Computer Physics Communications 237 (2019) 98–109

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

An efficient streaming algorithm for spectral proper orthogonal decomposition

Oliver T. Schmidt [a,*], Aaron Towne [b]

[a] University of California San Diego, La Jolla, CA 92093, USA
[b] University of Michigan, Ann Arbor, MI 48109, USA

ARTICLE INFO

Article history:
Received 17 November 2017
Received in revised form 12 November 2018
Accepted 14 November 2018
Available online 23 November 2018

Keywords:
Proper orthogonal decomposition
Principal component analysis
Spectral analysis

ABSTRACT

A streaming algorithm to compute the spectral proper orthogonal decomposition (SPOD) of stationary random processes is presented. As new data becomes available, an incremental update of the truncated eigenbasis of the estimated cross-spectral density (CSD) matrix is performed. The algorithm requires access to only one temporal snapshot of the data at a time and converges orthogonal sets of SPOD modes at discrete frequencies that are optimally ranked in terms of energy. We define measures of error and convergence, and demonstrate the algorithm's performance on two datasets. The first example considers a high-fidelity numerical simulation of a turbulent jet, and the second example uses optical flow data obtained from high-speed camera recordings of a stepped spillway experiment. For both cases, the most energetic SPOD modes are reliably converged. The algorithm's low memory requirement enables real-time deployment and allows for the convergence of second-order statistics from arbitrarily long streams of data. A MATLAB implementation of the algorithm along with a test database for the jet example, can be found in the Supplementary material.

1. Introduction

The ability to represent complex dynamics by a small number of dynamically important modes enables the analysis, modeling and control of high-dimensional systems. Turbulent flows are a prominent example of such systems [1,2]. Proper orthogonal decomposition (POD), also known as principle component analysis (PCA) or Karhunen–Loève decomposition, is a popular modal decomposition technique to extract coherent structures from experimental and numerical data. In its most common form [3], POD is conducted in the time domain. It is computed from a time series of snapshots and expands the flow field into a sum of products of spatially orthogonal modes and coefficients with random time dependence. POD modes are optimally ranked in terms of their variance, or energy. These properties make POD modes well suited for low-order models based on Galerkin projection of the Navier–Stokes equations [4,5].

Besides its definitions in the temporal and spatial domains, POD can also be formulated in the frequency domain. This variant of POD called *spectral proper orthogonal decomposition* (SPOD), dates back to the early work of Lumley [6] and takes advantage of temporal homogeneity. This makes it ideally suited for statistically (wide-sense) stationary data [7]. SPOD provides orthogonal modes at discrete frequencies that are optimally ranked in terms of energy, and that evolve coherently in both space and time. Perhaps predictably, this optimal space–time representation of the data comes at a cost—very long time series are necessary in order to converge the second-order space–time statistics. This data demand also becomes apparent when comparing SPOD to other modal decomposition techniques, for example the popular dynamic mode decomposition (DMD) [8]. For stationary data, SPOD modes correspond to optimally averaged DMD modes computed from an ensemble of stochastic realizations of a process [7], for example multiple repetitions of the same experiment. In Section 5, we use two databases consisting of 10,000 and 19,782 snapshots, respectively. Evidently, the SPOD problem quickly becomes computationally unmanageable for data with large spatial dimensions.

In this paper, we address this issue by proposing a low-storage *streaming SPOD* algorithm that incrementally updates the SPOD as new data becomes available. Similar algorithms are often referred to as *incremental*, *learning*, *updating*, *on-the-fly* or *online* algorithms in the literature. Streaming algorithms for DMD have been developed recently [9,10], for example. The proposed streaming SPOD algorithm utilizes incremental updates of the singular value decomposition (SVD) of the cross-spectral density (CSD) matrix of the data. SVD updating has been an active research topic for almost half a century, see e.g. [11,12]. In this work, we build on Brand's [13] incremental singular value decomposition (SVD) by specializing the method to updates of the estimated CSD matrix. Originally developed for computer vision and audio feature extraction [14],

* Corresponding author.
E-mail address: oschmidt@ucsd.edu (O.T. Schmidt).

https://doi.org/10.1016/j.cpc.2018.11.009
0010-4655/

the algorithm has been employed for recommender systems [15], semantics [16], design optimization [17], and a wide spectrum of other machine learning and data mining applications.

The paper is organized as follows. We first introduce standard or *batch* SPOD in Section 2 before deriving the streaming algorithm in Section 3. Measures of error and convergence are defined in Section 4. In Section 5, we demonstrate streaming SPOD on two datasets: a high-fidelity large eddy simulation (LES) of a turbulent jet and experimental optical flow from high-speed camera data of a stepped spillway. The effect of eigenbasis truncation is addressed in Section 6. In Section 7, we conclude with a discussion of the algorithm's computational efficiency and utility in real-time and big data settings.

2. Batch SPOD

SPOD is the frequency-domain counterpart of standard time-domain or spatial POD. SPOD yields time-harmonic modes that represent structures that evolve coherently in both time and space [7]. The method is based on an eigendecomposition of the CSD, which in turn is estimated from an ensemble of realizations of the temporal discrete Fourier transform (DFT) in practice. The CSD can be estimated using standard spectral estimation techniques such as Welch's method [see e.g. 18] from an ensemble of snapshots. The SPOD formalism is derived from a space–time POD problem under the assumption of wide-sense stationarity. The reader is referred to [7] for the derivation of the method and an assessment of its properties. In particular, the method's relations to DMD and the resolvent operator are interesting from a modeling perspective, as they link SPOD to concepts from dynamical systems and hydrodynamics stability theory.

Fig. 1 serves as a visual guide through the batch algorithm. We start with an ensemble of n_t snapshots $\mathbf{q}_i = \mathbf{q}(t_i) \in \mathbb{R}^n$ of a wide-sense stationary process $\mathbf{q}(t)$ sampled at discrete times $t_1, t_2, \ldots, t_{n_t} \in \mathbb{R}$. By $\mathbf{q}$ we denote the state vector. Its total length n is equal to the number of grid points n_x times the number of variables n_{var}. The temporal mean corresponds to the ensemble average defined as

$$\overline{\mathbf{q}} = \frac{1}{n_t}\sum_{i=1}^{n_t} \mathbf{q}_i \in \mathbb{R}^n. \tag{1}$$

We collect the mean-subtracted snapshots in a data matrix

$$\mathbf{Q} = [\mathbf{q}_1 - \overline{\mathbf{q}}, \mathbf{q}_2 - \overline{\mathbf{q}}, \ldots, \mathbf{q}_{n_t} - \overline{\mathbf{q}}] \in \mathbb{R}^{n\times n_t} \tag{2}$$

of rank $d \leq \min\{n, n_t - 1\}$. With the goal of estimating the CSD, we apply Welch's method to the data by segmenting $\mathbf{Q}$ into n_{blk} overlapping blocks

$$\mathbf{Q}^{(l)} = [\mathbf{q}_1^{(l)} - \overline{\mathbf{q}}, \mathbf{q}_2^{(l)} - \overline{\mathbf{q}}, \ldots, \mathbf{q}_{n_{\text{freq}}}^{(l)} - \overline{\mathbf{q}}] \in \mathbb{R}^{n\times n_{\text{freq}}} \tag{3}$$

containing n_{freq} snapshots each. If n_{ovlp} is the number of snapshots by which the blocks overlap, then the jth column of the lth block $\mathbf{Q}^{(l)}$ is given as

$$\mathbf{q}_j^{(l)} = \mathbf{q}_{j+(l-1)(n_{\text{freq}}-n_{\text{ovlp}})} - \overline{\mathbf{q}}. \tag{4}$$

We assume that each block can be regarded as a statistically independent realization under the ergodicity hypothesis. The purpose of the segmentation step is to artificially increase the number of ensemble members, i.e. Fourier realizations. This method is useful in the common scenario where a single long dataset with equally sampled snapshots is available, for example from a numerical simulation. In situations where the data presents itself in form of independent realizations from the beginning, segmenting need not be applied. This is the case, for example, if an experiment is repeated multiple times. Next, the temporal (row-wise) discrete Fourier transform

$$\hat{\mathbf{Q}}^{(l)} = [\hat{\mathbf{q}}_1^{(l)}, \hat{\mathbf{q}}_2^{(l)}, \ldots, \hat{\mathbf{q}}_{n_{\text{freq}}}^{(l)}] \in \mathbb{R}^{n\times n_{\text{freq}}} \tag{5}$$

of each block is calculated. A windowing function can be used to mitigate spectral leakage. All realizations of the Fourier transform at the kth frequency are subsequently collected into a new data matrix

$$\hat{\mathbf{Q}}_k = [\hat{\mathbf{q}}_k^{(1)}, \hat{\mathbf{q}}_k^{(2)}, \ldots, \hat{\mathbf{q}}_k^{(n_{\text{blk}})}] \in \mathbb{R}^{n\times n_{\text{blk}}}. \tag{6}$$

At this point, we introduce the weighted data matrix

$$\mathbf{X}_k = \frac{1}{\sqrt{n_{\text{blk}}}}\mathbf{W}^{\frac{1}{2}}\hat{\mathbf{Q}}_k = [\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(n_{\text{blk}})}] \in \mathbb{R}^{n\times n_{\text{blk}}}, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a positive-definite Hermitian matrix that accounts for quadrature and possibly other weights associated with the discretized inner product

$$\langle \mathbf{a}, \mathbf{b}\rangle_E = \mathbf{a}^*\mathbf{W}\mathbf{b}. \tag{8}$$

The inner product (8) induces the spatial energy norm $\|\cdot\|_E = \sqrt{\langle\cdot,\cdot\rangle_E}$ by which we wish to rank the SPOD modes. The product

$$\mathbf{S}_k = \mathbf{X}_k\mathbf{X}_k^* \in \mathbb{R}^{n\times n} \tag{9}$$

defines the weighted CSD matrix of the kth frequency. A factor of $\frac{1}{n_{\text{blk}}}$ seen in other definitions of the CSD is absorbed into our definition of the weighted data matrix in Eq. (7).

SPOD is based on the eigenvalue decomposition

$$\mathbf{S}_k = \mathbf{U}_k\boldsymbol{\Lambda}_k\mathbf{U}_k^* \tag{10}$$

of the CSD matrix, where $\boldsymbol{\Lambda}_k = \text{diag}(\lambda_{k_1}, \lambda_{k_2} \ldots, \lambda_{k_{n_{\text{blk}}}}) \in \mathbb{R}^{n_{\text{blk}}\times n_{\text{blk}}}$ is the matrix of ranked (in descending order) eigenvalues and $\mathbf{U}_k = [\mathbf{u}_{k_1}, \mathbf{u}_{k_2}, \ldots, \mathbf{u}_{k_{n_{\text{blk}}}}] \in \mathbb{R}^{n\times n_{\text{blk}}}$ the corresponding matrix of eigenvectors. Equivalently, we may consider the economy SVD of the weighted data matrix

$$\mathbf{X}_k = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^*, \tag{11}$$

where $\boldsymbol{\Sigma}_k = \text{diag}(\sigma_{k_1}, \sigma_{k_2} \ldots, \sigma_{k_{n_{\text{blk}}}}) \in \mathbb{R}^{n_{\text{blk}}\times n_{\text{blk}}}$ is the matrix of singular values and $\mathbf{V}_k = [\mathbf{v}_{k_1}, \mathbf{v}_{k_2}, \ldots, \mathbf{v}_{k_{n_{\text{blk}}}}] \in \mathbb{R}^{n\times n_{\text{blk}}}$ the right singular vector matrix. This can be shown by rewriting the CSD in terms of the SVD of the data matrix as $\mathbf{S}_k = \mathbf{X}_k\mathbf{X}_k^* = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^*\mathbf{V}_k\boldsymbol{\Sigma}_k\mathbf{U}_k^* = \mathbf{U}_k\boldsymbol{\Lambda}_k\mathbf{U}_k^*$. Throughout this paper, we assume that all Fourier realizations of the flow are linearly independent. In the final step, the SPOD modes ϕ and modal energies σ^2 are found as

$$\boldsymbol{\Phi}_k = \mathbf{W}^{-\frac{1}{2}}\mathbf{U}_k = [\boldsymbol{\phi}_{k_1}, \boldsymbol{\phi}_{k_2}, \ldots, \boldsymbol{\phi}_{k_{n_{\text{blk}}}}] \in \mathbb{R}^{n\times n_{\text{blk}}} \tag{12}$$

$$\text{and } \boldsymbol{\Sigma}_k = \text{diag}(\sigma_{k_1}, \sigma_{k_2} \ldots, \sigma_{k_{n_{\text{blk}}}}) \in \mathbb{R}^{n_{\text{blk}}\times n_{\text{blk}}}, \tag{13}$$

respectively. The weighting of the eigenvectors in Eq. (12) guarantees orthonormality

$$\boldsymbol{\Phi}_k^*\mathbf{W}\boldsymbol{\Phi}_k = \mathbf{I} \tag{14}$$

under the inner product (8).

3. Streaming SPOD

Two aspects of the batch SPOD algorithm make it computationally demanding for large datasets. First, n_{freq} snapshots must be loaded into memory and operated upon simultaneously in order to compute the required Fourier modes. Second, n_{blk} realizations of the Fourier mode at a given frequency of interest must be loaded into memory and operated upon simultaneously to compute the singular value decomposition that produces the SPOD modes. In the following subsections, we develop strategies to overcome these two challenges, leading to a streaming algorithm that requires access to only the most recent data snapshot and recursively updates the d most energetic SPOD modes for each frequency of interest. A graphical illustration of the streaming algorithm is shown in Fig. 2.

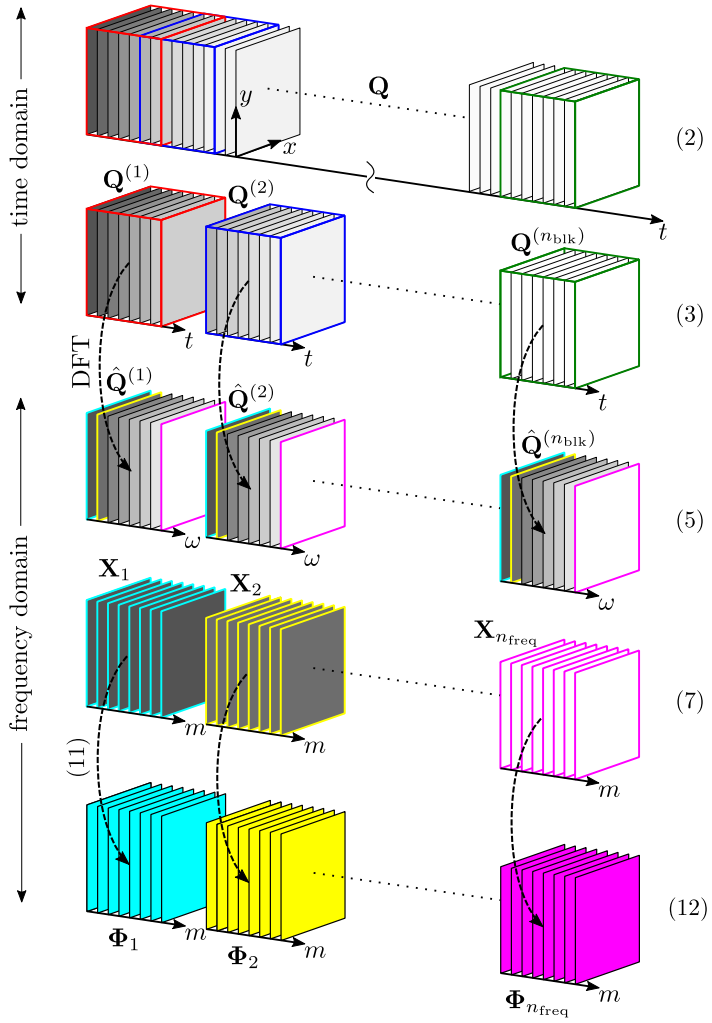

Fig. 1. Illustration of the batch SPOD algorithm. Each rectangular slice represents a snapshot and the numbers in parentheses denote the equations in the text. The data is first segmented, then Fourier transformed, then reordered by frequency, and finally diagonalized into SPOD modes.

3.1. Streaming Fourier sums

Ideally, a streaming SPOD algorithm would require access to only one snapshot of the data at a time, e.g., the solution computed in a simulation or measured in an experiment at the most recent time instant. The batch SPOD algorithm does not have this property because the discrete Fourier modes in Eq. (5) are typically computed using the Fast Fourier Transform (FFT) algorithm, which requires simultaneous access to n_{freg} snapshots. This requirement can be relaxed by computing the Fourier modes using the original definition of the discrete Fourier transform rather than the FFT algorithm.

Consider the definition of the discrete Fourier transform,

$$\hat{\mathbf{q}}_k^{(l)} = \sum_{j=1}^{n_{\text{freq}}} \mathbf{q}_j^{(l)} f_{jk}, \tag{15}$$

where

$$f_{jk} = z^{(k-1)(j-1)} \tag{16}$$

and $z = \exp(-\mathrm{i}2\pi/n_{\text{freq}})$ is the primitive n_{freq}-th root of unity. Eq. (15) shows how each snapshot $\mathbf{q}_j^{(l)}$ contributes to each Fourier mode $\hat{\mathbf{q}}_k^{(l)}$ – specifically, the snapshot at time j is multiplied by the complex scalars f_{jk} and then added to the contributions of other time instances to obtain each Fourier mode.

This observation provides a way to compute the Fourier modes that requires access to only the most recent data snapshot. A new snapshot $\mathbf{q}_p$ will appear in block l if $1 < p-(l-1)(n_{\text{freq}}-n_{\text{ovlp}}) < n_{\text{freq}}$, in which case $\mathbf{q}_j^{(l)}$ is defined by Eq. (4) with $j = p-(l-1)(n_{\text{freq}}-n_{\text{ovlp}})$. The snapshot $\mathbf{q}_p$ can appear in multiple blocks if the overlap n_{ovlp} is nonzero. Next, each $\mathbf{q}_j^{(l)}$ is multiplied by the corresponding f_{jk} values for each $k = 1, \ldots, n_{\text{freq}}$ and added to previous terms to give the partial sum

$$\left[\hat{\mathbf{q}}_k^{(l)}\right]_{n_p} = \left[\hat{\mathbf{q}}_k^{(l)}\right]_{n_p-1} + \mathbf{q}_{n_p}^{(l)} f_{n_p k} = \sum_{j=1}^{n_p} \mathbf{q}_j^{(l)} f_{jk}. \tag{17}$$

Once $n_p = n_{\text{freq}}$ for block l, the Fourier mode is recovered as

$$\hat{\mathbf{q}}_k^{(l)} = \left[\hat{\mathbf{q}}_k^{(l)}\right]_{n_{\text{freq}}}. \tag{18}$$

This procedure has several desirable properties. First, by construction, it requires access to only the most recent data snapshot. This immediately reduces the memory required to compute the Fourier modes by a factor of approximately n_{freq} compared to a standard FFT algorithm. Second, no approximations have been made, so the computed Fourier modes are exact. Third, additional computational and memory saving may be obtained by computing the partial sums in Eq. (17) only for frequencies of interest. Often, the value of n_{freq} required to control spectral leakage and aliasing is

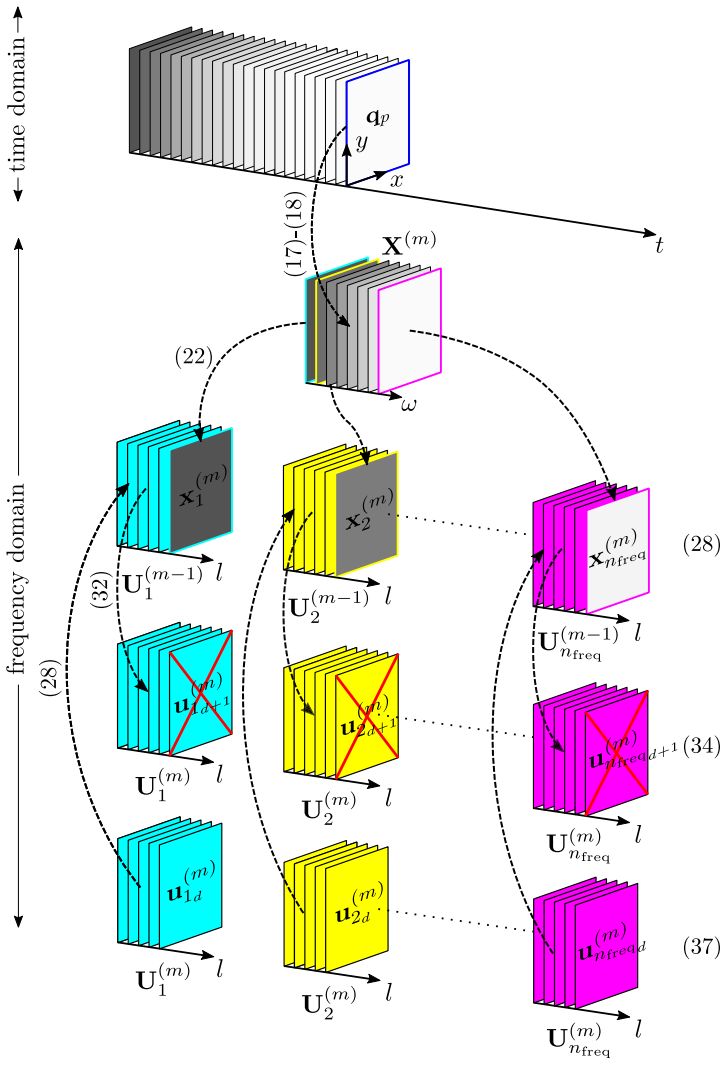

Fig. 2. Illustration of the streaming SPOD algorithm. Numbers in parentheses denote the equations. As soon as a new data snapshot becomes available, the partial Fourier sums are updated. Once the Fourier sums are completed, the old eigenbases for each frequency are augmented by the orthogonal complement from the new data. The basis rotation and truncation conclude the update.

larger than the number of frequencies actually needed for analysis. Standard FFT algorithms automatically compute every frequency, $k = 1, \ldots, n_{\mathrm{freq}}$, whereas it is straightforward to compute only the frequencies of interest using the streaming algorithm by including only those values of k.

The main drawback of the method is that computing all n_{freq} frequencies requires order ${n_{\mathrm{freq}}}^2$ operations, compared to $n_{\mathrm{freq}} \log n_{\mathrm{freq}}$ for an FFT algorithm. However, memory requirements, not operation counts, are the primary obstacle for applying SPOD to large datasets. Moreover, the increased operation count can be partially negated by computing only frequencies of interest, as described above.

3.2. Incremental updates of the CSD

The second aspect of batch SPOD that hinders its application to large datasets is the need to store many realizations of each Fourier mode in memory to compute the modes. To overcome this obstacle, we develop an algorithm that recursively updates the d most energetic SPOD modes for each frequency as new Fourier modes become available from the streaming Fourier algorithm. We require the updating algorithm to converge a fixed number of modes d to be able to operate within a strictly limited amount of memory. We start by adapting Brand's [13] incremental SVD algorithm to the special case of updating the eigendecomposition of the estimated CSD matrix. The best rank-d approximation used to truncate the eigenbasis and the initialization of the algorithm are discussed later in Sections 3.3 and 3.4, respectively.

The block-wise sample mean is readily updated through the recursive relation

$$\overline{\mathbf{q}}^{(m)} = \frac{m-1}{m}\overline{\mathbf{q}}^{(m-1)} + \frac{1}{m}\left[\frac{1}{n_{\mathrm{freq}}}\sum_{j=1}^{n_{\mathrm{freq}}}\mathbf{q}_j^{(m)}\right]. \tag{19}$$

Analogously, a rank-1 update of the CSD takes the form

$$\mathbf{S}_k^{(m)} = \frac{m-1}{m}\mathbf{S}_k^{(m-1)} + \frac{m-1}{m^2}\mathbf{x}_k^{(m)}\mathbf{x}_k^{*(m)} \tag{20}$$

and can be performed once the mth Fourier realization $\hat{\mathbf{q}}_k^{(m)}$ becomes available. Note that we use the sample CSD as an unbiased estimator for the unknown population CSD. The update formula for the CSD, Eq. (20), can be rewritten in terms of the data matrix $\mathbf{X}_k$ as

$$\mathbf{X}_k^{(m)}\mathbf{X}_k^{*(m)} = \frac{m-1}{m}\mathbf{X}_k^{(m-1)}\mathbf{X}_k^{*(m-1)} + \frac{m-1}{m^2}\mathbf{x}_k^{(m)}\mathbf{x}_k^{*(m)} \tag{21}$$

by using definition (9). Analogous to Eq. (7), we denote by

$$\mathbf{X}_k^{(m)} = [\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(m)}] = \frac{1}{\sqrt{m}} \mathbf{W}^{\frac{1}{2}} \hat{\mathbf{Q}}_k^{(m)} \in \mathbb{R}^{n \times m} \tag{22}$$

the data matrix containing the first m weighted Fourier realizations at the kth frequency. We now insert the SVD of the data matrix

$$\mathbf{X}_k^{(m)} = \mathbf{U}_k^{(m)} \boldsymbol{\Sigma}_k^{(m)} \mathbf{V}_k^{*(m)} \tag{23}$$

into the update formula (21) to obtain an updating scheme

$$\mathbf{U}_k^{(m)} \boldsymbol{\Sigma}_k^{2(m)} \mathbf{U}_k^{*(m)} = \frac{m-1}{m} \mathbf{U}_k^{(m-1)} \boldsymbol{\Sigma}_k^{2(m-1)} \mathbf{U}_k^{*(m-1)} + \frac{m-1}{m^2} \mathbf{x}_k^{(m)} \mathbf{x}_k^{*(m)} \tag{24}$$

for the eigendecomposition of the CSD at iteration level m in terms of the eigendecomposition at level $m-1$ and the newly arrived data $\mathbf{x}_k^{(m)}$. For brevity, we factorize Eq. (24) and consider the data matrix

$$\mathbf{X}_k^{(m)} = \left[\sqrt{\frac{m-1}{m}} \mathbf{U}_k^{(m-1)} \boldsymbol{\Sigma}_k^{(m-1)} \mathbf{V}_k^{*(m-1)} \quad \sqrt{\frac{m-1}{m^2}} \mathbf{x}_k^{(m)} \right] \tag{25}$$

$$= \begin{bmatrix} \mathbf{U}_k^{(m-1)} & \mathbf{x}_k^{(m)} \end{bmatrix} \begin{bmatrix} \sqrt{\frac{m-1}{m}} \boldsymbol{\Sigma}_k^{(m-1)} \mathbf{V}_k^{*(m-1)} & \mathbf{0} \\ \mathbf{0} & \sqrt{\frac{m-1}{m^2}} \end{bmatrix} \tag{26}$$

instead of the product $\mathbf{X}_k^{(m)} \mathbf{X}_k^{*(m)}$. With the goal in mind to update $\mathbf{U}_k^{(m-1)}$ with the new data $\mathbf{x}_k^{(m)}$, Eq. (25) is factored into the matrix product given by Eq. (26). We seek to find the updated set of left singular vectors $\mathbf{U}_k^{(m)}$ in the column space of the augmented eigenbasis $\begin{bmatrix} \mathbf{U}_k^{(m-1)} & \mathbf{x}_k^{(m)} \end{bmatrix}$ and start by restoring orthonormality. The component of $\mathbf{x}_k^{(m)}$ that is orthogonal to $\mathbf{U}_k^{(m-1)}$ can readily be found from a partial step of the modified Gram–Schmidt (MGS) algorithm as

$$\mathbf{u}_k^{\perp(m)} = \mathbf{x}_k^{(m)} - \mathbf{U}_k^{(m-1)} \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)}. \tag{27}$$

Using Eq. (27), the multiplicand is recast into a product of a modified multiplicand $\begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix}$ with orthonormal columns and a matrix as

$$\begin{bmatrix} \mathbf{U}_k^{(m-1)} & \mathbf{x}_k^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)} \\ \mathbf{0} & \|\mathbf{u}_k^{\perp(m)}\| \end{bmatrix}. \tag{28}$$

Inserting Eq. (28) into Eq. (26) yields the expression

$$\mathbf{X}_k^{(m)} = \begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix} \times \begin{bmatrix} \mathbf{I} & \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)} \\ \mathbf{0} & \|\mathbf{u}_k^{\perp(m)}\| \end{bmatrix} \begin{bmatrix} \sqrt{\frac{m-1}{m}} \boldsymbol{\Sigma}_k^{(m-1)} \mathbf{V}_k^{(m-1)} & \mathbf{0} \\ \mathbf{0} & \sqrt{\frac{m-1}{m^2}} \end{bmatrix} \tag{29}$$

for the updated data matrix. Multiplying Eq. (29) with its conjugate transpose yields the updated CSD

$$\mathbf{X}_k^{(m)} \mathbf{X}_k^{*(m)} = \begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{U}_k^{*(m-1)} \\ \frac{\mathbf{u}_k^{\perp *(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix}, \tag{30}$$

where

$$\mathbf{M} = \frac{m-1}{m^2} \begin{bmatrix} m \boldsymbol{\Sigma}_k^{(m-1)^2} + \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)} \mathbf{x}_k^{*(m)} \mathbf{U}_k^{(m-1)} & \|\mathbf{u}_k^{\perp(m)}\| \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)} \\ \|\mathbf{u}_k^{\perp(m)}\| \mathbf{x}_k^{*(m)} \mathbf{U}_k^{(m-1)} & \|\mathbf{u}_k^{\perp(m)}\|^2 \end{bmatrix} \tag{31}$$

is a $m \times m$ Hermitian matrix. The remaining task is to recast the right-hand side of Eq. (30) into SVD form. This is achieved through an eigendecomposition $\mathbf{M} = \tilde{\mathbf{U}} \tilde{\boldsymbol{\Sigma}}^2 \tilde{\mathbf{U}}^*$ of $\mathbf{M}$. Equivalently, we may factor $\mathbf{M}$ as $\mathbf{M} = \mathbf{K}\mathbf{K}^*$ first, where

$$\mathbf{K} = \sqrt{\frac{m-1}{m^2}} \begin{bmatrix} \sqrt{m} \boldsymbol{\Sigma}_k^{(m-1)} & \mathbf{U}_k^{*(m-1)} \mathbf{x}_k^{(m)} \\ \mathbf{0} & \|\mathbf{u}_k^{\perp(m)}\| \end{bmatrix} = \tilde{\mathbf{U}} \tilde{\boldsymbol{\Sigma}} \tilde{\mathbf{V}}^*, \tag{32}$$

and compute the SVD of $\mathbf{K}$. Inserting the eigendecomposition $\mathbf{M} = \mathbf{K}\mathbf{K}^* = \tilde{\mathbf{U}} \tilde{\boldsymbol{\Sigma}}^2 \tilde{\mathbf{U}}^*$ into Eq. (30) yields

$$\mathbf{X}_k^{(m)} \mathbf{X}_k^{*(m)} = \underbrace{\begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix} \tilde{\mathbf{U}}}_{\mathbf{U}_k^{(m)}} \underbrace{\tilde{\boldsymbol{\Sigma}}}_{\boldsymbol{\Sigma}_k^{(m)}} \tilde{\boldsymbol{\Sigma}}^* \tilde{\mathbf{U}}^* \begin{bmatrix} \mathbf{U}_k^{*(m-1)} \\ \frac{\mathbf{u}_k^{\perp *(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix}$$

$$= \mathbf{U}_k^{(m)} \boldsymbol{\Sigma}_k^{(m)^2} \mathbf{U}_k^{*(m)}. \tag{33}$$

By noting that $\tilde{\mathbf{U}}$ is a rotation matrix that preserves orthonormality, the spectral theorem guarantees that this decomposition is unique, and therefore corresponds to the updated eigendecomposition of the CSD matrix. The updates of the eigenbasis and eigenvalues hence take the form

$$\mathbf{U}_k^{(m)} = \begin{bmatrix} \mathbf{U}_k^{(m-1)} & \frac{\mathbf{u}_k^{\perp(m)}}{\|\mathbf{u}_k^{\perp(m)}\|} \end{bmatrix} \tilde{\mathbf{U}} \quad \text{and} \tag{34}$$

$$\boldsymbol{\Sigma}_k^{(m)} = \tilde{\boldsymbol{\Sigma}}, \tag{35}$$

respectively. Besides the rotation (34), the implementation of the algorithm requires the MGS step (27) and the construction and SVD of $\mathbf{K}$, as defined in Eq. (32). Note that the large matrices $\mathbf{X}_k^{(m)} \mathbf{X}_k^{*(m)}$ and $\mathbf{U}_k^{(m-1)} \mathbf{U}_k^{*(m-1)}$ appearing in the derivation are never computed in the actual algorithm. Up to this point, no approximations have been made.

3.3. Eigenbasis truncation

The recursive rank-1 updates described by Eq. (34) add an additional vector to the eigenbasis of the CSD matrix at each step. In practice, however, we are interested in converging a fixed number d of the most energetic SPOD modes only. Fortunately, the basic property of the SVD guarantees that this best rank-d approximation is readily obtained by truncating the basis after the dth vector. Formally, we express this by partitioning the updated eigenbasis and matrix of singular values as

$$\mathbf{U}_k^{(m)} = \begin{bmatrix} \mathbf{U}_k'^{(m)} & \mathbf{u}_{k_{d+1}} \end{bmatrix} \text{ and } \boldsymbol{\Sigma}_k^{(m)} = \begin{bmatrix} \boldsymbol{\Sigma}_k'^{(m)} & \mathbf{0} \\ \mathbf{0} & \sigma_{k_{d+1}} \end{bmatrix}, \tag{36}$$

respectively, and letting

$$\mathbf{U}_k^{(m)} \leftarrow \mathbf{U}_k'^{(m)} \text{ and } \boldsymbol{\Sigma}_k^{(m)} \leftarrow \boldsymbol{\Sigma}'^{(m)} \tag{37}$$

as we update the basis during runtime. At this point, a truncation error is introduced as the vector component $\mathbf{u}_{k_{d+1}}$ that is orthogonal to the retained d eigenvectors is discarded. The batch SPOD algorithm, on the contrary, guarantees that every eigenvector is orthogonal to all other $n_{\text{blk}} - 1$ eigenvectors. We address the error resulting from the basis truncation in Section 4.

As before, the final step of the algorithm consists of obtaining the SPOD modes by weighting the CSD eigenvectors according to $\boldsymbol{\Phi}_k^{(m)} = \mathbf{W}^{-\frac{1}{2}} \mathbf{U}_k^{(m)} = [\boldsymbol{\phi}_{k_1}^{(m)}, \boldsymbol{\phi}_{k_2}^{(m)}, \ldots, \boldsymbol{\phi}_{k_d}^{(m)}] \in \mathbb{R}^{n \times d}$.

3.4. Initialization

Once the first Fourier realization becomes available, the eigenbasis is initialized as $\mathbf{U}_k^{(1)} \leftarrow [\mathbf{x}_k^{(1)}, \mathbf{0}, \ldots, \mathbf{0}] \in \mathbb{R}^{n \times d}$ and subsequently updated as $\mathbf{U}_k^{(2)} = [\mathbf{u}_{k_1}^{(2)}, \mathbf{u}_{k_2}^{(2)}, \mathbf{0}, \ldots, \mathbf{0}]$ at iteration level $m = 2$, and so on. Correspondingly, the singular value

matrix is initialized with the first Fourier realization as $\boldsymbol{\Sigma}_k^{(1)} \leftarrow \mathrm{diag}(\mathbf{x}_k^{*(1)}\mathbf{x}_k^{(1)}, 0, \ldots, 0) \in \mathbb{R}^{d\times d}$ before being updated as $\boldsymbol{\Sigma}_k^{(2)} = \mathrm{diag}(\sigma_{k_1}^{(2)}, \sigma_{k_2}^{(2)}, 0, \ldots, 0)$. The truncation of the eigenbasis is performed once the iteration level exceeds the number of desired SPOD modes, i.e. when $m \geq d+1$.

Alternatively, the eigenbasis can be initialized from a previously computed SPOD basis as $\mathbf{U}_k^{(1)} = \mathbf{W}^{\frac{1}{2}}\boldsymbol{\Phi}_k^{\mathrm{old}}$. Initializing the algorithm with an initial SPOD basis $\boldsymbol{\Phi}_k^{\mathrm{old}}$ obtained from a batch computation or a streaming computation with a larger value d has the benefit of reducing the truncation error. This follows directly from the best rank-d property of the SVD.

4. Errors and convergence

The errors of the approximation can be quantified by comparing the rank-d solutions at the mth iteration level to the reference solution $\boldsymbol{\Phi}_k^{\mathrm{batch}}$ and $\boldsymbol{\Sigma}_k^{\mathrm{batch}}$ obtained from the batch algorithm described in Section 2.

Errors with respect to batch solution. We define two error quantities

$$e_j^{\phi,\mathrm{batch}}(m) = \sum_{k=1}^{n_{\mathrm{freq}}} \left(1 - \max_j \left\langle \boldsymbol{\phi}_{k_j}^{(m)}, \boldsymbol{\phi}_{k_j}^{\mathrm{batch}} \right\rangle_E\right) \text{ and} \tag{38}$$

$$e_j^{\lambda,\mathrm{batch}}(m) = \sum_{k=1}^{n_{\mathrm{freq}}} \left| \frac{\lambda_{k_j}^{(m)} - \lambda_{k_j}^{\mathrm{batch}}}{\lambda_{k_j}^{\mathrm{batch}}} \right|. \tag{39}$$

that measure the error in the jth eigenvector and eigenvalue, respectively. The eigenvector error given by Eq. (38) is defined in terms of the inner product (8) and compares the patterns of two modes, i.e. it is 0 for identical and 1 for orthogonal modes. The maximum over the mode rank index is taken to ensure that the most similar modes are compared to each other. This is important as similar modes can swap order between iterations.

Convergence with respect to previous solution. Estimates for the convergence of the eigenvectors and eigenvalues are defined analogously in terms of their values at the previous iteration level $m-1$. The resulting measures of convergence

$$e_j^{\phi,\mathrm{prev}}(m) = \sum_{k=1}^{n_{\mathrm{freq}}} \left(1 - \max_j \left\langle \boldsymbol{\phi}_{k_j}^{(m)}, \boldsymbol{\phi}_{k_j}^{(m-1)} \right\rangle_E\right) \text{ and} \tag{40}$$

$$e_j^{\lambda,\mathrm{prev}}(m) = \sum_{k=1}^{n_{\mathrm{freq}}} \left| \frac{\lambda_{k_j}^{(m)} - \lambda_{k_j}^{(m-1)}}{\lambda_{k_j}^{(m-1)}} \right|, \tag{41}$$

for the jth eigenfunction and eigenvalue, respectively, can be monitored during runtime.

5. Examples

This section demonstrates the performance of the proposed streaming SPOD algorithm on two examples. The first example is a high-fidelity numerical simulation of a turbulent jet [19], and the second example is optical flow obtained from a high-speed movie of a stepped spillway experiment [20,21]. An overview of the databases and SPOD parameters is presented in Table 1. The SPOD parameters are chosen according to best practice. A discussion of how to choose them is beyond the scope of this work. The same applies to detailed physical interpretations of the results. Here, we focus on the performance and convergence of the streaming algorithm as compared to its offline batch counterpart. We use a Hanning window for the Fourier transformation and set the number of retained SPOD modes to $d = 5$. The effect of eigenbasis truncation is discussed in more detail in Section 6.

Table 1
Parameters for the two example databases and the SPOD. The spectral estimation parameters n_{freq}, n_{ovlp} and n_{blk} are identical for batch and streaming SPOD. d is the number of desired modes for the streaming algorithm.

Database					SPOD parameters			
Case	$\mathbf{q}$	n_{x_1}	n_{x_2}	n_t	n_{freq}	n_{ovlp}	n_{blk}	d
Jet	p^{symm}	175	39	10,000	256	128	78	5
Spillway	$[u, v]$	224	160	19,782	512	256	77	5

5.1. Example 1: large eddy simulation of a turbulent jet

The turbulent jet is a typical examples of a stationary flow. A number of studies, see e.g. [22] for an early experimental and [23] for a recent numerical example, use SPOD to analyze jet turbulence. Our first is example is an LES of a Mach 0.9 jet at a jet diameter-based Reynolds number of $1.01 \cdot 10^6$ [19]. The LES was calculated using the unstructured flow solver "Charles" [24]. The database consists of 10,000 snapshots of the axisymmetric component of the pressure field obtained as the zeroth azimuthal Fourier component of the flow. We choose to resolve 129 positive frequencies by setting $n_{\mathrm{freq}} = 256$. Each block therefore consists of 256 snapshots. We further use a 50% overlap by letting $n_{\mathrm{ovlp}} = 128$. This results in a total of 77 blocks for the spectral estimation, each of which represents one realization of the temporal Fourier transform. The first snapshot of the database is visualized in Fig. 3. The chaotic nature of the flow becomes apparent at first glance.

Fig. 4(a) shows the batch SPOD spectrum obtained for the spectral estimation parameters listed in Table 1. Each line represents the energy spectrum associated with a single mode index j. The total number of modes is equal to the number of blocks, i.e. $n_{\mathrm{blk}} = 77$ in this example. Most of the energy is concentrated in the large-scale structures that dominate at low frequencies. The roll-off of the distribution at higher frequencies is indicative of an energy cascade that transfers energy from larger to smaller scales. Over a certain frequency interval $0.1 \lesssim f \lesssim 0.6$, the first mode is significantly more energetic that the other modes. This low-rank behavior has important physical implications discussed elsewhere [25]. The spectra of the five leading modes are replicated in Fig. 4(b) and compared to the results obtained using streaming SPOD ($\circ$ symbols). It can be seen that the two results are almost indistinguishable. This provides a first indication that the streaming SPOD algorithm accurately approximates the SPOD eigenvalues. We will quantify this observation in the context of Fig. 6.

After establishing that the modal energies are well approximated by the streaming algorithm, we next examine the modal structures. Fig. 5 shows a side-by-side comparison of the first ($j = 1$), third ($j = 3$) and fifth ($j = 5$) modes at two representative frequencies ($f = 0.1$, top half and $f = 0.6$, bottom half). The leading modes (first and fourth row) computed using streaming SPOD are almost indistinguishable from the reference solution for both frequencies. The third modes (second and fifth row) still compare well. More differences become apparent for the fifth modes. It has to be kept in mind though, that the subdominant modes are in general more difficult to converge. This exemplifies the importance of being able to converge second-order statistics from long data sequences.

In Fig. 6, we next investigate the errors and convergence behavior for the jet example in terms of the quantities defined in Eqs. (38)–(41). The eigenvalue error in Fig. 6(a) drops by approximately one order of magnitude from beginning to end. As anticipated from Fig. 4(b), the eigenvalue error is generally small, i.e. below the per mil range after the first iteration. The eigenvalue convergence is addressed in Fig. 6(b). Starting from the end of the initialization phase (gray shaded area), the convergence measure drops by about two orders of magnitude. The error and

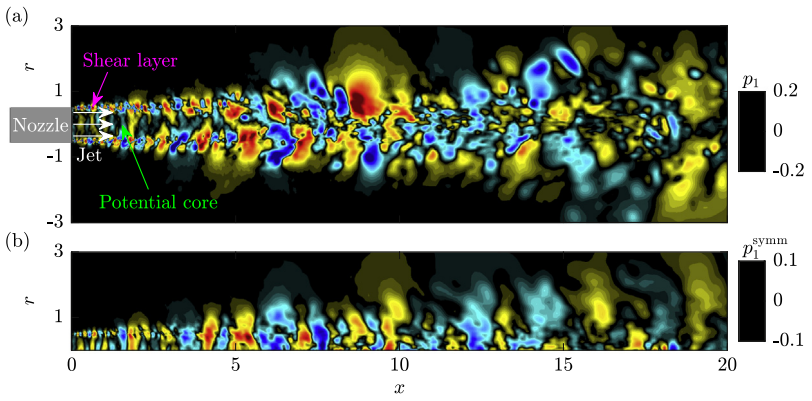

Fig. 3. Fluctuating pressure of the first snapshot of the turbulent jet LES: (a) streamwise plane; (b) symmetric pressure component only. The boundary layer inside the nozzle is turbulent, whereas the flow inside the potential core is laminar. The potential core collapses after approximately 5 jet diameters.

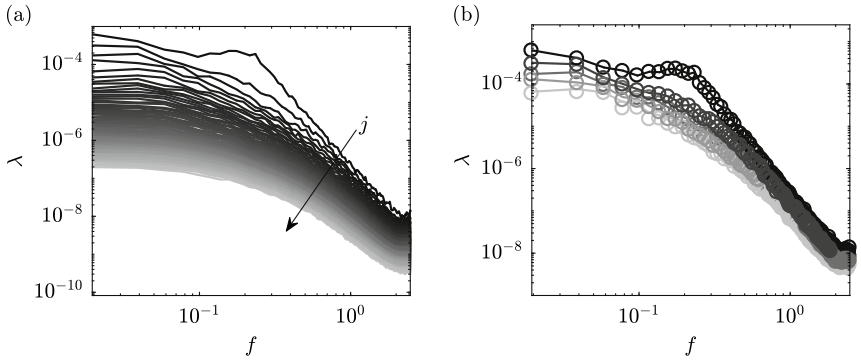

Fig. 4. SPOD energy spectra of the turbulent jet obtained using batch SPOD and streaming SPOD: (a) all $n_{\text{blk}} = 77$ eigenvalues computed using batch SPOD (——); (b) $d = 5$ leading eigenvalues calculated using streaming SPOD (○). The batch solution (——) is shown for comparison. j indicates the mode index from black ($j = 1$, most energetic) to light gray ($j = n_{\text{blk}}$ in (a) and $j = d$ in (b), least energetic).

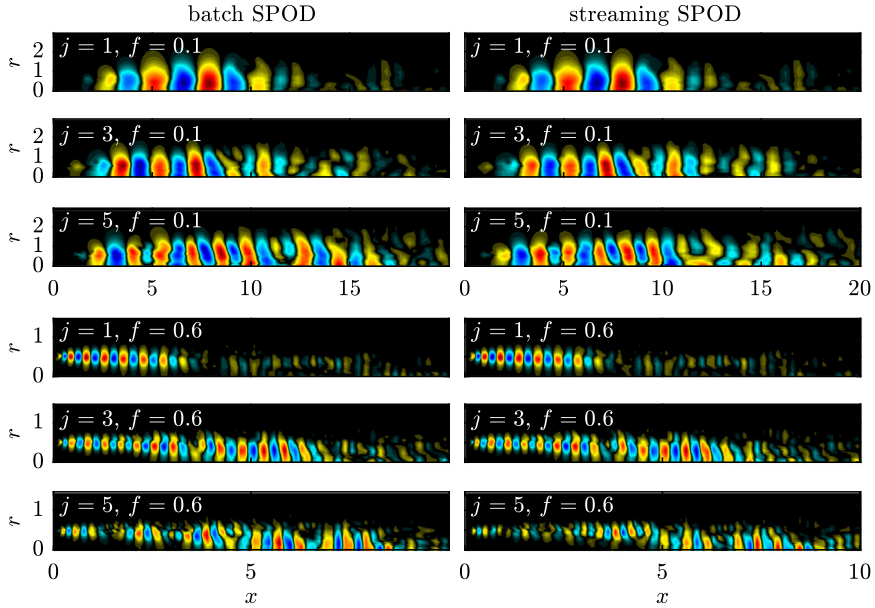

Fig. 5. Side-by-side comparison of SPOD modes of the pressure field calculated using batch SPOD (left column) and streaming SPOD (right column) for the jet example.

convergence of the eigenvectors are investigated in Fig. 6(c) and 6(d), respectively. It is observed that the eigenvector error drops monotonically for all five modes. The similarity of the leading batch and streaming SPOD modes previously seen in Fig. 5 is reflected by the small error of 0.6%. Similarly, the differences in the fifth modes result in a 25% error according to the metric. Since the eigenvalue is accurately predicted at the same time, we conclude that this large error is primarily a result of the slow statistical convergence of the

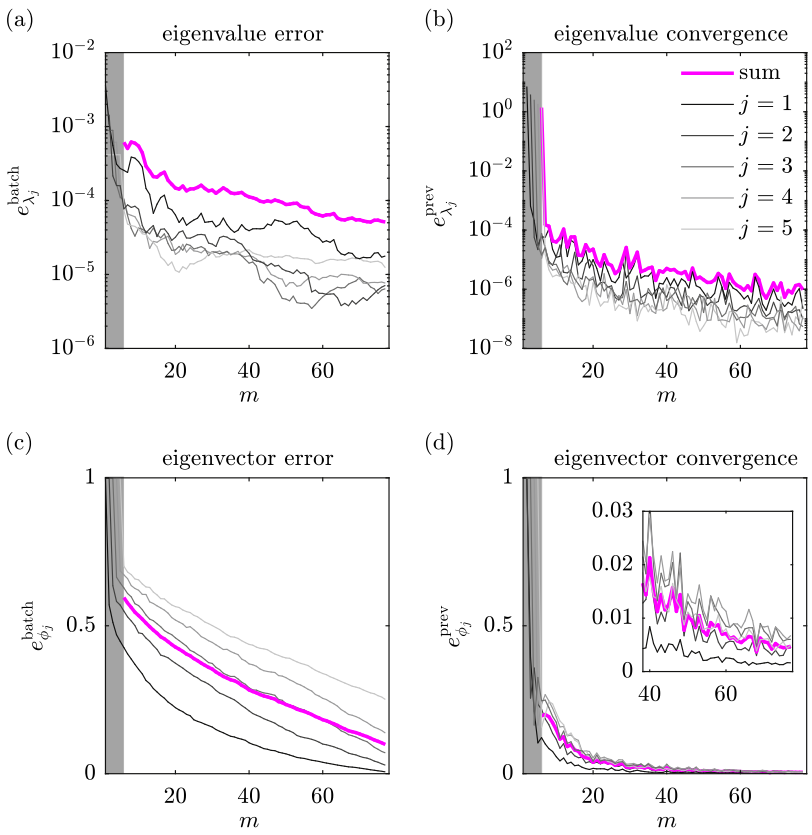

Fig. 6. Streaming SPOD error and convergence for the turbulent jet: (a) eigenvalue error; (b) eigenvalue convergence; (c) eigenvector error; (d) eigenvalue convergence. The magenta lines show the cumulative error in (a, c) and the mean of the convergence metric in (b, d), respectively. The shaded area demarcates the initial region $1 \leq m \leq d+1$ in which the eigenbasis is still rank deficient.

subdominant modes. The inset in Fig. 6(d) exemplifies this slow convergence. After about 50 iterations, the errors of most modes are below 1%.

5.2. Example 2: optical flow of a stepped spillway

The second example is that of a laboratory stepped spillway [20]. Stepped spillways are hydraulic structures designed to control flow release and to achieve high energy dissipation. The two-phase flow of the laboratory spillway is filmed using a high-speed camera and an optical flow algorithm [21,26] was used to estimate the streamwise and normal velocity components of the air–water mixture. The parameters of the optical flow database are summarized in Table 1.

Fig. 7 shows an example of the raw video data and a processed snapshot of the instantaneous streamwise velocity component. As for the jet example, we will not address the complex multi-phase physics of the setup, but focus on the performance of the streaming SPOD algorithm instead. We have selected the spillway as a second example to investigate the algorithm's performance under high noise conditions. The high noise level of the measurement is apparent in Fig. 7(b).

As before, spectra obtained using batch SPOD and the streaming version are compared in Fig. 8. It is observed that the modal energies asymptote towards a constant value for $f \gtrsim 200$. The plateau seen at these frequencies indicates the noise floor of the measurement. An inspection of the SPOD modes confirms this conjecture. Modes in this region are dominated by noise and show no apparent structure (not shown).

The comparison in Fig. 9 shows that the SPOD modes computed using the streaming algorithm closely resemble their batch SPOD counterparts. At the lower frequency (left), the SPOD modes are comprised of surface waves and oscillations of the shear-layer between the step ridges. Surface waves are the dominant structures at higher frequencies (right). Increasingly high noise levels are observed in the less energetic modes, in particular for the higher frequency case.

The eigenvalue error is studied in Fig. 10(a). Initially, the error is significantly larger as compared to the turbulent jet case shown in Fig. 6(a). Subsequently, a faster drop-off allows the eigenvalue error to recover values similar to those found for the jet example. The eigenvalue convergence behavior shown in Fig. 10(b) is very similar to that of the jet example.

The eigenvector error and convergence are plotted in Fig. 10(c) and 10(d), respectively. Both metrics occasionally undergo rapid changes, most prominently at iteration level $m = 45$. Sudden drops in the error are directly associated with peaks in the convergence measure. This behavior occurs when an eigenvector in the truncated basis gets replaced by a different structure. The re-orthogonalization of the eigenbasis after such an event leads a better correspondence with the batch solution. At the same time, the convergence measure spikes as a result of the change in modal structure. The error ranges between 10% (first mode) and 44% (fifth mode). The similarity of the modes depicted in Fig. 9 and the lower errors in the jet case, as seen in Fig. 6, strongly suggest that these relatively high errors are mainly associated with measurement noise.

6. Effect of eigenbasis truncation

Spectral estimation parameters aside, the desired number of SPOD modes d is the only additional user input required by the streaming algorithm. The basis truncation inevitably leads to an approximation error that originates from discarding the vector

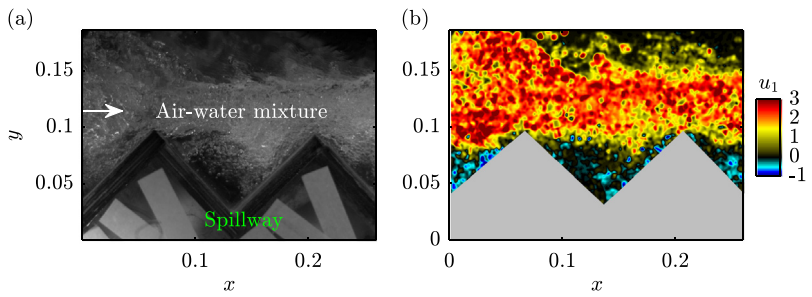

Fig. 7. First snapshot of the stepped spillway: (a) raw video frame; (b) streamwise velocity computed using optical flow. The air–water flow is characterized by instability growth, air entrainment and strong turbulence.

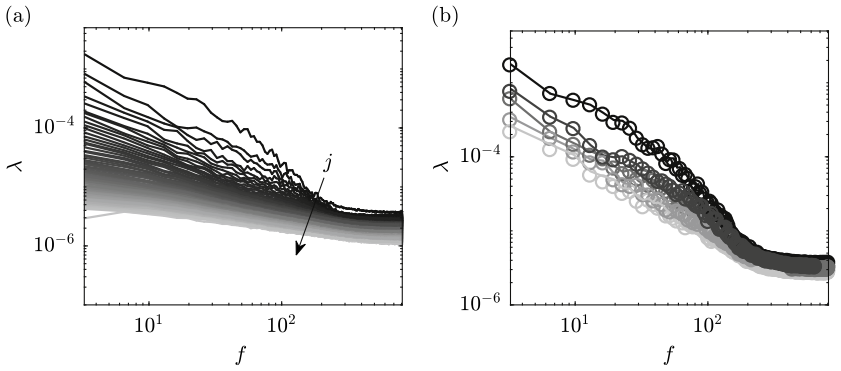

Fig. 8. Same as Fig. 4, but for the spillway example.

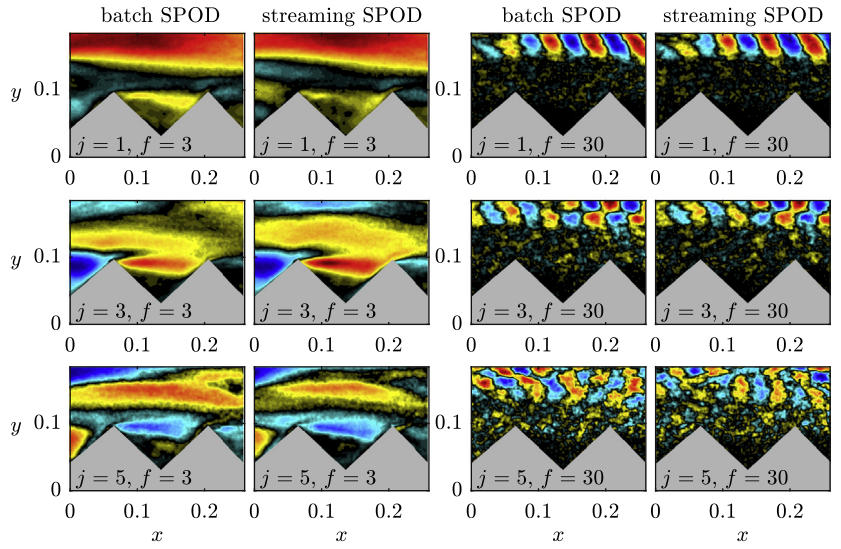

Fig. 9. Side-by-side comparison of SPOD modes calculated using batch SPOD (first and third column) and streaming SPOD (second and fourth column) for the spillway example. The streamwise velocity is shown.

component orthogonal to the span of the existing basis vectors $\mathbf{U}_k^{(m-1)}$.

Fig. 11 compares eigenvalue and eigenvector errors of the first SPOD mode for four different values of d. The jet and spillway examples are shown in Fig. 11(a, b) and 11(c, d), respectively. It is observed that even restricting the basis to a single vector, i.e. the most aggressive truncation possible, does not lead to significant errors. For $d \geq 5$, all error metrics shown in Fig. 11(a–c) are almost indistinguishable. Small differences are observed in the eigenvector error for the spillway example. In 11(d), the eigenvector error ranges between 10% for $d = 5$ and 2% for $d = 20$. As discussed in Fig. 9, this discrepancy is mainly related to data noise.

After establishing that retaining only a few eigenvectors is sufficient to control the truncation error, we now focus on the effect of truncation on the suboptimal modes. Analogous to Fig. 11, we compare the truncation errors of the fifth mode in Fig. 12. Its error characteristics are similar to the ones of the first mode.

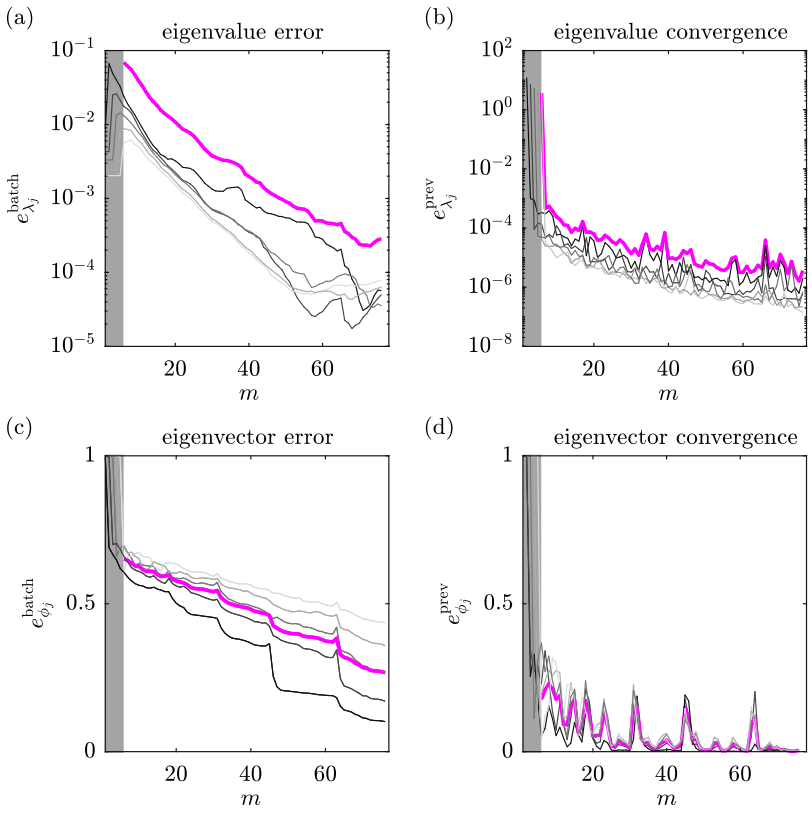

Fig. 10. Same as Fig. 6 but for the spillway example.

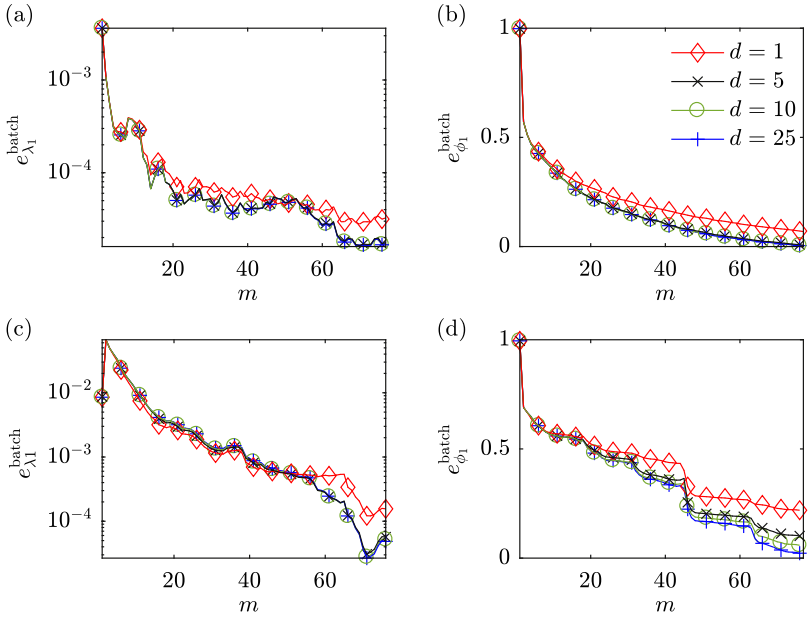

Fig. 11. Eigenvalue (left column) and eigenvector (right column) errors of the first SPOD mode for different numbers of basis vectors d: (a, b) turbulent jet; (c, d) spillway.

This can be seen by comparing Fig. 12(a, b) to Fig. 11(a, b). By increasing the basis size to $d = 10$, the final truncation errors are noticeably reduced, but adding more vectors does not lead to further reduction of the already low errors. For the spillway case shown in Fig. 12(c, d), the effect of noise in the data becomes apparent once more. Here, increasing the basis size from $d = 10$ to $d = 25$ reduces both the eigenvalue and eigenvector errors. At the same time, however, the earlier comparison of the mode shapes in Fig. 9 demonstrated that the coherent large-scale structures are accurately captured, even for $d = 5$.

The truncation error analysis suggests that the basis size d should be chosen somewhat larger than the desired number of modes. It is also important to emphasize that the definitions of the

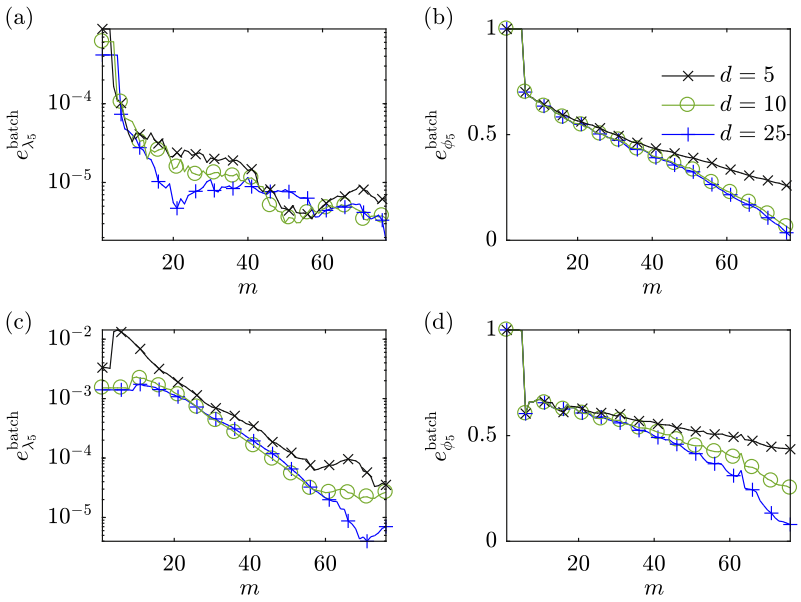

Fig. 12. Eigenvalue (left column) and eigenvector (right column) errors of the fifth SPOD mode for different numbers of basis vectors d: (a, b) turbulent jet; (c, d) spillway.

truncation errors rely on the batch solution as a reference, which itself may not be statistically fully converged.

7. Discussion

In this work, we introduce an algorithm that recursively updates the SPOD of large or streaming datasets. In Sections 4–6, we demonstrate that the algorithm is capable of converging the most energetic SPOD modes while lifting the requirement to store potentially prohibitively large amounts of data.

The batch algorithm requires storage of $n_x \times n_{\text{var}} \times n_t$ data points plus another $n_x \times n_{\text{var}} \times \left(\frac{n_{\text{freq}}}{2}+1\right) \times n_{\text{blk}}$ points for the spectral estimation of a real signal. In its simplest implementation, all data is loaded into memory simultaneously. If the dataset is too large to be stored in memory, the n_{blk} temporal Fourier transforms can be computed a priori and stored, fully or partly, on hard drive, and then be reloaded and processed frequency by frequency. The drawbacks of this approach are the significantly longer computing time resulting from the read/write operations, and the additional storage requirements. For higher-dimensional data, e.g. three-dimensional fields, snapshots totaling multiple terabytes are likely to be required to converge the second-order statistics, in particular those of subdominant modes. In such cases, batch SPOD may become computationally intractable altogether.

The streaming SPOD algorithm, on the other hand, has a much lower storage requirement of $n_x \times n_{\text{var}} \times n'_{\text{freq}} + n_x \times n_{\text{var}} \times \left(\frac{n'_{\text{freq}}}{2}+1\right) \times d$ data points for $\mathbf{X}_k^{(m)}$ and $\mathbf{U}_k^{(m)}$, respectively, plus a number of small fields that do not scale with the large dimensions in space and time. Here, $n'_{\text{freq}} \leq n_{\text{freq}}$ is the number of frequencies to be analyzed. Ideally, $\mathbf{X}_k^{(m)}$ and $\mathbf{U}_k^{(m)}$ are stored and updated in memory during runtime. Besides its low storage requirements, the algorithm achieves computational efficiency by employing MGS steps for the orthogonalization.

A useful implication of the ergodicity assumption is that it offsets the need to store a single long time-series. In Section 3, we used overlapping blocks to increase the number of Fourier samples in cases where the total number of snapshots is limited. A quite different scenario occurs when dealing with fast data streams. In such a scenario, we can take advantage of the fact that ergodicity permits arbitrarily long gaps between sampling periods of two consecutive data blocks $\mathbf{X}_k^{(m)}$ and $\mathbf{X}_k^{(m+1)}$. This is advantageous in experimental setups such as time-resolved particle image velocimetry (TR-PIV). It suffices to utilize n_{freq} consecutive snapshots at a time, perform the computationally costly cross-correlations to obtain velocity data, and update the SPOD eigenbasis before continuing to sample data. This procedure allows, in principle, to converge second-order flow statics over arbitrarily long time horizons.

Acknowledgments

O.T.S. gratefully thanks Tim Colonius and acknowledges support of the Office of Naval Research under grant no. N00014-16-1-2445 with Dr. Knox Millsaps as program manager for the portion of the work that was completed at Caltech. Special thanks are due to Patrick Vogler for reviewing the manuscript and sharing his insights, and to Tim Colonius, Andres Goza and Matthias Kramer for making valuable comments. The author gratefully acknowledges Matthias Kramer and Hubert Chanson for providing the experimental optical flow data. The experiments were undertaken in the hydraulics laboratory at the University of Queensland. The LES study was supported by NAVAIR SBIR project, under the supervision of Dr. John T. Spyropoulos. The main LES calculations were carried out on CRAY XE6 machines at DoD HPC facilities in ERDC DSRC.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2018.11.009.

References

[1] K. Taira, S.L. Brunton, S. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordeyev, V. Theofilis, L.S. Ukeiley, AIAA J. https://doi.org/10.2514/1.J056060.
[2] C.W. Rowley, S.T. Dawson, Annu. Rev. Fluid Mech. 49 (2017) 387–417.
[3] L. Sirovich, Quart. Appl. Math. 45 (3) (1987) 561–571.
[4] N. Aubry, P. Holmes, J.L. Lumley, E. Stone, J. Fluid Mech. 192 (1988) 115–173.

[5] B.R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, F. Thiele, J. Fluid Mech. 497 (2003) 335–363.
[6] J.L. Lumley, New York, 1970.
[7] A. Towne, O.T. Schmidt, T. Colonius, J. Fluid Mech. 847 (2018) 821–867, http://dx.doi.org/10.1017/jfm.2018.283.
[8] P.J. Schmid, J. Fluid Mech. 656 (2010) 5–28, http://dx.doi.org/10.1017/S0022112010001217.
[9] M.S. Hemati, M.O. Williams, C.W. Rowley, Phys. Fluids 26 (11) (2014) 111701.
[10] H. Zhang, C.W. Rowley, E.A. Deem, L.N. Cattafesta, arXiv preprint arXiv:1707.02876.
[11] P. Businger, BIT 10 (3) (1970) 376–385.
[12] R.D. De Groat, R.A. Roberts, in: Circuits, Systems and Computers, 1985. Nineteeth Asilomar Conference on, IEEE, 1985, pp. 601–605.
[13] M. Brand, Linear Algebra Appl. 415 (1) (2006) 20–30.
[14] M. Brand, Proc. Eur. Conf. Comput. Vis. (2002) 707–720.
[15] M. Brand, in: Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, 2003, pp. 37–46.
[16] P.D. Turney, P. Pantel, J. Artificial Intelligence Res. 37 (2010) 141–188.
[17] T. Braconnier, M. Ferrier, J.-C. Jouhaud, M. Montagnac, P. Sagaut, Comput. & Fluids 40 (1) (2011) 195–209.
[18] O.M. Solomon Jr., NASA STI/Recon Technical Report N 92.
[19] G.A. Brès, P. Jordan, V. Jaunet, M. Le Rallic, A.V.G. Cavalieri, A. Towne, S.K. Lele, T. Colonius, O.T. Schmidt, J. Fluid Mech. 851 (2018) 83–124, http://dx.doi.org/10.1017/jfm.2018.476.
[20] M. Kramer, H. Chanson, Environ. Fluid Mech. (2018) 1–19.
[21] G. Zhang, H. Chanson, Exp. Therm Fluid Sci. 90 (2017) (2017) 186–199.
[22] M.N. Glauser, S.J. Leib, W.K. George, Turbul. Shear Flows 5 (1987) 134–145.
[23] O.T. Schmidt, A. Towne, T. Colonius, A.V.G. Cavalieri, P. Jordan, G.A. Brès, J. Fluid Mech. 825 (2017) 1153–1181, http://dx.doi.org/10.1017/jfm.2017.407.
[24] G.A. Brès, F.E. Ham, J.W. Nichols, S.K. Lele, AIAA J. 55 (4) (2017) 1164–1184.
[25] O.T. Schmidt, A. Towne, G. Rigas, T. Colonius, G.A. Brès, J. Fluid Mech. 855 (2018) 953–982, http://dx.doi.org/10.1017/jfm.2018.675.
[26] T. Liu, A. Merat, M.H.M. Makhmalbaf, C. Fajardo, P. Merati, Exp. Fluids 56 (8) (2015) 1–23.