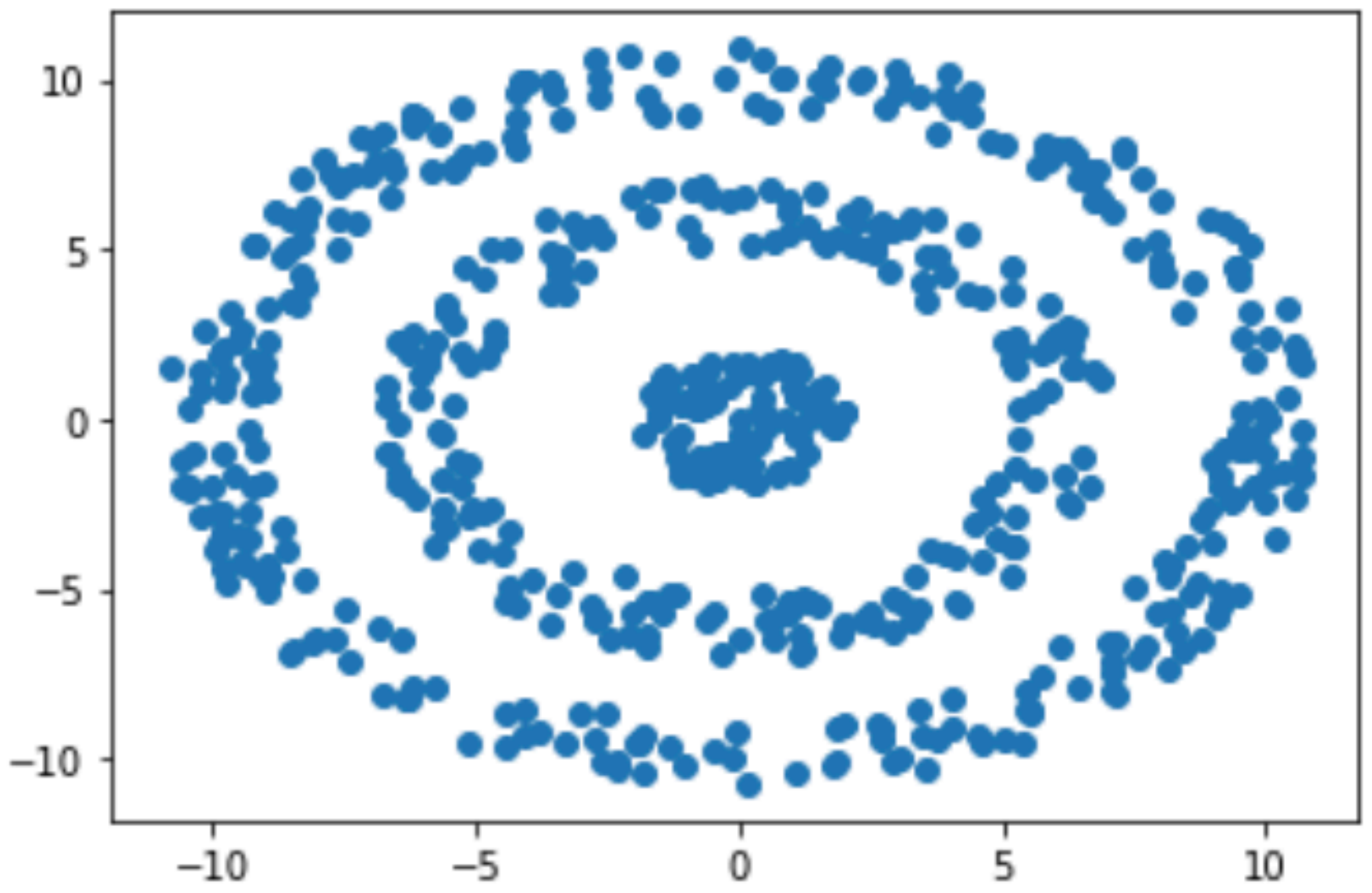
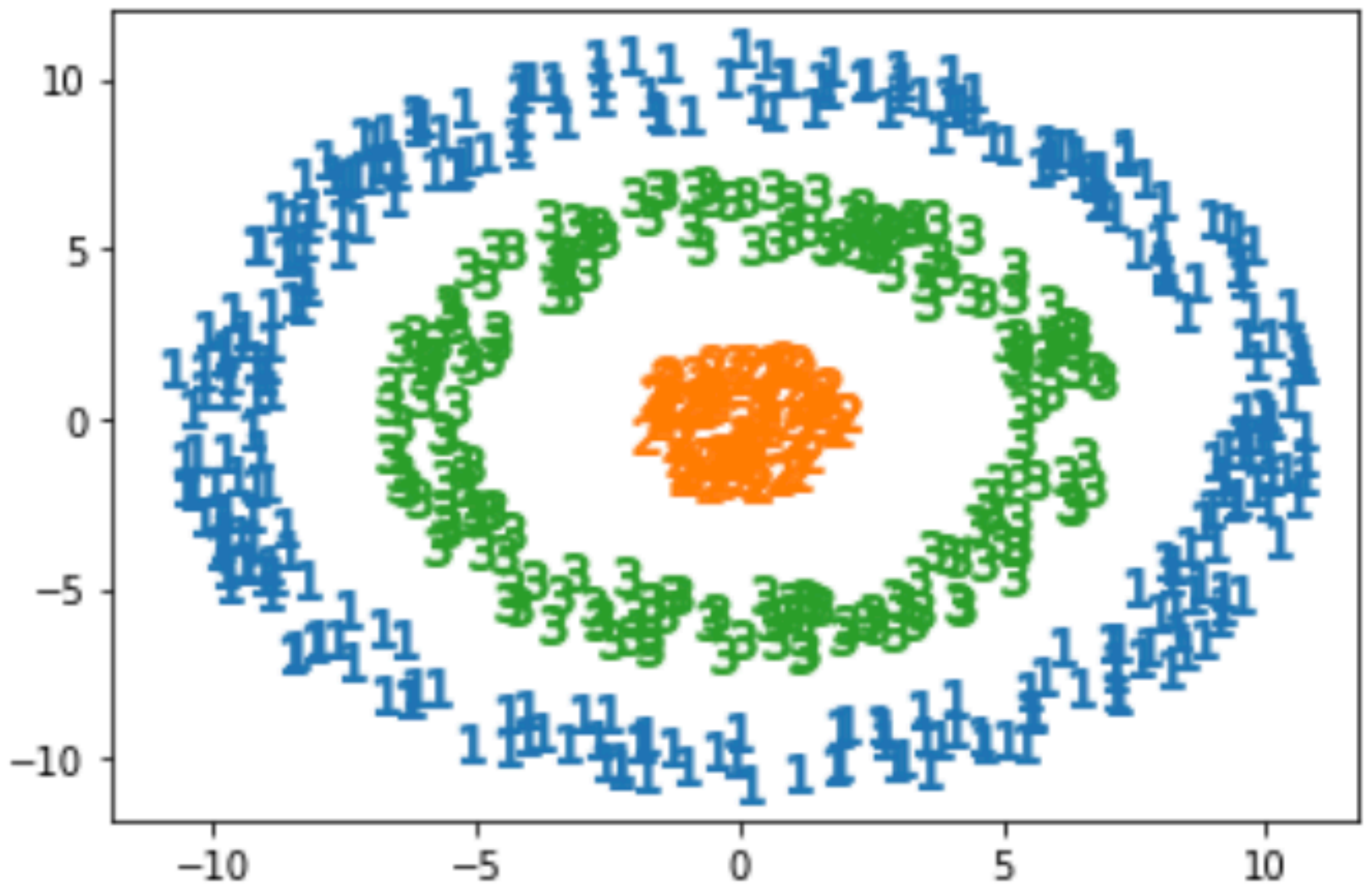
Spectral Clustering Method with Physics

DSECOP Fellow
Cunwei Fan
University of Illinois Urbana Champaign

Introduction

- Data science mostly consists of mathematics (linear algebra) and statistics
- Classical data science techniques are more similar to physics
- An example: Spectral Clustering





Overview

- Summary of the Module
- Going through the Module

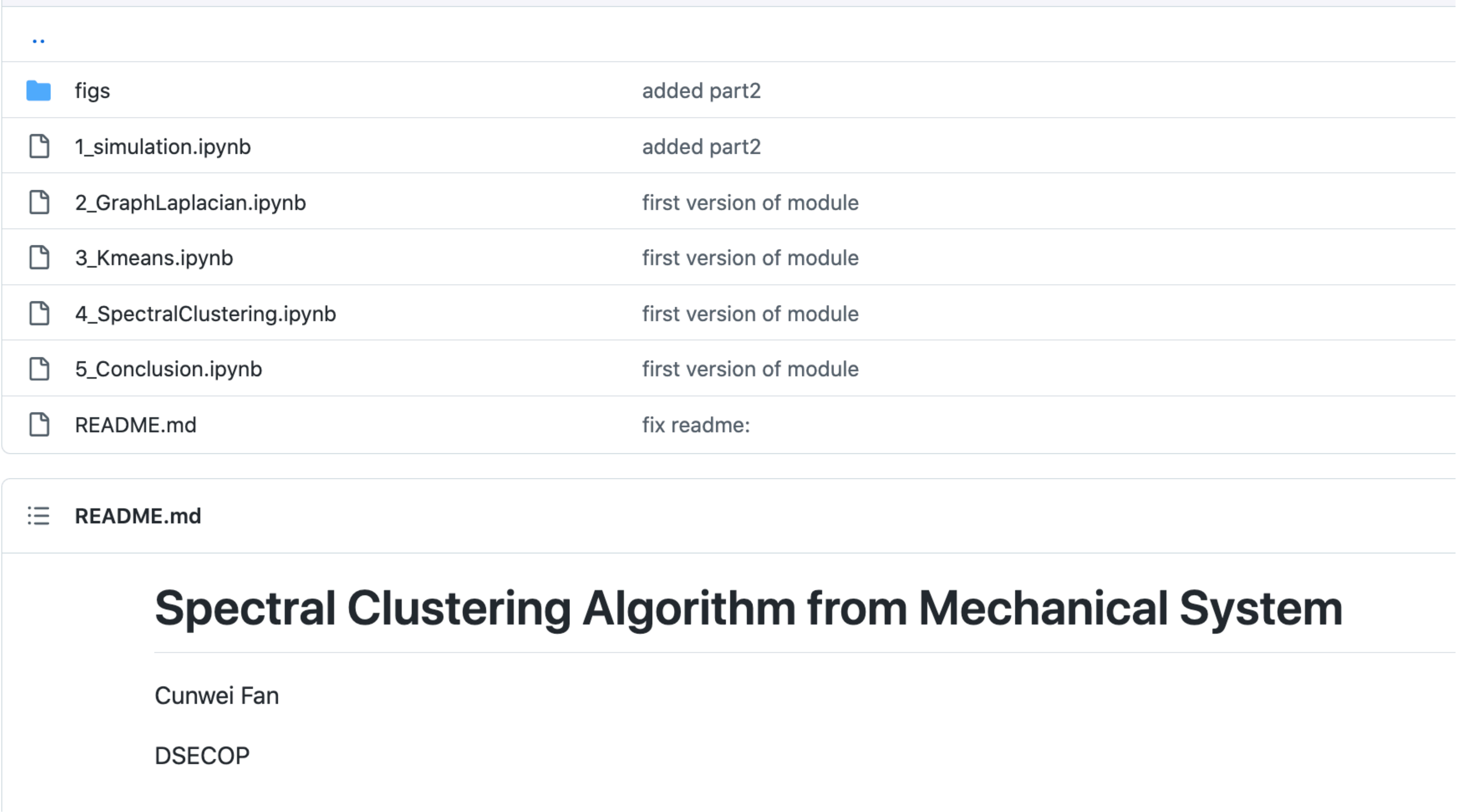
  Part 1: Coupled Oscillators

  Part 2: From K matrix to L

  Part 3: K-means

  Part 4: Spectral Clustering

  Part 5: Standard Package



..	
figs	added part2
1_simulation.ipynb	added part2
2_GraphLaplacian.ipynb	first version of module
3_Kmeans.ipynb	first version of module
4_SpectralClustering.ipynb	first version of module
5_Conclusion.ipynb	first version of module
README.md	fix readme:

README.md

Spectral Clustering Algorithm from Mechanical System

Cunwei Fan

DSECOP

Structure of the Module

- The module has 5 parts (suggested duration: 100 mins or 2 lectures)
- Can be a supplemental lecture for classical mechanics course
- Can serve as a good final project
- In each part, there are a couple of homework problems
- One quiz question and no exam

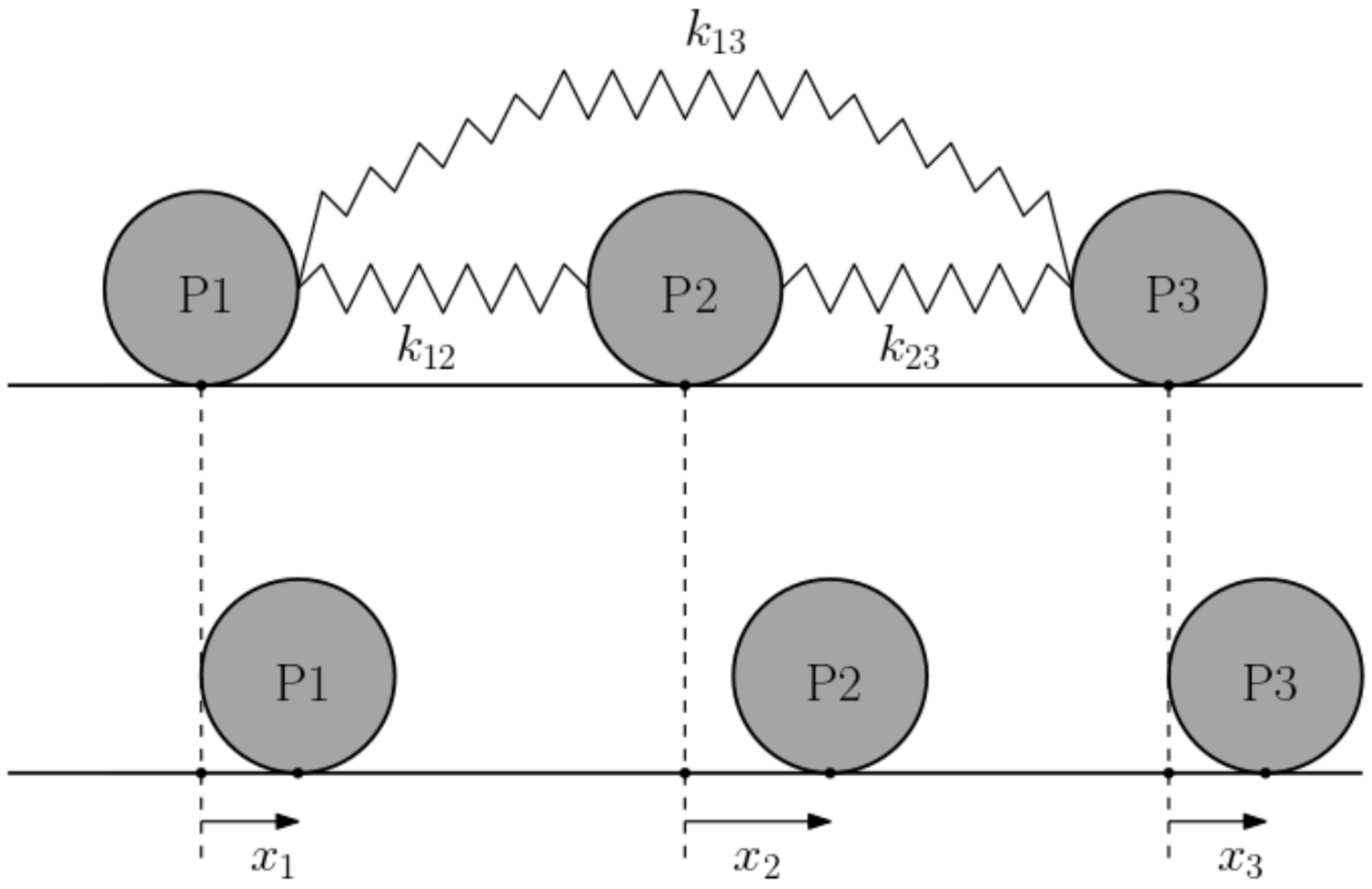
Features of Module

- Introduces the spectral clustering method in data science using familiar physics problem
- Spectral clustering is an important non-linear techniques in data mining
- This module does not require heavy coding
- Familiar with basic python script (numpy, matplotlib)
- Basic knowledge of coupled oscillators and ODEs are required

Part 1: Coupled Oscillators

- First introduce the physics problem
- Construct the equation of motion

$$
\begin{aligned}
m\frac{d^2x_1}{dt^2} &= -(k_{12}+k_{13})x_1 + k_{12}x_2 + k_{13}x_3 \\
m\frac{d^2x_2}{dt^2} &= -(k_{21}+k_{23})x_2 + k_{21}x_1 + k_{23}x_3 \\
m\frac{d^2x_3}{dt^2} &= -(k_{31}+k_{32})x_3 + k_{31}x_1 + k_{32}x_2
\end{aligned}
$$



- Give example code to solve the coupled ODEs

Part 1: Continued

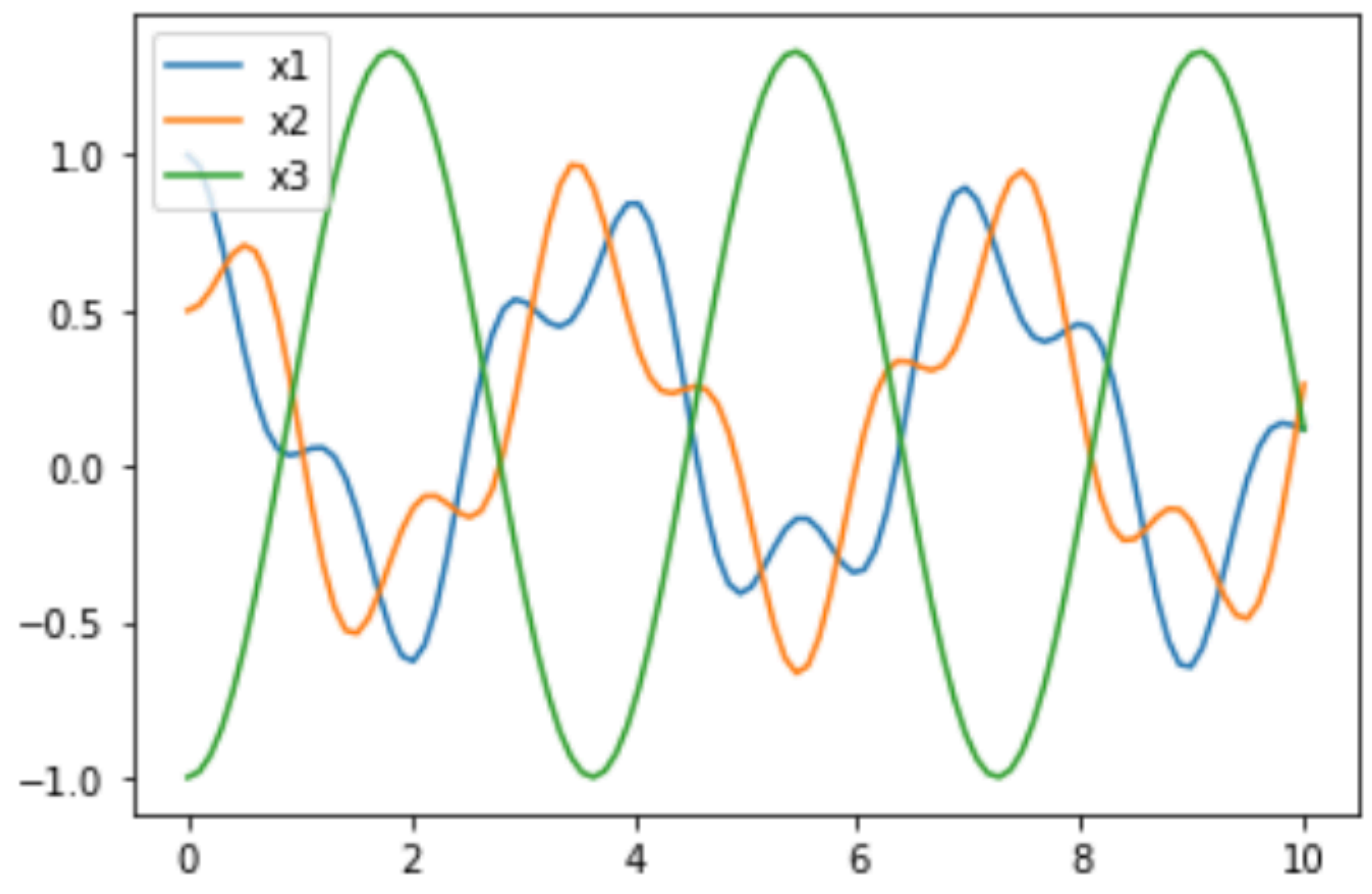
- Solve the ODEs numerically:
  - Detailed guidance on solving the equations using python.
- Play with different initial condition : let the students to realize: strongly coupled particles tend to move together.

$$\frac{dx_1}{dt} = v_1 dt$$
$$\frac{dx_2}{dt} = v_2 dt$$
$$\frac{dx_3}{dt} = v_3 dt$$
$$m\frac{dv_1}{dt} = -(k_{12}+k_{13})x_1 + k_{12}x_2 + k_{13}x_3$$
$$m\frac{dv_2}{dt} = -(k_{21}+k_{23})x_2 + k_{21}x_1 + k_{23}x_3$$
$$m\frac{dv_3}{dt} = -(k_{31}+k_{32})x_3 + k_{31}x_1 + k_{32}x_2$$

```
def produce_dv_dt(k):
    def dv_dt(t, xv):
        x1, x2, x3 = xv[0], xv[1], xv[2]

        dv1 = - (k[0,1]+ k[0,2])*x1 + k[0,1] *x2 + k[0,2]*x3
        dv2 = - (k[1,0]+ k[1,2])*x2 + k[1,0] *x1 + k[1,2]*x3
        dv3 = - (k[2,0]+ k[2,1])*x3 + k[2,0] *x1 + k[2,1]*x2

        return np.array([dv1, dv2, dv3])
    return dv_dt
```



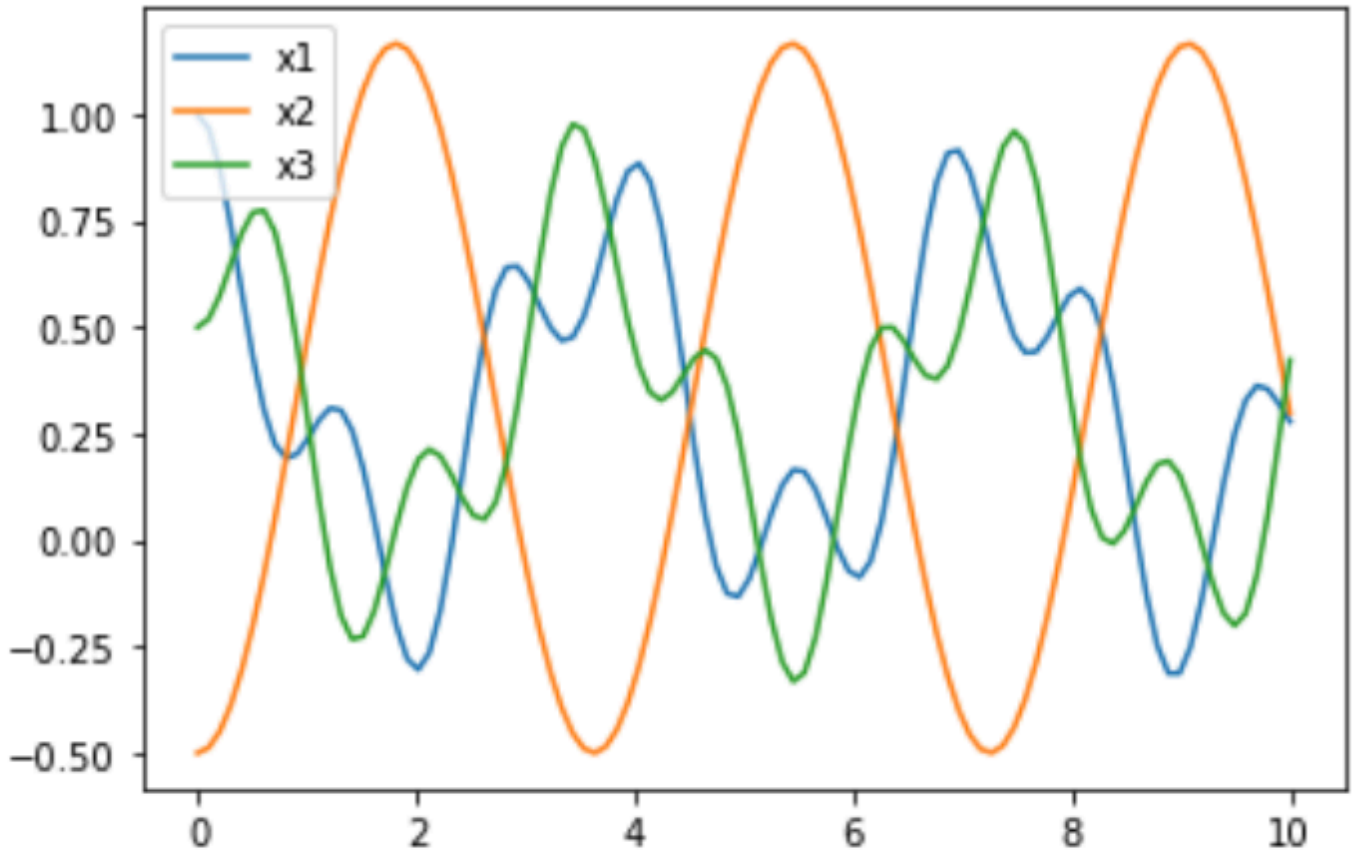
Part 2: Graph Laplacian

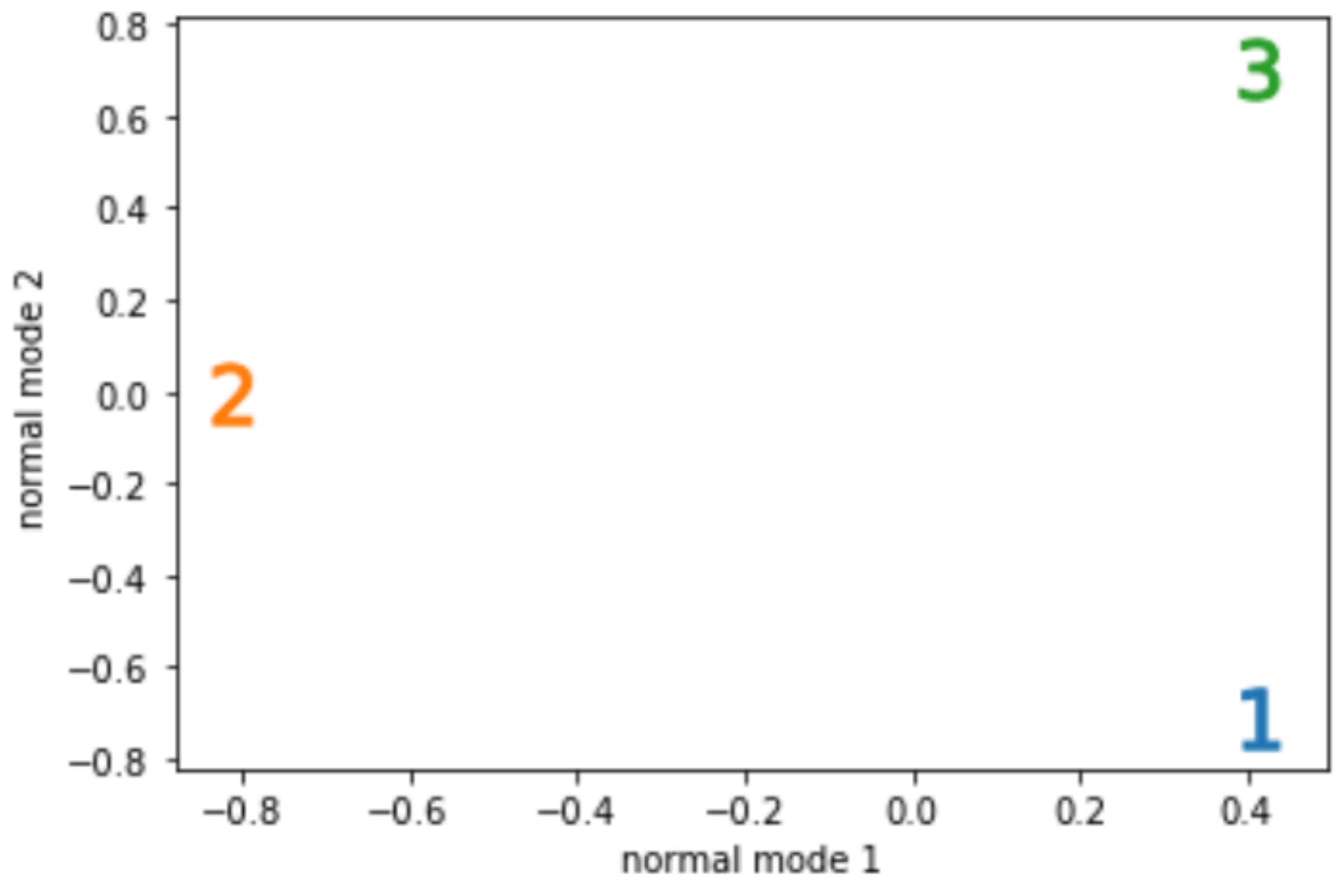
- Rewrite the equations and produce eigenvalue problems

$$\frac{d^2}{dt^2}\mathbf{x} = \begin{pmatrix} -(k_{12}+k_{13}) & k_{12} & k_{13} \\ k_{21} & -(k_{21}+k_{23}) & k_{23} \\ k_{31} & k_{32} & -(k_{31}+k_{32}) \end{pmatrix}\mathbf{x} \qquad \frac{d^2}{dt^2}\mathbf{x} = -L\mathbf{x} \qquad \omega^2 A = LA.$$

- Observed: strong coupled pairs oscillate together in lower energy mode:
  - Naturally: In first few eigenvectors, strongly coupled pairs have similar components
  - Let the student to realize first few eigenvectors can encode similarities

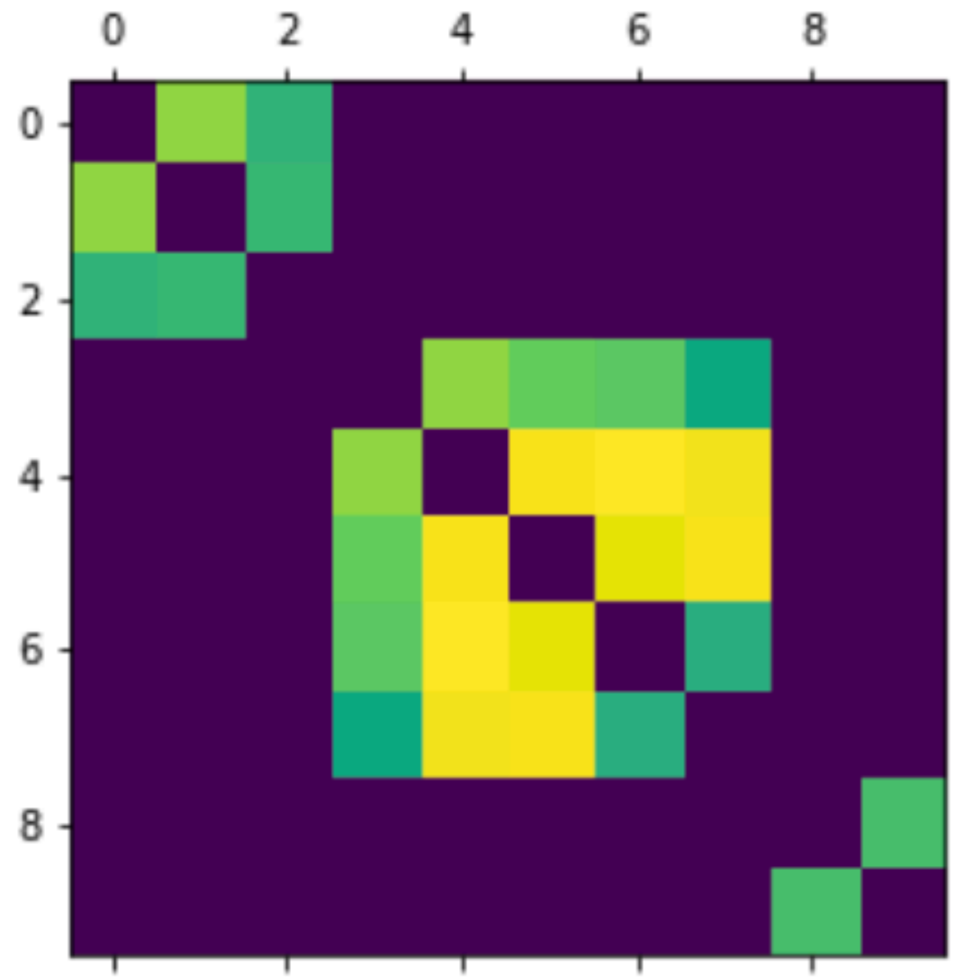
$$x(t) = \begin{pmatrix} 0.41 \\ -0.82 \\ 0.41 \end{pmatrix} \cos(\sqrt{3}t)$$

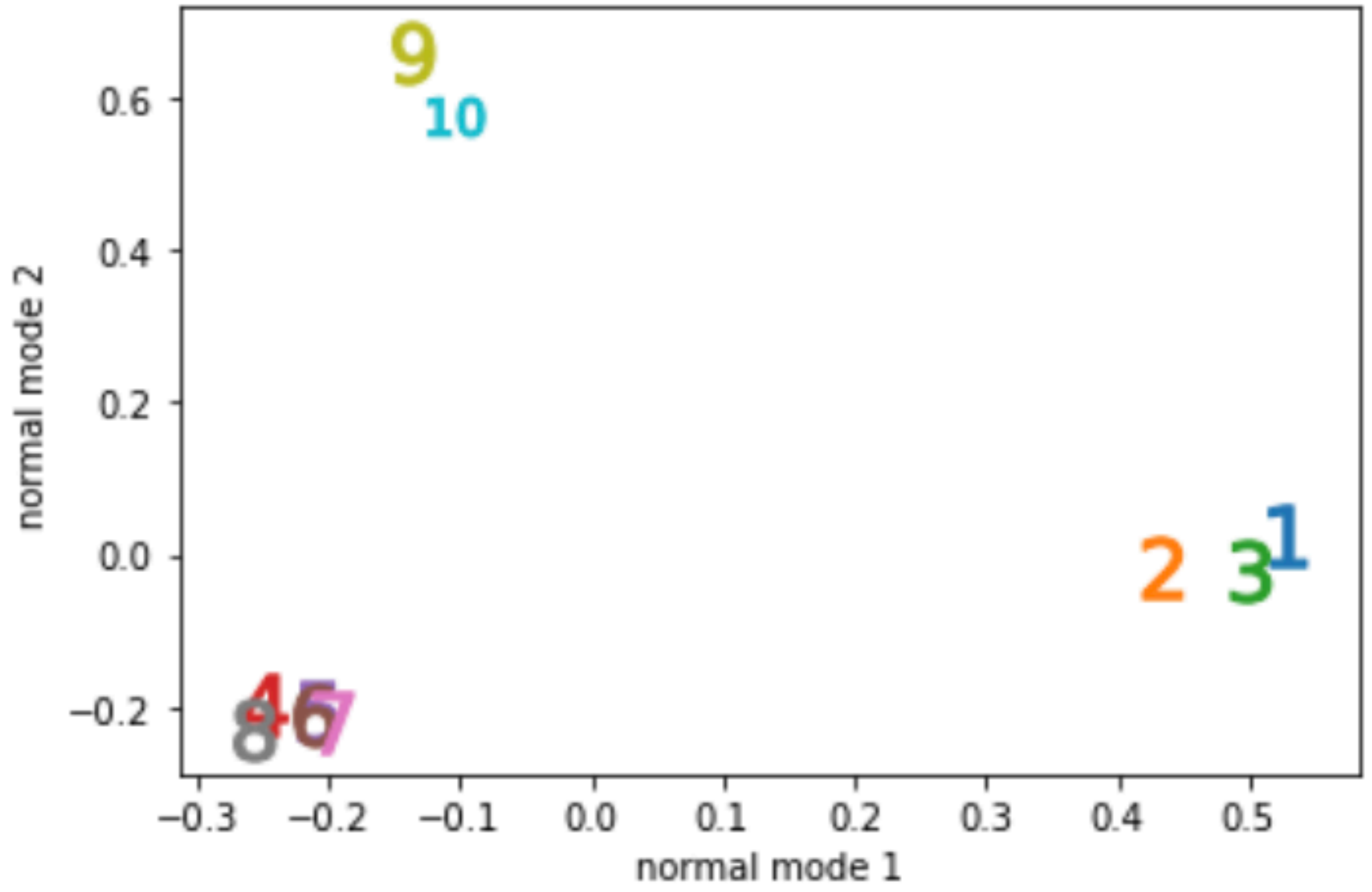




Part 2: Continued

- Generalize to Data Science:
  - If particles are data points
  - and K describes the similarity between data points
  - First few eigenvectors should group strongly coupled pairs
- Generalize to N body system
  - Give an example for 10 body systems





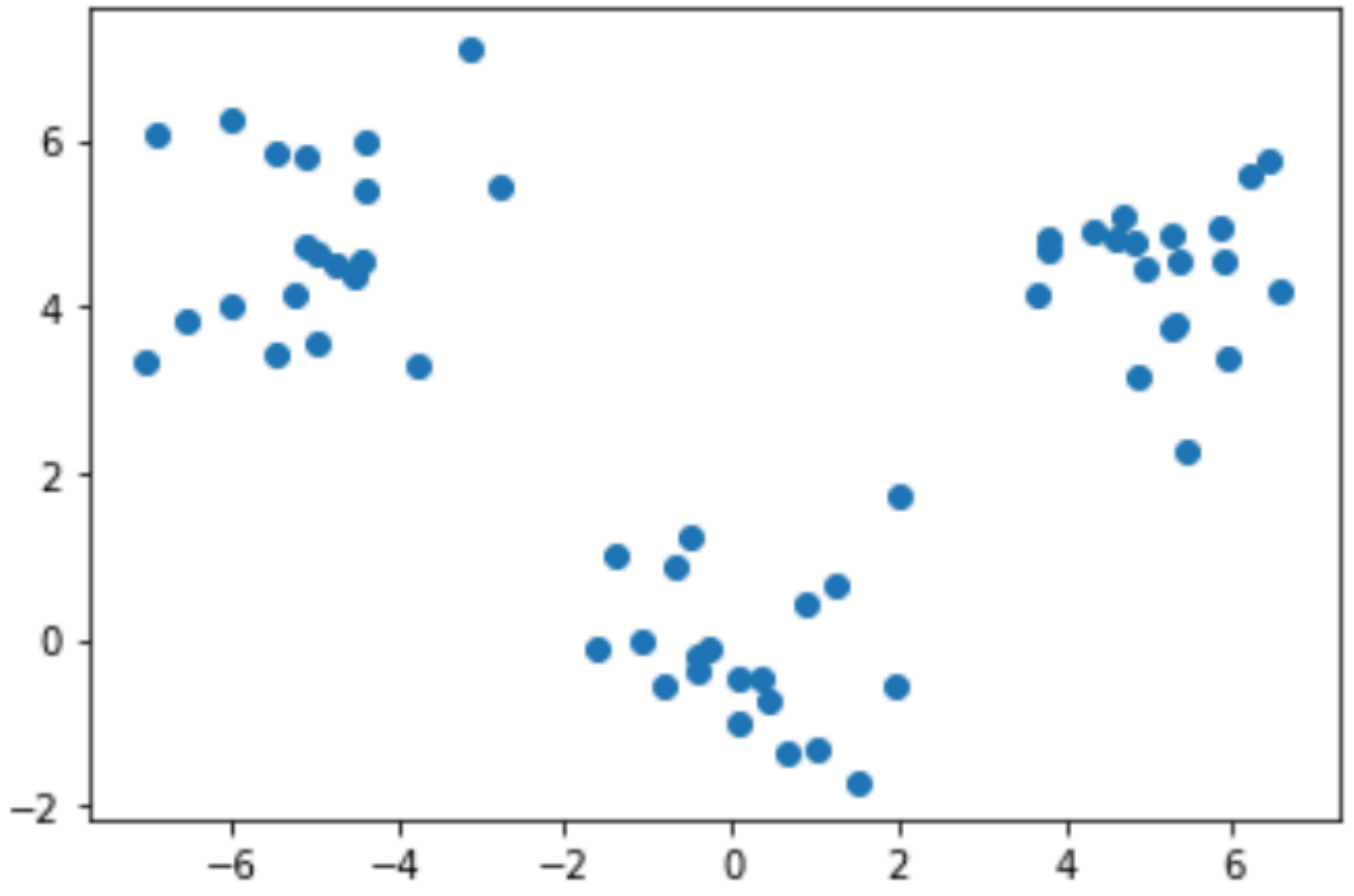
Part 3: K-Means

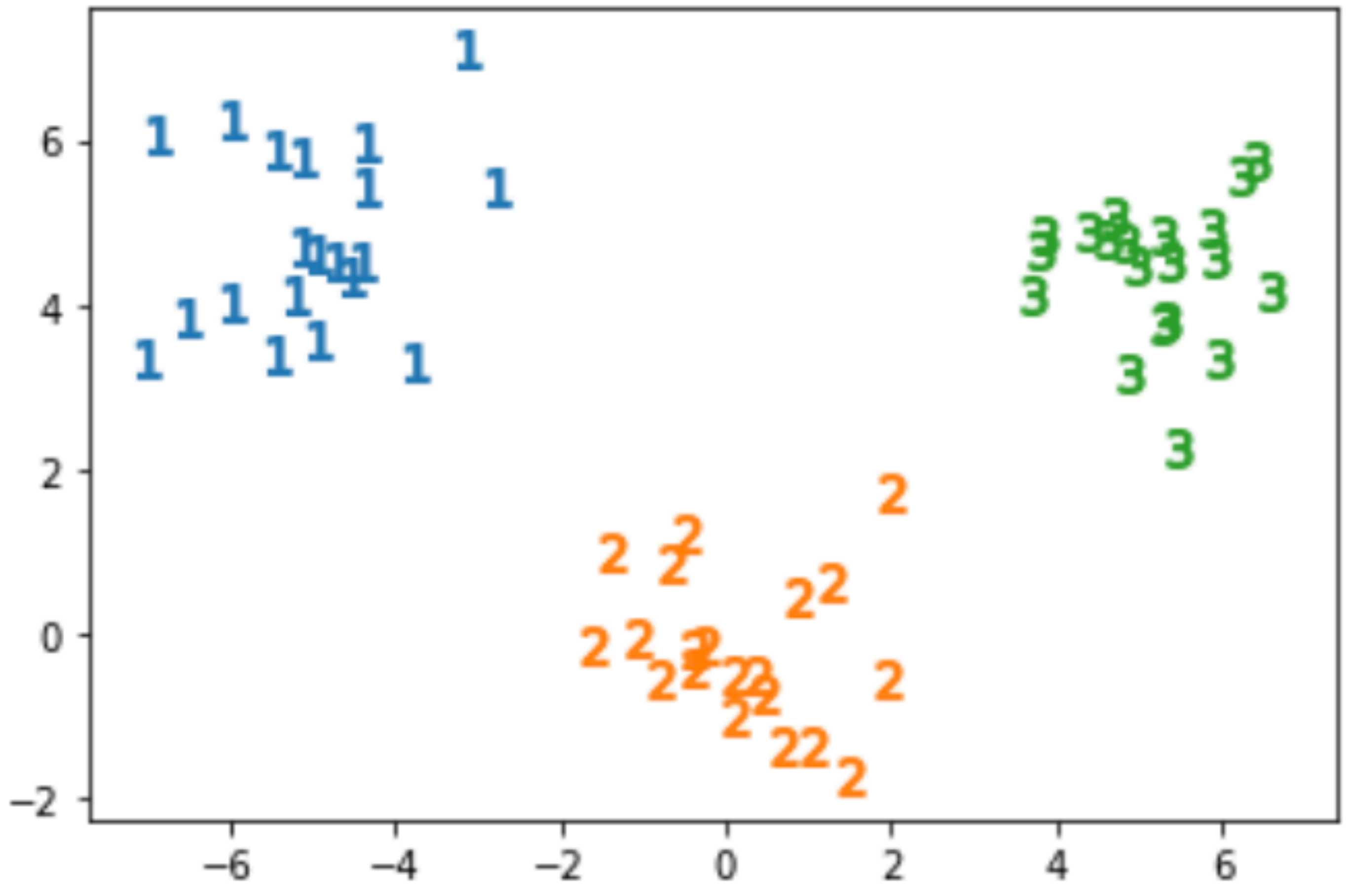
- Students already learned how to embed data into spectral spaces
- Use K-means method to cluster the points in the spectral spaces
- The introduction to K-means is straight forward and lead the students to implement it.

```
class KMeans:
    def __init__(self, n_clusters):
        self.n_clusters = n_clusters

    def fit(self, data, max_iter = 100):
```

```
km = KMeans(3)
km.fit(data)
cluster_assign = km.predict(data)
```





Part 4: Spectral Clustering

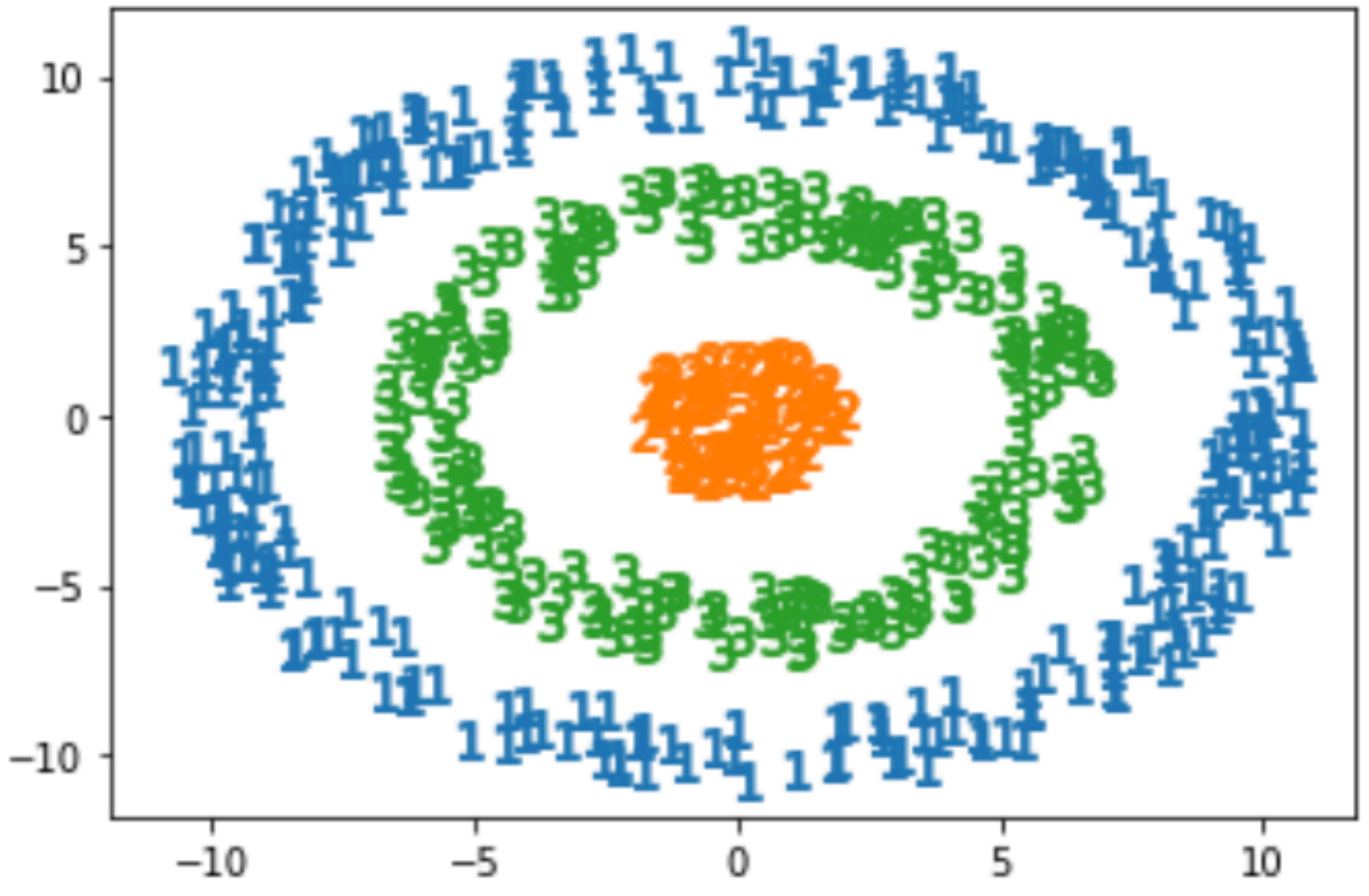
- Combine the previous parts:
  - Students learned to embed data using similarity matrix K
  - Students learned to cluster data points based on Euclidean distance
  - Combine together gives our desired algorithm

```
dists = np.sum( (data[:,None,:] - data[None,:,:])**2, axis = -1)
K = np.exp(-dists)

assignments = SpectralClustering(K, 3 , random_seed = 2100)

def plot_clustering(data, assign, ax = None):
    n = np.max(assign)+1
    if ax is None:
        ax = plt.gca()
    for i in range(n):
        ax.scatter(data[assign==i,0], data[assign==i,1], marker=f"${i+1}$", s = 20*5 )
    return ax

plot_clustering(data, assignments)
```



Part 5: Standard Package

- In this part, we conclude what is learned in this module
- It lets students to know they first embed using oscillators and then clustering.
- It also teaches students to use standard library for this algorithm ‘scikit-learn’. (It should be straightforward as we developed the code using the standard package style)

```
from sklearn.cluster import SpectralClustering

sc = SpectralClustering(n_clusters= 3,
                        n_components = 2,
                        assign_labels = "kmeans",
                        affinity = "rbf",
                        gamma = 1,
                        random_state = 2100
                        )

labels = sc.fit_predict(data)
```

Conclusion

- The module leads students to derive “data science” techniques using what they learned in physics class. This will increase students’ interests, not only in data science but also physics.
- The learned data science techniques is quite useful in real applications
- The module does not assume strong programming background
- The module leads the student to write “standard package” style codes
- The module provides useful homework problems to let them play with the new learned techniques.