Research paper

Sparse eigenbasis approximation: Multiple feature extraction across spatiotemporal scales with application to coherent set identification

Gary Froyland∗, Christopher P. Rock, Konstantinos Sakellariou

School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia

A R T I C L E   I N F O

Article history:
Received 5 December 2018
Accepted 12 April 2019
Available online 13 April 2019

Keywords:
Spectral clustering
Sparse approximation
Coherent structure
Transfer operator

A B S T R A C T

The output of spectral clustering is a collection of eigenvalues and eigenvectors that encode important connectivity information about a graph or a manifold. This connectivity information is often not cleanly represented in the eigenvectors and must be disentangled by some secondary procedure. We propose the use of an approximate sparse basis for the space spanned by the leading eigenvectors as a natural, robust, and efficient means of performing this separation. The use of sparsity yields a natural cutoff in this disentanglement procedure and is particularly useful in practical situations where there is no clear eigengap. In order to select a suitable collection of vectors we develop a new Weyl-inspired eigengap heuristic and heuristics based on the sparse basis vectors. We develop an automated eigenvector separation procedure and illustrate its efficacy on examples from time-dependent dynamics on manifolds. In this context, transfer operator approaches are extensively used to find dynamically disconnected regions of phase space, known as almost-invariant sets or coherent sets. The dominant eigenvectors of transfer operators or related operators, such as the dynamic Laplacian, encode dynamic connectivity information. Our sparse eigenbasis approximation (SEBA) methodology streamlines the final stage of transfer operator methods, namely the extraction of almost-invariant or coherent sets from the eigenvectors. It is particularly useful when used on domains with large numbers of coherent sets, and when the coherent sets do not exhaust the phase space, such as in large geophysical datasets.

© 2019 Published by Elsevier B.V.

1. Introduction

Spectral clustering has found broad application in areas such as network analysis, manifold learning (e.g. diffusion maps), Lagrangian dynamics, and stochastic processes. The output of a spectral method is the spectrum and eigenvectors of some matrix or linear operator, typically either of Laplace type such as a graph Laplacian or Laplace–Beltrami operator on a manifold, or of Markov type.1 The eigenvectors carry geometric structure to e.g. help discern the topology of an unknown manifold, cluster point data, or analyse nonlinear dynamics.

For example, the Laplace–Beltrami operator arises as the generator of heat flow on a manifold. The four leading eigenfunctions (corresponding to the largest four (real) eigenvalues \(0 = \lambda_1 \geq \cdots \geq \lambda_4\)) of this operator defined on a two-

∗ Corresponding author.

E-mail address: g.froyland@unsw.edu.au (G. Froyland).

1 often there is a "spectral mapping" relationship between these two types, of the form \(M = \exp(L)\), where \(M\) is Markov type and \(L\) is Laplace type.
dimensional “union of five disks” manifold\(^2\) are shown in the upper row of Fig. 1. These eigenfunctions – beyond the first “trivial” constant eigenfunction corresponding to \(\lambda_1 = 0\) and shown in Fig. 1 (upper left) – describe signed “heat modes” that decay most slowly under the heat flow, at rates \(\sim e^{\xi_i} t\), \(i = 2, \ldots, 4\). Because the four “small disks” at the periphery of the main disk have narrow channels connecting them to the main disk, the slowest decaying mode (second from the left in the upper row of Fig. 1) has a lot of heat (red) in the lower small disk and lower part of the main disk and is cool (blue) in the upper small disk and upper part of the main disk. The four “small disks” are correctly detected in the upper row of Fig. 1, but they are mixed together in the leading four eigenfunctions. We wish to separate these four features and this is what is achieved in the lower row of Fig. 1, by finding an approximate sparse basis through orthogonal rotation of the four leading eigenfunctions.

Arguably the most popular method of performing this separation is the embedding of (discrete versions of) the second to fourth eigenfunctions in \(\mathbb{R}^3\) and applying a hard clustering algorithm such as k-means; see e.g. [1] for a detailed description and the historical development of k-means. In this example, requesting four k-means clusters results in a grouping where each of the four small disks is merged with approximately one quarter of the main disk (see Fig. 2 (left)); this is because k-means partitions the dataset/domain. In order to at least partially isolate the peripheral disks, one must request five k-means clusters; see Fig. 2 (right). A key advantage of our approach is that we do not necessarily separate by partitioning, but instead may classify a large part of the dataset/domain as “unclustered”. Such an idea has been partly implemented in [2] by adding one more cluster to k-means as a “background” cluster, but we show in Sections 5.2 and 5.3 that this is not a good approach for more challenging dynamics. Other separation approaches are discussed in Section 2.1.

\(^2\) with homogeneous Neumann boundary conditions.
Perhaps the most similar precursor to our approach in the dynamical systems literature, applied in the context of Markov chains, is the search for a linear transformation of the leading eigenvectors [3] to make them as close as possible to a collection of indicator functions (vectors taking a small number of distinct values) spanning a similar subspace, under constraints of non-negativity and small intersection of support. The optimisation in [3] can be expensive for large vectors and large numbers of vectors, and will always return a full partition of the domain. An LU decomposition approach [4] has been effective in the case where the eigenvectors are already close to linear combinations of indicator vectors. This is a somewhat idealised situation and, in such a setting, all of the above methods should perform very well.

In the context of general Markov processes and Lagrangian dynamics, there are many situations where it is not instructive to partition all nodes or the entire manifold into almost-disconnected pieces according to the relevant eigenvectors. Rather, it is more informative to know that some parts of the network or manifold have a natural almost-disconnected structure, while the remainder (possibly a large remainder) of the network or manifold is well connected. The sparsity built into our new sparse basis approach will automatically identify those nodes or subsets of a manifold that do not belong to any almost-disconnected subset; such nodes or subdomains will be zeroed out by the sparsification.

In Section 5 we illustrate our approach on spectral algorithms designed to identify almost-invariant or coherent sets in dynamical systems on manifolds. The term almost-invariant set refers to a set that is approximately invariant under autonomous dynamics. The term coherent set refers to a time-dependent family of sets that are approximately equivariant under time-dependent or nonautonomous dynamics. In contrast to an almost-invariant set, which is fixed in phase space, the family of coherent sets are in general mobile in phase space as time evolves; see [5] for a short overview. Almost-invariant and coherent sets in the phase space of complicated dynamics are important because they are the most predictable features in the medium term. These concepts have been used in molecular dynamics to identify conformations to aid drug design [6], in atmospheric dynamics to identify atmospheric vortices [7], in oceanography to identify ocean gyres [8] and eddies [9,10], and in fluid dynamics [11]. Spectral approaches to the identification of coherent features rely on the dominant eigenvectors of transfer operators or related operators, such as the dynamic Laplacian [12,13]. There are various categories of constructions, based on the problem to be solved; see Table 1. The final stage of the extraction of almost-invariant or coherent sets from eigenvectors or singular vectors has previously proceeded either by time-consuming hand-selected thresholds or by common clustering methods such as k-means.

A main goal of this research is to describe simple and robust procedures for automatically separating features represented in the eigenvectors or singular vectors arising from transfer operator methods. Our separation procedures streamlines the use of these methods on domains with large numbers of coherent sets, particularly when the coherent sets do not exhaust the phase space. Although the dynamical context is our primary motivation, we believe our approach will translate well to other application domains where disentangling the output of spectral clustering is important. We have thoroughly tested our approach on a variety of nonlinear dynamics and have proposed heuristics that have consistently performed well in our tests. We illustrate our new approach by separating coherent features in idealised fluid flow dynamics and in large-scale ocean currents in the North Atlantic derived from satellite altimetry.

Our main contributions include a refinement of the standard eigengap criterion for determining suitable numbers of features. Our Weyl-inspired refinement strongly highlights and appropriately scales gaps in the spectrum, emphasising natural time scales. We also develop new vector-based heuristics for determining suitable numbers of features; these help to determine good choices of dimension for the subspaces and therefore the corresponding spatial scales of features. We develop a “reliability” heuristic for the sparse basis vectors, including feature rejection criteria. These vector-based heuristics supplement the eigengap criteria and we have found them to be particularly useful in realistic situations where there is no clear eigengap. Finally, we propose automatic methods of hard thresholding of the sparse basis to provide hard separation of features.

Table 1
Summary of transfer operator and dynamic Laplace operator constructions to identify almost-invariant and coherent sets, and the dynamical settings in which each construction is used.

<table>
<thead>
<tr>
<th>Construction</th>
<th>Autonomous/Nonauton. Dynamics</th>
<th>Finite/Infinity Time</th>
<th>Closed/Open Dynamics</th>
<th>Uses time derivative</th>
<th>Objects Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Left eigenvectors of transfer operator [33]</td>
<td>Autonomous</td>
<td>∞</td>
<td>Closed</td>
<td>No</td>
<td>Almost-invariant sets</td>
</tr>
<tr>
<td>2. Left eigenvectors of generator [13,34]</td>
<td>Autonomous or Periodic</td>
<td>∞</td>
<td>Closed</td>
<td>Yes</td>
<td>Almost-invariant sets</td>
</tr>
<tr>
<td>3. Right eigenvectors of transfer operator (resp. generator) [35,36]</td>
<td>Autonomous</td>
<td>∞</td>
<td>Open</td>
<td>No (resp. yes)</td>
<td>Basins of attraction</td>
</tr>
<tr>
<td>5. Singular vectors of transfer operator [7,37]</td>
<td>Nonauton.</td>
<td>Finite</td>
<td>Closed</td>
<td>No</td>
<td>Finite-time coherent sets</td>
</tr>
<tr>
<td>6. Eigenvectors of dynamic Laplacian [12,38]</td>
<td>Nonauton.</td>
<td>Finite</td>
<td>Closed</td>
<td>No</td>
<td>Finite-time coherent sets</td>
</tr>
</tbody>
</table>
An outline of the paper is as follows. Section 2 provides a brief background on hard separation of spectral clustering output, sparse principal components analysis, and transfer operator and dynamic Laplace operator constructions in nonlinear dynamics. Section 3 describes in detail the variant of sparse basis approximation we will use, based on the SPCart algorithm [14]. Section 4 discusses the use of our sparse basis algorithm, including new eigenvalue and vector based heuristics for selecting the size of the basis and assigning reliability to the sparse basis output, and simple thresholding procedures. We present our numerical examples in Section 5 and conclude in Section 6.

2. Background

2.1. Separation of spectral clustering output by hard clustering

Denote the eigenvalue (resp. eigenvector) output of a spectral clustering method by ordered lists of eigenvalues \( \lambda_1 \geq \lambda_2 \geq \cdots \) and their corresponding eigenvectors \( v_1, v_2, \ldots \). Suppose that we wish to separate the features encoded in the vectors \( v_1, \ldots, v_r \). One of the most popular separation approaches is to perform hard clustering on the \( p \times r \) data array \( V = [v_1, \ldots, v_r] \), treating each row of this array as a Euclidean point in \( \mathbb{R}^r \). Fig. 2 illustrates this procedure with \( r = 3 \) using k-means for the hard clustering. In the context of nonlinear dynamics, the k-means algorithm [1] has been applied to eigenvector data embedded in Euclidean space, for example [2,15–18] to identify coherent sets. A “soft” variant, fuzzy c-means, has earlier been used in the dynamics context on eigenvector data [19,20] to identify almost-invariant sets. Other variants include k-medians [21], k-medoids [22], and kernel k-means [23].

Other hard separation approaches for spectral clustering include [24], who try to find a rotation of the eigenvectors that brings them as close as possible to a collection of binary vectors, which represent a feature partition. The work of [25] builds on [24] by proposing rotating the eigenvectors to approximately achieve a collection of nonnegative vectors, followed by a maximum likelihood assignment. Related to this is [26] who search for nonnegative vectors “nearby” the eigenvectors via a penalty term. Another recent contribution includes [27], who search for sparse orthogonal vectors that maximise the trace of the usual bilinear form involving the Laplace matrix, followed by an application of k-means. All of the above approaches produce a partition of the dataset/domain (as does k-means), and this is something we wish to avoid through the use of sparse bases. Moreover, the explicit imposition of nonnegative vectors removes what we have found to be an important indicator of the quality of the separation, as we show in Section 4. There are several other related methods that involve manipulating the weights in the affinity matrix, but we omit discussing these because in our dynamical systems applications the Markov and Laplace operators that arise have additional special properties we wish to preserve.

2.2. Sparse principal component analysis

Principal component analysis (PCA) is an unsupervised learning technique often used for dimension reduction and feature selection. PCA may be viewed as the process of finding a linear projection from a high-dimensional space (occupied by the dataset) onto a low-dimensional space that preserves as much of the variance of the original dataset as possible. One drawback of PCA is that every variable in the lower-dimensional space generally depends on all of the variables in the high-dimensional space. This shortcoming motivated sparse principal component analysis (SPCA), which modifies PCA by imposing constraints (resp. adding a penalty) to force (resp. encourage) each variable in the lower-dimensional space to depend on only a subset of the original variables. In 2003, Jolliffe et al. proposed the first sparse principal component analysis technique, called SCoFLASS [28], and several versions of SPCA have appeared since then. Many such versions, termed deflation methods, find one sparse principal component at a time, then deflate the original data in that direction and find the next orthogonal principal component, e.g. [28,29]. To avoid sequential deflation finding a non-global optimum, so-called block methods have been proposed, which optimise all the subspaces at once [14,30–32].

Because of its simplicity, efficiency, and robustness, in this paper we will use a tailored version of the sparse principal component analysis by rotation and truncation (SPCart) algorithm [14], which is a block method. Letting \( \mathcal{V} = \text{span}\{v_1, \ldots, v_r\} \) we implicitly assume that the eigenvectors from our spectral clustering satisfy the following assumption.

**Assumption A.** There is a \( \mathcal{V} \)-approximating, \( r \)-dimensional subspace \( S \subset \mathbb{R}^d \) with a nonnegative sparse basis \( s_1, \ldots, s_r \) whose supports have little or no intersection.

**Assumption A** should hold when there exist pairwise disjoint subsets of the graph or manifold that are sufficiently strongly disconnected, provided suitable eigenvectors \( \{v_1, \ldots, v_r\} \) of a Laplace-type or Markov-type operator or matrix are chosen. Fig. 1 (lower row) and Fig. 3 (right column) illustrate the type of sparse basis vectors we will obtain.

2.3. Markov- and Laplace-type operators arising in nonlinear dynamics

Our examples in Section 5 will be drawn from nonlinear dynamics and we provide a brief overview here, referring the reader to original papers for further details. One has a smooth domain \( M \), typically a smooth \( d \)-dimensional manifold or subset of \( \mathbb{R}^d \), on which the nonlinear dynamics acts.
Fig. 3. Left: the 8 leading eigenvectors $v_1, \ldots, v_8$ (upper to lower) of the dynamic Laplacian constructed from the Bickley jet (see Section 5.1). Right: an approximate sparse basis $s_1, \ldots, s_8$ output by Algorithm 3.1 separating the 8 main coherent sets; small negative values of have been removed from the vectors $s_7, s_8$. 
There are several related linear operators that are intimately connected with the nonlinear evolution on phase space. These operators fall into two broad classes, which we will call Markov-type and Laplace-type. The former have their spectrum contained in \( \{ z \in \mathbb{C} : |z| \leq 1 \} \), the closed unit disk in the complex plane, and the spectrum of the latter is contained in \( \{ z \in \mathbb{C} : \Re(z) \leq 0 \} \), the closed half plane with non-positive real part.

In the most straightforward case, the dynamics is generated by repeated application of a transformation \( T : M \to M \) or by solving a differential equation \( \dot{x} = F(x) \). Both of these mechanisms generate autonomous dynamics because the underlying governing dynamics does not depend on time. In the former case one uses left \(^3\) eigenvectors of the transfer operator (Markov type) to identify subsets of the domain that are approximately invariant under repeated application of the dynamics; see Construction 1 in Table 1. In the latter case, one uses left eigenvectors of the generator (Laplace type) to detect almost-invariant sets in continuous time without trajectory integration; this is Construction 2 in Table 1. When there is open dynamics created by the existence of a “hole(s)” in phase space, one is interested in the basins of attraction; this is Construction 3 in Table 1, covering both discrete and continuous time. In many instances, there is a natural timescale of interest and one wishes to find subsets that are approximately invariant over this finite timescale; this is addressed in Construction 4 in Table 1.

Nonautonomous dynamics arises when the generating law varies with time. In discrete time one applies a sequence of different transformations \( T_0 \circ T_{n-1} \circ \cdots \circ T_1 \), and in continuous time, one solves a differential equation with time-dependent right-hand-side \( \dot{x} = F(x, t) \). Because the underlying dynamics changes over time, one wishes to find subsets that remain “coherent” over the finite time duration; this is addressed in Constructions 5 and 6 in Table 1. Finally, identifying coherent sets in time-dependent dynamics over an infinite time period is addressed in Construction 7 in Table 1. While we have described the dynamics above as deterministic, each of the Constructions 1–7 can also be applied to stochastic dynamics arising from transformations with additional noise, or stochastic differential equations.

Each of the Constructions 1–7 in Table 1 is a type of spectral clustering; Constructions 1, 3, 4, 5, and 7 produce Markov-type operators and Constructions 2, 3, and 6 produce Laplace-type operators. The eigenvectors arising from Constructions 4–6 are orthogonal in a suitable inner product space, while the vectors produced by Constructions 1, 2, 3, and 7 need not be. For the latter constructions, one applies e.g. QR factorisation to orthogonalise the vectors before applying Algorithm 3.1.

There are various ways in which the constructions in Table 1 can be numerically implemented. A thin spline implementation of Construction 5 is introduced in [40], and a spectral implementation in continuous time of Construction 5 is presented in [41]. A radial basis function implementation of Construction 6 is contained in [42] and a finite element implementation is introduced in [18]; the latter is the numerical implementation we will use for all dynamic Laplacian eigenvector computations in this paper. A graph-based method similar to Construction 6 is described in [2]. Diffusion map implementations of Constructions 5 and 6 are described in [15].

In Section 5 we will apply our method to output from Constructions 5 and 6; the former is of Markov type and the latter is of Laplace type. While the different constructions in Table 1 solve different dynamical problems, the properties of the eigenvectors from all constructions are strongly related because they arise from spectral approaches. We expect our methods to perform equally well on all constructions in Table 1.

3. Sparse eigenbasis approximation

Let \( v_1, \ldots, v_r \in \mathbb{R}^p \) be a set of linearly independent eigenvectors or singular vectors; typically \( r \ll p \). These vectors form a basis for a subspace \( V \subset \mathbb{R}^p \). We wish to transform these vectors to a basis of sparse vectors \( s_1, \ldots, s_r \in \mathbb{R}^p \) for a subspace \( S \subset \mathbb{R}^p \) with \( V \approx S \). Without loss of generality, we can assume that the vectors \( v_1, \ldots, v_r \) are an orthonormal basis; if not, we perform a QR factorisation of the matrix \( V = [v_1] \cdots [v_r] \) and extract the orthonormal columns. If the subspace \( V \) satisfies Assumption A, we can expect the sparse vectors \( s_1, \ldots, s_r \) to have supports with small overlaps, and therefore be close to orthogonal. Thus, we wish to find an orthogonal “rotation” matrix \( R \) that transforms \( v_1, \ldots, v_r \) into \( s_1, \ldots, s_r \) such that (i) the \( r \)-dimensional space \( S \subset \mathbb{R}^p \) spanned by \( s_1, \ldots, s_r \) is close to \( V \subset \mathbb{R}^p \) (approximate subspace preservation) and (ii) the vectors \( s_1, \ldots, s_r \) are sparse.

The approach we take is based on the SPCArt algorithm (sparse PCA via truncation and rotation [14]). We present below a simplified “full rank” version of SPCArt, tuned to the setting of Assumption A, which we call Sparse Eigenbasis Approximation (SEBA). Let \( \mathcal{E}^r \) denote the Stiefel manifold \( \{ A \in \mathbb{R}^{r \times r} : A^T A = I_r \} \) of \( r \times r \) orthogonal matrices, and let \( \mathbb{U}^{p \times r} = \{ A \in \mathbb{R}^{p \times r} : \text{each column of } A \text{ has } \ell_2 \text{ norm } 1 \} \). Define \( \| A \|_F := \sqrt{\sum_{j=1}^{p} \sum_{i=1}^{r} A_{ij}^2} \) to be the Frobenius norm and \( \| A \|_{1,1} := \sum_{j=1}^{p} \sum_{i=1}^{r} |A_{ij}| \) to be the \( \ell_{1,1} \) matrix norm. Given some small positive sparsity parameter \( \mu \), we wish to solve the following nonconvex optimisation problem for \( S \in \mathbb{U}^{p \times r} \) and \( R \in \mathcal{E}^r \):

\[
\arg\min_{S \in \mathbb{U}^{p \times r}, R \in \mathcal{E}^r} \frac{1}{2} \| V - SR \|_F^2 + \mu \| S \|_{1,1}.
\]
The $r$ columns of $S$ define a sparse basis \( s_1, \ldots, s_r \) for a subspace $S$ close to $\V$. The first term in (1) measures how close the rotated columns of $V$, namely $\VR^T$, are to the columns of $S$ (recall $\|VR^T - S\|_F = \|V - SR\|_F$ for orthogonal $R$). The second term measures the sparsity of the basis formed by the columns of $S$ using the common sparsity-inducing $\ell_1$ penalty.

Because of the nonconvexity of the problem (1), finding a global optimum $(S, R) \in \W \times \Theta$ is difficult. Hu et al. [14] proposed alternately fixing $R$ and optimising $S$, and fixing $S$ and optimising $R$ to find a local minimum $(S, R)$. Each of these individual optimisation problems are fast to solve exactly.

1. **Fixed $R$:** The optimisation for $S$ in (1) is exactly solved by “soft thresholding” [30]. Define a thresholding transformation $C_\mu : \R \rightarrow \R$ by $C_\mu(z) = \operatorname{sign}(z) \max(|z| - \mu, 0)$; $C_\mu$ is applied to vectors elementwise. For $j = 1, \ldots, r$, set the $j$th column of $S$ to $S_j = C_\mu((\VR^T)_j)/\|C_\mu((\VR^T)_j)\|$. 

2. **Fixed $S$:** The optimisation for $R$ becomes a Procrustes problem $\min_{R \in \Theta} \frac{1}{2} \|V - SR\|_F^2$. This problem may be efficiently exactly solved by polar decomposition [32]. Set $R = \text{Polar}(S^T \V)$, where $\text{Polar}(\cdot)$ is the orthonormal component of the polar decomposition; that is, $S^T \V = RH$, where $R$ is orthogonal and $H$ is symmetric and positive definite. If $S^T \V$ has singular value decomposition $S^T \V = PDQ^T$, then $R = PQ^T$. In MATLAB, we use $[P, -, Q] = \text{svd}(S^T \V, 0)$.

One initialises with $R = I_r$ and alternately applies steps 1 and 2 above until the change in $R$ is below a specified tolerance.\footnote{Note that if we define $S = VR^T$ for some $r \times r$ orthogonal (in fact, $R$ nonsingular is sufficient) matrix $R$, then the columns of $S$ will span the same subspace as the columns of $V$.} The parameter $\mu$ should be chosen less than $1/\sqrt{p}$ as $C_\mu$ sends the constant unit vector to the zero vector for $\mu \geq 1/\sqrt{p}$. Hu et al. [14] suggest using $\mu = 1/\sqrt{p}$ and with extensive experimentation we have found that this works well, as do values between 75% and 100% of $1/\sqrt{p}$. In our experiments we have used $\mu = 0.99/\sqrt{p}$ because in many cases the constant unit vector is part of our initial basis $\V$. Alternative thresholding methods were presented in [14], but in our experiments we found soft thresholding was the most robust and we use this in all computations. In summary, our sparse eigenbasis approximation (SEBA) algorithm is:

**Algorithm 3.1.** ((SEBA) - Input orthonormal $p \times r$ matrix $V$; output sparse $p \times r$ matrix $S$)

1. Set $\mu = 0.99/\sqrt{p}$ and $R = I_r$.
2. If a column of $V$ is very close to a constant vector, apply a small perturbation.
3. For $j = 1, \ldots, r$, set the $j$th column of $S$ to $S_j = C_\mu((\VR^T)_j)/\|C_\mu((\VR^T)_j)\|$.
4. Set $R = \text{Polar}(S^T \V)$.
5. If the matrix 2-norm of the difference between the revised $R$ in step 4 and the previous $R$ is larger than some tolerance, go to step 3; otherwise go to step 6.
6. For $j = 1, \ldots, r$, set $S_j = \text{sign}(\sum_{i=1}^p S_{ij})S_j$.
7. For $j = 1, \ldots, r$ set $S_j = S_j / \max_{1 \leq i \leq r} S_{ij}$.
8. Reorder the columns of $S$ so that $m_j := \min_{1 \leq i \leq r} S_{ij}$ is in decreasing order, $j = 1, \ldots, r$.

Step 2 avoids numerical pathologies associated with sparse thresholding a constant vector. In Step 6 we possibly change the sign of the columns of $S$ to ensure they are predominantly nonnegative and in Step 7 we scale the columns of $S$ so that their maximum value is 1; we will later interpret $S_{ij}$ as a likelihood of membership of index $i$ in the $j$th feature. Step 8 orders the columns of $S$ in terms of "reliability"; see Section 4.1 for a discussion.

As noted in [14], SPCArt (steps 1–4 of Algorithm 3.1) produces vectors $s_1, \ldots, s_r$ that should span a similar subspace to $\V$ because of the first term of (1). Because the input vectors $v_1, \ldots, v_r$ are orthonormal and the vectors $s_1, \ldots, s_r$ are arrived at by orthogonal rotation and truncation, the latter should also be close to orthogonal. The sparsity should be reasonably balanced because of the global optimisation of $R$ in step 4 and uniform thresholding across each vector in step 3. A short MATLAB code listing to implement Algorithm 3.1 is included in A.1.

4. **Using the SEBA Algorithm 3.1**

In this section we describe the “reliability” ordering in Step 8 of Algorithm 3.1, and introduce some rules of thumb for selecting an appropriate basis size for input to Algorithm 3.1 and how to extract a (sub)partition from the output of Algorithm 3.1. We also detail how to optionally apply weights to vector entries. Code is provided in Appendices A.2 and A.3.

4.1. **Ordering the sparse vectors in terms of “reliability”**

Under Assumption A, if Algorithm 3.1 is performing well we should be able to find a subspace $S$ that is a good approximation of $\V$ and which is spanned by a nonnegative sparse basis $s_1, \ldots, s_r$. Nonnegativity of $s_1, \ldots, s_r$ is not guaranteed by Algorithm 3.1 and we have found that if a sparse vector $s_j$ has one or more large negative entries, this is an indication that...
4.2. Selecting an appropriate number of input vectors

Consider a set \( v_1, \ldots, v_r \in \mathbb{R}^P \) of linearly independent eigenvectors as in Section 3. As one increases \( r \), the \( r \)th eigenvector becomes increasingly oscillatory, corresponding to increasingly rapidly decaying modes. The “spatial scale” of the features encoded in the \( r \)th eigenvector therefore decreases with increasing \( r \). Similarly, the \( r \)th eigenvalue describes the exponential decay rate of the \( r \)th eigenvector, and therefore a “temporal scale” for the corresponding features. In the dynamical systems context, the number \( r \) controls the smallest spatial scale at which eigenvectors identify almost-invariant or coherent sets, while the \( r \)th eigenvalue provides a lower bound on the temporal scale at which each of the almost-invariant or coherent sets mix with the rest of phase space. In particular, gaps in the spectrum indicate a jump in temporal scales, rather than a jump in spatial scales. The reliability ordering of Section 4.1 will guide the choice of a number \( k \leq r \) to indicate that Algorithm 3.1 has produced \( k \) reliable features at the spatiotemporal scale governed by \( r \).

4.2.1. Heuristics based on the spectrum

In spectral clustering, the eigengap heuristic (see [44] and references therein) is a common method for selecting an appropriate number of vectors for further analysis. The eigengap heuristic suggests to look for large gaps in the spectrum, i.e., if \( |\lambda_{r+1} - \lambda_r| \) is large compared to \( |\lambda_{i+1} - \lambda_i| \) for \( i = 1, \ldots, r - 1 \), one should truncate the collection of eigenvectors to \( v_1, \ldots, v_r \). A potential issue with this rule (apart from the absence of an unambiguous spectral gap) is that depending on the dimension of the underlying manifold, the natural asymptotic behaviour of \( \lambda_k \) as a function of \( k \) given by Weyl’s law [45] is not necessarily linear. We therefore propose a new variant of the eigengap heuristic by taking this asymptotic into account. If our phase space is a \( d \)-dimensional Riemannian manifold, denote by \( N(\lambda) \) the number of eigenvalues of the Laplace-Beltrami operator \( \Delta \) on \( M \) no greater than \( \lambda \) (applying either homogeneous Neumann or Dirichlet boundary conditions). Weyl’s law states that \( \lim_{\lambda \to \infty} N(\lambda)/\lambda^{d/2} = -(2\pi)^{-d} \omega_d |M| \) where \( \omega_d \) is the volume of the \( d \)-dimensional unit ball and \( |M| \) is the volume of \( M \). Thus, ordering the eigenvalues of \( \Delta \) as \( 0 \geq \lambda_1 \geq \lambda_2 \geq \cdots \), we have \( \lambda_r \sim -C r^{2/d} \) for a constant \( C \) as \( r \to \infty \).

Laplace-type matrix/operator, Neumann boundary conditions. For example, a graph Laplacian, a Laplace–Beltrami operator, or the dynamic Laplacian\(^6\) [12,13] of Sections 5.1 and 5.3 with homogeneous Neumann boundary conditions. In this setting, the leading eigenvalue is \( \lambda_1 = 0 \) and fitting the asymptotic \( \lambda_r \sim -C r^{2/d} \) to this condition suggests plotting \( \lambda_r/(r - 1)^{2/d} \) vs. \( r \) for \( r = 2, \ldots \), and looking for the largest drops from \( r \) to \( r + 1 \); see Fig. 4.

Laplace-type operator, Dirichlet boundary conditions. For example, a Laplace–Beltrami operator or the dynamic Laplacian with Dirichlet boundary conditions. The leading eigenvalue \( \lambda_1 \) is strictly negative (and unknown) so we plot \( \lambda_r/r^{2/d} \) vs. \( r \) for \( r = 1, \ldots \), and look for the largest drops from \( r \) to \( r + 1 \).

\(^6\) which is not necessarily a Laplace–Beltrami operator [46].
Markov-type operator. For example, a matrix or operator arising from diffusion maps [15], the various transfer operator constructions in Table 1, or the normalised finite-time transfer operator \([7,37]\) of Section 5.2. In these examples \(\lambda_1 = 1\) and the rest of the spectrum is contained in the unit circle in the complex plane. Taking logs and retaining only the real part, we obtain a Laplace-type spectrum with Neumann boundary conditions. Thus, we plot \(n/(\log \lambda_r)r^{(r-1)}\) vs. \(r\) for \(r = 2, \ldots\), and look for the largest drops from \(r \to r + 1\). In the case of transfer operators arising from continuous-time flows, this logarithmic transformation can be made rigorous via the spectral mapping theorem, see [13,34].

The above heuristics are applied to spectra arising from both transfer operators and the dynamic Laplacian in Section 5.

4.2.2. A heuristic based on sparse vectors

Following the arguments of Section 4.1, for the \(p \times r\) matrix \(S\) produced by Algorithm 3.1, we create a cumulative minimum value quantity \(\text{Min}(S) := \sum_{j=1}^{r} -m_j = -\sum_{j=1}^{r} \text{min}_{1 \leq \ell < p} S_{j\ell}\). Denoting by \(S(\ell)\) the sparse array produced by Algorithm 3.1 with \(\ell\) input vectors, one may then plot \(\text{Min}(S(\ell))\) vs. \(\ell\) and select those \(\ell\) for which there has been a large drop from \(r - 1\); i.e. \(\text{Min}(S(r)) - \text{Min}(S(r-1))\) is negative. The rationale for this choice is that \(\text{Min}(S(\ell))\) is typically increasing with \(\ell\) because it is a sum of non-negative values. If there is a drop from \(r - 1\) to \(r\) it means that despite adding an extra term to the sum, the sum decreases, and thus overall the new \(m_j\) values (based on \(\ell\) input vectors) are smaller than the old \(m_j\) (based on \(r - 1\) input vectors), indicating that the new sparse basis \(s_1, \ldots, s_r\) has better separated the \(r\) features. Note that such a heuristic implicitly assumes that all \(r\) sparse vectors are reliable, and no sparse vector will be rejected from the collection. In Section 5.2 we will combine this heuristic with the reliability heuristic of Section 4.1 to suggest a way to simultaneously select both \(r\) (the number of input vectors), and \(k\) (the number of reliable sparse vectors), with \(k < r\) after some sparse vectors have been rejected.

4.3. Extraction of a (sub)-partition from a sparse basis output

Recall that we wish to separate the features encoded in the eigenvectors \(v_1, \ldots, v_r\) resulting from a spectral clustering. Algorithm 3.1 has already done most of the hard work by forming a sparse approximate basis of the space spanned by the eigenvectors. Indeed one would often be very satisfied with this separation procedure. A simple and effective way of summarising the separated features is to create a “superposition vector”.

Superposition vector. \(s := \min\{1, \sum_{j=1}^{r} \max(S_{j\ell}, 0)\}\). This vector combines membership likelihoods across features, and may be interpreted as the likelihood that the \(j\)th index belongs to some feature. In MATLAB, this is realised as \(\text{super} = \min(1, \sum_{\ell=1}^{p} \max(S_{\ell\ell}, 0), 2)\); superposition vectors are illustrated in Figs. 9, 13, 14, and 22.

If a hard separation is required, then a final step is to threshold the sparse vectors to return which indices are in or out of a vector. If there is a two-dimensional phase space associated to the sparse vectors, then one may apply by inspection a manual threshold \(\tau\) so that feature \(j\) consists of grid cells or nodes with index \(i\) satisfying \(S_{ji} > \tau\).

For situations where visualisation is not possible, the main thing we desire is to ensure that distinct features are disjoint. We consider three algorithms. Algorithm 4.1 applies the minimum threshold \(\tau_{pu}\) to ensure that after thresholding, the columns \(S_j\) form a sub-partition of unity; that is \(\sum_{j=1}^{r} S_j \leq 1\) for each \(i = 1, \ldots, p\). Algorithm 4.2 selects the minimum threshold \(\tau_{dp}\) to ensure that after thresholding, the \(j\) have disjoint support. In each case, the principle of maximum likelihood is then applied to produce a feature vector \(a \in \{0, \ldots, r\}^p\) with \(a_i = j\) if feature \(i\) belongs to feature \(j\) for \(i = 1, \ldots, p\) and \(j = 1, \ldots, r\), and \(a_i = 0\) if feature \(i\) is unassigned. A third algorithm, described at the end of this section, simply applies the above maximum likelihood procedure without thresholding. Define a thresholding transformation \(H_{\mu} : \mathbb{R} \rightarrow \mathbb{R}\) by \(H_{\mu}(z) = z\) if \(z > \mu\) and \(H_{\mu}(z) = 0\) otherwise; \(H_{\mu}\) is applied to vectors elementwise.

**Algorithm 4.1** (Input: vectors \(s_1, \ldots, s_r \subset \mathbb{R}^p\) produced by Algorithm 3.1; output: thresholded \(s_1, \ldots, s_r\) vectors and feature vector \(a \in \{0, \ldots, r\}^p\).

1. For \(j = 1, \ldots, r\), set \(s_j \rightarrow \max(s_j, 0)\) to make the vectors nonnegative.
2. For each row \(i = 1, \ldots, p\), let \(s_{i1}, \ldots, s_{ir}\) be the values of \(S_{1i}, \ldots, S_{ri}\) in decreasing order.
3. Set the threshold \(\tau_{pu} := \max_{1 \leq \ell \leq p, 1 \leq j \leq r} \max(s_{j\ell})\). For each \(i = 1, \ldots, r\), hard threshold \(s_j \rightarrow H_{\tau_{pu}}(s_j)\).
4. For each \(i = 1, \ldots, p\), let \(j^* = \arg \max_{1 \leq j \leq r} s_{ij}\) (resolving ties arbitrarily). If \(s_{ij^*} > 0\), set \(a_i = j^*\), otherwise set \(a_i = 0\).

**Algorithm 4.2** (Input: vectors \(s_1, \ldots, s_r \subset \mathbb{R}^p\) produced by Algorithm 3.1; output: thresholded \(s_1, \ldots, s_r\) vectors and feature vector \(a \in \{0, \ldots, r\}^p\).

1. For \(j = 1, \ldots, r\), set \(s_j \rightarrow \max(s_j, 0)\) to make the vectors nonnegative.
2. For each row \(i = 1, \ldots, p\), let \(s_{i1}, \ldots, s_{ir}\) be the values of \(S_{1i}, \ldots, S_{ri}\) in decreasing order.
3. Set the threshold \(\tau_{dp} := \max_{1 \leq \ell \leq p} \max(s_{\ell2})\), and for each \(j = 1, \ldots, r\) hard threshold \(s_j \rightarrow H_{\tau_{dp}}(s_j)\).
4. For each \(i = 1, \ldots, p\), \(j = 1, \ldots, r\), if any \(s_{ij} > 0\) for \(j = 1, \ldots, r\), set \(a_i = j\), otherwise, set \(a_i = 0\).

Because a disjoint partition is a stronger requirement than a partition of unity, one has \(\tau_{dp} \geq \tau_{pu}\). If \(\tau_{dp} \geq 0.5\), then \(\tau_{dp} = \tau_{pu}\). We illustrate both algorithms in Fig. 5, where three synthetic sparse vectors \(s_1, s_2, s_3 \in \mathbb{R}^{5000}\) are plotted as dotted curves. These vectors have overlapping supports, and \(s_1 + s_2 > 1\) on part of the domain. The post-thresholding vectors are shown as solid curves in Fig. 5.
The likelihood ever = \( \mu \).

Appendix orthonormal weighted weighted obtained in horizontal resulting 5.

[7,18]

We define a weighted inner product \( \langle v, w \rangle_v := \sum_{i=1}^p v_i w_i \) denote the corresponding norm by \( \| \cdot \|_{\ell_2,v} \), and denote a weighted Frobenius norm by \( \| A \|_{F,v} := \sqrt{\sum_{i=1}^p \sum_{j=1}^n v_i A_{ij}^2} \). Let \( \| \cdot \|_{\ell_2,v} \) denote a \( \ell_1,v \) matrix norm by \( \| A \|_{1,v} := \sum_{i=1}^p \sum_{j=1}^n v_i |A_{ij}| \). Let \( V \in \mathbb{R}^{p \times r} \) be a matrix whose columns are pairwise orthonormal in the inner product \( \langle \cdot, \cdot \rangle_v \). Problem (1) can be generalised to

\[
\arg\min_{S \in \mathbb{R}^{p \times r}} \frac{1}{2} \| V - SR \|^2_F + \mu \| S \|_{1,v}.
\]

Denoting by \( D_v \) the diagonal matrix with \( v \) on the diagonal, we have that \( \| A \|_{1,v} = \| D_v A \|_{1,1} \) and \( \| A \|_{F,v} = \| D_v^{1/2} A \|_F \). Substituting \( V' = D_v^{1/2} V \) and \( S' = D_v^{1/2} S \), problem (2) is equivalent to

\[
\arg\min_{S \in \mathbb{R}^{p \times r}} \frac{1}{2} \| V' - S'R \|^2_F + \mu \| D_v^{1/2} S' \|_{1,1}.
\]

We generalise the soft threshold function \( C_\mu \) to \( C_{\mu,v} : \mathbb{R}^p \to \mathbb{R}^p \), defined by \( [C_{\mu,v}(v)]_i = C_{\mu v_i}(v_i) \). We prove in Appendix B that Problem (3) can be solved using Algorithm 3.1 with \( V \) replaced with \( V' \), \( \mu = 0.99/\sqrt{p} \) replaced with \( \mu = 0.99/\sqrt{p} \) in Step 1, \( C_\mu \) replaced with \( C_{\mu,v} \) in Step 3, and Step 6 replaced with \( S_j \to \text{sign}(v^T S_j) S_j \). After running

4.4. Weighting entries of the data vectors

In certain situations, one may wish to apply a weight vector \( 0 < v \in \mathbb{R}^p \) to the calculation of the objective in (1). The weight vector \( v \) allows one to emphasise (or deemphasise) specific indices \( i = 1, \ldots, p \). One situation where this may be appropriate is if one is constructing a transfer operator from a grid with differently sized grid cells; larger cells can be weighted according to their area or volume. For example when computing finite-time coherent sets of compressible flows as in [7,18], one could let \( v_i = \mu(B_i) \) (the reference measure \( \mu \) of the \( i \)-th box \( B_i \); see [7]) or \( v_i = a_i \) (the reference weight assigned to the \( i \)-th basis function; see Section 3.2.1, p1901 [18]).

We define a weighted inner product \( \langle v, w \rangle_v := \sum_{i=1}^p v_i w_i \) denote the corresponding norm by \( \| \cdot \|_{\ell_2,v} \), and denote a weighted Frobenius norm by \( \| A \|_{F,v} := \sqrt{\sum_{i=1}^p \sum_{j=1}^n v_i A_{ij}^2} \). Let \( \| \cdot \|_{\ell_2,v} \) denote a \( \ell_1,v \) matrix norm by \( \| A \|_{1,v} := \sum_{i=1}^p \sum_{j=1}^n v_i |A_{ij}| \). Let \( V \in \mathbb{R}^{p \times r} \) be a matrix whose columns are pairwise orthonormal in the inner product \( \langle \cdot, \cdot \rangle_v \). Problem (1) can be generalised to

\[
\arg\min_{S \in \mathbb{R}^{p \times r}} \frac{1}{2} \| V - SR \|^2_F + \mu \| S \|_{1,v}.
\]

Denoting by \( D_v \) the diagonal matrix with \( v \) on the diagonal, we have that \( \| A \|_{1,v} = \| D_v A \|_{1,1} \) and \( \| A \|_{F,v} = \| D_v^{1/2} A \|_F \). Substituting \( V' = D_v^{1/2} V \) and \( S' = D_v^{1/2} S \), problem (2) is equivalent to

\[
\arg\min_{S \in \mathbb{R}^{p \times r}} \frac{1}{2} \| V' - S'R \|^2_F + \mu \| D_v^{1/2} S' \|_{1,1}.
\]

We generalise the soft threshold function \( C_\mu \) to \( C_{\mu,v} : \mathbb{R}^p \to \mathbb{R}^p \), defined by \( [C_{\mu,v}(v)]_i = C_{\mu v_i}(v_i) \). We prove in Appendix B that Problem (3) can be solved using Algorithm 3.1 with \( V \) replaced with \( V' \), \( \mu = 0.99/\sqrt{p} \) replaced with \( \mu = 0.99/\sqrt{p} \) in Step 1, \( C_\mu \) replaced with \( C_{\mu,v} \) in Step 3, and Step 6 replaced with \( S_j \to \text{sign}(v^T S_j) S_j \). After running
Algorithm 3.1, set $S = D^{-1/2} S'$ to obtain a solution for Problem (2). The parameter $\mu$ should be chosen less than $1/\sqrt{\|V\|_1}$ to ensure $C_{\mu,v}(D^{1/2} v)$ is nonzero for every $v \in \mathbb{R}^p$ with $\|v\|_{2,v} = 1$; taking $\mu \geq 1/\sqrt{\|v\|_1}$ will result in $C_{\mu,v}(D^{1/2} v)$ being the zero vector when $v$ is the constant $\ell_{2,v}$-unit vector.

5. Examples

We perform numerical experiments on three systems: the Bickley jet, a turbulence simulation obtained from the Navier-Stokes equations, and North Atlantic ocean currents derived from satellite altimetry. We use the transfer operator (Construction 5 in Table 1) and the dynamic Laplacian (Construction 6 in Table 1) to detect coherent sets.

We can quantify the goodness of fit and sparsity of our solutions using the corresponding parts of the objective (1) being minimised.

- **$\ell_2$ subspace error:** For subspace approximation, note that for $p \times r$ orthogonal $V$, one has $\|V\|_2^2 = r$. Thus, reporting $\frac{1}{r} \|V - SR\|_2^2$ quantifies the subspace approximation error relative to the “zero” estimate $S = 0$; this value should lie between 0 and 1 (lower value means better approximation).

- **Absolute sparsity:** We report the absolute sparsity as $\|S\|_{0,1}/(pr)$, where $\|S\|_{0,1} := \sum_{i=1}^p \|S_i\|_0$. The absolute sparsity is the proportion of elements of $S$ that are nonzero, and therefore lies between zero and one (lower value means sparser).

- **Relative sparsity:** We report the relative ($\ell_1$) sparsity $\|S\|_{1,1}/\|V\|_{1,1}$. The $\| \cdot \|_{1,1}$ norm is used as the sparsity penalty term in Algorithm 3.1. The relative sparsity should lie between 0 and 1 (lower value means sparser).

In Sections 5.1 and 5.3 all time integrations are carried out using an explicit Runge-Kutta method of order (4,5) with adaptive step size, as implemented by MATLAB’s ode45 integrator, using the absolute and relative error tolerance $10^{-3}$. The methodologies listed in Table 1 apply to any finite dimension, we restrict ourselves to 2-dimensional examples to present our results with greater transparency.

5.1. Bickley jet

The Bickley jet is an idealised system introduced as a model of “banded chaos” by del-Castillo-Negrete and Morrison [47], which has become a popular test case for coherent structure detection methods [2,16-18,40,48,49]. The Bickley jet is a Hamiltonian system modeling a meandering jet with vortices on either side. Its stream function is

$$\psi(x, y, t) = -U_0 L_0 \tanh(y/L_0) + \sum_{i=1}^{3} A_i U_0 L_0 \sech^2(y/L) \cos(k_i(x - c_i t)).$$  \hfill (4)

We use parameter values $U_0 = 62.66$ m/s, $L_0 = 1770$ km, $A_1 = 0.0075$, $A_2 = 0.15$, $A_3 = 0.3$, $c_1 = 0.1446U_0$, $c_2 = 0.205U_0$, $c_3 = 0.461U_0$, $k_1 = 2/\tau_e$, $k_2 = 4/\tau_e$, $k_3 = 6/\tau_e$, $\tau_e = 6371$ km as in [49]. We consider the associated flow on the initial domain $\mathcal{M} = [0, 20] \times [-3, 3]$ periodic in the $x$-direction, over the time interval $[0, 40]$ days. We calculate the dynamic Laplacian using the “adaptive transfer operator method” of [18] on a Delaunay triangulation (alpha complex with $\alpha = 0.06$) of $\mathcal{M}$ with nodes on an initial grid of 600 $\times$ 180 points, on the time set $\mathcal{T} = [0, 40]$. There are clear eigengaps after the second and eighth eigenvalues, in both the unnormalised and Weyl-normalised plots (Fig. 4). These two eigengaps characterise two natural spatiotemporal scales for the Bickley dynamics.

We applied Algorithm 3.1 with $r = 2$ and $r = 8$ to the eigenvectors of the dynamic Laplacian. With $r = 2$, the two eigenvectors are the leading (constant) eigenvector and a second non-constant eigenvector. Algorithm 3.1 produces two sparse vectors whose supports separate the domain into upper and lower pieces; see Fig. 6a. The six vortices are also highlighted within the supports of the two sparse vectors. The span of these two sparse vectors is a reasonable approximation of the span of the two leading eigenvectors (subspace error of 11.8%). The sparse vectors have an absolute sparsity of 45.2% and a relative sparsity of 65.0%.

We apply Algorithms 4.1 and 4.2 to obtain a partition of the domain $A_1 \cup A_2 \subset \mathcal{M}$. Since the sparse vectors in Fig. 6a have disjoint support, both algorithms return the same sub-partition, formed by the supports of the two sparse vectors; see Fig. 6b. Note that the central jet between the upper and lower halves is also visible (zeroed as dark blue), using only information from the first nontrivial dynamic Laplacian eigenvector. k-means clustering creates a partition of the entire phase space; see Fig. 6c.

With $r = 8$, Algorithm 3.1 produces the sparse vectors in Fig. 3 (right column). The subspace error is low at 6.4%, as are the absolute sparsity and relative sparsity at 14.1% and 44.6%, respectively. The low absolute sparsity indicates that we have effectively separated the domain into disjoint features, with little overlap between the features.

We can again use either Algorithm 4.1 or Algorithm 4.2 to obtain a partition of the domain; see Figs. 7a and 7b respectively. Algorithm 4.1 produces a sub-partition of the domain consisting of almost the entire domain, while Algorithm 4.2 produces a sub-partition of the domain with a larger gap between the upper and lower regions corresponding to a central jet. Both of these results are similar to the k-means clustering result with $k = 8$ clusters, as shown in Fig. 7c.

Remark 5.1. The k-means algorithm can be viewed as a coarse type of sparse eigenbasis approximation in the following way. Apply k-means to the vectors $v_1, \ldots, v_r$ arising from a spectral clustering, requesting $r$ clusters. For $j = 1, \ldots, r$ define vectors
\[ \hat{s}_j \in \mathbb{R}^p \text{ as } \hat{s}_{ij} = 1 \text{ if index } i \text{ is contained in cluster } j, \text{ and } \hat{s}_{ij} = 0 \text{ otherwise.} \]

In the “ideal” situation where data points are very clearly separated into distinct clusters, the eigenvectors \( v_1, \ldots, v_r \) will be approximately constant, and the (sparse) vectors \( \hat{s}_1, \ldots, \hat{s}_r \) will approximately span the same space. By solving a Procrustes problem, one can also infer an orthogonal rotation implicitly performed by k-means to approximately rotate the vectors \( v_1, \ldots, v_r \) to the vectors \( \hat{s}_1, \ldots, \hat{s}_r \). This relationship with sparse eigenbasis approximation can help explain why k-means can be very effective in some (closer to “ideal”) cases where one expects to classify every data point (i.e. one partitions the dataset into features that exhaust the dataset).

5.2. Turbulent flow

We consider a numerical solution to the Navier–Stokes equations in two dimensions with random-in-phase forcing, introduced in [50]. Specifically, we consider the velocity field \( u : \mathbb{T}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2 \) which solves the equations

\[
\begin{align*}
\partial_t u + u \cdot \nabla u &= -\nabla p + \nu \Delta u + f \\
\nabla \cdot u &= 0
\end{align*}
\]

where the toral domain \( \mathbb{T}^2 = [0, 2\pi] \times [0, 2\pi] \); see [50] for details. We use the Ulam approximation \( P \) of the transfer operator \( P \) constructed on a \( 256 \times 256 \) grid, for a flow time of 50 time units, exactly as in [49]. Using the method of [7,37], we compute singular vectors of the matrix \( L \) (in the notation of [7,37], which is a slight reweighting of the stochastic matrix \( P \)). The second left and right singular vectors of \( L \) are shown in Fig. 8. The left and right images in Fig. 8 may be compared directly with [49, Figures 9(g) and 11(a)], respectively, where in the latter figures, only a binary plot of positive and negative values was displayed. In Fig. 8 there are three highlighted coherent regions: two with extreme positive values (red), and one with extreme negative values (dark blue). When applying Algorithm 3.1 to the first two left singular vectors (the leading left singular vector is constant and the second left singular vector is shown in Fig. 8 (left)), these three features are separated and highlighted in red; see Fig. 9. The sparse basis vectors shown in Fig. 9 are considerably more informative than either (i) a simple partition based on the sign of the singular vectors as in [49], or (ii) the result of a hard partitioning method such as k-means. The sparsity quantifiers in the case of \( r = 2 \) are 31.6% (subspace error), 38.4% (absolute sparsity) and 54.6% (relative sparsity).

Turbulence is often cited as a canonical example of dynamics operating over a broad range of spatial scales. This fact is backed up by the lack of a clear eigengap for the transfer operator singular value spectrum; see Fig. 10 (left). The Weyl
Fig. 8. Second left singular vector (left); second right singular vector (right).

Fig. 9. Superposition vector $\mathbf{s}$ (see Section 4.3 for the definition) of the two sparse vectors generated for the turbulence flow using two singular vectors ($r = 2$) (left). Push forward of the left image $\mathbf{s}$ by $L$ (right).

Fig. 10. Plot of $\log \lambda_r$ vs $r$ (left) and $\log \frac{\lambda_r}{r - 1}$ vs $r$ (right) for the turbulence flow.

Rescaling in Fig. 10 (right) shows some minor drops, which may provide weak eigengap information. In situations where there is no clear eigengap, we propose to use information from the sparse vectors output by Algorithm 3.1 to decide how many sparse vectors contain “reliable” coherent features. The upper envelope (dotted black) of Fig. 11 shows $\text{Min}(S(r))$ vs. $r$. There are several “drops” in this envelope, with the three largest drops at $r = 5, 30,$ and $36$. In this turbulence example, we illustrate a refinement of this choice of $r$, simultaneously taking into account the “reliability” of the sparse vectors.
5.2.1. A sparse vector heuristic for simultaneously selecting the number of input vectors \( r \) and number of features \( k \)

For some maximum number of columns \( r_{\text{max}} \) and each \( r = 2, \ldots, r_{\text{max}} \), we apply Algorithm 3.1 to the first \( r \) columns of the eigenvector matrix \( V \), and denote the resulting arrays \( S^{(r)} \). For each \( r \) and \( k \leq r \), let \( \text{Min}(S^{(r)}, k) := \sum_{j=1}^{k} -\min_{1 \leq i \leq p} S_{ij}^{(r)} \) be the sum of the minimum values of the first \( k \) sparse vectors \( S^{(r)} \). In Fig. 11, for each \( k \leq r \), we plot \( \text{Min}(S^{(r)}, k) \) vs. \( r \) as a coloured line, for each \( 2 \leq r \leq r_{\text{max}} \). The \( k \)th coloured line shows the “performance” (as determined by the minimum value heuristic) of Algorithm 3.1 at separating \( k \) features from various \( r \) input vectors. The best performance for \( k \) features is achieved when a single coloured line attains a minimum of \( \text{Min}(S^{(r)}, k) \) across the various values of \( r \). For a given \( k \), these minimal \( r \) are denoted \( r_{\text{min}}(k) := \arg \min_{k \leq r \leq r_{\text{max}}} \text{Min}(S^{(r)}, k) \) (when multiple \( r \) indices give the same \( \text{Min}(S^{(r)}, k) \) value, we take \( r_{\text{min}}(k) \) as the smallest of these). These “optimal” \((k, r)\) combinations are shown as black diamonds on Fig. 11. Fig. 12 shows a plot of \( r_{\text{min}}(k) \) vs. \( k \).

Notice that \( r_{\text{min}}(k) \) takes on only certain \( r \) values. These common \( r \) values (e.g. \( r = 30, 38, \ldots \) in Fig. 11) could be interpreted as representing natural spatial scales of the dynamics because one expects the eigenvectors (or singular vectors)
Fig. 13. Superposition vector \( s \) of the 7 most reliable features at initial time (left) according to the minimum value reliability criterion \( (r = 30) \) and its forward-time image (right).

Fig. 14. Superposition vector \( s \) of the 21 most reliable features at initial time (left) according to the minimum value reliability criterion \( (r = 38) \) and its forward-time image (right).

input to Algorithm 3.1 to become more oscillatory as \( r \) is increased, highlighting increasingly smaller spatial features. One may select any of these optimal \((k, r)\) combinations. To be conservative, in our experiments we select \( k \) at the lower end of each “vertical strip” of black diamonds. If we increase \( k \) we often simply obtain more coherent sets. For this example, we select \((k, r) = (7, 30)\) and \((21, 38)\) for further analysis; the corresponding sparse superposition vectors and their forward-time images are shown in Figs. 13 and 14. From these two figures one already sees 7 (resp. 21) clearly highlighted coherent sets in red that remain coherent after the flow time of 40 time units. The span of the collection of sparse vectors in both the case of \( r = 30 \) and that of \( r = 38 \) produce reasonable approximations of the span of the corresponding leading eigenvectors with a subspace error of 10.4% (resp. 9.8%). The sparse vectors have an absolute sparsity of 7.7% (resp. 7.1%) and a relative sparsity of 26.8% (resp. 25.2%).

5.2.2. Extraction of a subpartition from sparse vector output

One could apply a single global threshold to our selected sparse vectors according to Algorithms 4.1 and 4.2. We found that because Fig. 13 is already a collection of vectors with mostly disjoint supports, both of these algorithms produced a very low threshold with little change to the sparse vectors. On the other hand, the sparse vectors that are combined in Fig. 14 have “overlaps” of large values in small parts of phase space, causing Algorithms 4.1 and 4.2 to produce very high thresholds close to 1. In place of these thresholding algorithms, one could easily apply a global threshold manually by inspecting Fig. 13 or 14; for example a threshold of around 0.7–0.75 would produce coherent sets with small boundary lengths relative to enclosed area at the initial and final times.
To end this section we apply thresholding according to the Cheeger ratio. Such a thresholding has been applied to specific single vectors [12,13,42,49], but because Algorithm 3.1 has automatically separated all features, we may apply this thresholding to either the superposition vector $s$ or to each individual vector sparse vector $s_j$, $j = 1, \ldots, k$. Given a vector (corresponding to a discrete representation of a real-valued function on a manifold $M$), we select a threshold by choosing the level set that minimises a quantity we call the scale-invariant Cheeger ratio. Let $\Gamma$ be a $d$-dimensional manifold and let $T: M \to M$ be a volume-preserving transformation representing our dynamics. For simplicity of presentation, we assume that one application of $T$ represents the nonlinear dynamics over the full finite-time interval and that we compute the coherence via the Cheeger ratio at only the initial and final time; see [12,13] for more detail and generalisations. Let $\Gamma \subset M$ be a $d - 1$ dimensional submanifold disconnecting $M$ into two pieces $M_1$, $M_2$. The scale-invariant Cheeger ratio is

$$h^d(\Gamma) : = \frac{\ell_{d-1}(\Gamma) + \ell_{d-1}(T(\Gamma))}{2\min(\ell_d(M_1), \ell_d(M_2))}^\frac{1}{2}.$$  

where $\ell_{d-1}$ and $\ell_d$ are $(d - 1)$- and $d$-dimensional volume measure, respectively. In this example we treat each sparse vector $s_j$ separately, and each $\Gamma$ will be selected as a level set of $s_j$, $j = 1, \ldots, k$. In particular we choose the level set value so that (6) minimised. In terms of numerics, having selected a $(k, r)$ combination, for each $1 \leq j \leq k$, we linearly interpolate $s_j$ on a fine regular grid on $M$ to obtain a function $u_j : M \to \mathbb{R}$. For threshold values $r$ ranging from $\min_{x} u_j(x)$ to $\max_{x} u_j(x)$ we calculate $h^d(\Gamma r)$, where $\Gamma r$ is the level set $\{x \in M : u_j(x) = r\}$, and select the $r$ yielding the minimum value. We use MATLAB’s contourc command to calculate level sets; isosurface is the corresponding command for three dimensions. This takes approximately two minutes for 21 interpolated sparse vectors in $\mathbb{R}^{65536}$ on an Intel Core i7-6700 processor with 16GB RAM. Note that in contrast to Algorithm 4.1 or 4.2, we calculate a separate threshold for each vector before applying a hard thresholding.

We apply this thresholding procedure to the $(k, r)$ combination (7.30) whose superposition of sparse vectors is shown in Fig. 13; the result is the partition in Fig. 15. All features obtained are highly coherent, though two features are perhaps overly small. The scale-invariant dynamic Cheeger ratio for these two small sets has a very flat minimum over a large range of thresholds and so these two sets could be further enlarged without changing appreciably the Cheeger ratio (and therefore the finite-time coherence).

We also apply this thresholding procedure to the $(k, r)$ combination (21,38), whose superposition vector $s$ is shown in Fig. 14. The resulting thresholded sets are displayed in Fig. 16, coloured according to their reliability (a reliability index of 1 is the most reliable (coherent)). We retain all the coherent sets identified in Fig. 15, and gain another 15 coherent sets. In this case, while some of the additional coherent sets are relatively small, many of these sets cannot be significantly increased without introducing filamentation; that is, they do not have flat minima for their Cheeger ratios.

### 5.2.3. Extraction of a partition with k-means

The use of sparse vectors via Algorithm 3.1 clearly outperforms k-means clustering of the singular vectors. As an example, we apply k-means to the $(k, r)$ combination (21,38), namely using 38 eigenvectors and requesting $k = 21$ clusters; see Fig. 17. We add one extra cluster to give k-means a chance to find the complement of 21 coherent sets as a 22nd coherent set. The k-means clustering correctly detects some coherent sets, notably many of those in Fig. 15, but also produces many more highly filamented sets, and misses several very coherent features visible in Fig. 16. To a large extent, k-means fails in difficult examples such as turbulence because k-means imposes a partition of the phase space into coherent sets, when a partition is not reasonable. The superposition of the sparse vectors in Fig. 14 highlights the coherent sets in a much more precise
Fig. 16. Partition \((k, r) = (21, 38)\) of phase space showing 21 coherent sets at initial- (left) and forward- time (right) obtained by individual thresholding.

Fig. 17. Initial and forward-time plot of the k-means partition of state space into 22 clusters \((r = 38)\).

way than k-means can, and we can further refine Fig. 14 by thresholding, as in Fig. 16. Increasing \(k\) causes k-means to add more filamented features, while decreasing \(k\) causes k-means to miss some of the coherent features in Figs. 14 and 16. We also experimented with the use of the k-means sum of squared errors (the objective that k-means attempts to minimise) to determine the number of clusters, but we failed to find any “kinks” to identify any values for \(k\), as described for example in [51].

5.3. North Atlantic surface currents

As our final example, we consider a real-world dataset, namely geostrophic AVISO [52] velocity fields. We consider the 90-day period 15 January 2015 to 15 April 2015; the velocity fields are updated every 2 hours. We consider a synthetic initial \(291 \times 224\) Mercator grid of tracers on the domain \(\mathcal{M} = [-65^\circ, -35^\circ] \times [30^\circ, 50^\circ]\). After removing trajectories initialised on land, we numerically integrate the remaining 56,205 trajectories, linearly interpolating the velocity field between the AVISO-provided velocity fields spaced one day apart. Using trajectory positions’ every 3 days, we compute the dynamic Laplacian using the “adaptive transfer operator approach” of [18] on alpha complexes\(^8\) with zero Neumann boundary conditions; see [18] for details on implementation.

The second eigenfunction highlights the Gulf of St Lawrence in the extreme northwest of the 15 January 2015 domain as a coherent set evolving distinctly from the rest of the domain (not shown), and the third eigenfunction separates the domain at the Gulf Stream; see the sharp red/green interface in Fig. 18. One may directly compare Fig. 18 (lower) with [53, Figure 9]. The latter figure shows a satellite-based map of sea surface temperature at 5 April 2015. The Gulf Stream is seen

---

\(^7\) In particular, derivatives of the velocity field or flow map need not be estimated.

\(^8\) With the lowest alpha such that the alpha complexes are connected.
as a very strong temperature gradient emanating from the east coast of the USA. There is a very close match with the strong red/green interface (and also the red/orange interface between -50 and -40 longitude) in the eigenvector in Fig. 18 (lower). This demonstrates that the strong temperature gradient of the surface ocean is highly correlated with coherent transport of water parcels on the surface.

The spectrum for this dataset, shown in Fig. 19 (left) reveals little separation of scales; the largest drops in the Weyl rescaling (Fig. 19; right) are at $r = 3$ and $r = 30$. Therefore, we again select values for $r$ and $k$ using the heuristic described in Fig. 19 (right).
in Section 5.2.1. We construct the cumulative minimum value plot (Fig. 20) and plot $r_{\text{min}}(k)$ in Fig. 21. Our initial choice of $r_{\text{max}} = 50$ as our maximum number of eigenvectors was possibly too small to identify many of the coherent sets, because $r_{\text{min}}(k) = 50$ for almost all values of $2 \leq k \leq 50$. Increasing $r_{\text{max}}$ to 100, the coloured curves for $k = 1, \ldots, 40$ all start to plateau before $r_{\text{max}} = 100$, and so we consider this $r_{\text{max}}$ sufficiently large. We only plot the coloured curves for $k = 1, \ldots, 40$, to simplify the figure.
As discussed in Section 5.2.1, we want to consider the first $k$ value in a run of $k$ values with the same $r_{\text{min}}(k)$. We take $r = 96, k = 36$. We repeated the experiment with various $k$ and $r_{\text{min}}(k)$ pairs with $r_{\text{min}}(k)$ around 77–81, but the resulting regions were similar to a subset of those for $r = 96, k = 36$. The span of these 96 sparse vectors produced an approximation of the original subspace with an error of 3.9%. The absolute sparsity is 3% and the relative sparsity is 19.1%. We plot the sum of the resulting sparse vectors in Fig. 22, together with their forward time image. We obtain many coherent features, as well as some partial coherent features at the boundary. A zoom of two of the sparse vectors highlighting two different eddies at 28 February 2015 is shown in Fig. 23. The black dots in Fig. 23 indicate trajectory locations used in the construction of the
Fig. 23. Zoom of two sparse vectors produced by Algorithm 3.1 with \((k, r) = (36, 96)\), plotted at 28 February 2015. The black dots are locations of trajectory data used to compute the dynamic Laplace operator.

Fig. 24. Initial (left) and forward-time (right) plot of the k-means partition of state space for the North Atlantic dataset into 37 clusters using \(r = 96\) eigenvectors.

dynamic Laplacian. Note that we resolve high-resolution estimates of the eddy shapes using the finite-element approach of [18] from lower-resolution trajectory data.

The eddy field in Fig. 22 can be compared directly with a backward finite-time Lyapunov exponent field in [53, Figure 10 (upper right)]. The FTLE field suggests a Gulf Stream feature as in Fig. 18 (lower) and some eddies of similar size as Fig. 22; in both cases, the FTLE field is not cleanly identifying these figures. Fig. 4 [54] shows fields of encounter volume, exact diffusivity, long-time diffusivity, and diffusive timescale at 11 January 2015 on a domain approximately \(–75\) to \(–45^\circ\) longitude by \(35–42.5^\circ\) latitude. These four (spatially smaller) fields can be directly compared with the corresponding part of the larger domain in Fig. 22 (upper). The three eddies highlighted in Fig. 22 (upper) in the subdomain of [54, Figure 4] appear in the latter figure to varying extents, but not as sharply as in Fig. 22. Other features one might find in [54, Figure 4] do not appear in Fig. 22 (upper), possibly because they are less coherent than the many other features highlighted. The partition produced by k-means again proved ineffective; see Fig. 24.

6. Discussion

We proposed a simple, powerful post-processing of the output of spectral clustering to separate individual features or clusters. Spectral clustering outputs a truncated eigenspectrum of some Markov- or Laplace-type matrix or operator, with associated eigenbasis. Features or clusters are encoded in the vectors of this eigenbasis, but may be mixed together within several eigenvectors. We adapted sparse PCA methodologies to identify an approximate sparse basis for the given eigenbasis.
The use of sparsity (i) efficiently and reliably separates individual features and (ii) produces a natural cutoff to identify those data points that do not belong to any cluster.

Spectral clustering often relies on the existence of an eigengap to determine (i) the number of clusters or features and (ii) where to truncate the eigenspectrum. In many situations, there is no clear eigengap. We proposed a refinement of the standard eigengap procedure in Section 4.2.1 that takes into account the spectral structure of Markov and Laplace-type operators, to correctly scale the eigenspectrum to allow fair comparison of gaps along the spectrum. We further introduced vector-based heuristics in Section 4.2.2 based on the output of our sparse eigenbasis approximation (SEBA) algorithm to suggest a good choice of the size of eigenbasis r. As described in Section 4.2, these vector-based heuristics determine natural spatial scales, while the eigenvalues at eigengaps determine natural temporal scales. A rejection criterion to help determine suitable numbers of features or clusters k was proposed in Section 4.1. We described a method for simultaneously selecting k and r in Section 5.2.1.

We believe that our techniques are a useful toolkit for general spectral clustering problems. One specific motivation was the identification of almost-invariant and coherent sets in time-dependent nonlinear dynamical systems; in particular, simply and reliably extracting many such sets and determining corresponding natural spatial and temporal scales. We applied our techniques to three example nonlinear systems.

Firstly, the Bickley jet, which possesses a modest number of strongly coherent sets that were easily separated and extracted using eigenfunctions of the dynamic Laplace operator. In this example, other post-processing techniques such as k-means clustering also perform very well. Secondly, turbulent, randomly forced Navier-Stokes flow on a 2-torus, with coherent sets at multiple spatial scales and with no clear temporal scale separation. We identified good combinations for the “number of coherent sets” and the “eigenbasis size” and produced well-separated coherent sets through the superposition vector $s$. In contrast, post-processing methods such as k-means performed poorly, as expected for the situation where much of the phase space should not be classified as coherent. Thirdly, we analysed trajectory data in the North Atlantic Ocean derived from satellite altimetry. We found a large separation of natural spatial scales with a jump from scales commensurate with the entire ocean basin, to features around one degree of longitude across. Leading eigenvectors identified the Gulf of St Lawrence as a coherent set strongly dynamically disconnected from the rest of the domain, and the Gulf Stream as a persistent large-scale coherent transport barrier. No intermediate scales were identified; the next natural spatial scale occurred at the scale of mesoscale eddies, at which we identified numerous eddies.

Acknowledgements

The authors would like to thank Anthony J. Roberts for suggesting sparse PCA as a possibly useful methodology at a workshop in Blackheath in 2017, Mohammad Farazmand and Alireza Hadjighasem for providing the transition matrix used in Section 5.2, and Irina Rypina for providing the AVISO trajectory data used in Section 5.3. GF is partially supported by an ARC Discovery Project, CPR is supported by an Australian Government Research Training Program Scholarship, and KS is supported by an ARC Discovery Project.

Appendix A. Code

The code in this appendix is available as supplementary material for the online version of this article.
A.1. MATLAB code for Algorithm 3.1

```matlab
function S=SEBA(V,Rinit)

%V is pxr matrix (r vectors of length p as columns, assumed orthonormal)
%S is pxr matrix with columns approximately spanning the column space of V
%Rinit is optional orthonormal rxr matrix

maxiter=5000;
[V,~]=qr(V,0);  %Enforce orthonormality
[p,r]=size(V);
mu=0.99/sqrt(p);

S=zeros(size(V));
%Perturb near-constant vectors
for j=1:r
    if max(V(:,j))-min(V(:,j))<1e-14
        V(:,j)=V(:,j)+(rand(p,1)-1/2)*1e-12;
    end
end

%Initialise rotation
if margin==1
    Rnew=spye(r);
else
    [P,~,Q]=svd(Rinit,0);
    Rinit=P*Q';  %Ensure orthonormality of Rinit
    Rnew=Rinit;
end

R=0;

iter=0;
while norm(Rnew-R)>1e-14 && iter<maxiter
    iter=iter+1;
    R=Rnew;
    Z=V*R';
    %Threshold to solve sparse approximation problem
    for i=1:r
        S(:,i)=sign(Z(:,i)).*max(abs(Z(:,i))-mu,0);
        S(:,i)=S(:,i)/norm(S(:,i));
    end
    %Polar decomposition to solve Procrustes problem
    [P,~,Q]=svd(S'*V,0);
    Rnew=P*Q';
end

%Choose correct parity of vectors and scale so largest value is 1
for i=1:r
    S(:,i)=S(:,i)*sign(sum(S(:,i)));
    S(:,i)=S(:,i)/max(S(:,i));
end

%Sort so that most reliable vectors appear first
 [~, I] = sort(min(S), 'descend');
 S = S(:, I);
```
A.2. MATLAB code for Algorithm 4.1

```matlab
function [S,A,taupu] = subpartition_unity(S)
%SUBPARTITIONUNITY Apply hard thresholding to obtain a sub-partition of
%unity from the columns of S. Also create a maximum-likelihood
%sub-partition.

S=max(S,0); % Take non-negative part
S_descend=sort(S,2,'descend'); % Sort each row in descending order
S_sum=cumsum(S_descend,2); % Must be <=1 for partition of unity
taupu=max([S_descend(S_sum>1);0]); % Largest element where row sum > 1
S(S<=taupu)=0; % Apply hard thresholding

[M,A]=max(S,[],2); % Find column with largest element, by row
A(M==0)=0; % Suppress zero rows
```

A.3. MATLAB code for Algorithm 4.2

```matlab
function [S,A,taudp] = disjoint_support(S)
%DJSUPPORT Apply hard thresholding to obtain disjoint supports for
%the columns of S. Also create a maximum-likelihood sub-partition.

S=max(S,0); % Take non-negative part of S
S_descend=sort(S,2,'descend'); % Sort to find largest element in each row
taudp=max(S_descend(:,2)); % Take largest non-leading element
S(S<=taudp)=0; % Apply hard thresholding

[M,A]=max(S,[],2); % Find column with largest element, by row
A(M==0)=0; % Suppress zero rows
```

A.4. MATLAB code for maximum likelihood

```matlab
function [S,A] = max_likelihood(S)
%MAXIMUMLIKELIHOOD Apply maximum likelihood to obtain disjoint supports
%from the columns of S, resolving ties arbitrarily. Also create the
%corresponding sub-partition.

[M,A]=max(S,[],2); % Find column with largest element, by row
A(M<=0)=0; % Suppress zero rows
S(:)=0;
[r=size(S,2)];
for i=1:r
    S(A==i,i)=M(A==i); % Set original values for non-zero elements.
end
```
A.5. MATLAB code for producing a “natural spatial scales” plot of \((k, r)\) combinations

```matlab
function SpatialScalesPlot(V)
    \%V is pxR matrix (r vectors of length p as columns, assumed orthonormal)
    R=size(V,2);
    column_minima=nan(R-1,R);
    for i=1:R-1
        S=SEBA(V(:,1:i+1)); \% Column minima are in descending order
        column_minima(i,1:i+1)=-min(S);
    end
    minimum_sums=cumsum(column_minima,2);
    [~,best_r]=min(minimum_sums);
    cla reset;
    plot(best_r(1:R)+1,(1:R),'o','Color','k','MarkerSize',4,...
         'MarkerFaceColor','k'); \% Plot best r for each k=1,..,r
```

A.6. MATLAB code for producing a “stacked” sparse vector minimum value plot

Note that the first seven lines coincide with the code in Section A.5, so these lines need not be recomputed if one is creating both plots.

```matlab
function MinValStackedPlot(V,kmax)
    \%V is pxR matrix (r vectors of length p as columns, assumed orthonormal)
    \%kmax is the number of lines to output, assumed <= R
    R=size(V,2);
    column_minima=nan(R-1,R);
    for i=1:R-1
        S=SEBA(V(:,1:i+1)); \% Column minima are in descending order
        column_minima(i,1:i+1)=-min(S);
    end
    minimum_sums=cumsum(column_minima,2);
    [~,best_r]=min(minimum_sums);
    cla reset; hold on
    for j=1:kmax
        iStart=max(j-1,1);
        plot(iStart+1:R,minimum_sums(iStart:R-1,j),'o-','Linewidth',1.5);
    end
    plot(best_r(1:kmax)+1,best_r(1:kmax),'kd','MarkerSize',4,...
         'MarkerFaceColor','k'); \% Plot best r for each k
    plot([2,2:kmax],minimum_sums([1,R:R*(kmax-1)]),'k:','Linewidth',1.5);
    hold off
```

Appendix B. Proof that the expression (2) can be minimised as outlined in Section 4.4

We wish to solve problem (3), which is equivalent to problem (2). We can apply the same alternating optimisation scheme as in Section 3. With fixed \(S'\), problem (3) is the same as problem (1) with \(V\) replaced with \(V'\) and \(S\) replaced with \(S'\). This can be solved using Step 3 of Algorithm 3.1.
With fixed $R$, we use the property of the (weighted) Frobenius norm that $\|A\|_{F,\nu} = \|AR\|_{F,\nu}$ for every matrix $A \in \mathbb{R}^{P \times T}$ and orthogonal $R \in \mathbb{R}^{T \times T}$. To see this, note that $\|A\|_{F,\nu}^2 = \text{Trace}(A^T D_{\nu} A)$, so

$$\|A\|_{F,\nu}^2 = \text{Trace}(A^T D_{\nu} A(RR^T)) = \text{Trace}(R^T A^T D_{\nu} AR) = \|AR\|_{F,\nu}^2.$$ 

This lets us reformulate problem (3), for fixed $R$, as

$$\arg\min_{S \in \mathbb{R}^{P \times T}} \frac{1}{2} \|VR^T - S'\|_F^2 + \mu \|D^2_
u S\|_{1,1} = \arg\min_{S \in \mathbb{R}^{P \times T}} \sum_{j=1}^T \frac{1}{2} \|Z_j - S_j\|_2^2 + \mu \|D^2_
u S\|_{1,1},$$ 

(7)

where $Z_j$ is the $j$th column of $VR^T$ and $S_j$ is the $j$th column of $S$. Since $\|Z_j\|_2 = \|S_j\|_2 = 1$, we have $\|Z_j - S_j\|_2 = \|Z_j\|_2 + \|S_j\|_2 - 2\|Z_j - S_j\|_2 = 2 - 2\|Z_j - S_j\|_2$, so problem (7) is equivalent to $\arg\max_{S \in \mathbb{R}^{P \times T}} \|Z_j - S_j\|_2 - \mu \|D^2_
u S\|_{1,1}$. By a slight modification of Eqs. (5)–(7) in [30], taking $p = 1$, $A = Z_j^T$ and $x = 1$ and replacing $\gamma \|z\|_1$ with $\sum_{i=1}^p \nu_i^{1/2} |z_i|$, it can be shown that our problem (7) is solved for every $j$ by setting $S_j' = C_{\mu,\nu}(Z_j^T)/(\|D^2_{\nu} Z_j^T\|_2^2)$, exactly as in Step 4 of Algorithm 3.1 with $V$ replaced with $V'$ and $C_{\mu,\nu}$ replaced with $C_{\mu,\nu}$.

Lastly, we prove that $C_{\mu,\nu}(D^2_{\nu} V^2)$ is nonzero for every $v \in \mathbb{R}^P$ with $\|z\|_{2,2} = 1$, if and only if $\mu < 1/\sqrt{\|v\|_1}$. We start by noting that if $\mu < 1/\sqrt{\|v\|_1}$ then $C_{\mu,\nu}(D^2_{\nu} V^2)$ is nonzero for every $v \in \mathbb{R}^P$ with $\|z\|_{2,2} = 1$. To see this, choose any such $z$, then

$$1 = \|z\|_{2,2} = \sum_{i=1}^p v_i z_i^2 \leq \sum_{i=1}^p v_i \max_{1 \leq j \leq p} Z_j = \sqrt{\|v\|_1} \max_{1 \leq j \leq p} Z_j,$$

That is, there is some index $j$ with $Z_j \geq 1/\sqrt{\|v\|_1}$. Recalling that $v_i > 0$ for every $i = 1, \ldots, p$, $\mu < 1/\sqrt{\|v\|_1}$ implies $|v_i^{1/2} z_j| - \mu v_i^{1/2} > 0$, i.e. the $j$th element of $C_{\mu,\nu}(D^2_{\nu} V^2)$ is nonzero as required. To prove the reverse direction, note that the vector $c \in \mathbb{R}^P$ with every element equal to $1/\sqrt{\|v\|_1}$ has $\|c\|_{2,2} = 1$. But for $\mu \geq 1/\sqrt{\|v\|_1}$, $|v_i^{1/2} c_i| - \mu v_i^{1/2} \leq 0$ for every $i = 1, \ldots, p$, so $C_{\mu,\nu}(D^2_{\nu} c) = 0$, and the proof is complete.

Supplementary material


References


