Contents lists available at ScienceDirect

SoftwareX

Original software publication

MODULO: A software for Multiscale Proper Orthogonal Decomposition of data

Davide Ninni[1], Miguel A. Mendez [*]

von Karman Institute for Fluid Dynamics, EA Department, 1640 Sint-Genesius-Rode, Belgium

ARTICLE INFO

ABSTRACT

In the era of the Big Data revolution, methods for the automatic discovery of regularities in large datasets are becoming essential tools in applied sciences. This article presents an open software package, named MODULO (MODal mULtiscale pOd), to perform the Multiscale Proper Orthogonal Decomposition (mPOD) of numerical and experimental data. This novel decomposition combines Multi-resolution Analysis (MRA) and standard Proper Orthogonal Decomposition (POD) to allow for the optimal compromise between decomposition convergence and spectral purity of its modes. The software is equipped with a Graphical User Interface (GUI) and enriched by numerous examples and video tutorials (see Youtube channel MODULOmPOD). The MATLAB source codes and an executable for Windows users can be downloaded at https://github.com/mendezVKI/MODULO/releases; a collections of exercises in Matlab and Python are provided in https://github.com/mendezVKI/MODULO.

Code metadata

Current code version	v1.1.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_181
Legal Code License	GNU General Public License v3.0
Code versioning system used	None
Software code languages, tools, and services used	Matlab and Python
Compilation requirements, operating environments & dependencies	Running the source code requires MATLAB 2017B or higher. Required Toolboxes: Statistics and Machine Learning Toolbox v11.6, Signal Processing Toolbox v8.3 and Image Processing Toolbox v11.0). The executable can be installed by any Microsoft Windows user.
If available Link to developer documentation/manual	github.com/mendezVKI/MODULO
Support email for questions	davide.ninni@poliba.it, mendez@vki.ac.be

Software metadata

Current software version	v1.1.1
Permanent link to executables of this version	github.com/mendezVKI/MODULO/releases
Legal Software License	GNU General Public License v3.0
Computing platforms/Operating Systems	Microsoft Windows
Installation requirements & dependencies	None
If available, link to user manual - if formally published include a reference to the publication in the reference list	github.com/mendezVKI/MODULO
Support email for questions	davide.ninni@poliba.it, mendez@vki.ac.be

1. Motivation and significance

Data driven modal analysis aims at decomposing a dataset as a linear combination of elementary contributions called *modes*. This provides the fundamental framework for many areas of applied

---

* Corresponding author.
E-mail address: mendez@vki.ac.be (M.A. Mendez).
[1] Currently at Politecnico di Bari, Dipartimento di Meccanica, Matematica & Management, 70125 Bari, Italy.

mathematics and related applications, including pattern recognition, machine learning, data compression, filtering and model order reduction [1,2]. Linear decomposition consists in projecting the dataset onto a suitable space, spanned by a basis that is, hopefully, better capable of capturing the essential features of the data.

A *mode* is computed by projecting the data onto a certain element of a basis. In most of the engineering applications, the data results from the discretization, or sampling of a real quantity (e.g. grayscale entries in images, pressure fields in a fluid flow simulation or stress fields in solid mechanics) over a spatial discretization $\mathbf{x}_i$ and a temporal discretization t_k. Each of the modes produced by a decomposition has its own spatial and temporal structure.

In pattern recognition, one aims at linking modes to specific patterns of interest [3,4]. In filtering or data compression, one aims at removing modes that describe unwanted features or modes that do not significantly contribute to the data [5,6]. In machine learning, this is often a fundamental pre-processing step for many supervised or unsupervised problems [7]. In model order reduction, one aims at projecting partial differential equations onto the space spanned by a few of the leading modes [8,9], thus significantly reducing the computational complexity of a problem and eventually enabling system identification methods and control [10,11].

Contrary to classical tools such as the Discrete Fourier Transform (DFT) or Discrete Wavelet Transform (DWT), data-driven decompositions tailor their bases to the data investigated. The Proper Orthogonal Decomposition (POD, [12]), also known as Principal Component Analysis (PCA, [13]) or Karhunen–Loéve transform [14] depending on the field, is the most classic example. This decomposition is often implemented using the Singular Value Decomposition (SVD) and constructs its bases such that the error produced by truncating the decomposition to any $r < R$ is minimized.

The convergence optimality comes at the cost of setting no constraints on the frequency content of the structures constituting its modes. In many applications, however, it is of interest to have harmonic decompositions to facilitate physical interpretability. In fluid mechanics, this need has motivated the development of an alternative decomposition known as Dynamic Mode Decomposition (DMD, [15]), which extends the Fourier decomposition to the data-driven paradigm: frequencies are defined from the dataset and not imposed a priori.

There exist nevertheless cases for which both convergence optimality and spectral purity yields poor feature detection capabilities [16]. The lack of frequency constraints in the POD occasionally results in modes that capture phenomena occurring at very different scales. On the other hand, the constraint of purely harmonic modes prevents time–frequency localization and yields convergence problems for datasets that are not strictly periodic.

The Multiscale Proper Orthogonal Decomposition proposed in [16] allows for overcoming the limitations of the two with a hybrid method that combines their advantages. This decomposition has already been successfully used in various experimental [17–20] and numerical works [16,21]. The software package *MODal mULtiscale pOd* (MODULO) described in this work is a Matlab-based software developed at the von Karman Institute for Fluid Dynamics to perform mPOD, POD and DFT on numerical and experimental data. Equipped with a Graphical User Interface (GUI), an executable, and a set of exercises and video tutorials, the use of this software does not require direct interaction with the source code and can thus facilitate applied scientists that are unwilling to enter into technicalities linked to programming. In what follows, we refer the reader to each of the eight video tutorials in MODULO's youtube channel for a more detailed review of the software features.

2. Software description

2.1. Theoretical background

A brief description of the theoretical background of the code is here presented; for more details, the reader is referred to [16] and to the first three video tutorials in the youtube channel. As described in the first video, all the modal decompositions implemented are matrix factorizations of the form

$$D(\mathbf{x_i}, t_k) = \Phi(\mathbf{x_i})\, \Sigma\, \Psi(t_k)^T \tag{1}$$

where D is the data matrix to decompose, here assumed to be function of space ($\mathbf{x}_i$) and time (t_k). Regardless of the dimensionality (e.g. 2D or 3D, scalar or vector) of the data, we here assume that every temporal realization (snapshot) of the data is flattened into a vector and stored as a column of D. Every column of is thus a function of $\mathbf{x}_i$ and every row is a function of t_k. This matrix is thus of size $n_s \times n_t$, with n_s the number of spatial points and n_t is the number of temporal realizations (snapshot), and has rank $R = rank(D) \le N = min(n_s, n_t)$.

The matrices $\Phi \in \mathbb{C}^{n_s \times R}$ and $\Psi \in \mathbb{C}^{n_t \times R}$ collect the spatial and the temporal structures (bases) and $\Sigma = diag[\sigma_1, \sigma_2, \ldots, \sigma_R] \in \mathbb{R}^{R \times R}$ is the diagonal matrix collecting their importance (amplitude). More generally, if other independent variables are considered instead of space and time, Φ and Ψ contain a basis for the columns and the rows of D respectively.

The bold notation used ($\mathbf{x}_i$ or $\mathbf{i}$) denotes linear matrix indices. In the current version of the code, it is assumed that the data has uniform sampling both in space and in time, although the latter constraint can be relaxed for the POD. Consider, for example, a 2D velocity field from planar Particle Image Velocimetry (PIV) sampled over a grid $\mathbf{x_i} = [x_i, y_j]$ containing 128×128 points. Flattening the entire velocity field into the columns of D, the number of spatial points if $n_s = 2 \times 128 \times 128 = 16384$, having concatenated both vector components into a single snapshot.

The matrix D is constructed by assigning every kth snapshot $d_k[\mathbf{i}]$ to a column of D. Here k is the index over the time discretization $t_k = (k-1)\Delta t$, sampled at a frequency $f_s = 1/\Delta t$ with $k = [1, n_t]$. As the spatial structures $\phi_r[\mathbf{i}]$ are columns of Φ, and the temporal structures $\psi_r[k]$ are columns of Ψ, Eq. (1) can be written as a dyadic expansion:

$$D[\mathbf{i}, k] = \sum_{r=1}^{N} \sigma_r \phi_r[\mathbf{i}]\, \psi_r^T[k]. \tag{2}$$

If the summation is truncated at $r < N$, an approximation of the original data is obtained. Since Σ is a diagonal matrix, and both the spatial and temporal structures have unitary norm $\|\phi_r\| = \|\psi_r\| = 1\, \forall r \in [1, R]$, it is easy to see that the decomposition can be completed once either Φ or Ψ are given.

The decompositions implemented in MODULO are the POD, the DFT, and the mPOD, computing first the temporal structures Ψ. All these decompositions have an orthonormal temporal basis ($\Psi^{-1} = \Psi^{\dagger}$, with † denoting Hermitian transpose). Hence, the spatial structures can be easily computed as

$$\Phi = D\, \overline{\Psi}\, \Sigma^{-1}, \tag{3}$$

where the calculation of the diagonal matrix Σ is done from the normalization of the spatial structures, that is $\sigma_r = \|D\psi_r\|$.

The theoretical background for DFT and POD is provided in the second video tutorial while the third video tutorial is dedicated to the mPOD. The computation of the temporal structures for these three decompositions proceeds as follows.

-*DFT*. The temporal basis is the well known Fourier matrix, which can be written as $\Psi_{\mathcal{F}}(i, j) = w^{(i-1)(j-1)}$, with $w = e^{2\pi i/n_t}$ with $i = \sqrt{-1}$. The construction of this matrix is independent from the data, hence the DFT is not *data-driven*. In practice, the multiplication by the Fourier Matrix is carried out using the FFT algorithm.

-*POD*. The temporal basis is computed from the eigenvalue decomposition of $K = D^T D$:

$$\Psi_{\mathcal{P}} :\rightarrow K = \Psi_P \, \Lambda \, \Psi_P^T. \tag{4}$$

This matrix is known as temporal correlation matrix in the fluid mechanics community and the approach implemented is known as Sirovinch's snapshot method [22].

-*mPOD*. The temporal basis is computed via a combination of Multi-resolution analysis and eigenvalue decomposition. The fundamental idea of the mPOD is to perform POD at different scales, each retaining non-overlapping portions of the frequency spectra. For example, assuming that the dataset is sampled at $f_s = 1000$ Hz, one might decide to separate phenomena occurring in the range $[0 - 100]$ Hz, $[100 - 300]$ Hz and $[300 - 500]$ Hz and perform a POD on each of these independently. As described in [16], this could be done by using a filter bank, defined by a frequency splitting vector $F_V = [100, 300]$ Hz, to break the dataset into three contributions, and perform the POD in each of these separately. To reduce the computational cost of this operation, the mPOD performs the MRA on the temporal correlation matrix $K = D^T D$. Given a set of suitable transfer functions $\{\mathcal{H}_m\}_{m=1}^M$, with M the number of scales to identify, the mPOD breaks the correlation matrix as:

$$K = \sum_{m=1}^M K_m = \sum_{m=1}^M \Psi_{\mathcal{F}} \left[\widehat{K} \odot \mathcal{H}_m \right] \Psi_{\mathcal{F}} \tag{5}$$

where $\widehat{K} = \overline{\Psi}_{\mathcal{F}} K \overline{\Psi}_{\mathcal{F}}$ is the 2D Fourier transform of the correlation matrix and $\odot$ is the shur product, that is the entry by entry multiplication.

The filtering operation is designed to preserve the key properties of K: each contribution K_m is symmetric and positive definite and thus equipped with a set of orthonormal eigenvectors (POD modes) and non-negative real eigenvalues:

$$K_m = \Psi_{\mathcal{P}m} \Lambda_m \Psi_{\mathcal{P}m}^T = \sum_{j=1}^{n_m} \lambda_m \psi_{\mathcal{P}mj} \psi_{\mathcal{P}mj}^T, \tag{6}$$

where n_m is the number of non zero eigenvalues at each scale.

These spectral constraints impose that a mode (eigenvector) having frequency content in one scale has no frequency content in the others. Therefore, it is possible to show that the eigenvectors of all the scale are orthogonal complements that span the entire $\mathbb{R}^{n_t}$ space, that is $\sum_{m=1}^M n_m \approx n_t$.

The mPOD basis is then constructed by collecting the POD bases of all scales, sorted by amplitude:

$$\Psi_{\mathcal{M}} = [\Psi_1, \Psi_2 \dots \Psi_M] P_{\Sigma} \tag{7}$$

with P_{Σ} a permutation matrix to rank the structures in decreasing order of energy contribution. This decomposition generalizes POD and DFT: for $M \rightarrow 1$, the mPOD is a standard POD. At the limit $M \rightarrow n_t$, the mPOD is a DFT.

2.2. Software architecture

MODULO has a minimal Graphical User Interface (GUI), shown in Fig. 1, that opens after launching the executable. The decomposition process can be followed from the toolbar in the upper
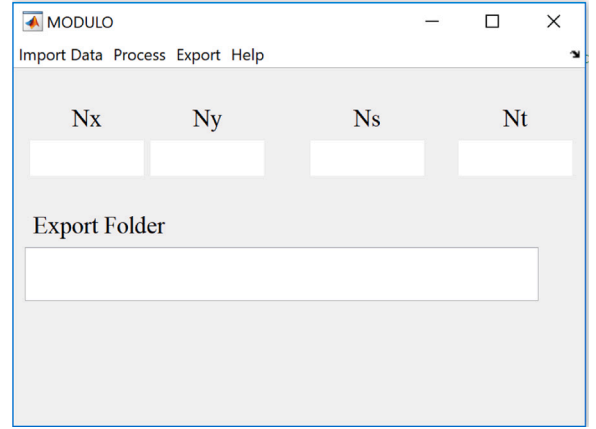

Fig. 1. Main menu of MODULO. Here the user sets the exporting folder and proceeds through the decomposition from the upper toolbar.
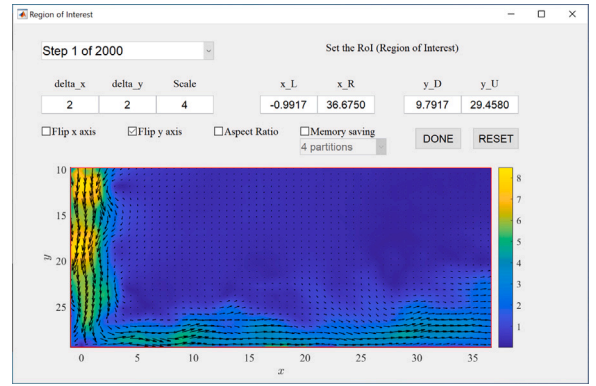

Fig. 2. Region of Interest window, to visualize the imported snapshots, select the portion of the domain that will be analyzed and several plotting parameters.

part of the main menu. An overview of MODULO's GUI is given in the fourth video tutorial.

In this first version of the software, the data is assumed to be 1D or 2D: the first raw of non-editable tabs shows the number of points along the *x*-axis (Nx), the *y*-axis (Ny), the total number of spatial points (Ns) and the number of time steps (Nt). In a 1D test case, it is assumed that Ns=Nx (Ny=0); in a 2D scalar test case one has Ns=Nx×Ny while a 2D vectorial test case yields Ns=2×Nx×Ny.

The dataset can be imported from the menu Import Data, which allows for two options: *embedded mesh* or *separated mesh*. The first option should be used if the mesh grid is stored in each of the data files; the second option should be used if the mesh is stored in another file.

Before importing the data, the user can mean-center all the snapshots, i.e., remove the average column (time average if the row domain is linked to time) from the snapshot matrix D. This option can be useful for plotting purposes in the DFT and is generally recommended for POD and mPOD of statistically stationary datasets. Once the files are loaded, the *Region of Interest* (RoI) GUI shown in Fig. 2 opens.

This GUI allows for setting the portion of the spatial domain that will be used in the decomposition. By default, MODULO considers the entire domain, but the user can introduce the ranges along the horizontal (x_L, x_R) and the vertical (x_L, x_R) axes. The upper left menu pop-up menu allows for selecting and preview any of the imported snapshots. In case of vector datasets, the parameters on the left allow for adjusting the quiver plot in
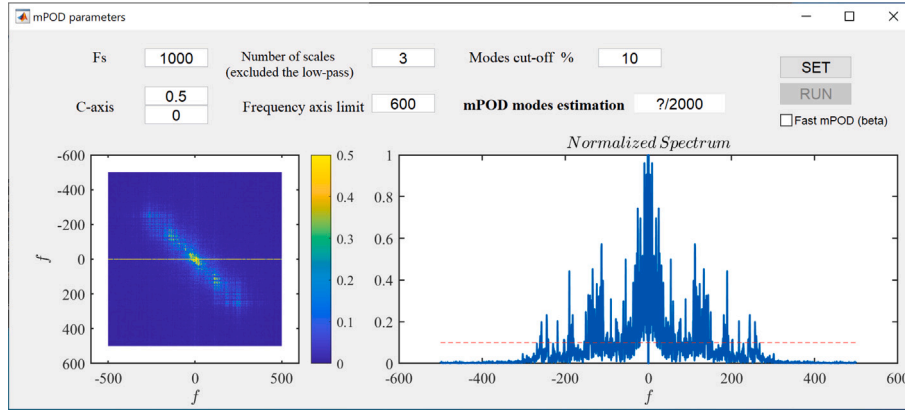
Fig. 3. GUI for introducing the mPOD parameters. During the setting definition, the user can monitor a contour of power spectral density matrix and its diagonal.

terms of spacing (`delta_x`, `delta_y`) and arrow length (`Scale`) while the bottom tick boxes can be used to flip the axes or adjust the axis aspect ratio to 1 : 1. The settings used at the step will be used in the exporting of the spatial structures of the decomposition.

Once these parameters are set, the user can either use the RESET button, to restore default values or the button DONE, to proceed with the importing of the data and the preparation of the matrix D. The *Memory saving* option is described in Section 2.3.1.

2.3. Process

From the `Process` menu, the user can select which of the three decompositions described in Section 2, is to be performed. By choosing POD or DFT algorithms, the user is asked to introduce the sampling frequency f_s and the extreme of the range of the modes to be exported. For all the decompositions, this indexing assumes that the modes are always exported in decreasing order of amplitude, even if DFT and mPOD are not energy-based. Nevertheless, for DFT and mPOD, the need for ordering the modes requires the calculation of the complete basis regardless of the number of exported modes, while for the POD only the modes to be exported are computed.

MODULO is a dimensionless software, hence the units in the sampling frequency need not to be specified: digital frequency bins are computed as $\Delta f = f_s/n_t$ and the frequency domain is $f_n \in [-f_s/2, f_s/2]$. Once these parameters are introduced, the decomposition begins in the case of DFT and POD, and a wait bar indicates the progress of the calculation. The fifth and sixth video tutorials are dedicated to the POD and the DFT respectively.

In the case of mPOD, additional parameters should be introduced, and a dedicated GUI appears to support the user in this process. This GUI is described in the seventh video tutorial and is shown in Fig. 3.

The window shows a contour of the value of the cross-spectral density matrix on the left and its diagonal on the right. On the top-left editable box, the user introduces the sampling frequency `Fs` and the number of scales *excluding the largest*. Assuming, for example, that one is interested in four scales, say $[0 - 100]$Hz, $[100 - 200]$Hz, $[200 - 300]$Hz, and $[300 - Fs/2]$Hz, the number to introduce here is 3. The reason for excluding the first (largest) scale from the counting is that this scale is always kept by default in MODULO while the other scales can be removed if the software is used as a low-pass filter.

The parameters `C-axis` and `Frequency axis limit` can be modified for plotting purposes; the first changes the upper limit in the color axis of the contour plot, the second sets the frequency limits in the axes of both figures.

The last input parameter in the figure is the `Modes cut-off`, which controls the number of modes computed as indicated in the non-editable text box below. A red line also indicates this percentage in the normalized spectrum (right plot). This number controls the number of modes, in each scale, that will be considered in the final mPOD basis: if this is set to, e.g., 10%, then only the modes that have at least 10% of the leading POD mode energy in each scale will be taken. Observe that this estimation is made *before* computing the decomposition based on the transfer function of the filters that will isolate the scales, and is thus only an estimation. This estimation is available only after the settings of the filters in each scale are introduced, from the SET button. The first parameter is the frequency splitting vector: its entries collect the cut-off frequencies on each scale. In the previous example this is $F_V = [100, 200, 300]$Hz (units are customary). The second input is a vector containing the *length of the filter kernels* that will be used to compute each scale. These are FIR filters with a Hamming window. Finally, the last input allows for introducing the *keep vector*. This indicates the intermediate scales that will be kept and allows for using the mPOD also as a filter. In the previous example, if the keep vector is set to $[1, 1, 0]$, then the highest scale $[300 - Fs/2]$Hz is removed from the decomposition. If the keep vector is set to $[0, 0, 0]$ then the mPOD will be equivalent to performing the POD of the dataset obtained by low-pass filtering to keep only the portion $[0 - 100]$Hz.

Once the *frequency splitting vector*, the *length of the kernels* and the *keep vector* are set, the RUN button to start the computation is enabled. As for the DFT, the user inputs the range of modes to be exported only at the end of the decomposition.

2.3.1. Memory saving

The largest matrix in every decompositions is the snapshot matrix $D \in \mathbb{R}^{n_s \times n_t}$. Depending on the computational resources available, this matrix can be prohibitively large, and the cost of matrix products such as the correlation matrix in (4) or the projection in (3) might exceed the available RAM. To cope with this limitation, MODULO offers a 'Memory Saving' option, from the RoI GUI shown in Fig. 2. This option is described in the eighth video tutorial.

When this option is active, all decompositions are performed without loading the snapshot matrix D in memory, but only a few partitions of it at a time. Therefore, the calculation of the time correlation matrix $K = D^T D$, needed for POD and mPOD, is computed from n_P 'column-wise' partitions of D. Each partition D_i^c is of size $n_s \times n_C$ with $n_C = n_t/n_P$. These partitions are saved as temporary *.mat* files and loaded one at a time during the calculation of the correlation, which is performed in blocks. This allows for limiting the number of stored entries to $n_s \times n_C \times 2$ entries, although at the
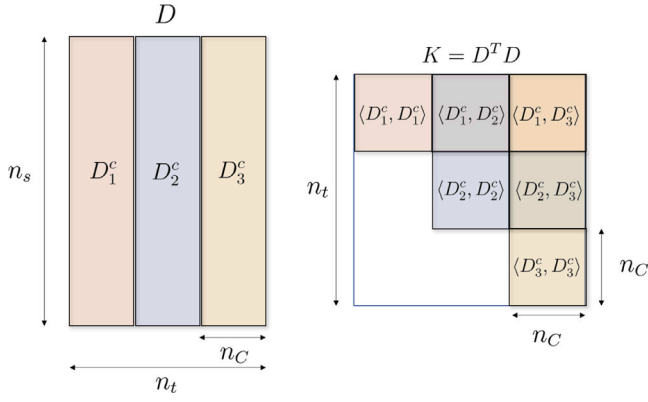
Fig. 4. Calculation of the temporal correlation matrix in (4) using three partitions to limit memory requirements. Only half of the blocks needed are computed; the remaining portion being symmetric.
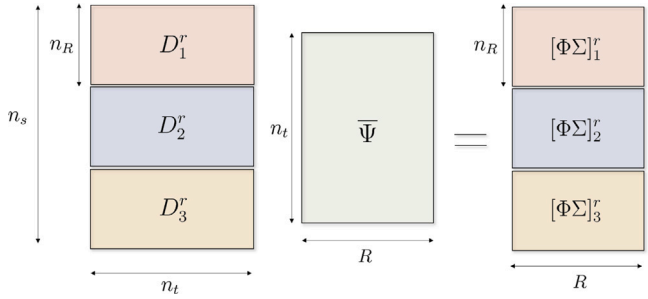
Fig. 5. Calculation of the final projection in (3) using three partitions to limit memory requirements. The final 'row-wise' blocks of $\Phi\Sigma$ must be regrouped into 'column-wise' blocks for the normalization step.

cost of increasing the number of reading/writing operations. The calculation steps for this matrix using three blocks is illustrated in Fig. 4. To limit the cost of the memory saving feature, MODULO takes advantage of the symmetry of the correlation, computing only the upper triangular part of it while the remaining portion is mirrored.

A similar approach is pursued in computing (3), namely the last projection step of every decomposition. In this case, D is split into n_P 'row-wise' partitions D_r, of size $n_R \times n_t$, and the projection is carried out independently in each portions. The calculation steps for this matrix using three blocks is illustrated in Fig. 5. The resulting projection is then assembled back to column-wise partition to compute the amplitude of each mode via column normalization of the matrix $\Phi\Sigma$.

In all the analyzed exercises, the increased computational cost of the memory saving option is of the order of a minute, depending on the decomposition and the size of each snapshot. Table 1 collects the computational time (in seconds) required to perform POD, DFT and mPOD without memory saving ($n_p = 1$) and with memory saving with $n_p = 4$ and $n_p = 12$. The calculations performed on a laptop with Intel(R) Core(TM) i7-7700HQ CPU with 2.8 GHz and 16 GB RAM.

The timing is given in output by MODULO. Hence the timing to prepare the dataset is excluded (since this step is done before selecting the decomposition), while for the mPOD this timing also excludes the preparation of the correlation matrix (since this is done before the mPOD setting GUI opens). The scope of Table 1 is thus not that of comparing the decomposition time

Table 1
Computational time (in seconds) of POD, DFT, and mPOD with no memory saving option ($n_p = 1$) and with memory saving with different partitions. The test case considered is the one from Exercise 4 in MODULO's Github repository, with each snapshot consisting of a 2D velocity field on a $n_s = 13680$ grid. The mPOD is performed with four scales. The time for preparing the dataset matrix is not included; for the mPOD, the timing also excludes the preparation of K.

	Time steps	$n_p = 1$	$n_p = 4$	$n_p = 12$
POD	500	0.56	12.38	17.22
	1000	1.20	23.23	32.22
	2000	4.97	50.87	68.19
DFT	500	3.37	29.49	37.12
	1000	6.5	57.19	73.14
	2000	14.39	110.96	149.78
mPOD	500	1.89	5.01	8.41
	1000	7.80	13.39	20.46
	2000	52.64	65.18	77.86

but comparing how the partitioning influences the timing. For the mPOD, four scales are chosen with kernel widths of 100 and all scales being kept.

The most expensive operation appears to be the final projection with normalization and sorting steps: since the DFT always requires computing n_t modes, this decomposition is more sensitive to the increase of the snapshots. On the other hand, as the POD requires no normalization, the computational costs are much reduced. In general, the computational time increases with the number of partitions due to the increased time spent in reading/writing operations. The price to pay to maintain a limited memory usage appears nevertheless acceptable. Finally, it is worth observing that in case the memory saving is not selected, but the computational resources are not sufficient for the calculation, a warning dialog appears. In this case, the user is strongly advised to either hit "Activate Memory Saving" (in which case the memory saving will be activated with the default number of partitions), either close the warning box and select one of the proposed partitions.

2.4. Export

Through the menu Export, the exporting folder can be chosen. In this folder (which is created if not already available), the modes are saved as .png and .xlsx files. In particular, the excel files are always saved (for the mesh, for the sigmas, for the spatial structures and for the temporal structures) while the pictures are saved only if the tick (checkbox) in the menu Export is signed. In the case of DFT, the temporal structures are not saved since these are sinusoidal with fixed frequencies ($\Psi_\mathcal{F}$ is known a priori as recalled in Section 1). More information on the exported data is discussed in the video tutorials of each decomposition.

3. Illustrative examples

The code repository currently includes five exercises that allow for testing all the features of MODULO and, at the same time, explore the limitations and strengths of each decomposition. These exercises are also solved using various commented *Matlab files* (sorted from 'A to D') in order to let the user follow the decomposition procedures using the source codes. The first exercise presents the analysis of a 1D scalar dataset, which collects the time-dependent velocity profile of a pulsating Poiseuille Flow. As described in the second video tutorial, this dataset can be analytically decomposed in eigenfunction and hence offers a
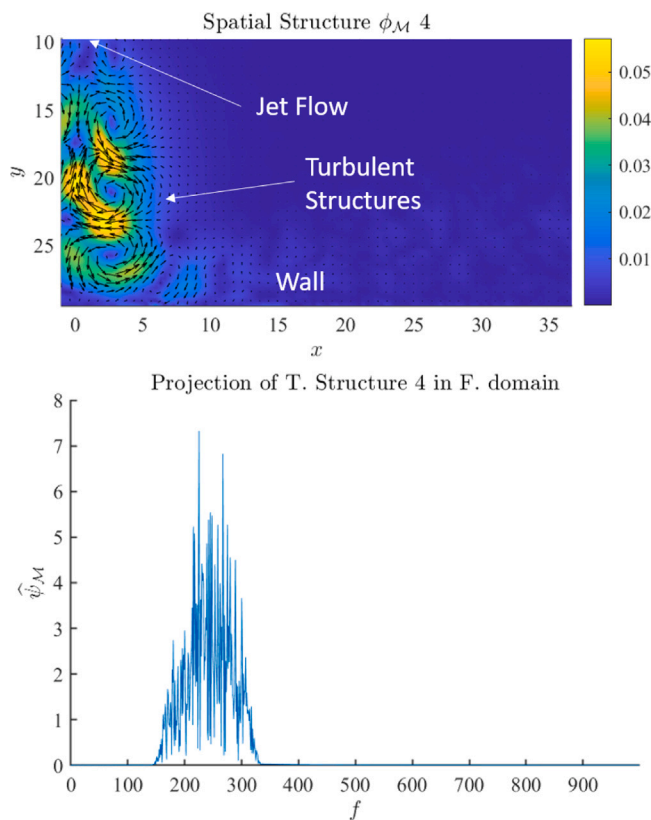
Fig. 6. Spatial structure and frequency content of the fourth mPOD mode in the TR-PIV velocity field of exercise four, consisting of a planar gas jet flow impinging on a flat wall.

comparison between data-driven and analytical decompositions. Moreover, being the flow sustained by two known source terms, the exercise allows for exploring the convergence and the time–frequency analysis capabilities of all the methods. The second and third exercises present the analysis of 2D scalar datasets. These were described in [17]. Both are useful to analyze the problem of uniqueness of the POD, which occurs when modes have similar energy content. The second exercise consists of a simple super-position of known modes while the third features the numerical solution of the nonlinear advection–diffusion of prescribed source terms. The fourth exercise, also presented in [17], presents the decomposition of the experimental data, which is velocity fields obtained via Time-Resolved Particle Image Velocimetry (TR-PIV). This allows for practice with modern experimental data. Fig. 6 shows an exemplary mPOD mode obtained in MODULO for this test case, which consists of a planar gas jet impinging on a flat wall. The top figure shows the spatial structure of the mode; the bottom one shows the frequency content of the associated temporal structure. This mode captures the turbulent structures evolving from the shear layer instability in the jet.

As described in the video tutorials three and seven, neither the POD nor the DFT can clearly isolate these structures. The POD is limited by the constraint of optimal convergence, which forces the decomposition to put multiple features in the same modes. The second is limited by the constraint of harmonic temporal structures, which does not let the DFT mode capture coherent patterns composed of multiple frequencies. Finally, the fifth exercise considers the velocity field obtained via TR-PIV of the flow past a cylinder in transient conditions. The dataset is described

in [19]. This test case consists of a much larger number of snap-shots, which forces most laptop computers to use the memory saving feature of MODULO.

4. Impact and conclusions

We have presented the functionalities of the open source software package MODULO, starting from the theoretical background. The software allows for performing classical modal decomposition such as POD and DFT as well as the novel mPOD. Moreover, thanks to its memory saving feature, MODULO is well suited to analyze relatively large datasets while keeping moderate memory requirements. While these decompositions are nowadays essential tools in fluid mechanics, their general framework is certainly of great interest to any applied scientist. Finally, the complete set of exercises available can also serve didactic purposes and encourage the novice to enter this important discipline.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

References

[1] Strang G. Linear algebra and learning from data. Wellesley-Cambridge Press, U.S.; 2019.
[2] Brunton SL, Kutz JN. Data-driven science and engineering. Cambridge University Press; 2019, http://dx.doi.org/10.1017/9781108380690.
[3] Bishop CM. Pattern recognition and machine learning. Springer-Verlag New York Inc.; 2006.
[4] Bouwmans T, Sobral A, Javed S, Jung SK, Zahzah E-H. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. Comp Sci Rev 2017;23:1–71. http://dx.doi.org/10.1016/j.cosrev.2016.11.001.
[5] Lu J, Kerns RT, Peddada SD, Bushel PR. Principal component analysis-based filtering improves detection for Affymetrix gene expression arrays. Nucleic Acids Res 2011;39(13):e86–e86. http://dx.doi.org/10.1093/nar/gkr241.
[6] Davydenko AY, Grayver AV. Principal component analysis for filtering and leveling of geophysical data. J Appl Geophys 2014;109:266–80. http://dx.doi.org/10.1016/j.jappgeo.2014.08.006.
[7] Alpaydin E. Introduction to machine learning. The MIT Press; 2014.
[8] Keiper W, Milde A, Volkwein S, editors. Reduced-order modeling (ROM) for simulation and optimization. Springer International Publishing; 2018, http://dx.doi.org/10.1007/978-3-319-75319-5.
[9] Quarteroni A, Manzoni A, Negri F. Reduced basis methods for partial differential equations. Springer International Publishing; 2016, http://dx.doi.org/10.1007/978-3-319-15431-2.
[10] Reynders E. System identification methods for (operational) modal analysis: Review and comparison. Arch Comput Methods Eng 2012;19(1):51–124. http://dx.doi.org/10.1007/s11831-012-9069-x.
[11] Noack BR, Morzyński M, Tadmor G, editors. Reduced-order modelling for flow control. Springer Vienna; 2011, http://dx.doi.org/10.1007/978-3-7091-0758-4.
[12] Holmes P, Lumley JL, Berkooz G. Turbulence, coherent structures, dynamical systems and symmetry. Cambridge University Press; 1996, http://dx.doi.org/10.1017/cbo9780511622700.
[13] Jackson JE. A user's guide to principal components. Wiley-Interscience; 2003.
[14] Wang R. Introduction to orthogonal transforms: With applications in data processing. Cambridge University Press (1888); 2012.
[15] Tu JH, Rowley CW, Luchtenburg DM, Brunton SL, Kutz JN. On dynamic mode decomposition: Theory and applications. J Comput Dyn 2014;1(2):391–421. http://dx.doi.org/10.3934/jcd.2014.1.391.
[16] Mendez M, Balabane M, Buchlin J-M. Multi-scale proper orthogonal decomposition of complex fluid flows. J Fluid Mech 2019;870:988–1036, Preprint available at https://arxiv.org/abs/1804.09646. http://dx.doi.org/10.1017/jfm.2019.212.

[17] Mendez M, Scelzo M, Buchlin J-M. Multiscale modal analysis of an oscillating impinging gas jet. Exp Therm Fluid Sci 2018;91:256–76. http://dx.doi.org/10.1016/j.expthermflusci.2017.10.032.

[18] Mendez M, Gosset A, Buchlin J-M. Experimental analysis of the stability of the jet wiping process, part II: Multiscale modal analysis of the gas jet-liquid film interaction. Exp Therm Fluid Sci 2019;106:48–67. http://dx.doi.org/10.1016/j.expthermflusci.2019.03.004.

[19] Mendez MA, Hess D, Watz BB, Buchlin J-M. Multiscale proper orthogonal decomposition (mPOD) of TR-PIV data—a case study on stationary and transient cylinder wake flows. Meas Sci Technol 2020;31(9):094014. http://dx.doi.org/10.1088/1361-6501/ab82be.

[20] Esposito C, Mendez M, Steelant J, Vetrano M. Spectral and modal analysis of a cavitating flow through an orifice. Exp Therm Fluid Sci 2021;121:110251. http://dx.doi.org/10.1016/j.expthermflusci.2020.110251.

[21] Amor C, Pérez JM, Schlatter P, Vinuesa R, Clainche SL. Soft computing techniques to analyze the turbulent wake of a wall-mounted square cylinder. In: Advances in intelligent systems and computing. Springer International Publishing; 2019, p. 577–86. http://dx.doi.org/10.1007/978-3-030-20055-8_55.

[22] Sirovich L. Analysis of turbulent flows by means of the empirical eigenfunctions. Fluid Dyn Res 1991;8:85–100. http://dx.doi.org/10.1016/0169-5983(91)90033-F.