^{-3}$ | 822.61 |
| $e$ | $5.187 \times 10^{-2}$ | 938.09 |
| $i$ | $1.304 \times 10^{-2}$ | 755.97 |
| $r$ | $7.316 \times 10^{-3}$ | 1036.93 |
| $d$ | $1.097 \times 10^{-2}$ | 977.91 |
| $c$ | $1.251 \times 10^{-2}$ | 977.03 |



**Fig. 12** Population conservation for both adaptive and projected results

of $T = 30$s with an output frequency of $\Delta t_o = \Delta t = 0.01$s. We consider a tank, that is, a rectangular domain with length $L_1 = 18$m, height $L_2 = 2$m. The boundary conditions for this
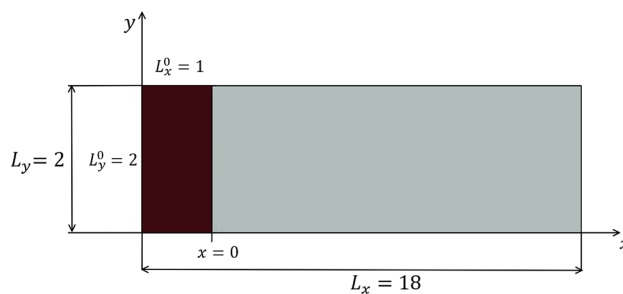


**Fig. 13** Scheme illustrating the initial conditions for the density-driven gravity flow example

case are no-slip for the velocity and no-flux for the transport equation, and the initial conditions are such that the heavy fluid is represented as a column with dimensions $L_x^0 \times L_y^0 = 1$m $\times$ 2m located at the left border of the tank and the light fluid fills the rest of the domain. Figure 13 illustrates the domain and the initial conditions.

To solve the governing equations, we implement the RBVMS formulation [35] for Eq. (28) using the FEniCS 2019.1 [4] framework to generate the snapshots for this example. The adaptive mesh refinement procedure returns the solution for $\mathbf{u}, p,$ and $\phi$. For this example, we only consider the snapshots of $\phi$ for our calculations, that is, an overall data reduction of 75%. Details of the formulation of the problem can be found in [7, 35]. We consider a fixed mesh simulation with $701 \times 101$ nodes and $700 \times 100$ cells, where each cell is divided into two linear triangles. We consider an interface-tracking adaptive mesh error indicator that flags and refines the mesh where the two fluids interact for the adaptive mesh simulation. The error indicator for the mesh refinement is $|\nabla \phi|$ being larger than a given tolerance. For this purpose, a mesh containing $175 \times 25$ cells is considered, the interfaces are refined considering two levels of refinement, and the mesh refinement is invoked at every time step. Figure 14 shows the results at $t = 10$s for both fixed and adaptive mesh simulations and the projection of the adaptive solution onto the same mesh used in the fixed mesh simulation.

AMR/C in this problem is advisable since the dynamics are predominant on the interface between the fluids. Most of the domain is not affected in the early stages of the simulation, and the use of fine meshes outside these regions may represent unnecessary computational effort. Figure 15 shows the number of nodes in the mesh during the simulation time for the adaptive mesh compared with the fixed $701 \times 101$ nodes mesh.

We observe that the number of nodes in the adaptive simulation is smaller than the fixed mesh for all the simulation time. Table 5 shows the time required for the simulation to run in comparison with the time spent on projecting the solutions onto the reference target mesh.
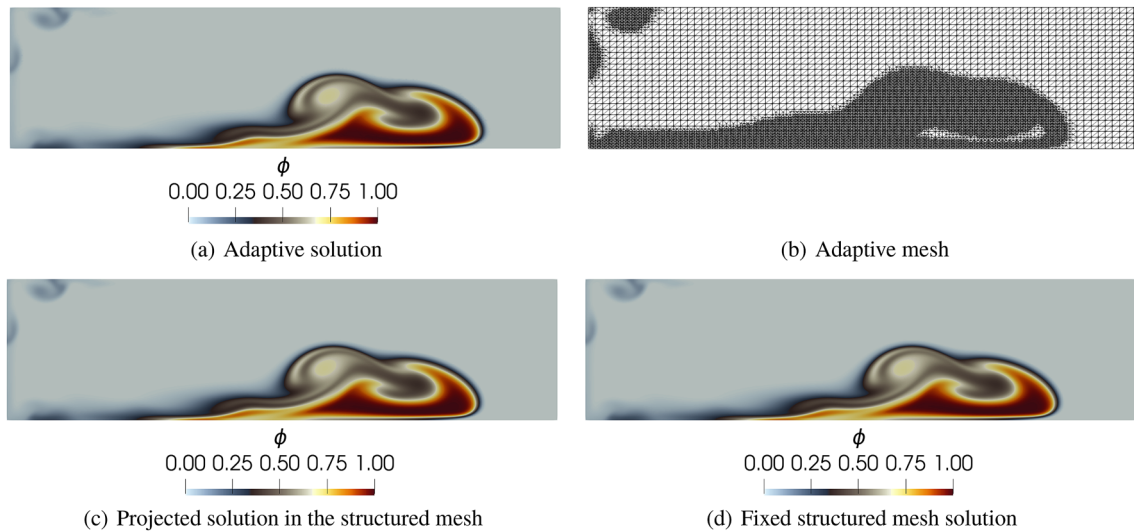
(a) Adaptive solution



(b) Adaptive mesh



(c) Projected solution in the structured mesh



(d) Fixed structured mesh solution

**Fig. 14** Results and mesh for the first 8 m of the domain at $t = 10$s
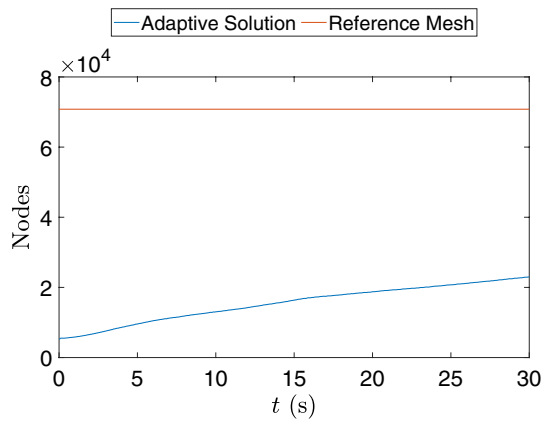


**Fig. 15** Number of mesh nodes in time for the adaptive solution and the proposed reference mesh for the density-driven gravity current example
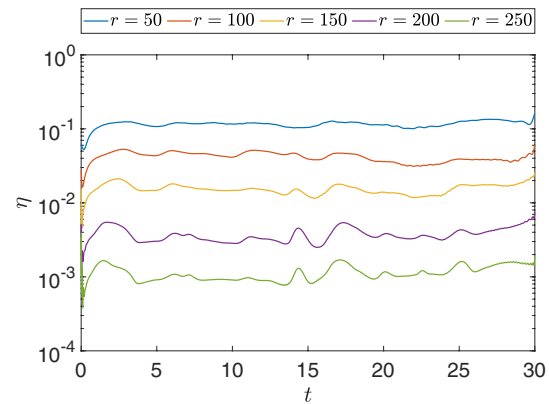


**Fig. 16** Relative error for the reconstruction considering different values of the rank $r$

**Table 5** Absolute and relative time required for the projection routine in comparison with the adaptive finite element simulation code (AMR/C FEM) for the lock-exchange example

| Code part | Absolute time (s) | Relative time (%) |
|---|---|---|
| AMR/C FEM | 15748.11 | 94.94 |
| Mesh projection | 839.42 | 5.06 |

**Table 6** Relative error between reconstructed data and the snapshots and speedup between DMD and the numerical simulation

| Rank $r$ | Relative error ($\eta_F$) | Speedup |
|---|---|---|
| 50 | $1.081 \times 10^{-1}$ | 558.53 |
| 100 | $3.472 \times 10^{-2}$ | 445.91 |
| 150 | $6.207 \times 10^{-3}$ | 230.67 |
| 200 | $2.024 \times 10^{-3}$ | 204.74 |
| 250 | $9.604 \times 10^{-4}$ | 189.17 |

Now we proceed applying DMD to the projected solution and reconstructing the solutions. Figure 16 shows the relative error for the reconstruction using different values of the rank $r$. The results are also confirmed in Table 6, where the relative error between the reconstructed and projected snapshot matrices is shown and the speedup computed for each

case. We observe that, for increasing values of $r$, the relative error decreases for all steps and affects the overall relative error of the matrices. That is, inserting more structures in the DMD basis yields better accuracy in terms of overall relative error. As for the speedup, we notice that it decreases for
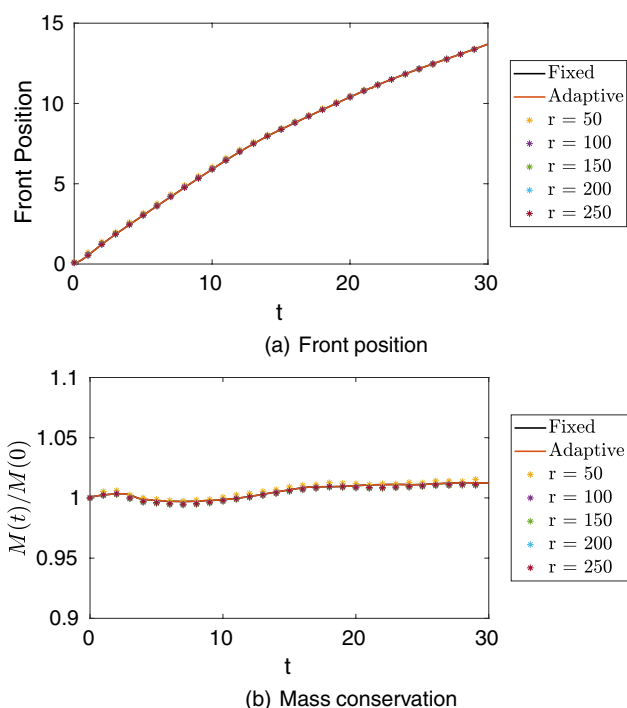
(a) Front position



(b) Mass conservation

**Fig. 17** Front position and mass conservation for the fixed mesh and adaptive mesh simulations and reconstructions with the target mesh
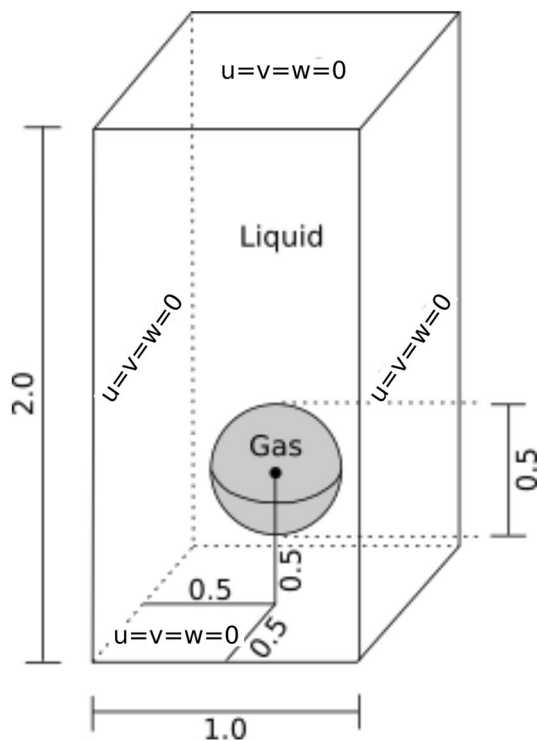
increasing values of $r$. Such a decrease occurs because larger values of $r$ directly affect the rSVD algorithm performance [7, 25] and increase the dimensions of the matrices for the computation of the DMD basis.

Most importantly, we can evaluate the quantities of interest common to density-driven gravity currents. Figure 17 shows the front position and mass conservation for both simulations and the respective reconstructions. We note that, for these quantities all reconstructions are in extremely good agreement with those computed with the fixed and adaptive meshes, approaching them, as expected, for higher values $r$.

### 4.2.2 Bubble rising problem

We now study a bubble rising 3D benchmark, whose task is to track the evolution of a three-dimensional bubble rising in a liquid column. The initial configuration is described in Fig. 18.

For this problem, we couple the Navier–Stokes equations with an interface capturing method called convected level-set [20, 33, 65]. The convected level-set is a method used to represent the interface between two phases and, by a convection equation, to move the interface as the flow evolves. A force that has an important role in bubble problems is the surface tension $\mathbf{F}_{st}$, which is applied using the Continuum Surface Model (CSF) [10].

We write the governing equations in their dimensional form as

$$\nabla \cdot \mathbf{u} = 0,$$

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot p \nabla \mathbf{u} + \nabla p - \mu \nabla^2 \mathbf{u} - \rho \mathbf{g} - \mathbf{F}_{st} = \mathbf{0},$$

$$\frac{\partial \alpha}{\partial t} + (\mathbf{u} + \lambda \mathbf{U}) \cdot \nabla \alpha - \lambda \operatorname{sgn}(\alpha) S = 0,$$

$$(29)$$

where $\rho$ is the density, $\mu$ is the dynamic viscosity, $\mathbf{g}$ is the acceleration of gravity vector, $\alpha$ is the level-set function, $\lambda$



**Fig. 18** Initial configuration and boundary conditions for the bubble rising problem

**Table 7** Rising bubble data

| Computational domain | $1 \times 1 \times 2$ | (m) |
|---|---|---|
| Grid sizes | 0.100 to 0.025 | (m) |
| Number of time steps | 240 | (–) |
| Time step | 0.0125 | s |
| Bubble radius | 0.25 | m |
| Initial bubble position | $(x, y) = (0.5, 0.5, 0.5)$ | m |
| Liquid density | 1000 | kg/m$^3$ |
| Liquid viscosity | 10 | kg/(ms) |
| Gas density | 100 | kg/m$^3$ |
| Gas viscosity | 1 | kg/(ms) |
| Surface tension | 24.5 | N/m |
| Gravity | 0.98 | m/s$^2$ |

(a) Adaptive mesh solution

(b) Coarse mesh projection

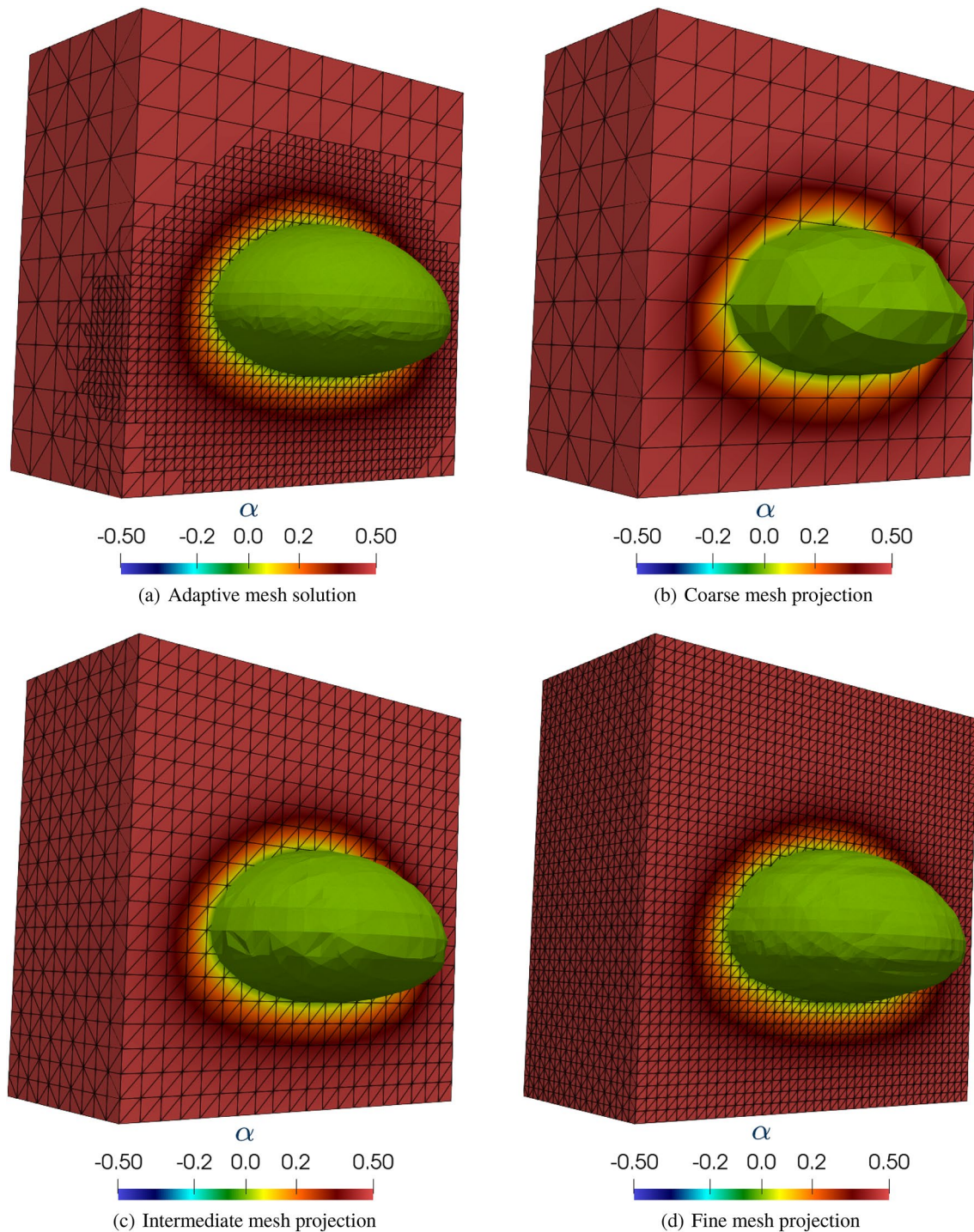(c) Intermediate mesh projection

(d) Fine mesh projection

**Fig. 19** Level-set solution detail at $t = 3.0\text{s}$ and projection to the coarse, medium and fine meshes

is a penalty constant, $\mathbf{U} = \text{sgn}\,(\alpha)\dfrac{\nabla\alpha}{||\nabla\alpha||}$, and $S$ a function related to the level-set signed distance function.

For the temporal integration, we apply the Backward–Euler method to the Navier–Stokes equations, while for the convected level-set, we use the BDF2 method. One

may find more details about the governing equations and methods in [33].

The initial configuration consists of a spherical bubble of radius $R = 0.25$ m centered at [0.5, 0.5, 0.5] m in a $[1 \times 1 \times 2]$ m domain. The no-slip boundary condition is

**Table 8** Absolute and relative time required for the projection routine in comparison with the adaptive finite element simulation code (AMR/C FEM) for the bubble rising example

| Target mesh | Code part | Absolute time (s) | Relative time (%) |
|---|---|---|---|
| Coarse | AMR/C FEM | 17345.92 | 99.97 |
| | Mesh projection | 4.41 | 0.03 |
| Intermediate | AMR/C FEM | 17390.03 | 99.85 |
| | Mesh projection | 24.56 | 0.14 |
| Fine | AMR/C FEM | 17653.70 | 98.91 |
| | Mesh projection | 194.87 | 1.09 |

applied to all boundaries. Table 7 lists the parameters used for this simulation.

We use an adapted mesh, initially with $10 \times 10 \times 20$ cells, with each cell divided into 6 linear tetrahedra. We refine the initial region where the bubble is located into two levels and, after the refinement, the smallest element has a size of 0.025 m. The adaptive mesh refinement is based on the flux jump of the level-set function error, in which $h_{max} = 2$. We

apply the adaptive mesh refinement every four time steps. The interface is modeled with $E = 0.05$, and the time step size is defined as $\Delta t = 0.0125s$. We output the projected solutions at every 2 time steps such that $\Delta t_o = 0.025s$. In this example, we consider three tetrahedral meshes for our projection strategy, presented in Fig. 19. The three meshes named coarse, intermediate and fine, present characteristic lengths similar to the three scales existent in the refinement levels of the adaptive mesh simulation. The coarse mesh represents the initial mesh on the adaptive simulation, with 12,000 elements and 2541 nodes. The intermediate mesh presents smaller elements equivalent to the generated elements after the first refinement level on the AMR/C simulation, totalizing 96,000 elements and 18,081 nodes and the fine mesh contains the smallest scales presented on the adaptive numerical solution with 768,000 elements and 136,161 nodes. The figures show the level-set solution on half of the domain and at $t = 3s$ for the adaptive mesh solution and the respective projections. We also present on Table 8 the time required for the projection of the solutions of $\alpha$ on the three



(a) Volume.

(b) Sphericity.
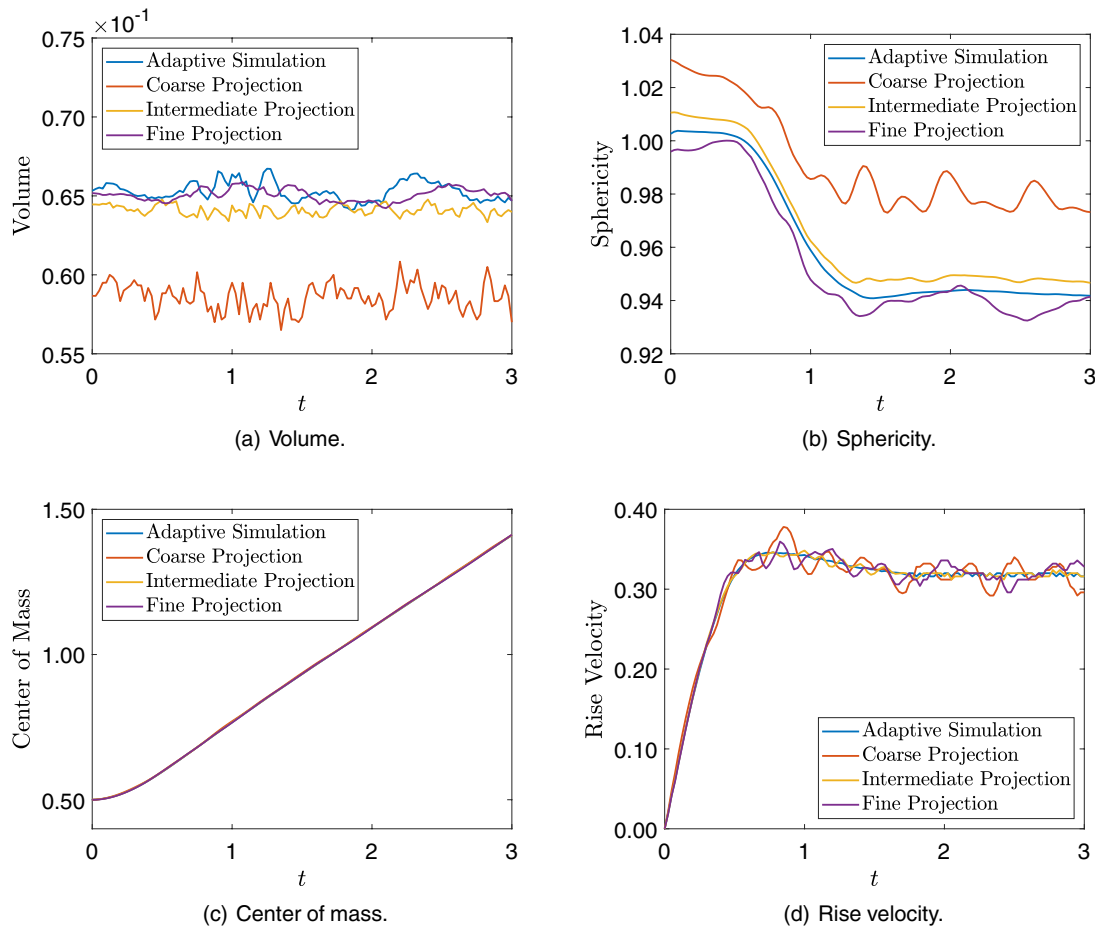
(c) Center of mass.

(d) Rise velocity.

**Fig. 20** Comparison between the simulation and projection of the 3D rising bubble quantities of interest

**Table 9** Relative error between reconstructed data and the projected snapshots and speedup between DMD and the numerical simulation

| Rank $r$ | Mesh | Rel. error ($\eta_F$) | Speedup |
|---|---|---|---|
| 5 | Coarse | $6.222 \times 10^{-2}$ | $1.33 \times 10^5$ |
| | Intermediate | $6.655 \times 10^{-2}$ | $7.95 \times 10^4$ |
| | Fine | $6.865 \times 10^{-2}$ | $1.53 \times 10^4$ |
| 10 | Coarse | $2.384 \times 10^{-2}$ | $1.21 \times 10^5$ |
| | Intermediate | $2.461 \times 10^{-2}$ | $3.60 \times 10^4$ |
| | Fine | $2.490 \times 10^{-2}$ | $1.22 \times 10^4$ |
| 15 | Coarse | $1.415 \times 10^{-2}$ | $1.15 \times 10^5$ |
| | Intermediate | $1.402 \times 10^{-2}$ | $2.46 \times 10^4$ |
| | Fine | $1.429 \times 10^{-2}$ | $7.55 \times 10^3$ |
| 30 | Coarse | $8.900 \times 10^{-3}$ | $1.09 \times 10^5$ |
| | Intermediate | $8.125 \times 10^{-3}$ | $3.29 \times 10^4$ |
| | Fine | $7.687 \times 10^{-3}$ | $6.66 \times 10^3$ |
| 45 | Coarse | $6.382 \times 10^{-3}$ | $1.02 \times 10^5$ |
| | Intermediate | $5.820 \times 10^{-3}$ | $5.35 \times 10^4$ |
| | Fine | $5.776 \times 10^{-3}$ | $5.80 \times 10^3$ |
| 60 | Coarse | $5.659 \times 10^{-3}$ | $8.74 \times 10^4$ |
| | Intermediate | $5.444 \times 10^{-3}$ | $1.45 \times 10^4$ |
| | Fine | $5.518 \times 10^{-3}$ | $5.53 \times 10^3$ |

Results presented for multiple values of $r$
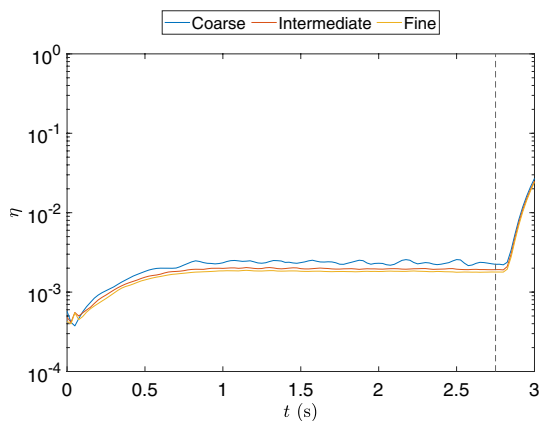


**Fig. 21** Relative error for the rising bubble example for the coarse, intermediate, and fine mesh solutions. The dashed line defines the start of the prediction phase

meshes in terms of absolute and relative time. We can see that the projection time is very small.

To verify if the bubble geometry and dynamics are being preserved during the mesh projections, we evaluate the quantities of interest such as the bubble volume, sphericity, center of mass, and rise velocity. Bubble volume and sphericity are related to the bubble geometry, while the position of the bubble center of mass and rise velocity regard the

bubble dynamics. The results are compared for the AMR/C simulation output and the projections on Fig. 20. The quantities of interest are computed using the adaptive snapshots as well as the projected snapshots and compared with each other. Regarding the geometry quantities of interest, we observe that the coarse mesh projection solution affects the bubble's geometry, leading to bad results regarding the bubble volume and sphericity compared with the AMR/C simulation. For the intermediate and the fine meshes, the bubble's geometry is not largely affected, and the results are compatible with the AMR/C results. As for the quantities of interest related to the bubble dynamics, no significant difference is observed on the projection of the three meshes. We observe that the center of mass is not affected by the projections compared with the simulation results. As for the rise velocity, we observe some minor differences in all projection cases.

The simulation proceeds until $T = 3.0$s, yielding a dataset containing 240 snapshots regarding the solution of $\alpha$ for each target mesh from the projections. We do not consider the use of velocities and pressure in the DMD analysis, reducing the required data in 80%. We consider the results for the first 2.75s to construct the basis and predict the last 0.25 s. We compare the DMD results in terms of relative error between the snapshot matrix and the obtained solutions for each case. The DMD solution for the coarse mesh is compared to the projected adaptive solution onto the coarse mesh and so forth. Results are evaluated for multiple values of $r$, such that $r = \{5, 10, 15, 30, 45, 60\}$. The results regarding accuracy and performance for multiples values of $r$ and the three meshes are presented in Table 9.

The results for $r = 60$ are shown in Fig. 21. We observe that the errors are stable for the reconstruction case, that is, the DMD solution before $t = 2.75$s. From that point, shown as a dashed line on the figure, the errors begin to grow exponentially for each predicted time step, while still remaining below 1% until around 2.9 s. We observe that the errors in the reconstruction case are different regarding the mesh used. That is, the errors are larger with respect to the minimum characteristic length of the projection meshes. However, we observe that the errors grow at the same rate on the prediction phase independently of spatial discretization.

We also show the results considering $r = 60$ for reconstruction at $t = 2.75$s and prediction at $t = 3.0$s in Fig. 22. We compare the DMD results for the three projected meshes, the adaptive mesh solution, and a fixed mesh solution. The fixed mesh solution is obtained by running the simulation with a mesh of 768,000 elements and 136,161 nodes, as the fine mesh used in the projection. We observe initially that the bubble geometry is better defined on the reconstruction than on the prediction figure. This better definition is directly related to the errors observed in Fig. 21. We observe that the coarse mesh results do not capture the bubble geometry with

**Fig. 22** Bubble contour at the vertical mid plane for the signal reconstruction ($t = 2.75$ s) and prediction ($t = 3.0$ s) last steps
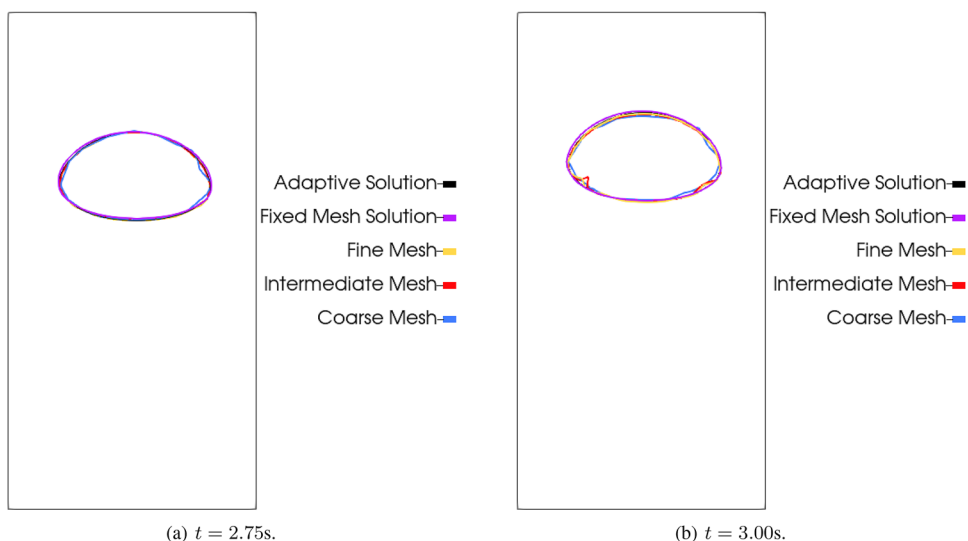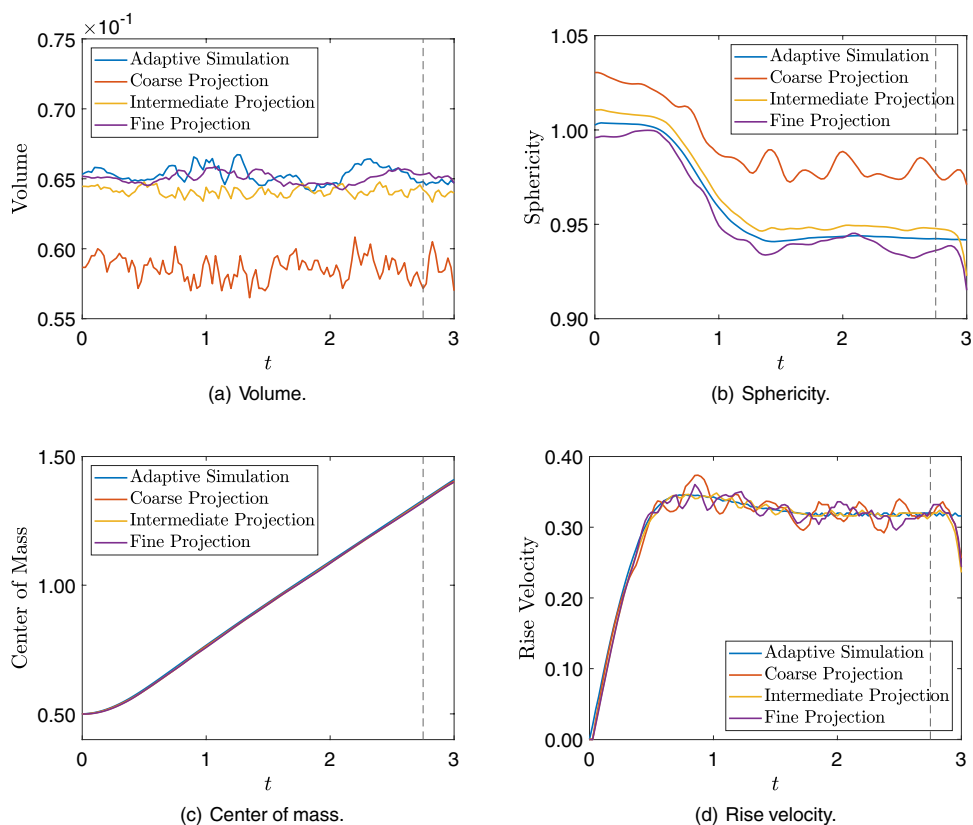


(a) $t = 2.75$s.

(b) $t = 3.00$s.

**Fig. 23** Comparison between the simulation and DMD signal plus prediction of the 3D rising bubble quantities of interest. The dashed line marks the beginning of the prediction regime for the DMD



(a) Volume.

(b) Sphericity.

(c) Center of mass.

(d) Rise velocity.

the same accuracy as the intermediate and fine meshes for the reconstruction results. As for the intermediate and fine meshes, they present similar results in comparison with the projected solutions. However, when we observe the prediction figure, we observe that instabilities inherent to DMD arise on the bubble contour, affecting the bubble geometry for the intermediate and fine mesh.

We now proceed comparing the results in terms of quantities of interest for the DMD results. Figure 23 shows the bubble volume, sphericity, center of mass, and rise velocity for the DMD results compared to the adaptive solution results. The same issue regarding sphericity on the coarse mesh projection is observed on the coarse mesh DMD results. However, for the intermediate and fine meshes, the values match the results observed for the projection. We

observe results in conformity for the center of mass and rise velocity as well. In terms of prediction, we observe that DMD accurately predicts the volume and the center of mass evolution. As for the other quantities of interest, the increasing exponential errors in the DMD prediction structure affect the quantities of interest for long time future predictions.

## 5 Conclusions

In this work, we propose a strategy to enable data-driven snapshot-based methods on finite element solutions obtained by AMR/C simulations. The use of AMR/C algorithms in finite element approximations of PDEs is known to reduce memory usage and to increase the efficiency of the simulations without compromising the accuracy. The simulation adapts the mesh with the evolution of the solution, refining regions of interest and coarsening regions that are not of interest. The adaptation process leads to different meshes during the simulations, and the solutions have different dimensions and topologies (or different connectivities and nodal coordinates) whenever the AMR/C algorithm is invoked. Snapshots with different dimensions prevent the use of snapshot-based algorithms such as DMD, where the snapshot matrix is built by stacking the solutions for each observation in columns. In this study, we considered a strategy to project the adaptive solutions on reference meshes such that all snapshots present the same dimensions and nodal indices, enabling the construction of snapshot matrices. The method employed to project the solutions onto the target reference mesh is the $\mathcal{L}^2$-projection, a simple, fast and versatile approach. The $\mathcal{L}^2$-projection is a common strategy present in several finite element libraries and frameworks and consists basically of solving a linear system, where the matrix is obtained by a self-adjoint operator enabling the use of efficient solvers. Despite presenting drawbacks, especially regarding properties' conservation, we investigate the use of the $\mathcal{L}^2$-projection as a postprocessing tool without any relevant issues. By postprocessing, we mean that the $\mathcal{L}^2$-projection algorithm is invoked only to insert the solutions on a reference function space for each time step and output the files. This strategy does not yield significant additional computational effort as we observe that the $\mathcal{L}^2$-projection routine required around 1% of the computational time required for the adaptive finite element code to run in most cases and 5% in the worst case. When the source code is not available to invoke the $\mathcal{L}^2$-projection, one can construct the solutions from the output files and project them onto the target reference mesh in a complete non-intrusive workflow.

We test the algorithm on several models presenting different dynamics and underlying physics. First, we present the results for DMD on a continuous SEIRD model for COVID-19 for fictional data in 1D and real data for Lombardy, Italy,

in 2D. The idea of considering short-time future estimates on COVID-19 models could improve the decision-making of public policies to avoid further contamination, and the use of AMR/C in the simulations, coupled with the presented projection scheme and DMD, can lead to fast and reliable predictions. The simulations are implemented using the `libMesh` library using its refinement built-in functions. We compare the solution, its projection in the reference mesh, and the adaptive solution and notice that no relevant errors are found for the projection strategy. In this example, the $\mathcal{L}^2$-projection consists of 1.47% and 1.48% of the total time required for the code to run for the 1D and 2D cases, respectively. As for the use of short-time future estimates, we consider DMD for predicting two weeks in the future given a set of snapshots. For the 1D case, we feed DMD with snapshots covering 30 days, while for the 2D case, the snapshot matrix comprises 46 days of simulation results. The DMD results are presented in terms of efficiency and accuracy compared to the projected solutions, and we observe that the predictions are mostly in good agreement except for the exposed compartment. To test the approach's agnosticism regarding the dynamics of the systems, we consider two different fluid dynamics problems: a 2D density-driven gravity current and a 3D bubble rising problem. We use a 3000-snapshot simulation to generate a basis containing the spatio-temporal coherent structures for the density-driven gravity flow. The lock-exchange simulation is solved using the `FEniCS` framework. The computational effort regarding the projection, in this case, corresponds to 5.06% of the overall computational time. As for the DMD analysis, we evaluate the results for different values of the rank $r$ and observe that, as we increase it, in this case, the overall relative errors decrease significantly. However, when evaluating the quantities of interest such as mass conservation and front position, even the worst case yields good approximations. For the 3D example, we use the `libMesh` library, testing three different reference meshes considering the three element sizes existent in the adaptive simulation. We investigate the occurrence of projecting the adaptive solution onto meshes that do not necessarily guarantee that all the scales obtained in the solution are preserved in terms of quantities of interest related to the bubble geometry and dynamics. We observe that the use of $\mathcal{L}^2$-projection on coarser meshes leads to issues regarding the shape of the bubble, affecting quantities of interest such as bubble volume and sphericity. When considering the $\mathcal{L}^2$-projection on meshes with approximately the same spatial resolution as the finest scale in the adaptive meshes, these effects are mitigated. As for the bubble dynamics, even coarser meshes reveal similar results for the bubble center of mass and rise velocity in comparison with finer mesh projections and the adaptive solution itself. In terms of computational effort, the $\mathcal{L}^2$-projection corresponds to 0.03%, 0.14% and 1.09% of the

total simulation time for the coarse, intermediate, and fine meshes, respectively. Proceeding to the DMD analysis, we present the results for a 220 time step reconstructions and a 20 time step prediction for the bubble rising problem. We test the approximations for multiple values of $r$ being $r = 60$ our best approximations. The results computed by DMD are presented in terms of efficiency and accuracy compared to the projected solutions. We also reevaluate the quantities of interest, now with the DMD approximation. For the reconstruction, results are practically identical to those observed for the projection, while for the prediction, the center of mass and bubble volume yield good results while sphericity and rise velocity are not in agreement with those obtained in the simulation.

Summarizing our findings, we highlight that the present approach, despite its simplicity:

- is versatile since every finite element library that presents AMR/C is often equipped with efficient projection algorithms;
- is fast in the sense that the $\mathcal{L}^2$-projection only requires solving a sparse, well-conditioned linear system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is the mass matrix, obtained by a self-adjoint operator, enabling the use of efficient solvers;
- is framework-agnostic, as noted in the results of our simulations implemented on `libMesh` and `FEniCS`, requiring only the standard procedures already in place in both libraries;
- is problem-agnostic, as observed on meshes with different spatial dimensions (1D, 2D, and 3D), topologies (structured and unstructured), and equations (convection-dominated and reaction–diffusion systems);
- preserves the dynamics computed on the adaptive snapshots for all the tested examples.

The results observed in this study reveal that the use of mesh projections for adaptive solutions preserves the dynamics inherent to the finite element solutions. However, this approach on large problems could be inefficient or prohibitive since the mesh projection on a sufficiently fine target reference mesh can lead to unfeasible DMD storage and computation times. As a future work, aiming to circumvent this issue, we plan to explore the use of data compression on DMD, once used for streaming purposes [24], in the mesh projection context. Also, it is known that DMD approximates the nonlinear dynamics existent in the dataset using a linear operator, being fragile to problems presenting a strong nonlinearity [43]. To deal with this, several Koopman-based frameworks, such as the Extended DMD [66], the Kernel DMD [67], and LANDO [5], use nonlinear observables to improve temporal predictions. In the future, we plan to evaluate our projection approach on nonlinear observables for further investigations.
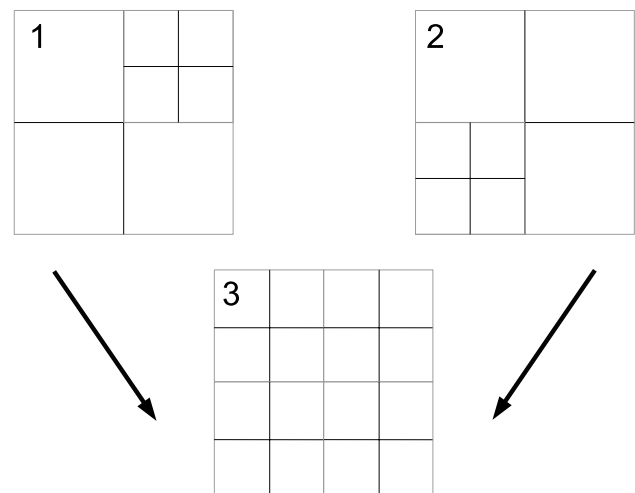


**Fig. 24** Depiction demonstrating the need for projection. Even though Mesh 1 and Mesh 2 have the same number of degrees of freedom, the different topologies require that, for a proper application of DMD, we must project both meshes onto a reference mesh 3 (Mesh 3). As shown, this reference mesh should be sufficiently fine to resolve all necessary in all areas of the domain

## Appendix

In this appendix, we provide a more detailed discussion of several points raised in Sect. 3 regarding the use of DMD on adapted meshes. As discussed in the main text, given the way DMD works by acting on the snapshot matrix, it is, therefore, essential for its proper application that all snapshots are of equal dimension; if the original dataset has been produced using different meshes, this may not be the case. Thus, it is crucial in such a case to use some sort of post-processing technique to provide a uniform snapshot size across the time series. For practical purposes, however, ensuring simply that all snapshots of uniform dimension is a necessary but not sufficient condition to usefully apply DMD. As illustrated in Fig. 24, we show an example of two meshes (labelled as 1 and 2) that both an have equal number of nodes. Assuming the same order of finite element approximation is used on each mesh, the resulting vectors $\mathbf{u}_1^h$ and $\mathbf{u}_2^h$ will have the same dimension. However, while the number of degrees of freedom is the same on each mesh, they are topologically distinct. Hence, to properly perform DMD in this instance, we must have a third mesh, which we call a *reference mesh* (labeled as mesh 3), on which we project *both* $\mathbf{u}_1^h$ and $\mathbf{u}_2^h$. We note that this mesh is refined everywhere, compared to the locally refined mesh 1 and mesh 2. In this way, we guarantee that the fine-scale frequencies captured by all the different mesh levels are properly represented by the DMD modes.

In the current work, we have performed this by using $\mathcal{L}^2$-projection of all snapshots onto a reference (or projected) mesh. Let $\mathbf{u}^h$ denote the finite element solution on the original mesh. Then we define the $\mathcal{L}^2$-projection onto the reference mesh as

$$(\mathbf{u}^{\text{proj}}, \mathbf{v}^{\text{proj}}) = (\mathbf{u}^h, \mathbf{v}^{\text{proj}}), \tag{30}$$

for all $\mathbf{v}^{\text{proj}}$ in $V^{\text{proj}}$. One may quickly verify that this satisfies the *Galerkin orthogonality* property on the reference mesh, i.e., for all $\mathbf{v}^{\text{proj}}$ in $V^{\text{proj}}$, the relation:

$$(\mathbf{u}^{\text{proj}} - \mathbf{u}^h, \mathbf{v}^{\text{proj}}) = 0$$

is satisfied. At the algebraic level, the problem (30) may be expressed as:

$$\mathbf{M}\mathbf{u}^{\text{proj}} = \mathbf{P}\mathbf{u}^h, \tag{31}$$

where $\mathbf{M}$ is the mass matrix on the FEM space $V_h^{\text{proj}}$ and $\mathbf{P}$ is a rectangular matrix consisting of the projection from $V^h$ onto $V^{\text{proj}}$. Letting $N_i^{\text{proj}}$ correspond to the shape functions of $V^{\text{proj}}$ and $N_i^h$ to $V^h$, the matrices have the following definitions:

$$\begin{aligned}(\mathbf{M})_{i,j} &= \int_\Omega N_i^{\text{proj}} N_j^{proj} \\ (\mathbf{P})_{i,j} &= \int_\Omega N_i^{\text{proj}} N_j^h.\end{aligned} \tag{32}$$

We then may view $\mathbf{u}^{\text{proj}}$ as

$$\mathbf{u}^{\text{proj}} = \mathbf{M}^{-1}\mathbf{P}\mathbf{u}^h. \tag{33}$$

Consider now DMD on Mesh 1:

$$\mathbf{Y}_2^h = \mathbf{A}^1\mathbf{Y}_1^h, \quad \text{where } \mathbf{Y}_1^h = \begin{bmatrix} | & | & & | \\ \mathbf{u}^h(t_0) & \mathbf{u}^h(t_1) & \dots & \mathbf{u}^h(t_m) \\ | & | & & | \end{bmatrix}. \tag{34}$$

Similarly, DMD defined on the reference mesh reads:

$$\begin{aligned}\mathbf{Y}_2^{\text{proj}} &= \mathbf{A}^{\text{proj}}\mathbf{Y}_1^{\text{proj}}, \text{ where } \mathbf{Y}_1^{\text{proj}} \\ &= \begin{bmatrix} | & | & & | \\ \mathbf{M}^{-1}\mathbf{P}\mathbf{u}^h(t_0) & \mathbf{M}^{-1}\mathbf{P}\mathbf{u}^h(t_1) & \dots & \mathbf{M}^{-1}\mathbf{P}\mathbf{u}^h(t_m) \\ | & | & & | \end{bmatrix}\end{aligned} \tag{35}$$

$$\mathbf{Y}_2^{\text{proj}} = \mathbf{A}^{\text{proj}}\mathbf{M}^{-1}\mathbf{P}\mathbf{Y}_1^h.$$

Similarly,

$$\mathbf{Y}_2^{\text{proj}} = \mathbf{M}^{-1}\mathbf{P}\mathbf{Y}_2^h. \tag{36}$$

From (35), (36):

$$\mathbf{M}^{-1}\mathbf{P}\mathbf{Y}_2^h = \mathbf{A}^{\text{proj}}\mathbf{M}^{-1}\mathbf{P}\mathbf{Y}_1^h, \text{ implying:}$$
$$\mathbf{Y}_2^h = \mathbf{P}^{-T}\mathbf{M}\mathbf{A}^{\text{proj}}\mathbf{M}^{-1}\mathbf{P}\mathbf{Y}_1^h, \tag{37}$$

where $\mathbf{P}^{-T}$ denotes the left-sided pseudoinverse of $\mathbf{P}$ (see e.g. [31]). Thus, rank$(\mathbf{P}) = \dim(\mathbf{u}_h)$, DMD may properly represent all relevant dynamics on the reference (projected) mesh.

**Theorem** *If every degree of freedom of the original mesh is also a degree of freedom of the reference mesh, then the matrix $\mathbf{P}$ is full-rank.*

**Proof** By assumption, for every $N_i^h$, the corresponding $N_i^{\text{proj}}$ (assumed to be more fine and hence greater in number) is defined such that

$$\text{supp}(N_i^{\text{proj}}) \subset \text{supp}(N_i^h).$$

This idea may be alternatively expressed as

$$V^1 + V^2 + \cdots V^n = V^{\text{proj}},$$

where the different $V^i$ are finite element spaces corresponding to the different adapted meshes. Such a condition was also used in [62]. We, therefore, have that

$$\int_\Omega N_i^h N_j^{\text{proj}} = \kappa_{ij} \int_\Omega N_i^h N_j^h, \tag{38}$$

where $0 < \kappa_{ij} \leq 1$ for all $i, j$ is a volumetric constant relating the measures of supp$(N_j^{\text{proj}})$ and supp$(N_j^h)$. By (38), the matrix $\widetilde{\mathbf{M}}$, defined such that

$$(\widetilde{\mathbf{M}})_{i,j} = \kappa_{ij} \int_\Omega N_i^h N_j^h,$$

which has the same rank (that is, $\dim(V^h)$) as $\mathbf{M}^h$ by construction. We then observe that, for $i, j \leq \dim(V^h)$:

$$(\mathbf{P})_{i,j} = (\widetilde{\mathbf{M}})_{i,j}.$$

Thus, we can express $\mathbf{P}$ as

$$\mathbf{P} = \begin{pmatrix} \widetilde{\mathbf{M}} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \widetilde{\mathbf{P}} \end{pmatrix}.$$

Assume now for the sake of contradiction that $\mathbf{P}$ is not full-rank. This implies that there exists a $\mathbf{b} \in \mathbb{R}^{\dim(V^h)}$ such that $\mathbf{Pb} = \mathbf{0}$. Then

$$\begin{pmatrix} \widetilde{\mathbf{M}}\mathbf{b} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \widetilde{\mathbf{P}}\mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{39}$$

implying that $\widetilde{\mathbf{M}}\mathbf{b} = 0$, contradicting the full-rank of $\widetilde{\mathbf{M}}$. Hence, we can conclude $\mathbf{P}$ is full-rank.

The above theorem requires an assumption that, while often the case in practice, is not strictly necessary in our

experience to obtain acceptable results using DMD. As mentioned, a similar condition was required for the related POD method in [62], in which a common finite element space $V^n$ was implicitly constructed from the different spaces from each adapted mesh. Issues regarding projection between meshes were also studied in [9, 21, 48], in which the interested reader may find more theoretical results regarding projection between meshes. □

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Ahmed N, Rebollo TC, John V, Rubino S (2017) A review of variational multiscale methods for the simulation of turbulent incompressible flows. Arch Comput Methods Eng 24(1):115–164

2. Ainsworth M, Oden JT (2011) A posteriori error estimation in finite element analysis, vol 37. Wiley, New York

3. Alla A, Balzotti C, Briani M, Cristiani E (2020) Understanding mass transfer directions via data-driven models with application to mobile phone data. SIAM J Appl Dyn Syst 19(2):1372–1391

4. Alnæs MS, Blechta J, Hake J, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The Fenics project version 15. Arch Numer Softw 3:100

5. Baddoo PJ, Herrmann B, McKeon BJ, Brunton SL (2021) Kernel learning for robust dynamic mode decomposition: linear and nonlinear disambiguation optimization (LANDO). arXiv:2106.01510

6. Baker N, Alexander F, Bremer T, Hagberg A, Kevrekidis Y, Najm H, Parashar M, Patra A, Sethian J, Wild S, Willcox K, Lee S (2019) Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. United States. https://doi.org/10.2172/1478744.

7. Barros GF, Côrtes AMA, Coutinho AL (2020) Dynamic mode decomposition for density-driven gravity current simulations. In: CILAMCE 2020-Proceedings of the XLI Ibero-Latin-American congress on computational methods in engineering. ISSN 2675-6269

8. Bazilevs Y, Takizawa K, Tezduyar TE (2013) Computational fluid-structure interaction: methods and applications. Wiley, New York

9. Bolten M, Donatelli M, Huckle T, Kravvaritis C (2015) Generalized grid transfer operators for multigrid methods applied on toeplitz matrices. BIT Numer Math 55(2):341–366

10. Brackbill JU, Kothe DB, Zemach C (1992) A continuum method for modeling surface tension. J Comput Phys 100(2):335–354

11. Brauer F, Castillo-Chavez C, Feng Z (2019) Mathematical models in epidemiology. Springer, Berlin

12. Brunton SL, Kutz JN (2019) Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge University Press, Cambridge

13. Brunton SL, Noack BR, Koumoutsakos P (2020) Machine learning for fluid mechanics. Annu Rev Fluid Mech 52:477–508

14. Brunton SL, Proctor JL, Tu JH, Kutz JN (2015) Compressed sensing and dynamic mode decomposition. J Comput Dyn 2(2):165–191

15. Burstedde C, Ghattas O, Gurnis M, Isaac T, Stadler G, Warburton T, Wilcox L (2010) Extreme-Scale AMR, SC '10: Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis, pp 1–12. https://doi.org/10.1109/SC.2010.25

16. Calmet H, Pastrana D, Lehmkuhl O, Yamamoto T, Kobayashi Y, Tomoda K, Houzeaux G, Vázquez M (2020) Dynamic mode decomposition analysis of high-fidelity CFD simulations of the sinus ventilation. Flow Turbul Combust 105(3):699–713

17. Carey GF (1997) Computational grids: generations, adaptation and solution strategies series in computational and physical processes in mechanics. Taylor & Francis, New York

18. Carey GF, Bicken G, Carey V, Berger C, Sanchez J (2001) Locally constrained projections on grids. Int J Numer Methods Eng 50(3):549–577

19. Codina R, Badia S, Baiges J, Principe J (2018) Variational multiscale methods in computational fluid dynamics. Encyclop Comput Mech Sec Ed 20:1–28

20. Coupez T (2007) Convection of local level set function for moving surfaces and interfaces in forming flow. AIP Conf Proc 908:61–66. https://doi.org/10.1063/1.2740790

21. Dickopf T, Krause R (2011) A study of prolongation operators between non-nested meshes. domain decomposition methods in science and engineering XIX. Springer, Berlin, pp 343–350

22. Donoho DL (1995) De-noising by soft-thresholding. IEEE Trans Inf Theory 41(3):613–627

23. Eckart C, Young G (1936) The approximation of one matrix by another of lower rank. Psychometrika 1(3):211–218

24. Erichson NB, Brunton SL, Kutz JN (2019) Compressed dynamic mode decomposition for background modeling. J Real-Time Image Proc 16(5):1479–1492

25. Erichson NB, Voronin S, Brunton SL, Kutz JN (2019) Randomized matrix decompositions using R. J Stat Softw 89(1):1–48

26. Fang F, Pain CC, Navon IM, Piggott MD, Gorman GJ, Allison PA, Goddard AJ (2009) Reduced-order modelling of an adaptive mesh ocean model. Int J Numer Methods Fluids 59(8):827–851

27. Farrell PE, Maddison JR (2011) Conservative interpolation between volume meshes by local Galerkin projection. Comput Methods Appl Mech Eng 200(1–4):89–100

28. Fonzi N, Brunton SL, Fasel U (2020) Data-driven nonlinear aeroelastic models of morphing wings for control: data-driven

nonlinear aeroelastic models. Proc R Soc A Math Phys Eng Sci 476:2239

29. Gavish M, Donoho DL (2014) The optimal hard threshold for singular values is $4/\sqrt{3}$. IEEE Trans Inf Theory 60(8):5040–5053

30. Geuzaine C, Remacle J-F (2009) Gmsh: a 3-d finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer Methods Eng 79(11):1309–1331

31. Golub GH, Van Loan CF (2013) Matrix computations, vol 3. JHU Press, Baltimore

32. Gräßle C, Hinze M (2018) POD reduced-order modeling for evolution equations utilizing arbitrary finite element discretizations. Adv Comput Math 44(6):1941–1978

33. Grave M, Camata JJ, Coutinho AL (2020) A new convected level-set method for gas bubble dynamics. Comput Fluids 209:104667

34. Grave M, Coutinho ALGA (2021) Adaptive mesh refinement and coarsening for diffusion–reaction epidemiological models. Comput Mech 67:1177–1199

35. Guerra GM, Zio S, Camata JJ, Rochinha FA, Elias RN, Paraizo PL, Coutinho AL (2013) Numerical simulation of particle-laden flows by the residual-based variational multiscale method. Int J Numer Methods Fluids 73(8):729–749

36. Guo H, Zhuang X, Rabczuk T (2019) A deep collocation method for the bending analysis of Kirchhoff plate. Comput Mater Contin 59(2):433–456

37. Hale JS, Schenone E, Baroli D, Beex LA, Bordas SP (2021) A hyper-reduction method using adaptivity to cut the assembly costs of reduced order models. Comput Methods Appl Mech Eng 380:113723

38. Halko N, Martinsson P-G, Tropp JA (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53(2):217–288

39. Hughes TJR, Scovazzi G, Franca LP (2017) Multiscale and stabilized methods. In: Stein E, Borst R, Hughes TJR (eds) Encyclopedia of computational mechanics second edition. https://doi.org/10.1002/9781119176817.ecm2051

40. Héas P, Herzet C (2020) Low-rank dynamic mode decomposition: optimal solution in polynomial-time. arXiv:1610.02962

41. Jovanović MR, Schmid PJ, Nichols JW (2014) Sparsity-promoting dynamic mode decomposition. Phys Fluids 26:2

42. Kirk BS, Peterson JW, Stogner RH, Carey GF (2006) libmesh: a c++ library for parallel adaptive mesh refinement/coarsening simulations. J Eng Comput 22(3):237–254

43. Kutz JN, Brunton SL, Brunton BW, Proctor JL (2016) Dynamic mode decomposition: data-driven modeling of complex systems. SIAM, Philadelphia

44. Kutz JN, Fu X, Brunton SL (2016) Multiresolution dynamic mode decomposition. SIAM J Appl Dyn Syst 15(2):713–735

45. Li A, Barati Farimani A, Zhang YJ (2021) Deep learning of material transport in complex neurite networks. Sci Rep 11:1

46. Li A, Chen R, Farimani AB, Zhang YJ (2020) Reaction diffusion system prediction based on convolutional neural network. Sci Rep 10:1

47. Löhner R (2008) Applied computational fluid dynamics techniques: an introduction based on finite element methods. Wiley, New York

48. Maddison J, Hiester H (2017) Optimal constrained interpolation in mesh-adaptive finite element modeling. SIAM J Sci Comput 39(5):A2257–A2286

49. Necker F, Härtel C, Kleiser L, Meiburg E (2002) High-resolution simulations of particle-driven gravity currents. Int J Multiph Flow 28(2):279–300

50. Pont A, Codina R, Baiges J (2017) Interpolation with restrictions between finite element meshes for flow problems in an ale setting. Int J Numer Methods Eng 110(13):1203–1226

51. Proctor JL, Eckhoff PA (2015) Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. Int Health 7(2):139–145

52. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707

53. Rasthofer U, Gravemeier V (2017) Recent developments in variational multiscale methods for large-eddy simulation of turbulent flow. Arch Comput Methods Eng 20:1–44

54. Rivara MC (1984) Mesh refinement processes based on the generalized bisection of simplices. SIAM J Numer Anal 21(3):604–613

55. Rowley CW, Mezic I, Bagheri S, Schlatter P, Henningson DS (2009) Spectral analysis of nonlinear flows. J Fluid Mech 641:115–127

56. Schmid PJ (2010) Dynamic mode decomposition of numerical and experimental data. J Fluid Mech 656:5–28

57. Schmid PJ (2011) Application of the dynamic mode decomposition to experimental data. Exp Fluids 50(4):1123–1130

58. Sirovich L (1987) Turbulence and the dynamics of coherent structures I, II and III. Q Appl Math 45(3):561–590

59. Taira K, Brunton SL, Dawson ST, Rowley CW, Colonius T, McKeon BJ, Schmidt OT, Gordeyev S, Theofilis V, Ukeiley LS (2017) Modal analysis of fluid flows: an overview. AIAA J 55(12):4013–4041

60. Tajdari M, Pawar A, Li H, Tajdari F, Maqsood A, Cleary E, Saha S, Zhang YJ, Sarwark JF, Liu WK (2021) Image-based modelling for adolescent idiopathic scoliosis: mechanistic machine learning analysis and prediction. Comput Methods Appl Mech Eng 20:374

61. Thompson RA (2015) Galerkin projections between finite element spaces. Technical report, Virginia Polytechnic Institute and State University. http://hdl.handle.net/10919/52968

62. Ullmann S, Rotkvic M, Lang J (2016) Pod-Galerkin reduced-order modeling with adaptive finite element snapshots. J Comput Phys 325:244–258

63. Viguerie A, Lorenzo G, Auricchio F, Baroli D, Hughes TJ, Patton A, Reali A, Yankeelov TE, Veneziani A (2021) Simulating the spread of COVID-19 via a spatially-resolved susceptible-exposed-infected-recovered-deceased (SEIRD) model with heterogeneous diffusion. Appl Math Lett 20:111

64. Viguerie A, Veneziani A, Lorenzo G, Baroli D, Aretz-Nellesen N, Patton A, Yankeelov TE, Reali A, Hughes TJ, Auricchio F (2020) Diffusion-reaction compartmental models formulated in a continuum mechanics framework: application to covid-19, mathematical analysis, and numerical study. Comput Mech 66:1131–1152

65. Ville L, Silva L, Coupez T (2011) Convected level set method for the numerical simulation of fluid buckling. Int J Numer Methods Fluids 66(3):324–344

66. Williams MO, Kevrekidis IG, Rowley CW (2015) A data-driven approximation of the Koopman operator: extending dynamic mode decomposition. J Nonlinear Sci 25(6):1307–1346

67. Williams MO, Rowley CW, Kevrekidis IG (2015) A kernel-based method for data-driven Koopman spectral analysis. J Comput Dyn 2(2):247–265