NTNU
Norwegian University of
Science and Technology

From data analysis to molecular understanding

Henrik Kiær

June 11, 2020

Chemical Engineering and Biotechnology
Theoretical Chemistry

Supervisor : Enrico Riccardi, IKJ
Co-Supervisor : Titus van Erp, IKJ
Submission date : June 2020
Norwegian University of Science and Technology
Department of Chemistry

Preface

This present paper is a Master's thesis for the course TKJ4900, "Theoretical Chemistry, Master's thesis" with an emphasis on chemometrics. The thesis is an extension of the course TKJ4510, "Theoretical chemistry, Project thesis" and as such, parts of the introduction, theory and simulation will be covering the same topics. The work for the thesis was conducted during the spring of 2020 at the Department of Chemistry at the Norwegian University of Science and technology. The written work in the paper includes a software update for the visualization program, PyVisA, within the simulation library for rare events, PyRETIS. It also contains a case study of two RETIS simulations of methane diffusion within S1 hydrates, with and without water vacancies. Here, the phenomena of cage-to-cage jumps of methane have been studied using PyVisA and PyRETIS. As there is an emphasis on chemometrics, this thesis is meant as a means to classify a mass of information generated by path sampling in order to guide a user to relevant descriptions. The idea is to complement the work of a user and guide him/her in the description of the outcome through means of post-processing of simulation results. Here, the new developed features within PyVisA have been utilized in order to describe the cage-jumps of methane within the hydrate system.

I would like to thank my supervisor Enrico Riccardi and co-supervisor Titus van Erp at the Department of Chemistry. I want to thank Enrico Riccardi for allowing me to work on this thesis, and for all the guidance, humour and ideas he has provided during my last year of studying. I also want to thank Titus van Erp for his guidance and knowledge on theory and feedback on my thesis. I also want to thank Anders Lervik and Ola Årøen for their interest and help in programming and design.

Abstract

Interpreting of results and data from molecular simulation can be a daunting task. The simulations can often output terabytes of data with high dimensionality and possibly many correlated descriptors. In order to better interpret and handle the large amounts of data generated, a software update for the library for visualization and post processing of data from molecular simulation, PyVisA, was developed. The updates include methods for 1- post processing of results such as recalculation of new collective variables, 2- unsupervised learning such as clustering and dimensionality reduction of simulation data, 3- computation of the correlation matrix between descriptors, 4- interactivity, animation and storage of trajectories and 5- sorting based on Monte Carlo procedures and available trajectory files.

In this master thesis, cage diffusion within S1 hydrates has been studied through RETIS simulations of cage-to-cage jumps of methane by using the newly developed features and software. A methane hydrate is a naturally occurring clathrate structure, capable of trapping guest molecules such as CH_4 and CO_2. Hence, the methane hydrates show potential both as an energy source and a means of storage for CO_2. The hydrates where studied with and without water vacancies. From the simulation results, PyVisA was able to successfully add new descriptors to existing simulation data as well as perform post processing of the data. As such, all results where produced either directly through PyVisA, or by analyzing data files created from PyRETIS. From the results it can be shown that the area of the six-membered ring, and the volume of the cage show increased values as the methane molecules move towards the ring. This behavior was observed for the system with and without water vacancies. The system with water vacancies also report a far lower rate constant and crossing probability than the system without water vacancies. Further, for the system with vacancies, Gaussian mixture clustering provided the best result in capturing regions in the potential energy. PCA was also applied to both systems, which efficiently reduced the number of dimensions with 90% variance retained.

Sammendrag

Tolkning av resultater og data fra molekylære simuleringer kan være svært krevende. Simuleringene kan ofte produsere mange terabyte av data med mange dimensjoner av høyt korrelerte deskriptorer. For å bedre tolke og behandle denne enorme mengden med data, har en software oppdatering for biblioteket for visualisering og postprosessering av data fra molekylsimuleringer, PyVisA, blitt utviklet. Oppdateringen inneholder metoder for 1- postprosessering av resultater som beregning av nye kollektive variabler, 2- ikke-veiledet læring som grupperingsanalyser og dimensjonsreduksjon av simuleringsdata, 3- beregning av korrelasjonsmatrisen mellom deskriptorer, 4- interaktivitet, animasjon og lagring av molekylære baner og 5- sortering basert på Monte Carlo prosedyre og lagrede data-filer.

I denne masteroppgaven har burdiffusjon i S1 hydrater blitt studert gjennom RETIS simuleringer av bur til bur hopp av metan. Dette ble gjort ved å bruke den nylig utviklede software-oppdateringen og de nye implementerte metodene i PyVisA. Et metanhydrat er en klatratstruktur av naturlig forekomst som har evnen til å fange gjestemolekyler som CH_4 og CO_2. Av denne grunn har metanhydratet stort potensiale både som en energiressurs og som et lagringsmedium for CO_2. Hydratet ble studert med og uten vannledighet. Fra simuleringsresultatene har PyVisA suksessfullt beregnet og inkludert nye deskriptorer til eksisterende simuleringsdata, samt utført postprosessering av dataen. Som sådan har alle resultater blitt produsert enten direkte gjennom PyVisA, eller gjennom å analysere datafilene produsert gjennom PyRETIS. Fra resultatene kan det vises at arealet av den seksleddede ringen metanet hopper gjennom og volumet av startburet viser økte verdier idet metane hopper ut av startburet. Dette for begge systemer. Systemet med manglende vannmolekyler viste også en lavere rate og krysningssannsynlighet enn systemet uten manglende vann. Videre, for systemet med vannmangel så gav Gaussisk gruppering det beste resultatet når regionene i den potensielle energien skulle grupperes. PCA var også anvendt til begge systemer, og oppnådde en effektiv reduksjon i antall dimensjoner der 90% av variansen var beholdt.

Overview of thesis

This thesis consists of a software update of the GUI for visualization, PyVisA, under the simulation program PyRETIS. It is then followed by a case study of methane diffusion within an S1 hydrate where the newly developed methods and feature are implemented. The first section of the thesis is an introduction to the current state of hydrate based carbon capture and storage. Following the introduction, comes a theoretical background for the simulations of rare events using interface sampling. This section will also contain the theory on the statistics and machine learning techniques which have been implemented into the program. After section 2, in section 3, the methods which have been implemented into PyVisA will be displayed and their relevance to post processing of data from molecular simulations will be discussed. In this section, the settings, order parameter and collective variables used in the simulation will also be presented. In section 4, the results from the simulations will be presented and their relevance to the system will be discussed before the overall conclusion is presented in section 5. Further work regarding this study and field is also presented in the final chapter.

Contents

1 Introduction **1**

2 Theory **3**

2.1 Interface sampling . . . 3

2.2 Unsupervised learning . . . 7

2.2.1 K-means clustering . . . 7

2.2.2 Gaussian mixture model . . . 9

2.2.3 Hierarchical clustering . . . 10

2.2.4 Spectral clustering . . . 12

2.3 Principal component analysis . . . 14

2.3.1 Proportion of explained variance . . . 17

3 Methods and developments **19**

3.1 Software Development . . . 19

3.1.1 Clustering . . . 21

3.1.2 Dimensionality reduction . . . 24

3.1.3 Post-processing . . . 27

3.1.4 Correlation Matrix . . . 28

3.1.5 Interactive plots and animation . . . 29

3.2 Simulation software . . . 32

3.3 Simulation . . . 33

3.4 Order parameter and collective variables . . . 33

4 Results and discussion **36**

4.0.1 Results from RETIS simulations . . . 36

4.1 Methane hydrate without water vacancies . . . 39

4.1.1 Energy terms and stable states 39
4.1.2 Breathing of water cages 42
4.1.3 Principal component analysis 44
4.2 Methane hydrate with water vacancies 46
4.2.1 Energy terms and stable states 47
4.2.2 Breathing of water cages 50
4.2.3 Other mechanisms of cage-to-cage diffusion 52
4.2.4 Movement of water vacancy 54
4.2.5 Principal component analysis 55

5 Conclusion **59**

6 Further work **61**

References **62**

Appendix **i**

A Input files for RETIS simulation **i**
A.1 Methane hydrate without vacancies i
A.1.1 Script file ii
A.2 Methane hydrate with vacancies vi
A.2.1 Script file viii

B Various plots **xvii**

1 Introduction

Since 1811, it has been known that natural gas can be hydrated and hydrate based carbon capture (HBCC) has been a topic of interest in recent years [1]. A hydrate is crystalline accumulation of gas and water in an ice-like cage [1, 2]. The water molecules, called the host, are able to trap and store guest molecules like CH_4 or CO_2 [3]. Methane hydrates, called S1 hydrates, are naturally occurring in deep sea sediments in areas with high pressure and low temperature, called the gas-hydrate-stability-zone (GHTZ) [1, 4, 5]. These methane hydrates represent a major potential as an energy source, but also as a climate threat. It is theorized that they can release their methane as a results of global warming and that the quantity of methane rivals that of the existing reserves of coal and natural gas combined [4, 5].

The methane from the hydrates can be harvested simply by depressurizing or heating [2]. While these harvesting methods are simple, they can cause destabilization of the hydrate reservoir. This can cause water leakage below the GHTZ which can halt the harvesting process [2, 5, 6]. In order to retain the structure of the hydrate reservoir, and one of the main reasons for the potential of HBCC is that the harvested methane can be replaced by CO_2. Hydrates are efficient container for CO_2 as one volume of CO_2-hydrates can release 175 volumes of CO_2 under standard conditions. Another benefit of HBCC is that it is operated at low temperature and pressure. This causes HBCC to require less energy than standard methods of CSS as around 70-90% of the operating cost in standard three stage CSS is from energy consumption [7]. With increasing pressure to develop climate friendly solutions for energy production and solutions for circular economy, HBCC presents a safe and carbon negative method for gas production [8].

One of the ways for the gas exchange to be performed is through cage-to-cage diffusion within the hydrate [9]. The S1 hydrate unit cell is made up of 46 water molecules forming two types of cages, see Fig. 1, consisting of five- and six-membered rings. When a guest molecule jumps from a donor to an acceptor cage, it can either be performed by the guest molecule jumping through a six membered or five membered ring [2]. When jumping through a six-membered ring, the acceptor cage will be another large cage, denoted as an L6L-jump. If the jump occurs through a five-membered ring, the acceptor and donor cages can be either small, or a large, and the jump is denoted as S5S-, S5L-, L5S- and L5L-jumps. This is because of the rings orientation in the unit cell leading to a one dimensional diffusion through the six-membered rings [2, 9, 10]. Since the jumps through the five-membered rings require much more energy then the L6L jumps, they will not be considered. The jumps can occur with and without water vacancies in the rings, but the energy barrier for diffusion without vacancies is significantly higher than if there are vacancies present. [9, 10].

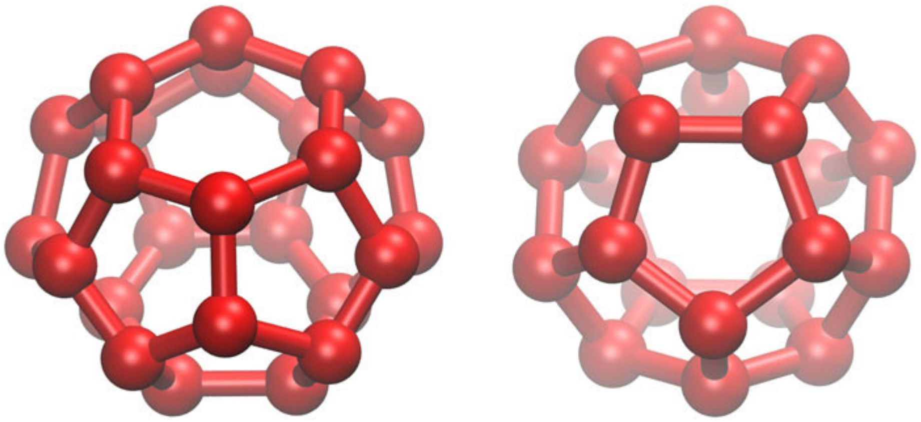


Figure 1: Illustration of the types of cages which make up an S1 hydrate. The large cage (left), and the small cage (right) forming a 6:2 ratio within the unit cell of the S1 hydrate. The figure is taken from Ref. [2].

In this work, the goal is to explore the cage-to-cage diffusion of methane in an S1 hydrate with and without water vacancies through RETIS simulations. In order to explore the diffusion, the simulations will be designed to perform the L6L jumps of the methane. Thanks to the substantial developments included in PyVisA, we perform data exploration and visualization to describe the physical process.

2 Theory

This section will focus on the theory for the techniques used in the simulations and the features that have been added to PyVisA. First the theory on interface sampling of rare event simulations, and then the unsupervised learning methods will be presented.

2.1 Interface sampling

Simulations based on molecular dynamics (MD) try to solve Newtons equations of motion for a system of molecules. MD methods are used numerous fields of research, from material science to biology and theoretical chemistry [11]. While MD is flexible, it suffers from needing a time step of between 0.5 and 2 femto seconds. This makes it hard to simulate reactions occurring at micro second time scales. [12]. Many methods, trying to overcome this simulation barrier to sample rare events, will either alter the potential energy surface and/or the dynamics of the reaction [12, 13, 14]. This causes a disturbance of the chemical phenomenon being studied [12]. One way to simulate rare events without disturbing the dynamics of the system is to sample unbiased MD trajectories by utilizing Monte Carlo schemes in path space [11]. Among the methods that sample trajectories, transition interface sampling, (TIS), and replica exchange TIS, (RETIS), have increased the efficiency of rare events simulation while still sampling the unaltered dynamics of the system [15, 16, 17].

RETIS, which is a TIS development, is a simulation method which samples MD trajectories in the phase space of the reaction by an MC procedure. In RETIS and TIS, the progress of a reaction is defined by an order parameter, λ (OP), which is a numerical descriptor that aims to capture the phenomenon. The OP will discriminate the reaction into reactant state A, $\lambda \leq \lambda_A$, and product state B, $\lambda \geq \lambda_B$ [11]. All other descriptors in the simulation are here called collective variables (CV).

TIS/RETIS aims to calculate the rate constant, k_{AB}, of a reaction, given as [11]:

$$k_{AB} = f_A P_A(\lambda_A|\lambda_B), \tag{1}$$

where f_A is the initial flux of trajectories that pass through state A, and $P_A(\lambda_B|\lambda_A)$ is the probability to reach the reactant state B, given that the trajectory started from state A [11, 18].

In most rare events, crossing the potential energy barrier of the reaction is not possible by means of a standard MD simulation. However, by defining path ensembles, which define regions between the product and reactant state, the TIS/RETIS algorithm is able to efficiently sample rare events [5, 18]. Each path ensemble is labeled as $[i^+]$ and has an interface λ_i of the order parameter. The first interface λ_0 is placed so that the first path ensemble $[0^+]$ defines the reactant state A. The following interfaces are then placed so that $\lambda_2 \geq \lambda_1 \geq \lambda_0$ until λ_N is reached which defines the product state B. This implies that the crossing probability can be expressed as the product of conditional crossing probabilities. The rate constant for the reaction is then calculated as [11]:

$$k_{AB} = f_A \prod_{i=0}^{N-1} P_A(\lambda_{i+1}|\lambda_i), \tag{2}$$

where instead of the probability of crossing into the product state from the reactant state from Eq. (1), the history dependent conditional crossing probability $P_A(\lambda_{i+1}|\lambda_i)$, is used. This is the probability that a path crosses the interface λ_{i+1} given that the path had its origin in λ_A, ended in either λ_A or λ_B and had at least one crossing of the interface λ_i [11]. An example of the path ensemble interfaces with molecular trajectories is shown in Fig. 2.

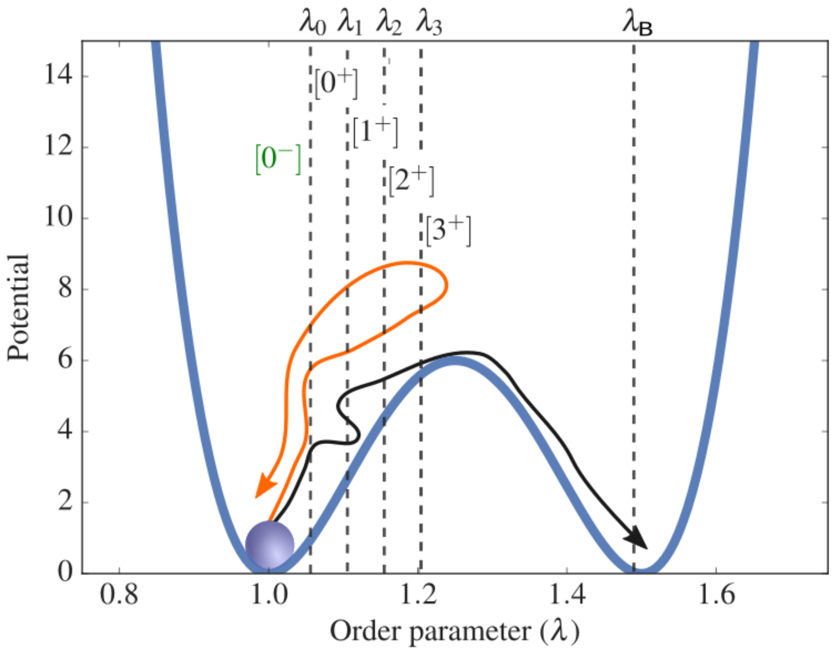


Figure 2: Illustration of the RETIS path ensemble interfaces for a simple 2D well potential with five path ensembles and two trajectories, one reactive (black) and one non-reactive (orange). The figure is taken from the PyRETIS website at Ref. [18].

One difference between TIS and RETIS is the calculation of the initial flux, f_A. In TIS, the initial flux is calculated by a MD simulation given as [15]:

$$f_A = \frac{N_c^+}{T_{\in A}}, \tag{3}$$

where N_c^+ is the number of positive crossings with interface $\lambda_A = \lambda_0$ and $T_{\in A}$ is the time spent in state A. RETIS, which obtains its results purely from path sampling simulations, avoids the need to calculate the flux through a MD simulation. This is done by introducing another path ensemble $[0^-]$ which is placed behind the first path ensemble. Ensemble $[0^-]$ contains all trajectories which start at λ_A, explore the reactant state and end at λ_A again. This allows the RETIS algorithm to calculate the flux through the average path lengths of the paths contained within the ensembles $[0^-]$ and $[0^+]$, $\langle t_{path}^{[0^-]}\rangle$ and $\langle t_{path}^{[0^+]}\rangle$ respectively. The calculation of the flux, now becomes [15]:

$$f_a = \frac{1}{\langle t_{path}^{[0^-]}\rangle + \langle t_{path}^{[0^+]}\rangle} \tag{4}$$

The most important of the path sampling methods, is the shooting move. The shooting move consists of taking a random time step from the last accepted path, altering the momentum at this point, and creating a new path by integrating forward and backwards in time [15]. In order to sample the correct distribution of paths there is a requirement for the trajectories to obey detailed balance [19, 20]. This implies that at equilibrium, the transitions between two states occur at the same rate. Detailed balance can be expressed as [15]:

$$\frac{P_{\text{gen}}[\boldsymbol{x}^{(o)} \to \boldsymbol{x}^{(n)}]}{P_{\text{gen}}[\boldsymbol{x}^{(n)} \to \boldsymbol{x}^{(o)}]}\frac{P_{\text{acc}}[\boldsymbol{x}^{(o)} \to \boldsymbol{x}^{(n)}]}{P_{\text{acc}}[\boldsymbol{x}^{(n)} \to \boldsymbol{x}^{(o)}]} = \frac{P[\boldsymbol{x}^{(n)}]}{P[\boldsymbol{x}^{(o)}]}, \tag{5}$$

where $P_{\text{gen}}[\boldsymbol{x}^{(o)} \to \boldsymbol{x}^{(n)}]$, and $P_{\text{gen}}[\boldsymbol{x}^{(n)} \to \boldsymbol{x}^{(o)}]$ are the probabilities to generate a new path (n) from an old path (o), and from the new path, generate the old path. $P_{\text{acc}}[\boldsymbol{x}^{(o)} \to \boldsymbol{x}^{(n)}]$, and $P_{\text{acc}}[\boldsymbol{x}^{(n)} \to \boldsymbol{x}^{(o)}]$ are the probabilities to accept paths from state o to state n and from state n to state o. Lastly $P[\boldsymbol{x}^{(o)}]$ and $P[\boldsymbol{x}^{(n)}]$ are the probabilities for the old and the new path.

The MC moves also need an acceptance criterion. When generating a new path, each time step of the last accepted trajectory will have the same probability to be chosen as the "shooting point" and the velocities are regenerated according to a Boltzmann distribution. From the shooting point, the path is generated through integration forwards and backwards in time until a stable state is reached. The acceptance criterion for the newly generated path can then be written as [15]:

$$P_{\text{acc}}[\boldsymbol{x}^{(n)} \to \boldsymbol{x}^{(o)}] = \min\Big[1, \frac{L^{(o)}}{L^{(n)}}\Big], \tag{6}$$

where $L^{(o)}$ and $L^{(n)}$ are the lengths of the old and the new path. This means that longer paths are unfavorable and are likely to be rejected [15]. Both TIS and RETIS also use the time reversal move, which as the name implies, changes the time direction of a path. This is cheaper MC move as it does not require any MD steps, but will produce highly correlated paths [11]. Another difference between

TIS and RETIS is that RETIS also utilizes the swapping move. This MC move consists of swapping two paths if they are valid for each others path ensembles. This move acts between different path ensembles, and will increase the amount of accepted trajectories and reduce the correlation between successive trajectories within the same path ensemble [16].

2.2 Unsupervised learning

Unsupervised learning is part of an exploratory data analysis where the scientist is not interested in prediction, as there is no response variable. Instead, the goal is to discover patterns, subgroups and other behaviour in the data.

2.2.1 K-means clustering

K-means clustering is a technique for dividing and categorizing data into classes or clusters and is widely used within data mining and machine learning. The generic algorithm aims to divide a set of n observations into k distinct clusters [21]. If C_1,...,C_K are sets which denote the indices of all the observations in each of the K cluster, then they need to satisfy two properties [21]:

- $C_1 \cup \mathrm{C}_2 \cup ... \cup \mathrm{C}_K = \{1,....\mathrm{n}\}$, meaning that every observations must belong to one of the K clusters.
- $C_k \cap \mathrm{C}_k^{'} = \emptyset$, meaning that the clusters do not overlap, and that no observation can belong to more than one cluster.

The K-means algorithm aims to minimize the within-cluster variation (WCV), $W(C_k)$ which is a measure of how much the observations within the same cluster differ from each other [21]. The most common way to define the within-cluster variation is through squared euclidean distance and is defined as:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \tag{7}$$

where $|C_k|$ denotes the number of observation and x_{ij}, and $x_{i'j}$ are the points within the kth cluster,.

The standard algorithm, often called naive K-means, moves between two steps for n observations, p features and K clusters [21]:

1. Randomly assign an initial placement for the K clusters and assign the points to their closest cluster center.
2. Iterate until no observations change their cluster assignment:
    (a) For each cluster, compute the centroid where the centroid is the vector of the p feature means for all observations within each cluster.
    (b) Assign each observation to its closest centroid, defined by the euclidean distance.

An illustration of algorithm 2.2.1 is shown in Fig. 3.

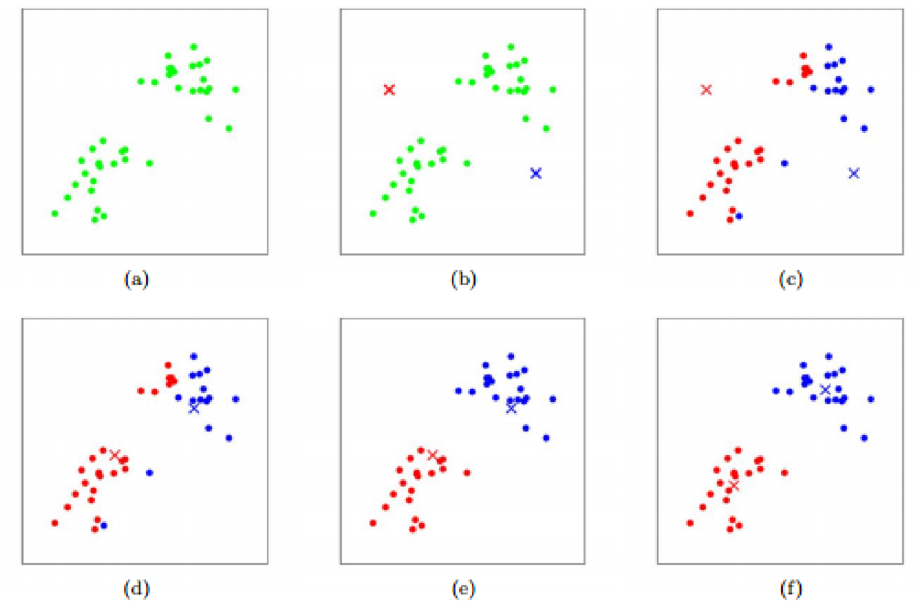


Figure 3: Illustration of the k-means algorithm from Stanford [22], where: in (b) two cluster centers are randomly placed for the initial observations; in (c) the observations are assigned to the nearest clusters before the centers of mass for the clusters are updated in (d); new assignments are done in (e) before the final adjustments of cluster centers are performed in (f).

Although the naive algorithm is guaranteed to decrease the WCV, it is not known if the clustering finds a local or a global optimum since the initial assignment of clusters is random. Hence it is recommended to run the algorithm several times in order to compare the WCV between the different runs to find the global minimum [21].

2.2.2 Gaussian mixture model

Gaussian mixture model (GMM) is a technique which tries to cluster data similarly to k-means, but utilizes the variance of the data as well. A Gaussian mixture model $\rho(\mathbf{x}|\mathbf{\Theta})$ is a weighed sum of M > 1 cluster or components $\rho(\mathbf{x}|\boldsymbol{\theta}_m)$. Each component in the model is expressed as a normal distribution of the form [23]:

$$\rho(\mathbf{x}|\mathbf{\Theta}) = \sum_{m=1}^{M} \alpha_m \rho(\mathbf{x}|\boldsymbol{\theta_m}), \tag{8}$$

where $\mathbf{x} = [x_1, x_2, ..., x_d]^T$ is the d-dimensional data vector, $\boldsymbol{\theta}_m = \{\boldsymbol{\mu}_m, \, \mathbf{\Sigma}_m\}$, is the mean vector and the covariance matrix and α_m is the weight of component $m = 1, ..., M$. The weights are also strictly positive and $\sum_{m=1}^{M} \alpha_m = 1$. $\mathbf{\Theta}$, called the Gaussian mixture is the set of parameters $\{\alpha_1, ..., \alpha_M, \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M\}$ and denotes the shape of the Gaussian distributions for the components [23].

The general algorithm for GMM consists of an expectation step, and a maximization step, called the EM-iteration, that are performed consecutively until the log-likelihood [23]:

$$\log \rho(\chi|\mathbf{\Theta}) = \log \prod_{i=1}^{N} \rho(x^i|\boldsymbol{\theta}_m), \tag{9}$$

converges to a local optimum where $\chi = \{x^1, x^2, ..., x^N\}$ are N independent and identically distributed samples and $\mathbf{\Theta}$ is the Gaussian mixture.

In the expectation step, the data χ is assumed incomplete while the complete data set, $\gamma = (\chi, Z)$, is determined by estimating the set of variables contained within $Z = \{\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_M}\}$, and each $\boldsymbol{z_m}$ is an N-dimensional vector $[z_m^1, z_m^2, ..., z_m^N]$. The log-likelihood of the complete data is then [23]:

$$\log \rho(\gamma|\mathbf{\Theta}) = \sum_{i=1}^{N} \sum_{m=1}^{M} z_m^i \log\left[\alpha_m \rho(\boldsymbol{x^i}|\boldsymbol{\theta}_m)\right], \tag{10}$$

where z_m^i is equal to:

$$z_m^i = P(m|\boldsymbol{x^i}, \boldsymbol{\Theta^t}), \tag{11}$$

which is the posterior probability, and $\boldsymbol{\Theta^t}$ is the estimation of the Gaussian parameters after t iterations of the EM-iteration. This corresponds the probability that sample i, belongs to cluster m [23, 24].

In the maximization step, the parameters of the Gaussian mixture, $\boldsymbol{\Theta^{t+1}}$ are updated through the estimate of the variables z_m^i. In the model, this corresponds to updating the variables α_m^{t+1}, $\boldsymbol{\mu_m^{t+1}}$ and $\boldsymbol{\Sigma_m^{t+1}}$, which are the weight, the expectation values, and the correlation matrix of the normal distribution, given by [23]:

$$\alpha_m^{t+1} = \frac{1}{N}\sum_{i=1}^{N} z_m^i, \tag{12}$$

$$\boldsymbol{\mu_m^{t+1}} = \frac{\sum_{i=1}^{N} z_i \boldsymbol{x_i}}{\sum_{i=1}^{N} z_m^i}, \tag{13}$$

and

$$\boldsymbol{\Sigma_m^{t+1}} = \frac{\sum_{i=1}^{N} z_m^i (\boldsymbol{x_i} - \boldsymbol{\mu_m^{t+1}})(\boldsymbol{x_i} - \boldsymbol{\mu_m^{t+1}})^T}{\sum_{i=1}^{N} z_m^i} \tag{14}$$

By repeating the EM-iteration, the likelihood of all the points belonging to all the clusters is computed, and from these results, the parameters of all the clusters are updated [23, 24]. Through these iterations, the points will eventually be assigned to the cluster they most likely to belong to and the results can be visualized [25].

2.2.3 Hierarchical clustering

Hierarchical clustering begins by treating each observation as its own cluster, before merging observations and creating larger and larger clusters. This merging can be done until a single cluster containing all the observations are left [26] and the user can chose the amount of clusters to use, or until a predefined number of clusters is reached. The method begins by calculating the proximity matrix for all points, before merging the two closest observations to a new cluster, and re-computing the proximity matrix [27]. This style of hierarchical clustering is called agglomerative or bottom-up clustering as it starts with the leaves and combines them until it reaches the root [21].

In order to produce the proximity matrix, a dissimilarity measure between two points, and a metric for measuring the similarity between clusters, often called a linkage criterion, must be chosen. There are several methods for calculating the distance between two points, such as the Euclidean distance defined as [21]:

$$||x_i - x_j|| = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}, \tag{15}$$

where x_i and x_j are two dimensional observations which contain the values (a, b). An illustration of the agglomerative algorithm is shown is Fig. 4 where eight points are being clustered.

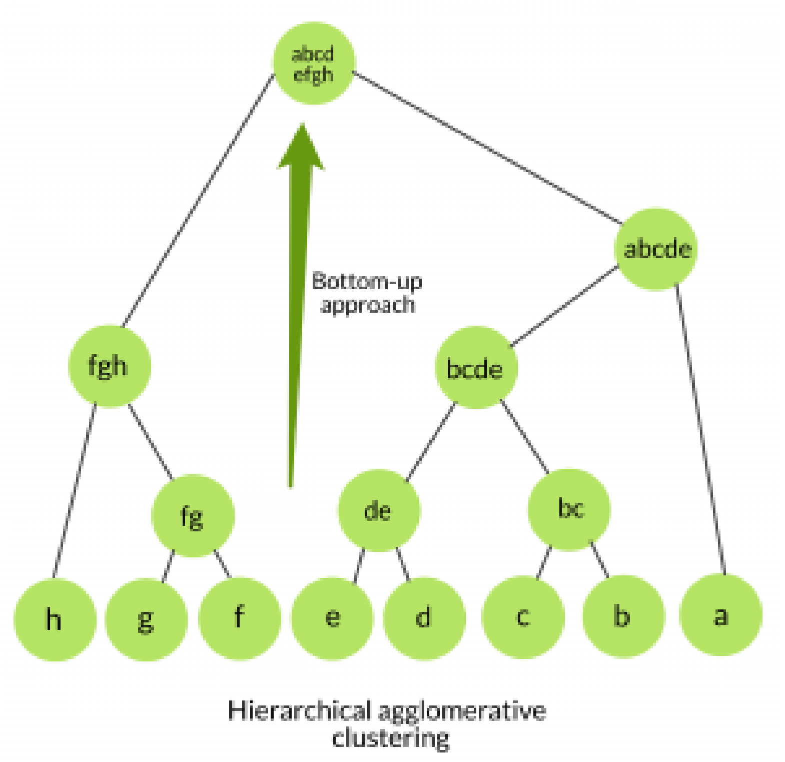


Figure 4: Illustration of the hierarchical agglomerative clustering algorithm taken from the computer science portal GeeksforGeeks at Ref. [28].

For the linkage criterion, the complete, single, average and centroid linkage is often used. The complete and single linkage is the maximum and minimum distance between any two points in two different clusters. The average linkage is the average distance between all points in two clusters, and the centroid linkage is the dissimilarity between the centroids. This is a mean vector of length p, of two clusters [21].

The general algorithm for agglomerative hierarchical clustering is defined as follows [21]:

1. Define a dissimilarity measure for all the $\binom{n}{2}$ = n(n-1)/2 pairwise dissimilarities. Each observation is treated as its own cluster.
2. for i = n, n-1 ,...., 2 do:
   (a) Identify the pair of clusters that are the least dissimilar, and fuse them. This becomes a new cluster.
   (b) Compute the new pairwise dissimilarities between all i-1 cluster that remain.
3. The merging is repeated until the desired amount of clusters remain.

2.2.4 Spectral clustering

Spectral clustering is an emerging method for data exploration that can often outperform traditional techniques like k-means while still being implementable with simple algebra by treating the clustering as a graph partitioning problem [29]. In any clustering scheme, a measure of similarity or dissimilarity, s_{ij} between points x_i and x_j is often needed in order to determine assignments to clusters. But if no more information between the observations other than similarity is known, a way to express the data is through a similarity graph $G = (V, E)$. Where V is the set of vertices, $\{v_1, v_2, ..., v_n\}$, in a dataset of n observation where each vertex v_i represents one observation x_i and E is the set of edges connecting the vertices. Two vertices v_i and v_j will have an edge, e_{ij}, if the similarity s_{ij} is positive, or larger than some threshold where the edge is weighted by s_{ij} [29]. The clustering problem now becomes a matter of partitioning the similarity graph such that the vertices within one cluster are as similar as possible. The edges within a cluster will then be highly weighted and the edges between groups of vertices will have low weights as they are as dissimilar as possible [29].

In spectral clustering there are many ways to transform a dataset of n observations with pairwise similarities into a similarity graph. What all these methods have in common is that they aim to produce undirected graphs. In graph theory an undirected graph G, is a graph where for every edge e_{ij} connecting the vertex v_i to v_j, there is also an edge e_{ji} connecting vertex v_j to v_i, meaning that all edges are symmetric. See Fig. 5 where an undirected and a directed graph is shown. If G is assumed to be weighted, then all edges also carry a non-negative weight w_{ij}, where $w_{ij} = w_{ji}$. If $w_{ij} = 0$, there would not be an edge between the vertices v_i and v_j. The weights of a graph is defined within the adjacency matrix of the graph, W, and by summing the weights on the edges from a vertex, v_i, one can also find the degree, of said vertex, d_i defined by [29]:

$$d_i = \sum_{N}^{j=1} w_{ij}, \tag{16}$$

where d_i defines how many edges a vertex, v_i, has.

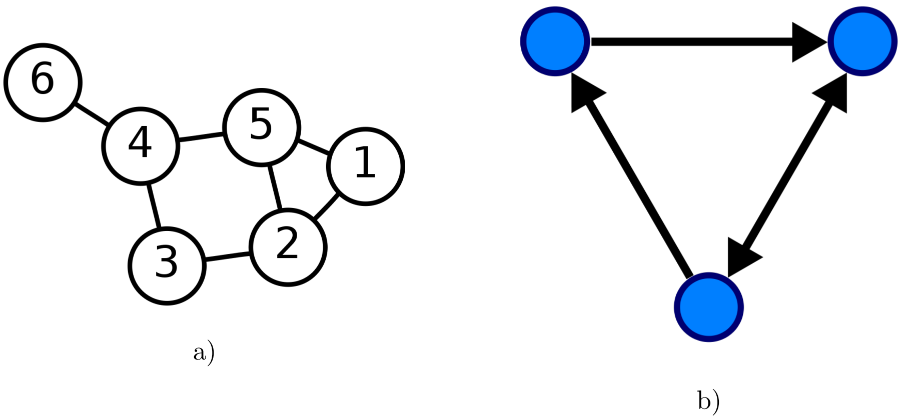


Figure 5: Illustration of two graphs. Sub-image a) shows an undirected graph consisting of six vertices and seven edges, and sub-image b) shows a directed graph consisting of three vertices and four edges. The image is taken from Ref. [30].

One of the main types of similarity graphs used in spectral clustering is the k-nearest neighbour. In the k-nearest neighbour graph the vertex v_i is connected to the k nearest other vertices, hence the name nearest neighbours. In order to ensure the resulting graph is undirected, two measures can be made as the neighbourhoods of the vertices are not symmetrical. The first method is to neglect the direction of edges, where if vertex v_i is in the neighbourhood of v_j , then they are both connected and similarly if v_j was in the neighbourhood of v_i. This creates what is called *k-nearest neighbour graph.* The second method is to create an symmetrical edge e_{ij} between vertices v_i and v_j if they are both in each others neighbourhood. This results in the *mutual neighbourhood graph.*[29]

After the similarity graph has been created, the graph Laplacian is computed, which is defined as [29]:

$$L = D - W, \tag{17}$$

where D is the degree matrix, which is a diagonal matrix that contains all the degrees, $d_1, d_2, ..., d_n$ on the diagonal given by Eq. (16), and W is the adjacency matrix containing all the weights. The benefit of using the Laplacian is that it will have the degrees for all vertices on its diagonal, and the negative weights for the edges of the vertices on the off-diagonal entries [29, 31].

Then, the first k eigenvectors, $u_1, u_2, ..., u_k$, of L are computed where k is the amount of clusters. The eigenvectors will contain information on how to segment the nodes in the graph and on the basis of these eigenvector, k-means clustering is performed to assign the points to the clusters [31].

2.3 Principal component analysis

Principal component analysis (PCA) is a widely used technique for unsupervised learning, dimensionality reduction and data visualization. The core idea, is that with n observations in p dimensions, all p dimensions are not necessarily of equal importance. PCA will therefore try to find a low-dimensional representation of the data while trying to capture as much of the variance in the original data as possible. [21]

The different principal components (PC's) are found through linear combinations of the p variables in the original system [21]. Given a set of features $X_1, X_2, ..., X_p$, the first PC is given as the normalized combination of the features:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p, \tag{18}$$

which have the largest variance and where ϕ are called the loadings and $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ [21].

These loadings make up the first principal components loading vector $\phi_1 = (\phi_{11}, ..., \phi_{p1})^T$ and is a measure of how much of the original variable is represented in the PC. Therefore it is important that they are normalized in order to give equal importance to the original variables in regards to the size of their variance. Then the first PC, for a dataset $\mathbf{X}$ of size nxp, is computed by finding the linear combination of loadings and sample feature values with the largest sample variance of the form [21]:

$$z_{i1} = \phi_{11}x_{11} + \phi_{21}x_{12} + ... + \phi_{p1}x_{1p}, \tag{19}$$

with a constraint that the loadings are normalized and assuming that the variables in $\mathbf{X}$ have been centered so as to have mean zero. This is assumed because the only points of interest is the variance. This in turn, is a maximization problem defined by [21]:

$$\underset{\phi_{11},\dots,\phi_{p1}}{\text{maximize}}\Big\{\frac{1}{n}\sum_{i=1}^{n}\Big(\sum_{j=i}^{p}\phi_{j1}x_{ij}\Big)^2\Big\} \text{subject to} \sum_{j=1}^{p}\phi_{j1}^2 = 1. \tag{20}$$

The problem, can also be written as:

$$\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2, \tag{21}$$

where $z_{11}, ..., z_{n1}$ are called the scores of the first principal component and have an average of zero, as they are a linear combination of the variables of $\mathbf{X}$ which have been centered to have a zero mean.

The interpretations of the scores and loadings, can be thought of as the following. The loadings define the direction of the vector in the feature space with the most variance. These directional values are then projected onto the n observations in the dataset giving the scores of the principal component, such that $z_{11} = \phi_{11}x_{11}$ etc. Then in order to find the second PC, Z_2, one has to find the linear combination of the variables that has maximum variance and is uncorrelated, and therefore orthogonal to Z_1 [21]. This process is illustrated in Fig. 6.

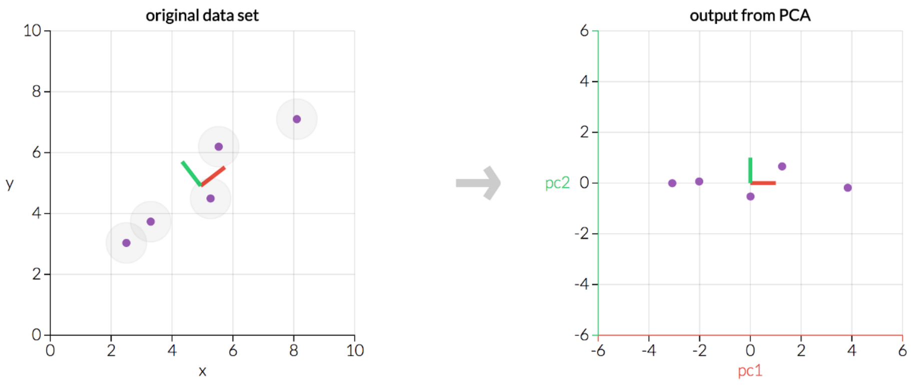


Figure 6: Illustration of PCA being performed on a dataset. The red and green line defines the directions of largest and second largest variance in the dataset. The second image shows the first two principal components plotted against each other, which have been normalized to a mean zero. The image is taken from Ref. [32].

Similarly to the first PC, the scores of the second PC, $z_{12}, z_{22}, ..., z_{n2}$, take the form of:

$$z_{i2} = \phi_{12}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p2}x_{ip}, \tag{22}$$

where $\phi_{12}, ..., \phi_{p2}$ are the loadings of the second PC which make up the second PC loading vector ϕ_2. Then, the maximization problem similar to Eq. (20) must be solved, but for the variables corresponding to the second PC, Z_2. Then for the third PC, one finds the linear combination of the variables giving the third largest amount of variance, while being uncorrelated to Z_1 and Z_2, and the same for the rest of the PC's [21].

Another useful property of PCA is that the first PC provides the best line and the first two PC's provide the best plane in a p-dimensional space that is closest to the n observations in terms of average squared euclidean distance [21]. This property continues with increasing number of PC's and dimensions in the system. This implies that the first M principal component score and loading vectors provide the best approximation in M dimensions to the ith observation,

x_{ij}, in terms of euclidean distance [21]. This can be written as:

$$x_{ij} \sum_{m=1}^{M} z_{im}\phi_{jm}. \tag{23}$$

Assuming the original data matrix is column centered, the principal components score and loading vectors can, given a large enough dataset, give a good approximation of the data. [21]

2.3.1 Proportion of explained variance

When performing PCA, the goal is to have as few PC's as possible, while retaining as much variance as possible. If, in a p-dimensional dataset, one creates p PC's, there is no dimensionality reduction, and the dataset is simply re-created. There is also the risk of choosing to few PC's and not retaining enough variance to be able to capture anything of interest in the data. The proportion of variance explained (PVE) by each component is the amount of variance that PC is able to capture [21]. By assuming the variables in a dataset has mean zero, the total variance can be explained as [21]:

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2, \tag{24}$$

and the variance explained by the mth PC is [21]:

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \Big(\sum_{j=1}^{p} \phi_{jm} x_{ij} \Big)^2. \tag{25}$$

Hence, the PVE corresponding to the mth PC is given by [21]:

$$\mathrm{PVE}(Z_m) = \frac{\frac{1}{n} \sum_{i=1}^{n} \Big(\sum_{j=1}^{p} \phi_{jm} x_{ij} \Big)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}. \tag{26}$$

In order to assess the amount of PC's necessary in an analysis, the cumulative explained variance (CEV), which is the sum of the PVE of the PC's is plotted against the number of PC's. A plot of the CEV is shown in figure 7. By visual inspection of the CEV plot, one can find the amount of PC's to use. As a rule of thumb one should choose the smallest amount of PC's that is needed in order to explain a sizeable amount of variance, often above 80% [21]. Another method is to look for the elbow in the CEV plot, which is where the increase in explained variance per principal component decreases. In Fig. 7 the shoulder appears to be around 15 principal components. This is because after about 15 principal components, the model retains about 85% variance, and slope of the graph rapidly decreases, meaning that the model becomes more complex while the increase in variance retained is only slightly increasing.

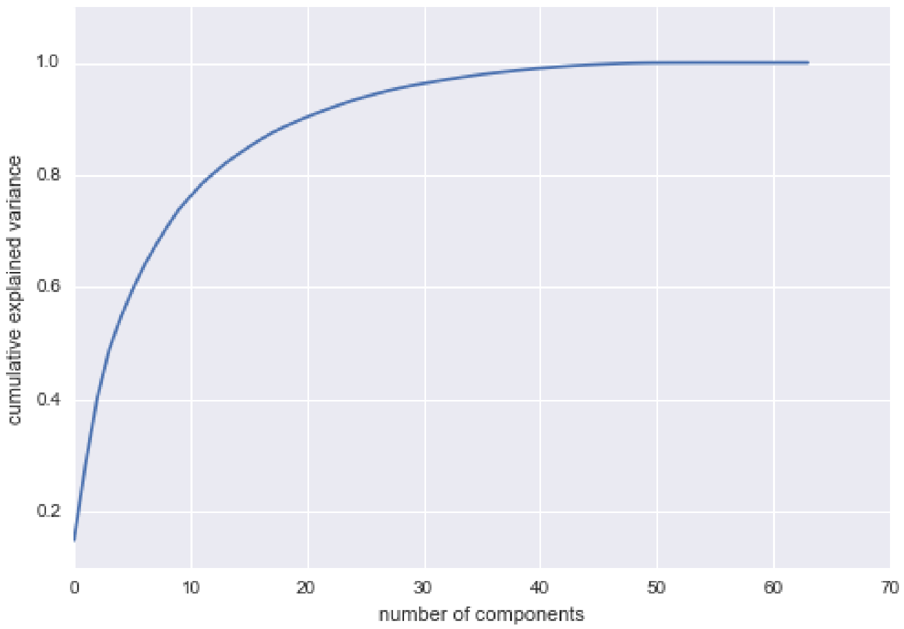
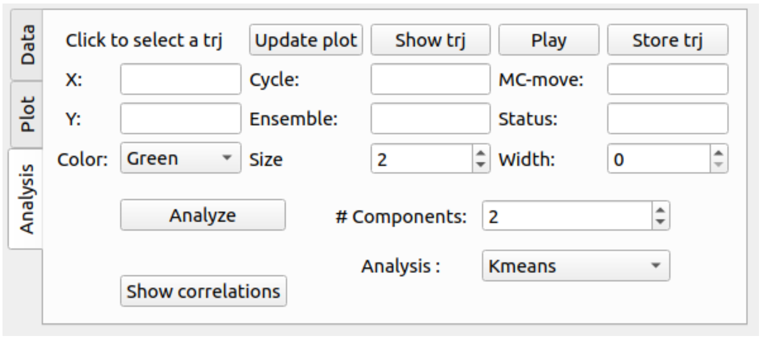


Figure 7: Plot of cumulative explained variance with a shoulder at around 15 components. The figure is taken from Ref. [33].

3 Methods and developments

3.1 Software Development

In this section the software developments that have been added to PyVisA will be presented. Their purpose and benefit towards guiding a user in data exploration and analysis will also be discussed. The methods have primarily been added to a new page of the GUI, named Analysis, which contains clustering, PCA, calculation of the correlation matrix and the options for interactivity, storage and animation, see Fig. 8. The program has also been extended to load singular trajectories and configurations, interactively select and visualize whole trajectories on the plots as well as animate and store said trajectories. Further, the option to perform post-processing on the trajectory files in order to add more collective variables to the simulation has also been implemented.

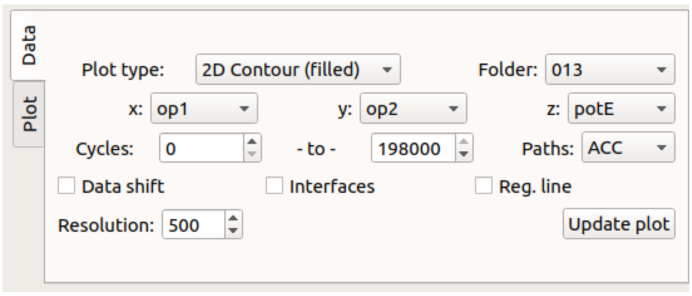


X: | Cycle: | MC-move:
Y: | Ensemble: | Status:
Color: Green | Size 2 | Width: 0
Components: 2
Analysis : Kmeans

Figure 8: Analysis tab, showing the different options for interactivity, animation, clustering and correlation matrix. The Analysis drop-down menu consists of the option: K-means-, Hierarchical-, Gaussian mixture-, spectral clustering and PCA.

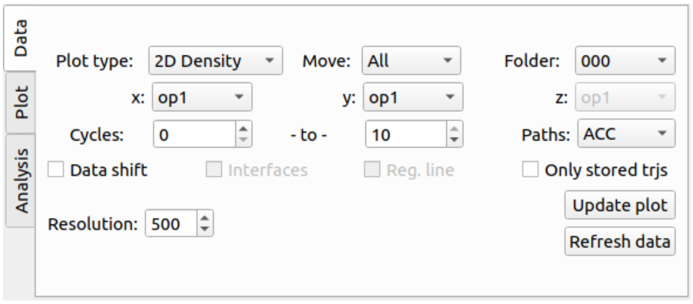
The data collection of the program has also been updated and compressed by the introduction of a trajectory class which contains all the features of a PyRETIS trajectory. The information and features of the trajectories have been stored into an object by utilizing the pandas dataframe to store large amounts of data [34]. PyVisA can also perform a new collection of the data, which is useful if the user is visualizing an ongoing simulation. This option is available through the ”Refresh data” button on the Data-tab, see Fig. 9 b).

With the introduction of the trajectory class, the ability to sort the simulation data has also been improved by 1- the option to sort trajectories based on their MC-generation move and 2- the option to only show the data from the stored trajectory files. This a useful feature in the analysis as the trajectory files are the ones that can be used for recalculations of new collective variables. These options for selection and sorting, are available in the Data-tab of PyVisA, which is shown is Fig. 9, where the old and the new Data-tab is displayed.



Plot type: 2D Contour (filled)
Folder: 013
x: op1
y: op2
z: potE
Cycles: 0 - to - 198000
Paths: ACC
Resolution: 500

a)



Plot type: 2D Density
Move: All
Folder: 000
x: op1
y: op1
z: op1
Cycles: 0 - to - 10
Paths: ACC
Resolution: 500

b)

Figure 9: The old, a), and the updated, b), Data-tab in PyVisA. The new tab has been fitted with option to sort based on MC generation moves, available trajectory files, and can perform a refresh of the data to visualize the development of an ongoing simulation. Sub-image a) is taken from the pyretis website at Ref. [35].

In order for the user to be able to load data from specific folder, or singular trajectories, the commands for the program has also been updated. This makes PyVisA able to test behaviour and check for correctness with a smaller sample than by loading the entire simulation. This makes the program flexible in loading data, either from the command window, or through the "load data" option in the File-menu. This improved ability to sort the data also aids in the search for outliers. This is because the user has an improved chance of pinpointing the outliers with a smaller subset of the data, as well as being able to get the information about the outlier through the interactivity of the plots.

3.1.1 Clustering

The following methods of clustering have been added as features of analysis in PyVisA: k-means, agglomerate hierarchical with average linkage, Gaussian mixture model and spectral clustering with k-nearest neighbourhood with 30 neighbours. In order try to generate stable clusters in the hierarchical method, the average linkage criterion was chosen as single and centroid linkage tends to yield uneven or distorted clusters [21, 27].

There are many implications in clustering that will greatly affect the outcome of the analysis. The amount of clusters will be unknown, and there is no consensus on a single best approach to clustering and data analysis. K-means clustering is often the standard approach for many when performing post-processing. However it has its cons, and it will often be useful to be able to perform clustering with different algorithms. One possible pitfall of k-means and hierarchical clustering is that every observation will be assigned to a cluster, which also means that noise and outliers will be included. This can cause the clusters to be distorted since it now contains outliers which are likely to not belonging to any clusters [21]. K-means also utilize the euclidean distance measure, and is therefore best suited to handle circular clusters. If there are clusters of other shapes, it is likely that the k-means algorithm will assign observations to the wrong cluster, as the cluster boundaries are overlapping, as showing in Fig. 10. In order to cluster data from different shapes of clusters, algorithms like GMM or spectral clustering can be more flexible.

The benefit of using algorithms like GMM is that it performs soft clustering as GMM also utilizes the variance of the data, whereas k-means and hierarchical performs hard clustering [24]. In hard clustering, each observation is assigned to a cluster which can be right or wrong. But in soft clustering, the observations are assigned to the clusters with a certain probability of belonging to that cluster [36]. Then, if k-means provides staggered results, and its suspected that there is a lot of noise in the data set, GMM clustering might be a more suitable algorithm to utilize.

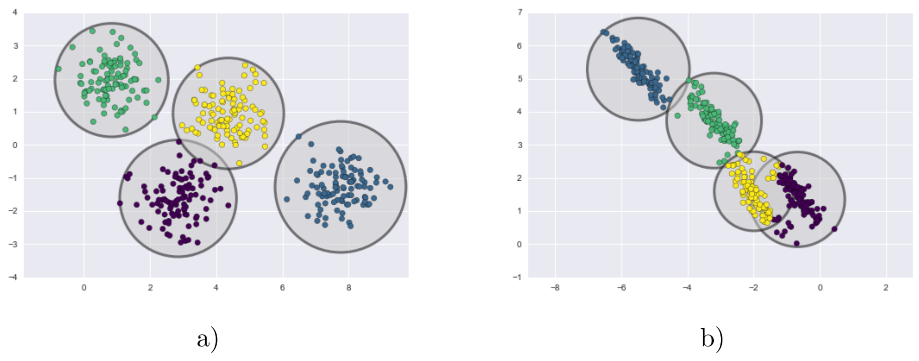
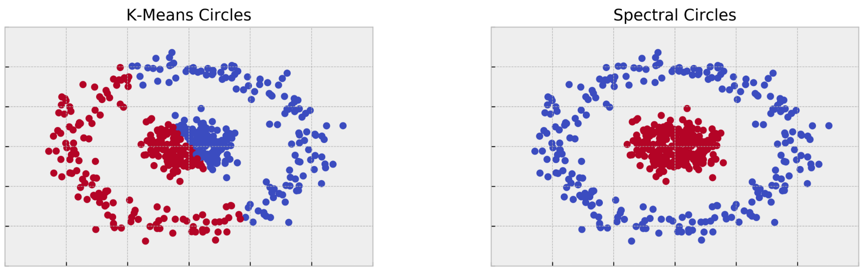


Figure 10: Illustration of two types of clusters, one circular, a, and one elliptical, b. K-means clustering can often struggle with clusters with elliptical shape as in sub figure b. The figure is taken from Ref. [24].

Spectral clustering is also more flexible than k-means as the algorithm assumes little of the shapes of the clusters. Spectral clustering can easily cluster data forming intertwined spirals and can handle large datasets, assuming that the Laplacian is sparse, which is ensured by using the k-nearest neighbourhood similarity graph. This is shown in Fig. 11 where both spectral and k-means clustering have been performed for circular dataset. While the risk of getting stuck in a local optimum is low, the method is generally unstable for different values of the parameters involved in generating the similarity graphs. As such, spectral density is not a good choice for a black box method [29, 31]. Clusters of elliptical or possibly spiral shape could occur in simulations for example if one measure dihedral angles [20]. Therefore it is beneficial to be able to use spectral clustering in combination with other methods, as a means of specialization after initial inspection of a dataset.



a) b)

Figure 11: Illustration of k-means and spectral clustering on a circular dataset. The figure is taken from Ref. [31].

Clustering methods are also sensitive to perturbations of the data, meaning that by removing a random subset of the data, the results from clustering will be different compared to results generated by using all the data [21]. Since clustering can give non-robust results, it is important to perform clustering on smaller subsets of the data with varying amount of clusters. This will also apply when analyzing results from path sampling, as there are possibilities that one path will be repeated which occurs if the last accepted path is copied as the new path. The choice of visualizing data based on the different MC generation moves can reduce the amount of repeated data by choosing to only visualized the trajectories initialized through a shooting move. This can also help optimize the different percentage of moves for the simulation. The ability to cluster smaller subsets is also improved with the new features for loading data within PyVisA.

Clustering techniques can also be useful in the detection of energy minimums. As changes in potential energy can indicate the stability of a system, there might be areas that contain a high density of the simulations trajectories. In order to visualize these areas, either density plots or clustering techniques can be used. These stable states and energy minimums can be of highly irregular shapes. This is because they can be a product of many different descriptors for the system. Therefore the algorithms like GMM and spectral clustering might be best suited to handle such clusters. The clustering algorithms can also serve as a validation for the density plots, improving the users ability to discern different states of the system. Clustering methods also help in analyzing outlier of the simulation as k-means will react strongly to their presence.

As there is no best method of clustering defined, it is important to have a range of different methods available. This allows the user to easily and readily cycle through different methods of unsupervised learning. One generic approach might be to start off with k-means clustering or hierarchical, as these are the fastest of the methods, and from these results, get indications as to the appropriate amount of clusters. Then, after getting indications of how many clusters to use through visual inspection and k-means, other more specialized clustering algorithms like spectral or GMM can be used. Spectral and GMM clustering will likely be more flexible in regards to the shapes of the clusters, and might results in a more precise clustering of the data. From this, the user can compare methods, and get a better grasp of the system.

3.1.2 Dimensionality reduction

The option to perform principal component analysis was added to PyVisA in order to facilitate dimensionality reduction. The software used was from the machine learning library scikit-learn in Python [37]. There are vast possible application for the usage of PCA, which is why the simulation data, the scores from the PC's, the loadings, correlation matrix and explained variance will be saved and compressed to a hdf5 file. This will allow each user to tailor the analysis to their specific needs and simulations.

The analysis and visuals being generated are meant as a guideline for where the user should continue their search. The PCA option in PyVisA will generate plots of the scores of the first two principal components in order to inspect if there is some behaviour of interest between the principal components. This is illustrated in Fig. 12 where the first two PC's have been plotted for a small test set of 10 cycles. The loadings and cumulative explained variance will also be plotted, as a control for the reliability of the principal components. This is illustrated in Fig. 13 where the CEV has been generated for the same test set as in Fig. 12.

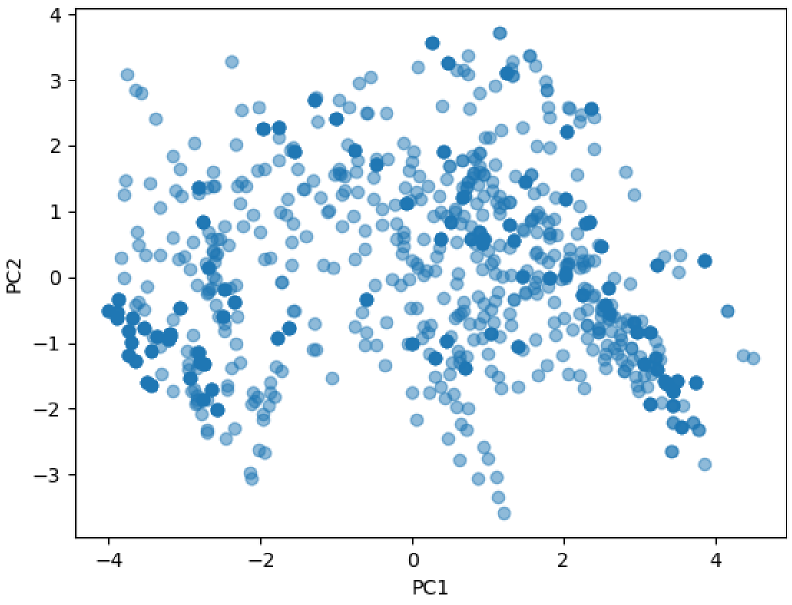


Figure 12: Example of a plot of the first two principal components plotted against each other from PyVisA, for a small test set consisting of 10 cycles [5].

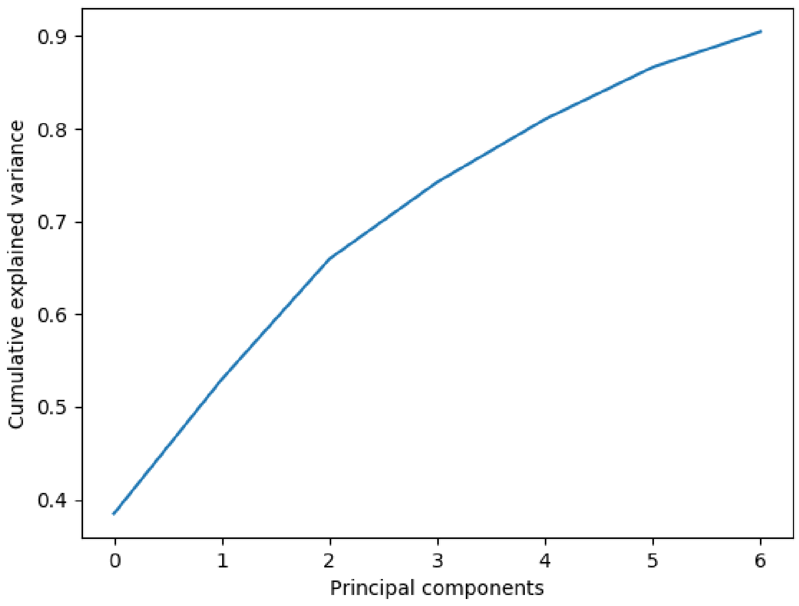


Figure 13: Example of the plot of the cumulative explained variance from PyVisA, for a small test set consisting of 10 cycles [5].

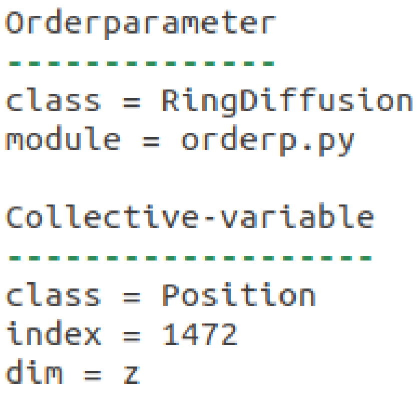
In chemical reactions and the study of rare events, one descriptor might only cover a part of the reaction and the reaction pathway. As reaction are often complex mechanisms it might be necessary to utilize multiple order parameter descriptors in order to find the transition states. PCA will be able to provide the user with these descriptors as the principal components are linear combinations of the original descriptors. This can also apply to stable states and energy minimums of the system.

Another use of PCA is in combination with the different clustering methods that have been implemented. Many classical methods for data exploration, such as k-means, suffer when the dimensions of the data become to large [38]. In the case of k-means clustering, there may not be any clear structure or separation between the cluster, the amount of cluster can be difficult to choose, and the placement of the cluster centers can cause faulty cluster assignments. To combat this inefficiency, it is often useful to perform PCA on the dataset prior to performing the clustering. By performing the PCA, which aims to reduce the dimensions d, and clustering which aims to compress and label the n observations, one produces a compression of dimensions and observations. The user can then utilize the compressed data files generated by PyVisA to explore the data further.

3.1.3 Post-processing

The post-processing feature that has been added to PyVisA consists of a tool that was developed during the autumn of 2019 [5] and is available as an option through the File-menu. The post-processing tool collects the trajectory files from a complete simulation, and by using the functionality for recalculation within PyRETIS [39], copy the existing order parameter and compute new collective variables. The newly computed descriptors can then be visualized by PyVisA.

The prerequisites for the computations is that the user adds the code for the new collective variables to the order parameter script file or another independent python file. Then by writing the names and inputs of the new collective variables to the input .rst file, the calculations can proceed. An example for adding an additional collective variable is shown beneath in Fig. 14. For further reading on this topic, see the PyRETIS website [40] where an example on the use of PyVisA is available and a tutorial for the new features will be available after the release of the software update.



```
Orderparameter
--------------
class = RingDiffusion
module = orderp.py

Collective-variable
-------------------
class = Position
index = 1472
dim = z
```

Figure 14: Formatting for addition of collective variables in the input file of PyRETIS.

The benefits of being able to perform post processing on simulation results are many. Firstly, a simulation of a rare event is time consuming as the simulation might need thousands of cycles which all perform costly molecular dynamic steps. Therefore it can be beneficial to run the simulation, and then add the new collective variables after inspecting the initial data. This can save time as there is no need to run another simulation, and the user can customize new descriptors for the system after reviewing the original results. Secondly, one can also use the new descriptors to perform statistics on the system and validate the outcomes. This can be done in order to reveal mechanisms and behaviours in the data that where not initially available. This facilitates exploration of data, and can be beneficial in describing a molecular system or reaction. If the user discovers a high degree of outliers in the values of the descriptors, it is also possible to design new collective variables that try to capture these. If new descriptors are added to the system, the order parameter will not be changed in order to retain the validity of the rate calculations. [5]

3.1.4 Correlation Matrix

This feature will calculate and display the pairwise Pearson correlations between all descriptors in the simulation [21]. An example of the correlation matrix is shown in Fig. 15 This is a useful way to start the analysis of the CV's of a simulation as it can give early indications of where to start with visualizing results and performing post-processing. It can also be used for dimensionality reduction as it can single out the descriptors which make a small contribution to the system. In Fig. 15, this could be op10, as is shows almost a linear negative relation to the order parameter, meaning that it might be a superfluous descriptor.

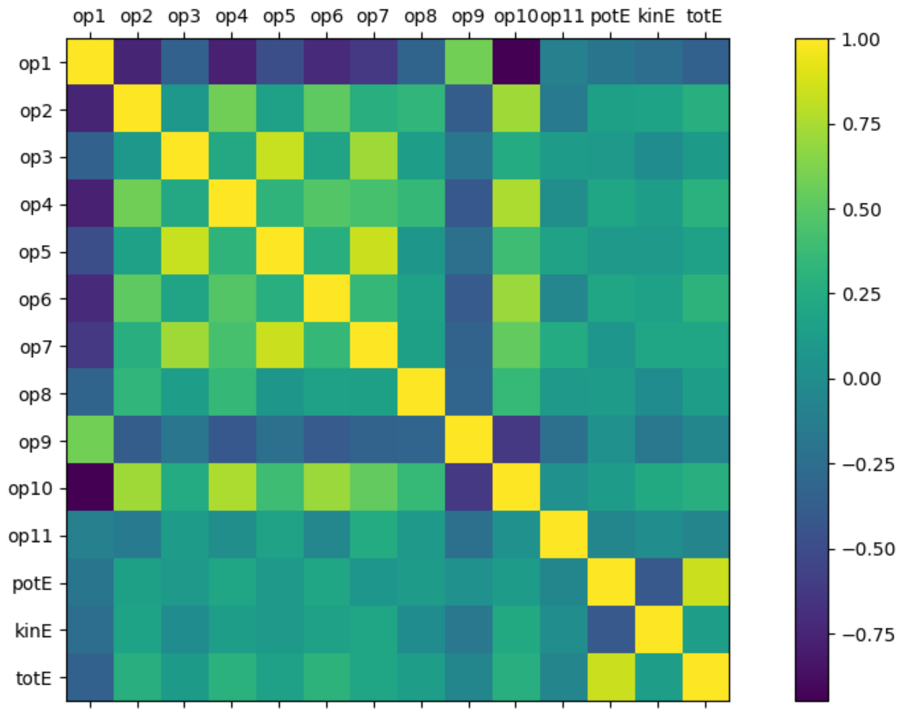
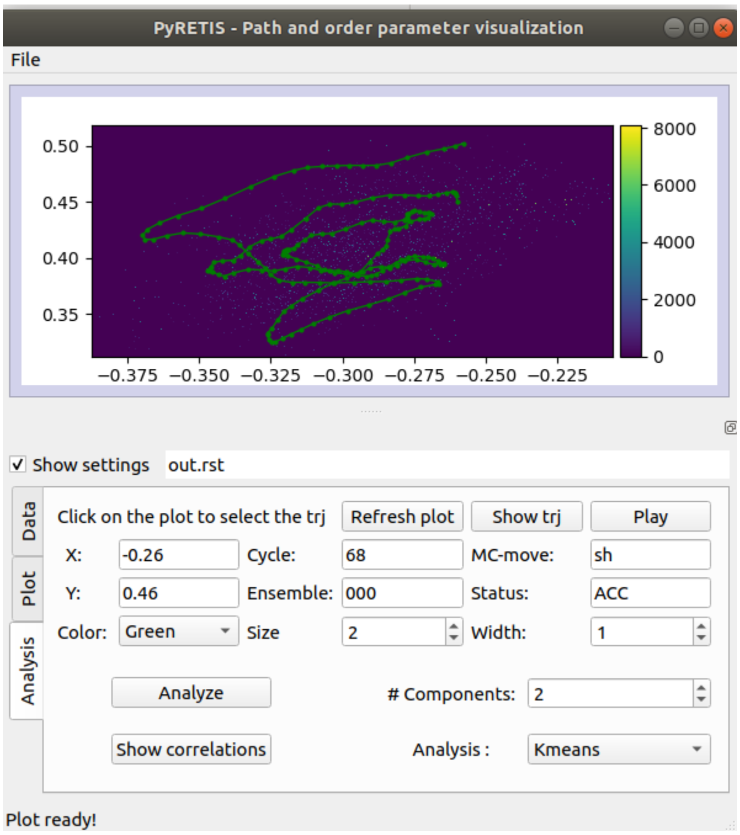


Figure 15: Correlation matrix computed by PyVisA for small test set of 10 cycles and 11 descriptors.

3.1.5 Interactive plots and animation

The functionality to pick points on the plot has been introduced where PyVisA will calculate the closest point in relation to where the user picks. The closest point will be colored in a selected color and information regarding the origin of the point, i.e the coordinates, cycle number, ensemble name, status, and MC generation move will be displayed. Further, there is the option to show the whole trajectory to which the selected point belongs. Here the user can select color, size of points and width of intersecting lines between points. This is illustrated in Fig. 16, where a point has been selected and the trajectory has been displayed in green from a density plot in PyVisA. An intersecting line between the points in the trajectory has been chosen and the information about the trajectory is displayed.



Show settings: ✓ out.rst

Click on the plot to select the trj

X: -0.26 Cycle: 68 MC-move: sh
Y: 0.46 Ensemble: 000 Status: ACC
Color: Green Size 2 Width: 1

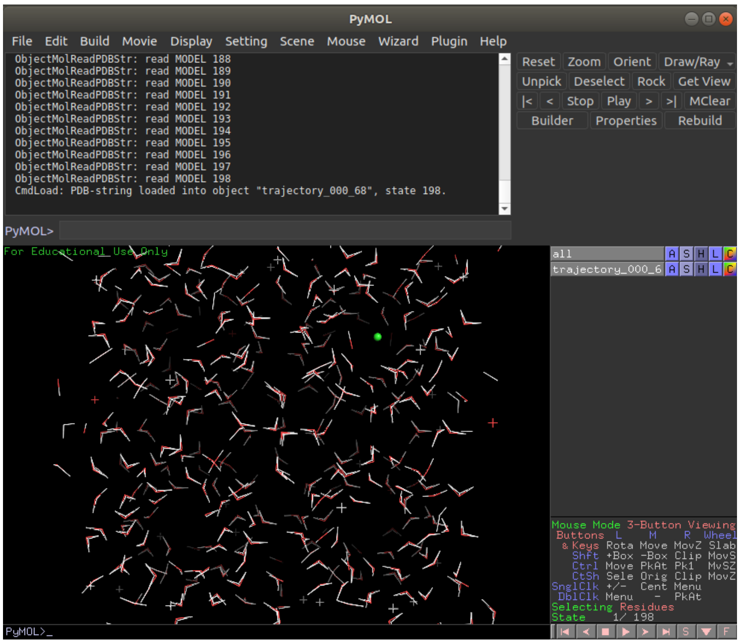
Components: 2
Analysis : Kmeans

Plot ready!

Figure 16: Example of the interactive picking on the plots in PyVisA for a test set of 100 cycles for the methane hydrate system [2].

This feature can be helpful in visualizing reactive pathways in the simulation. It can also facilitate further inspections into specific trajectories if there appears to be an underlying behaviour of interest in the plotted data. In combination with a density plot, it can be useful to show a whole trajectory as it can illustrate how a system moves between different densely populated areas of the descriptors. As Fig 16 is only made up of 100 cycles, one can easily distinguish the different points. But if one wishes to single out one trajectory from several thousands cycles over several ensembles, this will be more difficult.

The ability to select and visualize single trajectories will also be helpful when coupled with the possibility to animate the chosen trajectory. As the amount of data from a simulation can easily be on the scale of terabytes, it can be beneficial to visualize how the molecules move and interact. This can greatly increase the understanding of the system and the phenomenon that is being studied as well as provide a molecular understanding to the data. In order to play the trajectories, the molecular visualization system PyMOL [41], has been integrated so that a user can play a chosen trajectory, given that the trajectory files exist, and a valid configuration file is given. An example of the PyMOL interface is shown if Fig. 17 and shows the methane hydrate system [2].


PyMOL
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw/Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Builder Properties Rebuild
PyMOL>
For Educational Use Only
all
trajectory_000_6
A S H L C
Mouse Mode 3-Button Viewing
Selecting Residues
State 1/ 198

```
ObjectMolReadPDBStr: read MODEL 188
ObjectMolReadPDBStr: read MODEL 189
ObjectMolReadPDBStr: read MODEL 190
ObjectMolReadPDBStr: read MODEL 191
ObjectMolReadPDBStr: read MODEL 192
ObjectMolReadPDBStr: read MODEL 193
ObjectMolReadPDBStr: read MODEL 194
ObjectMolReadPDBStr: read MODEL 195
ObjectMolReadPDBStr: read MODEL 196
ObjectMolReadPDBStr: read MODEL 197
ObjectMolReadPDBStr: read MODEL 198
CmdLoad: PDB-string loaded into object "trajectory_000_68", state 198.
```

Figure 17: Example of the PyMOL interface for visualization of molecular systems and trajectories showing the methane hydrate system [2].

3.2 Simulation software

The simulations in this work have been performed using PyRETIS. PyRETIS is a open source python library for the simulation of rare events and emphasizes TIS and RETIS simulation [42]. PyRETIS can make use of external engines for molecular dynamic integration's, and GROMACS was used as the engine. The engine is a numerical integrator that solves Newtons equations of motion which are used in the dynamics of the simulation [5, 42]. GROMACS is a molecular simulations package that can be used to increase the efficiency of a simulation, as it excels at calculating non-bonded interactions, which often is the dominating term in the simulation [5, 43].

3.3 Simulation

Two RETIS simulations of methane diffusion within an S1 hydrate has been run. The first simulation contained no water vacancies, and the second simulation contained one water vacancy. Then, by using the features available in PyVisA, new collective variables where computed and analyzed for both simulations. Thanks to these features, it is possible to understand how this system behaves and create descriptors that can capture phenomena of interest.

The system for the simulations without water vacancies was developed by Magnus Waage [2], and consists of a simulations box of dimensions 2.385 nm x 2.385 nm x 2.385 nm, and 8 unit cells of hydrates. Both simulations where performed at a temperature of 200 K, and at 1 bar of pressure. The pressure in the simulation is quite low in comparison to the formation pressure of the hydrates, but as the dimensions of the simulation box are not moving, the pressure has little significance to the system.[2, 5]

In the first simulation, without water vacancies, 3075 cycles where run, with 25 sub-cycles and 22 path ensembles. In a RETIS simulation a cycle is defined as one MC-move for each ensemble and a sub-cycle is the amount of MD steps that the external engine, GROMACS is this case, will perform before PyRETIS calculates the order parameter and collective variables [11]. The frequency for storing the trajectory files where set to 25 for both simulations [42].

In the second simulation with water vacancies, 6000 cycles where run, with 19 path ensembles and 25 sub-cycles. In order to include the water vacancy, one water molecule from the six-membered ring separating the acceptor and donor cage was removed. As the free energy barrier for the cage jump is expected to be lower with vacancies, less path ensembles where used. This will also increase the overall speed of the simulation.

3.4 Order parameter and collective variables

The OP for the system is labeled as "Cage jump" and is created by Waage et al [2]. It is defined by a vector in the direction between the centers of the starting cage and the donor cage for the L6L jump. The progress of the reaction is measured by the distance that the methane molecule has traveled along this vector, where 0.0 nm indicated the center of the six-membered ring separating the two large cages, shown in green in Fig. 18 where an illustration of the OP is shown [2, 5]. This vector is made in order to remove the effect of the water molecules moving in the hydrate structure as the distance between an interface and centers of the cages can vary by more than 0.5 Å [2].

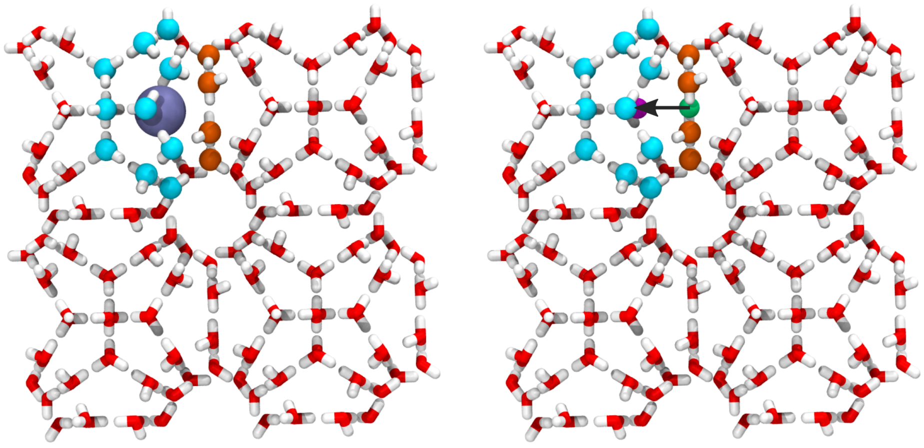


Figure 18: Illustration of the order parameter for the methane hydrate system, with the black arrow indicating the vector between the cages where the methane (dark blue) travels. The figure is taken from Ref. [2].

The following collective variables where created in order to create a framework with correlated and uncorrelated descriptors for the system [5]. In the first simulation without water vacancies, the collective variables that where calculated through the post-processing tool [5], and added to PyVisA are:

- Area of the ring separating the acceptor and donor cage.
- Volume of starting cage.

In the second simulation containing water vacancies, the recalculated collective variables are:

- Area of the ring separating the acceptor and donor cage.
- Volume of starting cage
- The x, y and z coordinates of the methane molecule.

- Distances between the methane and the centers of the two rings connected to the removed water molecule, adjacent to the six-membered ring.
- Number of water molecules in the ring.

The area of the ring separating the acceptor and donor cage and the volume of the starting cage where made in an attempt to visualize the breathing of the cages in respect to the cage-to-cage diffusion. These descriptors where included in both simulations to investigate how the breathing changes with water vacancies.

In the second simulation, the coordinates of the methane molecule are also included. This is because the z-coordinate of the methane has been shown in the previous work to have an almost negatively linear relationship with the order parameter [5]. This is because the system is oriented such that the OP-vector is defined along the z-axis. With water vacancies, it might be of interest to see if the same behaviour occurs. This correlation might be observed if a surrounding water molecule fills vacancy in the ring, thereby moving the vacancy throughout the structure. The L6L jump is the most likely mechanism for the system without vacancies, but as there are now vacancies present, other mechanisms may occur such as the L5S and L5L jump. This is because removed water molecules was also a part of two five-membered rings. In order to investigate if the methane exits one of the two rings connected to the removed water molecule, the distances to the centers of these rings are included. The rings are defined by the remaining water molecules in the original rings. This will also make it visible if the water molecules move, as it will greatly affect the calculation of the centers of the rings. The descriptors will be negative as long as the distance between the starting cage to the center of the ring is larger than the distance between the cage center and the methane. This is done in order to make the descriptors similar to the order parameter. The movement of the water vacancies is also a phenomenon of interest as it is likely to affect the diffusion. Therefore a collective variable defining the number of water molecules in the ring between the acceptor and donor cage will also be added. This is performed by defining a sphere in the initial positions of the water molecules in the ring, and checking if these positions are filled. This CV will be compared to the order parameter. See appendices A.1 and A.2 for the scripts containing the collective variables.

4 Results and discussion

In the following section, the results from the two RETIS simulations will be presented and briefly discussed as accurate calculations of the rate constant and crossing probability is not within the scope of this report. Then, the collective variables that were computed using the recalculation tool of PyVisA will be presented and discussed in relation to the order parameter and to each other. Further the results that where generated through the features for clustering and dimensionality reduction in PyVisA will be presented and their relevance in classifying the mass of data will be discussed. This will be done for both simulations, first the simulation without water vacancies and then the simulation with vacancies.

4.0.1 Results from RETIS simulations

Table 1: Summary of the results from the RETIS simulation without water vacancies calculated by PyRETIS.

Property	**Symbol**	**Value**	**Relative Error (%)**
Crossing probability	$P(\lambda_B\|\lambda_A)$	$5.039 \cdot 10^{-20}$	198
Flux [1/ps]	f_A	0.758	2.679
Rate constant [1/ps]	k_{AB}	$3.821 \cdot 10^{-20}$	198

Table 2: Summary of the results from the RETIS simulation with water vacancies calculated by PyRETIS.

Property	**Symbol**	**Value**	**Relative Error (%)**
Crossing probability	$P(\lambda_B\|\lambda_A)$	$5.895 \cdot 10^{-7}$	40.872
Flux [1/ps]	f_A	0.708	2.809
Rate constant [1/ps]	k_{AB}	$4.173 \cdot 10^{-7}$	40.968

From table 1 and 2 it is clear that the probability and rate of the simulation with vacancies is considerably lower than without vacancies. This is to be expected as the free energy barrier for the cage jump is lower with vacancies. The free energy barrier for the jumps with and without vacancies where calculated by Waage et al [2] to be 14.7 k_BT and 31.7 k_BT respectively at 280 K. With a lower energy barrier, the jump will be more likely to occur and both the probability and rate constant will increase. The relative error where calculated by PyRETIS by a block average analysis. The initial flux shows a lower relative error than both the crossing probability and rate constant. This could be because the rate and crossing probability are both products of all the path ensembles, leading to a cumulative error rate. The crossing probability and rate constant also has similar relative errors, this for both simulation.

In Fig. 19 and Fig. 20, the total probability for performing the L6L jump without and with water vacancies is shown. From Fig. 19 it is possible to see that the five last path ensembles, all contain reactive trajectories that are able to perform the cage jump. It is also possible to see the development of the probabilities between the different ensembles, where the crossing probability is rapidly decreasing after the fourth path ensemble with interface at -0.2 nm. It also illustrates the cause behind the vastly different relative errors between the initial flux and the rate constant. From Fig. 20, more reactive pathways can be seen with a wider range of probabilities. This can indicate that there are several paths for the methane to jump through the ring as the methane has a larger area to jump through when the ring is broken. The curvature of the graph is also less steep compared to Fig. 19, as there is a higher probability for the cage jump to occur.

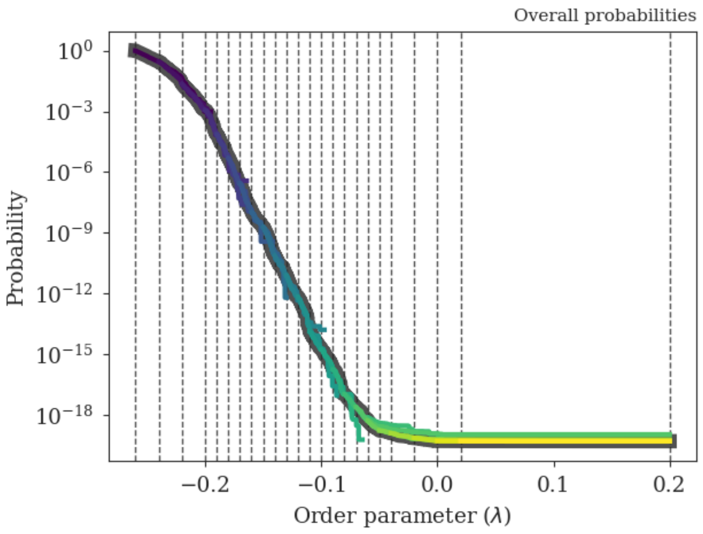


Figure 19: Total crossing probability for the simulation without vacancies

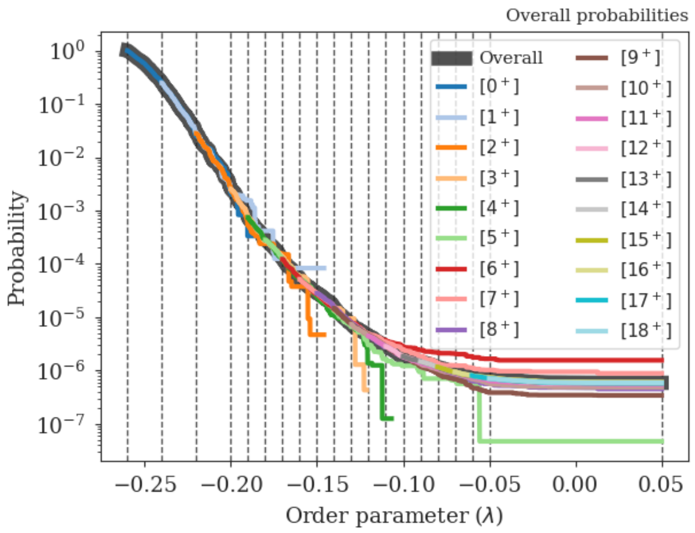


Figure 20: Total crossing probability for the simulation with vacancies

4.1 Methane hydrate without water vacancies

While the simulation without water vacancies need approximately six days to run the 3075 cycles, the recalculation of the collective variables only needed 10 minutes. Although this corresponds to a reduction of data, determined by the frequency at which the trajectory files are stored, the speed of calculations are vastly different.

4.1.1 Energy terms and stable states

In Fig. 21, the correlation matrix is shown. The matrix show a strong positive correlation between the potential and total energy, while little to no correlation between kinetic and total energy,

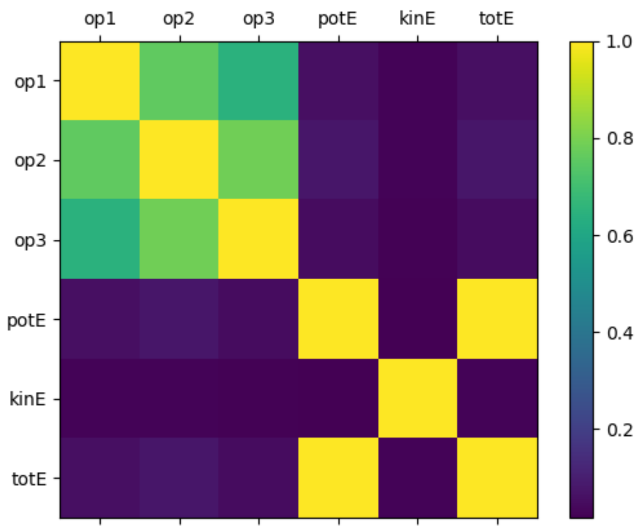


Figure 21: Correlation matrix for the order parameter, collective variables and energies for the simulation without water vacancies. The labels are: op1 = Cage jump order parameter; op2 = Area of ring separating acceptor and donor cage; op3 = Area of starting cage.

In order to inspect the energies further, a density plot was made of the potential and kinetic energies, see Fig. 22. For the density plot, the data from all ensembles and cycles was used in order to visualize the sampling space. In this plot the potential energy shows two distinct clusters at around -1000 kJ/mol and -18 000 kJ/mol while the kinetic energy varies around 1700 kJ/mol to 1950 kJ/mol. The two extremities might represent stable states of the system, but there is little to no correlation between the order parameter and potential energy. This indicates that the differences in potential energy could stem from the water molecules in the system and not from the jump of the guest molecule. As there are seemingly no discontinuities in the order parameter and collective variables this could also be a readout problem occurring between PyRETIS and GROMACS. It is also clear from the correlation matrix that the order parameter and collective variables have little to no correlation with the different energy descriptors. This is illustrated in Fig. 23 where a scatter plot has been generated for the OP and potential energy, with a color map from the kinetic energy. The data plotted is from all path ensembles and cycles, with only the accepted trajectories generated through the shooting move. This is done in order to visualize as many unique trajectories as possible. The scatter plot is consistent with the results from the correlation matrix, and shows the two distinct regions of the potential energy with only small fluctuations in the kinetic energy. A reactive pathway has also been highlighted in red. The trajectory is displaying erratic behavior which might further indicate that the jumps in potential energy does not stem from the cage-to-cage diffusion, but rather a readout issue between GROMACS and PyRETIS as the OP is affected by neither the potential nor the kinetic energy. However due to the simulation time needed and time limitations, a new simulation could not be performed.

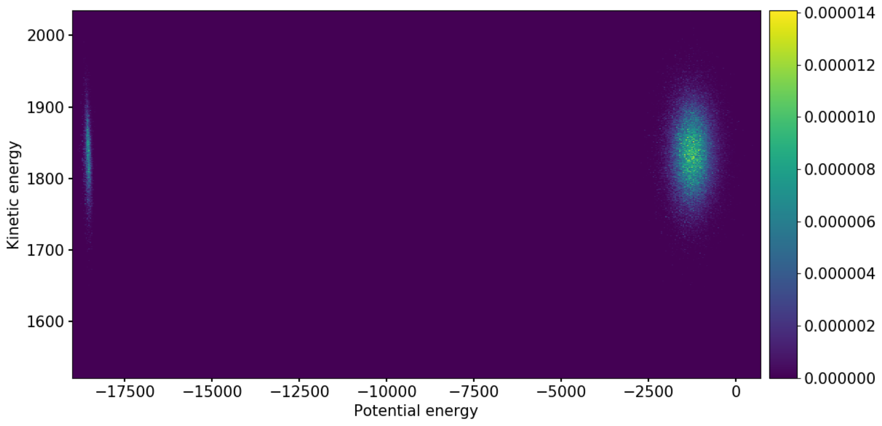


Figure 22: Density plot of the potential [kJ/mol] and kinetic energies [kJ/mol] for all cycles, ensembles and trajectories from all ensembles with only the accepted trajectories generated with a shooting move

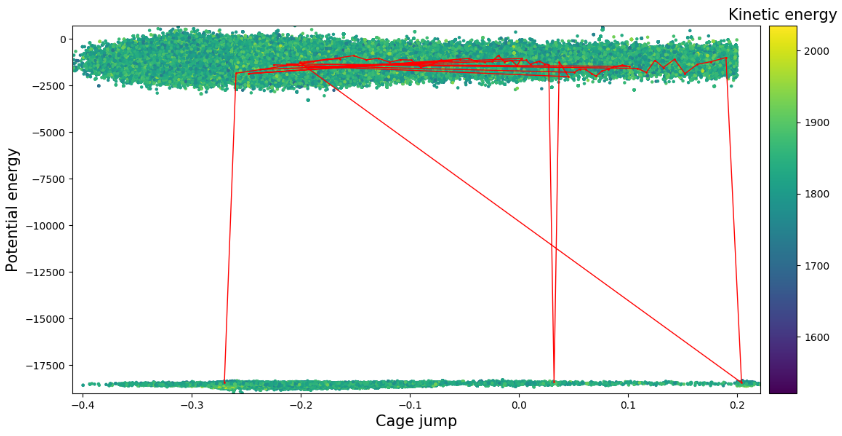


Figure 23: Scatter plot of the cage jump [nm], potential [kJ/mol] and kinetic energies [kJ/mol] from all ensembles with only the accepted trajectories generated with a shooting move. One reactive trajectory has been highlighted in red with an intersecting line between points in the trajectory.

4.1.2 Breathing of water cages

The breathing of the water cages are described by the area of the ring separating the acceptor and donor cage, and the volume of the starting cage, labeled as op2 and op3 in the correlation matrix. When plotting the order parameter against each of the two collective variables, see Fig. 24 and Fig. 25, both plots show a similar behavior. This is in accordance to the results from the correlation matrix where the collective variables show a correlation of around 70%. Both the area of the six-membered ring and the volume of the starting cage seem to be increasing as the values of the order parameter goes towards 0.0 nm. This is when the methane is nearing the center of the ring separating the acceptor and donor cage. The area of the ring and volume of the cage seem to be most frequently in the range of 0.40 to 0.55 nm^2, and 0.25 to 0.28 nm^3 respectively. The increase in the collective variables could be indications of the breathing of the cage, where the cage expands as the methane is jumping between the cages. However, there are also reactive pathways showing lower values for the CV's as the cage-jump occurs.

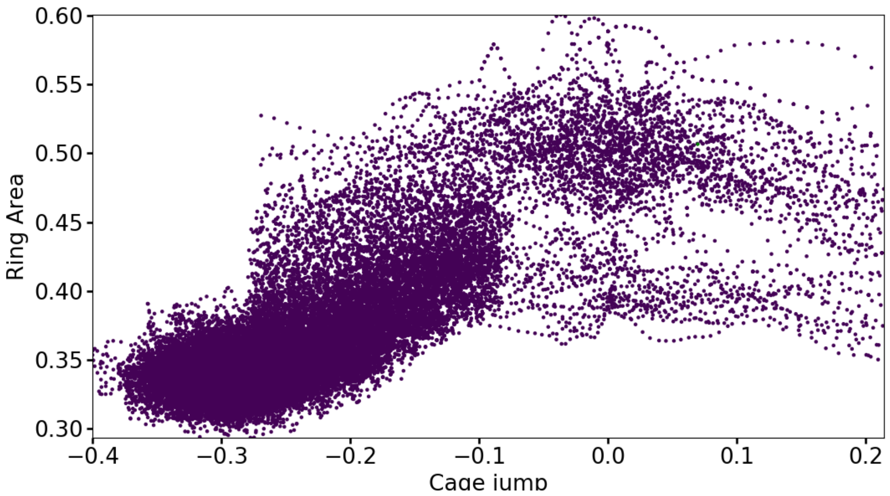


Figure 24: Scatter plot of the cage jump [nm] versus the area of the ring separating the acceptor cage [nm^2] for all accepted trajectories in all ensembles and cycles.

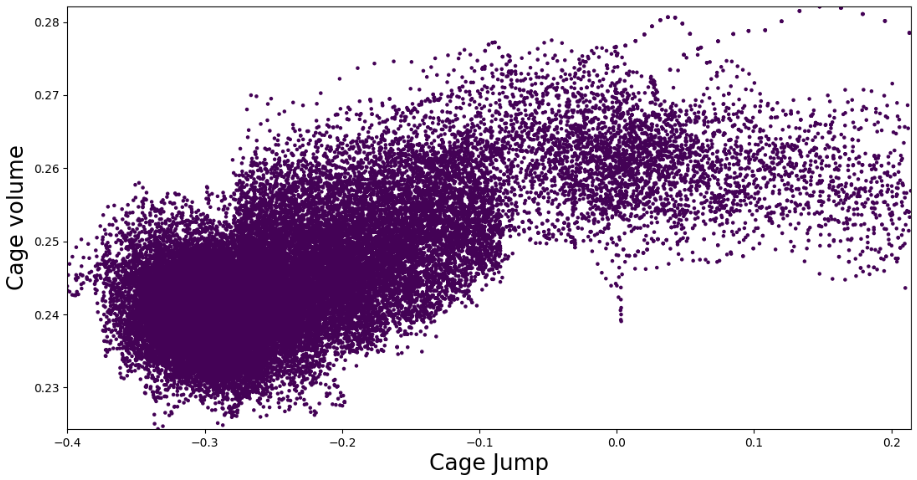


Figure 25: Scatter plot of the cage jump [nm] versus the the volume of the starting cage [nm^3] for all accepted trajectories in all ensembles and cycles.

4.1.3 Principal component analysis

Through principal component analysis of the data from the simulation, three components where chosen. With three components, the new model retained 90% variance of the original data. The cumulative explained variance and loadings of the components is shown in Fig. 26. By analyzing the plot of the scores from the two first components, two distinct clusters are visible. See Fig. 27 for the scores plot. Through the loadings, it is visible that the order parameter and collective variables have a high positive representation within the first component. The potential and total energy has a high negative representation in the second component and the third component is singularly made up of the negative of the kinetic energy. This means that by compressing the data into three components, the first component explains the OP and CV's, the second component explains the potential and total energy, and the third component is the kinetic energy. This also indicates that in the scores plot, the OP and CV's make up the lower cluster and the potential and total energy make up the top cluster. Although the scores are distinct, they are not completely uncorrelated, which can also be seen from the loadings. In the loadings, PC1 and PC2 have some presence of each others main descriptors. From the PCA results one can learn that the system is separated into three terms, the OP and CV's, the potential and total energy and lastly the kinetic energy. PCA also achieves a reduction of dimensions, from six to three descriptors, making it possible to visualize all the simulation data simultaneously.

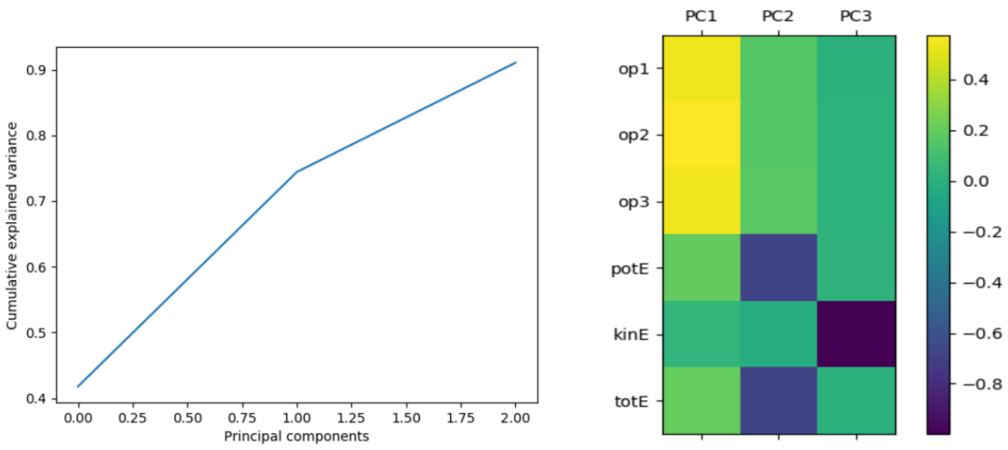


Figure 26: Cumulative explained variance (left) and loadings for all three components (right) for the simulation without water vacancies, where the labels are: op1 = Cage jump order parameter; op2 = Area of ring separating acceptor and donor cage; op3 = Area of starting cage.

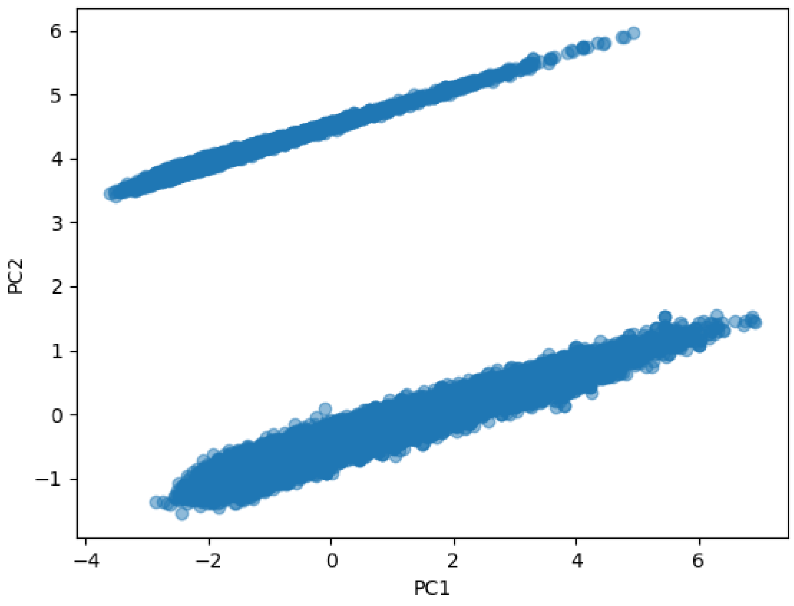


Figure 27: Score plot of the two first principal components.

4.2 Methane hydrate with water vacancies

In order to inspect the results from the simulation, and to get an initial start of the analysis, the correlation matrix was constructed through PyVisA, see Fig. 28. Similarly with the simulation without water vacancies, the energies show little correlation with the OP and CV's. The potential and total energy are strongly correlated while the kinetic energy only shows a minor correlation with the other energy terms. It is also made clear that the z-coordinate of the methane molecule, labeled as op8, correlates almost completely in a negative manner with the order parameter. When plotted, the OP and z-coordinate have an R^2 value of 0.97, see Fig. 38 in appendix B. The number of water molecules in the ring separating the acceptor and donor cage, op9, is mostly uncorrelated as it is a quantized descriptor.

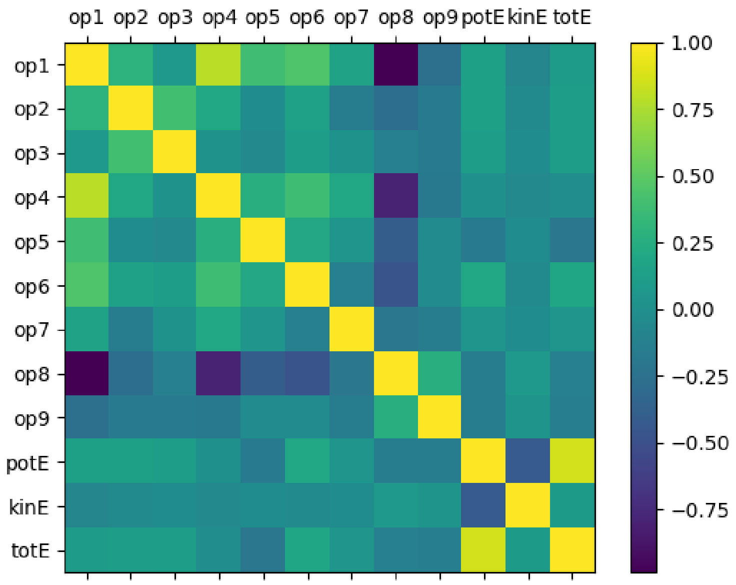


Figure 28: Correlation matrix of the order parameter, collective variables and energies from the simulation with water vacancies. The labels are as follows: op1 = Cage jump order parameter; op2 = Area of ring separating acceptor and donor cage; op3 = Area of starting cage; op4 and op5 = Distances between methane and the centers of the two five-membered rings in which the removed water molecule was a part of; op6 - op8 = The x, y and z coordinate of the methane molecule and op9 = number of water molecules in the ring separating the two cages.

4.2.1 Energy terms and stable states

In this simulation, the potential energy also shows two clusters similarly to the simulation without vacancies. See Fig. 22 for the density plot. Here, it can be seen that the potential energy vary between -18100 kJ/mol to -18600 kJ/mol and the kinetic energy vary between 1700 kJ/mol to 1950 kJ/mol. This might indicate that the lowest value cluster of the potential energy from the simulation without water vacancies are the values caused by the cage jump and that the highest value cluster is the background values of the water molecules in the system. To further visualize the clusters of the potential energy, k-means, Gaussian mixture and spectral cluster plots was created using two cluster, see Fig. 39, 40 and 41 in appendix B. The Gaussian mixture provided the best clustering results as k-means and spectral clustering was not able to define the elliptical clusters. This might occur as there is simply so much data to cluster that the general shape of the clusters become less meaningful in the algorithms. So unless the variance of the data is taken in to account when creating the clusters, the correct shapes will not be produced. This also visualizes the general problem with k-means where the class boundaries become linear.

By creating a density plot of the potential energy with the order parameter, there are indication of a stable state at values for the cage jump around -0.26 nm, see Fig. 30. This is at the interface of the first path ensemble, which means that the high density areas consist of many of the trajectories included in the $[0^-]$ and $[0^+]$ ensembles. Here the potential energy has its lowest values at around -18500 kJ/mol. By further inspection, the same values are measured when the cage jump has values above 0.05 nm. This might indicate that the methane enters a new stable state when it is nearing the center of the acceptor cage.

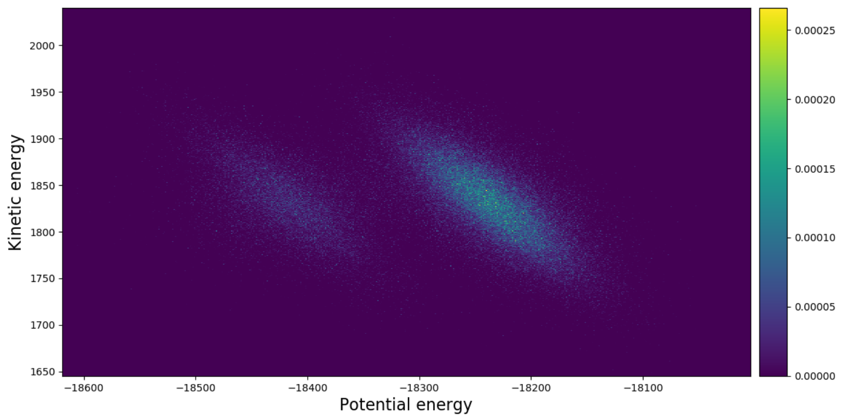


Figure 29: Density plot of the cage jump [nm] versus the kinetic energy [kJ/mol] for all accepted trajectories in all ensembles and cycles for the system with one water vacancy.

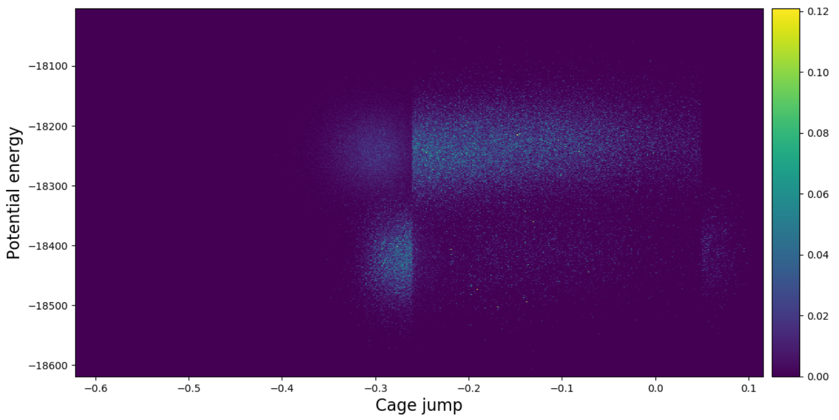


Figure 30: Density plot of the cage jump [nm] versus the potential energy [kJ/mol] for all accepted trajectories in all ensembles and cycles for the system with one water vacancy.

4.2.2 Breathing of water cages

Through inspection of the correlation matrix, the area of the six-membered ring and the volume of the starting cage, labeled op2 and op3, both have a slight correlation with the order parameter. Density plots where created with the data from all accepted trajectories from all cycles and ensembles, see Fig. 31 and 32. The plots aim to visualize all of the sampling space and shows that the behaviour of both descriptors are similar to how the they behave during the simulation without water vacancies, see Fig. 24 and 25. This implies that the same breathing mechanisms occur during the cage-jump. For both simulations the majority of the reactive pathways have values for the descriptors which increase as the methane jumps between cages. The largest increase in values for both descriptors occur until the cage jump reaches -0.2 nm, where they behave in increasingly linear fashion beyond these points. The values lower than -0.2 nm mainly belong to the $[0^-]$ and $[0^+]$ path ensembles. The increase is most visible at the interface of the $[0^+]$ at -0.26 nm. This could correspond to the stable state visualized through the potential energy and the order parameter in Fig. 30.

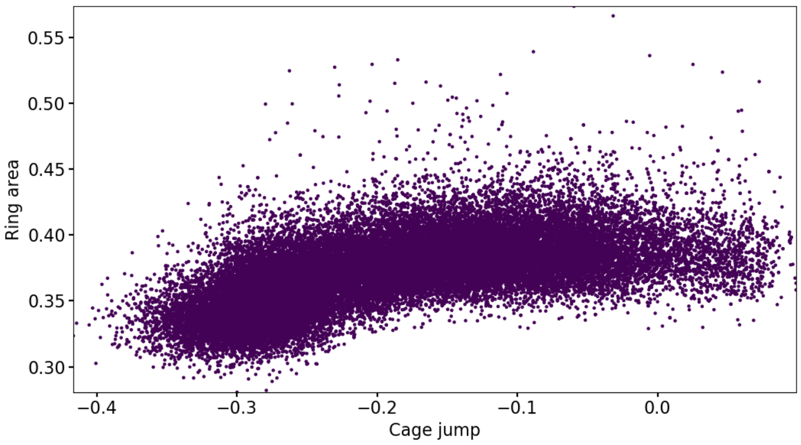


Figure 31: Scatter plot of the cage jump [nm] versus the area of the ring separating the acceptor cage [nm^2] for all accepted trajectories in all ensembles and cycles for the system with vacancies.

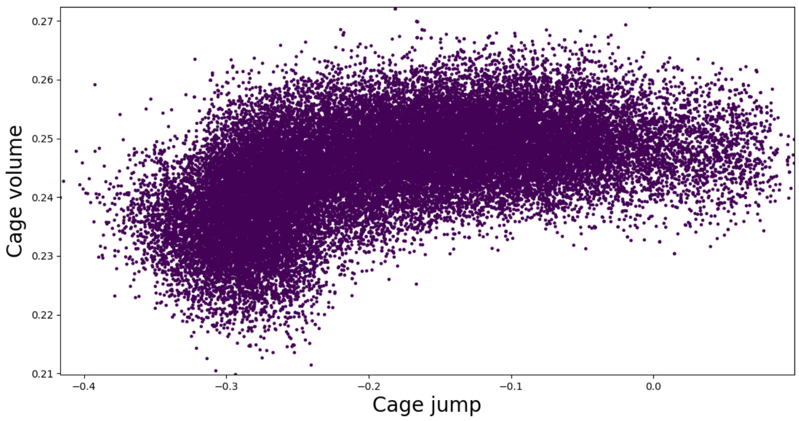


Figure 32: Scatter plot of the cage jump [nm] versus the volume of the starting cage [nm^3] for all accepted trajectories in all ensembles and cycles for the system with vacancies.

4.2.3 Other mechanisms of cage-to-cage diffusion

In Fig. 33 and 34, the order parameter has been plotted against the distances to the centers of the two cages that the removed water molecule was connected to. The distances will be negative until the distance between the center of the starting cage to the centers of the rings are smaller than the distance between the cage center and the methane molecule. As the structures in the hydrate are quite flexible, and the distances use the centers of the ring and cage, these descriptors are not an absolute measure. Instead they act as a guidance in order to establish if there are other mechanisms of cage-jumps present. From the figures, it can be seen that the distances are increasing negatively until the values of the cage jump reach 0.0 nm $\pm$ 0.1 nm. This could correspond to the flexibility of the water molecules in the hydrate. For both figures, the change in sign for the descriptors is most likely corresponding to the methane exiting the starting cage through the six-membered ring. This implies that there were no other mechanisms for cage-jumps during the simulation. With the water vacancy, the six-membered ring represent the path of least resistance for the cage jumps. It can also be seen that the lowest areas of potential energy correspond with findings from the other descriptors. The potential energies are lowest for values of the OP around the $[0]^+$ path ensemble, as in areas where the methane molecule has entered the next cage. These figures also no sudden or large changes in the descriptor values, which could indicate that the water molecules have not moved further throughout the structure.

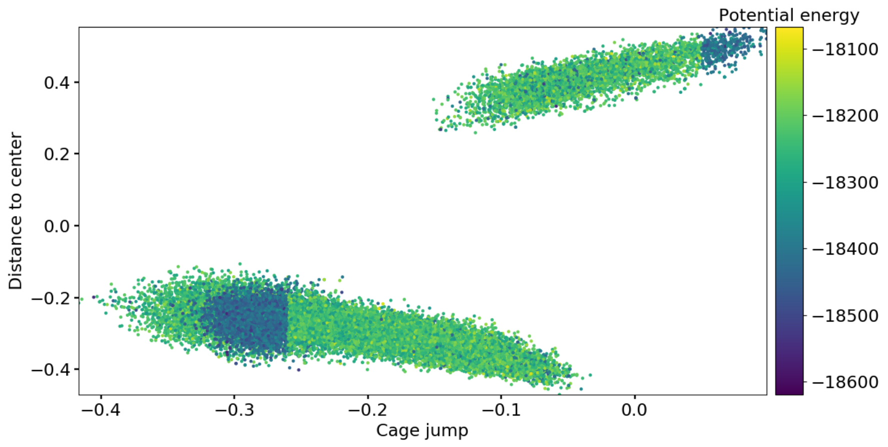


Figure 33: Scatter plot of the cage jump [nm] and the distance to the center of one of the two five-membered rings [nm] that where connected to the removed water molecule, with a color map from the potential energy. This descriptor is labeled as op4 in the correlation matrix.

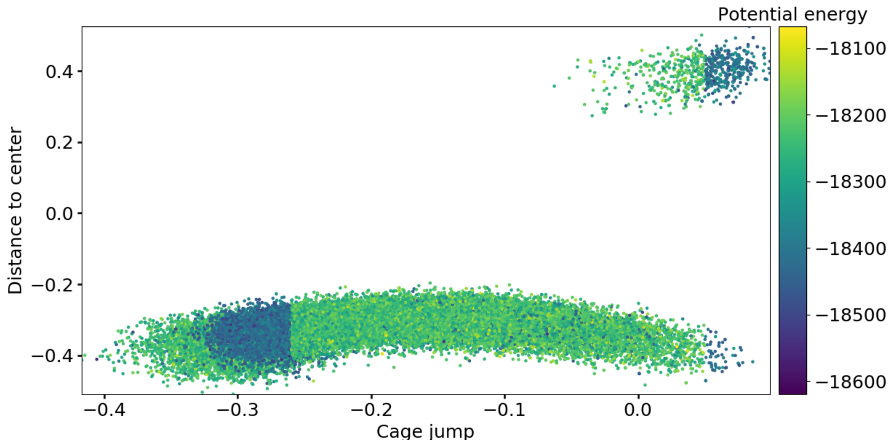


Figure 34: Scatter plot of the cage jump [nm] and the distance to the center of one of the two five-membered rings [nm] that where connected to the removed water molecule, with a color map from the potential energy. This descriptor is labeled as op5 in the correlation matrix.

4.2.4 Movement of water vacancy

Another important aspect of the hydrate system containing water vacancies is the movement of the vacancy. The initial vacancy in the ring can be filled if another water molecule enters the ring, and the vacancy moves throughout the structure. This would likely affect both the rate constant and the crossing probability as well as the trajectories in the simulation. In order to inspect the movements further, a plot of the order parameter and the number of water molecules in the ring has been made, see Fig. 35. From the graph it can be seen that the amount of molecules in the ring start at 5, before varying between 5 and 2. As the number of molecules in the ring should vary between 5 and 6, the low values could indicate that the structure has collapsed. However, through animating and inspecting reactive trajectories from the last path ensemble, this is not the case. Therefore it is visible that the structure is very flexible which is the reason that the descriptor provides values of 2 and 3 molecules. Because of this flexibility, it is not clear whether the number of molecules originate from the vacancy moving or the fluctuations of the water molecules. However, the plot does show that the vacancy in the ring does not get filled. From the order parameter and the rest of the collective variables, op2 - op8, no further evidence regarding the movement of the vacancy and its effect on the system could be found.

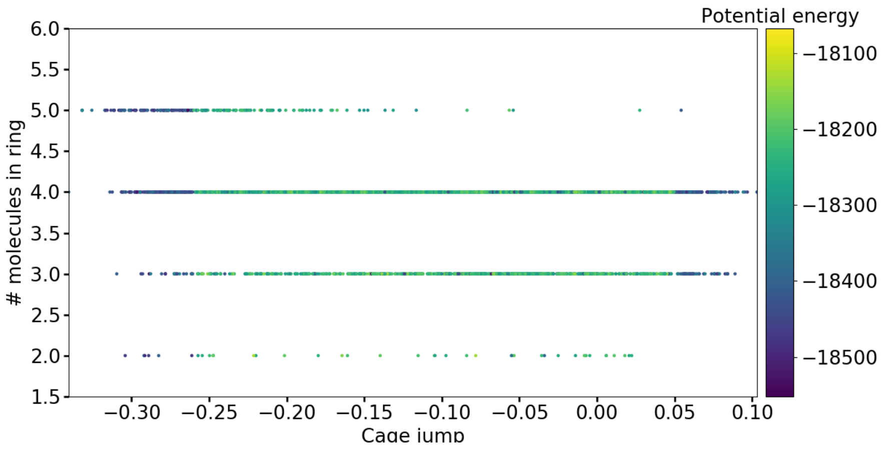
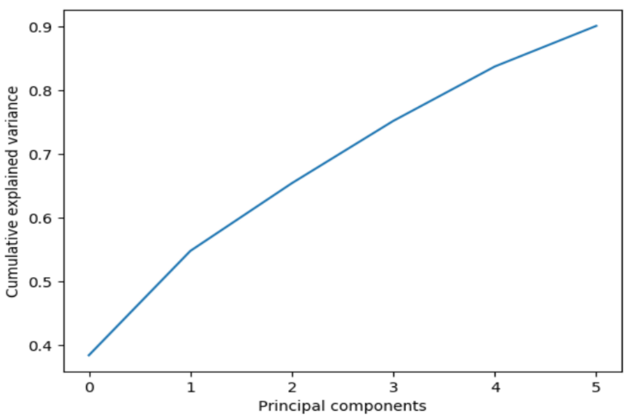


Figure 35: Scatter plot of the cage jump [nm] and the number of water molecules in the ring separating the acceptor and donor cages for all cycles in the last path ensemble with a color map from the potential energy [kJ/mol].

4.2.5 Principal component analysis

After the recalculation done through PyVisA, there are now 12 descriptors for the system describing the diffusion. The last CV, labeled op9, counting the number of water molecules was not included in the PCA as it is not a continuous descriptor. Six principal components where chosen. With six principal component, 90% of the variance was retained. See Fig. 36 for the cumulative explained variance plot and loadings matrix. From the loadings matrix, it can bee seen that the largest terms in PC1 is the order parameter, and op8, which is the z-coordinate of the methane molecule. The loading of the two descriptors are at ± 0.45, as the have a strong linear and negative correlation. Similarly as in the simulation without water vacancies, the second component PC2, consists mostly of the potential and total energy. In PC3, the dominating terms are op2, op3 , op4 and op6. This is the area of the six-membered ring, the volume of the cage, the distance to one of the five-membered rings and the x-coordinate of the methane molecule. The area and volume descriptors are represented negatively, while the distance measure is represented positively. These is also a positive representation of op7, which is the y-coordinate of the methane molecule. Although the largest positive loadings come from the distance descriptor and the y-coordinate of the methane, the two descriptor do not correlate, as shown in the correlation matrix, see Fig. 28. The lack of correlation could occur from the nature of the distance descriptor, as it fluctuates with the flexibility of the water molecules, and changes signs. If the flexibility of the water molecules where accounted for in the descriptor, a larger correlation might have been observed. In PC4, the highest representation of the loading come from the kinetic and total energies. The kinetic energy has the highest value of -0.85, while the total energy has a loading value of -0.36. This is to be expected as the total energy is the sum of the potential and kinetic energies. In both PC5 and PC6, the distance to the other five-membered ring, op5, and the y-coordinate of the methane molecule, op7, are the descriptors with the highest representation in the loadings. The only exception is that the first distance descriptor op4, also has a somewhat large representation in PC6. These two components are also the only ones that show no representation of the energy terms.

Through visual inspection of the scores plot from the two first principal components, see Fig. 37, two regions can be observed. From values of PC1 above 4, there seems to be two clusters that are seemingly not correlated to the first component. This area of the scores plot, could be the potential and total energy terms, similarly to the PCA results from the first simulation. The other region of the scores plot is harder to interpret, and shows no clear behaviour between the two components. This area could also consist of two clusters, but is harder to discern as there is a low degree of separation. The two clusters could then be a representation of the values for the descriptors op1 - op4, and op8, which would give an indication to the cage jump and the breathing of the cage.



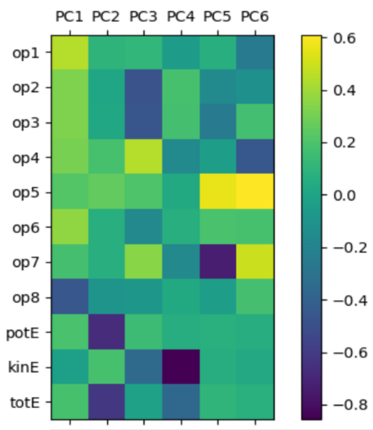


Figure 36: Cumulative explained variance (left) and loadings for all six components (right) for the simulation with water vacancies, where the labels are: op1 = Cage jump order parameter; op2 = Area of ring separating acceptor and donor cage; op3 = Area of starting cage; op4 and op5 = Distances between methane and the centers of the two five-membered rings in which the removed water molecule was a part of; op6 - op8 = The x, y and z coordinate of the methane molecule.

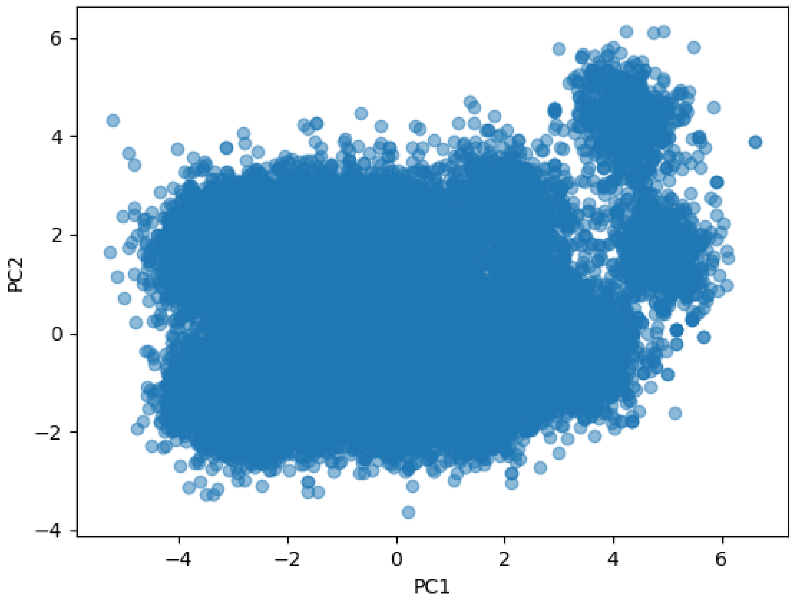


Figure 37: Score plot of the two first principal components.

5 Conclusion

In this thesis a software update for the visualization program for data from molecular simulation PyVisA was developed. The program has been extended to include features for recalculation of collective variables, calculation of the correlation matrix, dimensionality reduction, clustering, interactivity and animation of trajectories and improved options for selection and storage of trajectories. The idea is to improve the users ability to perform post-processing of data from molecular simulations, and guide the user in the search for latent variables and help to classify the mass of data generated from path sampling.

As a case study, and in order to utilize the new features on a molecular system, RETIS simulations of cage-to-cage diffusion within an S1 hydrate was performed and studied. As there are limitations to the lengths of simulations that could be performed, the simulation results act as a proof of concept for the potential and flexibility for performing post processing and the visualization methods that have been added to PyVisA. The simulations contained the methane hydrate, with and without water vacancies in the ring structure. In order to study the diffusion, a set of collective variables where designed and where added to the simulation by using the recalculation tool of PyVisA. The collective variables where the area of the ring separating the acceptor and donor cage, and the volume of the starting cage for both simulations. For the second simulation with vacancies, the x, z and z coordinates of the methane molecule and the distances to the centers of the two adjacent five-membered rings was also included as well as the number of water molecules in the six-membered ring with the vacancy.

In the simulation without water vacancies, principal component analysis was performed. With three principal components, 90% of the variance was retained, whereas the OP and CV's where mostly present in the first component, the total and potential energy where mostly represented by the second component, and the third component was solely comprised of the kinetic energy. While the scores plot displayed two distinct clusters, it is worth to mention that the potential energy displayed erratic behaviour throughout the simulation and as such, no physical interpretations about the energies could be used. There where also indications of increasing values for the volume of the starting cage and area of the ring separating the acceptor and donor cage when the methane performed the L6L jump.

The simulation with water vacancies showed similar results from the principal component analysis, where 90% variance was retained by using six components.

As there where more descriptors in the original data, the descriptors where not as clearly represented in the loadings of the components. However, the order parameter, the energies and some the descriptors involved with the breathing of the cages showed a clear correlation in the loadings that were not present in the correlation matrix. For the collective variables, the same behaviour for the breathing of the cage was observed in relation to the order parameter and the system shows a stable state at around the first path ensemble at values of -18450 kJ/mol. The potential energy also shows two distinct regions, where Gaussian mixture clustering provided the best fit for classifying the data. Further, through analyzing the collective variables that where added, there where no signs of mechanisms of cage jumps other than the L6L jump occurring during the simulation. There where also no clear evidence regarding the movement of the vacancy as the structure is very flexible. However the collective variables suggest that the vacancy in the ring did not get filled during the simulation.

6 Further work

Further work following this Master thesis, is the integration of the developments done during this project into main program, making it available as part of the open source library of PyRETIS. For the simulation results it is worth to emphasize that convergence of the simulation and accurate calculations of rate constants where not within the aim of this thesis as there where time limitations for the length of the simulations that could be performed. Therefore, longer simulations could be run in future in order to improve the behaviour of the system and the statistical analysis. Here, a collective variable describing the movement of the water vacancies could also be included to further research its effect on the diffusion. This CV could utilize the number of hydrogen bonds of the water molecules to locate the vacancy. The molecules surrounding the vacancy will have a lower amount of hydrogen bonds compared to the rest of the molecules in the hydrate. Hence, by finding these water molecules and averaging their positions, the coordinates of the vacancy can be found. Another simulation could also be run in order to investigate the erratic behaviour of the potential energy for the system without vacancies. The behaviour of potential energy might be caused by a readout problem with GROMACS and PyRETIS and the interface between the two could be investigated in order to reproduce and avoid this issue.

References

[1] Erhlich Desa. Submarine methane hydrates-potential fuel resource of the 21st century. 2001.

[2] Magnus H. Waage, Thuat T. Trinh, and Titus S. van Erp. Diffusion of gas mixtures in the si hydrate structure. *The Journal of Chemical Physics*, 2018.

[3] Hossein Dashti et al. Carbon capture and storage from fossil fuel use. *Journal of Natural Gas Science and Engineering*, 2015.

[4] World Ocean Review. Living with the oceans. a report on the state of the world's oceans. `https://worldoceanreview.com/en/wor-1/ocean-chemistry/climate-change-and-methane-hydrates/`, 2010.

[5] Henrik Kiær. A quantitative description of molecular structures: post processing of rare event simulation results. Project report in TKJ4510, Department of Chemistry, NTNU – Norwegian University of Science and Technology, Dec. 2019.

[6] Yamamoto et al. Thermal responses of a gas hydrate-bearing sediment to a depressurization operation. *RSC Advances*, 2017.

[7] H. Herzog and D. Golomb. Carbon capture and storage from fossil fuel use. *Encycl. Energy*, 2004.

[8] Jyoti Shanker Pandey and Nicolas von Solms. Hydrate stability and methane recovery from gas hydrate through ch4 –co2 replacement in different mass transfer scenarios. *MDPI*, 2019.

[9] Peters et al. Path sampling calculation of methane diffusivity in natural gas hydrates from a water-vacancy assisted mechanism. *Journal of the American Chemical Society*, 2016.

[10] E.D. Sloan and C.A. Koh. *Clathrate Hydrates of Natural Gases*. Taylor Francis Group, New York, third edition, 2008.

[11] Anders Lervik, Enrico Riccardi, and Titus S. van Erp. Pyretis: A well-done, medium-sized python library for rare events. *Journal of Computational Chemistry*, 2017.

[12] Enrico Riccardi, Anders Lervik, Sander Roet, Ola Aarøen, and Titus S. van Erp. Pyretis 2: An improbability drive for rare events. *Journal of Computational Chemistry*, 2019.

[13] Alessandro Laio and Francesco L Gervasio. Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. *Reports on Progress in Physics*, 71(12):126601, nov 2008.

[14] Fugao Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86:2050–2053, Mar 2001.

[15] Raffaela Cabriolu, Kristin M. Skjelbred Refsnes, Peter G. Bolhuis, and Titus S. van Erp. Foundations and latest advances in replica exchange transition interface sampling. *The Journal of Chemical Physics*, 147(15):152722, 2017.

[16] Titus S. van Erp. Reaction rate calculation by parallel path swapping. *Physical Review Journals*, 2007.

[17] Peter G. Bolhuis. Rare events via multiple reaction channels sampled by path replica exchange. *The Journal of Chemical Physics*, 129(11):114108, 2008.

[18] The PyRETIS team. Introduction to pyretis and rare event methods. `http://www.pyretis.org/current/user/intro.html#user-guide-retis-theory`, 2019.

[19] D. Frenkel and B. Smit. Understanding molecular simulations from algorithms to applications, 2002. (Academic Press, San Diego, California, USA.

[20] Andrew R. Leach. *Molecular Modelling - Principles and Applications*. Pearson, second edition, 2001.

[21] Gareth James. *An introduction to statistical learning : with applications in R*. Springer, 2013.

[22] Andrew Ng Chris Piech. K means. `https://stanford.edu/~cpiech/cs221/handouts/kmeans.html`, 2012.

[23] Franz Pernkopf and Djamel Bouchaffra. Genetic-based em algorithm for learning gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1344–1348, 2005.

[24] Cory Maklin. Gaussian mixture models clustering algorithm explained. https://towardsdatascience.com/gaussian-mixture-models-d13a5e915c8e, 2019.

[25] Scikit Learn. Gaussian mixture models. https://scikit-learn.org/stable/modules/mixture.html.

[26] Zhongheng Zhang, Fionn Murtagh, Sven Van Van Poucke, Su Lin, and Peng Lan. Hierarchical cluster analysis in clinical research with heterogeneous study population: highlighting its visualization with r. *Annals of Translational Medicine*, 5(4), 2017.

[27] Chaitanya Reddy Patlolla. Understanding the concept of hierarchical clustering technique. https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec, 2018.

[28] Debomit Dey. Ml — hierarchical clustering (agglomerative and divisive clustering). https://www.geeksforgeeks.org/ml-hierarchical-clustering-agglomerative-and-divisive-clustering/.

[29] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[30] Graph (discrete mathematics). https://en.wikipedia.org/wiki/Graph_(discrete_mathematics).

[31] William Fleshman. Spectral clustering-foundation and application. https://towardsdatascience.com/spectral-clustering-aba2640c0d5b, 2019.

[32] Victor Powell and Lewis Lehe. Principal component analysis - explained visually. https://setosa.io/ev/principal-component-analysis/.

[33] Jake VanderPlas. Python data science handbook - essential tools for working with data. https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html, 2016.

[34] Pandas. Pandas documentation. https://pandas.pydata.org/pandas-docs/stable/index.html, 2020.

[35] The PyRETIS team. Pyvisa: Visualization and analysis of path sampling results. http://www.pyretis.org/current/examples/examples-pyvisa.html, 2019.

[36] Mario Lucic, Olivier Bachem, and Andreas Krause. Strong coresets for hard and soft bregman clustering with applications to exponential family mixtures. *arXiv preprint arXiv:1508.05243*, 2015.

[37] Scikit Learn. sklearn.decomposition.pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[38] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1434–1453. SIAM, 2013.

[39] The PyRETIS team. pyretis.tools package. http://www.pyretis.org/current/api/pyretis.tools.html\#module-pyretis.tools.recalculate_order, 2019.

[40] The PyRETIS team. Transport of methane in a si hydrate. http://www.pyretis.org/current/examples/examples-gromacs-hydrate.html, 2019.

[41] PyMOL. Pymol by schrødinger. https://pymol.org/2/.

[42] The PyRETIS team. Introduction to pyretis and rare event methods. http://www.pyretis.org/current/user/intro.html#id9, 2019.

[43] Gromacs. Gromacs.org. http://www.gromacs.org/.

A Input files for RETIS simulation

A.1 Methane hydrate without vacancies

```
Retis Methane-hydrate
=====================

Simulation
----------
task = retis
steps = 3000
interfaces = [-0.26, -0.24, -0.22, -0.20, -0.19, -0.18, -0.17,
              -0.16, -0.15, -0.14, -0.13, -0.12, -0.11, -0.10,
              -0.09, -0.08, -0.07, -0.06, -0.05, -0.04, -0.02,
               0.00,  0.02,  0.20]


System
------
units = gromacs

Engine settings
---------------
class = gromacs
gmx = gmx
mdrun = gmx mdrun
input_path = gromacs_input
timestep = 0.002
subcycles = 25
gmx_format = g96

TIS settings
------------
freq = 0.5
maxlength = 20000
aimless = True
allowmaxlength = False
zero_momentum = False
```

```
rescale_energy = False
sigma_v = -1
seed = 0

RETIS settings
--------------
swapfreq = 0.5
relative_shoots = None
nullmoves = True
swapsimul = True

Initial-path
------------
method = kick
kick-from = previous

Orderparameter
--------------
class = CageJump
module = orderp.py

Collective-variable
-------------------
class = AreaAndVolume
module = orderp.py

Output
------
order-file = 1
trajectory-file = 25
```

A.1.1 Script file

```
import logging
import numpy as np
import scipy
from scipy.spatial import ConvexHull
from pyretis.orderparameter import OrderParameter, Distance
```

```python
from pyretis.orderparameter.orderparameter import Distance
logger = logging.getLogger(__name__)  # pylint: disable=invalid-name
logger.addHandler(logging.NullHandler())


class CageJump(OrderParameter):
    """CageJump(OrderParameter).

    This class defines the L6L jump order parameter for
    the methane hydrate system.

    Attributes
    ----------
    name : string
        A human readable name for the order parameter
    index : integer
        This selects the particle to use for the order parameter.
    periodic : boolean
        This determines if periodic boundaries should be applied to
        the position or not.

    """

    def __init__(self):
        """Initialise the order parameter.

        Parameters
        ----------
        name : string
            The name for the order parameter
        index : tuple of ints
            This is the indices of the atom we will use the position of.
        periodic : boolean, optional
            This determines if periodic boundary conditions should be
            applied to the position.

        """
        super().__init__(description='Ring diffusion for hydrate')
        self.idx1 = np.array([56, 64, 104, 112, 200, 208], dtype=np.int16)
        # convert to atom index:
        self.idx1 *= 4
        self.idx1 -= 3
        # convert to 0 index:
        self.idx1 -= 1
        self.idx2 = np.array([56, 64, 72, 80, 104, 112, 136, 152, 168, 176,
                              200, 208, 232, 248, 264, 272, 296, 304, 328,
                              336, 344, 352, 360, 368], dtype=np.int16)
        # convert to atom index:
        self.idx2 *= 4
        self.idx2 -= 3
        # convert to 0 index:
        self.idx2 -= 1
        self.idxd = 1472  # index for diffusing atom

    def calculate(self, system):
        """Calculate the order parameter.

        Here, the order parameter is just the distance between two
        particles.

        Parameters
        ----------
        system : object like :py:class:`.System`
            This object is used for the actual calculation, typically
            only `system.particles.pos` and/or `system.particles.vel`
            will be used. In some cases `system.forcefield` can also be
            used to include specific energies for the order parameter.

        Returns
        -------
        out : float
            The order parameter.

        """
        pos = system.particles.pos
        resl = 1.0e3
        cm1 = np.average(np.rint(pos[self.idx1] * resl) / resl, axis=0)
        cm2 = np.average(np.rint(pos[self.idx2] * resl) / resl, axis=0)
        cmvec = cm2 - cm1
        molvec = np.rint(pos[self.idxd] * resl) / resl
        molvec -= cm1
        orderp = -np.dot(cmvec, molvec) / np.sqrt(np.dot(cmvec, cmvec))
        return [orderp]

class AreaAndVolume(OrderParameter):
    """
    AreaAndVolume(OrderParameter)

    This order parameter calculates the area of the six-membered ring
    which the methane molecule jumps through when performing the L6L jump,
    and the volume of the starting cage.

    Attributes:
    ----------
    periodic : boolean
        This determines if periodic boundaries should be applied to
        the position or not.

    """
    def __init__(self, periodic=True):
        super().__init__(description="Area of ring and volume of starting cage")
        self.periodic = periodic
        self.idx1 = np.array([220, 252, 412, 444, 796, 828], dtype=np.int16)
        self.idx2 = np.array([220, 252, 284, 316, 412, 444, 540, 604,
                              668, 700, 796, 828, 924, 988, 1052, 1084,
                              1180, 1212, 1308, 1340, 1372, 1404, 1436, 1468],
                              dtype=np.int16)

    def calculate(self, system):
        pos = system.particles.pos
        ar_ring = ConvexHull(pos[self.idx1]).area
        vol_cage = ConvexHull(pos[self.idx2]).volume
        return [ar_ring, vol_cage]
```

A.2 Methane hydrate with vacancies

```
Retis Methane hydrate
=====================

Simulation
----------
task = retis
steps = 6000
interfaces = [-0.26, -0.24, -0.22, -0.20, -0.18, -0.16,
              -0.15, -0.14, -0.13, -0.12, -0.11, -0.10,
              -0.09, -0.08, -0.07, -0.06, -0.05, 0.05]


System
------
units = gromacs

Engine settings
---------------
class = gromacs
gmx = gmx
mdrun = gmx mdrun
input_path = gromacs_input
timestep = 0.002
subcycles = 25
gmx_format = g96

TIS settings
------------
freq = 0.5
maxlength = 20000
aimless = True
allowmaxlength = False
zero_momentum = False
rescale_energy = False
sigma_v = -1
seed = 0
```

```
RETIS settings
--------------
swapfreq = 0.5
relative_shoots = None
nullmoves = True
swapsimul = True

Initial-path
------------
method = kick
kick-from = previous

Orderparameter
--------------
class = CageJump
module = orderp.py

Collective-variable
-------------------
class = AreaAndVolume
module = orderp.py

Collective-variable
-------------------
class = L5Jump
module = orderp.py
methane_idx = 1468
water_idx = [316, 988, 1084, 1180]

Collective-variable
-------------------
class = L5Jump
module = orderp.py
methane_idx = 1468
water_idx = [796, 924, 988, 1340]

Collective-variable
-------------------
```

```
class = Position
index = 1468
dim = x

Collective-variable
-------------------
class = Position
index = 1468
dim = y

Collective-variable
-------------------
class = Position
index = 1468
dim = z

Collective-variable
-------------------
class = nrInRing
module = orderp.py

Output
------
order-file = 1
trajectory-file = 25
```

A.2.1 Script file

```python
import logging
import numpy as np
import mdtraj
from scipy.spatial import distance, ConvexHull
from pyretis.orderparameter import OrderParameter, Distance
from pyretis.orderparameter.orderparameter import Distance
logger = logging.getLogger(__name__)  # pylint: disable=invalid-name
logger.addHandler(logging.NullHandler())
```

```python
class CageJump(OrderParameter):
    """CageJump(OrderParameter).

    This class defines the L6L jump order parameter for
    the methane hydrate system.

    Attributes
    ----------
    name : string
        A human readable name for the order parameter
    index : integer
        This selects the particle to use for the order parameter.
    periodic : boolean
        This determines if periodic boundaries should be applied to
        the position or not.

    """

    def __init__(self):
        """Initialise the order parameter.

        Parameters
        ----------
        name : string
            The name for the order parameter
        index : tuple of ints
            This is the indices of the atom we will use the position of.
        periodic : boolean, optional
            This determines if periodic boundary conditions should be
            applied to the position.

        """
        super().__init__(description='Cage jump for hydrate')
        self.idx1 = np.array([56, 64, 104, 112, 200], dtype=np.int16)
        # convert to atom index:
        self.idx1 *= 4
        self.idx1 -= 3
        # convert to 0 index:
        self.idx1 -= 1
        self.idx2 = np.array([56, 64, 72, 80, 104, 112, 136, 152, 168,
                              176, 200, 208, 232, 248, 264, 272, 296,
                              304, 328, 336, 344, 352, 360,], dtype=np.int16)
        # convert to atom index:
        self.idx2 *= 4
        self.idx2 -= 3
        # convert to 0 index:
        self.idx2 -= 1
        self.idxd = 1468  # index for diffusing atom

    def calculate(self, system):
        """Calculate the order parameter.

        Here, the order parameter is just the distance between two
        particles.

        Parameters
        ----------
        system : object like :py:class:`.System`
            This object is used for the actual calculation, typically
            only `system.particles.pos` and/or `system.particles.vel`
            will be used. In some cases `system.forcefield` can also be
            used to include specific energies for the order parameter.

        Returns
        -------
        out : float
            The order parameter.

        """
        pos = system.particles.pos
        resl = 1.0e3
        cm1 = np.average(np.rint(pos[self.idx1] * resl) / resl, axis=0)
        cm2 = np.average(np.rint(pos[self.idx2] * resl) / resl, axis=0)
        cmvec = cm2 - cm1
        molvec = np.rint(pos[self.idxd] * resl) / resl
        molvec -= cm1
        orderp = -np.dot(cmvec, molvec) / np.sqrt(np.dot(cmvec, cmvec))
        return [orderp]
```

```python
class AreaAndVolume(OrderParameter):
    """
    AreaAndVolume(OrderParameter)

    This order parameter calculates the area of the six-membered ring
    which the methane molecule jumps through when performing the L6L jump,
    and the volume of the starting cage.

    Attributes:
    -----------
    periodic : boolean
        This determines if periodic boundaries should be applied to
        the position or not.
    """

    def __init__(self, periodic=True):
        super().__init__(description="Area of ring and volume of starting cage")
        self.periodic = periodic
        self.idx1 = np.array([220, 252, 412, 444, 796], dtype=np.int16)
        self.idx2 = np.array([220, 252, 284, 316, 412, 444, 540, 604,
                              668, 700, 796, 828, 924, 988, 1052, 1084,
                              1180, 1212, 1308, 1340, 1372, 1404, 1436], dtype=np

    def calculate(self, system):
        pos = system.particles.pos
        ar_ring = ConvexHull(pos[self.idx1]).area
        vol_cage = ConvexHull(pos[self.idx2]).volume
        return [ar_ring, vol_cage]


class L5Jump(OrderParameter):
    """L5Jump(OrderParameter)

    This class defines the collective variable used for the
    methane hydrate system

    """
```

```python
def __init__(self, methane_idx, indices):
    """Initialize the order parameter.

    Attributes
    ----------
    methane_idx : integer
        Index of the methane molecule in the system
    indices : list
        This selects the particles in the ring we want to use.
    periodic : boolean
        This determines if periodic boundaries should be applied to
        the position or not.
    """
    super().__init__(description='L5 jump for methane.')
    self.methane_idx = methane_idx
    self.periodic = True
    self.water_idx = indices
    self.cage_idx = [220, 252, 284, 316, 412, 444, 540, 604,
                     668, 700, 796, 828, 924, 988, 1052, 1084,
                     1180, 1212, 1308, 1340, 1372, 1404, 1436, 1468]

def calculate(self, system):
    """Calculate the collective variable.

    Here the descriptor is the distance between the methane molecule,
    and the center of the selected ring.

    Parameters
    ----------
    system : object like :py:class:`.System`
        This object is used for the actual calculation, typically
        only `system.particles.pos` and/or `system.particles.vel`
        will be used. In some cases `system.forcefield` can also be
        used to include specific energies for the order parameter.

    Returns
    -------
    out : float
```

```
            The order parameter.

        """

        pos = system.particles.pos
        ch4 = pos[self.methane_idx]
        water = pos[self.water_idx]

        ring = ConvexHull(water)
        cage = ConvexHull(self.cage_idx)
        centroid_ring = np.mean(ring.points[ring.vertices, :], axis=0)
        centroid_cage = np.mean(cage.points[cage.vertices, :], axis=0)
        dist_cage_ring = distance.euclidean(centroid_ring, centroid_cage)
        dist_cage_ch4 = distance.euclidean(centroid_cage, ch4)
        dist_ring_ch4 = distance.euclidean(centroid_ring, ch4)

        if dist_cage_ring > dist_cage_ch4:
            sign = -1
        else:
            sign = 1
        corrected_distance = sign * dist_ring_ch4
        return [corrected_distance]


class Position(OrderParameter):
    """A positional order parameter.

    This class defines a very simple order parameter which is just
    the position of a given particle.

    Attributes
    ----------
    index : integer
        This is the index of the atom which will be used, i.e.
        ``system.particles.pos[index]`` will be used.
    dim : integer
        This is the dimension of the coordinate to use.
        0, 1 or 2 for 'x', 'y' or 'z'.
    periodic : boolean
```

```
        This determines if periodic boundaries should be applied to
        the position or not.

    """

    def __init__(self, index, dim='x', periodic=False):
        """Initialise the order parameter.

        Parameters
        ----------
        index : int
            This is the index of the atom we will use the position of.
        dim : string
            This select what dimension we should consider,
            it should equal 'x', 'y' or 'z'.
        periodic : boolean, optional
            This determines if periodic boundary conditions should be
            applied to the position.

        """
        txt = 'Position of particle {} (dim: {})'.format(index, dim)
        super().__init__(description=txt, velocity=False)
        self.periodic = periodic
        self.index = index
        self.dim = {'x': 0, 'y': 1, 'z': 2}.get(dim, None)
        if self.dim is None:
            msg = 'Unknown dimension {} requested'.format(dim)


class nrInRing(OrderParameter):
    """nrInRing(OrderParameter)

       This class defines the collective variable used for counting
       the amount of water molecules in the ring separating the acceptor
       and donor cage.
       """

    def __init__(self):
        """Initialize the order parameter."""
        super().__init__(description='Water vacancy movement')
        self.periodic = True
        self.indices = np.arange(0, 1465, 4).tolist()
        self.init = [[1.805, 1.504, 1.203],
                     [1.805, 2.105, 1.203],
                     [1.572, 1.662, 1.203],
                     [1.572, 1.947, 1.203],
                     [2.037, 1.662, 1.203],
                     [2.037 ,1.947, 1.203]]



    def calculate(self, system):
        """Calculate the collective variable.

        Here the descriptor is the distance between the methane molecule,
        and the center of the selected ring.

        Parameters
        ----------
        system : object like :py:class:`.System`
            This object is used for the actual calculation, typically
            only `system.particles.pos` and/or `system.particles.vel`
            will be used. In some cases `system.forcefield` can also be
            used to include specific energies for the order parameter.

        Returns
        -------
        out : float
            The order parameter.

        """
        pos = system.particles.pos
        water_pos = pos[self.indices]
        counter = 0
        radius = 0.085
        for water in water_pos:
            for i in self.init:
                diff = np.subtract(water, i)
                dist = np.sum(np.power(diff, 2))
                if dist < radius ** 2:
                    counter += 1
        return [counter]
```

B Various plots

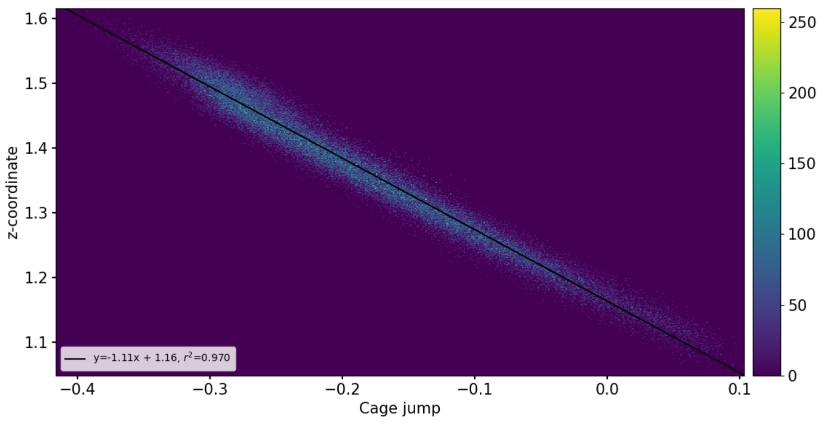


Figure 38: The order parameter plotted against the z-coordinate of the methane molecule for the simulation with water vacancies.

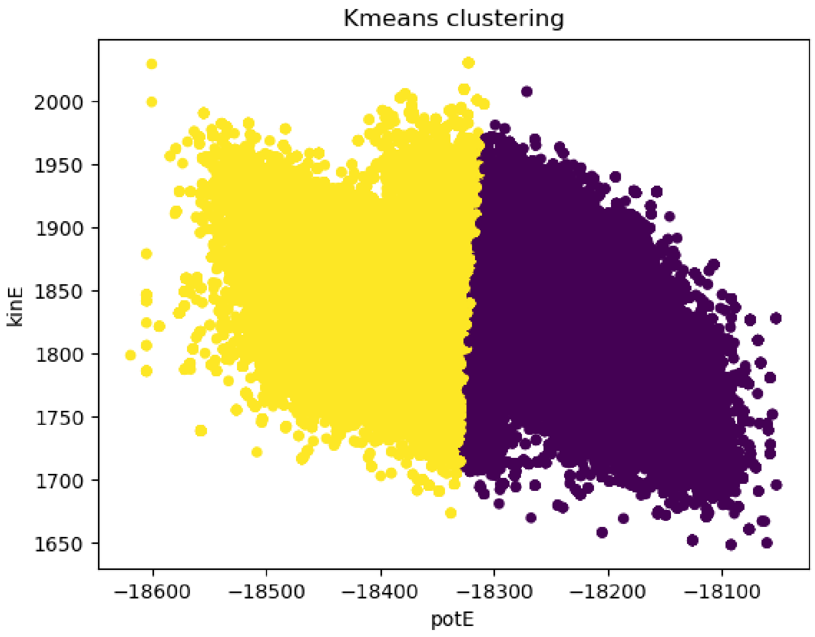


Figure 39: K-means cluster plot of the potential and kinetic energy with two clusters for all cycles and ensembles for the simulation with water vacancies.

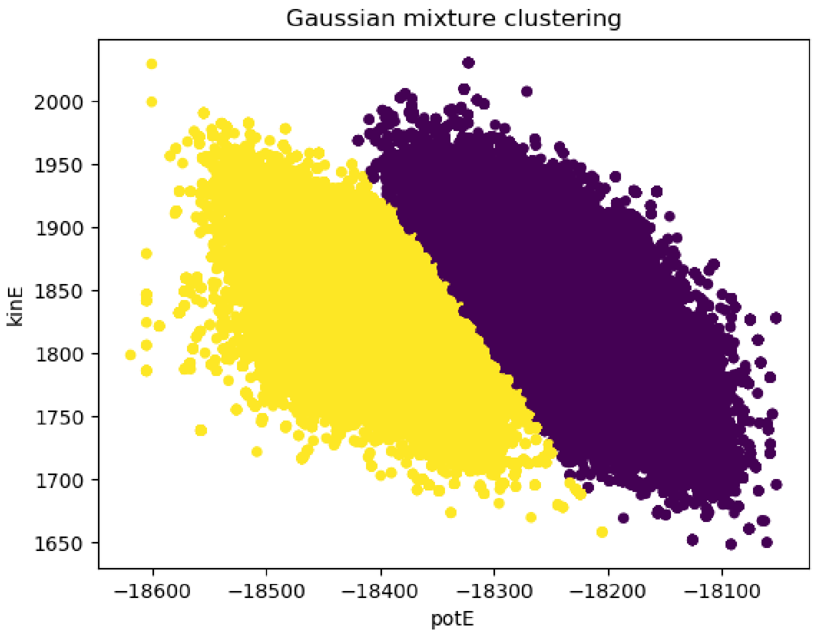


Figure 40: Gaussian mixture cluster plot of the potential and kinetic energy with two clusters for all cycles and ensembles for the simulation with water vacancies.

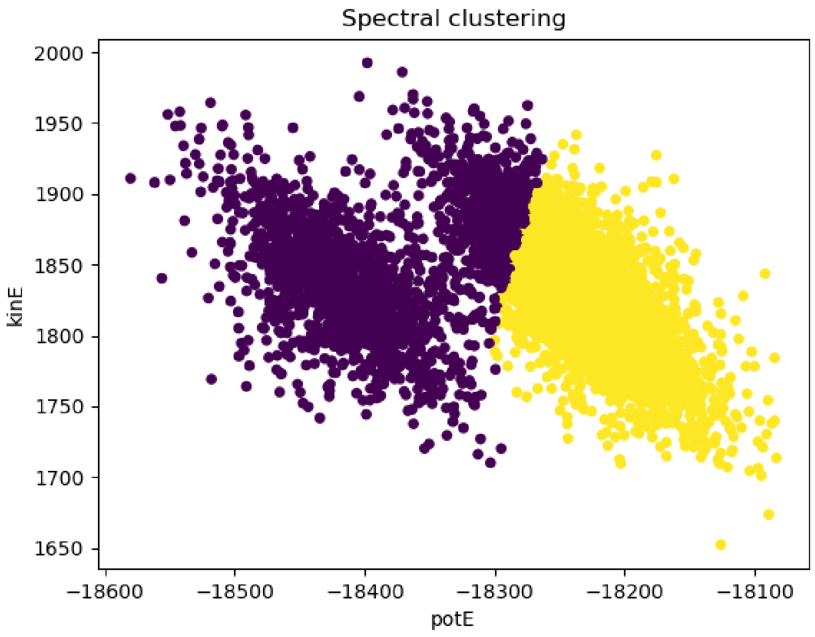


Figure 41: Spectral cluster plot of the potential and kinetic energy with two clusters for all cycles and ensembles for the simulation with water vacancies.