# Analysis of Diffusion in Heterogenous Fluids using Clustering Algorithms.

Mont Grunthal

Florida State University

282 Champions Way, Tallahassee

mjg18g@my.fsu.edu

## 1. Abstract:

Mean squared displacement is a common metric used in statistical mechanics. The mean squared displacement of a particle provides information about how far it has moved in some given time. This is a common measurement used in molecular dynamics, microbiology, and other fields concerned with the motion of sets of particles in a medium. One can calculate the diffusion constant of a particle in a fluid from the Mean Squared Displacement. However, if there is more than one species of particle in a mixture, the rate of diffusion calculated from the trajectories of the particles in the mixture will not reflect the true diffusion constants of any particle. The issue of determining which diffusion constants correspond to which fluid from trajectories is non-trivial. The work proposed here will allow researchers to determine the diffusion constants of mixed particles in a fluid using machine learning when the number of distinct types of particles is not necessarily known. Additionally, this work seeks to inform researchers what methods might be best when handling this task. The clustering algorithms used are k-means and DBSCAN. Following from that, there will be a comparison of the performance of different methods proposed here. The performance comparison will focus on the accuracy of the various methods, not necessarily their efficiency. After running simulations across a range of

possibly problematic cases, we found that k-means could accurately estimate the diffusion values of both particle sets 99% of the time with DBSCAN lagging at 70%. Given this information, we recommend running the program using DBSCAN only when the number of particle sets in the mixture is unknown. The DBSCAN method outputs an estimate for the number of particle sets, which could then be used in a much more accurate k-means run.

**Keywords:**

Brownian Motion, k-means, DBSCAN, Mean Squared Displacement.

## 2.0. Introduction:

### 2.1. Brownian Motion and Mean Squared Displacement:

Brownian motion is the random motion particles experience when suspended in a fluid. This is the result of the constituent molecules of the fluid colliding with the suspended particle at a very fast rate. At any moment, the net force of these collisions accelerates the particle in an effectively random direction. This leads to the particle taking what is essentially a random walk through the medium which it's suspended in. More rigorously, a random walk is a trajectory where, at each time step, a particle moves in a direction based on some random variable. A one-dimensional example of this world be a particle on a line in one dimension moving left or right based on the result of a coin toss. Brownian motion can be understood as the continuous version of the random walk. For

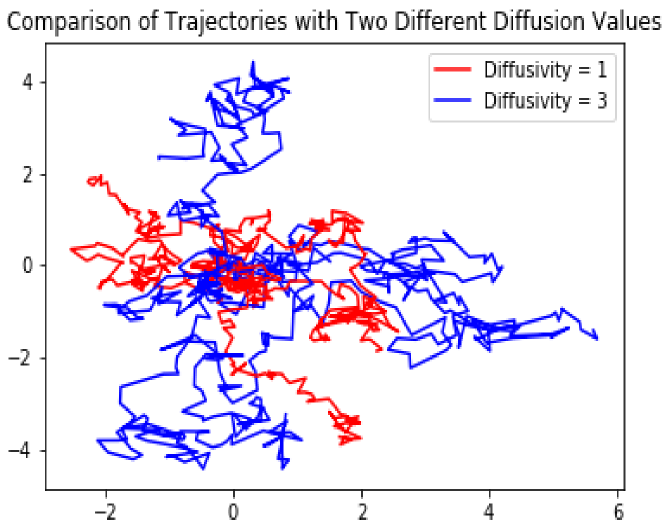

*Figure 1: Two particles with different diffusion values taking a random walk over a period of time. Highlighting the unpredictable and discontinuous nature of this type of motion*

sets of particles in a Newtonian fluid, the distribution of their locations at a given time is described by a gaussian curve. This gaussian distribution describes the diffusion of the particles. The x axis of this gaussian curve is the distance from the origin and the y axis is the likelihood of finding a particle at that location. In order to find the gaussian distribution which describes a particle in practice, we need to know the diffusivity constant of that particle. This brings us to mean squared displacement, a computational method which encodes not just the diffusivity of a particle but also a measure of the distance it has travelled. Mean squared displacement can be defined from a trajectory as follows (michalet, 3):

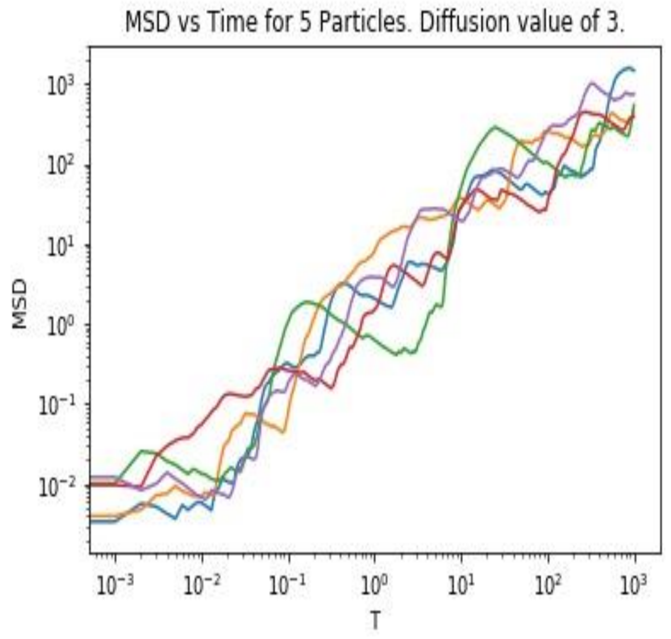

*Figure 2: The mean squared displacment of five particles with respect to an increasing time step. We can see that by avreging this curve, we can gain an accurate estimate of mean squared displacment.*

$$MSD \equiv \langle |x(t) - x_0|^2 \rangle.$$

Initially, the random nature of this motion made it difficult to estimate how far a particle undergoing Brownian motion would travel in each time step. However, Einstein formalized the mathematics describing Brownian motion and afforded us with the tools necessary to empirically describe these motions. One such tool was the mean squared displacement of a particle. This is what we need to tune the gaussian distribution to our situation. Mean squared displacement is a time average of the squared displacement. This tells us the distance a particle has traveled after a given time, and, more importantly for this work, an estimate of the diffusivity constant of the particle (Michalet, 1). The relationship between diffusivity and mean squared displacement is as follows: $MSD = 2 * d * D * t$. Where d is the dimension of the space in which the particle is traveling, t is the time that the particle has been traveling, and D is the diffusivity of that particle. All we need to calculate mean squared displacement is a particles trajectory over time. Given the mean squared displacement we can calculate the diffusivity and tune the probability density function to perfectly describe the position of particles in a medium at a given time. This information can be used to determine other properties of both the particle and the medium. For example, temperature is related to diffusivity in the following manner for certain molecules with spherical geometries:

$$D \propto T^{3/2}$$

**2.2. Cluster Analysis:** The challenge of grouping observations into clusters based on their features is a classic problem in data mining. The interest in clustering, however, goes far beyond data mining. There has been an increasing (since the advent of computing) interest in cluster analysis in many modern scientific fields such as materials science, biology, and even the social sciences, psychology and sociology. In general, cluster analysis partitions a set of observations into groups based on some set of features (Shu, 116). Groups are partitioned based on some measure of similarity within one or more of the features of the observations. It is common for similarity to be measured as a function of all features concurrently. By using all features in the computation of similarity, the overall intra-cluster variation is minimized. It is important to note, and possibly not obvious from figure 3, that

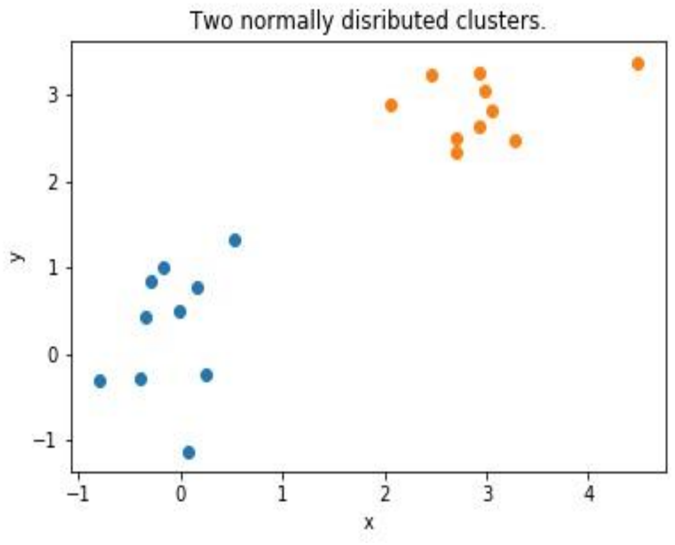

*Figure 3: Two sets of points, distibuted normally, and colored according to groupings determened by k-means.*

clustering does not assign observations to a cluster, in the sense that it classifies individual observations as being part of a group. Instead, clustering generates volumes in space containing the observations in a given cluster (Shu, 118). Thus, each object belongs to exactly one group. The fact that we are drawing partitions in space instead of classifying objects differentiates cluster analysis from smallest space analysis, factor analysis, and multidimensional scaling (Bailey, 62). This causes cluster analysis and its techniques (excluding methods such as fuzzy Kmeans) to satisfy exhaustiveness and mutual exclusivity distinguishing it from the above analyses. In the example shown in figure three, the function used to measure similarity was Euclidean distance thus the two groups were partitioned based on which “center of mass” they were closest to (more on this in the Kmeans section). Metrics used for assessing similarity need not be Euclidian distance. Or even distance for that matter. It is possible to build a contingency table and run a chi squared test based on the frequency of these types of variables. Similarity can then be assessed from the results of this test. It is possible to build a contingency table and run a chi squared test based on the frequency of ordinal and nominal features. In the case of using a chi squared test as a similarity function, objects could possibly be clustered based on their correlation coefficient. In the context of distance being used to determine similarity between observations, distance can be defined many ways, but the function for distance must satisfy the conditions of a metric function. These are as follows (Meszena, 302):

1. $D(x,y) = D(y,x)$,
2. $D(x,x) = 0$,
3. $D(x,y) > 0$, if $x!=y$,
4. $D(x,y) <= D(x,z) + D(y,z)$.

If assumptions 3 or 4 are not satisfied, D(x,y) is considered a pseudo-metric or semi-metric respectively. Such metrics are permissible in specific circumstances. The selection of a similarity measure is only one step in the process of cluster analysis. Additionally, one must select variable over which to run the analysis. “Methods of choosing variables can be dichotomized as descriptive or theoretical,” (Bailey, 63). When choosing variables descriptively, we go through a preprocessing step, and assess which variables are relevant to the analysis. On can also discard variables which are constant or nearly constant for all objects. As these variables will not add anything assessing similarity between objects. Similarly, variables whose values vary randomly add nothing worthwhile to the analysis and can be omitted. Omitting constant and random features reduces the size of the variable space, making the analysis less computationally expensive (. Alternatively, features for analysis can be chosen theoretically. For example, when determining the diffusivity constants of mixed fluids, we would like to group individual particle trajectories by their diffusion constants. We know that diffusivity can be thought of as a function of mean squared displacement. Thus, it is reasonable to choose the mean squared displacement of each particle as the feature on which to measure similarity. Bailey makes note of a possible disadvantage in using theory to determine which features to measure in *Cluster Analysis.* He notes that, often in social sciences, theoretical values can be highly abstract and difficult to assign value to. Examples being conformity, social integration, and alienation. It should be noted that while both of these methods for determining which features to use have strengths and weaknesses, using existing theory to make the determination does not expose your study to the subjectivity of descriptively choosing features. And as such, should be used unless there is compelling reason to choose features descriptively in a given analysis. The importance of reducing the dimensionality of the space we are attempting to partition through clustering cannot be understated. This is the curse of dimensionality.

**2.3. K-means Clustering Algorithm:** k-means is a clustering algorithm developed in the mid 1950's by Stuart P. Lloyd. It has been well studied in the following decades and is now a standard algorithm in cluster analysis. The algorithm attempts to partition the space containing N observations into K clusters. There are many variations of this algorithm, but the simplest case is Lloyd's method, which functions as follows: Given K, the number of clusters, and, X = {$X_{1,}$ $X_{2,}$ $X_3$, …, $X_n$} where X is the set of d dimensional observations. Assign each element of X to an element of set S where S = {$S_1$, $S_{2,}$ $S_3$, …, $S_k$}. The elements of X should be assigned to elements of S such that the variance within each element of S is minimized. Formally, the objective is to minimize the following function for element in S.

$$\sum_{i=1}^{k} |S_i|\, Var(S_i)$$

In practice, this minimization is achieved iteratively through the following steps.

1. Specify K and select K observations randomly to be the initial centroids.
2. Update the location of the centroids following the objective function. The new m value is: $\frac{1}{|s_j|}\sum_{x\in s} x$
3. Assign each point to the new closest centroid. Distance between the point x and the centroid m is computed as the least squared Euclidian distance: $||x - m||^2$.
4. Repeat until a maximum number of iterations is reached or the minimum value for the change in location of the centroids is reached.

The conditions for maximum iterations or minimum change between steps are necessary because it is not guaranteed that k means will converge to an optimal solution. The figure bellow illustrates how centroids move towards a solution across several iterations.

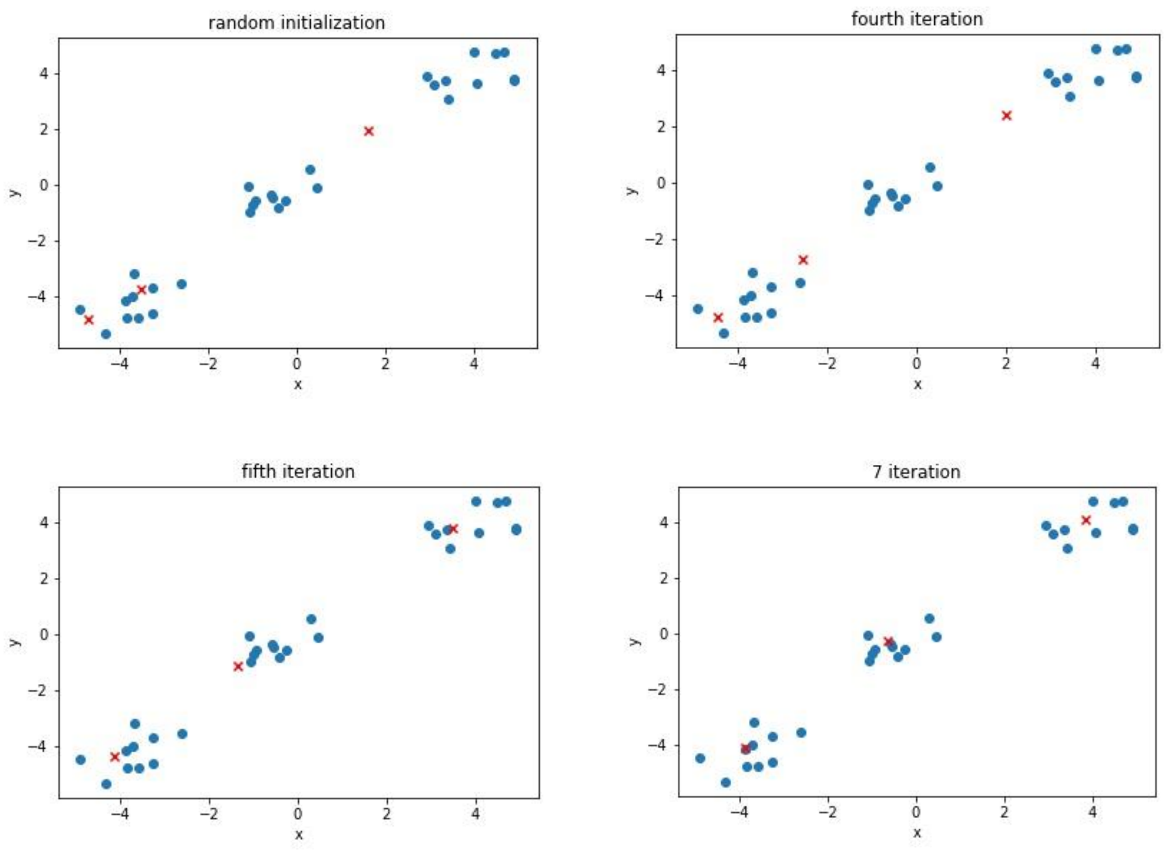

*Figure 5: Three centroids iteratively converging on the three groups.*

While in this example, k-means converged on an optimum solution in relatively few iterations, there are serious issues with Lloyd's method and k-means in general. The run time of most k-means algorithms is O(*nkdi*). Where n is the number of objects being clustered, k in the number of centroids, d is the dimensionality of each of the n objects, and I is the number of iterations run. This is less than ideal for real world problems where there might be millions of objects. In addition to this, determining the optimum solution in Euclidean space is NP-hard (Mahajan et al). Finally, having to determine the number of centroids oneself can represent a significant issue if one is dealing with large datasets where preprocessing is costly and time consuming. As mentioned earlier, there have been many improvements on k-means since its origins. The variations on Lloyd's method are numerous in both number and in the ways, they improve upon Lloyd's method. Some examples of the variation are fuzzy k-means, k-medians, and k-medoids. Fuzzy k-means softens the requirements of k-means and allows observations to fall into more than one cluster. Where k-means assigns an observation to one and only one cluster, fuzzy k-means assigns membership to a cluster as a value in [0,1]. And, k-medians minimizes the $L_1$ norm between observations and clusters. Finally, k-medoids generalizes this to minimize arbitrary norms.

**2.4. DBSCAN Clustering Algorithm:** A newer algorithm, DBSCAN has one immediately obvious advantage over k-means. One does not need to specify the number of clusters oneself. This is handled by the method. Like k-means, DBSCAN is a very capable and relatively well studied algorithm. Furthermore, DBSCAN too is the progenitor of a family of revised and improved clustering algorithms. The intuition behind DBSCAN is simple, areas with high-density are considered clusters and areas of low-density separate clusters. This is achieved using the following algorithm:

1. Define a threshold for the minimum number of points and a radius $\varepsilon$.
2. Start by choosing a random point. If there are more than the minimum threshold for number of points within $\varepsilon$ units of a point, that point is considered a "core point."
3. All points within $\varepsilon$ distance of the core point are part of one cluster, and any other core points within this radius are also considered connected as part of this cluster.

4. Once a cluster has been searches exhaustively, choose a point that has not been evaluated and begin the process again.
5. Once all the points in the set of observations have been parsed, points outside of ε distance away from any core point is discarded as noise.

DBSCAN has a number of advantages over other clustering algorithms. The ability to recognize and dismiss noise as well as the fact that DBSCAN does not require the user to input a parameter specifying the number of clusters. Additional advantages are to follow. DBSCAN is capable of partitioning space into arbitrary surfaces. This allows the algorithm to define clusters which are not linearly separable, something k-means is incapable of (Schubert, 2). DBSCAN is capable of using any metric function as a distance function. Which Lloyd's algorithm was unable to do until to do, promoting the invention of k-medoids. Finally, the nearby point threshold and ε parameter can be tuned to best apply to the data (Shubert, 7).

Of course, DBSCAN is not without its flaws. Poor choice of the input parameters can cause poor performance, this might be unavoidable if the data is opaque and difficult to understand. Following from this, if there are large differences in density over the domain, choosing an advantageous value for the parameters becomes impossible (Shubert, 9).

As for the performance of DBSCAN, since DBSCAN must query each point in the data and calculate the distances between each point, care must be taken to maintain efficiency. If the parameters are chosen in such a way that noise is discarded effectively, then the average runtime of DBSCAN is $O(n\, log\, n)$. This represents a marked improvement over k-means. However, in the worst case, DBSCAN achieves a time complexity of $O(n^2)$. Space complexity is either $O(n)$ or $O(n^2)$ depending on whether a nonmatrix or matrix implementation is used to store distance values, respectively.

## 3. Methods:

The first step in this program is to either import or simulate particle data. If imported the data will be a two-dimensional slice of particle trajectories through time. If simulated, the data will be N Brownian particles evolving every dt seconds for T seconds. In either case, the data for N particles is placed in an array. For the simulation, this array has dimensions N by T/dt. In order to accurately determine the diffusion constants for each fluid, mean squared displacement is calculated for each particle trajectory. Once each particle has a value for mean squared displacement assigned to it, the diffusion constant for that particle is calculated from mean squared displacement. A clustering algorithm is then used to group the diffusion values for each particle into one cluster for each fluid in the mixture. Once the particles are grouped the average diffusion constant for each fluid is calculated in order to best estimate the true diffusion values of the fluids.

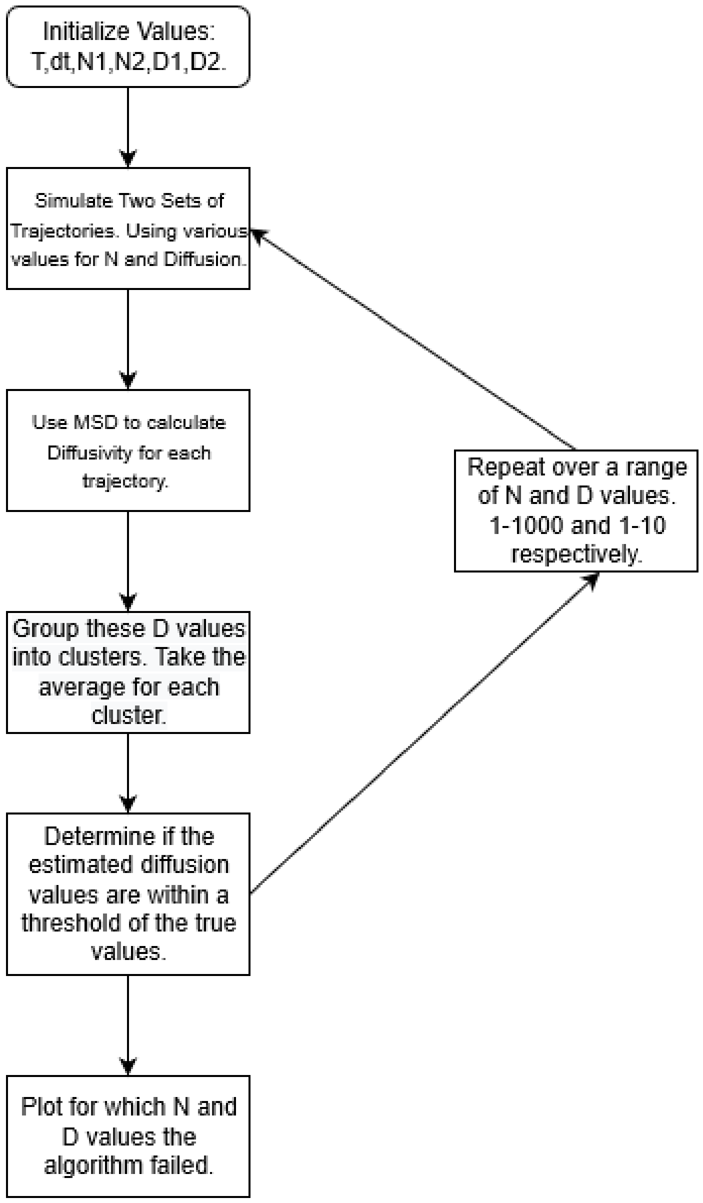
Initialize Values: T,dt,N1,N2,D1,D2.
Simulate Two Sets of Trajectories. Using various values for N and Diffusion.
Use MSD to calculate Diffusivity for each trajectory.
Group these D values into clusters. Take the average for each cluster.
Determine if the estimated diffusion values are within a threshold of the true values.
Plot for which N and D values the algorithm failed.
Repeat over a range of N and D values. 1-1000 and 1-10 respectively.

*Figure 6: top level overview of the process we developed.*

In this study, two sets of particles with different diffusion constants and quantities were used. The diffusion value of the first set of particles is fixed at one unit. While the diffusion value of the second set of particles varies geometrically from 0.1 to 10 units. This allows us to study how the program preforms when the values for sets one and two become very similar. Additionally, the number of particles in sets one and two vary between [1,1000] and [1000,1] in steps of 20 particles through the simulation to assess whether the statistics break down when there are very few of one set of particles. At each pair of values for the number of particles in each set, trajectories over the entire range of diffusion constants are simulated. Allowing us to pinpoint under exactly what conditions each clustering method fails to yield accurate results. An error in the estimated diffusion values is calculated at each set of values for diffusion and N. We then plot these errors such that we can visualize where these statistics fail. Failure is likely to occur at very low N values or when the values for diffusion between the two particles become sufficiently small. Failure here is defined to be when the program incorrectly guesses how many unique species there are in the mixture. Or, when the absolute difference between the true diffusion value for the either particle set and the estimated diffusion value for that set is greater than the tolerance, 0.1 in this study.

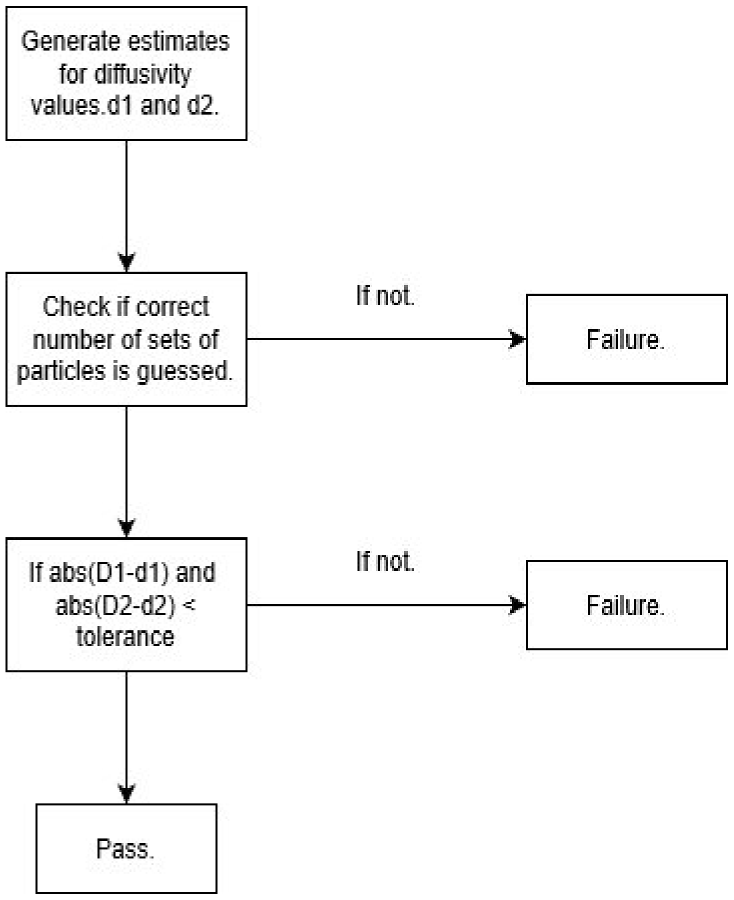
Generate estimates for diffusivity values.d1 and d2.
Check if correct number of sets of particles is guessed.
If not.
Failure.
If abs(D1-d1) and abs(D2-d2) < tolerance
If not.
Failure.
Pass.

*Figure 7: Schematic of the steps which are needed for a run to be considered successful.*

## 5. Results:

Our initial expectation was that our program would see most of its failures result from two possible situations. The first situation we predicted was when one of the particle sets became very small relative to the other. In this case, the cause for failure could be that there would be insufficient data for the diffusivity for that set that our average for diffusivity would lack the information necessary to make a viable estimate for the true diffusivity constant for that set. The other possible point of failure for the case of small set size would be that the small set would be discarded as noise by the clustering algorithm. It was assumed that this would be a particular issue for the DBSCAN method, which is noted for its ability to disregard noise. As a result, DBSCAN may incorrectly estimate fewer sets than there are.

The other problematic situation we foresaw was when the diffusivity values for the two sets become too similar to be separated into distinct groups. In this situation, both clustering algorithms used in the different methods are susceptible to failure. However, given that k-means is equipped with the information that there are two sets, it would seem to be less likely to group different populations together.

We see both of these possible failure situations manifest in our results. The results thus far indicate that both Kmeans and DBscan preform very well when the number of particles in both sets is large and their respective diffusion constants are sufficiently different. It was noted above that DBSCAN may be more susceptible to error when the set sizes become very small and the diffusion values become very similar. Our results indicate that DBSCAN did in fact tend to fail when the diffusivity values became too similar. However DBSCAN was was able to distinguish between the two sets of diffusion values even when one set only contained a single element. This raises the question, considering DBSCANS inherent disregard for noise, why is it capable of detecting a single particle as a second cluster? Conversely, when the difference between the diffusion values for particle sets 1 and 2 become very small (roughly a 0.04-unit difference at the low end), the k-means method did not fail to distinguish the diffusion values.

Both methods did fail in some cases, with k-means outperforming DBSCAN in all trials. Over ten trials, the k-means method returned sufficiently accurate diffusion constants in an average of 99.48% of the simulations for each trial. While DBSCAN was correct 70.19% present of the simulations for each trial. The graphs below show the result of one trial, plotting passing runs in blue and failures in red. The x axis is the ratio of the diffusion values and the y axis is the ratio n1/(n1+n2).

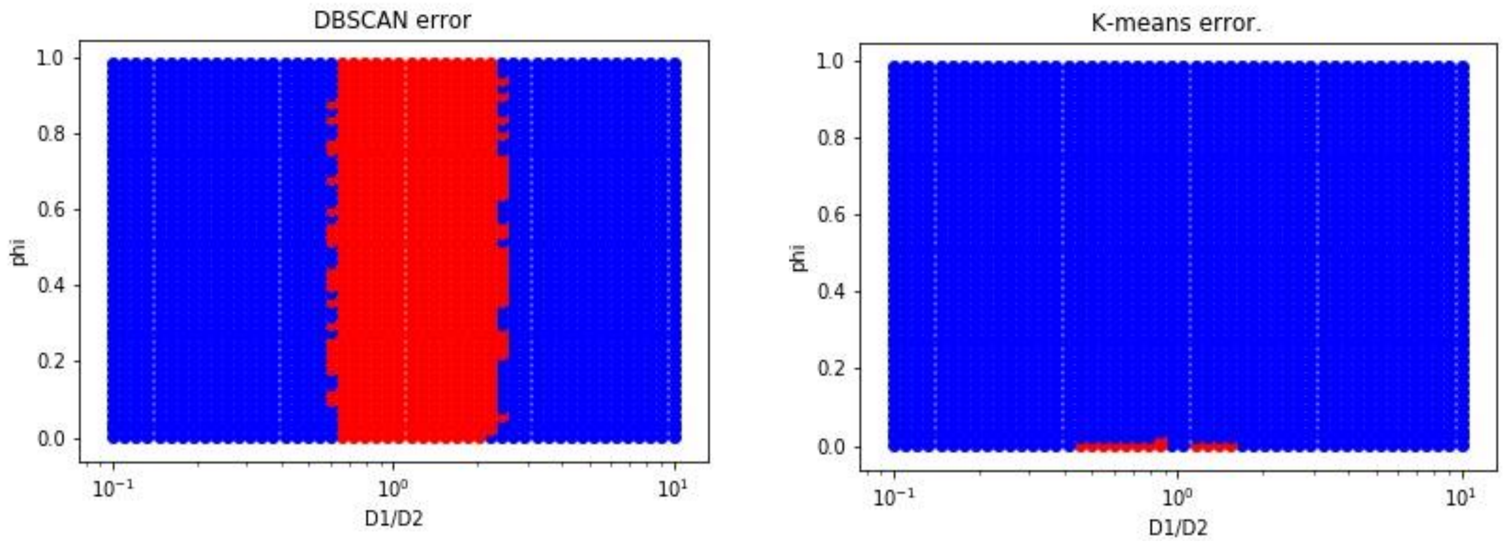

*Figure 8: Plots of the phase space from the simulation. Red dots indicate at which set sizes and diffusivity ratio our method failed. Blue dots indicate where it succeeded.*

We a test of worst-case scenario for our program, where there was only one particle in set one and 999 particles in set 2 and their respective diffusion values being within 0.1 units of each other. This test was run 10,000 times with the simulation running for one second and the particle evolution time step equaling 0.001 seconds on each run. In this situation, DBSCAN passed zero percent of the time and k-means passed 100  percent of the times. In the best-case scenario, where each particle set had a large number of particles (500) and the diffusion values were 1 and 10 respectively, DBSCAN passed 95.18 percent of the time and k-means passed 99.98 percent of the time.

As an extension of our results, one should run the k-means method whenever the number of distinct populations are known, only deferring to DBSCAN when that is not known. A potentially effective method for determining diffusion values for particle populations in a mixture where the number of distinct populations is not known is to first run the DBSCAN method, and then, using the estimate for the number of species in the mixture yielded from the run, use the k-means method to gain a more accurate estimate for the true diffusion values for the populations.

## References:

MacQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967. https://projecteuclid.org/euclid.bsmsp/1200512992

Michalet, X. (2010, October). Mean square displacement analysis of single-particle trajectories with localization error: Brownian motion in an isotropic medium. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3055791/

Bailey, K. (1975). Cluster Analysis. *Sociological Methodology, 6*, 59-128. doi:10.2307/270894

Lavenda, B. (1985). Brownian Motion. *Scientific American, 252*(2), 70-85. Retrieved February 17, 2020, from www.jstor.org/stable/24967570

FÜSTÖS, L., MESZÉNA, G., & MOSOLYGÓ-SIMON, N. (1981). CLUSTER ANALYSIS. *Acta Oeconomica, 26*(3/4), 291-334. Retrieved February 19, 2020, from www.jstor.org/stable/40728857

Mahajan M., Nimbhorkar P., Varadarajan K. (2009) The Planar k-Means Problem is NP-Hard. In: Das S., Uehara R. (eds) WALCOM: Algorithms and Computation. WALCOM 2009. Lecture Notes in Computer Science, vol 5431. Springer, Berlin, Heidelberg

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst. 42, 3, Article 19 (July 2017), 21 pages. DOI:https://doi.org/10.1145/3068335

Shu, X. (2020). CLUSTER ANALYSIS. In *Knowledge Discovery in the Social Sciences: A Data Mining Approach* (pp. 115-132). Oakland, California: University of California Press. doi:10.2307/j.ctvw1d683.8

FÜSTÖS, L., MESZÉNA, G., & MOSOLYGÓ-SIMON, N. (1981). CLUSTER ANALYSIS. *Acta Oeconomica, 26*(3/4), 291-334. Retrieved May 2, 2020, from www.jstor.org/stable/40728857