SORBONNE UNIVERSITÉ

M1 PHYSICS INTERNSHIP REPORT

Unsupervised Classification of Particle Trajectories Using Artificial Intelligence

---

Author:
Dominik T. KÜHN

Supervisor:
Prof. Dr. Andrea CIARDI

Examiners:
Prof. Dr. Nicolas RODRIGUEZ
Prof. Dr. Florent BABOUX

Presented to the
Faculté des Sciences et Ingénierie

Prepared at
LERMA laboratory

27th June 2023

Contents

Chapter 1

Introduction

Computer simulations are one of the most important tools in plasma physics and are key to understanding how microscopic processes at ion and electron scales lead to macroscopic evolution and dynamics. The applications are numerous, ranging from astrophysics and space weather over atmospheric sciences to fusion research. Due to the large variety of interactions that the particles in a plasma are subjected to, i.e. heating and acceleration via instabilities, E x B-drift and discharge [1, 2], reconnecting and non-reconnecting current sheets [3], etc., it is difficult to connect the global behaviour of a plasma with the fundamental interactions and laws governing the system. Classifying particle trajectories in a plasma by finding common traits of the trajectories of large numbers of particles would allow for an insight into the interactions that cause the different observed trajectories and thus link the microscopic interactions with the system's global dynamics. Traditional approaches to particle classification often depend on manual examination or the establishment of predetermined selection criteria. These methods can be time-consuming and impractical when dealing with a large number of particles. Moreover, they tend to be subjective, potentially overlooking novel and unexpected physics. For instance, in the study of particle acceleration in plasmas, researchers typically analyse particle trajectories in order to identify groups of particles exhibiting common or unique behaviour. However, this analysis is typically restricted to a relatively small subset of trajectories, usually focusing on particles with high energy levels. Consequently, acceleration mechanisms that do not produce the highest energies may go unnoticed due to the vast amount of data that needs to be analysed. In this context, the automatic and unsupervised clustering of trajectories using machine learning techniques offers a promising opportunity for exploring new research directions and potentially revealing previously unknown mechanisms. An instance where automated classification becomes valuable is in distinguishing between shock surfing and shock drift acceleration, which can only be differentiated by examining the trajectories of particles. By employing automated classification methods, these distinctive trajectory patterns can be identified effectively [4, 5].

One such machine learning technique, called unsupervised learning, enables the discovery of hidden patterns, structures, and relationships within data, without the need for labelled training examples. Widely adopted unsupervised learning methods for classifications are k-means clustering, hierarchical clustering, and density-based clustering, which have all shown promising results in plasma physics and can be used for capturing different particle behaviours and identifying important plasma structures [6–8]. Such clustering algorithms group similar particles based on their shared characteristics, allowing for the identification of distinct particle populations within the plasma. Another approach in unsupervised classification is dimensionality reduction. Plasma simulations often generate high-dimensional datasets due to the large number of variables involved. Dimensionality reduction techniques,

such as principal component analysis (PCA) or autoencoders, are used to extract the most relevant features from the data, reducing its dimensionality while preserving the essential information. By reducing the complexity of the dataset, these methods facilitate the visualization and interpretation of particle distributions, aiding in the discovery of underlying plasma dynamics, while simultaneously radically reducing computation costs.

To summarise, unsupervised particle classification using machine learning techniques offers a promising avenue for enhancing plasma simulations. By automating the particle analysis process, researchers can gain a deeper understanding of plasma behaviour, uncover hidden patterns, and explore complex plasma phenomena more efficiently. The application of clustering algorithms and dimensionality reduction methods empowers researchers to identify distinct particle populations and extract meaningful information from high-dimensional datasets. An attempt at this has been made in the paper *Automatic Particle Trajectory Classification in Plasma Simulations* by S. Markidis et. al. [6], which is the foundation and main subject of this internship.

In the following chapters I will introduce the laboratory of which I was part for the durations of my internship (ch. 2), the context and important concepts of my work (ch. 3 and 4), the article my internship was based upon (ch. 5), results and insights that could be gained so far (ch. 6) and what still remains to be done during my time at LERMA (ch. 7).

Chapter 2

Presentation of the Lab

The Laboratory for Studies of Radiation and Matter in Astrophysics (with the French acronym LERMA) is a research oriented laboratory operated by CNRS, Observatoire de Paris, Sorbonne Université and Université de Cergy-Pontoise. It is organised into 4 main research groups, which are:

- Galaxies and Cosmology: Studying the development of the early universe, galaxy formation and dynamics, clusters of galaxies, dark matter, active galactic nuclei and star formation.
- Dynamics of the Interstellar Medium and Stellar Plasmas: Researching the characterisation of the ISM cycle, modeling ISM evolution, chemical diagnostics of ISM dynamics and the turbulant and radiative transport in stellar and circumstellar plasmas.
- Molecules in the Universe: Working on gas-surface interaction, molecular processes in gas phase, exotic isotopic spin ratios and molecular parameters for planetary and terrestrial atmospheres, and ISM.
- Instrumentation and Remote Sensing: Advancing THz components and subsystems, THz heterodyne instruments, characterisation of clear, cloudy and rainy atmospheres, as well as of Earth, planets and comets surfaces, and data processing, storage and diffusion.

During my internship, which was funded by the Sorbonne Centre for Artificial Intelligence (SCAI), I have worked in Extreme Laboratory and Plasma Astrophysics Group of the Dynamics of Interstellar Medium and Stellar Plasmas pole at LERMA, under the supervision of Prof. Dr. Andrea Ciardi and with occasional intellectual exchange with Dr. Weipeng Yao, a post-doctorial researcher in the same research group. With PIC simulations being frequently used by the research group, the researchers often need to deal with large amounts of particle data. The avenue of research I pursued during my internship only recently found the interest of Professor Ciardi, who employed me to work through and expand upon research that has already been done outside the confines of LERMA. Hence, at least thematically, I was the only person working on the proposed topic in the research group.

Chapter 3

PIC Simulations

One of the most widely used approaches to plasma simulations is the so-called Particle-in-Cell (PIC) method to simulate kinetic particles in systems where dissipation and other microscopic processes play an important role [9]. To provide a contextual foundation and emphasise the relevance of automatic particle classification, this chapter provides an introduction to PIC simulations. In addition, data from a PIC simulation is used to develop the machine learning algorithm described later (see ch. 5.1).

Systems of large numbers of particles that interact with each other through electromagnetic interactions are coupled to each other via their respective fields, that are made up of the contributions of the entire collection of particles. To compute the field at a certain position, the individually created fields of the particles are summed to give the overall field felt. As the effect of an electric charge in plasmas is shielded for distances larger than the Debye-length $\lambda_D = \sqrt{\epsilon_0 kT/ne^2}$, the interaction range can be assumed to be the Debye-length [10]. This means that all charged particles within that distance have an effect on a test-charge moved along a random trajectory in a volume. For strongly coupled systems, the particle density is large, leading to a small λ_D and a collection of only a few point-charges (for electrons and simplified ion models) within the screening radius λ_D: $N_D \propto n\lambda_D^3 \propto n^{-1/2}$. Due to the fast decay of the electric interaction with distance to the source, an individual particle's field will mostly be felt in its direct vicinity, which results in stark changes in the electric field when coming close to some. Thus the particles in a volume moving around due to thermal motion feel only a weak field from the contributions of all other particles, until they come close to another particle and feel a rapidly changing field, leading to a quick change to the trajectory of said particle. With all particles moving around like this, the electric field measurable in any place within the volume spanned by the Debye-length changes constantly and strongly.

In the opposite case of a large N_D and small n, there is always a large number of particles close to the point of measurement, leaving the electric field steady as opposed to constantly and drastically changing. Hence, a system with great N_D and low density n is considered weakly coupled.

PIC simulations consider a system of N particles as a subsystem of finite-size - or computational- particles, that each make up a collective of N_p physical, or "real", particles, that are close to each other in phase space. Treating them as computational particles with a certain shape in phase space, solving a set of equations known as the Vlasov-Maxwell equations, makes many computational aspects of the simulation much more feasible and allows for the computation of diffuse plasmas, in other words making simulations of weakly coupled systems attainable. The general idea of the finite-size particles used for PIC simulations is that of "clouds" of particles, that within the cloud don't interact with each other and keep the cloud shape intact,

while the clouds still interact with each other as collections of particles. Two of such finite-size particles at a distance thus act like point-like particles influencing each other. In proximity however, as soon as the computational particle volumes overlap, the overlapping physical particles within the finite-size particles don't contribute to the overall interaction anymore, making the interaction between the finite-size particles weaker [11]. Two finite-size particles at the same position therefore feel no potential from the other, while two point-like particles would experience an infinitely strong field, in the limit of occupying the same position. In other words, the introduction of finite-size particles reduces the total potential energy in a volume, while keeping the kinetic energy the same. Looking at the definition of the plasma coupling constant:

$$\Lambda = \frac{E_{th}}{E_{pot}} \tag{3.1}$$

this means that with a small number of particles to compute, the simulation can yield a high Λ, corresponding to weakly coupled systems.

As the interactions at a distance larger than the size of the computational particles are the same as with physical particles, the use of finite-size particles doesn't change the collective behaviour of the system and is therefore well suited for the simulation of weakly coupled plasmas. The cumulative electric field of all particles felt by a test-particle moving along a trajectory through the dense system is steady because the finite-size particles that are close enough to cause a strong change in the electric field only interact weakly, simulating well the physical reality where there are always enough particles around a test-particle to keep the field homogeneous. At the same time, the particles far apart only contribute weakly to the field due to their distance. Trajectories are similarly smooth because in close proximity to the computational particles, their potential diminishes.

These systems are governed by sets of equations that have to be solved numerically. In most cases of interest for the described simulation method, the particles in the plasmas are non-relativistic, which means the distribution function of the particles in phase-space has to solve the Vlasov–Maxwell set of equations [9]:

$$\left(\partial_t + \mathbf{v}\cdot\nabla_{\mathbf{x}} + \frac{\mathbf{F}_L}{m_s}\cdot\nabla_{\mathbf{v}}\right) f_s = 0 \tag{3.2}$$

where $f_s(\mathbf{x}, \mathbf{v}, t)$ describes the distribution function for a species of particles s in phase space, m_s the mass of the particles of type s, $\mathbf{x}$ and $\mathbf{v}$ the position and velocity of a phase-space element, c the speed of light in a vacuum and

$$\mathbf{F}_L = q_s\left(\mathbf{E} + \mathbf{v}\times\mathbf{B}\right) \tag{3.3}$$

the Lorentz force acting on a particle with charge q_s. The distribution function f_s can be considered the probability of finding a particle in a region spanned by d$\mathbf{x}$ and d$\mathbf{v}$ around the point $\mathbf{x}$, $\mathbf{v}$. In case of relativistic particles, some equations must be modified, the overall approach remains valid however. The fields are computed from the Maxwell equations:

$$\begin{aligned}
\nabla\cdot\mathbf{B} &= 0\\
\nabla\cdot\mathbf{E} &= \rho/\epsilon_0\\
\nabla\times\mathbf{B} &= \mu_0\mathbf{J} + \mu_0\epsilon_0\partial_t\mathbf{E}\\
\nabla\times\mathbf{E} &= -\partial_t\mathbf{B}
\end{aligned} \tag{3.4}$$

where ϵ_0 and μ_0 represent the permittivity and permeability in a vacuum and ρ and $\mathbf{J}$ the charge and current densities which are calculated using the distribution function:

$$\rho(\mathbf{x}, t) = \sum_s q_s \int f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \tag{3.5}$$

$$\mathbf{j}(\mathbf{x}, t) = \sum_s q_s \int \mathbf{v} f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \tag{3.6}$$

integrating over velocity space.

The distribution function f_s of a species of particles s is really a superposition of the distribution functions f_p of computational particles, that each describe a larger number of physical particles that are close in phase space:

$$f_s(\mathbf{x}, \mathbf{v}, t) = \sum_p f_p(\mathbf{x}, \mathbf{v}, t) \tag{3.7}$$

The idea of a superposition of distribution functions of computational particles forms the basis of the PIC method. Each computational particle's distribution function is dependant on a set of parameters (usually two per spatial dimension representing the position and velocity of the computational particle), that determine the solution of the Vlasov equation.

To better understand the need for automatic classification of particles, a simple calculation of the size of the data produced by PIC simulations shows that for each time step and particle in a 3 dimensional simulation, at least 6 values have to be saved (3 for space coordinates and 3 for the velocities), usually more when including the field information at the particles' positions.

To illustrate the magnitude of the challenge, let's consider an example of a kinetic simulation of asymmetric magnetic reconnection with cold ions[12]. The simulation was conducted in two spatial and three velocity dimensions using a grid size of 6400×5120 cells, with an average of 100 particles per cell. The simulation spanned 190,476 time steps and required approximately 14 million CPU hours to complete. The resulting dataset amounted to a massive 125 terabytes.

Manual analysis of such a vast amount of data would be practically impossible. Even approaches that involve selecting data exhibiting specific physical properties beforehand may not be effective since determining the properties to look for would require prior knowledge. However, if the simulated plasma could be automatically classified into distinct classes based on particle behaviour, it would greatly simplify the task of discovering interesting phenomena and comprehending the global dynamics resulting from microscopic interactions within the plasma.

Chapter 4

A Basic Introduction to Machine Learning

In order to find and isolate interesting particle dynamics in the large datasets created by PIC-simulations, some methods other than manual analysis have to be devised. Reducing a big set of data to a small number of interesting processes to be further studied by a researcher is a task that becomes more important the larger the simulations become. A suitable candidate for automatically selecting or classifying parts of data is "Machine Learning" (ML). There are many methods and tools that are considered machine learning. In this chapter, only concepts important to automatic classification of data shall be introduced. This typically involves the following key components:

- The Samples: The data that is to be classified should consist of a number of samples, each including the same general set of information. For example a sample might be the trajectory of a particle, which includes the x and y positions of the particle at specified time steps. Various trajectories represent various samples to work with.
- The Features: Each sample has to have the same number of features, or relevant characteristics. Like in the example above, a feature of a sample might be the x-position at $t = 0$ or the average energy of a particle. These features provide the algorithm with meaningful input for learning or making predictions or decisions. The selection and engineering of features are essential to ensure the algorithm can capture relevant patterns and relationships in the data.
- The Algorithm: This includes the pre-processing of any data, as well as the actual analysis of the data's features to make decisions on them and produce an output.
- The Evaluation: The evaluation helps determine the effectiveness of the model or algorithm and whether it is ready for deployment. It can i.e. be an evaluation of the precision of the model, the accuracy or the reproducibility of the results.

For the implementation of the machine learning algorithm, an understanding of the pre-processing step PCA, the unsupervised clustering algorithm and the evaluation of the clustering are paramount. A basic introduction to all of these concepts is given below:

4.1 PCA

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in machine learning and data analysis and often part of the preprocessing of data. It aims to transform a high-dimensional dataset into a lower-dimensional representation while retaining the most important information and minimising the loss of variance. A principal component is a linear combination of the original features in a dataset that captures the most significant variation or patterns in the data. In other words, it represents a new axis or direction in the feature space along which the data exhibits the maximum variance (see fig. **??**). The number of principal components is equal to the number of features in the original dataset. The PCA entails the finding of these principle components and the subsequent projection of the data onto them, in order to describe the largest possible variance with the smallest number of features. Principal components are orthogonal to each other, meaning they are linearly independent and uncorrelated. This orthogonality property ensures that each principal component captures a distinct and unique pattern in the data. Furthermore, the principal components are sorted based on the amount of variance they explain, allowing for the identification of the most important patterns or dimensions in the dataset. Reducing the dimensionality of the data by keeping and analysing only the first principle components of the data therefore only results in minimal information loss.

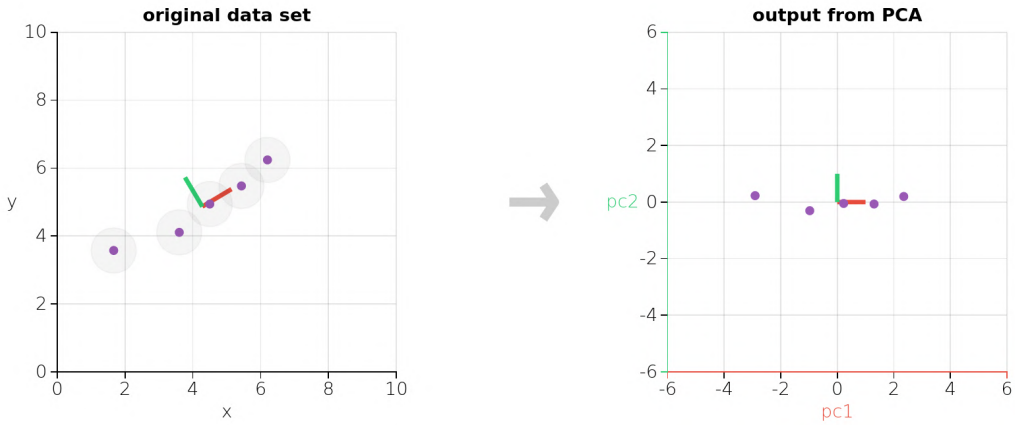


Figure 4.1: Visual representation of a PCA, with data projection on the principal components, the axes along which the data has the maximimum variance. Figure source: `https://setosa.io/ev/principal-component-analysis/`

Before applying PCA, it is necessary to ensure standardisation of the data. This step involves subtracting the mean from each feature and scaling the data to have unit variance. Standardisation removes any biases caused by differences in scale or units across features. An example to illustrate the necessity of this in most cases might be a feature of the samples that corresponds to a length given in kilometres, whose variance is of the order of the variances of other features. However, if that feature gets translated into centimeters, it still carries the same amount of information, while the variance will now dominate all other variances and the first principal component would be very similar to said feature. After this preprocessing step, the covariance matrix of the dataset has to be determined. While variance measures the

spread or dispersion of a random variable, covariance measures the degree of linear relationship between two random variables and is therefore vital in the reduction of redundant information. The eigenvectors of the covariance matrix represent the principal components of the data, and the corresponding eigenvalues indicate the amount of variance explained by each component. The eigenvectors and eigenvalues can be found through numerical means like singular value decomposition (SVD) or eigendecomposition. Typically, the principal components are ordered based on their corresponding eigenvalues, with the highest eigenvalue indicating the most important component. To reduce dimensionality, one can select the top-k principal components that capture a significant amount of the data's total variance and project the original data onto the k chosen principal components.

Mathematically, this procedure can be represented as follows: Let X be the standardised dataset of dimension $n \times m$, where n is the number of samples and m is the number of features. The covariance matrix C of dimension $m \times m$ is computed as

$$C = (X^T \cdot X)/(n-1)$$

where X^T denotes the transpose of the matrix X.
By performing i.e. eigendecomposition on C, we obtain the eigenvalues λ_i and the m-dimensional eigenvectors V_i. The eigenvectors, sorted in descending order of their eigenvalues, form the matrix

$$V = [V_1|V_2|...|V_m]$$

To create the reduced-dimension dataset, we select the top-k eigenvectors corresponding to the largest eigenvalues. By multiplying the original standardised dataset X by the selected eigenvectors, we obtain the $n \times k$ dataset

$$Y = X \cdot V'$$

where V' represents the matrix containing only the top-k eigenvectors.

4.2 Unsupervised clustering

Unlike supervised learning, where the algorithm is provided with labelled data to learn from, unsupervised clustering discovers patterns and relationships solely from unlabelled input data. It aims to partition the data into distinct clusters based on similarity or proximity between data points in the feature space of the samples. There are several clustering algorithms that might be used, including but not limited to k-means clustering, mean-shift clustering, DBSCAN, Gaussian Mixtures and hierarchical clustering, each with its strengths and weaknesses for specific types of data and structures within them. After standardisation and dimensionality reduction through PCA, the chosen clustering algorithm is applied to a dataset and data points are assigned to different clusters based on shared or similar features or proximity to each other. The defining metric for the algorithm to determine what "proximity" or "similarity" in a dataset means can vary and the choice can heavily affect the clustering outcome. Some examples for similarity and distance metrics are Euclidean distance, cosine similarity and city-block distance [6].

The k-means clustering technique is one of the most widely used and straightforward clustering algorithms, and used in countless fields from image compression to plasma physics. One of its fundamental properties and origin of its name is the need to specify the number of clusters k, that ought to be found in the provided dataset. With the number of clusters, their initial centroid positions, meaning their centre in feature space, need to be set. These centroids can be randomly selected or initialised using specific strategies. After initialisation, the assignment of the data points to the k clusters follows. In the assignment step, each data point is assigned to the cluster whose centroid is closest to it in terms of Euclidean distance, or another distance metric chosen. The distance between a data point and a cluster centroid represents the similarity between the point and the centroid. The goal of the clustering algorithm is to minimise the distance between data points and their assigned cluster centroids. After assigning each data point to a cluster, the centroids of the clusters are updated to the means of all data points assigned to the clusters. This step ensures that the centroids always make up the actual centre of the data points in their respective clusters. The assigning and updating steps are repeated iteratively until convergence or a pre-defined maximum number of iterations is reached. Convergence occurs when the assignment of data points to clusters no longer changes, which means that the algorithm has found stable clusters, and the final centroids represent the cluster centres.

Once the k-means algorithm has converged, the resulting clusters can be evaluated based on various metrics, such as the within-cluster sum of squares (WCSS) or silhouette coefficient, which will be discussed in the following section. If the evaluation suggests that the clustering results are unsatisfactory, the algorithm can be refined by adjusting the initialisation strategy, the number of clusters, or applying other modifications. For an a priori unknown number of clusters in the data, a common strategy to identify the clusters with k-means is to start with a small number of clusters and continuously increasing k while recording the WCSS scores of each clustering in a so-called elbow-plot. The larger k, the smaller the WCSS has to be. However, the rate of change of this measure changes when the optimal number of clusters is found, as the centroids are placed to minimise the distances between the data points and as soon as all clusters to place centroids in are exhausted, the largest distances in-between the clusters will not be part of centroid-data-point distances anymore. At this point in which the WCSS's rate of decline slows, also called the elbow, lies the ideal number of clusters to be found. It's important to note that k-means clustering assumes that the clusters are spherical, and the algorithm aims to minimise the sum of squared distances within each cluster. It is sensitive to the initial centroid positions and may converge to suboptimal solutions. To mitigate these issues, it is common to run the k-means algorithm multiple times with different initialisations and choose the clustering result with the lowest WCSS or other evaluation metrics. Additionally, another important draw back of the k-means algorithm is the fact, that cluster assignments are definitive and no certainty for the accuracy of the assignment is given. A data point that is exactly half-way between centroids is therefore assigned to one of the two clusters and it is not clear in the assignment, that the point is just as likely to be in the other cluster. Still, the algorithm's simplicity and efficiency make it a popular choice for clustering tasks, especially when the number of clusters is known or can be estimated.

4.3 Cluster evaluation

Cluster evaluation is a process of assessing how well unsupervised clustering algorithms have captured the underlying structure of the data. It involves quantitatively measuring the performance of clustering algorithms by measuring the compactness of clusters and the separation between different clusters.

Some commonly used methods for cluster evaluation are:

- Within-Cluster Sum of Squares (WCSS): WCSS measures the compactness of clusters. It calculates the sum of squared Euclidean distances between each data point and its cluster centroid. Lower values of WCSS indicate better clustering, as it suggests that data points within each cluster are closer to their centroid.
- Silhouette Coefficient: The silhouette coefficient measures both the compactness of clusters and the separation between different clusters. For each data point, it calculates the average distance to other data points within its cluster a and the average distance to the data points in the nearest neighbouring cluster b. The silhouette coefficient is then computed as $(b - a)/max(a, b)$, ranging from -1 to 1. A value close to 1 indicates well-separated clusters, while a value close to -1 suggests poor clustering for data points.
- Rand Index: The Rand index is a measure of similarity between two data clusterings. It compares the agreement between pairs of data points in terms of whether they are assigned to the same or different clusters in the evaluated clustering and a reference clustering (such as ground truth labels). The Rand index ranges from 0 to 1, where 0 indicates no agreement and 1 indicates identical clusterings.

No single evaluation method is universally applicable or definitive. The choice of evaluation method depends on the nature of the data, the specific clustering task, and the desired evaluation criteria. Multiple evaluation methods can be used in combination to gain a comprehensive understanding of the clustering performance.

4.4 Neural Networks

Naturally, there are many more aspects to machine learning. One notable example is the field of Neural Networks, which copy basic biological schemes found in brains with the name-giving neurons, and so called Deep Learning. A functional disadvantage of this technology in the context of automatic classification of particles in large datasets is the need to train the network on pre-existing and pre-labelled data, essentially supervised learning. Only with "correct" classifications of particles and their trajectories can a Neural Network learn to classify data correctly. What constitutes a correct classification in these situations is oftentimes ambiguous at best (see ch. 5.3), making the creation of suitable training data a difficult problem. Once trained, such a Neural Network holds great potential for classification tasks however, and might even pose a more powerful option than the unsupervised methods, yet it remains to be seen, whether the need to learn in a supervised manner reduces its ability to pick up on new behaviour that it was not trained.

Chapter 5

A First Approach to Classification

The article *Automatic Particle Trajectory Classification in Plasma Simulations* by S. Markidis et. al. proposes a workflow to analyse and classify a given dataset of particle trajectories in an unsupervised manner [6]. The primary goal of my internship was to understand the proposed workflow, critically analyse it and propose improvements. As I discovered in the attempt to reproduce the results of the article, some procedures discussed and claims made are not accurate or do not correspond to the actual code used by the authors. In this chapter I will discuss the algorithm that I used during my internship (see fig. 5.1), and highlight the differences with the proposed workflow in the original paper (see fig. A.1) where appropriate.

The general workflow as implemented can be summarised with this chart:

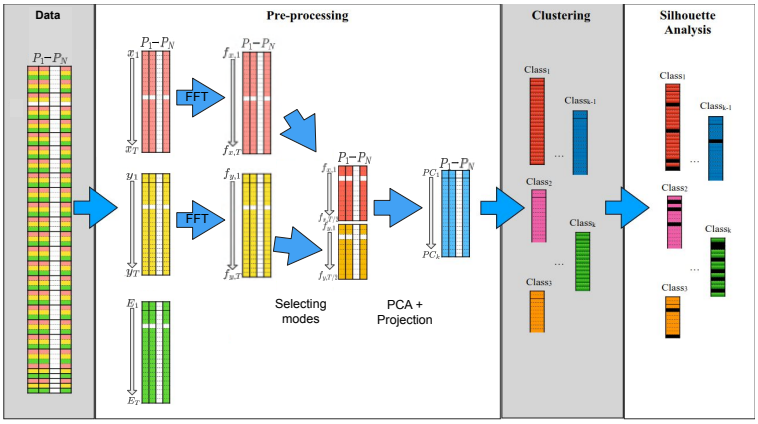


Figure 5.1: The workflow for unsupervised classification of particle trajectories in plasma simulations, as implemented during my internship, and not as presented in the original article [6].

First, from the data including x, y, and energy information on each particle for each time step the x- and y-trajectories are collected. Secondly, the trajectories are transformed using Fast-Fourier-Transform and the positive frequency modes are selected to then perform the PCA on. Finally, the projected and reduced dataset is clustered into classes and the silhouette scores of the particles in each class are calculated to make further assessments about the clustering. All individual steps, from the data production, over the pre-processing to the clustering are explained in detail in the subsequent sections.

5.1 The Particle Trajectory Dataset

The data I worked on was the original data used in the article[1]. It was produced using a PIC simulation. To be able to compare the results of the classification to pre-existing literature, the simulation is for a well-known and studied problem, the GEM challenge. This particular set-up involves a simplified geometry of magnetic reconnection, often referred to as 2D3V, where the positions are projected onto a 2D plane, while the velocities stay three-dimensional. An example for the occurence of magnetic reconnection in nature is around Earth's Magnetosphere [12], or in the Sun's magnetic field, releasing massive amounts of kinetic energy in form of solar flares [13]. The data used in the paper contains 4 intervals of 300 time steps each, in which 40,000 electrons were simulated, evolving according to the Vlasov-Maxwell set of equations. The time steps were chosen to be of $\omega_p \Delta t = 0.25$ where $\omega_p = \sqrt{\frac{ne^2}{m\epsilon_0}}$ is the ion plasma frequency. For each particle and every time step, the spatial trajectory and kinetic energy was recorded in arbitrary units. Figure 5.2 shows all trajectories of the simulated electrons in the magnetic reconnection set-up.

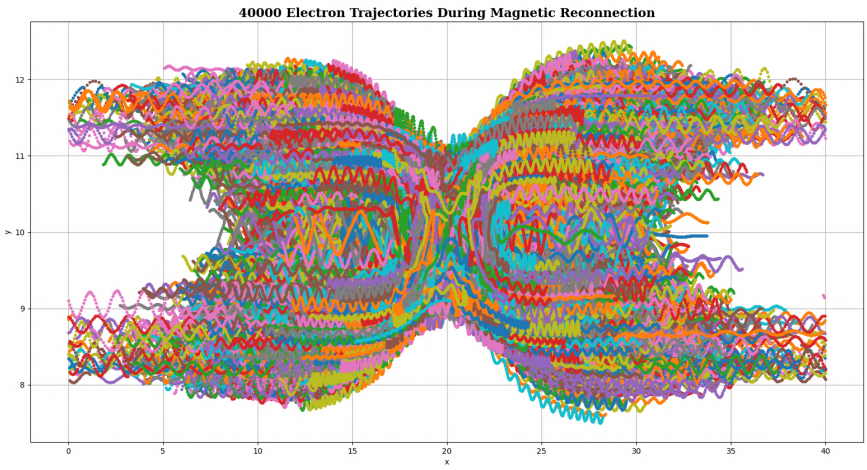


Figure 5.2: The entire data set plotted, with x and y in arbitrary units The general geometry of magnetic reconnection is visible, with many trajectories following the separatrices and many others crossing the x-point. It is evident, that a manual classification of trajectories is hardly possible with datasets of this size.

5.2 Pre-processing

The first step in the pre-processing of the data before the clustering algorithm is to remove the direct spatial representation of the trajectories. The authors' reasoning behind this is that if the clustering were performed directly on the trajectories in their original state, the algorithm might find clusters based on the starting positions, or classify all particles moving along the separatrices as similar and miss the more interesting behaviour like trajectories leading to acceleration (see figure A.2).

[1]The dataset and original MATLAB code can be found under `https://github.com/smarkidis/trajectory_classification_plasma`

For this, the origin of each particle is first removed from the data to obtain the particles' individual evolutions and then the evolutions are normalised by dividing the trajectory by the norm of the end-position. After this, the amplitude spectrum of the FFT is taken for the x- and y-positions of all trajectories and renormalised once more, in order to prevent a clustering on the strongest frequency, resulting in 150 Fourier modes, representing the positive frequencies of the transform, for each spatial dimension. On this pre-processed data with each Fourier mode in x and y as a feature, and each particle as a sample, PCA is applied, to obtain a more efficient representation of the data in the new feature space of principle components (see ch. 4.1). This represents the first significant deviation between the article and the actual implementation, in that, according to the authors, the PCA is run over the combination of Fourier modes on x and y, as well as the kinetic energy average of each particle. In practice however, the article's results do not match with those obtained, once the energy is taken into account. Instead, the energy is entirely neglected in every computational step towards clustering, and the PCA thus only yields principal components which are linear combinations of the x and y Fourier modes. Any clustering done on this data can only implicitly take into account the kinetic energy, as with constant time steps, the velocity and consequently the kinetic energy are encoded in the x- and y-representation of the data. Since the velocity is the derivative of the change of coordinates, it is not clear, whether this relationship could be picked up by the clustering algorithm without an explicit scheme being implemented.

With this new representation of the data projected on the principal components, the data is still 300 dimensional. For cost and efficiency reasons, the dimensionality of the data set has to be reduced further. By neglecting components that do not hold any significant information, this is now easily accomplished. To this end, the property of the eigenvalues of the original covariance matrix being the explained variance of the data along the corresponding eigenvector or principal component (ch. 4.1) is used. The first PC contains approximately 36.79% of the total variance. Plotting the variance explained by each component in blue, and the accumulated variance by all previous and the indicated PCs in orange in a so-called *pareto* plot, fig. 5.3 shows the informational significance of the components in the new representation.

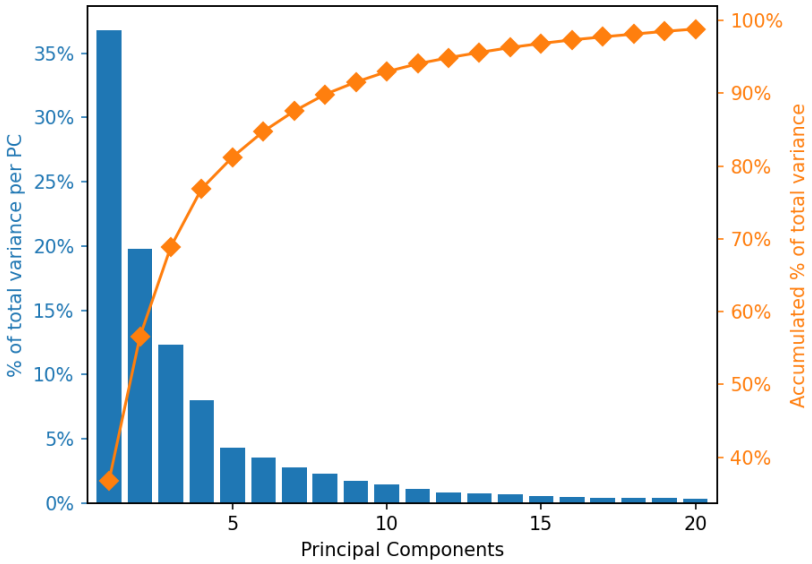


Figure 5.3: The variance explained by each principal component in blue, and the accumulated variance by all PCs up to that point in orange, for the case of 150 spectral modes of each spatial dimension and no energy included in the PCA.

With this information a reasonable compromise between cost of computation and loss of information can be made. In the article, the cut-off is made after the first 20 PCs, which hold 98.8% of the total variance of the data.

5.3 Classification

Instead of the original 900 features of each sample, due to the neglecting of the energy and the decision to keep only the first 20 principal components of the PCA, the data is now 20 dimensional. To classify the particles based on their trajectories by finding clusters in this data, the authors chose to use the in chapter 4.2 introduced k-means algorithm, with a maximum number of iteration $max_{iter} = 1000$ and run it 50 times with different initial conditions to find the clustering that minimises the WCSS. Instead of the elbow-approach to find the most likely number of clusters present in the data, they chose a different approach: they begin with a large prospected number of clusters of $k = 25$ and plot of each class the 25 most representative particle trajectories, that means the particles with the highest silhouette scores in each class. As visible in fig. 5.4, the data is not split into clearly distinguishable clusters, but instead the samples sit tightly together in feature space. This makes the selecting of representative trajectories important, as they are the ones exhibiting the least amount of properties shared with other clusters, and are thus the best candidates to represent a clusters uniqueness. By then iteratively reducing k, the merging of previously separate trajectory classes can be observed. If there are several classes that present identical or very similar features, this process helps eliminating these duplicates, until only the most fundamental trajectory categories are left. At this point, the value of k found should correspond to the number of fundamental interactions known from the literature [14, 15]. In the article, that is the case, as they find 12 fundamental trajectory classes, which correspond to distinct trajectories

known from the literature. The reason the authors chose this approach instead of the more commonly implemented elbow-approach, is due to the physical reality of the system, where realistically most trajectories do not stem from a single type of interaction in the plasma, but rather always a combination of subsequent interactions which mix trajectory classes and make a clear distinction between them impossible.

Even in cases where clusters in the data are clearly separated, the outcomes of clustering are heavily influenced by the choice of distance metric employed by the clustering method. The selection of a specific distance metric, such as Euclidean distance, may not accurately represent the spatial geometry of a high-dimensional space, as present in the processed 20 dimensional [16]. However, the Euclidean distance metric is technically the only metric utilised in the k-means algorithm [17], and other metrics are not implementable in the available scikit-learn k-means method for python codes. The original work of S. Markidis et. al. was done in Matlab, which presents the option to use different distance metrics. To be able to qualitatively and quantitatively compare the results of the article with my own, I adapted the Matlab code to use the same Euclidean distance metric I used in the python implementation. Figure 5.4 shows the clustering found in the projection of the data onto the first two principal components using the Euclidean distance metric. Evidently, a clear separation of the samples in the 2D representation of $PC1$ and $PC2$ is not visible and seems arbitrary.

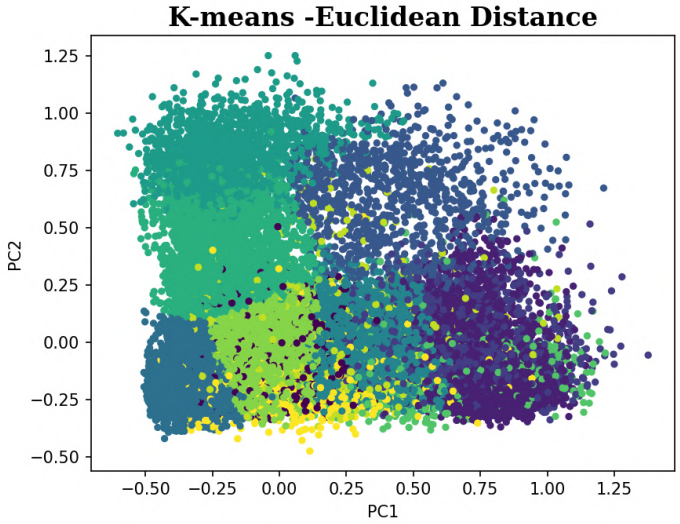
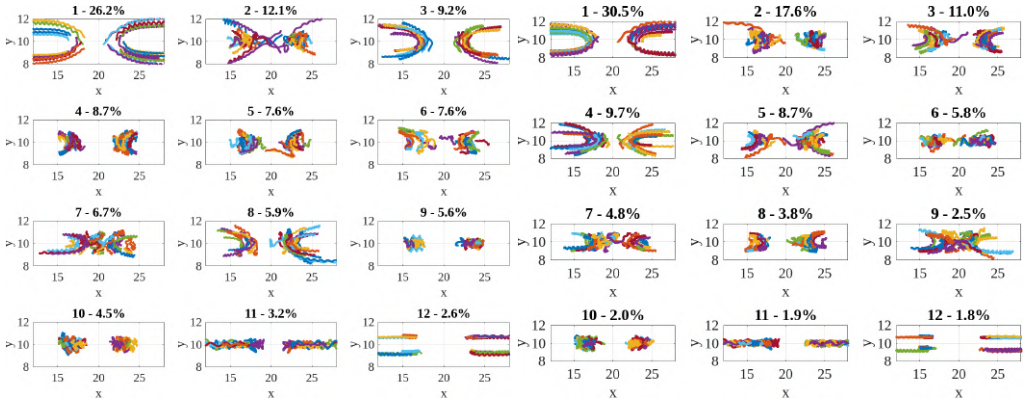


Figure 5.4: 2D projection on the first two principal components of the clustered data with 150 Fourier modes per spatial dimension and no energy, using k-means clustering on a Euclidean distance metric. Each colour represents one of the 12 clusters.

Clearer separations might be found in the full 20 dimensional space, which unfortunately is not easily accessible for human analysis. Different distance metrics, namely cosine (dis-)similarity used in the original work, and Euclidean distance yield slightly different clusterings in the 2D projection (compare fig. A.10). An interesting result of the comparison of the final clustering between the two different distance metrics is that the trajectory classes found are very similar, however a deviation in the population of the respective classes can be observed. Figures 5.5a and 5.5b show this in detail.



(a) Trajectory classes with cosine similarity (b) Trajectory classes with euclidean distance

Figure 5.5: The 12 classes found by the k-means algorithm with $max_{iter} = 1000$ and different distance metrics employed. The classes are comparable but the population of the classes seems to vary strongly. The numbers over each plot signal the class ID, and the relative population of that class, compared to the total number of particles in the simulation.

The difference in cluster population is interesting and suggests that experimentation on different metrics for high-dimensional feature space can yield insights into more efficient or more accurate classifications. For now, the knowledge that Euclidean distance clusters the trajectories into similar categories as cosine similarity is sufficient to use this metric for further exploration of the intricacies of the algorithm and aligns well with the expectation of the metrics to behave similarly in high dimensions [18]. With this first adaptation of the original algorithm conducted, there are many other choices made in the original article, that demand validation or justification. This will be the core of the next chapter.

Chapter 6

Adaptation of the Algorithm

The particle trajectory clustering described in the article *Automatic Particle Trajectory Classification in Plasma Simulations* by S. Markidis et. al. has been successfully reproduced in Python. Several discrepancies between the procedures explained in the article and the actual data used have been uncovered. To understand what effects these differences have, what could be done to improve the physical significance of the classes, and what other particle classes might be found, the algorithm was adapted and the results were studied.

6.1 Fourier Transform and Cut-Off

The choice to use a Fourier transform on the particle trajectories is important for the clustering. An aspect that makes clustering with spectral, instead of spatial information, seem like a good choice is the natural gyrating behaviour of the electrons in a magnetic field. This leads to the sinusoidal shape of the trajectories seen in fig. 5.2. For non-relativistic particles (like the ones in the simulations studied here) the gyro-frequency only depends on the strength of the magnetic field:

$$\Omega_{c,e} = \frac{eB}{m_e} \tag{6.1}$$

The radius of the gyration around a field line, called Larmor radius, depends on the velocity component perpendicular to the field line $v_\perp$ and the magnetic field B:

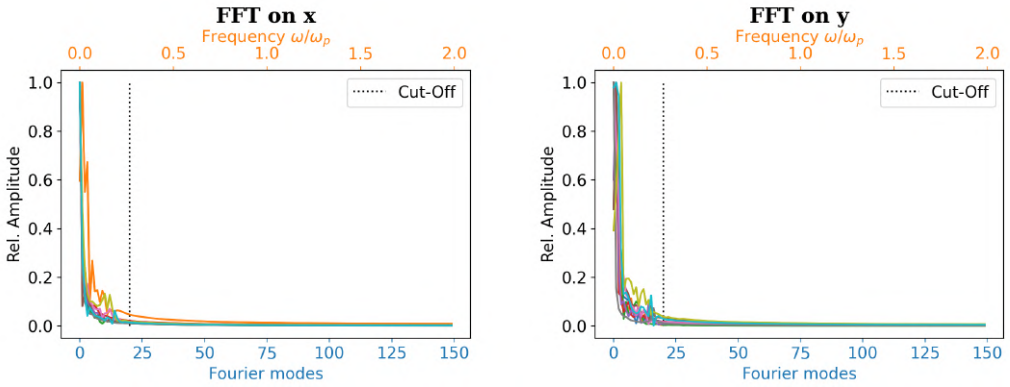
$$r_L = \frac{v_\perp}{\Omega_{c,e}} = \frac{m_e v_\perp}{eB} \tag{6.2}$$

An acceleration of a particle such that $v_\perp$ changes results in a change in r_L, that is represented as the amplitude of the gyro-frequency in the Fourier transform. This change in amplitude is picked up by the Fourier transform. However, a change in B similarly changes r_L and thus the amplitude in the spectral representation of the trajectory. Without information on the field at each position, it is hence not possible to attribute a change in Larmor radius as picked up by the transform to an acceleration of the particle. The employment of Fast-Fourier-Transform (FFT) on the particle trajectories is therefore not necessarily the best method of picking up on accelerating processes in the plasma, as the energy information is only implicit. Alternatives that I have explored, such as the clustering on the untransformed standardised energy, will be discussed in sections 6.3, 6.4, and 7.

In [6] the authors chose to apply the Fourier transform to the trajectories and include all 150 Fourier modes. To verify if all modes are relevant to the clustering and what effect a cut-off at lower frequencies might have on the results, I ran the code

over different amounts of Fourier modes while, as far as possible, keeping all other variables, like number of PCs to project the data on, constant. To fully compare the different scenarios, I decided to visualise the projection of the clustering on the first two principal components, the *pareto* plots to understand what effect the inclusion of more modes has on the PCA, as well as the resulting trajectory clustering. The code was run over the following number of included modes per spatial dimension: 1; 2; 5; 10; 15; 20; 25; 30; 50; 100; 120; 150

The shape and the clustering of the projected data changes for each additional mode, until 15 modes for each dimension are included (see figures A.4a - A.4d in Appendix A). With 15 modes and more, the same clustering as with the whole spectrum could be recovered (fig. A.4 - A.6). By plotting the Fourier transforms of the coordinates in figures 6.1a and 6.1b, this can be easily understood, as the amplitude of frequencies higher than $0.2\omega_p$, or 15 modes on x, and higher than $0.27\omega_p$, or 20 modes on y vanishes. The spectra of all particles are shown in figures A.10a and A.10b. Physically, most information on the trajectory of particles is stored in the first few modes. Accordingly, by only looking at the first frequency modes of the Fourier transforms, the general shape of the trajectory is retained, but higher order oscillations are smoothed out. To maximise preserved information while minimising the stored data, I cut-off all Fourier modes after the first 20 for both coordinates.



(a) Fourier transform on x-positions of 10 selected particles.

(b) Fourier transform on y-positions of 10 selected particles.

Figure 6.1: Plot of Fourier spectra on both spatial coordinates shows the significance of the first Fourier modes, and the vanishing frequencies above a threshold of $0.27\omega_p$.

6.2 PC Cut-Off

After the pre-processing of the data by means of standardisation and FFT, the dimensionality reduction using PCA is the next important step that is performed. The decision how many principal components to use to represent the data controls both the speed of the following clustering process and the quality of the clustering. The authors settled on 20 PCs, which include 98.8% variance. Naturally, different values for the cut-off have to be implemented and the clustering results compared, in order to find how much variance has to be preserved to give a good clustering, and whether or not the same classes can be found with fewer than 20 components. I employed the same methodology as in section 6.1, where I looked at the clustering

projected onto the first two PCs, the *pareto* plots, and the trajectories of the 25 most representative particles of each class, while changing the the amount of principal components to project the data on and keeping the 20 Fourier modes per coordinate constant. I found, that using only 8 PCs, which contain 90.1% of the original total variance, is sufficient to recover the classes found with 20 PCs (see figures A.7-A.9). An increase of the dimensionality on the other hand, does not have any significant impact on the clustering.

6.3 Including Energy Average

As already established, the energy only implicitly enters the clustering algorithm. This fact is reflected in the classes found in the particle trajectories, when visualising the energy evolution of the most representative candidates for each class in figure 6.2.

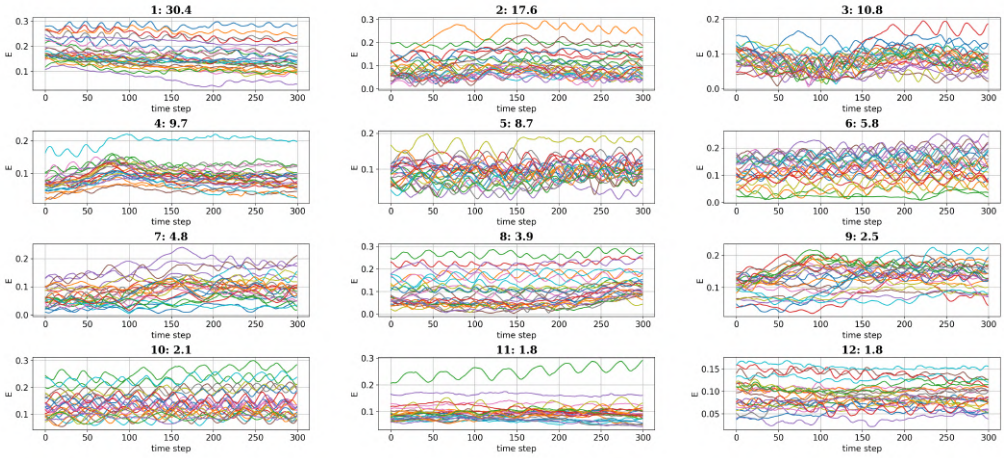


Figure 6.2: Energy evolution of the 12 classes found by clustering over the original 150 Fourier modes of each coordinate with 20 PCs and no explicit energy information included. The classes are heterogeneous and not clearly distinguishable.

No clear characteristics can be distinguished between the different classes, and each class shows no homogeneous behaviour. Acceleration processes where a clear increase in the kinetic energy is expected are hence not identified as a distinct class from other processes with, on average, conserved kinetic energy. The oscillation observed in the kinetic energies can physically be interpreted as the gyrating motion of the particles within an electric field, that decelerates them during one half of the gyration, and accelerates them during the other. To address the issue of the failed clustering of similar energy evolutions of particles, one approach that I explored was the inclusion of the average energy of the particles in the clustering, which was claimed to also have been done in [6]. The average energy of each particle is appended to the list of Fourier modes on x and y, and the principal components are computed. The following clustering yields no significant improvement in both spatial and energy evolutions, to the clustering done on only spatial information (see comparison between classes in figures A.11 and A.12).

6.4 Clustering on Energy

The trajectory classes found through clustering done on the spectral representation of the trajectories and the average energy of each particle still do not demonstrate homogeneous energy evolutions. Since the evolution of the particles' energies is fundamental for acceleration processes, including the full energy evolution in the clustering algorithm is a logical next step. Understanding how a clustering on only the energy information for each particle would perform is critical in finding an improved classification method. I took the energy for each particle and standardised it by division of the particles' maximum energies, so the energies would range from 0 to 1. The representative energy evolutions of the 12 classes found by the k-means clustering on the standardised energy in fig. 6.3 now show clear trends and a shared evolution within their classes. Easily identifiable acceleration processes are present in classes **2**, **6**, **11** and **12** while the particles in class **9** are being decelerated. Other evolutions shared by particles like an acceleration and subsequent deceleration in class **8** are also found, suggesting a shared history of interactions.

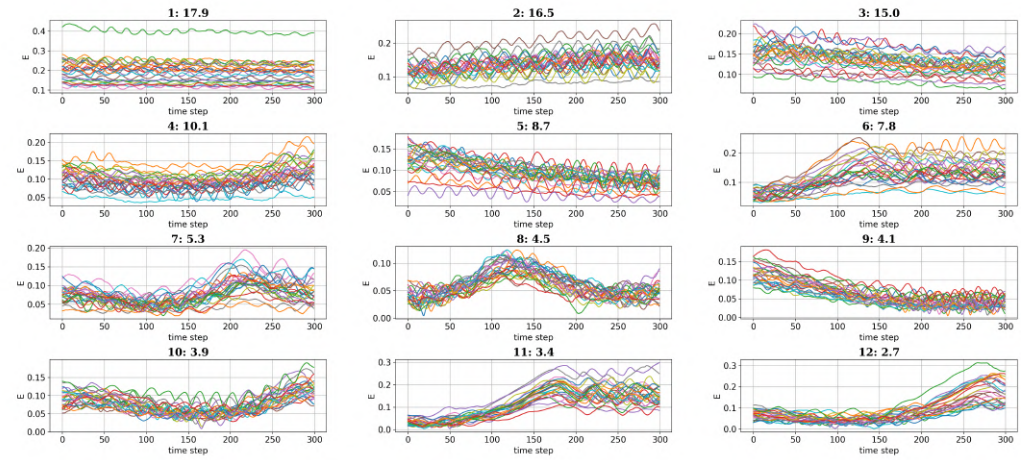
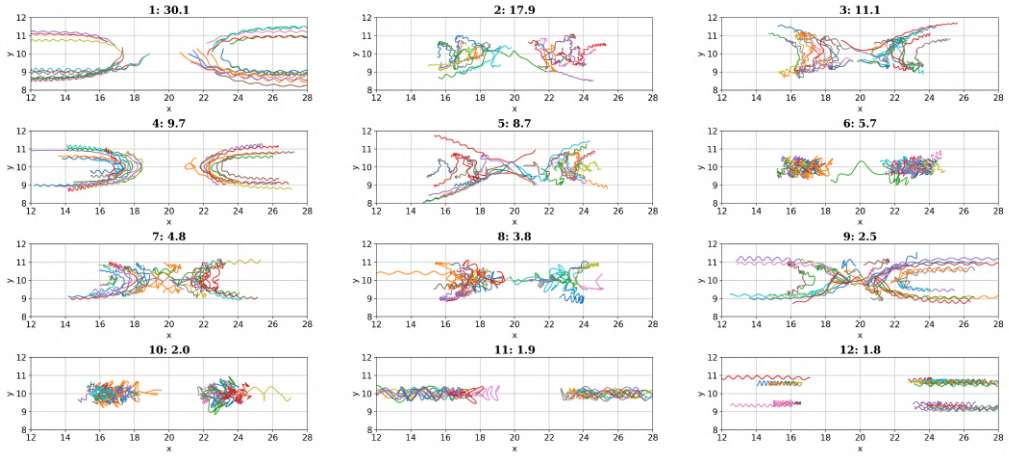


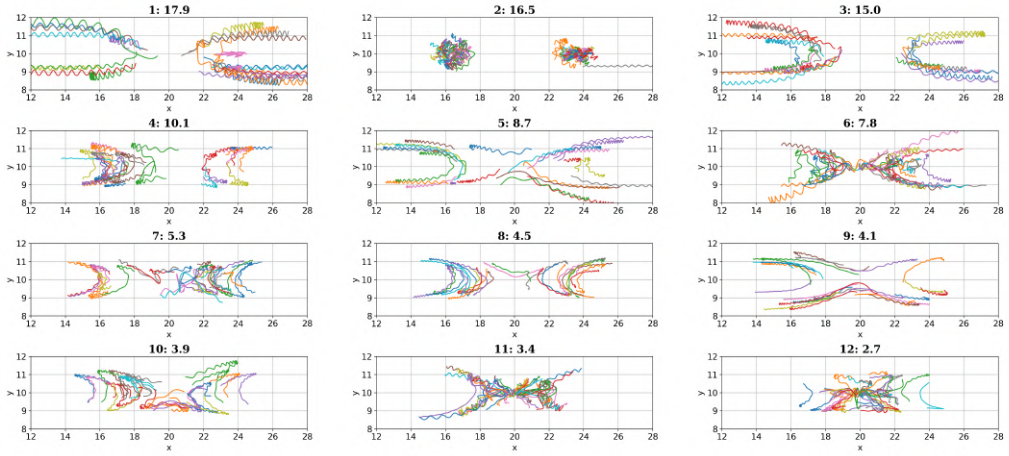
Figure 6.3: Energy evolutions of 12 trajectory classes found by clustering over the 300 energies saved for each particle. Clear acceleration and deceleration trends are visible. Other more complex evolutions like an acceleration and subsequent deceleration shared by many particles are also found.

Under inspection of the representative trajectories of the 12 classes found by the k-means clustering on the energy, interesting differences to the ones found previously come to light. Figure 6.4 shows the new trajectory classes and the ones found by clustering over the trajectories and average energies. Evidently, new classes were identified, that either were not found before, or were thought to be parts of the same class, and thus merged together by reduction of k. Examples of this might be the classes **b.5**, **b.7**, **b.8** and **b.9**, which, to the human eye, look very similar and could be mistakingly merged into classes **a.3** and **a.5**. The energy evolutions of the classes, however, clearly show that even though the trajectories may appear similar, the processes along them are very distinct. Another fascinating insight provided by this representation of the new clustering, is that we cannot reproduce all the classes previously found by only regarding the trajectores. This can have two implications: Firstly, the actual spatial evolution of the particles is not very important for its overall evolution, as some very distinct trajectories, like in **a.11** or **a.12**, do not reappear in the classes found just by regarding the energy evolution. Secondly, the fact that not

all of the previously found classes have been recovered with the energy-approach means that there are more than 12 classes to be found. This interpretation of the results is coherent with the observation made before, that it seems as if multiple classes that have varying energetic behaviours share similar trajectory characteristics. This will be picked up on by purely trajectory clustering. In order to reproduce all previous classes in the energy-approach, a higher number of classes would have to be used.



(a) Clustering on trajectories and average energy.



(b) Clustering on energy.

Figure 6.4: Comparison between 12 trajectory classes found by clustering over the original 150 Fourier modes of each coordinate with 20 PCs and the average energy in the clustering, and clustering only over the energy. The found classes are practically identical, with only marginal differences in population.

Chapter 7

Discussions

Automated classification methods have a big potential to bring forward new fields of research, for instance, in the study of particle acceleration in plasmas, where researchers typically analyse particle trajectories in order to identify groups of particles exhibiting common or unique behaviour. Due to the vast amounts of data that need to be analysed, this analysis is typically restricted to a relatively small subset of trajectories, usually focusing on particles with high energy levels. As a result, acceleration mechanisms that do not produce the highest energies may go unnoticed. In this context, the automatic and unsupervised clustering of trajectories using machine learning techniques offers a promising opportunity for exploring new research directions and potentially revealing previously unknown mechanisms.

During my internship I critically analysed the workflow proposed by S. Markidis et. al. in [6] for such an automated classification of trajectories, reproduced their results and found discrepancies between the code and the reported methods used. I found fundamental flaws and proposed improvements and adaptations to the algorithm. I have not developed the workflow myself, but extended upon the already existing one. The work has been done on a given set of data from a PIC simulation. However, the same workflow and algorithm is appliable to other types of data and particle simulations, like fluid-based or hybrid models. In the future, the usage of different clustering algorithms beside k-means has to be studied. There are many algorithms that are promising as a suitable substitute, like the Mean-shift and DBSCAN algorithms, which both do not require the specification of the number of clusters to be found within a dataset. The clustering results have to be compared quantitavely, for example by means of directly calculating the silhouette score or the Calinski-Harabasz index for every particle and potentially summing over the candidates for each class to get an overall class-performance. Another idea I want to pursue during my remaining time at LERMA is a hybrid clustering algorithm, where I aim to find the number of clusters k in the data using an algorithm like Mean-shift, and then using that k to apply k-means on the data.

Appendix A

Appendix

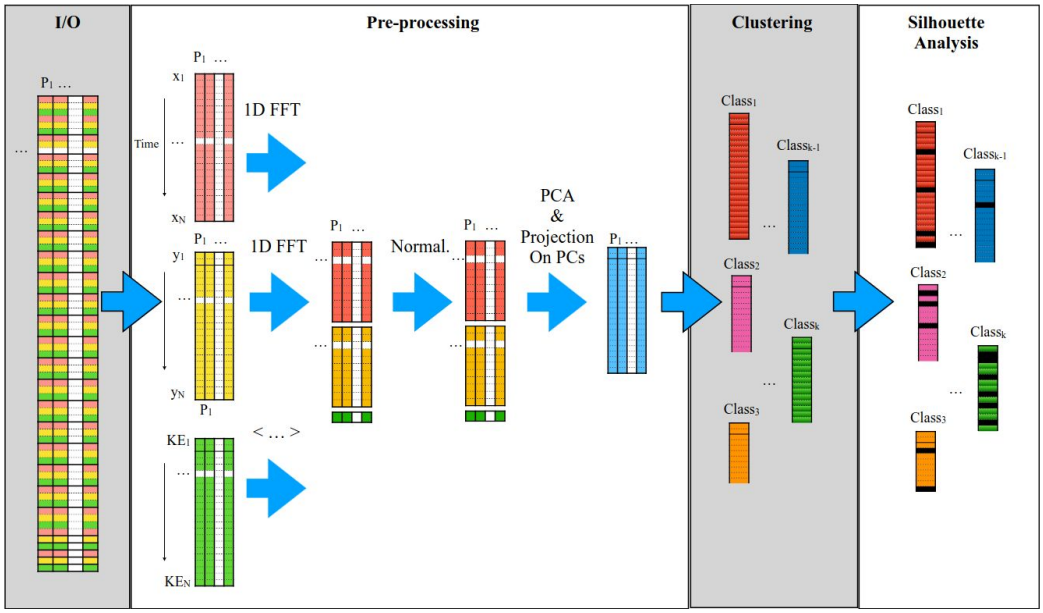


Figure A.1: The workflow proposed in the article Automatic Particle Trajectory Classification in Plasma Simulations [6].

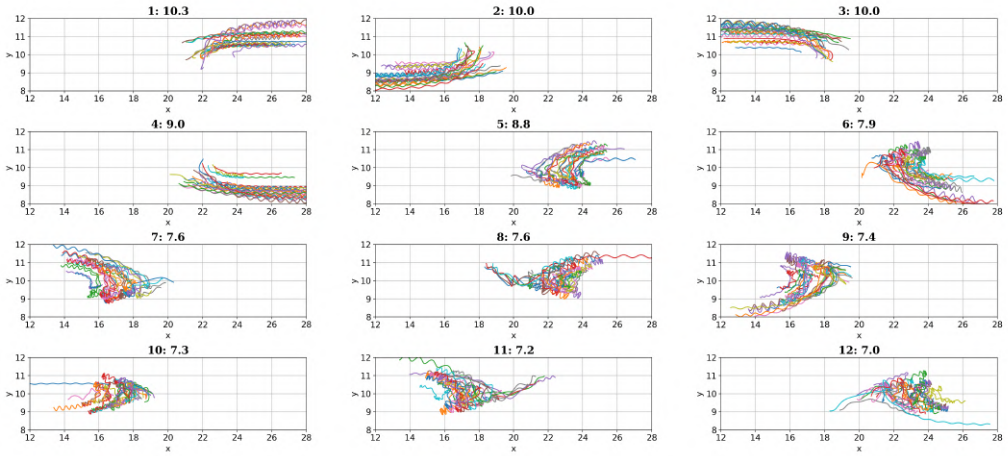
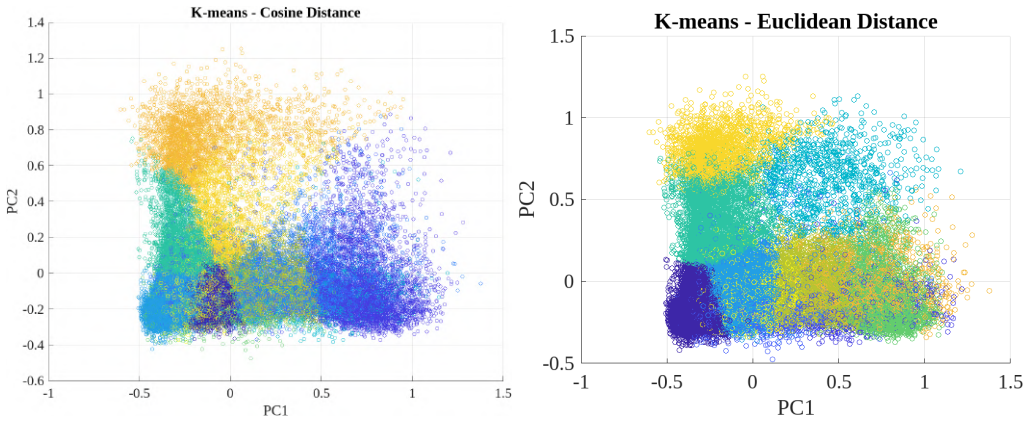
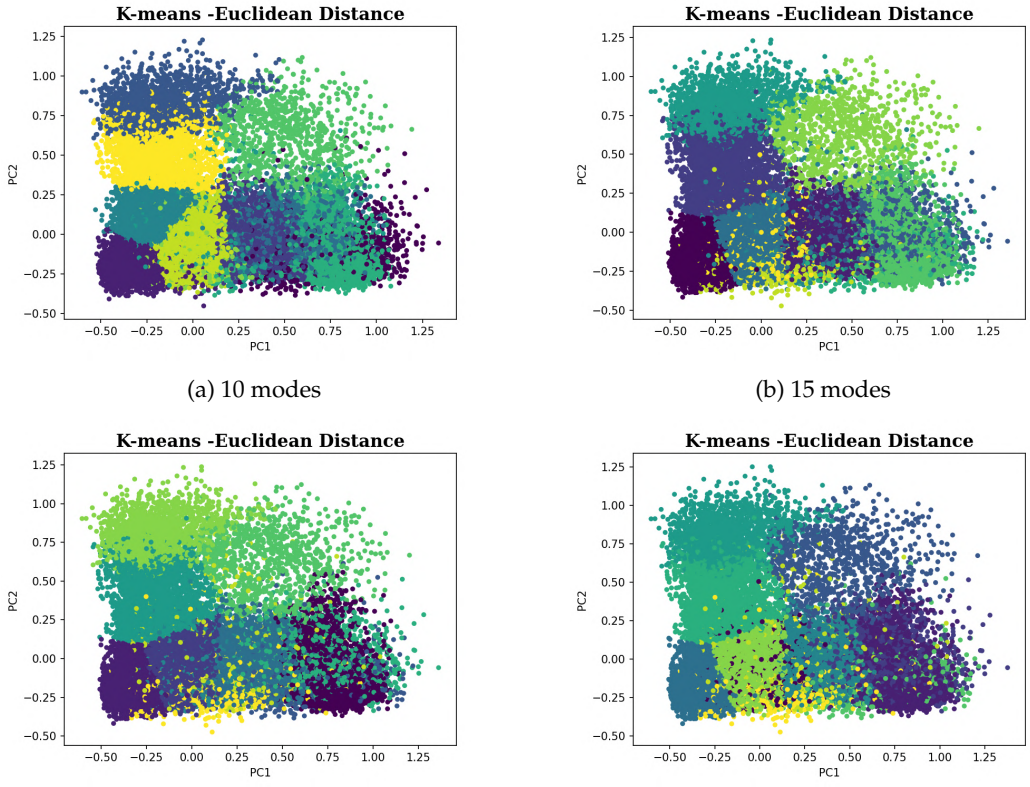


Figure A.2: The trajectory classes found by applying the clustering algorithm on the standardised data without FFT.



(a) Cosine similarity (b) Euclidean distance

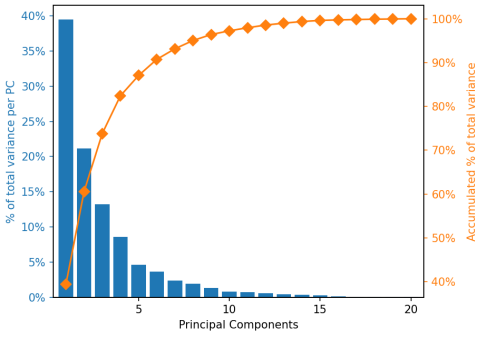
Figure A.3: Comparison between clustering using Cosine similarity and Euclidean distance metric in 2D projection of the data on PC1 and PC2. The PCA was done on 150 Fourier modes for each spatial dimension and no energy was included.



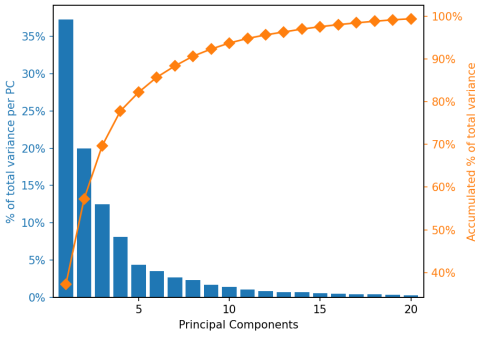
(a) 10 modes (b) 15 modes

(c) 20 modes (d) 150 modes

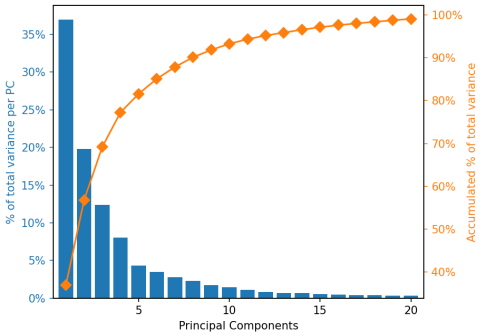
Figure A.4: Projected data on PC1 and PC2 after clustering. Between 10 and 15 included Fourier modes per spatial coordinate, the clustering still visibly changes. After 15 modes, no change is visibly discernible anymore.



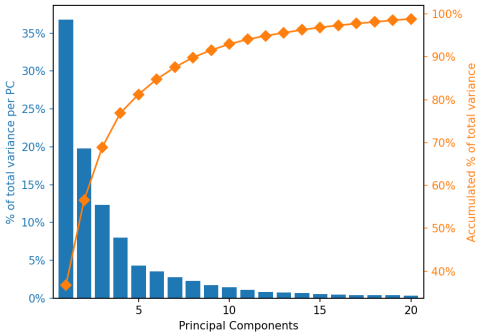
(a) 10 modes



(b) 15 modes

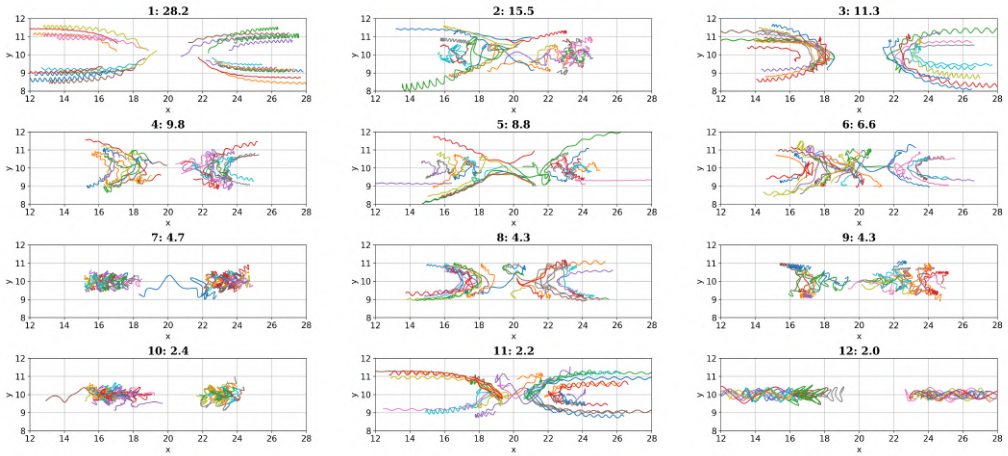


(c) 20 modes

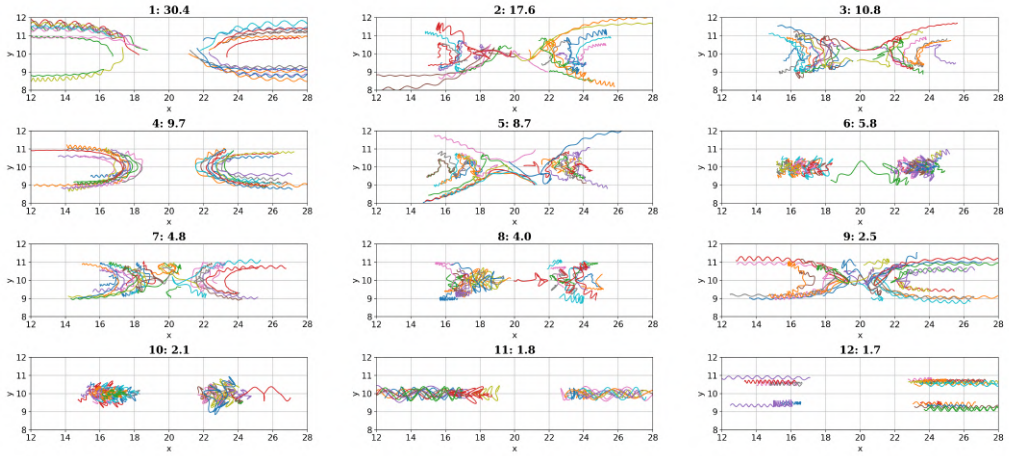


(d) 150 modes

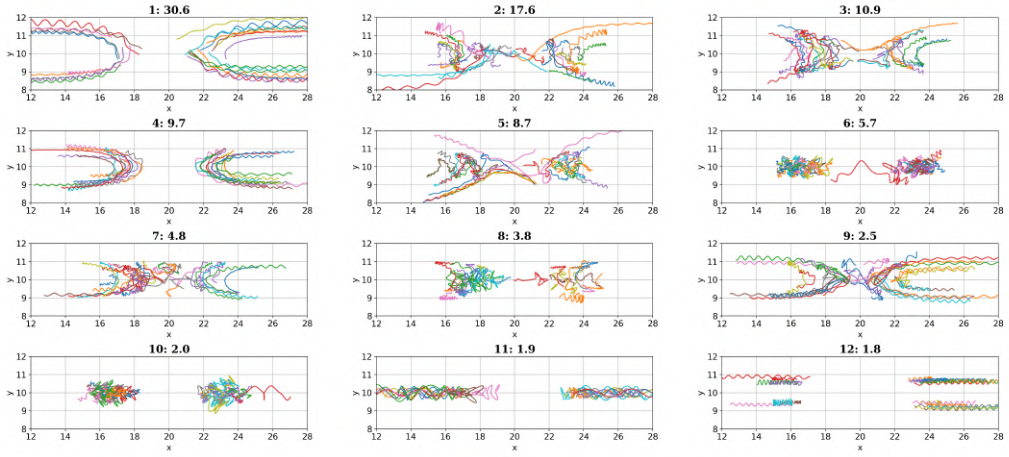
Figure A.5: Pareto plots for 10, 15, 20 and 150 included Fourier modes. Between 10 and 15 included Fourier modes per spatial coordinate, the variances of the principal components still change noticeably, especially in the later components. After 15 modes, no change is visibly discernible anymore.



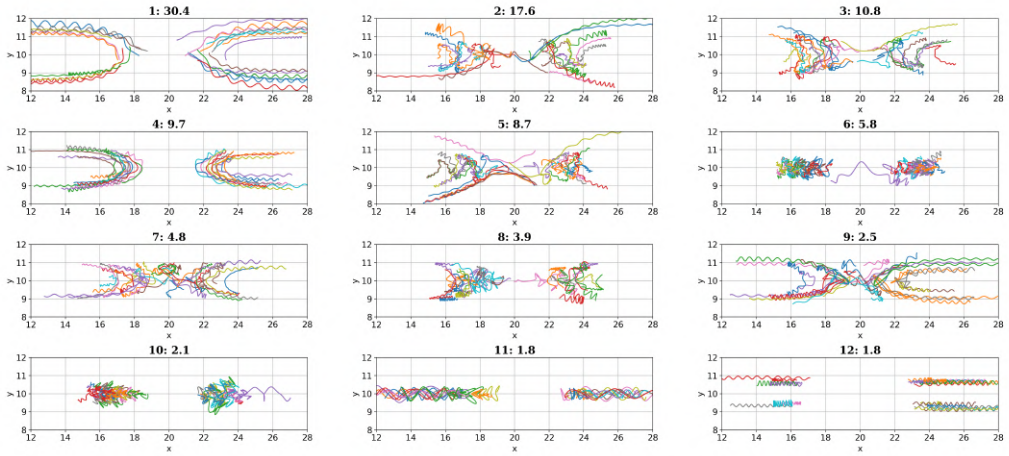
(a) 10 modes



(b) 15 modes

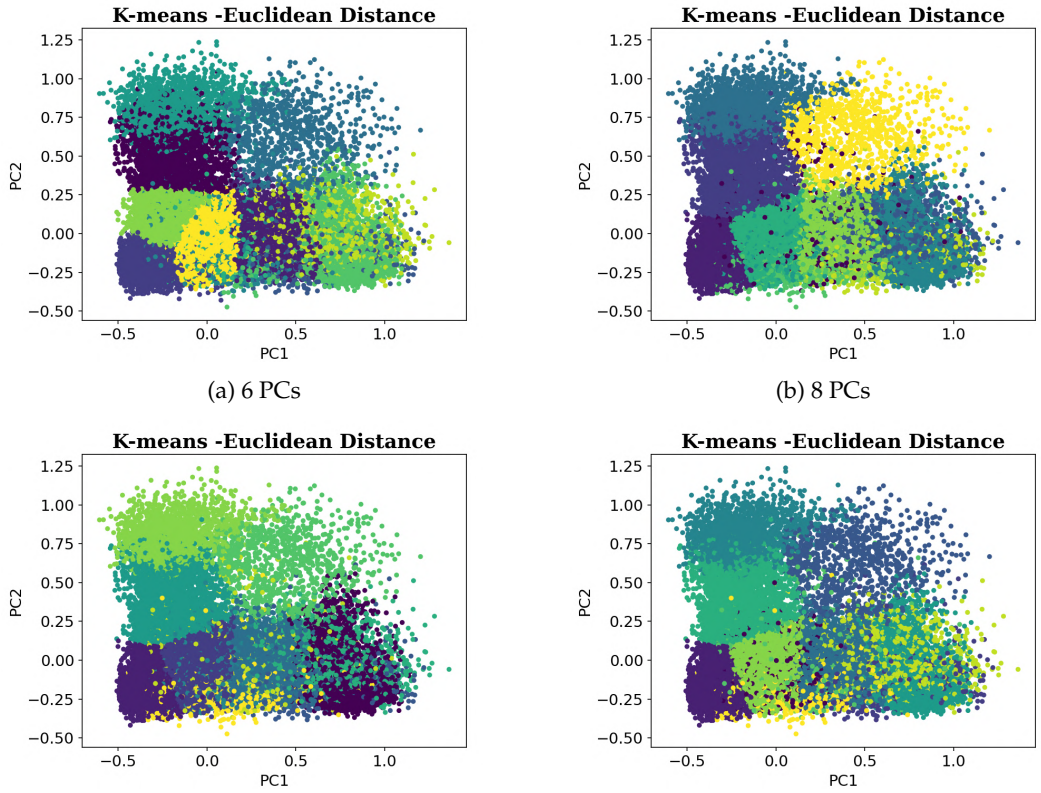


(c) 20 modes



(d) 150 modes

Figure A.6: The 25 most representative particle trajectories of each class found by the clustering algorithm. The biggest change appears between 10 and 15 included Fourier modes per spatial coordinate, where the clustering still visibly changes. After 15 modes, only individual trajectories change their designation, and anomalies appear and disappear.



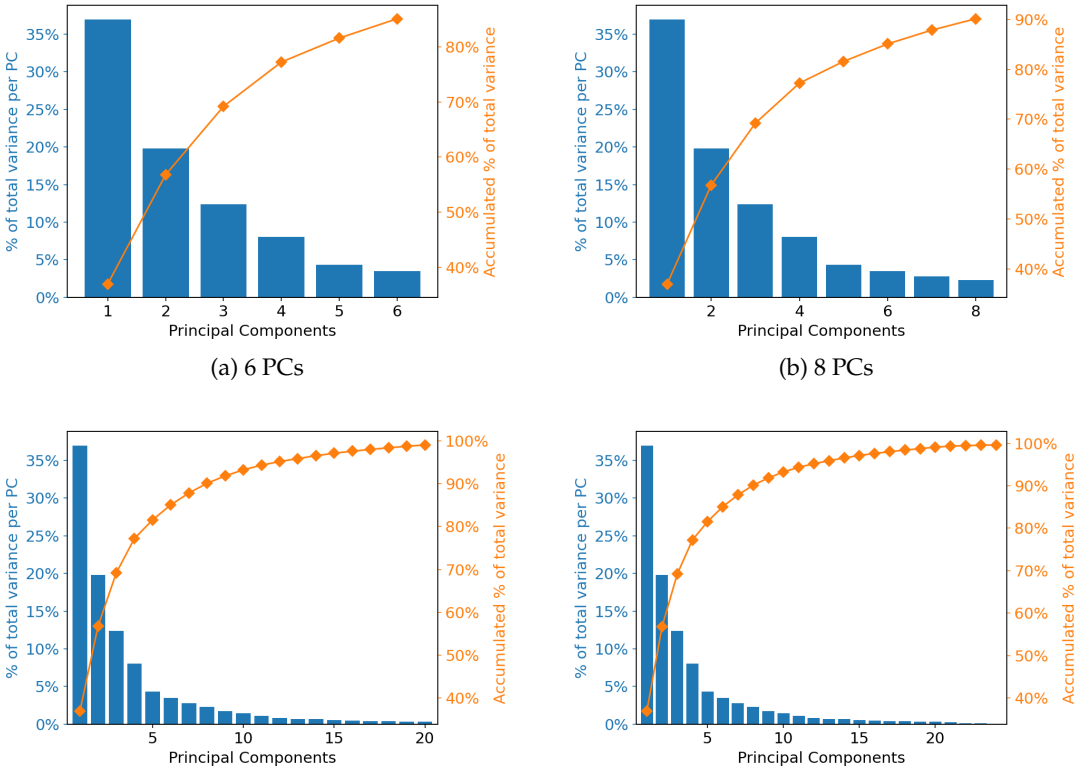
(a) 6 PCs

(b) 8 PCs

(c) 20 PCs

(d) 24 PCs

Figure A.7: Projected data on PC1 and PC2 after clustering for the projection of the data on 6, 8, 20 and 24 principle components, and 20 Fourier modes per spatial dimension. Between 6 and 8 included principal components, the clustering still visibly changes. After 8 PCs no change is discernible anymore. 24 PCs does not give a different clustering, or visibly changed projection, than 20 or even 8 PCs.



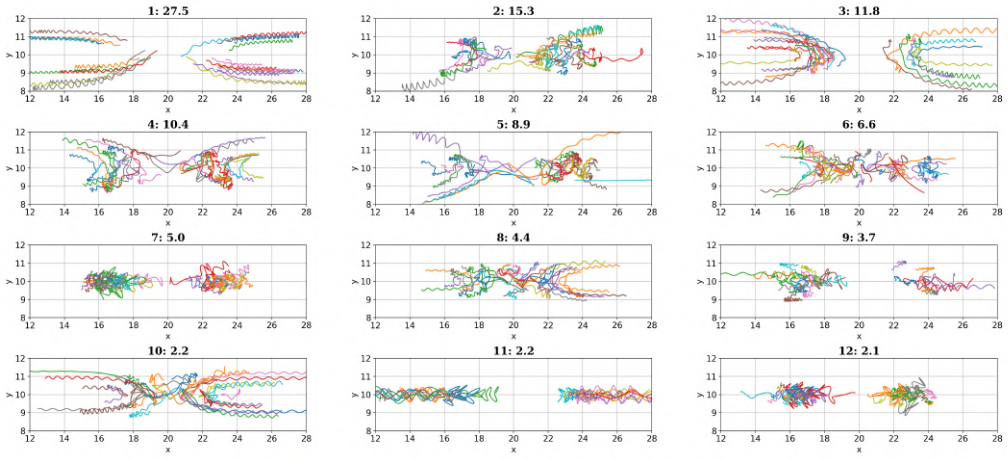
(a) 6 PCs

(b) 8 PCs

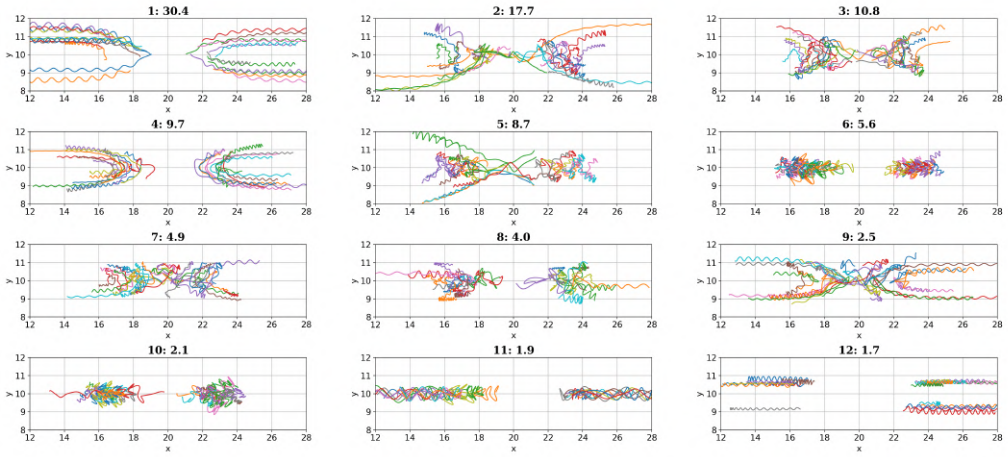
(c) 20 PCs

(d) 24 PCs

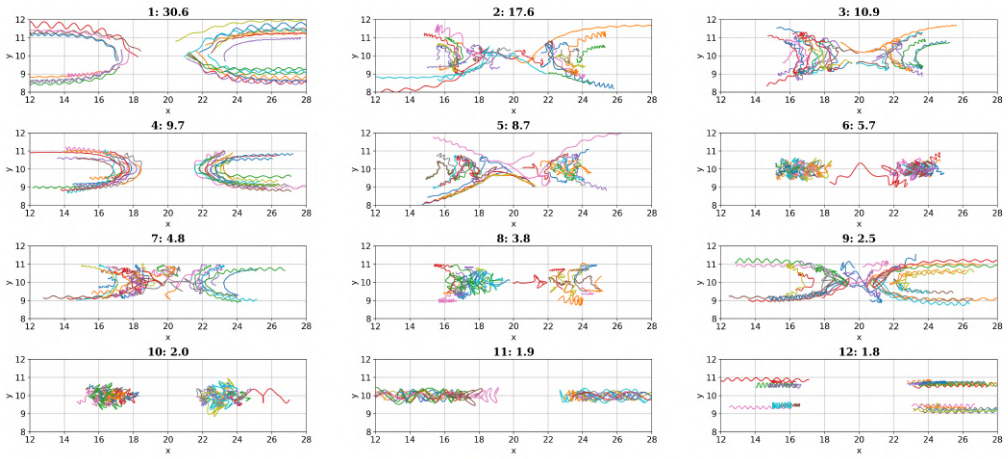
Figure A.8: Pareto plots for projection of data on 6, 8, 20 and 24 principal components. Between 6 and 8 included PCs, the variances do not change much, the two additional PCs in A.8b each carry around 2.5%, elevating the accumulated variance to 90% of the total variance. Between 8 and 20 PCs, the accumulated variance increases by 9%. The additional 4 principal components included in the last case describe around 0.6% of the total variance.



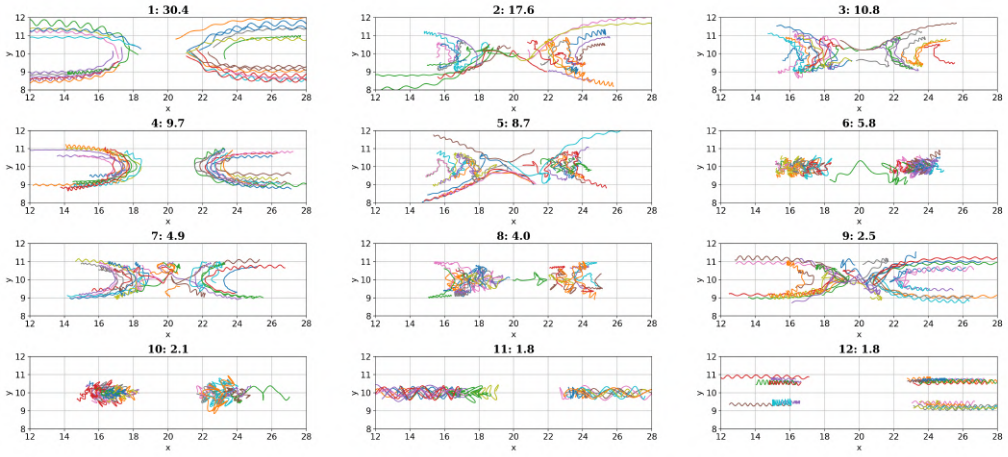
(a) 6 PCs



(b) 8 PCs

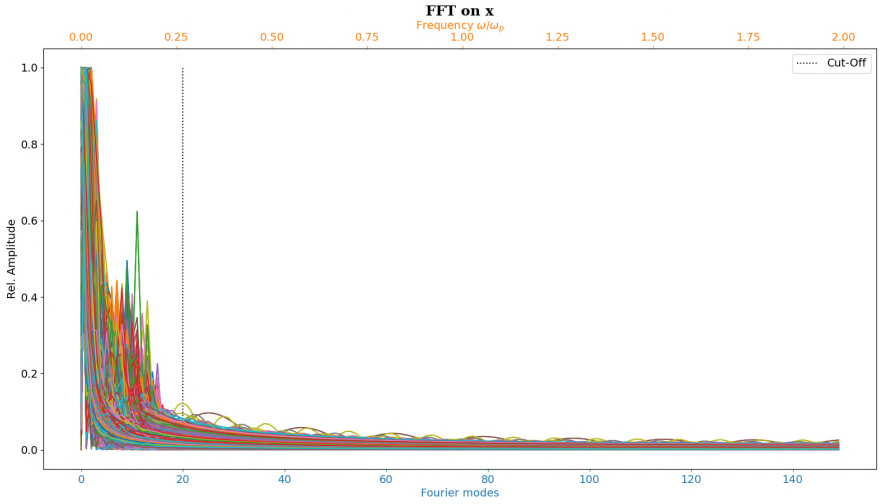


(c) 20 PCs

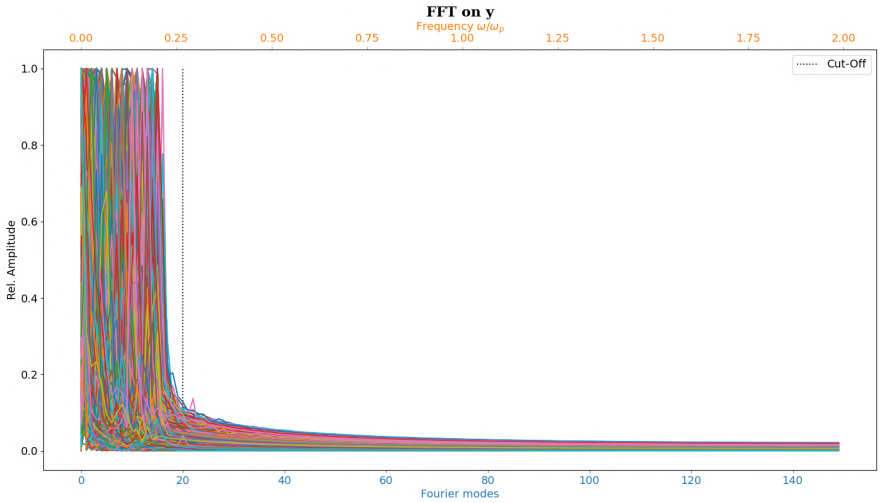


(d) 24 PCs

Figure A.9: The 25 most representative particle trajectories of each class found by the clustering algorithm for the projection of the data on 6, 8, 20 and 24 principal components. Between 6 and 8 included PCs, the trajectory classes and their populations change starkly, despite an increase of explained variance by only 5%. For the remaining PCs, the additionally explained variance does not change much anymore; the classes of fig. A.9b, A.9c and A.9d are both in trajectory behaviour and population practically identical. Only individual trajectories appear or disappear from the classes, or their 25 most representative trajectories.

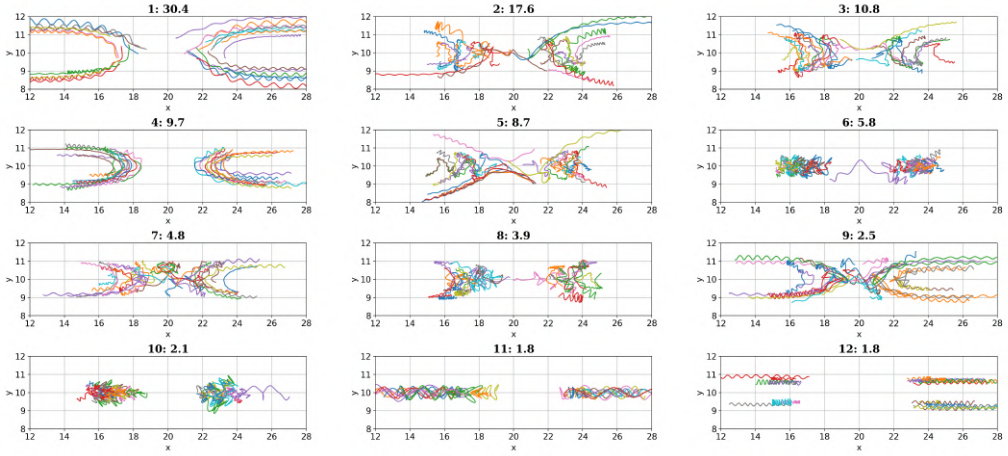


(a) Fourier transform on x-positions.

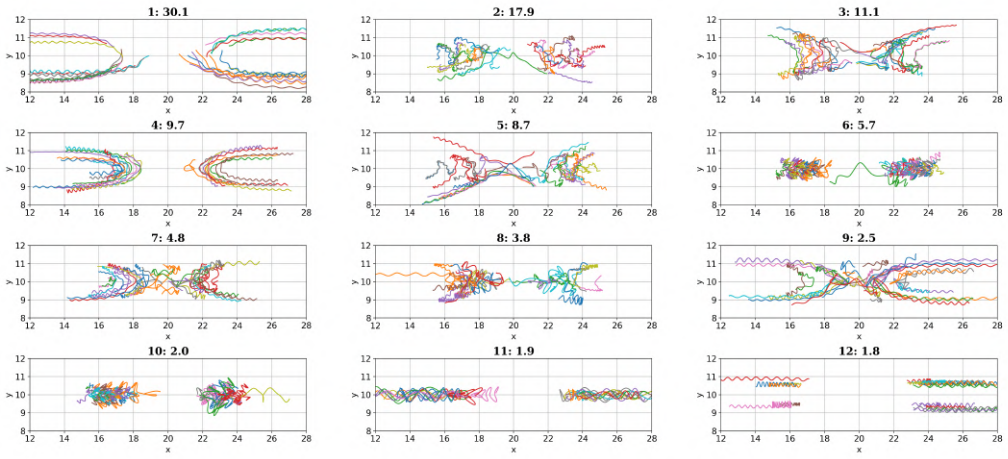


(b) Fourier transform on y-positions.

Figure A.10: Plot of Fourier spectra on both spatial coordinates shows the significance of the first Fourier modes, and the vanishing frequencies above the chosen cut-off.

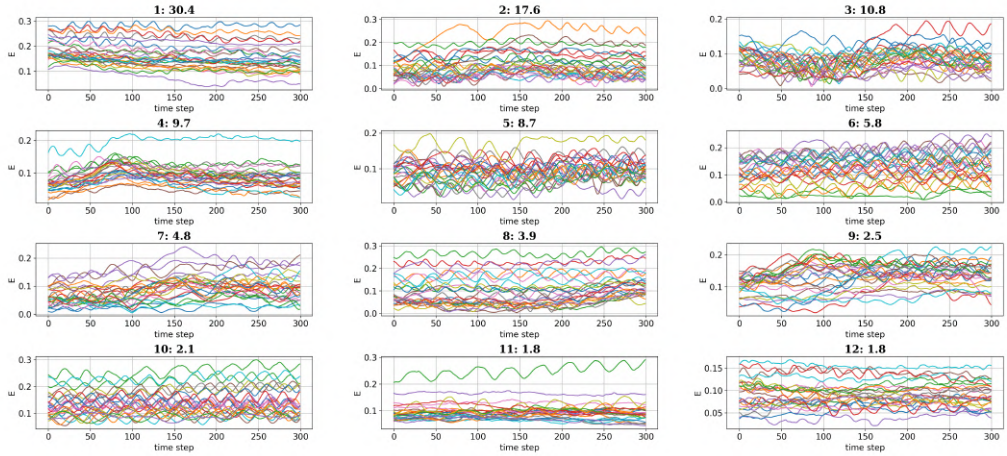


(a) No energy information included.

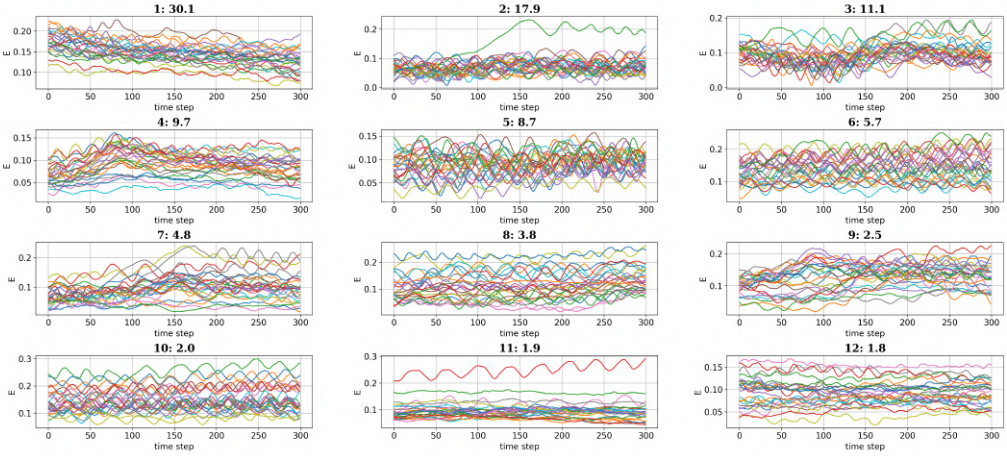


(b) Average energy included.

Figure A.11: Comparison between the 12 trajectory classes found by clustering over the original 150 Fourier modes of each coordinate with 20 PCs and no explicit energy information included, and the clustering found by inclusion of the average energy in the clustering. The found classes are practically identical, with only marginal differences in population.

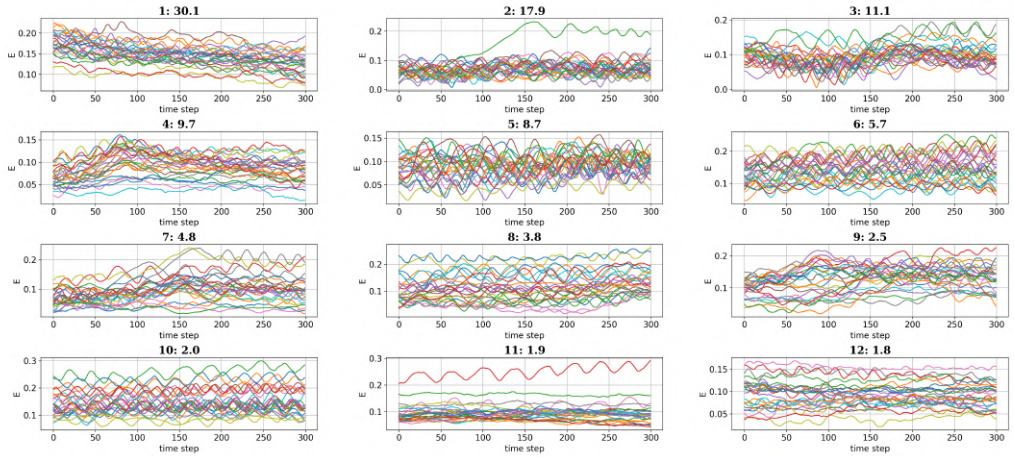


(a) No energy information included.

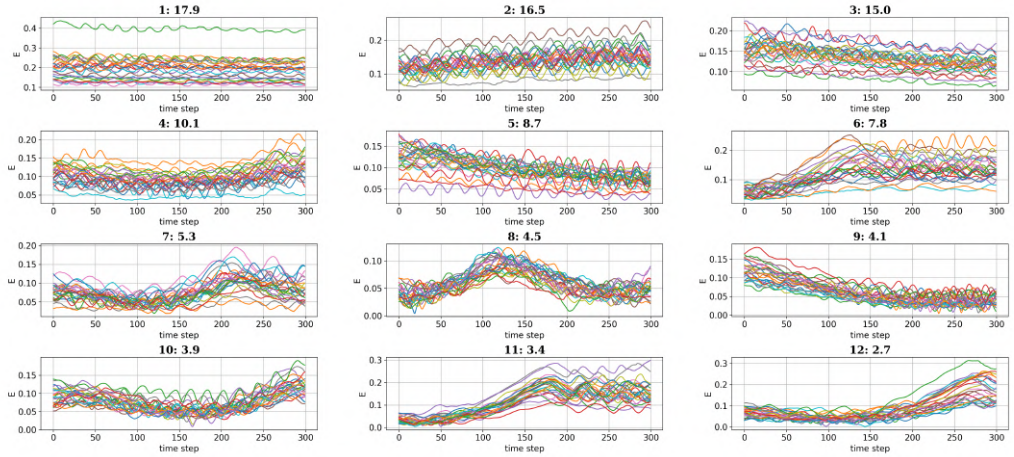


(b) Average energy included.

Figure A.12: Comparison between the energy evolutions of the 12 trajectory classes found by clustering over the original 150 Fourier modes of each coordinate with 20 PCs and no explicit energy information included, and the clustering found by inclusion of the average energy in the clustering. In both cases the classes are heterogeneous and not clearly distinguishable. The inclusion of the average energy in the clustering does not seem to have any effect on the clustering.



(a) Clustering over spectral trajectories and average energy.



(b) Clustering over energy.

Figure A.13: Comparison between the energy evolutions of the 12 trajectory classes found by clustering over the original 150 Fourier modes of each coordinate with 20 PCs and the average energy included, and the clustering found by using only the 300 energies saved for each particle. In the clustering over the energy, clear acceleration (b.2,6,11,12) and deceleration (b.9) trends are visible. Other evolutions shared by many particles like an acceleration and subsequent deceleration (b.8) are also found, suggesting a shared history of interactions.

Acknowledgements

I want to express my gratitude towards Prof. Dr. Andrea Ciardi, who proposed this project to me and who was always there to answer my questions and suggest new papers to read. Without his guidance and support my internship would not have been nearly as productive, insightful or entertaining. Thank you also to Dr. Weipeng Yao, who was kind enough to engage in discussions about my work and propose papers to learn more about some intricacies of plasma simulations. I'd also like to thank the entire research group for the warm welcome and fun discussions about everything from plasma physics to quantum computers during lunchtime. Finally, a thank you is also owed to the Sorbonne Centre for Artificial Intelligence, who kindly provided me with financial support throughout my internship.

Bibliography

[1] O. Chapurin and A. Smolyakov. "On the electron drift velocity in plasma devices with E×B drift". In: *Journal of Applied Physics* 119.24 (June 2016). 243306. ISSN: 0021-8979. DOI: 10.1063/1.4954994. eprint: https://pubs.aip.org/aip/jap/article-pdf/doi/10.1063/1.4954994/13518719/243306_1_online.pdf. URL: https://doi.org/10.1063/1.4954994.

[2] Igor D. Kaganovich et al. "Physics of E×B discharges relevant to plasma propulsion and similar technologies". In: *Physics of Plasmas* 27.12 (Dec. 2020). 120601. ISSN: 1070-664X. DOI: 10.1063/5.0010135. eprint: https://pubs.aip.org/aip/pop/article-pdf/doi/10.1063/5.0010135/13784006/120601_1_online.pdf. URL: https://doi.org/10.1063/5.0010135.

[3] Nikos Sioulas, Heinz Isliker, and Loukas Vlahos. "Particle heating and acceleration by reconnecting and nonreconnecting current sheets". In: *Astronomy &amp Astrophysics* 657 (2021), A8. DOI: 10.1051/0004-6361/202141361. URL: https://doi.org/10.1051/0004-6361/202141361.

[4] W. Yao et al. "Laboratory evidence for proton energization by collisionless shock surfing". en. In: *Nature physics* 17.10 (2021), 1177–1182. ISSN: 1745-2473. DOI: 10.1038/s41567-021-01325-w. URL: http://dx.doi.org/10.1038/s41567-021-01325-w.

[5] E. L. M. Hanson et al. "Shock Drift Acceleration of Ions in an Interplanetary Shock Observed by MMS". In: *The Astrophysical Journal Letters* 891.1 (2020), p. L26. DOI: 10.3847/2041-8213/ab7761. URL: https://dx.doi.org/10.3847/2041-8213/ab7761.

[6] Stefano Markidis et al. *Automatic Particle Trajectory Classification in Plasma Simulations*. 2020. arXiv: 2010.05348 [physics.comp-ph].

[7] Imen Chakroun et al. "Using Unsupervised Machine Learning for Plasma Etching Endpoint Detection". In: Jan. 2020, pp. 273–279. DOI: 10.5220/0008877502730279.

[8] Mayur R. Bakrania et al. "Using Dimensionality Reduction and Clustering Techniques to Classify Space Plasma Regimes". In: *Frontiers in Astronomy and Space Sciences* 7 (2020). ISSN: 2296-987X. DOI: 10.3389/fspas.2020.593516. URL: https://www.frontiersin.org/articles/10.3389/fspas.2020.593516.

[9] Giovanni Lapenta. "Particle simulations of space weather". In: *Journal of Computational Physics* 231.3 (2012). Special Issue: Computational Plasma Physics, pp. 795–821. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2011.03.035. URL: https://www.sciencedirect.com/science/article/pii/S0021999111001860.

[10] G. Livadiotis and D. J. McComas. "Electrostatic shielding in plasmas and the physical meaning of the Debye length". In: *Journal of Plasma Physics* 80.3 (2014), 341–378. DOI: 10.1017/S0022377813001335.

[11] Giovanni Lapenta. "Kinetic Plasma Simulation: Particle In Cell Method". In: (Aug. 2015). DOI: 10.13140/RG.2.1.3319.2801.

[12] J. Dargent et al. "Kinetic simulation of asymmetric magnetic reconnection with cold ions". In: *Journal of Geophysical Research: Space Physics* 122.5 (2017), pp. 5290–5306. DOI: https://doi.org/10.1002/2016JA023831. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2016JA023831. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016JA023831.

[13] Xiaoli Yan et al. "Fast plasmoid-mediated reconnection in a solar flare". en. In: *Nature communications* 13.1 (2022), p. 640. ISSN: 2041-1723. DOI: 10.1038/s41467-022-28269-w. URL: http://dx.doi.org/10.1038/s41467-022-28269-w.

[14] T. W. Speiser. "Particle trajectories in model current sheets: 1. Analytical solutions". In: *Journal of Geophysical Research (1896-1977)* 70.17 (1965), pp. 4219–4226. DOI: https://doi.org/10.1029/JZ070i017p04219. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/JZ070i017p04219. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JZ070i017p04219.

[15] Seiji Zenitani and Tsugunobu Nagai. "Particle dynamics in the electron current layer in collisionless magnetic reconnection". In: *Physics of Plasmas* 23.10 (Oct. 2016). 102102. ISSN: 1070-664X. DOI: 10.1063/1.4963008. eprint: https://pubs.aip.org/aip/pop/article-pdf/doi/10.1063/1.4963008/14025593/102102_1_online.pdf. URL: https://doi.org/10.1063/1.4963008.

[16] Charu Aggarwal, Alexander Hinneburg, and Daniel Keim. "On the Surprising Behavior of Distance Metric in High-Dimensional Space". In: *First publ. in: Database theory, ICDT 200, 8th International Conference, London, UK, January 4 - 6, 2001 / Jan Van den Bussche ... (eds.). Berlin: Springer, 2001, pp. 420-434 (=Lecture notes in computer science ; 1973)* (Feb. 2002).

[17] ttnphns (https://stats.stackexchange.com/users/3277/ttnphns). *Why does k-means clustering algorithm use only Euclidean distance metric?* Cross Validated. URL:https://stats.stackexchange.com/q/81494 (version: 2020-07-21). eprint: https://stats.stackexchange.com/q/81494. URL: https://stats.stackexchange.com/q/81494.

[18] Gang Qian et al. "Similarity between Euclidean and cosine angle distance for nearest neighbor queries". In: vol. 2. Mar. 2004, pp. 1232–1237. DOI: 10.1145/967900.968151.