MPhil DIS Report 24

CRSiD: tmb76

University of Cambridge

June 21, 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, 18, 22].

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) [19] 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) [14] 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) [21] 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 "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) [20] 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 [9]. Second, deep learning (Graph Neural Network) and symbolic regression are combined to create a general framework to recover equations of physical systems [8]. 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. So that, second, the Gaussian Mixture Model clustering algorithm can be used to identify groups with a similar balance of terms. And finally, using Sparce Principal Component Analysis to identify the active terms in each cluster.

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 the terms of the equation from them. One important requirement is that the computed must all balance out to 0 for each point in time and space. And this comes down to computing the terms 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].

The main idea of the method proposed here is to reorganize these physical space fields into an equation space where each coordinate is one of the term 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:

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

Where f_i is the i^{th} term in the equation, itself a function of the physical variables. Again, one must make sure that for each sample, the terms all balance out to 0.

4.2 Gaussian Mixture Model Clustering

Geometrically, by clustering points in this equation space, one can identify groups of points which have a similar balance of terms. Here the algorithm chosen 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 [16]. It has the advantage of being able to identify clusters of 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 [17].

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 iteratively evaluates the posterior probability of each point belonging to each cluster, and updates the parameters with weighted averages [10]. This converges to the maximum likelihood estimates of the parameters. The algorithm is explained in greater detail in Algorithm 1 in the Appendix.

Another key output of the GMM is that its probabilistic nature allows for the identification of covariances between the terms in each cluster, as these are defined by a gaussian distribution and its covariance matrix.

4.3 Sparse Principal Component Analysis

With the points grouped in clusters of similar balance of terms, the next step is to identify which are active and which can be dropped to simply the equation. 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 kept. 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 [15]. 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, which leads to some coefficients being shrunk down to 0 [23]. Again, SPCA is explained in greater detail in Algorithm 2 in the Appendix.

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 whose dominant balance regimes were well known (Turbulent boundary layer, Geostrophic balance currents), others were more complex and less understood (Optical pulse generation, rotating detonation engine). For these latter cases, the paper therefore presents a first plausible identification of dominant balance regimes. In this chapter, the aim is to discuss the reproducibility of the results of the paper, as well as testing and discussing the drawbacks of the method.

5.1 Portability of the code

One great quality of the 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 todays 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 explicitely 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 lead to some troubleshooting to get the data to be the same as the one used in the paper. And for the flow past a cylinder case, a file needed in the setup of the Nek5000 simulation software was missing [12]. For the bursitng neuron case, the time for which the data was generated was wrongly set. Similarly for the gesotrophic 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 results 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 to follow the logic of the code. From this, the code was first written for the turbulent boundary layer case. And explicitely 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]. Not only check that the results can be reproduced but to do so using alternative code. The aim here is to have code differently written which in practice does the same thing. This is to make sure none of the main results from Callaham et al. are due to a "lucky" bug in their code.

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 3 main steps of the method, the large majority of the code is the handling of the data, switching from equation space to physical space, and getting the unique balance models. Most of it is written using `Numpy`, which is the most efficient way. In the alternative code however, `Pandas` was mostly used. In terms of performance, it is close to `Numpy`, especially for the size of the data. For some code sections, however, some impractical work arounds were required, particularly to replace the `numpy.unique` method.

The data for this case study is obtained from the John Hopkins database, which ran a direct numerical simulation of a boundary layer over a stationary plate [13]. In this database are given the x and y coordinates as well as the following variables: u, the streamwise component of velocity, v, the wall-normal component of velocity, p, the fluid pressure, and $(\overline{u'v'})$ & $(\overline{u'^2})$, the wall-normal and streamwise Reynolds'

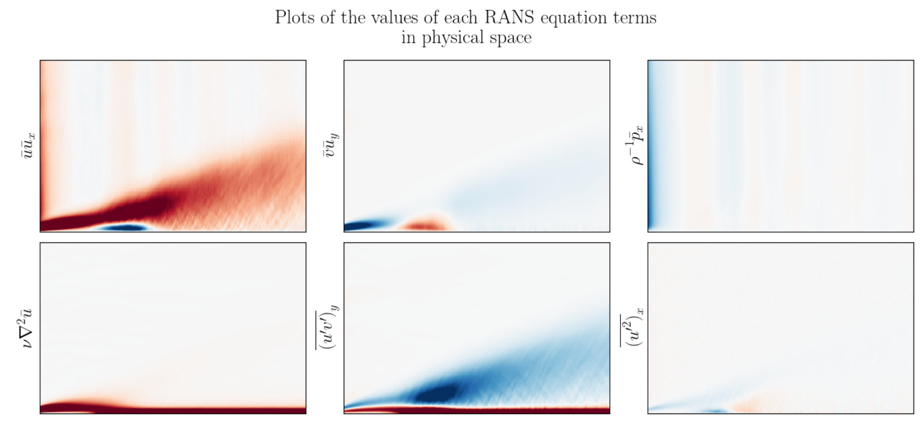


Figure 5.1: Plot of the 6 terms in the RANS equation (Eq 5.1), using the original Callaham et al. code

stress term. From these variables, the terms of the streamwise component of the Reynolds Averaged Navier-Stokes Equation need to be calculated:

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

In order to get the derivative terms, the method employed by Callaham et al. makes use of `scipy.sparse`'s sparse matrices. These sparse matrices are built so that when computing the matrix product with the fields or variables, the 2nd order forward/central/backward difference derivative following the method described in 'Fundamentals of Numerical Computations' [11]. Though it has the advantage of being portable for all variables, it is much slower than using the `numpy.gradient` function which does the same computation but more explicitely, when setting the `edge_order` argument to 2. With the derivatives computed, space fields of each term in the equation can be plotted (see Fig. 5.1)

The next step is to use a Gaussian Mixture Model to cluster the data in equation space. Quite judiciously, Callaham et al. chose to use the `sklearn` library's implementation of the GMM algorithm. It is likely a very efficient implementation of the algorithm, being written and optimised by experienced developpers. With the arguments chosen, the algorithm is initialised usimg the `'k-means++'` method. This method initialises the means of the gaussians using the centroids found by the K-Means clustering algorithm (see Algorithm 3), itself initialised based on the empirical probability distributions of the data [1]. The covariance matrices are then initialised 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, and the algorithm stops when the difference between the new and old log-likelihood is less than 10^{-3}.

The alternative code written follows the same initialisation, explicitely using `sklearn`'s K-Means algorithm, with `'k-means++'` initialisation. The Expectation-Maximisation algorithm is then implemented as described in Algorithm 1, using the total log-likelihood difference as the convergence criterion.

For Sparse principal component analysis, the `sklearn` library's implementation is used. The method used is based on an extension of space PCA, which makes use of structured regularization to constrain the sparsity patterns

With SPCA done, the active terms can be identified and from here, the original and alternative code only differ in whihc library they predominantly use. The original code uses mostly `Numpy`, whilst the alternative code uses `Pandas`, which sometimes had similar functions and sometimes needed a work around.

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 why Callaham et al. used `scipy.sparse` is likely because it is more generalisable to all variables, however, they also used `numpy.gradient` for other case studies. And using `numpy.gradient` leads to considerable speed ups.

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. And plotting the current clustering in physical space gives the results in Figure 5.4. As can be seen 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 criterion or some of the covariance matrices regularisation that is done in the `sklearn` implementation [17].

Then comes applying SPCA to the points in each cluster to identify which terms

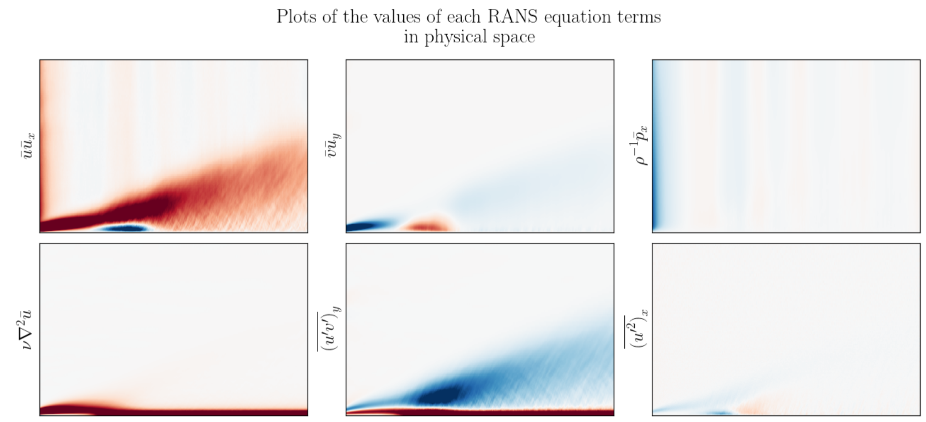


Figure 5.2: Plot of the 6 terms in the RANS equation (Eq 5.1), using an alternative code

are active. 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

One can see the active for the 2 codes in Fig. 5.6

5.3 Exploration of other algorithms

5.3.1 Spectral clustering

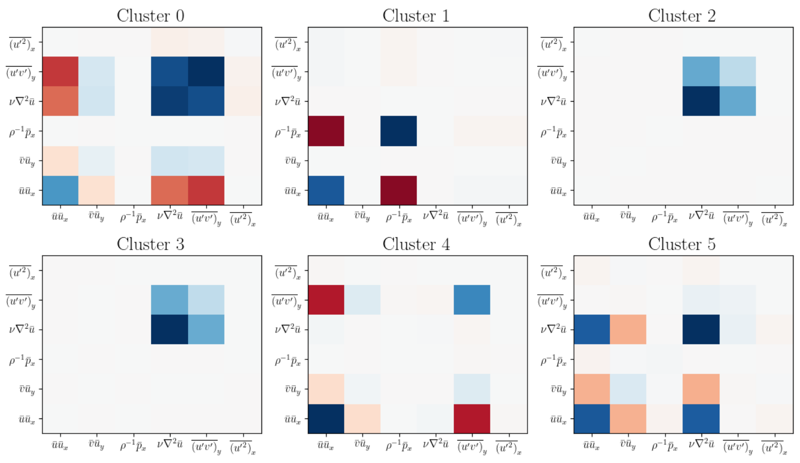
5.3.2 K-Means

5.3.3 Weighted K-Means

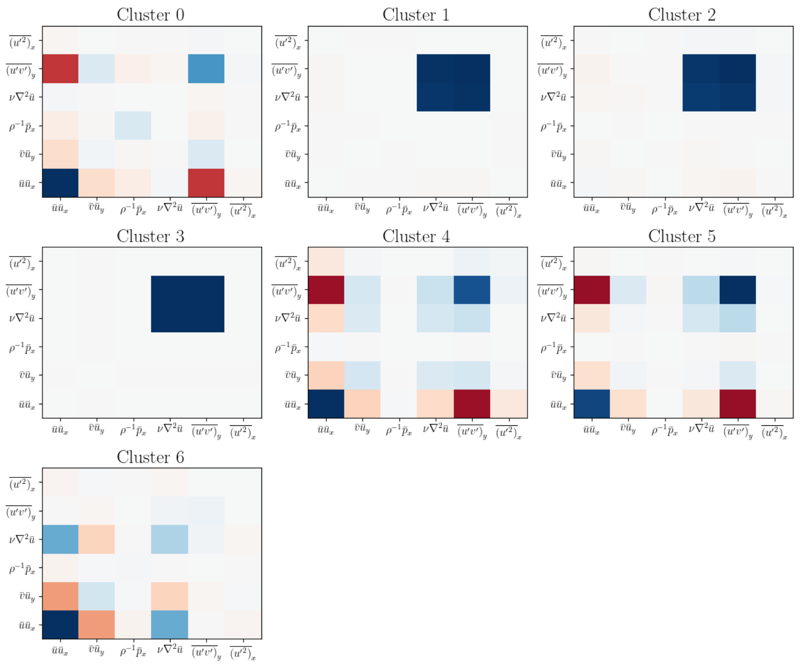
5.4 Stability Assessment

5.4.1 Under different number of clusters set

5.4.2 Under different training set size

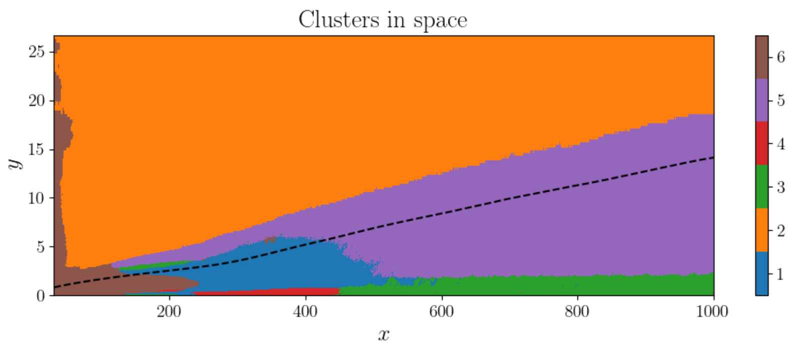


(a)

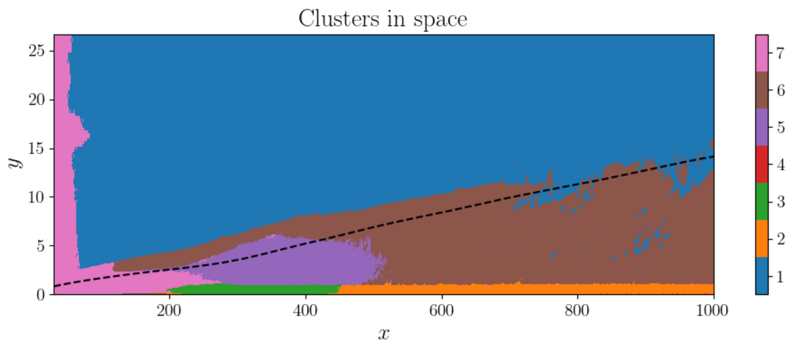


(b)

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

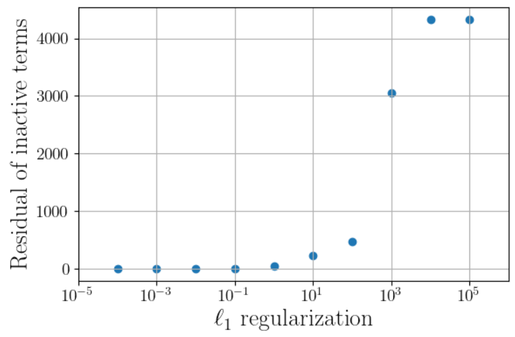


(a)

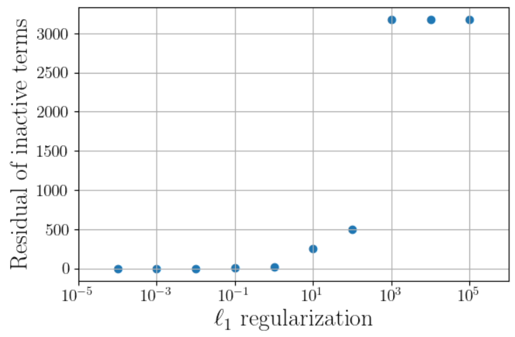


(b)

Figure 5.4: Plot of the clusters found by the `sklearn` (a) and custom (b) GMM algorithm

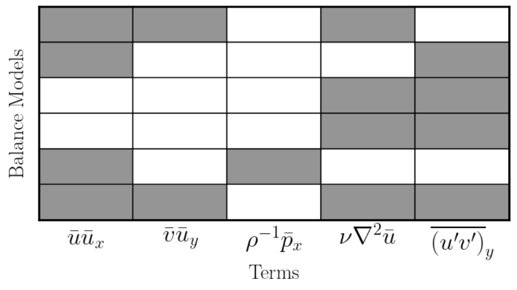


(a)

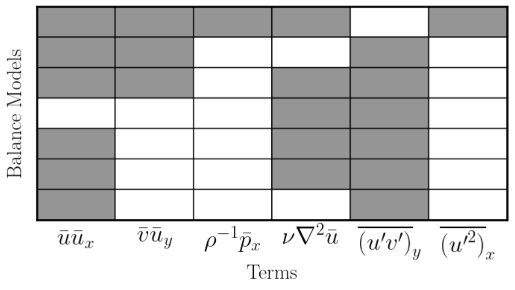


(b)

Figure 5.5: Plot of the residuals of the SPCA algorithm, when using the original Callaham et al. code (a) and an alternative code (b)



(a)



(b)

Figure 5.6: Plot of the terms identified as active by the SPCA algorithm, when using the original Callaham et al. code (a) and an alternative code (b)

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

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

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

[9] 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.

[10] 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.

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

[12] 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.

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

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

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

[16] 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.

[17] 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.

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

[19] 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.

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

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

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

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