The Instructions for Koopman Operator computation code

Section 1. Load your dataset

1. Preparing your dataset as the "data" variable, whose rows are the state variable, and columns are the time points.
2. For the continuous time system, we are using *ode45* to generate the dataset, you need to specify the T (simulation time), dT (time interval).
   For the discrete time system, we are simply evolving the system in a for-loop, and only the T(simulation time steps) needs to be specified.
   If multiple initial conditions specified, stack their trajectories into the "data" variable.

```
21 -  for i = 1:N
22 -      [t, z] = ode45(f,[0:dT:T],z0(:,i));
23 -      X = [X z(1:end-1,:)'];
24 -      Y = [Y z(2:end,:)'];
25 -      data = [data z'];
26 -  end
```

3. For the user-defined dataset, in case you are applying your own dataset, you can simply load your dataset, and assign to the "data" variable. (be careful with the rows and columns)

Section 2. Choose your dictionary function

1. By default the dictionary functions are chosen as a set of Gaussian functions(one kind of RBF, Radial basis function), so we only need to choose the "centers"(mean), and "sigma"(standard deviation) for these functions.

2. To choose the centers appropriately, k-means clustering method is being applied. (Make sure the statistics toolbox has been installed in your matlab)

```
33      % Radial Basis Functions ---Problems defined on irreg
34 -    Psi = @RBF;
35 -    sigma = 0.05; % Specify here for all Gaussian diction
36 -    Nk = 500; % Specify here as number of dictionary func
37 -    [~, centers] = kmeans(data',Nk); % k-means clustering
38 -    centers = centers'; % centers size of 2 x Nk
```

   You only need to specify the number of centers, "Nk" variable and it will be passed to the kmeans function. The output "centers" will be converted to a matrix, in which each column corresponding to one dictionary function.

3. To choose the sigma, in the code we have, we are using the same sigma for all dictionary functions. Hence we need to choose the sigma appropriately, such that those Gaussian functions won't overlap each other(will happen when sigma is very large), and all the trajectories is covered by the range of Gaussian functions.(won't be covered if sigma is too small)

Section 3. Compute the G and A matrix

1. In this section, we are applying the formula shown in the class, to compute the G and A matrices.

$$\boldsymbol{G} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\Psi}(\boldsymbol{x}_m)^* \boldsymbol{\Psi}(\boldsymbol{x}_m)$$

$$\boldsymbol{A} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\Psi}(\boldsymbol{x}_m)^* \boldsymbol{\Psi}(\boldsymbol{y}_m)$$

2. The implementation in the code firstly generate two time series data from the dataset we obtained in Section 1, each column of X and Y satisfying, Y(i) = X(i-1), such that Y(i) = T(X(i)), i = 2,3,… , M.

```
41 -    X = data(:,1:end-1);
42 -    Y = data(:,2:end);
```

3. Then we just use the for-loop iterations, and go through all the snapshots of (X(i), Y(i)) pair, to compute the G and A matrix.

```
43 -    G = 0;
44 -    A = 0;
45 -    M = size(X,2); % length of time-series
46 -    for i = 1:M
47 -        PsiX = Psi(X(:,i),centers,sigma);
48 -        PsiY = Psi(Y(:,i),centers,sigma);
49 -        G = G + PsiX'*PsiX;
50 -        A = A + PsiX'*PsiY;
51 -    end
52 -    G = G/M;
53 -    A = A/M;
54
55      % EDMD
56 -    Kdmd = pinv(G)*A;
```

4. In the line 56, the finite dimensional approximation of Koopman Operator is computed by

$$\boldsymbol{K} \triangleq \boldsymbol{G}^{+}\boldsymbol{A}$$

where + denotes the pseudo-inverse.

5. In line 59, we can choose how many Eigenfunctions we will plot in the next section.

```
59 -    n = 4; % compute eigenfunctions corresponding
```

Section 4. Plot the dominant Eigenfunction of K matrix

1. Before computing the Koopman Eigenfunction, we need to specify the state space where the Eigenfunctions are being plotted. For the 2D example, it is very straightforward to define these boxes in x-y space. The Nx and Ny can be chosen smaller to reduce the simulation time in Matlab, or can be chosen larger for better resolution.

```
60      % Construct X-Y boxes
61 -    Nx = 100;
62 -    Ny = 100;
63 -    x_limit = [-1.5 1.5];% range of x
64 -    y_limit = [-.4 .4];% range of y
65 -    xmax=x_limit(2);xmin=x_limit(1);
66 -    ymax=y_limit(2);ymin=y_limit(1);
67 -    dx=(xmax-xmin)/(Nx);
68 -    dy=(ymax-ymin)/(Ny);
69 -    xvec = xmin+dx/2:dx:xmax-dx/2;
70 -    yvec = ymin+dy/2:dy:ymax-dy/2;
71 -    [xx,yy] = meshgrid(xvec,yvec);
72 -    x = reshape(xx,1,[]);
73 -    y = reshape(yy,1,[]);
74 -    XY =  [x;y];
```

2. Each column of V is the right eigenvector of K matrix, Each column of W is the left eigenvector of K matrix

```
76 -    [V, D, W] = eig(Kdmd); % D are eigenvalues of Koopman Operator
```

3. Line 80-86, we first plot the eigenvalues with the unit circle in complex plane.

```
80      % Plot eigenvalues in complex plane
81 -    figure
82 -    th=0:0.1:2*pi;
83 -    plot(cos(th),sin(th),'k-')
84 -    hold on;
85 -    plot(diag(D),'*')
86 -    grid on
```

4. Then we sort the eigenvalues in descending order, and plot the eigenfunctions corresponding to first n eigenvalues.

```
95 -            eigfunc1_Kpm(i,j) = Psi(XY(:,i),centers,sigma)*V(:,idx(j));
```

For PF eigenfunctions and Koopman eigenfunctions, the only difference exists in line 95, that $\Psi(x)V(j)$ is for the Koopman eigenfunctions, and $\Psi(x)W(j)$ is for the P-F eigenfunctions.