# MPhil DIS Report 24

*CRSiD: tmb76*

*University of Cambridge*

June 24, 2024

# Contents

# Chapter 1

# Executive Summary

# Chapter 2

# Introduction

One of the key steps of scientific method is reproducibility. Results must be reproducible by others, ensuring that the same conclusions can be drawn multiple times. If a result cannot be reproduced, 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) [4] paper and to test the robustness of their method.

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, due to multiple asymptotic local behaviours where the system is dominated by a subset of the terms. 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 [6]. This method, known as dominant balance or scale analysis, has been a powerful tool in physics.

And though extremely useful, dominant balance also requires expertise and is usually done by hand in time-consuming proofs. This report discusses and verifies a novel approach, developped by Callaham et al. (2021) [4], which explores using data from a physical system and machine learning methods to identify dominant balances algorithmically. First, this report will focus on the paper and the research surrounding it. Delving into what the rationale behind the method is, and how it performed on a series of case studies, as well as verifying it through reproducing the results with alternative code. This will be done primarily focusing on one of the case studies, but also for most of the others. Additionally, other algorithms than the method's chosen one are used to test the robustness of the method. Second, the method will be used on a new dataset, from simulations of elasto-inertial turbulence, a property of polymer laden flow.

# Chapter 3

# 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 for a given asymptotic case. Simplifying the equations also leads to easier computations by avoiding unnecessary complications of the model. Taking the example of meteorology, where modelling the entire atmosphere from the full Navier-Stokes equations of motion would have an immense computational cost. And a large part of improvements in numerical weather predictions can be attributed to scale analysis [3, 5, 21, 25].

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 a few decades ago. But with the wealth of computational power and data science techniques there is today, an attempt at automating dominant balance can be made. First is the Portwood et al. (2016) [22] paper which used a cumulative distribution function on local density gradients to separate each region of a stratified turbulent flow. But the method used is highly tailored for its case study, as the gradient of one of the terms is used, knowing it has dynamics discerning qualities. Further, the results are interpreted through expert analysis of the identified regions. Second is the work carried out in Lee & Zaki (2018) [16] where an algorithm to detect different dynamical regions is introduced. But again it is through the use of case-specific variables (vorticity), which restrict the use of this algorithm to certain flows. Finally, Sonnewald et al. (2019) [24] used a K-Means clustering algorithm to identify dynamically distinct regions in the ocean. And though they do introduce the concept of using the terms in the governing equations as features, the identification of active terms is done through comparison of the magnitudes of each term in the equation. In other words, identifciation of dominant terms is not done algorithmically but "man-

ually". 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) [23] contributed to a breakthrough using symbolic regression to find linear and non-linear differential equations. And this was improved in Brunton et al. (2016) [2]. As symbolic regression is expensive, the problem was approached with sparse regression, which for high-dimensional problems means identifying a sparse governing equation. The rationale behind this is that governing equations usually having only a subset of terms being important, as in dominant balance. Lejarza & Baldea (2022) further advanced this by using multiple basis functions and a non-linear moving horizon optimization 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 [9]. 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) [4] attempt to fill in the identification of dominant balance models. They propose a novel approach to take in simulated or measured data from a 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 essentially automatically learn the governing equations and asymptotic regimes of that equation for any given physical system. This is a very powerful result however, as Schmidt & Lipson (2009) 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. This is an important point which will be further discussed in this report.

# Chapter 4

# Methodology

The method proposed by Callaham et al. (2021) [4] can be summarized in three steps. First, representing the data in an equation space, with the terms as features. Second, using the Gaussian Mixture Model clustering algorithm to identify groups with a similar balance of terms. Finally, using Sparse Principal Component Analysis to identify the active terms in each cluster. This section will delve into the details of each of these steps, with a summary flow chart in Figure **??**.

## 4.1 Data & Equation Space Reprensentation

As previously mentionned, obtaining the data can be done through simulations or measurements, with almost no restrictions as to what equations can be studied. The data comes in the form of space-time fields of physical variables: $u(\vec{x}, t)$, where $u$ is the physical variable considered (e.g. $\vec{u}$, $p$, etc. for the Navier-Stokes equations). The variables must be sufficient so that one may derive all the terms of the equation from them. One important requirement is that the computed terms must all balance out to 0 for each point in time and space. This comes down to how one computes said terms. They must be as they are in the equation, since this results in a linear covariance structure, as each term is balanced by a linear combination of the other terms [4, Supplementary Information]. For example, if a term is a squared comnbination of variables, it must be computed as such, and not as the combination of variables.

The fundamental idea of the Callaham et al. (2021) method is to take these physical-space fields of terms and organize them into an equation space where each dimension 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:

$$\vec{f} = \begin{bmatrix} f_1(u(\vec{x},t),\ldots) \\ f_2(u(\vec{x},t),\ldots) \\ \vdots \\ f_k(u(\vec{x},t),\ldots) \end{bmatrix} \tag{4.1}$$

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 0, or that the residual is 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.

## 4.2 Gaussian Mixture Model Clustering

Geometrically, by clustering points in this equation space, one can identify groups of points that have a similar balance of terms. 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 [19]. 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 one wants the algorithm to find, and therefore how many Gaussian distributions the data is assumed to have been generated from [20]. Thus, for each case study, a number of clusters to identify needs to be chosen. 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.

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. The expectation step then evaluates the sum posterior probability of each point belonging to each cluster. And the maximisation step updates the parameters with weighted averages of those posterior probabilities [11] (see Alg. 1, in section 8.1). This converges to the maximum likelihood estimates of the parameters (see Fig. 4.1). Once fitted, cluster membership of new data points can be determined by taking the cluster with the highest probability at that point.

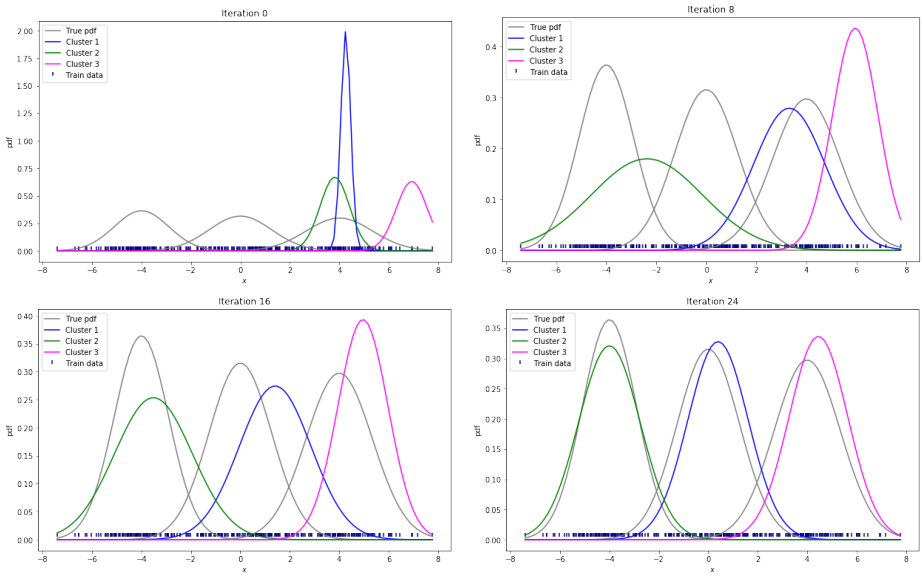



Figure 4.1: Example of a Gaussian Mixture Model fit to data. The data is generated from 3 Gaussian distributions, and the GMM algorithm fits 3 Gaussians to it. [8, 1D Example Notebook]

Another key output of the GMM is in its probabilistic nature which has the covariance matrix of each cluster/Gaussian as part of its output, thus describing the covariance between each term in the equation within that cluster. This gives insight into which terms could be active in each cluster, and also helps illustrate how the method can identify active terms geometrically. 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 the terms are non-negligible.

## 4.3 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 is done through Sparse Principal Component Analysis (SPCA). Principal Component Analysis (PCA) is a method that is usually 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 [17]. SPCA differs from PCA in that it adds a sparsity constraint to the principal components, resulting in most coefficients of the PCs being zero. This is done by adding a L1 penalty to the objective function of PCA for the number of non-zero coefficients, leading to some coefficients being shrunk down to 0 [26] (see Alg. 2, in section 8.1).

# Chapter 5

# Conducted research

To test and validate this method, Callaham et al. used a series of case studies [4]. While some were well-studied physical systems with well-known dominant balance regimes (e.g., turbulent boundary layer, geostrophic balance currents), others were more complex and less understood (e.g., optical pulse generation, rotating detonation engine). For these latter cases, the paper presents a first plausible identification of dominant balance regimes. In this chapter, the aim is to discuss the reproducibility of the results presented in the paper, as well as to test and discuss the limitations of the method.

## 5.1 Portability of the code

One great quality of Callaham et al. (2021) [4] is the sharing of their code in the form of runnable notebooks. With the overarching 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 useful reproduction of the results, where explicitly different results can be tested.

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 final notebook versions and their uploading to the repository. This led to some troubleshooting to get the data to match the data used in the paper. For the flow past a cylinder case, a file needed in the setup of the Nek5000 simulation software was missing [13]. For the bursting neuron case, the times for which the data was generated was wrongly set. Similarly, for the geostrophic current 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 re-

sults in the paper are 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 [7], and the time was set to the same as the one used in the paper.

Nevertheless, the code was otherwise easy to run, and the notebook format was very useful for following the logic of the code. From this notebook, the code was first written for the turbulent boundary layer case. Explicitly alternative code was then written to test the reproducibility of the results.

## 5.2 Reproducing of the results

Reiterating previously made points, one of the main goals of this project is to verify the results of Callaham et al. (2021) [4]. The objective is not only to check that the results can be reproduced but to do so using alternative code. The aim here is to have code written differently that, in practice, performs the same functions. This is 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.

### 5.2.1 Turbulent Boundary Layer

Because the underlying method is essentially the same for all case studies, the choice was made to put particular focus on just one of the cases, and less so on the others. The scenario chosen was the boundary layer in transition to turbulence.

Throughout the Callaham et al. (2021) [4] 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 the most efficient way. 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. For some code sections, however, impractical workarounds were required, particularly to replace the `numpy.unique` method.

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 [15]. This database provides the $x$ and $y$ coordinates as well as the following vari-

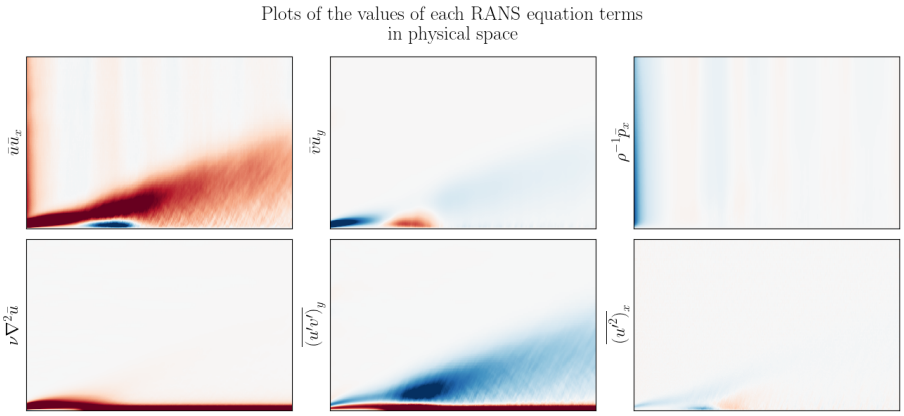



Figure 5.1: Plot of the 6 terms in the RANS equation (Eq 5.1), using the original Callaham et al. code

ables: $u$, the streamwise component of velocity; $v$, the wall-normal component of velocity; $p$, 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:

$$\bar{u}\bar{u}_x + \bar{v}\bar{u}_y = \rho^{-1}\bar{p}_x + \nu\nabla^2\bar{u} - (\overline{u'v'})_y - (\overline{u'^2})_x \tag{5.1}$$

To compute the derivative terms, the method employed by Callaham et al. uses `scipy.sparse`'s sparse matrices. These sparse matrices are constructed so that when computing the matrix product with the fields or variables, the second-order forward/-central/backward difference derivative is obtained, following the method described in 'Fundamentals of Numerical Computations' [12]. 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 computation 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. 5.1).

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 probability 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:

$$\log \mathcal{L}(\vec{X}|\vec{\theta}) = \sum_{i=1}^{N} \log \left( \sum_{k=1}^{K} w_k \mathcal{N}(\vec{x}_i|\vec{\mu}_k, \vec{\Sigma}_k) \right) \tag{5.2}$$

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

The alternative code was thus written 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.

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`, which sometimes has similar functions and sometimes requires workarounds.

### 5.2.2 Results

The first result to check is the terms obtained in Fig. 5.1. These were obtained with the original code, which made use of `scipy.sparse`. The alternative code, which used `numpy.gradient` instead, gave the same results (see Fig. 5.2). The reason Callaham et al. likely used `scipy.sparse` is because it is more generalizable to all variables; however, they also used `numpy.gradient` for other case studies. Using `numpy.gradient` also leads to considerable speedups.

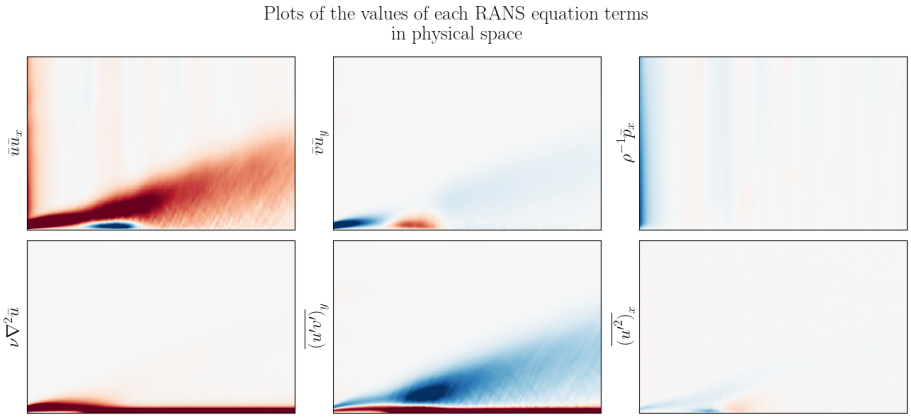
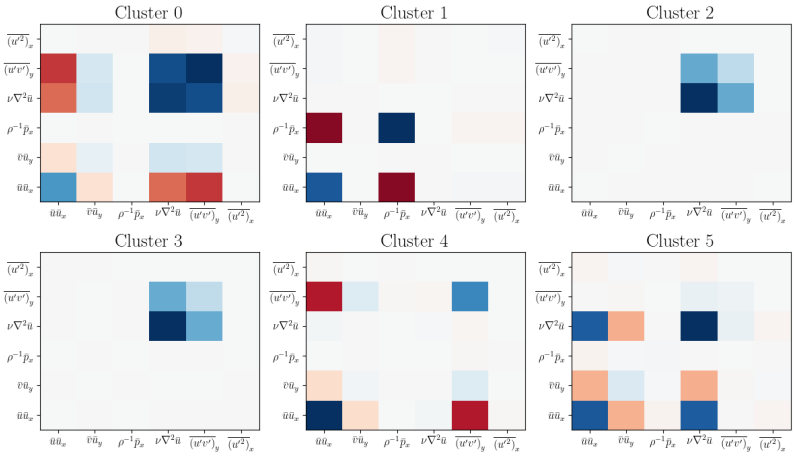
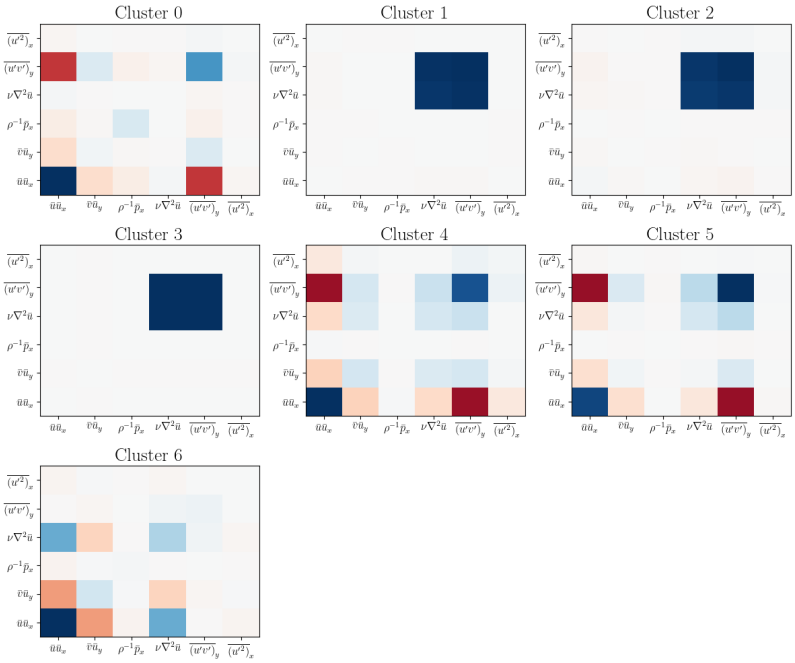



Figure 5.2: Plot of the 6 terms in the RANS equation (Eq 5.1), using an alternative code

The next step is to cluster the data in equation space. The resulting clusters' covariance matrices were obtained and are plotted in Figure 5.3. Plotting the current clustering in physical space gives the results in Figure 5.4. As can be seen, the results do differ, with the alternative code needing 7 clusters to identify the transitional layer (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 performed in the `sklearn` implementation [20].

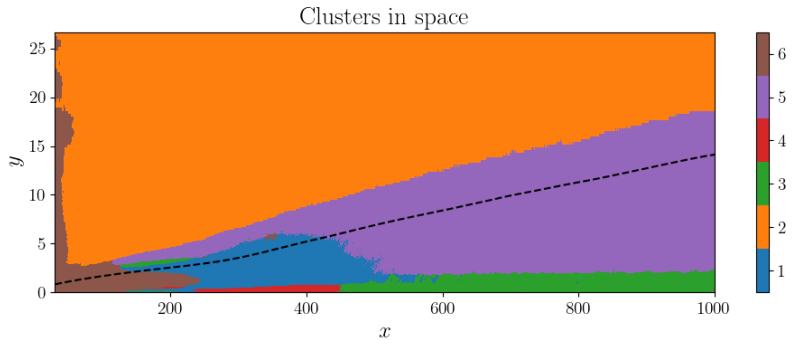



(a)

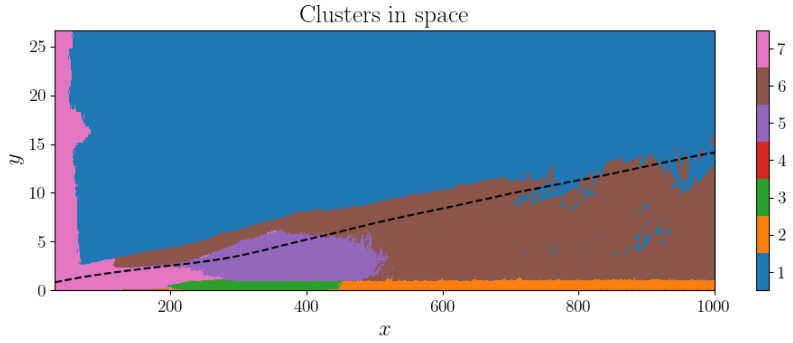



(b)

Figure 5.3: Covariance matrices of for each of the clusters found by the `sklearn` (a) and custom (b) GMM algorithm




(a)




(b)

Figure 5.4: Plot of the clusters found by the `sklearn` (a) and custom (b) GMM algorithm. The cluster numbers here match the ones in Figure 5.3, but not 0 indexed

Then comes applying SPCA to each cluster to identify which terms are active in it. In this case, the `sklearn` function is still used, though with alternative code to handle the results. Due to the different clustering obtained, the optimal alpha values had to be different for each method. 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 inertial sublayer unique and separate from the background free-stream (see Fig. 5.6). 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 plotted in Figure 5.5. Similarly to Figure **??**, some clusters are well identified by both methods. In both methods, the spatial arrangements of the clusters are the same, though their dynamics sometimes differ, or even switch (free flow vs. inertial sublayer in orange). Overall, similar key dynamics are identified in most of the clusters. For example, the viscous sublayer in blue and green has the viscous forces and streamwise Reynolds' stress terms as active. The inflow region crucially has the streamwise Reynolds' stress term as active. And in both cases, the transitional layer is identical.

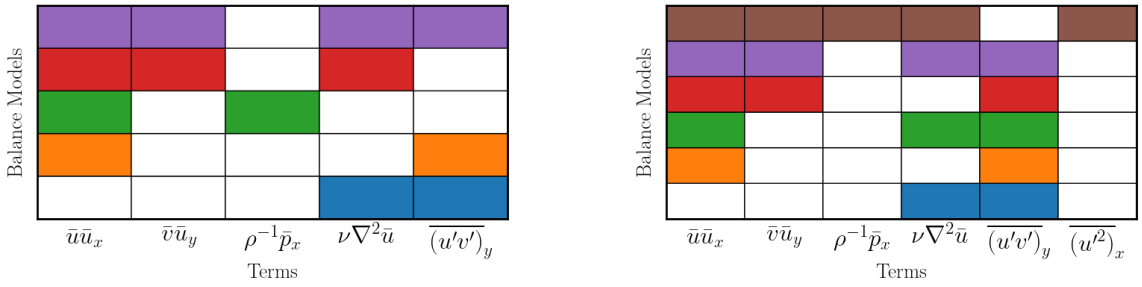
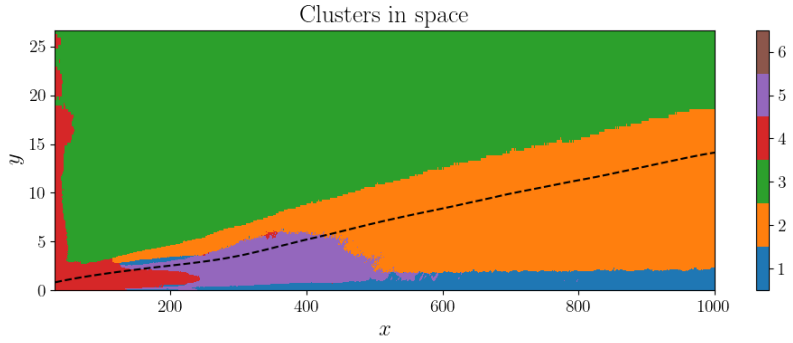
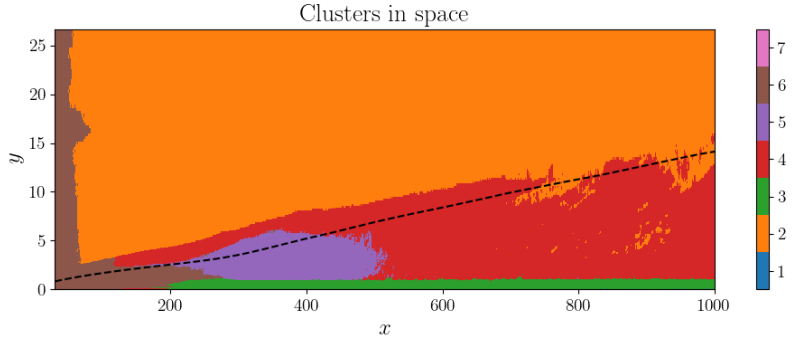



Figure 5.5: Plot of the unique balance models found after applying SPCA, when using the original Callaham et al. code (a) and an alternative code (b). Here, the cluster colors will match the ones in Figure 5.6

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 important dynamics in the boundary layer. It is also important to note that, though the overall method presented in Callaham et al. (2021) [4] brings large improvements in terms of generalizing the




(a)




(b)

Figure 5.6: 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 5.5

identification of dominant balance regimes for different cases, there are still two parameters to set: the number of clusters to find when using the GMM algorithm, and the alpha value for the SPCA algorithm. These parameters can have a large impact on the results obtained. This is a key point to keep in mind and will be further discussed in the Stability Assessment section.

## 5.3 Exploration of other algorithms

In the interest of further testing Callaham et al.'s method, and following the Supplementary Discussion in the Supplementary Information of the paper [4], other clustering algorithms could be used compared to Gaussian Mixture Models. This section explores a few options and discusses the results obtained.

### 5.3.1 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. As a result, spectral clustering is a convincing candidate for this, and it was also suggested in the Supplementary Discussion of the paper [4, Supplementary Information].

Using spectral clustering, the main drawback is the computational complexity $\mathcal{O}(n^3)$, where $n$ is the number of samples. Thus, it is not the most efficient algorithm for larger datasets, which most physical systems likely will be. Furthermore, it is not possible to sequentially add new points to the graph. Training it on a fraction of 0.01 of the full dataset, the results are shown in Figure 5.7. Because it is hard to predict cluster membership for all points, the solution was to plot the cluster memberships as a scatter plot. Overall, spectral clustering deals well with the different-shaped clusters, as it can be seen that the same main structure in the flow was captured. However, looking at the dynamics identified, it is once again the case that some of the key dynamics are identified, notably the inertial and viscous sublayers (orange and green). However, the inertial sublayer here extends vertically into the region identified by the original code as the laminar inflow region. Furthermore, a large cluster, which covers the previously identified transitional and laminar inflow region, is simply considered to have all terms active (except the wall-normal stress term).

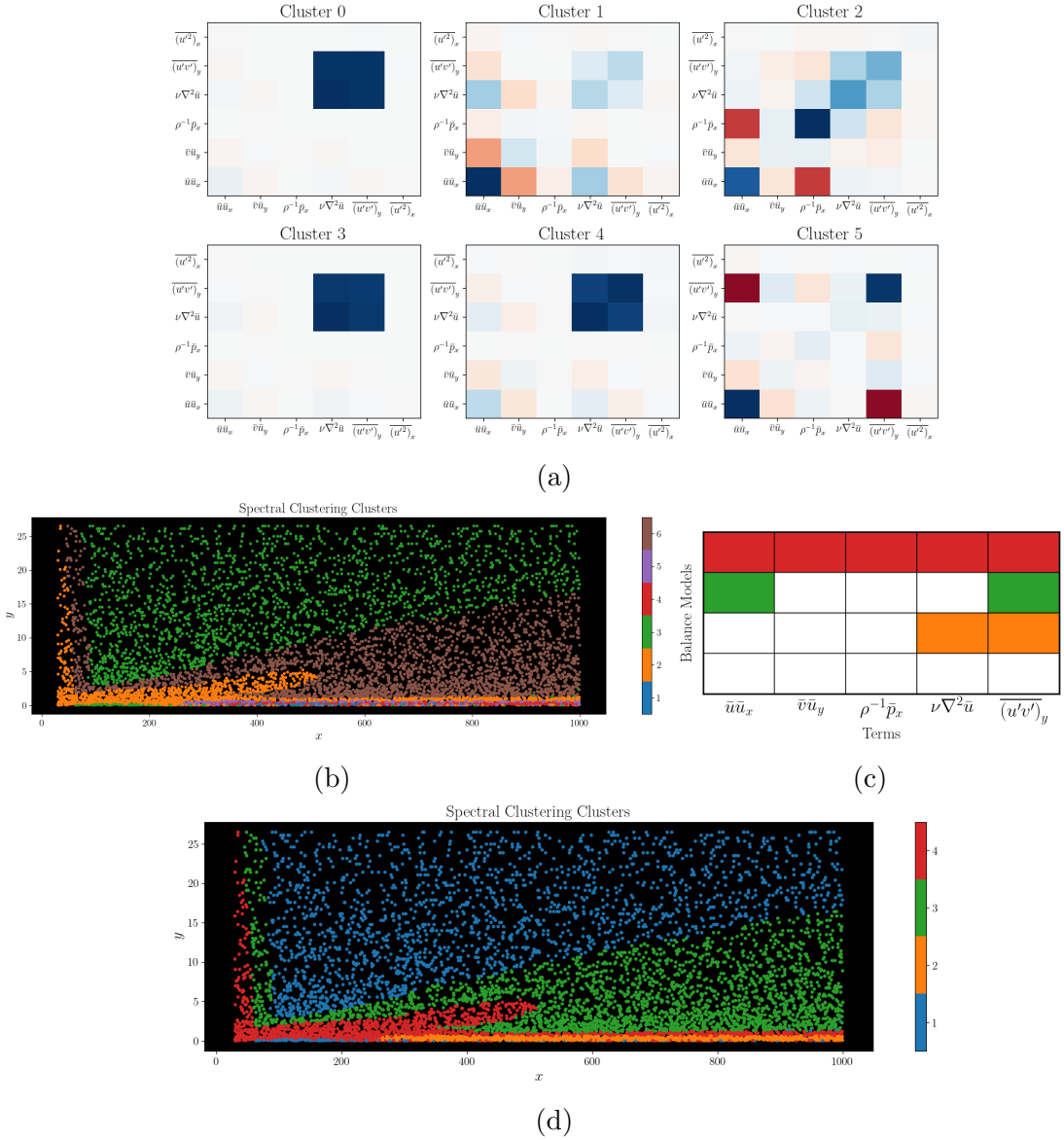



Figure 5.7: Results of the Spectral Clustering algorithm for the turbulent boundary layer case. **(a)** Covariance matrices of for each of the clusters found by the Spectral Clustering algorithm. **(b)** Plot of the clusters found by the Spectral Clustering algorithm. **(c)** Plot of the unique balance models found after applying SPCA. **(d)** Plot of the unique balance model clusters in physical space. Identified balance models include....

Therefore, though it provides a non-Euclidean based clustering algorithm, spectral clustering is not the most appropriate for this case study, or at least the aims of the method proposed by Callaham et al. (2021) [4]. It is computationally expensive, cannot accept new data points, and additionally can require one extra parameter to set (`n_neighbours`) when using nearest neighbors to build the graph.

### 5.3.2 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 any 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 may force the algorithm to identify the different clusters in the cloud of points near the origin.

The results of the K-Means algorithm are shown in Figure 5.8. Using the larger cluster number did help in identifying distinct regions; however, the dominant balance regimes identified in them do differ significantly from the ones identified by the original code. Again, clusters with similar dominant balances to those originally identified extend into other regions that had been identified to have very different dominant balance regimes (orange cluster in Figure 5.8).

### 5.3.3 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 mimicking the effect of spreading them apart. This should encourage the algorithm to identify multiple clusters in the cloud of points near the origin. The weights are set as follows:

$$w_i = 1 - (\tanh^2(\frac{1}{2}|\vec{OX}|)), \text{ where } \vec{OX} \text{ is the vector from the origin to the sample} x_i \tag{5.3}$$

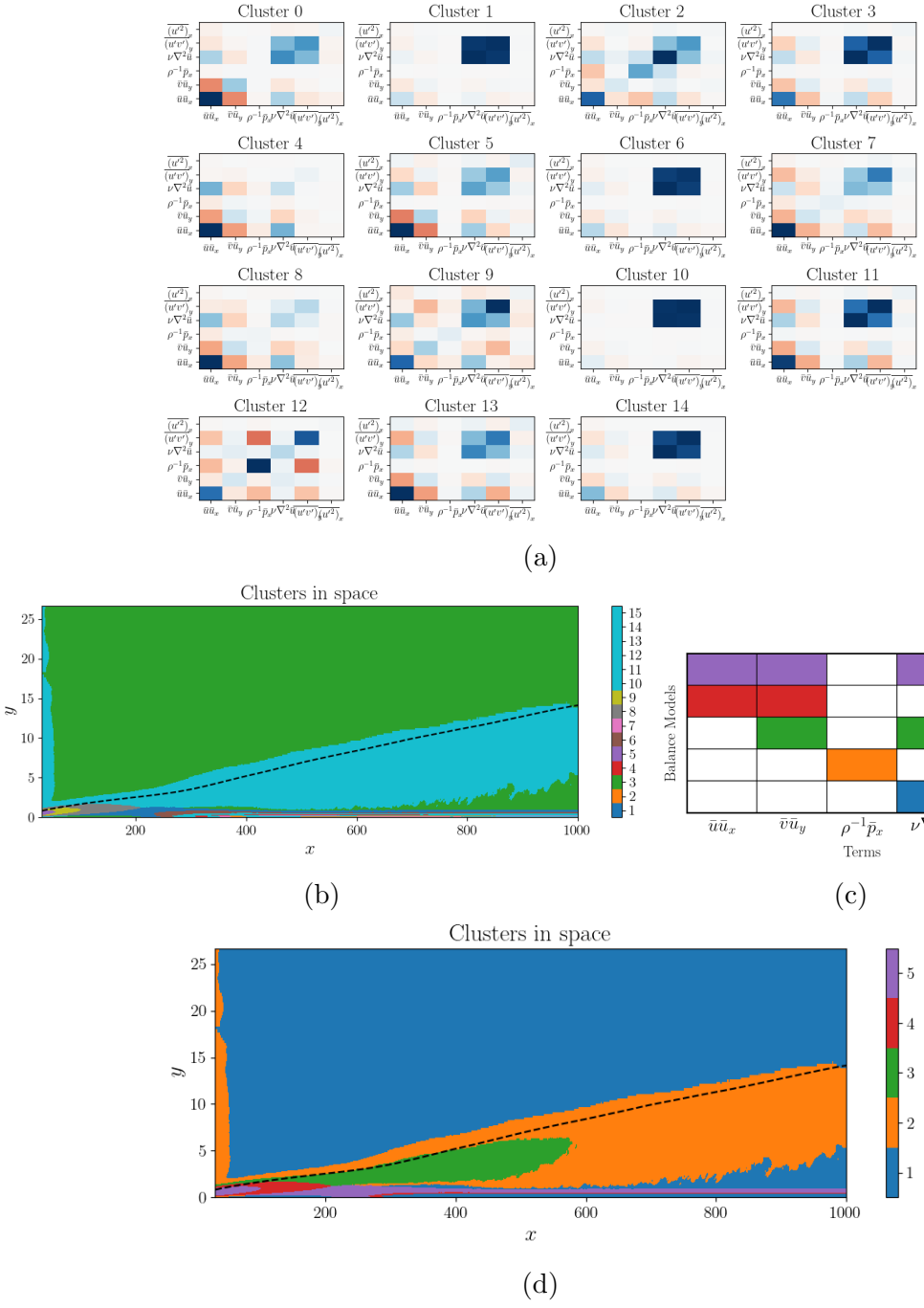
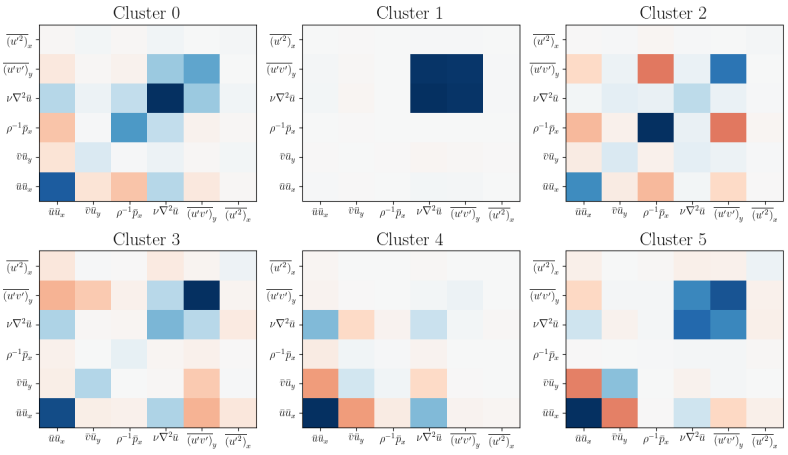
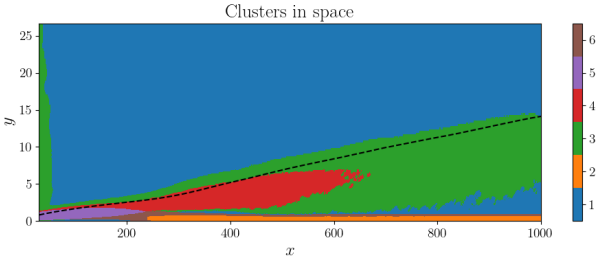


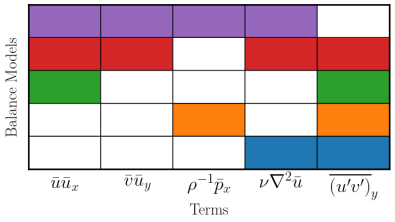

Figure 5.8: Results of the K-Means algorithm for the turbulent boundary layer case. **(a)** Covariance matrices of for each of the clusters found by the K-Means algorithm. **(b)** Plot of the clusters found by the K-Means algorithm. **(c)** Plot of the unique balance models found after applying SPCA. **(d)** Plot of the unique balance model clusters in physical space. Identified balance models include....



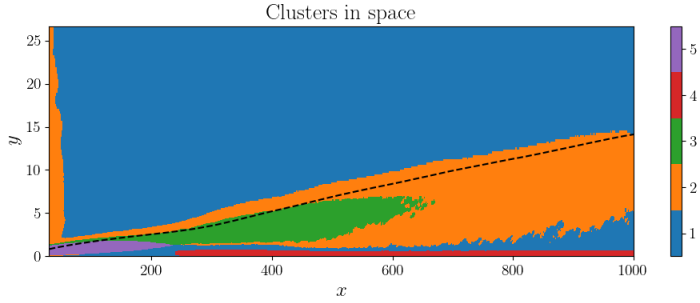

(a)




(b)




(c)




(d)

Figure 5.9: 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. **(c)** Plot of the unique balance models found after applying SPCA. **(d)** Plot of the unique balance model clusters in physical space. Identified balance models include....

The results of the Weighted K-Means algorithm are shown in Figure 5.9. 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 they are not assigned to the same locations as in the original code.

### 5.3.4 Summary

Overall, the Gaussian Mixture Model clustering algorithm offers really good flexibility thanks to its small number of hyperparameters, and for adding 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.

## 5.4 Stability Assessment

Though Callaham et al. (2021) [4] 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.

### 5.4.1 Under different number of clusters set

The first test that can be 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 be similar thanks to applying Sparse PCA and combining clusters with identical dominant balances. However, this is not guaranteed. If one were to combine two clusters with 2 terms active, and one shared active term, the resulting cluster would have the 3 terms active. Inversely, by increasing the number of initial clusters, this could result in dividing clusters into smaller, unidentically

balanced clusters.

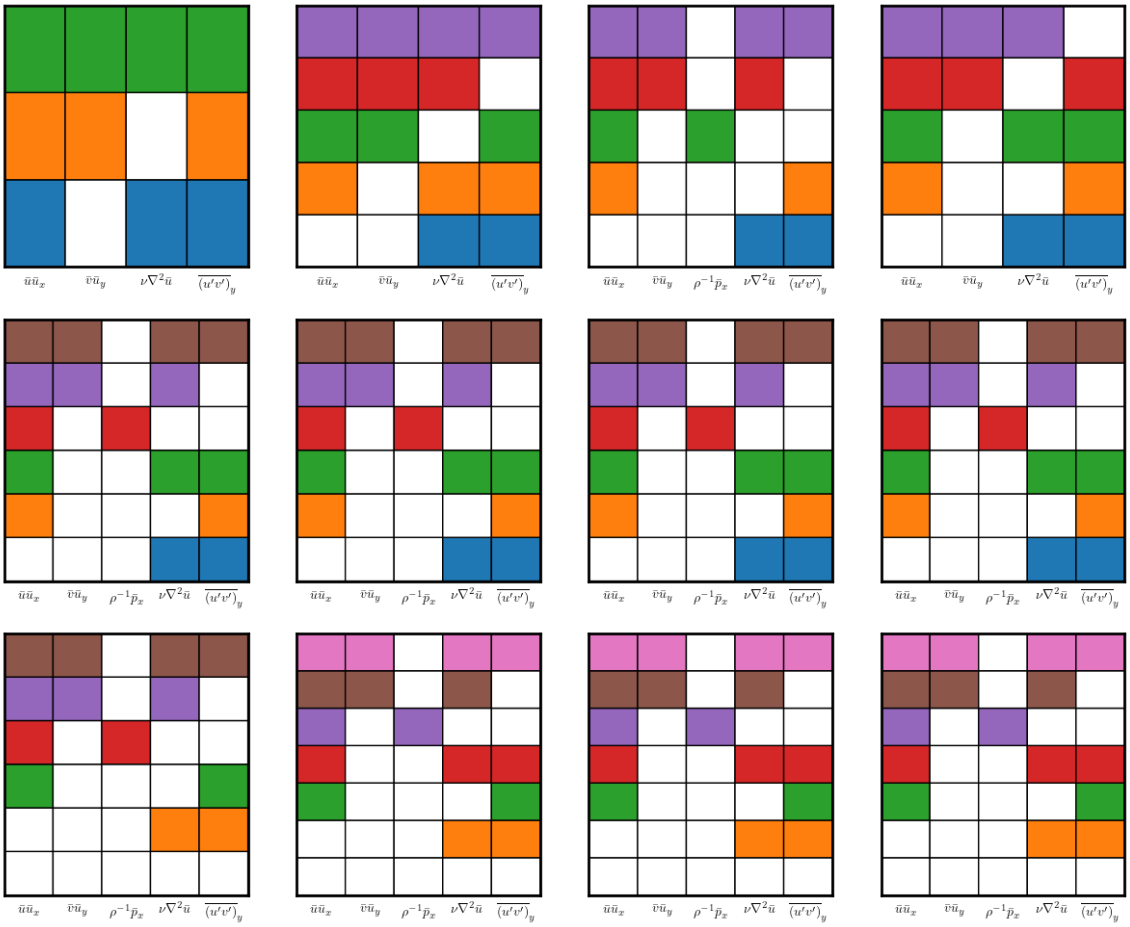



Figure 5.10: Plot of the obtained balance models for different initial cluster number

The algorithm was run for multiple cluster numbers, and the balance models were obtained for each case. These are all plotted in Figure 5.10. It can be seen that the results are mostly stable, with a slow increase in the number of final unique balance models found. However, checking the actual clustering in space and comparing to the balance models obtained...

### 5.4.2 Under different alpha values

### 5.4.3 Under different training set size

### 5.4.4 discussion

# Chapter 6

# Elasto-inertial turbulence

## 6.1 Background

## 6.2 Methodology

## 6.3 Results

## 6.4 Discussion

# Chapter 7

# Conclusion

# Chapter 8

# Appendix

## 8.1 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 $\alpha$, 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 [18], and structured sparse PCA [14] 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 $0 \leq k < n_{components}$

# Bibliography


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

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] 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.

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

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

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

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

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

[19] 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.

[20] 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.

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

[22] 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.

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

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

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

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