

Clustering Analysis of Fast Ion Driven Instabilities

John Gresl1  
William Heidbrink1Shaun Haskey2  
Boyd Blackwell3

1*Department of Physics and Astronomy, University of California Irvine, California 92612, USA*2*Princeton Plasma Physics Lab, Princeton University, New Jersey 08540, USA*  
3*Plasma Research Laboratory, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia*

Submitted: xxxxx

# Abstract

Beam ions often drive Alfvén eigenmodes and other instabilities unstable in DIII-D. Many of these modes have been unambiguously identified but some frequently occurring features have been neglected. In this work, datamining and analysis techniques1 that successfully analyzed magnetics data from the H-1NF Heliac are applied to arrays of magnetic and electron cyclotron emission (ECE) data from DIII-D. The clustering techniques group instabilities with similar toroidal magnetic features into clusters of identical mode numbers. Similar analysis is performed on DIII-D’s poloidal magnetic array and ECE probes. Something about ECE results here, later.

# Introduction

Plasma is one of the four fundamental states of matter and makes up an estimated 99% of the matter in the observable universe. A plasma can be thought of as an quasineutral medium of unbound positively and negatively charged particles. These moving charged particles generate local magnetic fields which affect the motion of other nearby particles leading to *collective behavior*, or motion that depends on the physical state of the plasma in local regions.

Harnessing nuclear fusion is one of the prime motivators of studying plasma physics. Nuclear fusion produces more energy per amount of fuel than any other available fuel source. For instance, one gallon of heavy water (water with all of the hydrogen atoms replaced with deuterium atoms) provides 10,000 times more energy when fused than a gallon of gasoline. Deuterium-tritium (D-T) plasmas are known as the most efficient plasma for energy production due to their high mass-to-charge ratio, making it easier to overcome the weak force and fuse together. There are four main reactions that occur in D-T plasmas.

Above, D is a deuterium ion, T is a tritium ion, p is a proton, n is a neutron, and is a nucleus.

Many research plasmas require temperatures on the order of 108 K, making the plasma hot enough to destroy anything it comes into contact with2. For this reason, strong magnetic fields are used to shape and contain the plasma. The introduction of these magnetic fields, while necessary, can lead to some undesirable effects such as unwanted resonances, instabilities, and particles escaping from the plasma and colliding with the inner walls. The study of these interactions is crucial for creating higher quality magnetically confined plasmas.

# Background Physics

Cyclotron Motion

Charged particles trapped in magnetic fields exhibit circular orbits and is known as cyclotron motion. For a non-stationary particle moving at velocity, , with charge, , in a magnetic field, , the force, is equal to:

It is easy to show that there is no work done by this force.

Since is perpendicular to both and , where W is work. The total energy of the particle does not change. We can rewrite velocity in terms of a new basis

where and represent the components of velocity perpendicular and parallel to the magnetic field, respectively. We can then rewrite .

We can say goes to zero since its magnitude, , is equal to 0 since is parallel to by definition.

This force causes charged particles to rotate in circles as they travel along magnetic field lines, all while keeping their parallel velocity constant (see figure 1). Equating the magnetic force to the centripetal force and solving for the radius of curvature yields an expression for the radius of curvature of this gyrating charge known as the Larmor radius, .

From the Larmor radius, one can also determine a parameter known as the cyclotron frequency, :

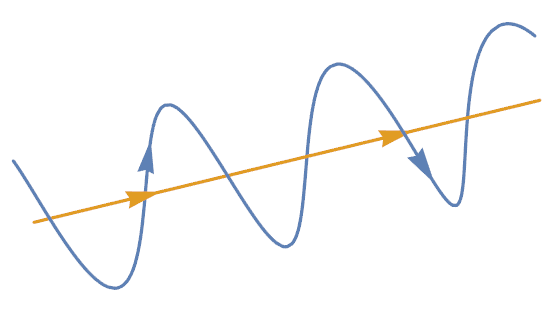


Figure 1. Graphical representation of the path of a charged particle (blue) travelling along a magnetic field line (yellow).

When electrons undergo cyclotron motion, they emit radiation in the form of photons. This is known as electron cyclotron emission (ECE) and is an important diagnostic when studying plasmas.

Alfvén Waves and Eigenmodes

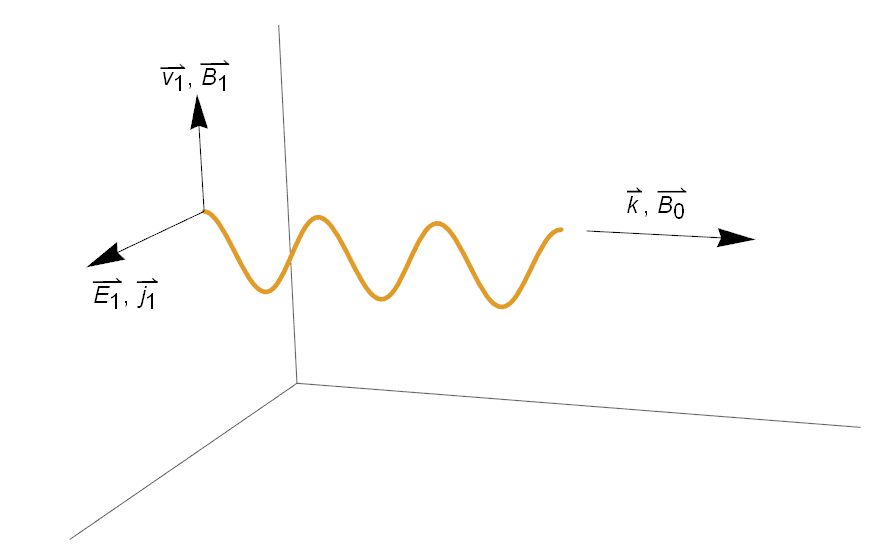


Figure 2. Geometry of an Alfvén wave. The wave has along , and perpendicular to and and perpendicular to both and .

Alfvén waves are low-frequency (relative to ) travelling oscillation that travels along a magnetic field line3. The motion of an Alfvén wave is analogous to a wave travelling along a stretched string. In this analogy, the tension in the magnetic field lines is the same as the tension in the string, and the Alfvén wave is the result of the “plucking” of the string.

From Maxwell’s equations, we can derive the velocity at which these Alfvén waves travel to be . [See appendix A for actual derivation]. The Alfvén velocity refers to the characteristic speed in which perturbations of the lines of the force travel.

Particles within the plasma may interact undesirably with the external magnetic fields creating modes or local instabilities. Many of these instabilities are detrimental to the plasma’s health and can sometimes cause energetic particles to escape the plasma and damage expensive equipment on the inner walls. For our experiment, we are interested in the particles that become trapped, or locked, in the Alfvén waves and refer to them as Alfvén eigenmodes (AE’s).

# The Device and Diagnostics

This experiment is done exclusively with data gathered from the DIII-D tokamak and diagnostics which have already been installed. DIII-D is a torus-shaped plasma device which confines plasma by using strong magnetic fields. The tokamak has a major radius of 1.67 m and a minor radius of 0.67 m. A coil of large electromagnets (B-coils) are positioned like rings around the tokamak to produce toroidal magnetic fields on the order of 2-3 Tesla. Field shaping coils are also used to provide additional stability and confinement to the plasma.

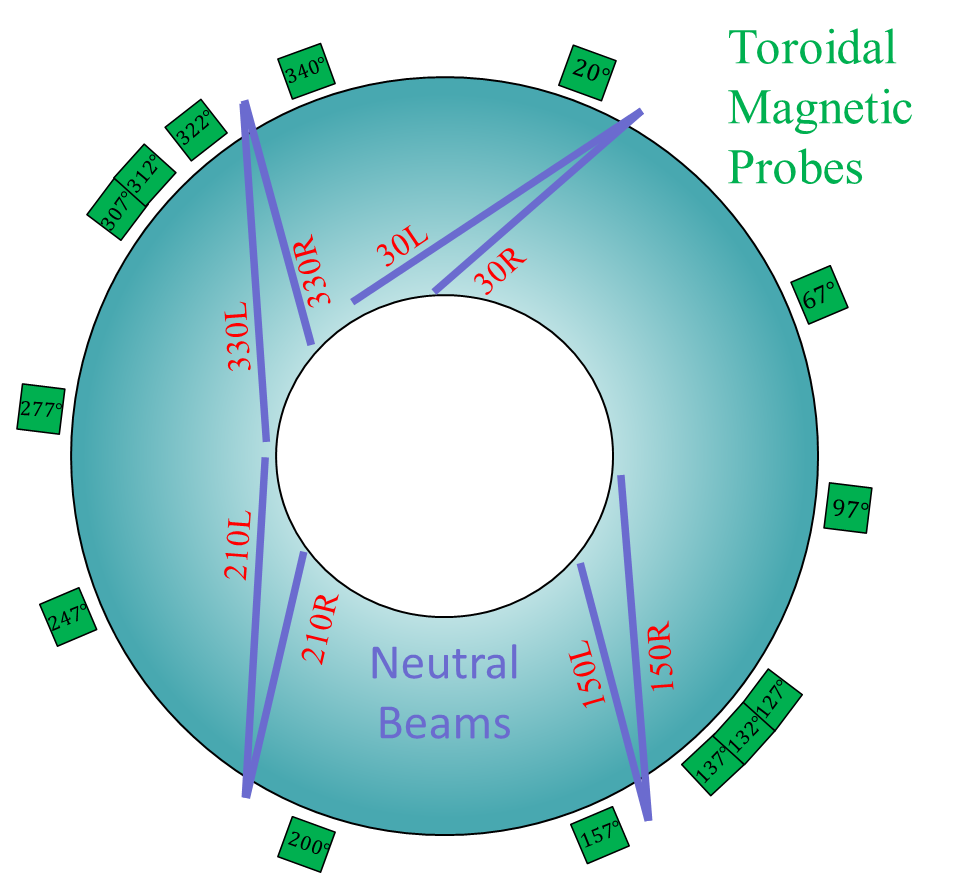


Figure 3. Cartoon of top down view of DIII-D. The 4-pairs of neutral beam injectors are shown in purple and the various toroidal magnetic Mirnov probes are shown in green with their angular position indicated.

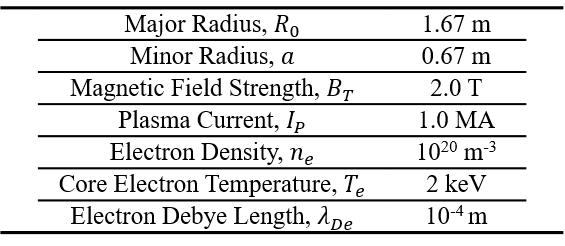
DIII-D also utilizes 4 pairs of neutral beam injectors which function as an auxiliary energy source for the plasma. These neutral beams supply high energy neutral particles to the plasma where they can interact with other particles by collisions. The orientation of these neutral beam injectors can be seen in figure 4. One capability which sets the neutral beams at DIII-D apart from others is its ability to run their beams at a very high confidence over the range 40 kV to 75 kV *without* the need for special calorimetry or test shots4.

Magnetic Mirnov Probes

Among the many diagnostics at DIII-D, the magnetic Mirnov probes are one of the most important. The probes consist of coils of wire which measure the changing magnetic field caused by moving charges in the plasma. We are interested in the main toroidal magnetic Mirnov array and the 322-degree poloidal magnetic Mirnov array. We will refer to these arrays as the *toroidal array* and the *poloidal array*, respectively. The toroidal array consists of 14 probes and lies on the

toroidal midplane of the tokamak. The arrangement of the 14 probes in the toroidal array can be seen in figure 3, along with the orientation of the neutral beam injectors. The poloidal array consists

Table 1. Typical plasma parameters at DIII-D.



of 31 (verify) probes and encircles the tokamak poloidally at the 322-degree mark.

Electron Cyclotron Emission (ECE)

The ECE array at DIII-D consists of a 40-channel heterodyne radiometer that provides the electron temperature at a given radius and time. Fluctuations within the plasma can cause the cyclotron frequencies of the electrons, , to vary in time and space. Alfvén eigenmodes frequently affect the mode structure and can cause fluctuations in which make the ECE arrays a helpful tool for studying Alfvén eigenmodes.

Instabilities and Alfvén Eigenmodes in the DIII-D Tokamak

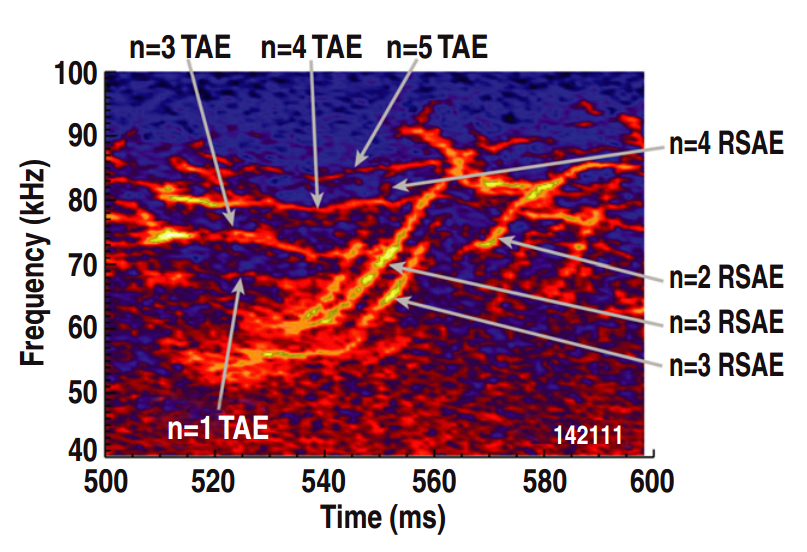


Figure 4. Frequency vs. time spectrogram for shot 142111 with different Alfvén eigenmodes identified. Frequency peaks in the fourier transform are shown in bright yellow. Figure taken from9.

Instabilities can manifest within DIII-D’s plasma in the form of resonances or Alfvén eigenmodes. These instabilities are studied using magnetohydrodynamics (MHD) and treating the plasma as a fluid. Alfvén eigenmodes can be very detrimental to confinement5 and it has even been found that particles trapped in these Alfvén eigenmodes reduce the beam power6, making it harder to achieve ignition. The study of these Alfvén eigenmodes and how to suppress their detrimental effects is of great importance for larger tokamaks in the future, such as ITER.

The three Alfvén eigenmodes we are interested in are toroidal Alfvén eigenmodes (TAE’s), reversed shear Alfvén eigenmodes (RSAE’s), and Beta Induced Alfvén eigenmodes (BAE’s). These different Alfvén eigenmodes are identified by analyzing a frequency vs. time spectrogram.

# Data Analysis

Clustering Analysis

Clustering is the act of sorting objects into groups based on user determined clustering parameters. Performing clustering analysis on bulky data sets is an effective method of uncovering an underlying structure. One common clustering algorithm, known as *k-means* uses distance between two points as the clustering parameter and groups objects together based on how far apart they are. A general k-means algorithm behaves as follows: let’s say you have n-data points in a 2-dimensional space that you would like to separate into k-clusters. The algorithm begins by selecting k-cluster centers (or *means*) and then grouping each data point to the mean that it is closest too. Once each data point has been classified, the mean of each group is calculated and all the data points are regrouped based on this new set of means. This process of grouping, calculating the mean, and regrouping will iterate a set amount of times, or until a convergence criteria is reached. A more in-depth example of this process is given in Appendix B using the python programming language.

The three clustering algorithms used in this experiment were k-means, expectation maximization with gaussian mixture models (EM-GMM), and expectation maximization with von-mises mixtures (EM-VMM). EM-GMM is a clustering algorithm which uses a gaussian probability distribution function to determine the likelihood that a data point belongs to a cluster. K-means and EM-GMM are fantastic clustering algorithms due to their simplicity and robust design, however, they will both fail to adequately form clusters within periodic data (make a plot to illustrate this if time). When attempting to cluster periodic data such as the phases of certain waves, we must consider that a phase of - is equal to a phase of + and form our clusters accordingly. This can be done with a few simple modifications to the EM-GMM probability distribution function, transforming it into a von-mises probability distribution function:

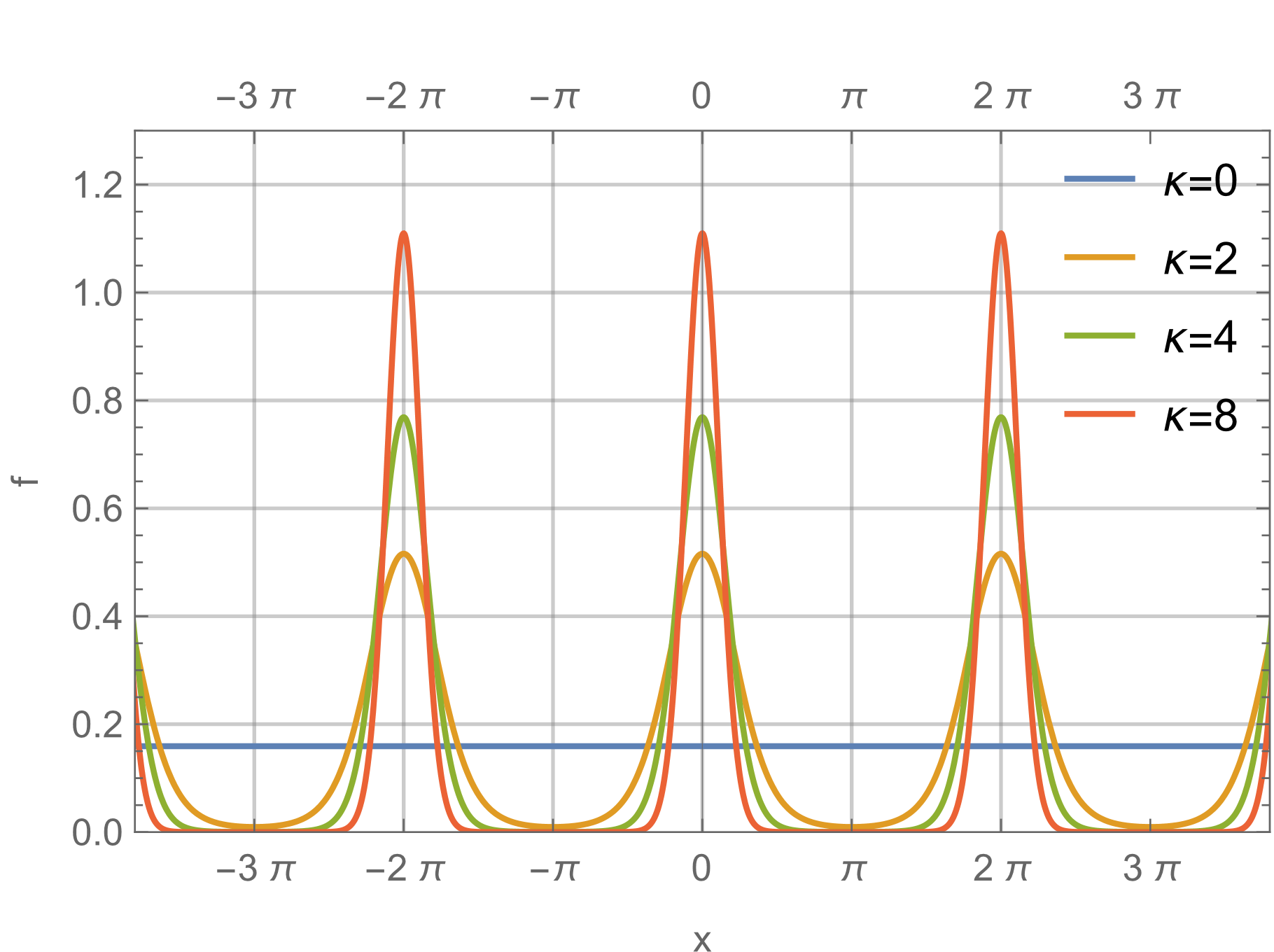


Figure 5. Example of a von-mises distribution function with the mean, , centered at 0 with several different values. set equal to 0 represents a uniform distribution function.

where is position of the data point, is the mean or cluster center, is a parameter related to the density of the cluster, and is a modified Bessel function of the first kind with order 0.

Data

For this project, we decided to examine DIII-D shots numbered 159243 through 159257 each with a time window from 300ms-1400ms. Timeseries probe data is collected and then segmented into very small overlapping time chunks where we then perform short time fourier transform (STFT) analysis. The STFT spectra is averaged for each time chunk and the highest peaks are selected from the spectra. High peaks within the spectra signify a prominent frequency at that time chunk.

The phase difference between each probe and a fixed reference probe is calculated and is saved in a phase difference array (PDA). Clustering analysis is then performed on the PDA. We begin with a k-means algorithm due to its ability to quickly locate approximate cluster centers. Once the approximate cluster centers are located, we switch over to an EM-VMM algorithm until we reach reasonable convergence. The low-density clusters identified by the EM-VMM algorithm are filtered out based on their value from the von-mises probability distribution function and analysis continues to the next shot. Because the analysis of each shot is independent of each other, we can utilize multiple processing threads simultaneously to dramatically decrease the amount of time required for each analysis run.

After each shot is analyzed, clustered, and filtered, we perform the clustering algorithms again on *all* the shots at once. A typical run of the analysis algorithm clustering every shot at once (without first clustering and filtering individual shots) can take upwards of *80* GB of memory! By going through the trouble of clustering, un-clustering, and then re-clustering, we can bring down the memory usage to a much more modest 20-40 GB. In summary, talk about reducing memory footprint of program and think of suggestions that may help.

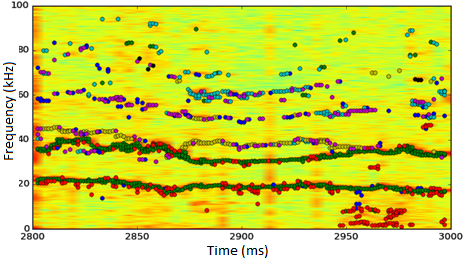
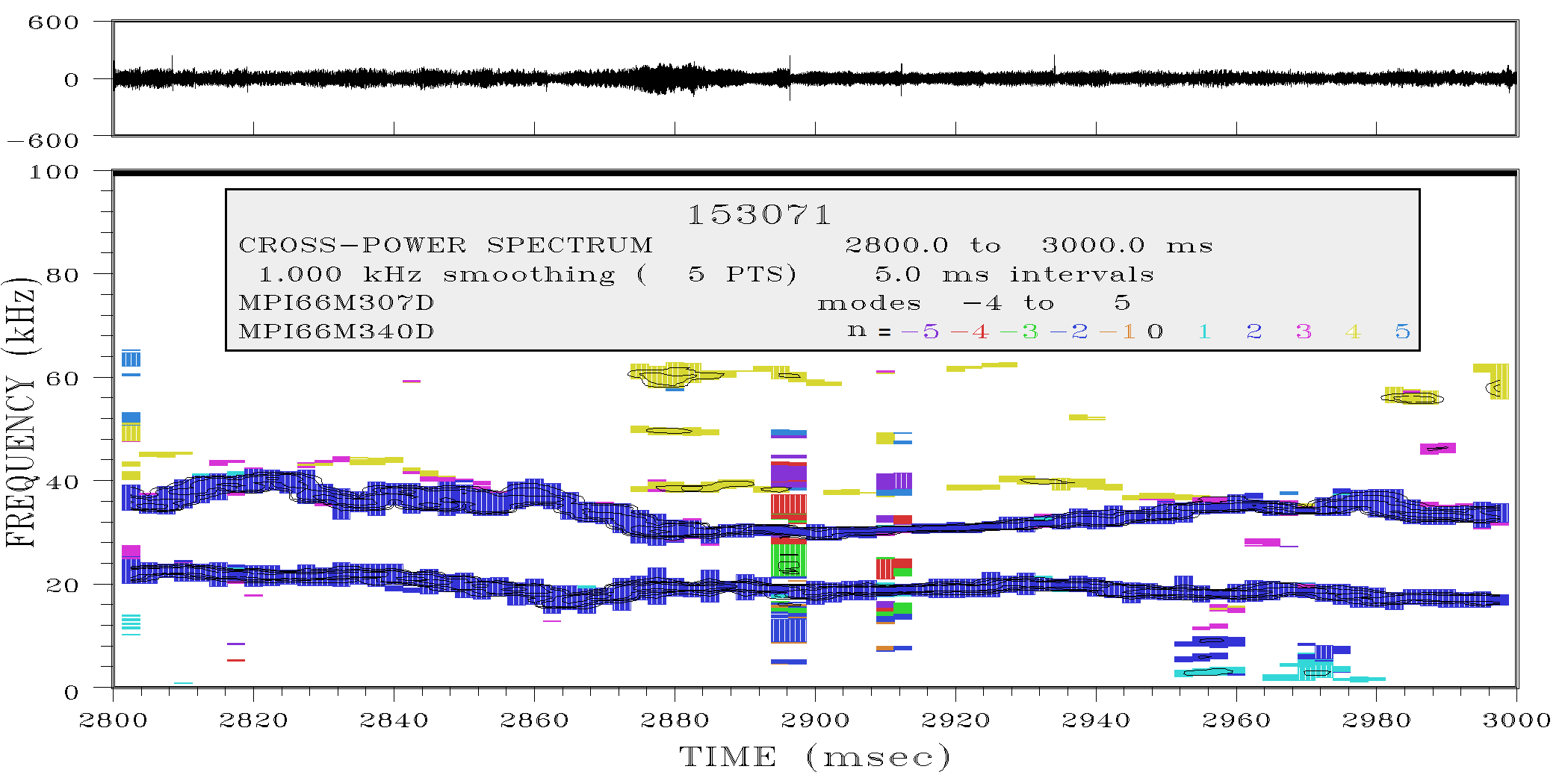


Figure 6. Comparison with our clustering algorithm (top) with a different mode identifying algorithm (bottom). Top: Different clusters plotted in different colors. Bottom: Different toroidal mode numbers plotted in different colors.

When developing a new method, it is in good practice to routinely test the method on a sample set of data where the outcome is already known. This becomes increasingly more important as the complexity of your problem increases. For this reason, we analyzed a series of shots (table 2) using our clustering algorithm. This series of shots has already been extensively studied7,8 because of the distinct n=2 neoclassical tearing mode (NTM) throughout the entire 200 ms time window. Here, n refers to the toroidal mode number.

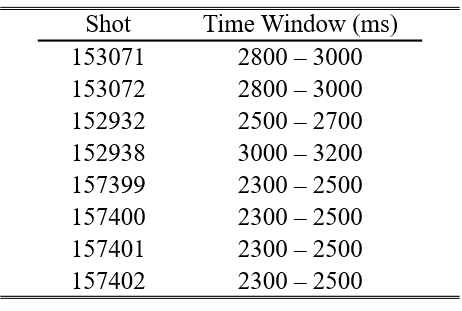


Table 2. Shot list and time windows used for the verification process. Each shot has a 200 ms time window which we are interested in.

**Bibliography**

1. Haskey, S. R., Blackwell, B. D. & Pretty, D. G. Clustering of periodic multichannel timeseries data with application to plasma fluctuations. *Comput. Phys. Commun.* **185,** 1669–1680 (2014).

2. Heidbrink, W. W. Basic physics of Alfvn instabilities driven by energetic particles in toroidally confined plasmas. *Phys. Plasmas* **15,** (2008).

3. Chen, F. F. *Introduction to Plasma Physics and Controlled Fusion*. (Springer, 1984).

4. DIII-D Beams. (2016). Available at: https://diii-d.gat.com/diii-d/Beams\_feat. (Accessed: 16th May 2017)

5. Ogawa, K. *et al.* A study on the TAE-induced fast-ion loss process in LHD. *Nucl. Fusion* **53,** 53012 (2013).

6. Heidbrink, W. W., Strait, E. J., Doyle, E., Sager, G. & Snider, R. T. An investigation of beam driven Alfvén instabilities in the DIII-D tokamak. *Nucl. Fusion* **31,** 1635–1648 (2011).

7. Popov, A. M. *et al.* Simulation of neoclassical tearing modes (NTMs) in the DIII-D tokamak. I. NTM excitation. *Phys. Plasmas* **9,** 4205 (2002).

8. Prater, R. *et al.* Stabilization and prevention of the 2/1 neoclassical tearing mode for improved performance in {DIII-D}. *Nucl. Fusion* **47,** 371 (2007).

9. Todo, Y., Zeeland, M. A. Van & Heidbrink, W. W. Fast ion profile stiffness due to the resonance overlap of multiple Alfvén eigenmodes. *Nucl. Fusion* **56,** 112008 (2016).

# Appendix

A. Derivation of Alfvén Velocity

Refer to figure 2 for definitions of variables and terms. From Maxwell’s equations, we have

|  |  |  |
| --- | --- | --- |
|  |  | [] |

Since is only oriented in the  direction and is only oriented in the direction, only the x component if nontrivial. Our equation becomes

|  |  |  |
| --- | --- | --- |
|  |  | [] |

The thermal motion of the particles is not important, so we can use the solution to the ion equation of motion, [4], with equal to zero.

|  |  |  |
| --- | --- | --- |
|  |  | [] |

Which separates into

|  |  |  |
| --- | --- | --- |
|  |  | []  [] |

Solving for explicitly in [6] yields

|  |  |  |
| --- | --- | --- |
|  |  | [] |

We can plug this into [5] and solve for

|  |  |  |
| --- | --- | --- |
|  |  | [] |

Which is the velocity of the ions in the x direction. To find the electron velocity, we consider the transformation where and the limit where

|  |  |  |
| --- | --- | --- |
|  |  | [] |

This term goes to zero since the ; the Larmor gyrations are negligible so the electrons only have an drift in the y direction. Substituting [8] and [9] into [3] yields

|  |  |  |
| --- | --- | --- |
|  |  | []  [] |

where is the square of the ion plasma frequency.

Now, we take into consideration the fact that Alfvén wave frequencies are much below the ion cyclotron frequency, . In this limit, [11] becomes

|  |  |  |
| --- | --- | --- |
|  |  | [] |

|  |  |  |
| --- | --- | --- |
|  |  | [] |

where is the mass density. Recall that and [13] becomes

|  |  |  |
| --- | --- | --- |
|  |  | [] |

The denominator of [14] can be recognized as the relative dielectric constant for low frequency perpendicular motions, .

|  |  |  |
| --- | --- | --- |
|  |  | [] |

is much larger than one for most laboratory plasmas3, so [14] can be approximated as

|  |  |  |
| --- | --- | --- |
|  |  | [] |

And we have arrived at our result for the Alfvén velocity.

|  |  |  |
| --- | --- | --- |
|  |  | [] |

B. K-Means Clustering Example in Python

This code example may be found in my GitHub repository at:  
<https://github.com/SyntaxVoid/PyFusionDIIID/blob/master/Writing/ClusteringExample.py>

**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**import** random  
  
  
*## 0. Create a Point class and some usefull functions***class Point:** *## A point is created by calling Point(x,y)* **def** \_\_init\_\_(self, *x*, *y*)**:** self.x **=** *x* self.y **=** *y* **return  
  
 def plot\_self**(self, *axes*, *marker*, *color*)**:** *## Plots itself on a given matplotlib axes with a certain color  
 axes*.plot(self.x, self.y, marker**=***marker*, color**=***color*)  
  
  
**def distance**(*point1*, *point2*)**:** *## Returns the distance between point1 and point2* **return** np.sqrt((*point2*.x **-** *point1*.x) **\*\*** 2 **+** (*point2*.y **-** *point1*.y) **\*\*** 2)  
  
  
**def random\_point**(*a*, *b*)**:** *## Returns a point object with x,y values between a and b.* **return** Point(random.uniform(*a*, *b*), random.uniform(*a*, *b*))  
  
  
**def cluster\_mean**(*cluster*)**:** *## Returns the mean value of a cluster of points* x\_sum **=** 0  
 y\_sum **=** 0  
 **for** point **in** *cluster***:** x\_sum **+=** point.x *## a += b represents a = a + b* y\_sum **+=** point.y  
 x\_avg **=** x\_sum **/** len(*cluster*)  
 y\_avg **=** y\_sum **/** len(*cluster*)  
 **return** Point(x\_avg, y\_avg)  
  
  
*## 1. Create a random set of 1000 data points*n\_points **=** 1000  
points **=** []  
**for** i **in** range(n\_points)**:** points.append(  
 random\_point(2, 8)) *# This will add a random point with x,y values between 2 and 8 to our array of points  
  
## 2. Generate our first set of 7 random cluster centers*n\_clusters **=** 7  
means **=** []  
**for** cluster\_id **in** range(n\_clusters)**:** means.append(random\_point(2, 8))  
  
*## 3. Perform clustering*n\_iterations **=** 50  
**for** i **in** range(n\_iterations)**:** *## Iterate the process 50 times* clusters **=** [] *## Clear our clusters before each iteration* **for** cluster\_id **in** range(n\_clusters)**:** clusters.append([]) *## Create an empty cluster for as many clusters as we have* **for** point **in** points**:** *## Iterate over every data point* dists **=** []  
 **for** mean **in** means**:** dists.append(distance(point, mean)) *## Calculates the distance from the current point to each mean* closest\_mean\_index **=** dists.index(min(dists)) *## Returns the index of the smallest distance* clusters[closest\_mean\_index].append(point) *## Add the point to the appropriate cluster* **for** cluster\_id **in** range(n\_clusters)**:** means[cluster\_id] **=** cluster\_mean(clusters[cluster\_id]) *## Calculate new mean for each cluster  
  
## 4. Plot your clusters*colors **=** ["blue", "orange", "red", "green", "black", "yellow", "cyan"] *## Add more colors here for each cluster you have*figure, axes **=** plt.subplots(1,1) *## Create our figure and axes objects***for** cluster\_id **in** range(n\_clusters)**:** *## Iterate over each cluster* **for** point **in** clusters[cluster\_id]**:** *## Iterate over each point in the cluster* point.plot\_self(axes**=**axes, marker**=**"o", color**=**colors[cluster\_id])  
 mean **=** means[cluster\_id] *## Next few lines plot circles around each cluster* mean\_radius **=** max([distance(point, mean) **for** point **in** clusters[cluster\_id]])  
 axes.add\_artist(plt.Circle((mean.x, mean.y), mean\_radius, color**=**colors[cluster\_id], alpha**=**0.25))  
  
plt.xlabel("X"); plt.ylabel("Y")  
plt.title("Clustering Example", fontsize=20); plt.grid(); plt.show()

Once the above code has been run, it should produce a figure like the one below. Notice that the clusters intersect each other and some points appear to lay in more than one cluster. This is simply an artifact of our choice to draw the cluster circles based on the most distant point from the cluster mean. Also, our input data was inherently random so one should not expect to see dense clusters.

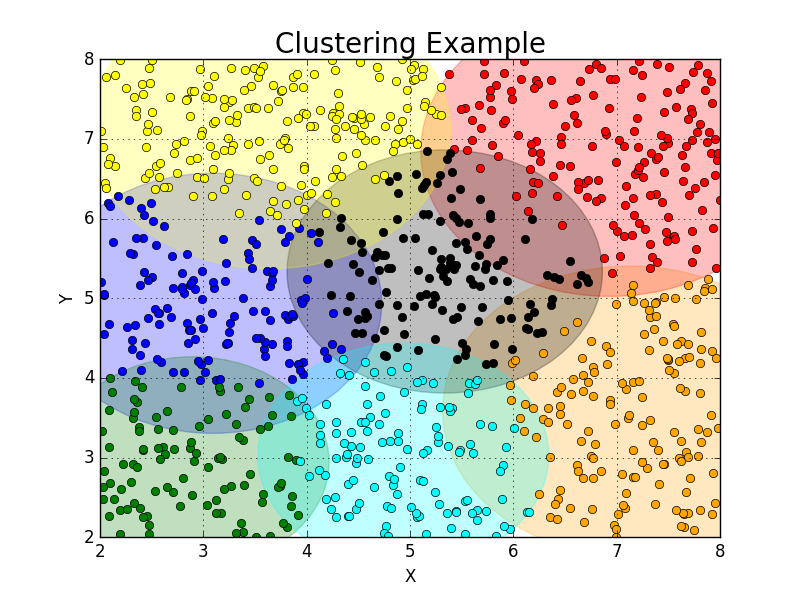


Figure 5. Sample output from ClusteringExample.py using 1000 random data points and a k-means algorithm with 7 total clusters. Clusters plotted in different clusters with a large circle indicating the radius of the most-distant point.