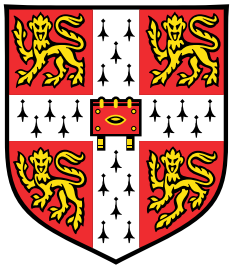
MPhil DIS Project 24

Learning Dominant Physical Processes with Data-Driven Balance Models



CRSiD: tmb76

Department of Physics

University of Cambridge

Submitted in partial fulfilment of the requirements of the MPhil degree in Data Intensive Science

Hughes Hall

January 24, 2025

Acknowledgements

The author would like to first and foremost thank Dr. Richard Kerswell for his guidance and support throughout this project. His advice was short and to the point, and it always managed to give a clear new perspective on the project, which was extremely helpful. The author would also like to thank Miguel Beneitez for his help in getting to grips with the Elasto-Inertial Turbulence data, and for taking the time to get it ready.

Contents

Chapter 1

Introduction

For many problems in engineering and physical sciences, equations involve a large number of terms and complex differential equations. Simulating them can be computationally expensive or unnecessarily so, as there exist regions where only a subset of the terms dominate the equation. In such cases, one can simplify the equations to a balance between these dominant terms, and simulate the system with sufficient accuracy and relatively lower computational cost [7]. This method, known as dominant balance or scale analysis, has been a powerful tool in physics.

Though extremely useful, dominant balance usually requires expertise and is mostly done by hand through time-consuming proofs. This report discusses and verifies a novel approach, developped by Callaham et al. (2021) [5], which explores using data from a physical system and machine learning methods to identify dominant balances algorithmically.

Like for any research, one of the key steps of the scientific method is reproducibility. Results must be reproducible by others, ensuring that the same conclusions can be drawn multiple times. Otherwise, it may be considered erroneous, or simply a random occurence. This project therefore has for a core aim to evaluate the reproducibility of the results of the Callaham et al. (2021) [5] paper and to test the robustness of their method. First discussing the research surrounding the paper, the report will then be delving into the details of the method. Reproducing the results for one of the case studies covered in the paper with the use of alternative code will then be explored, additionally exploring the use of algorithms other than the method's chosen one. Finally, the method will be used on a new dataset, hopefully shedding some light on a flow called Elasto-Inertial Turbulence.

Chapter 2

Background

As aforementionned, dominant balance or scale analysis is a powerful tool in simplifying the modelling of physical processes. Importantly, it helps better understand the physics at play in a system. By identifying the subset of terms that truly matter in an equation, one can deal with lighter computations by avoiding unnecessary complication of the model. Taking the example of meteorology, modelling the entire atmosphere using the full Navier-Stokes equations of motion for all scales would have an immense computational cost. And a large amount of improvements in Numerical Weather Predicitons can be attributed to scale analysis [4, 6, 26, 34].

However, this process can be slow as it requires considerable expertise from researchers. And for most of the well studied physical systems, this was done by hand over decades (cf. above references). But with the wealth of computational power and data science techniques nowadays, an attempt at automating dominant balance can be made. First is the Portwood et al. (2016) [27] paper which used a cumulative distribution function on local density gradients to separate each region of a stratified turbulent flow. The method used was highly tailored for its case study, with the gradient of one of the terms being used, knowing it had dynamics discerning qualities. And results were interpreted through the knowledge of the authors. Second is the work carried out in Lee & Zaki (2018) [19] where an algorithm to detect different dynamical regions is introduced. Again, this is through the use of case-specific variables (vorticity), which restrict the use of this algorithm to certain flows. Finally, Sonnewald et al. (2019) [31] used a K-Means clustering algorithm to identify dynamically distinct regions in the ocean. Here, they introduced the idea of using the terms in the governing equations as features. However, the identification of active terms is done through comparison of the magnitudes of each of the terms in the equation. In other words, identification of dominant terms is not done algorithmically but "manually". Thus, these methods are mostly designed for specific

case studies and partly rely on expert knowledge to interpret the results.

A similar challenge in data science and machine learning has been to directly find the laws and equations that govern a system from data. Schmidt & Lipson (2009) [29] contributed to a breakthrough using symbolic regression to find linear and non-linear differential equations. And this was improved in Brunton et al. (2016) [3], approaching with a less expensive sparse regression, which for high-dimensional problems means identifying a sparse governing equation. This depended on governing equations usually having only a subset of terms being important, as in dominant balance. Lejarza & Baldea (2022) [20] further advanced this by using multiple basis functions and a non-linear moving horizon optimization algorithm to learn governing equations from noisy data. Deep learning methods have also been used in this effort. First, where the lagrangians are learned, therefore learning how to model complex physical systems, and learning symmetries and conservation laws, where other networks failed [10]. Second, deep learning (Graph Neural Network) and symbolic regression are combined to create a general framework to recover equations of physical systems [11]. This method has the advantage of being generalisable and therefore useable to extract plausible governing equations for unknown systems.

This generalisable quality is precisely the gap that Callaham et al. (2021) [5] attempt to fill in the identification of dominant balance models. They propose a novel approach to take in simulated or measured data from virtually any physical system, and extract dominant balance models with minimal user input. This means one could use it in conjunction with the above governing equation identifying methods, and explore plausible governing equations and asymptotic regimes of an unknown physical phenomena. Though this is a very powerful result, as Schmidt & Lipson (2009) [29] noted for their work, this method should be seen as a guiding tool to help indicate where scientists should focus their attention, rather than a definitive answer.

Chapter 3

Methodology

The method proposed by Callaham et al. (2021) [5] can be summarized in three steps. And this section aims to delve into the details of each of these steps.

3.1 Data & Equation Space Representation

For this method, obtaining the data can be done through simulations or measurements, with almost no restrictions as to what equations can be studied. The data typically comes in the form of space-time fields of physical variables: $u(\vec{x}, t)$, where u is the physical variable of interest (e.g. $\vec{u}$, p, etc. for the Navier-Stokes equations). It is essential that one can compute all the terms of the governing equation from the variables. Additionally, the computed terms must all balance out to 0 for each point in time and space. This relies on having the computed terms be as they appear in the equation, which ensures a linear covariance structure, as each term will be balanced by a linear combination of the other terms [5, Supplementary Information].

The fundamental idea of the Callaham et al. (2021) method is to take these fields of terms in the physical-space and organize them into an equation-space, where each dimension or feature is one of the terms in the equation, and each sample is a point in space and time. Thus, each sample will be a vector $\vec{f} \in \mathbb{R}^k$ where k is the number of terms in the equation, such that:

$$\vec{f} = \begin{bmatrix} f_1(u(\vec{x}, t), \ldots) \\ f_2(u(\vec{x}, t), \ldots) \\ \vdots \\ f_k(u(\vec{x}, t), \ldots) \end{bmatrix} \tag{3.1}$$

Where f_i is the i^{th} term in the equation, itself a function of the physical variables. Again, one must ensure that for each sample, the terms all balance out to a residual

at least several orders of magnitude smaller than the terms themselves. This is in order to make sure that the equation studied or data used is not invalid.

3.2 Gaussian Mixture Model Clustering

By clustering the points in equation space, groups of points that have a similar balance of terms can be identified. Here, the chosen algorithm is the Gaussian Mixture Model (GMM) clustering algorithm. This algorithm relies on the assumption that the data has been generated from a mixture of a finite number of Gaussian distributions with unknown parameters [24]. It has the advantage of being able to identify clusters with varying shapes and sizes. It only requires one hyperparameter, which is the number of clusters the algorithm must find, and therefore how many Gaussian distributions the data is assumed to have been generated from [25]. This number is not fixed by the method so must be chosen for each case study. Generally, the choice should be conservative, aiming for a number of clusters that is likely to be greater than the true/expected number of dominant balance regimes in the system. This is because if the number is higher than it needs to be, it will likely be the case that the next steps in the method will identify some of the clusters as having identical dominant balances, and these will be combined.

The way GMMs fit to the data is by using the Expectation-Maximisation algorithm. This algorithm starts from an initial guess for the parameters of the gaussian distributions: w_0, the weight for that distribution, μ_0, the vector mean of size $[1 \times n_{features}]$, and $\mathbf{\Sigma}_0$, the $[n_{features} \times n_{features}]$ covariance matrix, for each Gaussian. The expectation step evaluates the likelihood of a given point belonging to each cluster, for the current parameters. And the maximisation step updates the parameters with weighted averages of those posterior probabilities [12] (see Alg. 1, in section 7.1). This converges to the maximum likelihood estimates of the parameters (see Fig. 3.1). Once fitted, cluster membership of new data points can be determined by taking the cluster with the highest probability at that point.

Another key output of the GMM is in its probabilistic nature as each Gaussian is defined by a covariance matrix of the equation's terms. This gives insight into

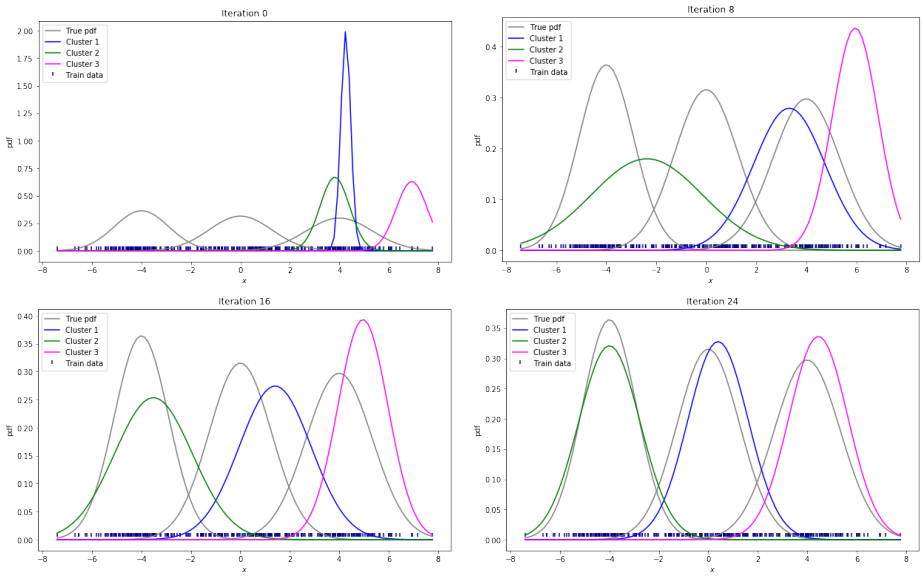


Figure 3.1: Example of a Gaussian Mixture Model fit to data. The data is generated from 3 Gaussian distributions, and the GMM algorithm fits 3 Gaussians to it. [9, 1D Example Notebook]

which terms could be active in each cluster, and helps illustrate the intuition behind identifying active terms geometrically with this method. If only a subset of the terms in a cluster have non-zero covariance, then it means that it is only along that subset of dimensions that the cluster varies, and hence that the terms are likely to be non-negligible.

3.3 Sparse Principal Component Analysis

With the points grouped in clusters of similar balance of terms, the next step is to identify which terms are active and which can be dropped to simplify the equation for that cluster. This could be done by applying a threshold to the covariance matrices obtained from the GMM clustering. But it is done more robustly through Sparse Principal Component Analysis (SPCA). SPCA is a variant of Principal Component Analysis (PCA) which is a method used to reduce the dimensionality of the data, whilst maximising the information retained. The new dimensions onto which the data is projected are called principal components and are the directions in which the data has the greatest variance [21] (see Fig. 3.2). SPCA differs from PCA in that it

adds a sparsity constraint to the principal components, adding a ℓ-1 penalty to the objective function of PCA for the number of non-zero coefficients. This then leads to some coefficients being shrunk down to 0 when the ℓ-1 penalty factor (α) is large enough [35] (see Alg. 2, in section 7.1). When setting a low value for α, SPCA will be more lenient in judging if a term is active in a cluster. The higher the value, the sparser the principal components will be. Too low, and all clusters will have a full balance, with all terms active. Too high, and all clusters will have an empty balance, with no terms active.

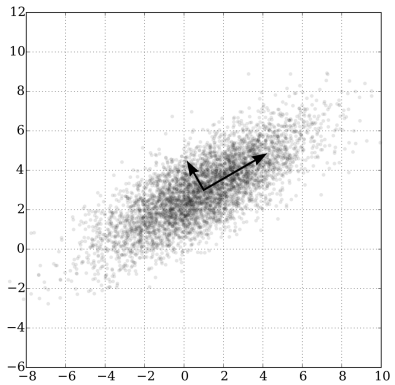


Figure 3.2: Example of a 2D dataset projected onto its first 2 principal components (black arrows). The first principal component is the direction of greatest variance, and the second is orthogonal to it. [33]

Taking the GMM clustered data, the SPCA algorithm is applied to each cluster, and only the leading component is extracted. By taking the leading component, the algorithm is identifying the one direction along which there is the most variance within that cluster. This will indicate which terms vary the most within that cluster, and therefore which ones matter more. Then applying the sparsity constraint, the vector obtained will have some of its coefficients shrunk to 0, and the non-zero coefficients will indicate the active terms in that cluster.

To help judge what value of α to pick, Callaham et al. (2021) propose to apply SPCA for a range of α values. Then a residual error is computed, and is defined as the ℓ-2 norm of the neglected terms across all clusters. This illustrates the trade-

off between sparsity (simplifying the model) and retaining information (keeping the model accurate), helping to pick an optimal value (see Fig. 4.16). Once SPCA is applied, each cluster is then associated to a sparse binary vector defining their active terms (1 if active, 0 if not). The balance models are then checked for uniqueness, and if some clusters have the same balance, they are combined.

Chapter 4

Conducted research

To test and validate this method, Callaham et al. used a series of case studies [5]. In this chapter, the aim is to discuss the reproducibility of the results presented in the paper, as well as to test and discuss its drawbacks.

4.1 Portability of the code

One great quality of the Callaham et al. (2021) [5] paper is the sharing of their code in the form of runnable notebooks. This is tied to the motivation for this project being the importance of reproducibility in today's code-rich research environment. It is key that scientists share their code so one may verify how the results were obtained. This is also a great asset as it allows for a more thorough understanding and test of the method, by writing explicitely different code.

Overall, the code is of great quality, though some portability issues were encountered. Some of the dependencies were not stated in their README. More importantly, it seemed some of the data-generating code was modified between running their notebooks for the final time and uploading it to the repository. This led to some troubleshooting to get the data to match the one used in the paper. Unfortunately, for the flow past a cylinder case, a file needed in the setup of the Nek5000 simulation software was missing [15]. For the bursting neuron case, the times for which the data was generated was wrongly set, not starting from $t = 0$ s. Similarly, for the Gulf of Mexico's currents data, the remote reading code was selecting snapshots of the data that did not match the ones used in the paper, and the first 45 were set to zero, which had no real use. Further, the results in the paper were actually obtained from a different snapshot than the one stated in the paper. For this report, the data was simply downloaded from the HYCOM database [8], and the time was set to the same as stated in the paper.

Nevertheless, the code was otherwise easy to run, and the notebook format is very useful to better understand the logic of the code. Following this code, the code was first written for the turbulent boundary layer case. And explicitly alternative code was then written to test the reproducibility of the results.

4.2 The turbulent boundary layer case

To ensure that none of the main results from Callaham et al. are due to a "lucky" bug in their code, and were obtained through random chance, alternative code is written which, in practice, performs the same functions. Also, because the underlying method is essentially the same for all case studies, it was decided to put particular focus on one of the cases: the turbulent boundary layer.

4.2.1 Reproducing the code

Throughout the Callaham et al. (2021) [5] notebooks, aside from the three main steps of the method, the large majority of the code involves handling the data, switching from equation space to physical space, and obtaining the unique balance models. Most of it is written using `Numpy`, which is arguably a very good choice for efficient and trustworthy numerical operations. In the alternative code, however, `Pandas` was primarily used. In terms of performance, it is close to `Numpy`, especially for the size of the data in this study, and provides similar practical functions. For some code sections, however, impractical workarounds were required, particularly to replace the `numpy.unique` method, when finding and combining duplicate balance models.

The data for this case study is obtained from the John Hopkins database, which conducted a direct numerical simulation of a boundary layer over a stationary plate [18]. This database provides the x and y coordinates as well as the following variables: $\overline{u}$, the streamwise component of velocity; $\overline{v}$, the wall-normal component of velocity; $\overline{p}$, the fluid pressure; and $(\overline{u\prime v\prime})$ and $(\overline{u\prime^2})$, the wall-normal and streamwise Reynolds stress terms, respectively. From these variables, the terms of the streamwise component of the Reynolds-Averaged Navier-Stokes (RANS) equations need to be calculated (see Eq. (5.1)).

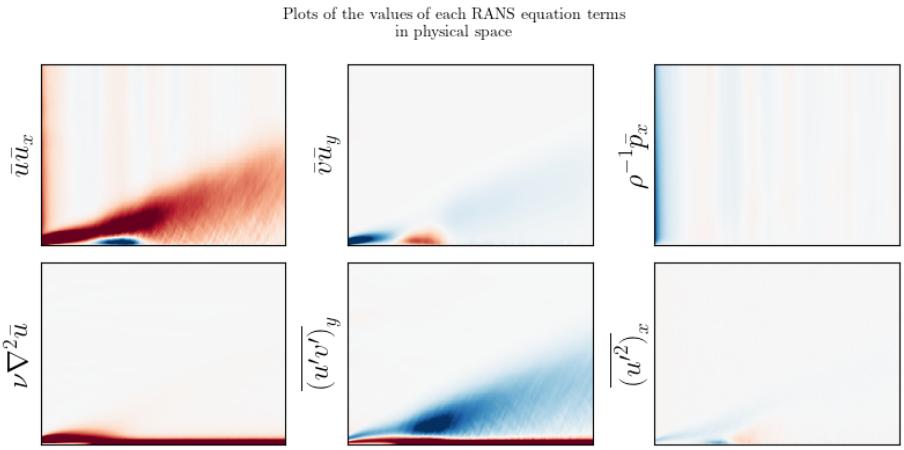


Figure 4.1: Plot of the 6 terms in the RANS equation (Eq 5.1), using the original Callaham et al. code

$$\bar{u}\bar{u}_x + \bar{v}\bar{u}_y = \rho^{-1}\bar{p}_x + \nu\nabla^2\bar{u} - (\overline{u'v'})_y - (\overline{u'^2})_x \tag{4.1}$$

To compute the derivative terms, the method employed by Callaham et al. is based on `scipy.sparse`'s sparse matrices library. The aim is to build a derivative operator sparse matrix so that when computing the matrix product of that matrix with the fields or variables, the second-order forward/central/backward difference derivative is obtained, following the method described in 'Fundamentals of Numerical Computations' [13]. Though it has the advantage of being portable for all variables, it is much slower than using the `numpy.gradient` function, which performs the same computations, albeit more explicitly, when setting the `edge_order` argument to 2. With the derivatives computed, spatial fields of each term in the equation can be plotted (see Fig. 4.1).

The next step is to use a Gaussian Mixture Model (GMM) to cluster the data in equation space. Callaham et al. judiciously chose to use the `sklearn` library's implementation of the GMM algorithm. This implementation is likely very efficient, being written and optimized by experienced developers. With the chosen arguments, the algorithm is initialized using the `'k-means++'` method. This method initializes the means of the Gaussians using the centroids found by the K-Means clustering algorithm (see Algorithm 3), which is itself initialized based on the empirical prob-

ability distributions of the data [1]. The covariance matrices are then initialized as the covariance matrices of the clusters found by the K-Means algorithm. The algorithm then uses the Expectation-Maximisation algorithm as expected. Convergence is evaluated using the log-probability of the data:

$$\log \mathcal{L}(\vec{X}|\vec{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} w_k \mathcal{N}(\vec{x}_i|\vec{\mu}_k, \vec{\Sigma}_k) \right) \tag{4.2}$$

The algorithm stops when the difference between the new and old log-likelihood is less than 10^{-3}.

The alternative code was thus written aiming to follow the same initialization, explicitly using `sklearn`'s K-Means algorithm with `'k-means++'` initialization. The Expectation-Maximization algorithm is then implemented as described in Algorithm 1, using the total log-likelihood difference as the convergence criterion.

For Sparse Principal Component Analysis (SPCA), the `sklearn` library's implementation is used. The method employed is based on an extension of sparse PCA, which utilizes structured regularization to constrain the sparsity patterns. This is also done for the alternative code, as the `sklearn` implementation employs high level methods which were hard to reproduce [16, 23]. However, it was decided to use a parallelising package to speed up the computation of the SPCA residuals. Here, the `joblib` package was used, providing a simple way to distribute jobs [17].

With SPCA completed, the active terms can be identified. From this point, the original and alternative code only differ in which library they predominantly use. The original code primarily uses `Numpy`, while the alternative code uses `Pandas`. Alternative code was thus also written in handling the balance model results, getting the unique ones, and plotting them in space.

4.2.2 Results

First, checking the computation of the RANS equation's terms, the `scipy.sparse` method's obtained terms are plotted in Fig. 4.1. The alternative code, which used `numpy.gradient` instead, gave the same results (see Fig. 4.2).

The next step involves clustering the data in equation space. A clear way of comparing results is to examine the obtained covariance matrices. In Figure 4.3, the

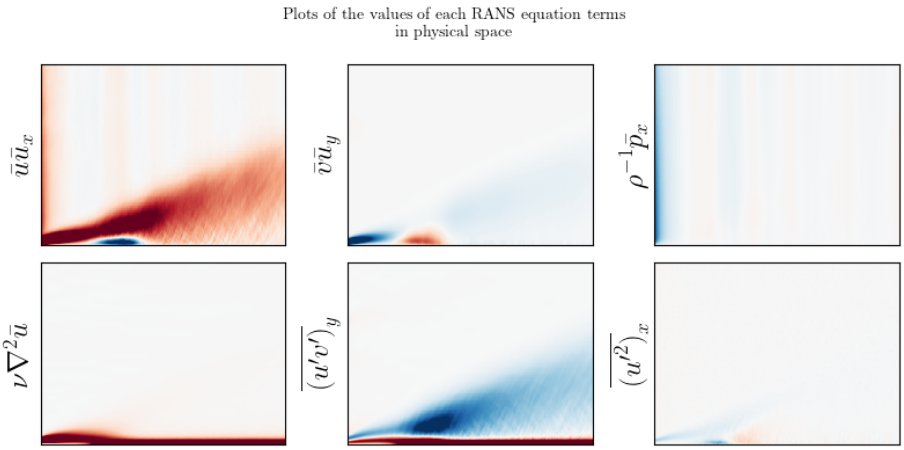
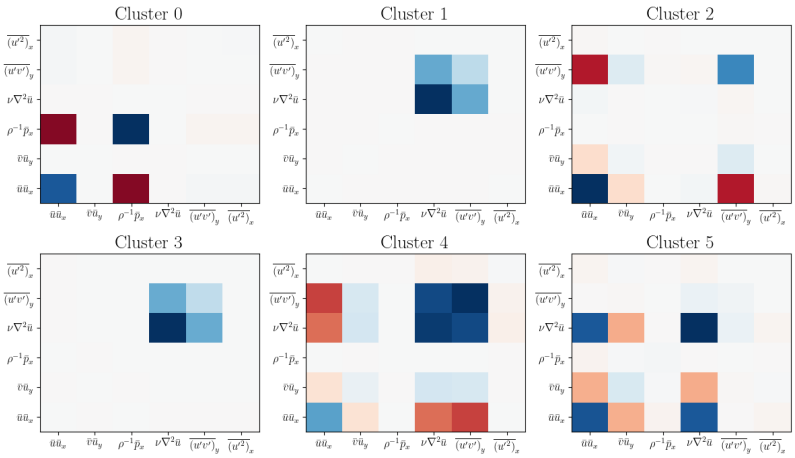
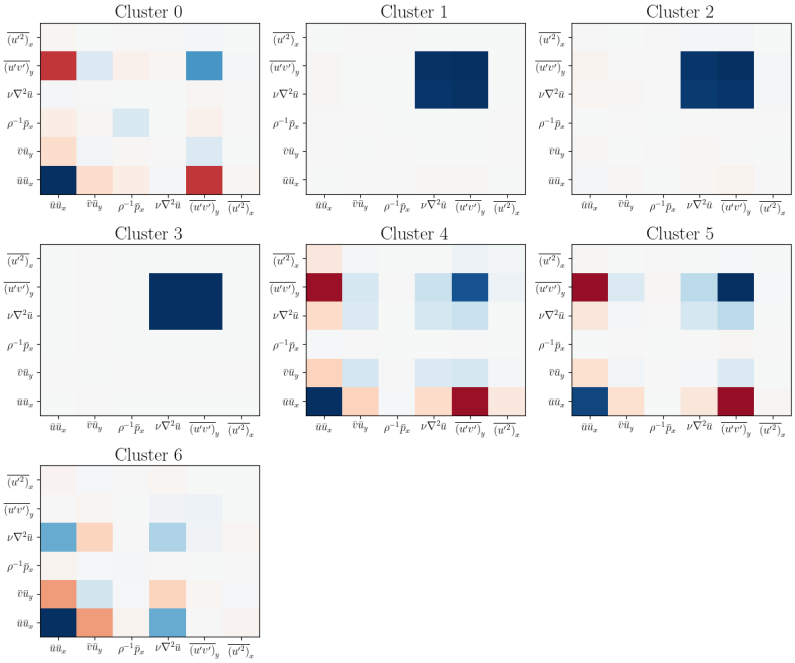


Figure 4.2: Plot of the 6 terms in the RANS equation (Eq 5.1), using an alternative code

covariance matrices obtained when using `sklearn`'s Gaussian Mixture Model (GMM) routine, as well as when using the custom GMM implementation are shown. It can already be seen that some clusters clearly have the more important terms. This is the case within each method (Cluster 1 and 3 in Fig. 4.3(a)) but also between them: clusters 1 in (a) and (b), or 5 in (a) and 6 in (b), etc. Plotting these clusters in physical space gives the results in Figure 4.4. From fluid dynamics theory, Callaham et al. were able to assign names for each clustered region in Figure 4.4(a), and these will be discussed later with Figure 4.5. As can be seen, the results do differ, with the alternative code needing 7 clusters to identify the transitional layer (purple in Fig. 4.4(a) & (b)), instead of 6 for the original code. The difference could be explained by either the different convergence criteria or some of the covariance matrix regularizations which is performed in the `sklearn` implementation [25].

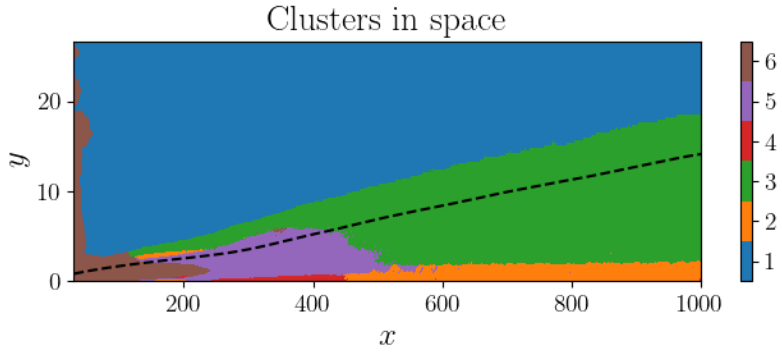


(a)

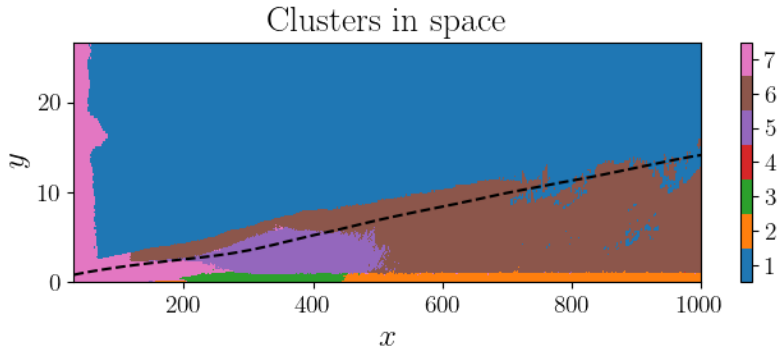


(b)

Figure 4.3: Covariance matrices of for each of the clusters found by the `sklearn` (a) and custom (b) GMM algorithm. The colorscale is not too important, as the magnitude of the covariance matrix is more important.



(a)

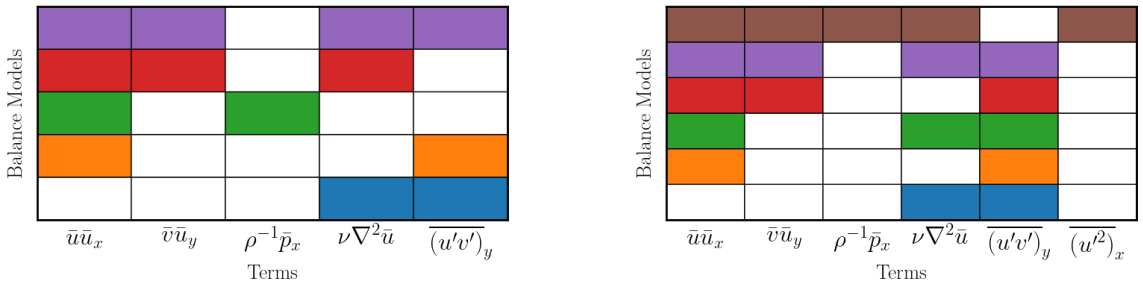


(b)

Figure 4.4: Plot of the clusters found by the `sklearn` (a) and custom (b) GMM algorithms. The cluster numbers match the ones in Figure 4.3, with the difference of not being 0-indexed (Cluster 1 here is cluster 0 in Fig. 4.3)

Then comes applying SPCA to each cluster to identify which terms are active in it. Due to the different clustering obtained, the optimal alpha values needed changing between the methods. For the original code, the optimal alpha value was set to 10, but had to be set to 7 for the alternative code, in order to keep the purple cluster in Fig. 4.6(b) from being combined with the brown one. The results of the SPCA algorithm are $n_{clusters}$ sets of active terms. In the case where some clusters have the same active terms, they are combined. This results in a set of unique balance models, which are shown color-coded in Figure 4.5. As the turbulent boundary layer is a well-studied physical phenomenon, clusters in Fig. 4.5 and 4.6 have been assigned names to describe the dynamics at play in them. From these plots, the alternative code was able to similarly identify important dynamics in the boundary layer. As in Figure 4.3, some clusters are well identified by both methods. The two spatial arrangements of the clusters are the same, though their dynamics sometimes differ, or even switch, such as the orange cluster in Fig. 4.5 having the same dynamics but describing two different regions in Fig. 4.6. Overall, similar key dynamics are identified in most of the clusters. For example, the viscous sublayers in blue (a) and green (b) of Fig. 4.6 has the viscous forces and streamwise Reynolds' stress terms as active in both cases. The inflow region in both cases has the streamwise Reynolds' stress term as inactive, which is the main characteristic of that region. And in both cases, the transitional layer is identical (purple cluster).

Overall, there is a good agreement between the results, considering differences in the specifics of the Gaussian Mixture Modelling clustering algorithm. The alternative code was able to reproduce the identification of pivotal dynamics in the boundary layer. It is also important to note that setting the value of the hyperparameters (number of clusters, alpha value) played a key role here. These parameters can have a large impact on the results obtained. This is a key point and will be further discussed in the Stability Assessment section.



(a) (b)

Figure 4.5: Plot of the unique balance models found after applying SPCA, when using the original Callaham et al. code (a) and an alternative code (b). Here, the cluster colors will match the ones in Figure 4.6. Multiple regions are identified. The inflow region (red): where the flow is still laminar, and the streamwise Reynolds stress term is inactive. The free-stream region (green): where only inertial and pressure gradient forces matter, and the viscous forces are inactive. The inertial sublayer (orange): where the inertial forces are the most active. The transitional layer (purple): where the flow is transitioning from laminar to turbulent, hency why a lot of the terms are active. Finally, the viscous sublayer (blue): where the flow is dominated by the viscous forces.

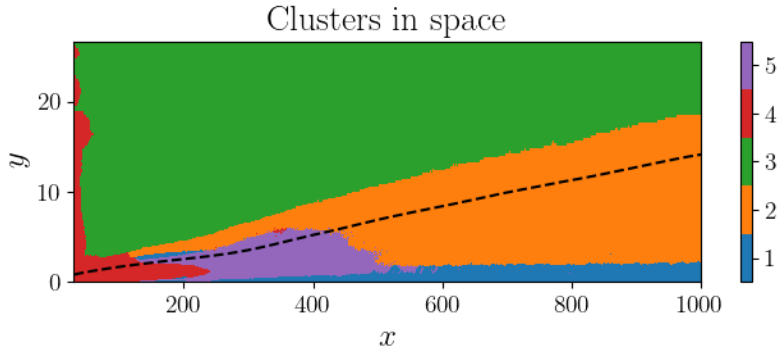
4.3 Exploration of other algorithms

In the interest of further testing Callaham et al.'s method, and following the Discussion in the Supplementary Information of the paper [5], other clustering algorithms could be used compared to Gaussian Mixture Models. This section explores a few options and discusses the results obtained.

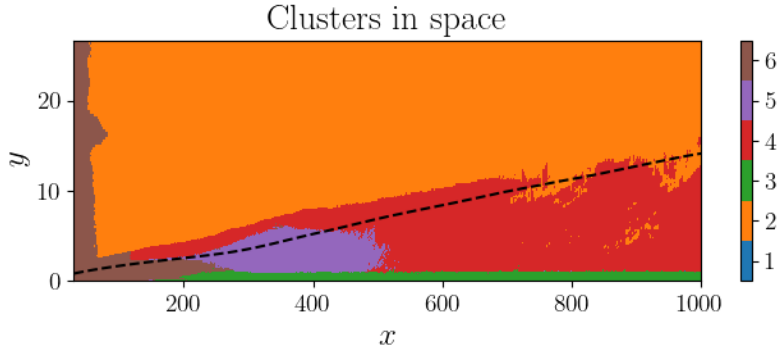
4.3.1 Spectral clustering

Because of the nature of the data being dealt with, the similarity measure used by the clustering algorithm being Euclidean may not be the most appropriate [32]. As a result, spectral clustering is a convincing candidate for this, and it was also suggested in the Supplementary Discussion of the paper [5, Supplementary Information] (see Alg. 3, in section 7.1).

When using spectral clustering, a significant drawback is the computational complexity, which is at worst $\mathcal{O}(n^3)$, where n is the number of samples. This makes the algorithm less efficient for larger datasets, which are common in physical systems. Furthermore, it is not possible to sequentially add new points to the graph.

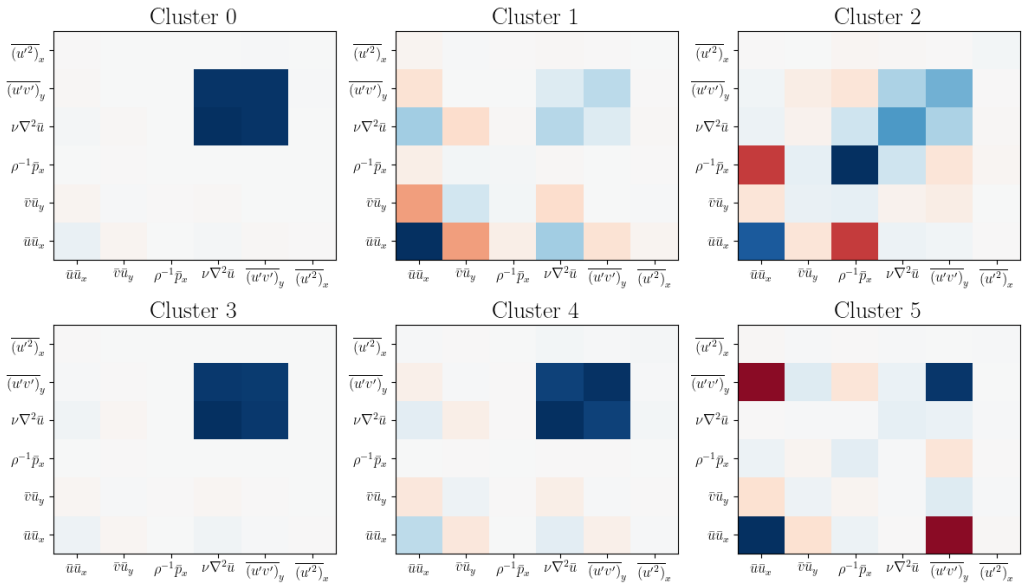


(a)

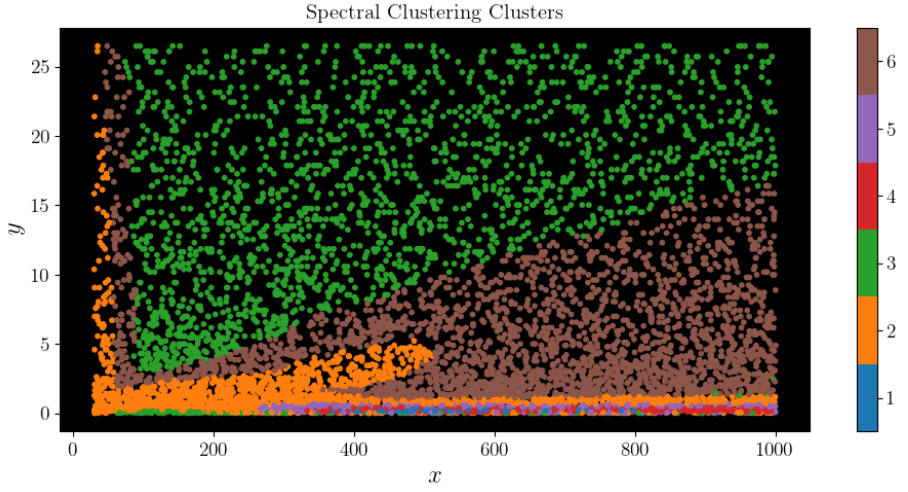


(b)

Figure 4.6: Plot of the unique balance model clusters in physical space, when using the original Callaham et al. code (a) and an alternative code (b). **(a)** Identified balance models include a laminar inflow region (red), a free-stream region (green), an inertial sublayer (orange), a transitional layer (purple), and a viscous sublayer (blue). **(b)** Identified balance models include an inflow region with low Reynolds' stress (brown), a free-stream region (orange), an inertial sublayer with stream-wise and wall normal inertial forces (red), a transitional layer (purple), and a viscous sublayer (green and blue). The cluster colors here match the ones in Figure 4.5



(a)



(b)

Figure 4.7: Results of the Spectral Clustering algorithm for the turbulent boundary layer case. **(a)** Covariance matrices of for each of the clusters found by the Spectral Clustering algorithm. **(b)** Plot of the clusters found by the Spectral Clustering algorithm.

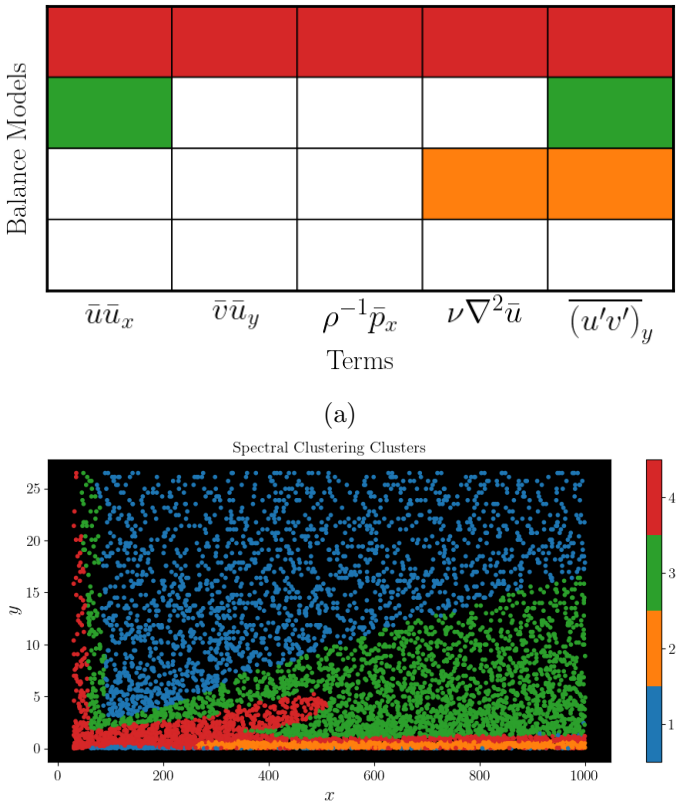


Figure 4.8: More results for the Spectral Clustering algorithm method. **(a)** Plot of the unique balance models found after applying SPCA. **(b)** Plot of the unique balance model clusters in physical space. Identified balance models include a viscous sublayer (orange), an inertial sublayer (green), a transitional and inflow layer (red), and a no-dynamics free-flow region (red).

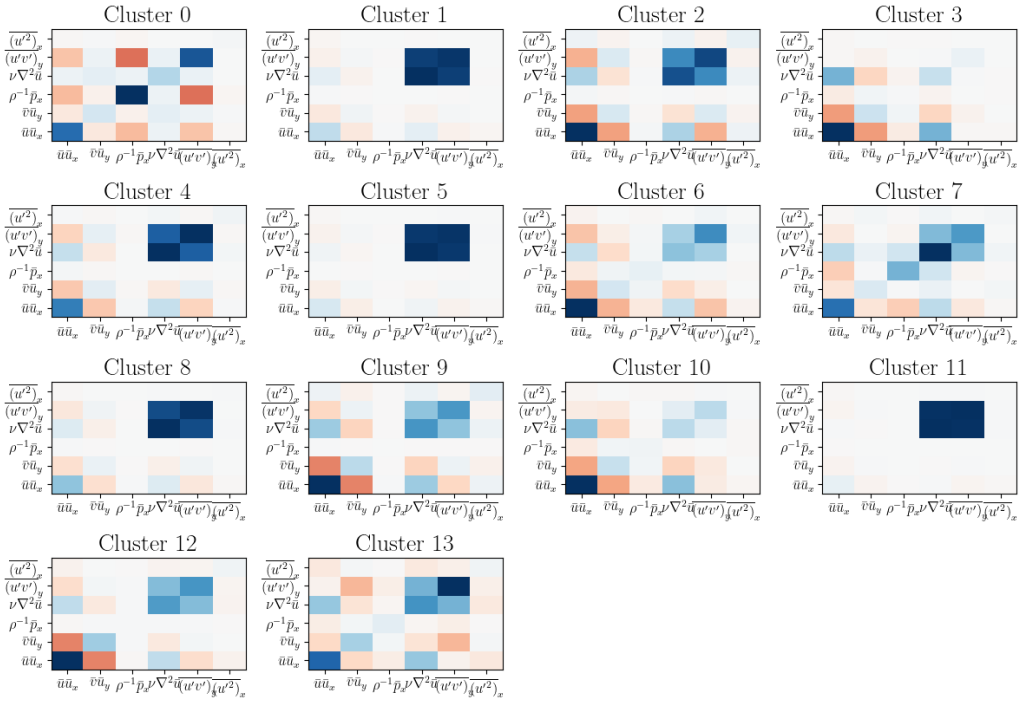
By training it on a small fraction (0.01) of the full dataset, the results are shown in Figure 4.7 and 4.8. Because prediction of cluster membership for the rest of the dataset is hard, the solution was to plot the cluster memberships as a scatter plot. Overall, spectral clustering deals well with the different-shaped clusters, as the main structure in the flow was captured. And some of the key dynamics are identified, notably the inertial (green) and viscous sublayers (orange) (see Fig. 4.8). However, the inertial sublayer here extends vertically into the region identified by the original code as the laminar inflow region. Moreover, a large cluster, which spans the previously identified transitional and laminar inflow region, is simply considered to have all terms active (except the wall-normal stress term). One other novelty is that the free-flow region was identified to have no active terms.

Therefore, though it provides a non-Euclidean based clustering algorithm, spectral clustering is not well suited for this case study, or at least the aims of the method proposed by Callaham et al. (2021) [5]. It is computationally expensive, cannot accept new data points, and additionally requires one extra parameter to set (`n_neighbours`) when using nearest neighbours to build the graph.

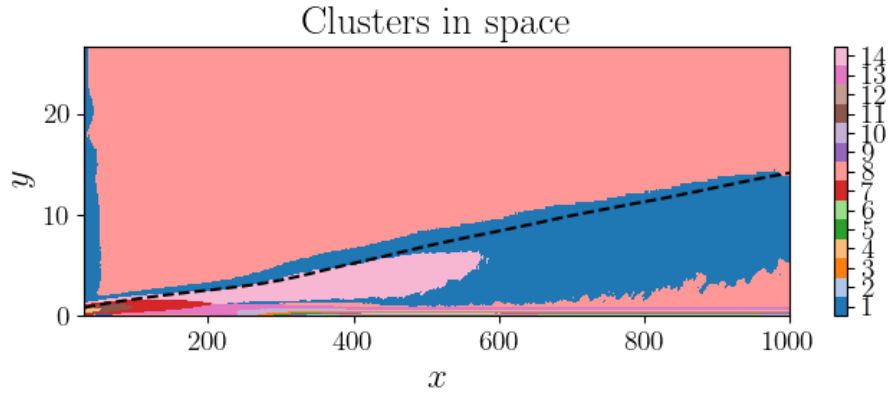
4.3.2 K-Means

The next algorithm used is the K-Means algorithm. Compared to spectral clustering, it is much less computationally expensive and is able to accept new data points. The main drawback is that it may not be best suited for some physical system. For example, in the case of the turbulent boundary layer, the clusters are not spherical. More importantly, a great majority of points are near the origin, and K-Means clustering will struggle to distinguish that group of points as multiple clusters. One workaround used here is to set a higher number of clusters to find, which will force the algorithm to identify multiple clusters in the cloud of points near the origin.

The results of the K-Means algorithm are shown in Figure 4.9 and 4.10. Using the larger cluster number did help in identifying distinct regions; however, the dominant balances identified do differ significantly from the ones in the original code. Again, clusters with similar dominant balances between the two methods here extend into other regions that had been identified to have very different dominant balance regimes (e.g. orange cluster in Figure 4.10).

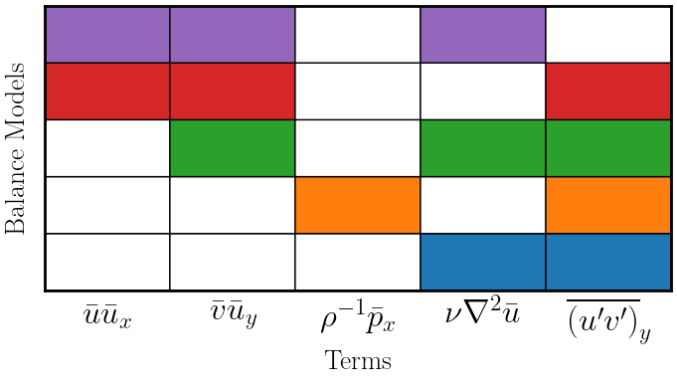


(a)

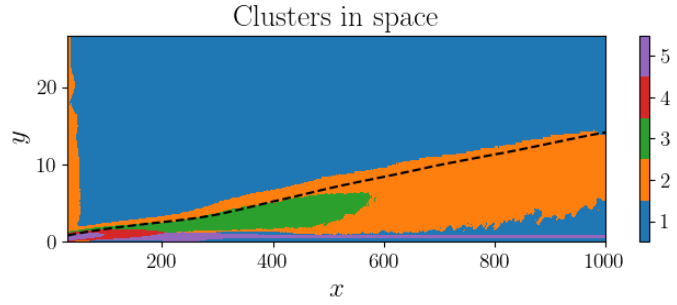


(b)

Figure 4.9: Results of the K-Means algorithm for the turbulent boundary layer case. **(a)** Covariance matrices of for each of the clusters found by the K-Means algorithm. **(b)** Plot of the clusters found by the K-Means algorithm.



(a)



(b)

Figure 4.10: More results for the K-Means algorithm method. **(a)** Plot of the unique balance models found after applying SPCA. **(b)** Plot of the unique balance model clusters in physical space. Identified balance models include an inflow region (purple), a inertial sublayer and inflow region (orange), a transitional layer (green), and a region that spans the free-flow region and the viscous sublayer(blue).

4.3.3 Weighted K-Means

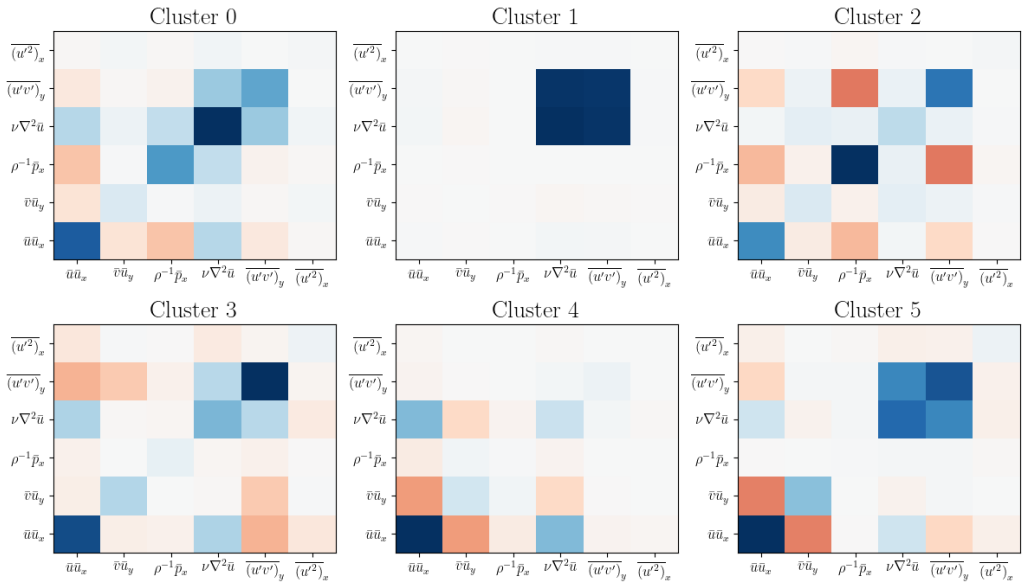
One solution to K-Means relying solely on Euclidean distance is to apply a weight to the samples. This is done to give more importance to the samples that are closer to the origin, essentially spreading them apart. This should encourage the algorithm to identify multiple clusters in the cloud of points near the origin. The weights are set to depend on the distance to the origin as follows (see Fig. 7.1):

$$w_i = 1 - (\tanh^2(\frac{1}{2}|\vec{OX}|)), \text{ where } \vec{OX} \text{ is the vector from the origin to the sample} x_i \tag{4.3}$$

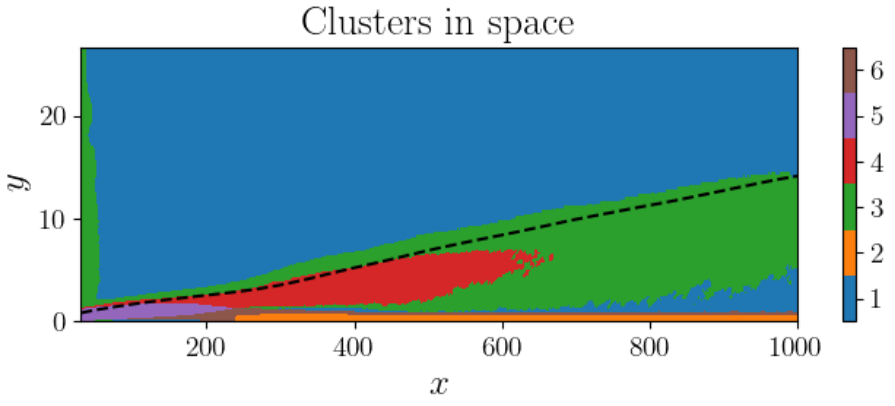
The results of the Weighted K-Means algorithm are shown in Figure **??**. The results are quite similar to standard K-Means; however, they were obtained by having the GMM find 6 clusters only. Once again, some of the key dominant balance regimes are identified, but their location sometimes differ from the original code.

4.3.4 Summary

Overall, the Gaussian Mixture Model clustering algorithm offers really good flexibility thanks to its small number of hyperparameters, and ability to predict cluster membership for new data points. Additionally, it is able to identify clusters of different shapes and sizes, which is important when differentiating different dominant balances close to the origin. Another possible path to explore could be a mixture of other distributions, e.g., Cauchy.

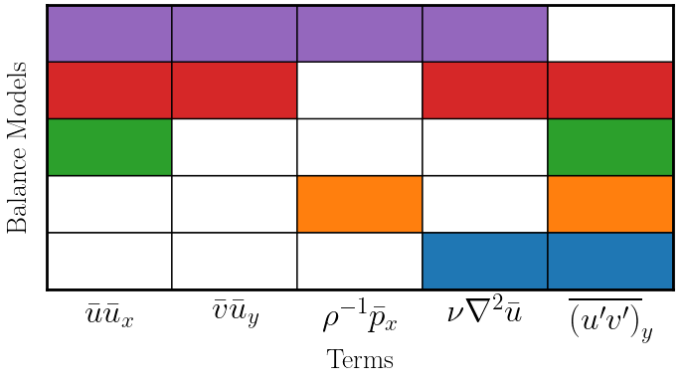


(a)

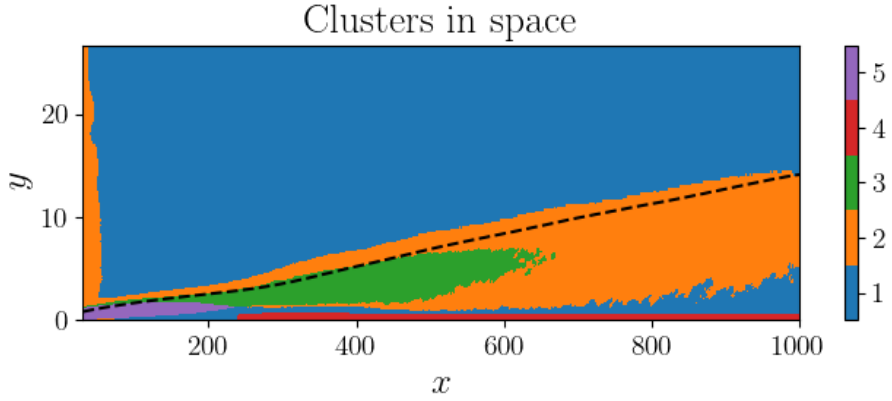


(b)

Figure 4.11: Results of the Weighted K-Means algorithm for the turbulent boundary layer case. **(a)** Covariance matrices of for each of the clusters found by the Weighted K-Means algorithm. **(b)** Plot of the clusters found by the Weighted K-Means algorithm.



(a)



(b)

Figure 4.12: More results for the Weighted K-Means algorithm method. **(a)** Plot of the unique balance models found after applying SPCA. **(b)** Plot of the unique balance model clusters in physical space. Identified balance models include an inflow region (purple), an inertial sublayer and inflow region (orange), a transitional layer (green), a region that spans the free-flow region and the viscous sublayer(blue), and finally, a region that spans the viscous sublayer (red)

4.4 Other case studies

Briefly, case studies other than the turbulent boundary layer were also tested. With the exception of the flow around a cylinder and the rotating detonation engine which were not tested due to the data generating files. These were written using the already written turbulent boundary layer case, and following the paper and its supplementary information [5]. The choice of hyperparameter was purely based on trying to obtain results as close as possible to the ones in the paper. Results are shown in extra plots in the Appendix (see Fig. 7.2, 7.3, 7.4, 7.5, 7.6, and 7.7). Overall, the results are quite similar to the ones in the paper, with some key dynamics being identified in all cases. Differences are usually limited to one or two terms being considered active or inactive erroneously for a given balance model. In some cases, instructions were lacking in terms of how the data should be transformed to its equation space form, and some assumptions had to be made. For example, with the optical burst case, because of dealing with complex numbers, the results were obtained by using the real part of the data.

4.5 Stability Assessment

Though Callaham et al. (2021) [5] presents a well-generalizable method for unsupervised identification of dominant balance regimes, there are still two hyperparameters to set: the number of clusters to find when using the GMM algorithm, and the alpha value for the SPCA algorithm. These can have a large impact on the results obtained. This section aims to explore the stability of the results obtained when changing these hyperparameters and discuss the implications of this when using the method for previously unstudied physical systems.

4.5.1 Under different number of clusters set

The first test carried out is to see how results differ when setting a different number of clusters. Ideally, the cluster number set should not matter greatly. With the exception of very small numbers and approaching the limit: $n_{clusters} \approx n_{samples}$, the results should return to the same "true" or expected value by applying Sparse

PCA and combining clusters with identical dominant balances. However, this is not guaranteed. This is becauses by combining clusters, one also combines the active terms of the 2 clusters. And thus inversely, by increasing the number of initial clusters, this could result in dividing clusters into smaller, unidentically balanced clusters.

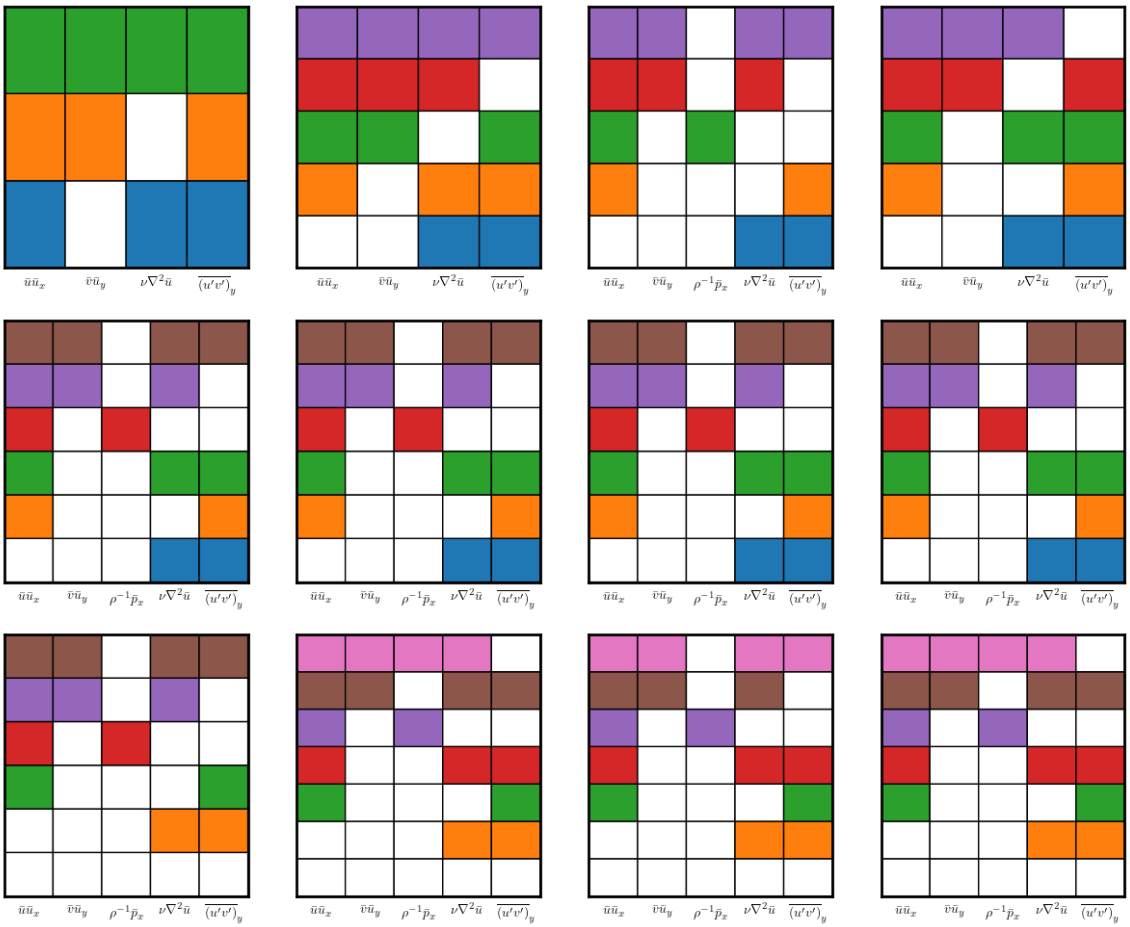


Figure 4.13: Plot of the obtained balance models for different initial cluster number. Reading top to bottom and left to right, the cluster numbers are 4 to 15

The algorithm was run for multiple cluster numbers, and the balance models were obtained for each case. These are all plotted in Figure 4.13. It can be seen that the results are mostly stable, with a slow increase in the number of final unique balance models found (see Fig. 4.14), but with identical balances identified, except for low cluster numbers. Importantly, the relevant case of 6 clusters seems to be unique and

changes quite a lot for the 5 and 7 clusters case.

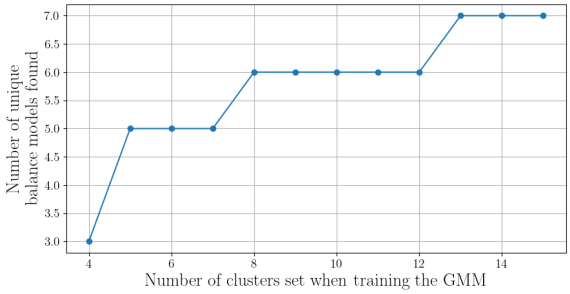
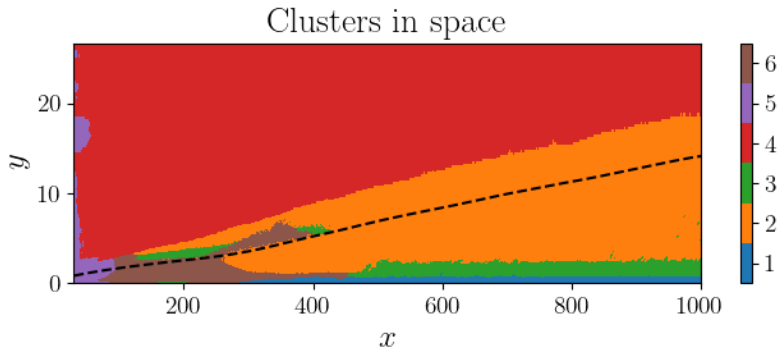


Figure 4.14: Plot of the number of unique balance models identified for different number of cluster numbers

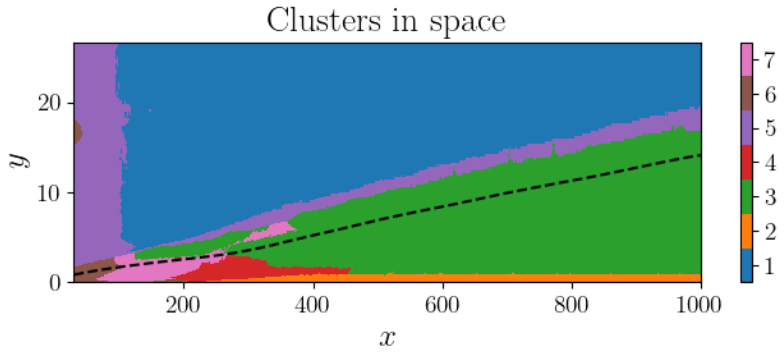
However, checking the actual clustering in space in Fig. 4.15 and comparing to the balance models obtained, it can be seen that having identified the same balance models does not necessarily translate to the exact same clustering in space.

4.5.2 Under different alpha values

To test the stability of the algorithm under changes to the second parameter: α, for the ℓ-1 regularization of the SPCA problem, the same test is done. Taking the case where the cluster number was set to 6, the unique dominant balance models were obtained for values of α going from 0.1 to 17. This is a choice based on the SPCA residuals curve obtained for the 6 clusters scenario (see Fig. 4.16) which shows these values being in an area of acceptable trade-off between sparsity and accuracy.



(a)



(b)

Figure 4.15: Final unique balance models identified when the cluster number was set to 9 (a) and 14 (b).

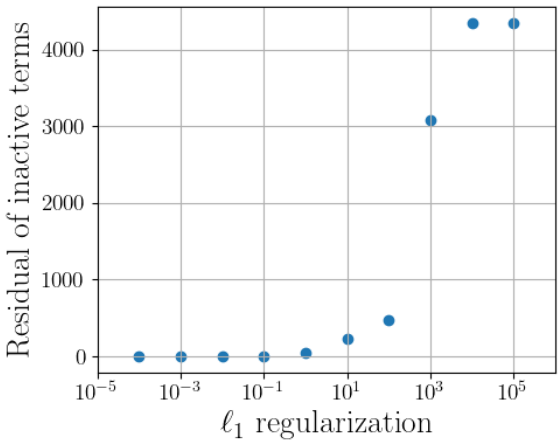


Figure 4.16: Plot of the SPCA residuals for different values of α. Recall, the residuals are computed as the ℓ-2 norm of the inactive terms for all clusters

Looking at the results in Fig. 4.17, there is a lot more variation in the results obtained, with as expected a lot more active terms and less balance models for small α values and the inverse case for larger values.

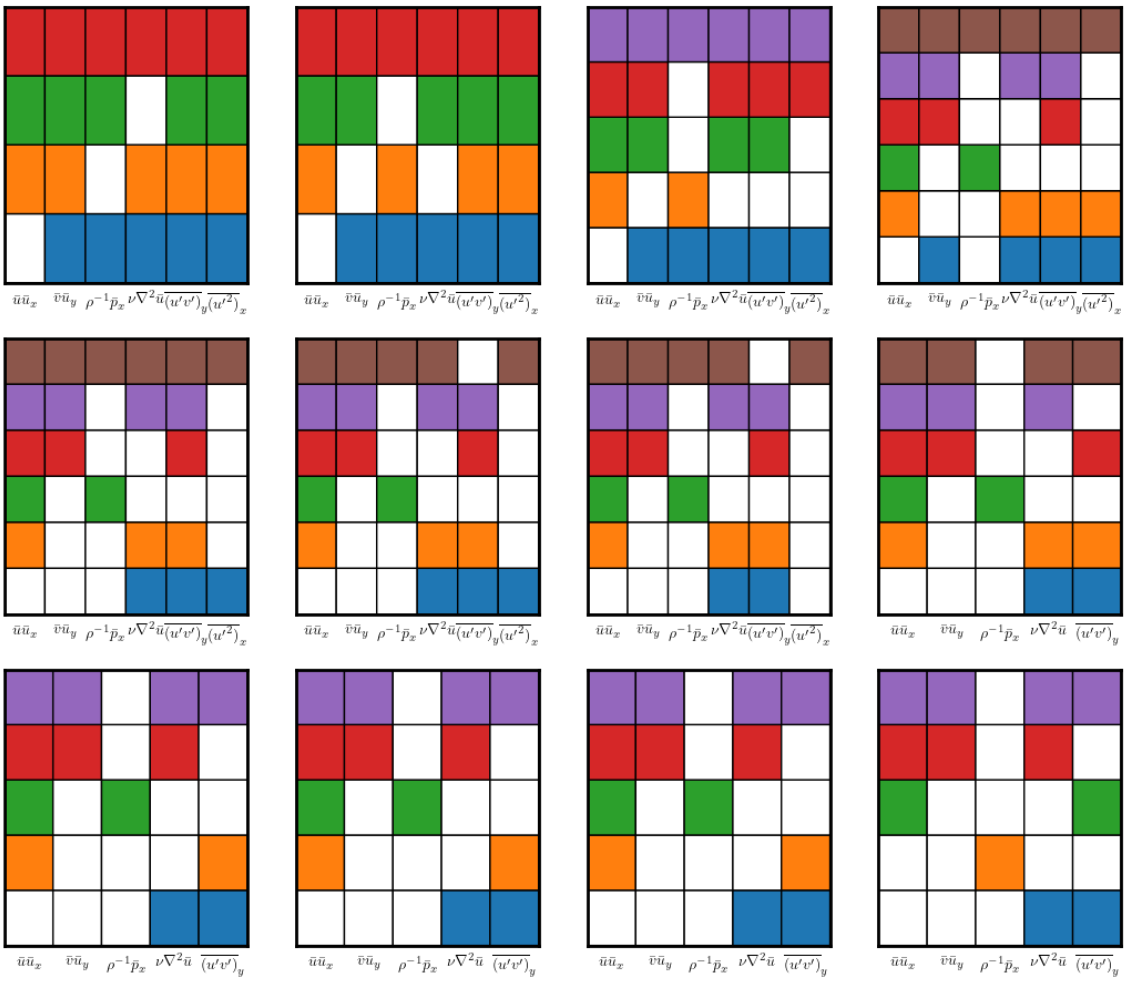
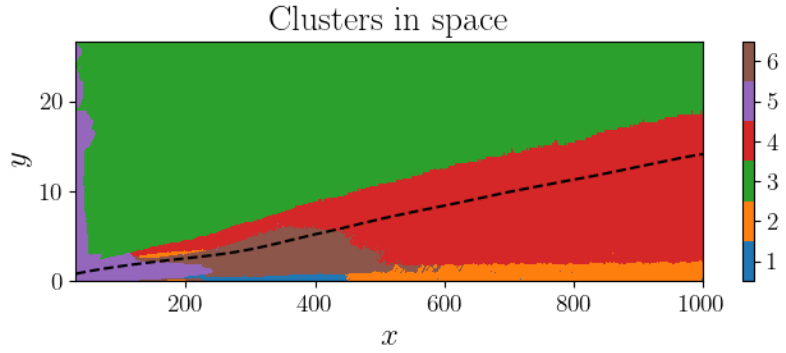
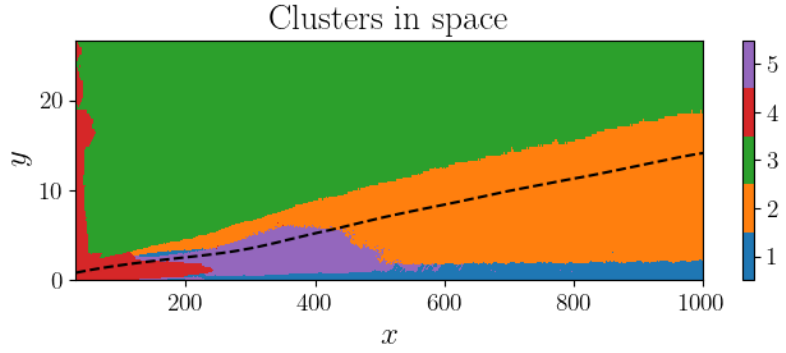


Figure 4.17: Plot of the obtained balance models for different ℓ-1 regularization term factor, α. Reading top to bottom and left to right, the values are 0.1, 0.2, 0.5, 1, 2, 5, 7, 8, 10, 12, 15, and 17

However, for values close to 10, which was selected in the paper, there is good stability, with values from 10 to 15 yielding the same results. And checking that it is still the case in physical space (see Fig. 4.18) shows that though the dominant balance terms are slightly different, the general result stays the same.



(a)



(b)

Figure 4.18: Final unique balance models identified when α was set to 8 (a) and 12 (b).

4.5.3 Under different training set size

Finally, the effect of the training set size on the results is tested (see Fig. 4.19).

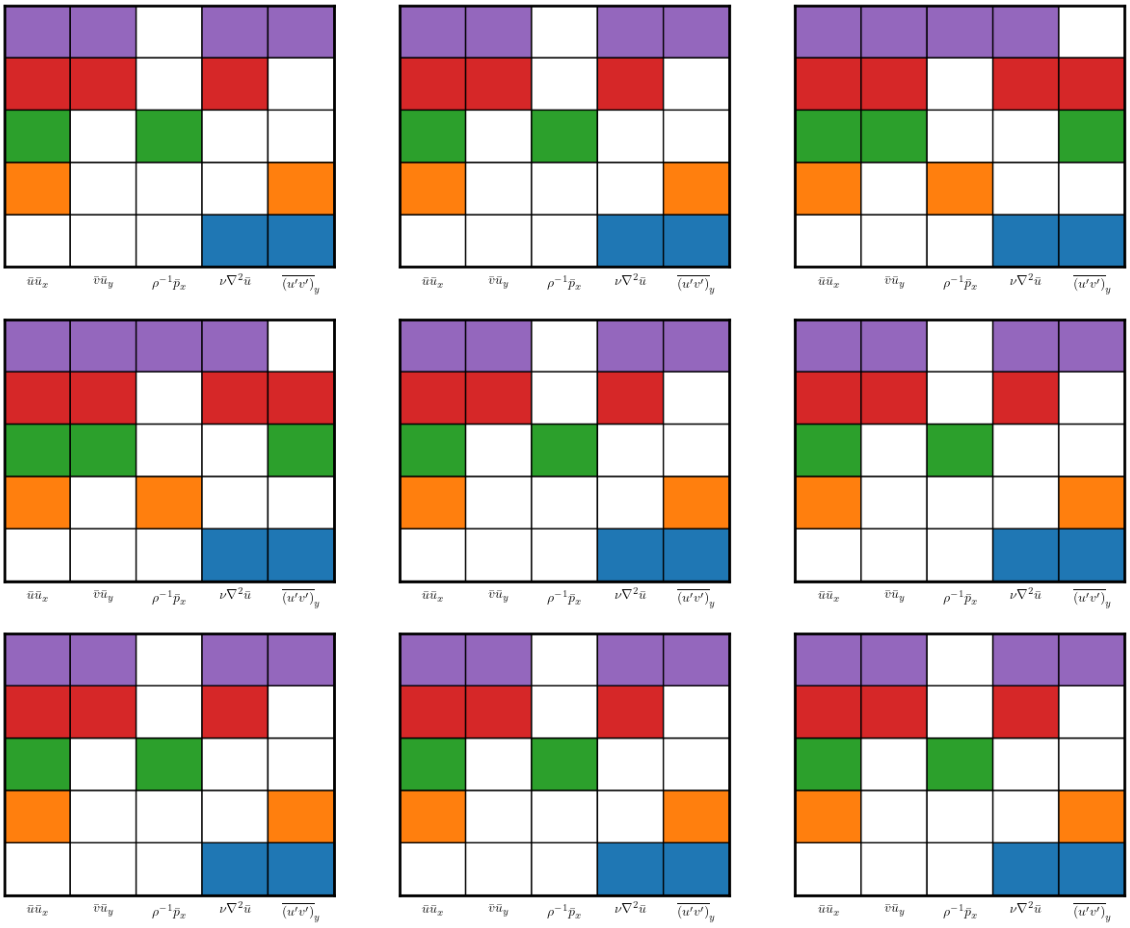
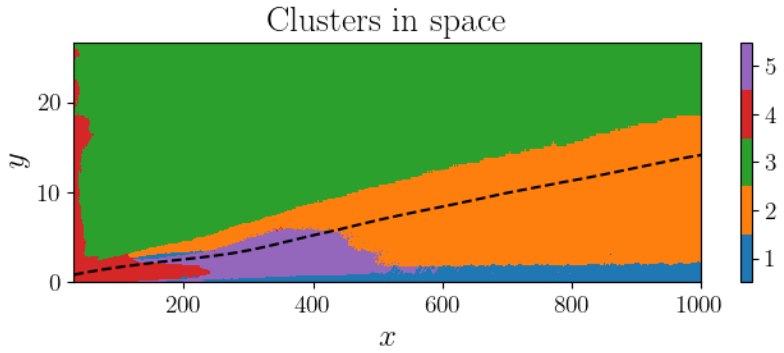


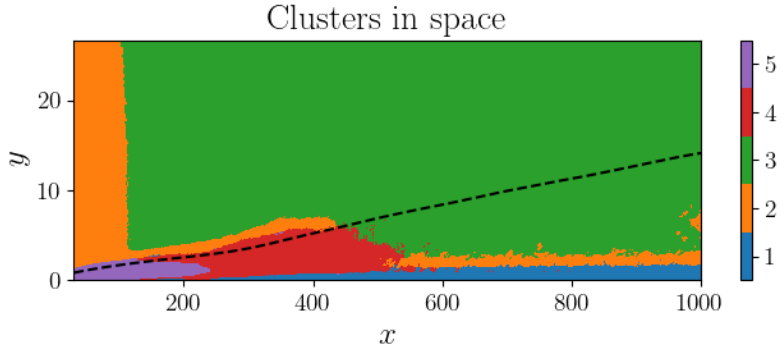
Figure 4.19: Plot of the obtained balance models for different training set size (as a fraction of the full dataset). Reading top to bottom and left to right, the training set size fractions are 0.01, 0.02, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 0.9. Except the first one, there are 2 results that reappear. They are plotted in space in Fig. 4.20.

From these results and plotting the dominant balance models in space, there is a seemingly random instability to the clustering in space, as the inertial sublayer, which delineates the vertical extent of the boundary layer is sometimes not well sectioned (see Fig. 4.20), and this happens even for low and high training set size but not all. When this occurs, the dominant balances change with, for example, the green region in Fig. 4.20(b) now occupying most of the space and having dynamics

that seem to combine those of the inertial sublayer and of the free flow region.



(a)



(b)

Figure 4.20: Final unique balance models identified when the raining set size fraciton was set to 0.1 (a), and 0.5 (b)

4.5.4 Discussion

From the above results, though the Callaham et al. (2021) [5] method made considerable effort to have a minimal amount of parameters, the 2 that must be set do have a considerable impact on the results. And in the absence of expert understanding which exists for the boundary layer case, it is not immediately clear what parameters must be selected as many are essentially valid.

Chapter 5

Elasto-inertial turbulence

As part of this project, there was an opportunity to attempt using the unsupervised identification of dominant balance method on a much more complex polymer-laden flow, which exhibits a special type of turbulence named Elasto-Inertial Turbulence (EIT). This is a relatively recent topic in fluid dynamics, and there is an interest in seeing whether the Callaham et al. method could be used to identify unknown dominant balance regimes in this flow, hopefully shedding some light on the dynamics at play, particularly near the boundaries.

5.1 Background

EIT was introduced around 2013 in a paper by Samanta et al. [28], describing this new type of turbulence occuring in visco-elastic fluids. These are polymer-laden flows, which are found in several industrial application where small amounts of polymers are added to liquids to reduce turbulent drag in pipelines. Samanta et al. [28] also explored the conditions under which EIT occurs, and found it did so at high shear rates, through suppressing and replacing Newtonian turbulence. It also demonstrated that EIT was dominated by elastic stresses, and that their balance with inertial stresses was key in determining the onset of EIT. More recent advances include the use of Direct Numerical Simulations (DNS) for a Finitely Extensible Nonlinear Elastic fluid under the Peterlin approximation (FENE-P) in the EIT regime [2]. This is essentially simulating visco-elastic fluids in the conditions where EIT is found to occur. The first is a paper by Sid, Terrapon, and Dubief [30] which confirmed the existence of two dimensional elasto-inertial instabilities and highlighted the role of small elastic scales in sustaining EIT. The second paper, by Dubief et al. [14], used DNS and found a coherent structure in EIT, which sustain turbulence through a balance of elastic and inertial stresses, and hence maintain EIT. There has thus been significant progress in understanding EIT. It is first dominated by elastic stresses when there is

high polymer stretch and shear rates. Second are the inertial stresses particularly in the onset of EIT, and the viscous stresses matter less [14, 28, 30]. However, Dubief et al. [14] still leaves some questions on the dynamics and interactions within EIT's coherent structures unanswered. This is the aim of using the Callaham et al. method on DNS data of EIT [2].

5.2 Methodology

The data for this case study is obtained from a two-dimensional DNS of the FENE-P model. This model has the following governing equations:

$$\begin{aligned} \partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p &= \frac{\beta}{Re}\Delta\mathbf{u} + \frac{1-\beta}{Re}\nabla\cdot\mathbf{T}(\mathbf{C}), \\ \partial_t \mathbf{C} + (\mathbf{u}\cdot\nabla)\mathbf{C} + \mathbf{T}(\mathbf{C}) &= \mathbf{C}\cdot\nabla\mathbf{u} + (\nabla\mathbf{u})^T\cdot\mathbf{C} + \frac{1}{Re\cdot Sc}\Delta\mathbf{C}, \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{5.1}$$

where:

$$\begin{aligned} \mathbf{T}(\mathbf{C}) &= \frac{1}{\mathrm{Wi}}\left(f(\mathrm{tr}\mathbf{C})\mathbf{C} - \mathbf{I}\right), \\ f(x) &= \left(1 - \frac{x-3}{L_{\max}^2}\right)^{-1} \end{aligned} \tag{5.2}$$

$\mathbf{T}(\mathbf{C})$ is the Polymer Stress Tensor, $\mathbf{u}$ is the velocity field, $\mathbf{C}$ is the conformation tensor, and p is the pressure. This simulation was carried out in the context of the Beneitez et al. (2024) paper [2], and the data was obtained from the authors. In that study, the dimensionless parameters were set to $\beta = 0.9$, $Re = 1000$, $Sc = 500$, and $Wi = 50$, and L_{max}, the polymer extensibility was set to vary between [70, 130]. These simulations aimed to explore the dynamical connections betweem the arrowhead structure discovered in Dubief et al. (2022) [14] and EIT. Results showed there being 4 coexisting regimes (called attractors): a laminar state, a Steady Arrowhead Regime (SAR), EIT and a Chaotic Arrowhead Regime (CAR)(see Fig. 5.1).

For this case study, focus was consecrated to the CAR regime, obtaining a series of 512 by 512 images (snapshots) of the velocity components, pressure and its time

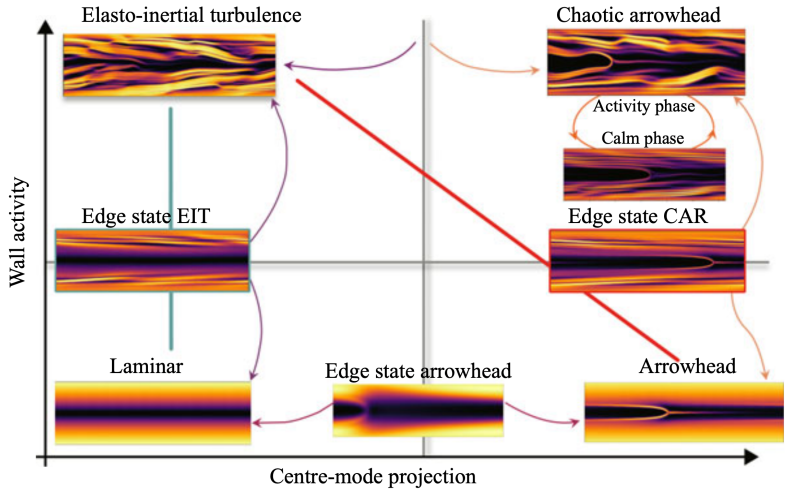


Figure 5.1: Plot of the attractors found in the DNS of the FENE-P model. The Elasto-Inertial Turbulence (EIT) (top-left), the Chaotic Arrowhead Regime (CAR) (top-right), the laminar state (bottom-left), and the Steady Arrowhead Regime (SAR) (bottom-right). There were also some edge-states found in the simulations, which are found in between different basins of attraction in the state-space. Image from Beneitez et al. (2024) [2]

variant gradient, and the conformation tensor's 4 components through time. With these variables, the terms can be computed. However, to avoid approaching this problem too bluntly with the full set of equation, only the streamwise component of the first equation of Eq. (6.1) is studied (see Eq. (6.3)). This simplifies the problem and still gives us the opportunity to test the usefulness of the Callaham et al. method in a more complex, less studied flow.

$$\begin{aligned}\partial_t u + uu_x + vu_y + p_x = \frac{\beta}{Re}(u_{xx} + u_{yy}) \\ + \frac{1-\beta}{Re}\left[\left(\frac{1}{Wi}\left(1 - \frac{\mathrm{tr}\mathbf{C} - 3}{L^2_{\max}}\right)^{-1} C_{xx} - 1\right)_x \right. \\ \left. + \left(\frac{1}{Wi}\left(1 - \frac{\mathrm{tr}\mathbf{C} - 3}{L^2_{\max}}\right)^{-1} C_{xy}\right)_y\right]\end{aligned} \tag{5.3}$$

Once the terms are computed, these are used as features for each point in space. As discussed in the Stability Assessment section, choosing the hyperparameters for

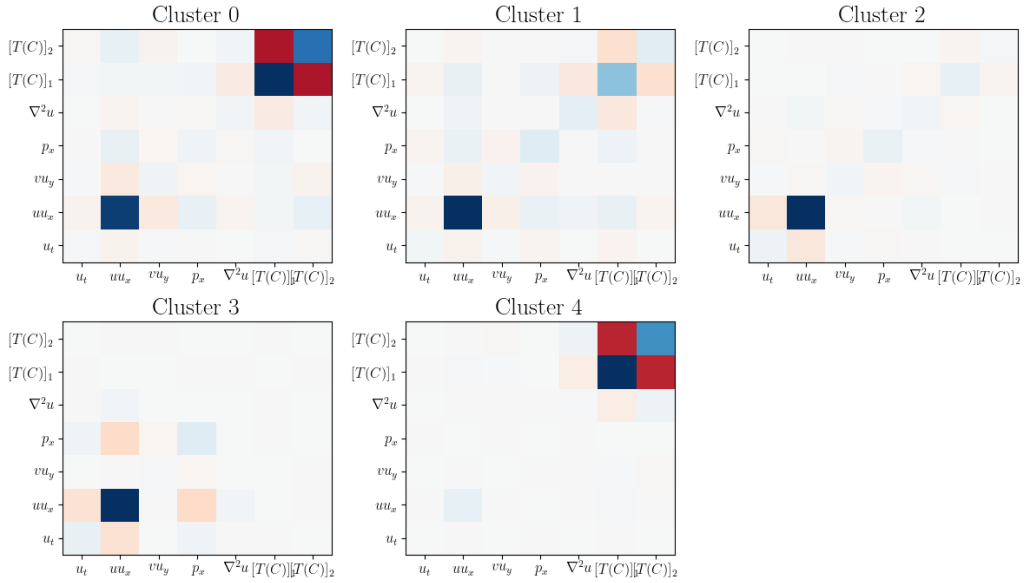
this case study is uncertain. Following the ordering of the method, the number of clusters is first chosen. The reasoning behind this step was to get the covariance matrices of the clusters for an increasing number of clusters, and then choose the number of clusters for which clusters start to have repeated covariance matrices, therefore indicating that most of the different balance models have likely been identified.

The results of the GMM algorithm for the EIT case study are shown in Figures 5.2 and 5.3. From these covariance matrices, we see that between 9 and 11 clusters, only repeats are obtained, with no clusters having new or at the least significantly different covariance matrices than in the 9 clusters case. So 9 clusters was the chosen number of clusters to find.

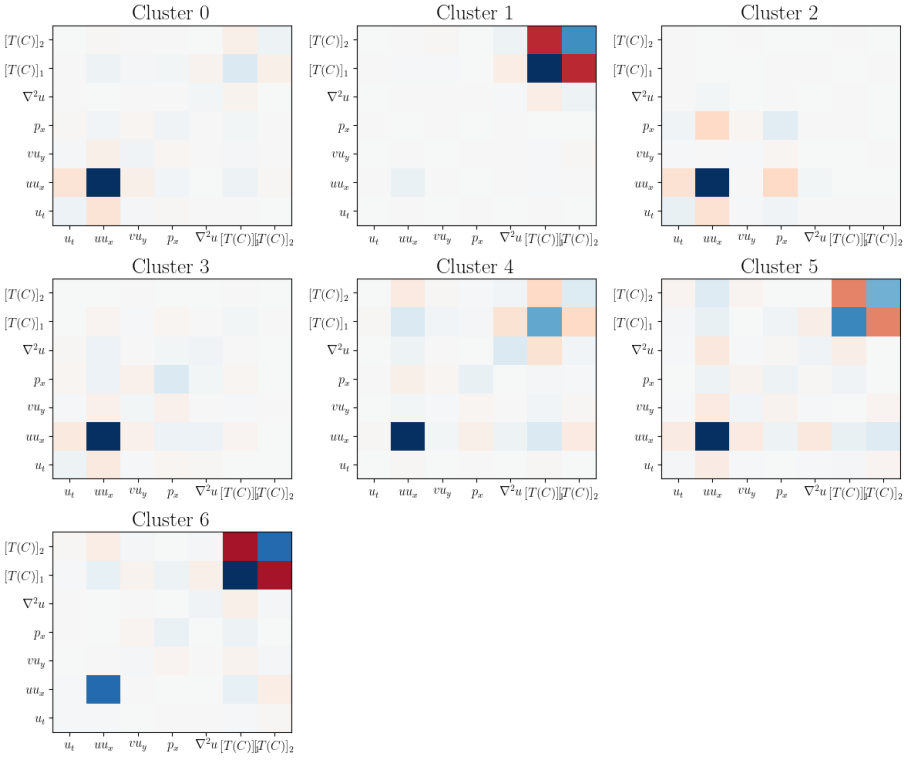
Choosing the value for α is a little more ambiguous. A more brute force approach is chosen, simply looking at the multiple results for a range of α values. The range itself is chosen to be in the trade-off region between sparsity and accuracy, as seen in the residuals plot for the 9 clusters case (see Fig. 5.4).

Now based on the 9 covariance matrices (see Fig. 5.3(a)), it would not be unreasonable to expect to find 4 to 7 unique balance models. Looking at the balance models found for the different alpha values in Fig. 5.5,

Clearly there is a lot of variation between $\alpha = 1$ and $\alpha = 2$, with the number of unique balance models decreasing from 8 to 4, and results for multiple α values will be studied and observed.

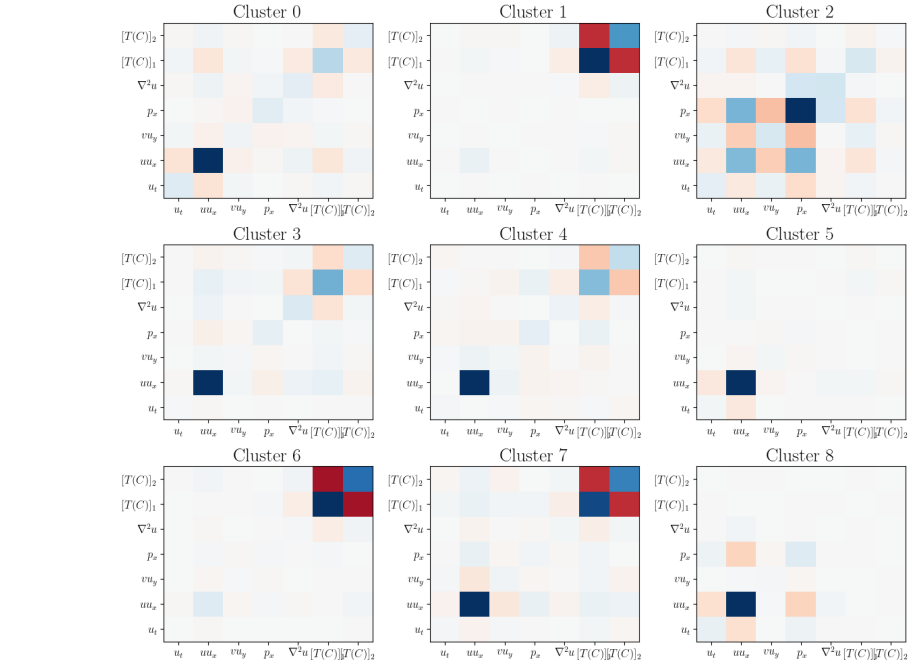


(a)

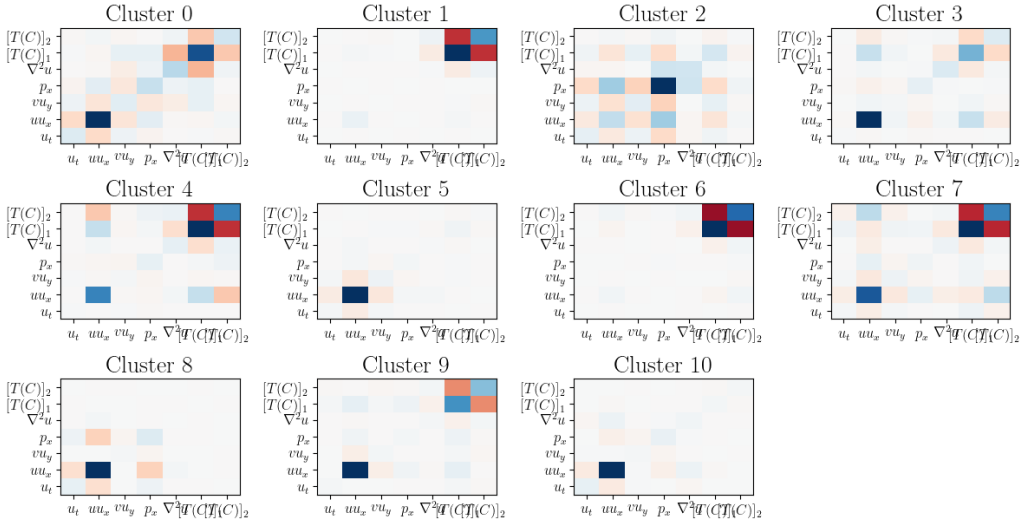


(b)

Figure 5.2: Covariance matrices of the clusters found by the GMM algorithm for 5 (a) and 7 (b) clusters.



(a)



(b)

Figure 5.3: Covariance matrices of the clusters found by the GMM algorithm for 9 (a) and 11 (b) clusters.

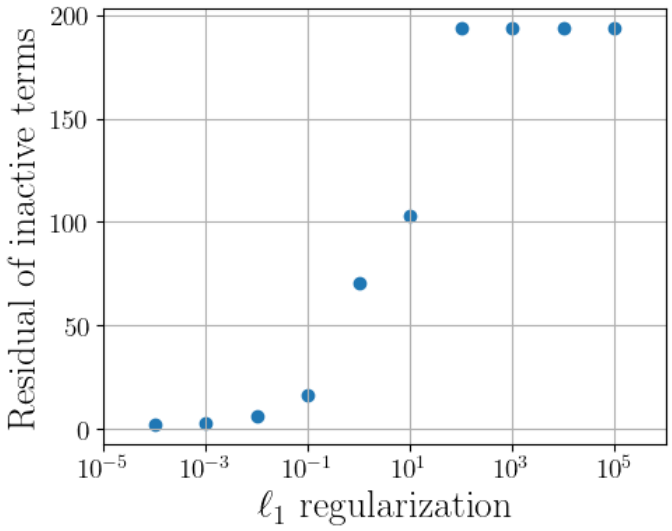


Figure 5.4: Plot of the SPCA residuals for different values of α for the EIT case. Recall, the residuals are computed as the ℓ-2 norm of the inactive terms for all clusters

5.3 Results

First, the data contained multiple snapshots and trajectories so the 15$^{\text{th}}$ snapshot of the 5$^{\text{th}}$ trajectory was chosen to be studied. For visualisation purposes, a plot of the first diagonal component of the comformation tensor, C_{xx}, is shown in Fig. 5.6. In that plot, the chaotic arrowhead structure is clearly visible.

Getting the terms for the streamwise component of the velocity equation did raise one issue, which was the value of the residuals of the equation. By putting all terms on the same side of the equation, the residuals were computed and plotted in Fig. 5.7. As can be seen the residuals are overall low but the colormap shows that there some peak regions which reach order of magnitude 1. And compared to the terms' orders of magnitude (1, 10^{-1}, 10^{-2}), this is quite high. There are a couple of possible reasons for this, the first being the computing of the time derivative terms which relies on using a 2$^{\text{nd}}$ order finite difference scheme, using the previous and next snapshots. This is a fairly strong assumption as one time step in these simulations are consequent. The second can be attributed to cumulative errors in the other derivatives. On the same note, the data ussed does have a spectral version, which could possibly have smaller cumulative errors when differentiating, so this could be

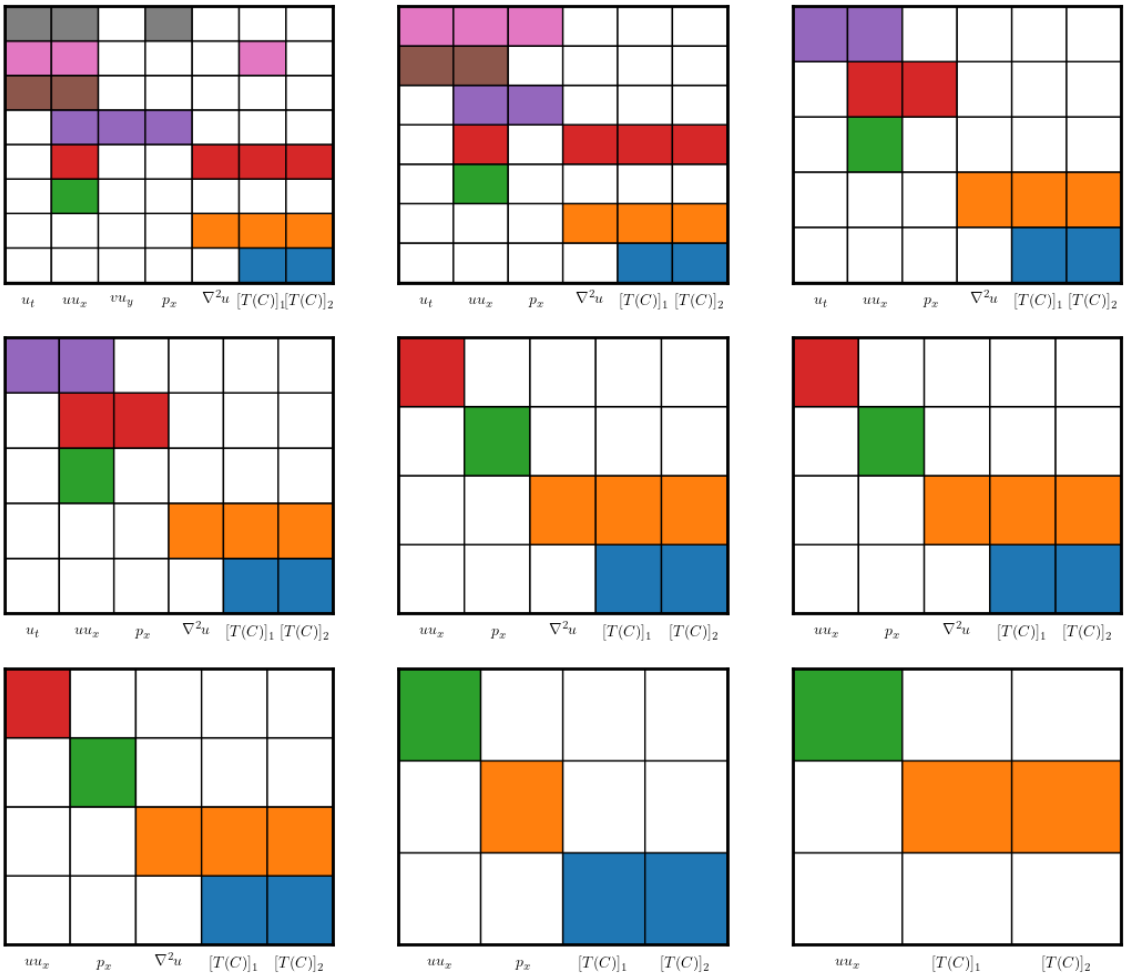


Figure 5.5: Plot of the obtained balance models for different ℓ-1 regularization term factor, α. Reading top to bottom and left to right, the values are 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 3.5, 4

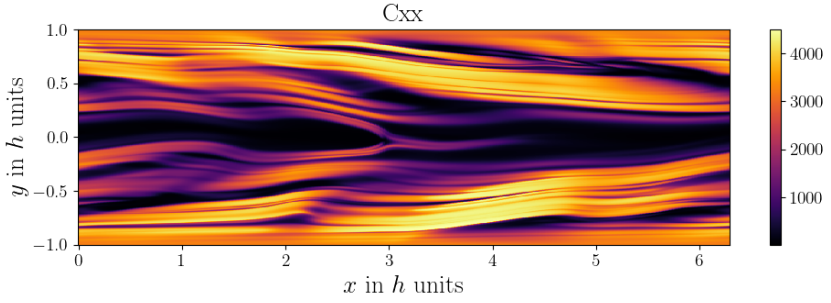


Figure 5.6: Plot of the first diagonal component of the conformation tensor, C_{xx}, for the 15^{th} snapshot of the 5^{th} trajectory.

a possible direction to explore in future work. Overall this does not necessarily invalidate our results but does mean that caution must be taken when interpreting the results.

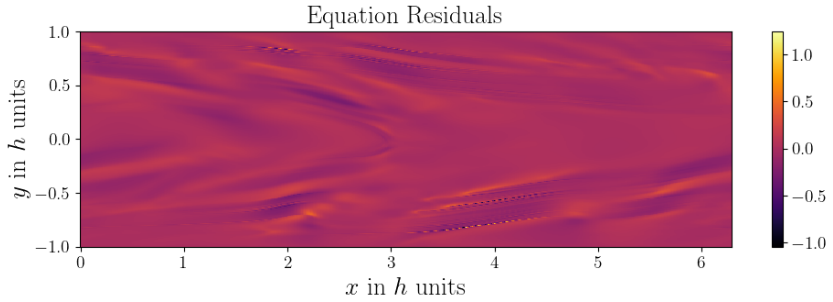


Figure 5.7: Plot of the residuals of the streamwise component of the FENE-P model equation for the EIT case study.

Having chosen the number of clusters to be 9, the clustering in space was obtained and is plotted in Fig. 5.8. Though with the large number of clusters, it is hard to discern any visible patterns yet.

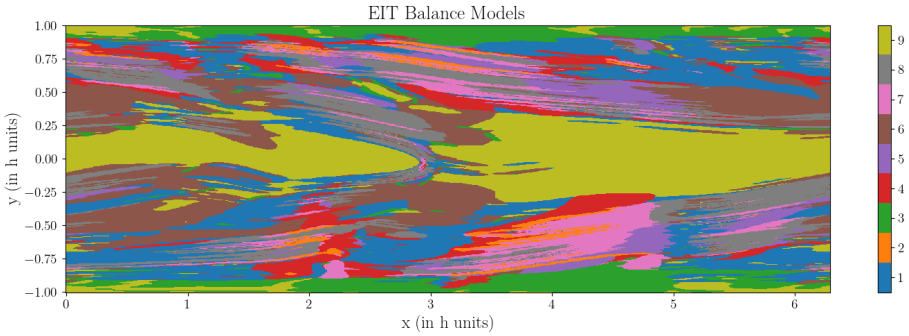
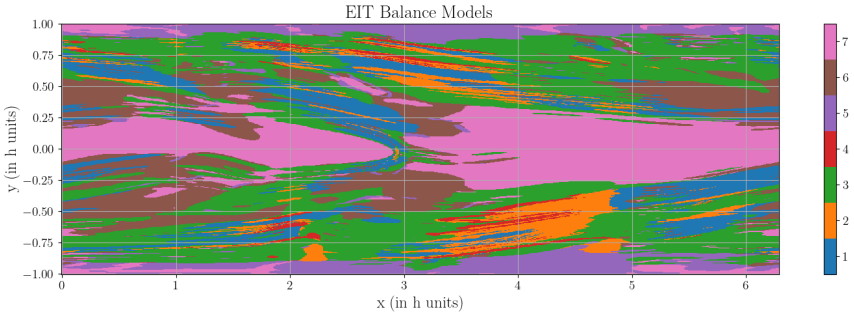
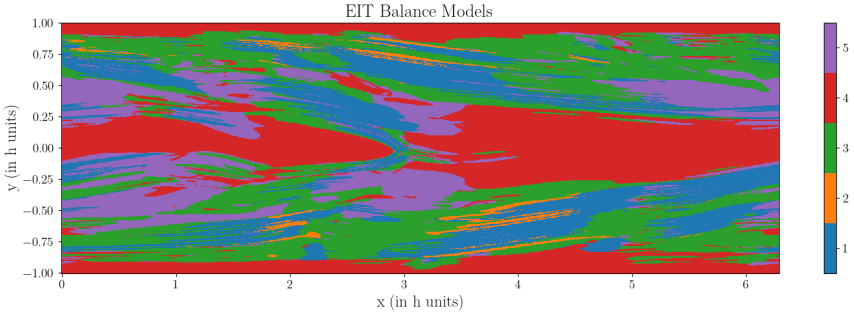


Figure 5.8: Plot of the clustering in space for $n_{clusters} = 9$ for the EIT case study.

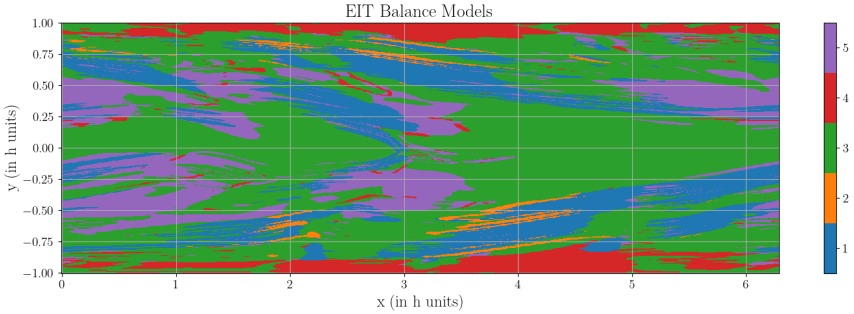
Moving on to getting the balance models, Fig. 5.5 already shows the balance models obtained for different α values. Recalling our expected number of balance models to be between 4 and 7, based on Fig. 5.3, balance models in space were obtained for values of alpha between 1.25 and 2. The results are shown in Fig. 5.9. For clarity, the associated balance models are shown in Fig. 5.10.



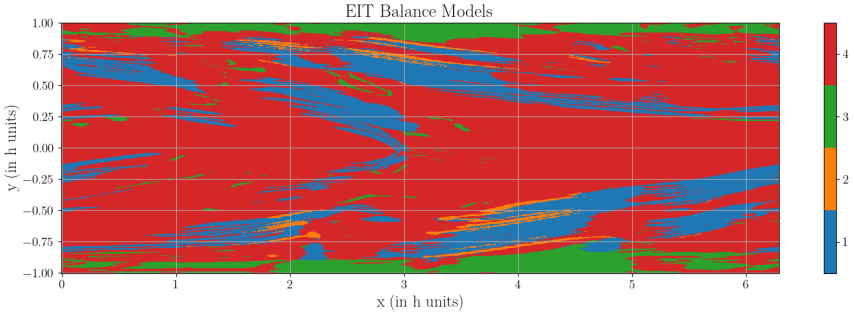
(a)



(b)

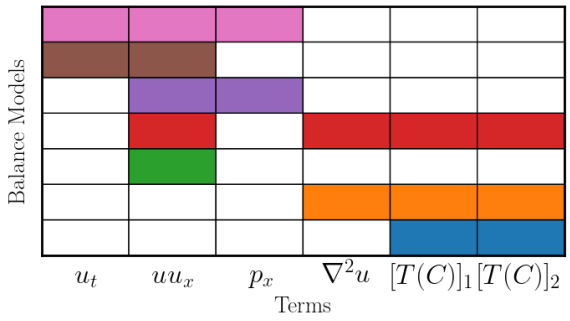


(c)

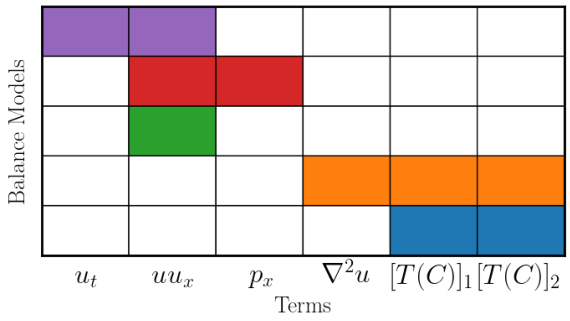


(d)

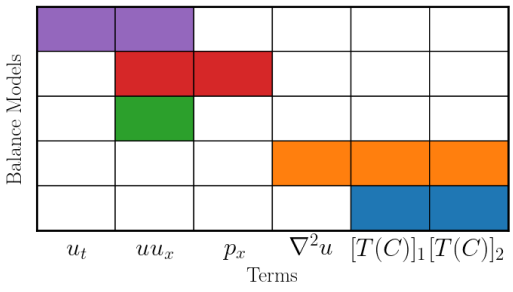
Figure 5.9: Plot of the unique balance models in space found for α values of 1.25 (a), 1.5 (b), 1.75 (c), and 2 (d).



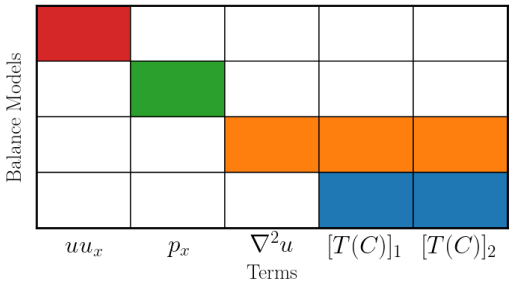
(a)



(b)



(c)



(d)

Figure 5.10: Plot of the unique balance models found for α values of 1.25 (a), 1.5 (b), 1.75 (c), and 2 (d).

5.4 Discussion

Overall, Fig. 5.9 and 5.10 show that there is a clear 2 or 3 groups of dominant balance regimes. The first are the clusters which have neither elastic or viscous stresses (pink, brown, purple, red except for $\alpha = 1.25$, and green). These occupy the majority of the channel, being primarily along the walls or the centre of the channel, with the exception of the where the arrowhead is. The second group are the orange and blue clusters which are associated with regions with strong elastic and viscous stresses, with the latter only being in the orange cluster. From literature [2, 14, 28], this fits with conditions for which EIT is maintained. In terms of location, elastic stresses are localised to the sides of the channel, and where the arrowhead structure is. The viscous stresses are more localised, only on the sides of the channels for x positions that are aligned with where the arrowhead structure stops.

Chapter 6

Conclusion

Thus, this report has evaluated the reproducibility and robustness of the method proposed by Callaham et al. (2021). By reproducing results with alternative code and comparing them with the original code's results, this project has confirmed the validity of the original findings while uncovering some limitations and areas for improvement. By testing other clustering algorithms, the choice of the GMM algorithm was reinforced, demonstrating its effectiveness in identifying dominant balance regimes for a variety of datasets. Furthermore, the stability analysis provided valuable insights into the sensitivity of the method to hyperparameters, highlighting the importance of careful selection to ensure robust and reliable results.

The exploration of elasto-inertial turbulence (EIT) further highlighted the flexibility of the method, identifying plausible dominant balance regimes and providing insights into the dynamics of the flow. Despite encountering some challenges with code portability and data consistency, the overall method is simple to implement and can be applied to a wide range of physical systems.

The application of the method to new datasets suggests promising avenues for future research, particularly in extending the approach to other types of flows and refining the algorithm for better accuracy and efficiency. Ultimately, this work contributes to the broader effort of leveraging data-driven techniques to enhance our understanding and modeling of complex physical phenomena.

Chapter 7

Appendix

7.1 Algorithms

Algorithm 1: Gaussian Mixture Model clustering

Data $\vec{x} \in \mathbb{R}^{n \times n_{features}}$, number of clusters/Gaussians to fit K ▷ Inputs
Cluster assignments $z \in \mathbb{R}^n$ ▷ Output
$w_k \leftarrow \frac{1}{k}$ ▷ Initialisation of the weights
$\mu_\mathbf{k} \leftarrow \mathbf{c_k}$ ▷ Initialise the means as K-Means centroids
$\mathbf{\Sigma_k} \leftarrow \mathbf{\Sigma}_{C_k}$ ▷ Initialise as the covariance matrices of the clusters
$C_k \leftarrow w_k \times \mathcal{N}(\mu_\mathbf{k}, \mathbf{\Sigma_k})$ ▷ Initialise the clusters/Gaussians
Expectation-Maximisation:
$\mathcal{L} \leftarrow 0$ ▷ Initialise the log-likelihood
for $i <$ max-iter **do**
    Expectation step:
    for $k \in K$ **do**
        $\mathbf{b_k} \leftarrow \frac{w_k \times f(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum_{j=1}^{K} w_j \times \mathcal{N}(\mathbf{x}|\mu_\mathbf{j}, \mathbf{\Sigma_j})}$ ▷ Compute the responsibilities for each cluster
    end for
    Maximisation step:
    for $k \in K$ **do**
        $w_k \leftarrow \frac{1}{n} \sum_{i=1}^{n} \mathbf{b}_k$ ▷ Update the weights
        $\mu_\mathbf{k} \leftarrow (\sum \mathbf{b}_k \mathbf{x}) / \sum \mathbf{b}_k$ ▷ Update the means
        $\mathbf{\Sigma_k} \leftarrow (\sum \mathbf{b}_k (\mathbf{x} - \mu_k)^2) / \sum \mathbf{b}_k$ ▷ Update the covariance matrices
    end for
    $\mathcal{L}_{\text{new}} \leftarrow \sum_{i=1}^{n} \log(\sum_{k=1}^{K} w_k \times f(x_i|\mu_\mathbf{k}, \mathbf{\Sigma_k}))$ ▷ Compute the log-likelihood
    $\epsilon \leftarrow \mathcal{L} - \mathcal{L}_{\text{new}}$ ▷ Compute the difference in log-likelihood

22: **if** $\epsilon < 10^{-4}$ **then**
23: **break**
24: **end if**
25: $\mathcal{L} \leftarrow \mathcal{L}_{\text{new}}$ ▷ Update the log-likelihood
26: **end for**

Where:
- $f(\mathbf{x}|\mu_{\mathbf{k}}, \mathbf{\Sigma_k})$ is the probability density function of a multivariate normal distribution

Algorithm 2: Sparse PCA Algorithm

Data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ (n: samples, p: features), number of components k, sparsity controlling parameter α, maximum iterations max_iter, tolerance tol ▷ Inputs

Sparse components $\mathbf{V} \in \mathbb{R}^{p \times k}$ ▷ Outputs

Initialize $\mathbf{V} \leftarrow$ random, $\mathbf{U} \leftarrow$ random ▷ Initialization

Initialization:

Center the data matrix $\mathbf{X}$ by subtracting the mean of each column

Initialize $\mathbf{V}$ with random values

for $i = 1$ to max_iter **do** ▷ Iterate to optimize components and loadings

  Update Loadings:

  for $j = 1$ to k **do** ▷ Update each loading vector

    $\mathbf{u}_j \leftarrow \arg\min_{\mathbf{u} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \alpha\|\mathbf{u}_j\|_1$

  end for

  Update Components:

  for $j = 1$ to k **do** ▷ Update each component vector

    $\mathbf{v}_j \leftarrow \arg\min_{\mathbf{v} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \alpha\|\mathbf{v}_j\|_1$

  end for

  Convergence Check:

  Compute the reconstruction error: error $\leftarrow \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F$

  if error $<$ tol **then**

    break

  end if

end for

Return: Sparse components $\mathbf{V}$

In practice, `sklearn` uses a dictionary learning for sparse coding [23], and structured sparse PCA [16] for the sparse PCA algorithm. But both are attempting to solve the optimization problem: $(U^*, V^*) = \arg_{U,V} \min \frac{1}{2}||X - UV||_{\text{Fro}}^2 + \alpha||V||_{1,1}$subject to $||U_k||_2 <= 1$ for all $0 \leq k < n_{components}$

Algorithm 3: Spectral Clustering Algorithm

1: Given data points $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ ▷ Inputs

2: Construct similarity graph $G = (V, E)$ where each vertex v_i represents a data point $\vec{x}_i$. ▷ Graph construction

3: Choose a similarity function, here, k-nearest neighbours: Connect v_i and v_j if one of these is among the k-nearest neighbours of the other.

4: **Step 1: Compute Laplacian**

5: Construct the adjacency matrix A where:

$$A_{ij} = \begin{cases} 1 & \text{if } i, j \in E, \text{ the Edge set} \\ 0 & \text{otherwise} \end{cases}$$

6: Construct the degree matrix D where $D_{ii} = \sum_j A_{ij}$

7: Construct the unnormalized Laplacian $L = D - A$

8: **or** construct the normalized Laplacian $L_{\text{sym}} = I - D^{-1/2} A D^{-1/2}$

9: **Step 2: Compute Eigenvectors**

10: Compute the first k eigenvectors $u_1, u_2, \ldots, u_k$ of L **or** L_{sym}

11: Form the matrix $U \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns

12: **Step 3: Cluster in embedding space**

13: Apply k-means clustering to the rows of U to obtain clusters $C_1, C_2, \ldots, C_k$

14: **Output:** Clusters $A_1, A_2, \ldots, A_k$ where $A_i = \{j | \vec{u}_j \in C_i\}$

This is for an unweighted undirected graph. [32]

7.2 Extra Plots

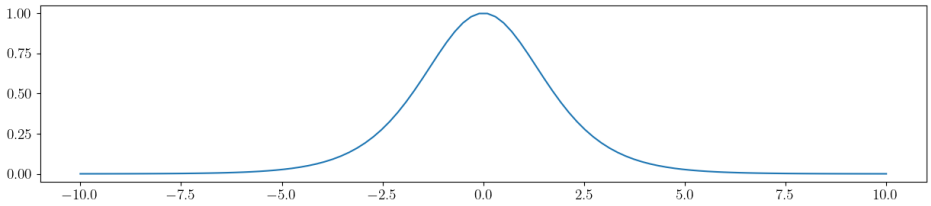
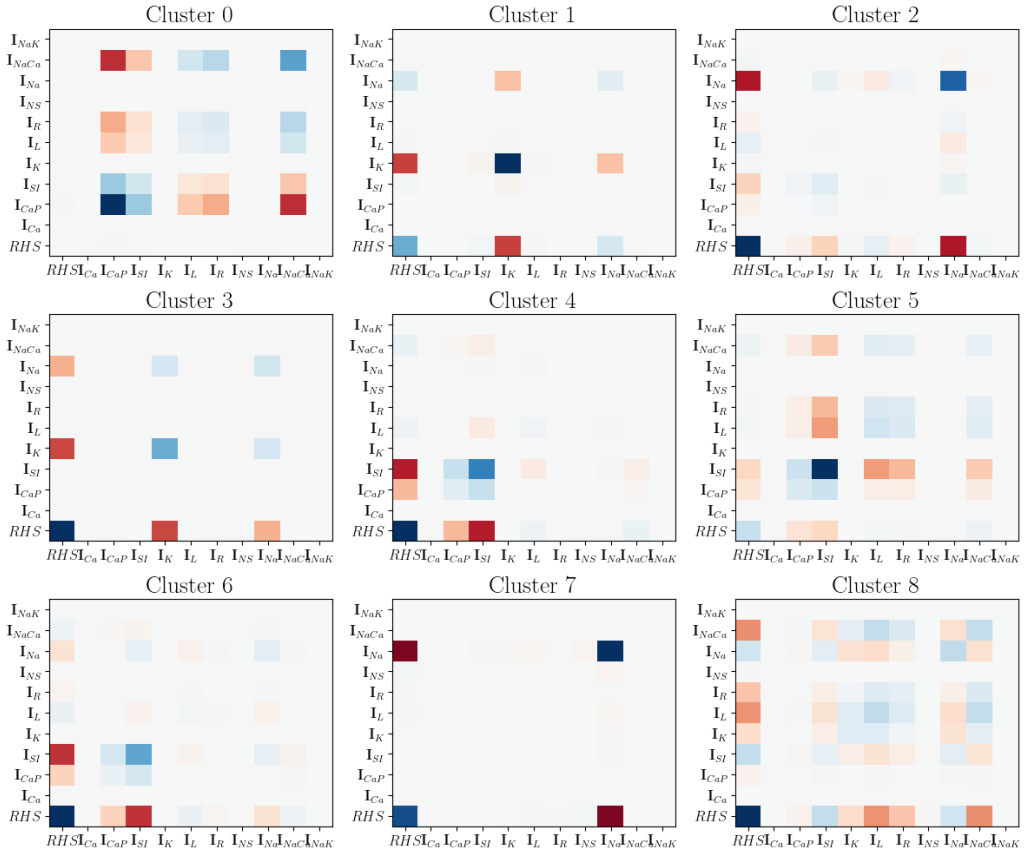
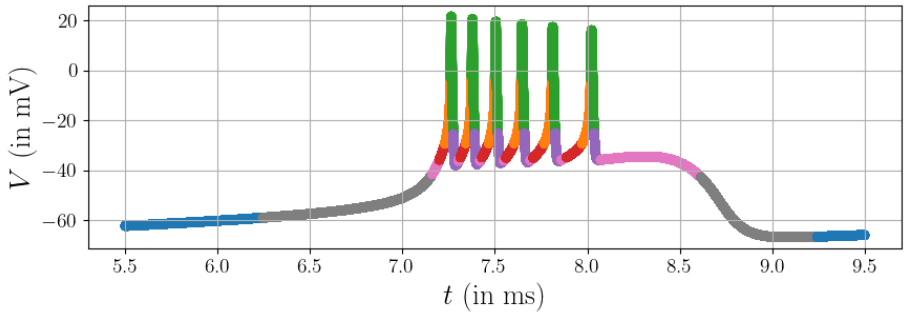


Figure 7.1: Plot of the sample weight function applied to the turbulent boundary layer data when using the weighted K-measn algorithm. The function is a hyperbolic tangent function with a maximum value of 1 and a minimum value of 0. This is a function of the distance of the sample from the origin in equation space.

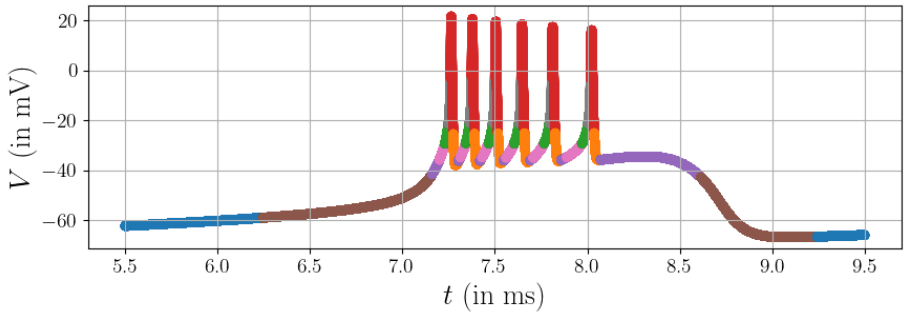


(a)

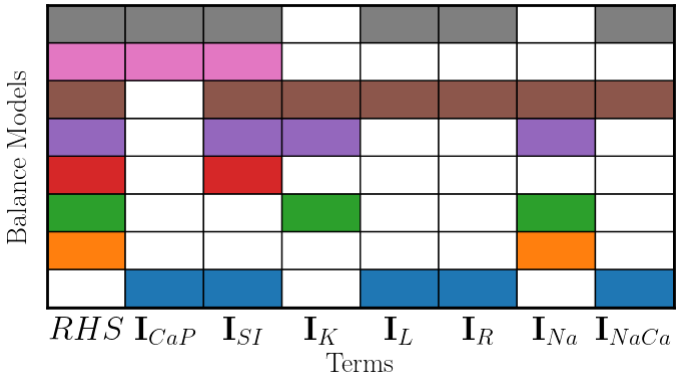


(b)

Figure 7.2: Results for the Bursting R15 Neuron case. **(a)** Covariance matrices for each of the clusters found by the GMM algorithm. **(b)** Plot of the clusters found by the GMM algorithm in physical space.

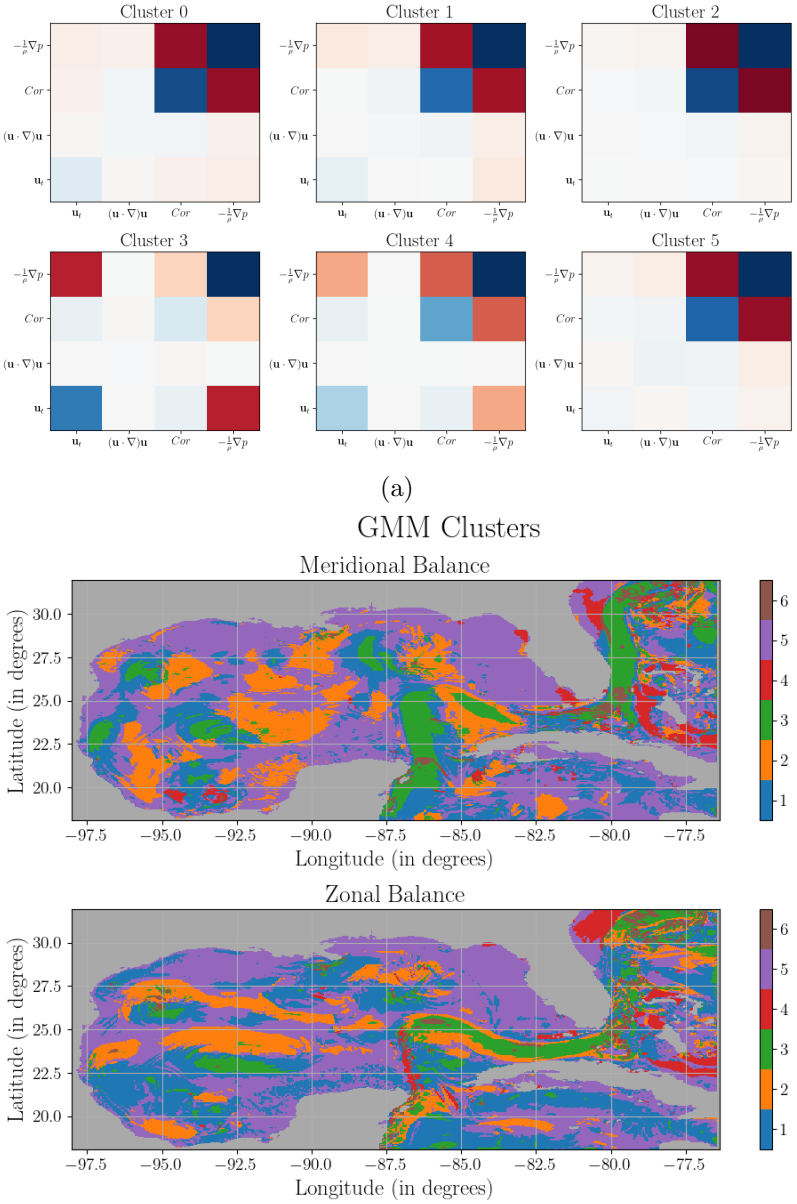


(a)



(b)

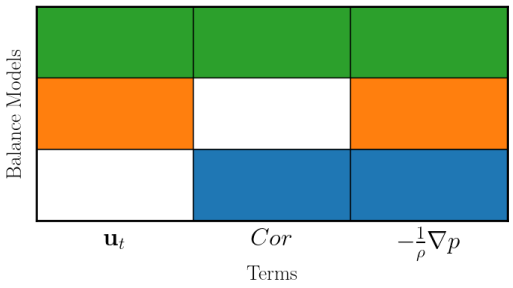
Figure 7.3: Second part of the results for the Bursting R15 Neuron case. **(a)** Plot of the clusters found by the GMM algorithm. **(b)** Plot of the unique balance models in t-V space found after applying SPCA. The spikes start with the voltage being balanced by sodium currents only in the rising part (orange), then sodium and potassium at the peak (green), and then with the slow inward Calcium current added (purple), which differs from the paper's results. Overall, the peak-cluster is too large and should stay in the positive mV region. The grey and blue clusters follow the same dynamics as in the paper, where mostly calcium dependence terms dominate.



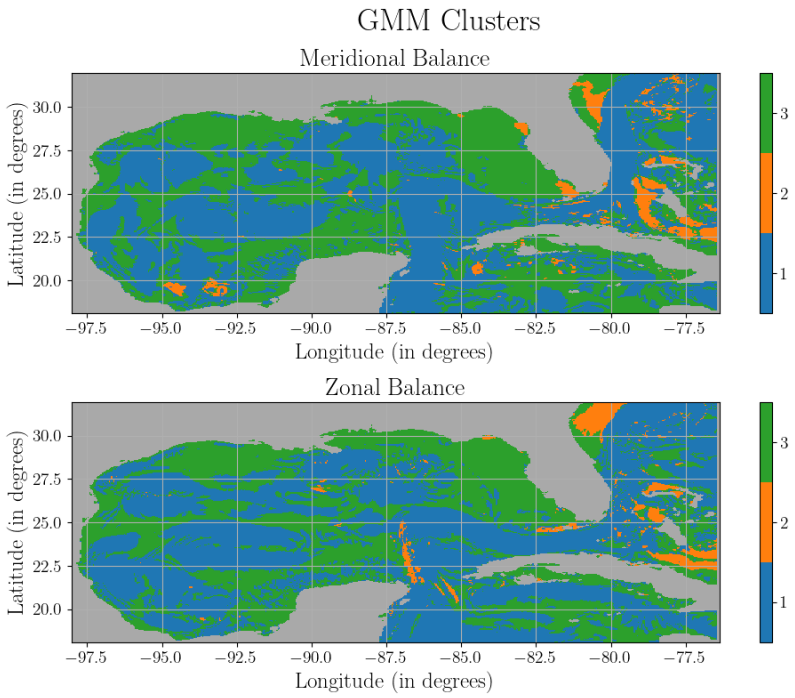
(a)

(b)

Figure 7.4: Results for the geostrophic balance in oceanic currents case. **(a)** Covariance matrices of for each of the clusters found by the GMM algorithm. **(b)** Plot of the clusters found by the GMM algorithm.

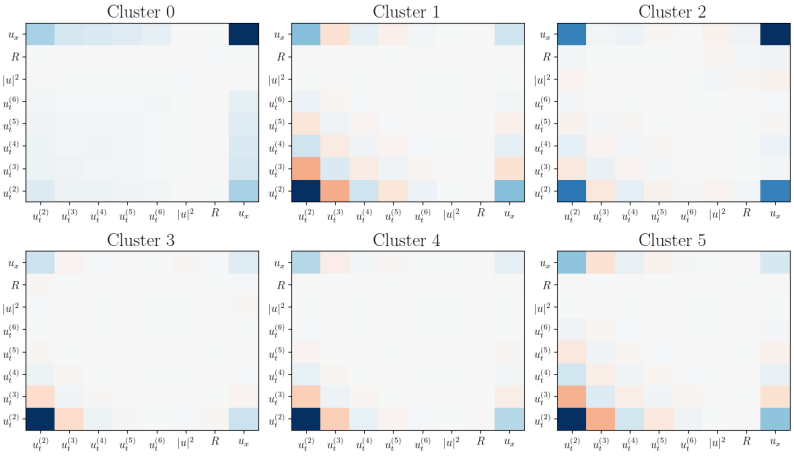


(a)

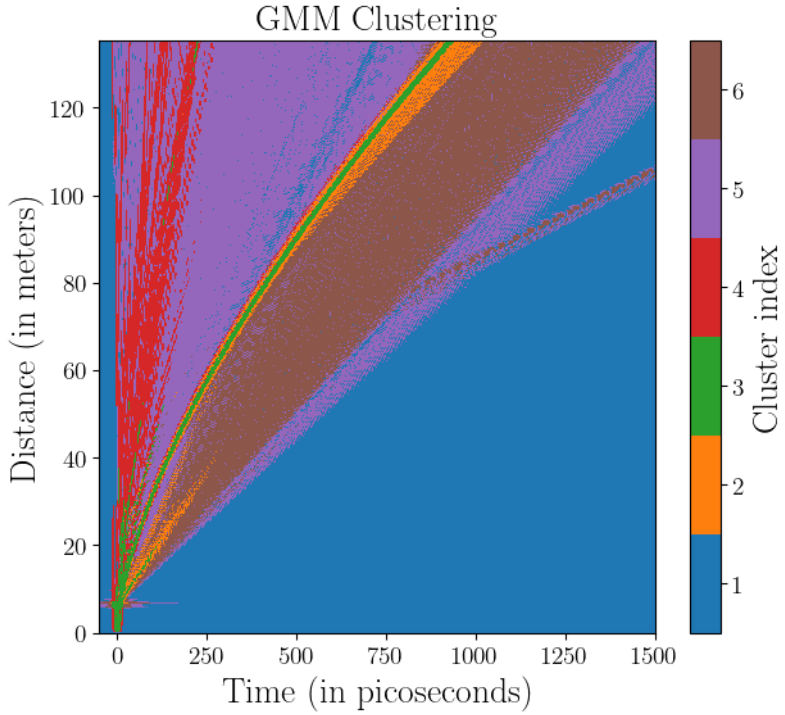


(b)

Figure 7.5: Second part of results for the geostrophic balance in oceanic currents. **(a)** Plot of the unique balance models found after applying SPCA. **(b)** Plot of the unique balance model clusters in physical space. Thus similar results to the paper's are obtained, with the geostrophic balance (blue) holding where there are vortices. The main difference with the paper is the number and size of these regions which is due to the fact that the snapshots used (in time and date) are not the same, so conditions were different..

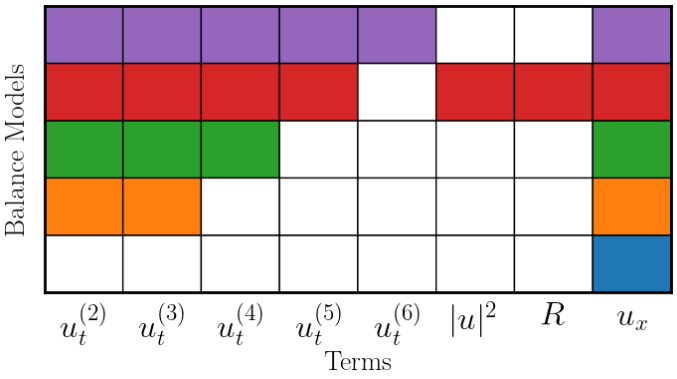


(a)

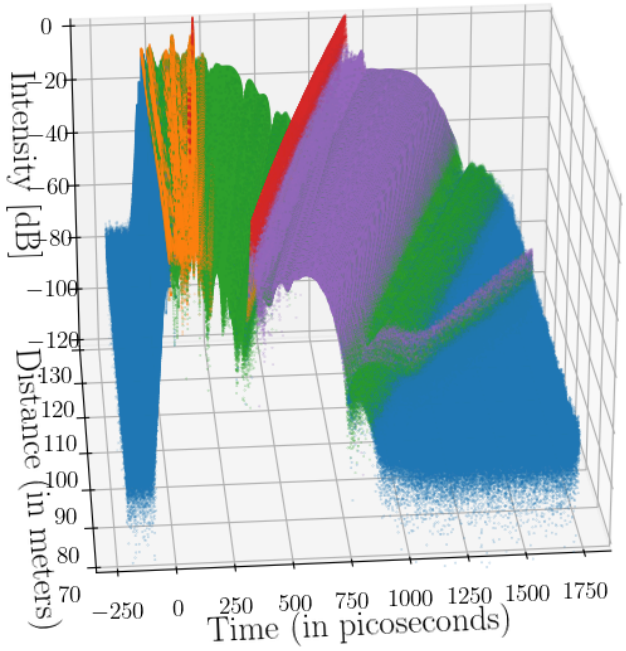


(b)

Figure 7.6: Results for the optical pulse case. **(a)** Covariance matrices of for each of the clusters found by the GMM algorithm. **(b)** Plot of the clusters found by the GMM algorithm.



(a)



(b)

Figure 7.7: Second part of results for the optical pulse case. **(a)** Plot of the unique balance models found after applying SPCA. **(b)** Plot of the unique balance model clusters in 3D projection of physical space, with height given by the intensity of the solitons. Overall, similar dominant balances were found. The strongest soliton identified balance (red), which was found in the paper to be the only region where the instantaneous nonlinear response was identified as active, though with an extra active term (Raman term wihtout cubic Kerr nonlinearity). But it is still the only identified balance with just the nonlinearity term active.

7.3 AI Use Declaration

For the coding of this project, the following AI tools were used:

1. GitHub Copilot's autogeneration tool was used when writing the documentation and the comments for the code. It was also used in the case of repetitive parts of the code, taking the example of the stability assessment python file, where the same loop is essentially used 3 times but with a different varying parameter each time, or for the large plot of the terms fields in the EIT Notebook. It was also used extensively to write the bibtex references by giving it links or copy pasting the pages of articles.

2. ChatGPT was used:

   - To generate a detailed plan with recommended word counts for the executive summary, and a list of recommended figures to include. The plan generated only allocated 50 words to the turbulent boundary layer reproducing section which was not enough so the plan was modified to allocate a lot more words to that section.

   - It was also used to provide summaries for the literature on EIT.

   - It was asked for recommendations on if any other library other than sklearn would have a SPCA implementation: It recommended SPAMS [22], by the same authors as the dictionary learning algorithm used in sklearn [23]. This was not used in the end as the documentation was not clear on how to use it, and it seemed to require 2 complexity parameters, which would have made it harder to use than sklearn's implementation, on top of losing the quality of having not many hyperparameters to tune. It also recommended the mlxtend library, however, for this it seemed to simply lack an implementation of SPCA.

   - Extensive use was made when debugging issues with the Containerisation of this project. Because of the packages installed in the environment such as clawpack, mlxtend or spams, there were a lot of conflict issues which were hard to resolve as Docker error messages can be a bit obscure.

   - It was used in the proof reading of the report, giving paragraphs of the report one at a time and asking it to give me where spelling mistakes were.

- When trying to plot the balance models, which uses latex formulations, running this on the docker brought up a matplotlib error saying it failed to process a string because latex could not be found. ChatGPT suggested installing missing latex packages when building the docker image: "dvipng texlive-latex-extra texlive-fonts-recommended cm-super". This fixed the issue.

- It was also used early on when trying to understand how the scipy.sparse library functionned, asking it to explain step by step how the sparse matrices were being built in the original Callaham et al. code.

- In general, it was used often by simply giving it the traceback of an error, which would usually return an explanation of what the error was pointing to.

Bibliography

[1] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. volume 8, pages 1027–1035, 01 2007.

[2] Miguel Beneitez, Jacob Page, Yves Dubief, and Rich R. Kerswell. Multistability of elasto-inertial two-dimensional channel flow. *J. Fluid Mech.*, 981:A30, 2024.

[3] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[4] A. P. Burger. Scale consideration of planetary motions of the atmosphere. *Tellus*, 10(2):195–205, 1958.

[5] J.L. Callaham, J.V. Koch, and B.W. et al. Brunton. Learning dominant physical processes with data-driven balance models. *Nature Communications*, 12:1016, 2021.

[6] J. G. Charney. The dynamics of long waves in a baroclinic westerly current. *Journal of Atmospheric Sciences*, 4:136–162, 1947.

[7] J.G. Charney. On the scale of atmospheric motions. In R.S. Lindzen, E.N. Lorenz, and G.W. Platzman, editors, *The Atmosphere — A Challenge*. American Meteorological Society, Boston, MA, 1990.

[8] E. P. Chassignet, H. E. Hurlburt, O. M. Smedstad, G. R. Halliwell, P. J. Hogan, A. J. Wallcraft, R. Baraille, R. Bleck, J. A. Carton, P. Cornillon, E. Curchitser, G. Danabasoglu, J. Dukowicz, Y. Fujii, S. M. Griffies, R. W. Hallberg, R. L. Haney, A. R. Karspeck, S. Legg, C. M. Little, A. J. Miller, S. J. Marsland, A. Pirani, H. Regan, C. A. Scholz, W. H. F. Smith, H. L. Tolman, and E. D. Zaron. Hycom gulf of mexico 1/25° analysis and forecast system, 2009.

[9] Will Chen. How to code gaussian mixture models from scratch in python. *Towards Data Science*, 2020.

[10] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[11] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*, 2020.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[13] T. Driscoll. Fundamentals of numerical computation.

[14] Y. Dubief, J. Page, R. R. Kerswell, V. E. Terrapon, and V. Steinberg. First coherent structure in elasto-inertial turbulence. *Phys. Rev. Fluids*, 7:073301, Jul 2022.

[15] P. F. Fischer, J. W. Lottes, S. G. Kerkemeier, A. Cary, E. Merzari, A. Obabko, A. Siegel, P. Moinier, P. Vincent, H. Vincenti, et al. Nek5000: An open-source spectral element code for high-order simulation of turbulent flows, 2019.

[16] Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. *INRIA Willow Project, Laboratoire d'Informatique de l'Ecole Normale Superieure*, 2010.

[17] joblib contributors. joblib.parallel. `https://joblib.readthedocs.io/en/stable/generated/joblib.Parallel.html`, 2022.

[18] J. Lee and T. A. Zaki. The jhu turbulence databases (jhtdb) - transitional boundary layer data set, 2020.

[19] Jin Lee and Tamer A. Zaki. Detection algorithm for turbulent interfaces and large-scale structures in intermittent flows. *Computers & Fluids*, 175:142–158, 2018.

[20] Fernando Lejarza and Michael Baldea. Data-driven discovery of the governing equations of dynamical systems via moving horizon optimization. *Scientific Reports*, 12(1):11836, 2022.

[21] Jake Lever, Martin Krzywinski, and Naomi Altman. Principal component analysis. *Nature Methods*, 14(7):641–642, 2017.

[22] Julien Mairal. Spams: Sparse modeling software. `https://thoth.inrialpes.fr/people/mairal/spams/`, 2014.

[23] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. *Journal of Machine Learning Research*, 11:2177–2208, 2010.

[24] Ankur Moitra. Algorithmic aspects of machine learning. `https://ocw.mit.edu/courses/18-409-algorithmic-aspects-of-machine-learning-spring-2015/e339520c4069ca5e785b29a3c604470e_MIT18_409S15_chapp6.pdf`, 2015.

[25] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. Scikit-learn: Machine learning in python. `https://scikit-learn.org/stable/modules/mixture.html#gmm`, 2011.

[26] Norman A. Phillips. Geostrophic motion. *Reviews of Geophysics*, 1:123–176, 1963.

[27] G. D. Portwood, S. M. de Bruyn Kops, J. R. Taylor, H. Salehipour, and C. P. Caulfield. Robust identification of dynamically distinct regions in stratified turbulence. *Journal of Fluid Mechanics*, 807:R2, 2016.

[28] Devranjan Samanta, Yves Dubief, Markus Holzner, Christof Schäfer, Alexander Morozov, Christian Wagner, and Björn Hof. Elasto-inertial turbulence. *Proceedings of the National Academy of Sciences*, 110:10557 – 10562, 2012.

[29] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, pages 81–85, 2009.

[30] S. Sid, V. E. Terrapon, and Y. Dubief. Two-dimensional dynamics of elasto-inertial turbulence and its role in polymer drag reduction. *Phys. Rev. Fluids*, 3:011301, Jan 2018.

[31] Matthias Sonnewald, Carl Wunsch, and Patrick Heimbach. Unsupervised learning reveals geography of global ocean dynamical regions. *Earth and Space Science*, 6:784–794, 2019.

[32] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[33] Wikipedia contributors. Principal component analysis. `https://en.wikipedia.org/wiki/Principal_component_analysis`, 2022.

[34] Jun-Ichi Yano and M. Bonazzola. Scale analysis for large-scale tropical atmospheric dynamics. *Journal of Atmospheric Sciences*, 66:159–172, 2009.

[35] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.