In [ ]:

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import pandas as pd

from numpy import linalg
from scipy import integrate
from matplotlib import animation, rc

plt.rcParams["figure.figsize"] = (10,6)
```

In [ ]:

```
a = 0.7
b = 0.8
tau = 12.5

def fhn(t,x,a,b,tau,I): ## FHN RHS
    v = x[0] - (x[0]**3)/3 - x[1] + I
    w = (x[0] + a - b*x[1])/tau
    return [v,w]

def fhn_fixed_point(I): ##calculate fixed point and jacobian
    ## assume a = 0.7, b = 0.8, tau = 12.5
    tau = 12.5
    a = 0.7
    b = 0.8
    #inter = (np.sqrt(576*I**2 - 1008*I + 445) + 24*I - 21)**(1/3)
    #v_star = (inter/(2*(2**(1/3)))) - 1/(inter*(2**(2/3)))
    det = (np.sqrt(576*I**2 - 1008*I + 445) + 24*I - 21)
    v_star = ((2**(1/3))*(det**(2/3)) - 2)/(2*(2**(2/3))*(det)**(1/3)) ##simplified
    w_star = (v_star + a)/b

    jacobian = np.array(([1 - v_star**2, -1], [1/tau, -b/tau]))
    return v_star, w_star, jacobian

T_max = 100
dt = 0.1
tspan = np.arange(0,T_max+dt,dt)
vspan = np.linspace(-3,3,len(tspan))

dI = 0.01
I_list = np.arange(0.15, 1.5+dI, dI)
fhn_sol = np.zeros((len(I_list),len(tspan),2))
W_nullcine = np.zeros((len(I_list),len(tspan)))
V_nullcine = np.zeros((len(I_list),len(tspan)))

fhn_0 = [0,0]

for i in range(len(I_list)):
    sol = sp.integrate.solve_ivp(fhn, [0,T_max], y0 = fhn_0, args = (a,b,tau,I_list[i]),
                                 t_eval = tspan, method = 'RK45')
    fhn_sol[i,:,0] = sol.y[0,:]
    fhn_sol[i,:,1] = sol.y[1,:]
    W_nullcine[i,:] = b*vspan - a
    V_nullcine[i,:] = vspan - ((vspan)**3)/3 + I_list[i]

fp_fhn = np.zeros((3, len(I_list)))

for i, I in enumerate(I_list):
    fp_fhn [:,i] = I, fhn_fixed_point(I)[0], fhn_fixed_point(I)[1]
```

In [ ]:

```
# ### Depricated
# data = '0.75 2.72344875 -0.61066362   0.09077326 0.7  2.74056482      -0.45540864     0.12338276 1.1  2.70886462      -1.5
# clean_data = data.replace('\t',' ').split()

# rnn_fixed_points_old = np.zeros((int(len(clean_data)/4),4))
# for j in range(4):
#     rnn_fixed_points_old[:,j] = [float(x) for i, x in enumerate(clean_data) if i%4 == j]

# idx = np.argsort(rnn_fixed_points_old[:,0])
# rnn_fixed_points_old[:,:] = rnn_fixed_points_old[idx,:]
```

In [ ]:
```
rnn_pca_2 = pd.read_csv('output_1.csv').to_numpy()
rnn_pca_3 = pd.read_csv('output_2.csv').to_numpy()


rnn_pca_2[:,:] = rnn_pca_2[np.argsort(rnn_pca_2[:,0]),:]
rnn_pca_3[:,:] = rnn_pca_3[np.argsort(rnn_pca_3[:,0]),:]

scale_v = 2

rnn_2_wstar = (scale_v*rnn_pca_2[:,3] + a)/b ## uses explicit w-nullcine solution
rnn_3_wstar = (scale_v*rnn_pca_3[:,4] + a)/b

fp_compare = np.zeros((rnn_pca_2.shape[0],6)) ## [Iext, v* RNN, w* RNN, v* FHN, w* FHN, v* diff]
fp_compare[:,0] = rnn_pca_2[:,0]
fp_compare[:,1] = scale_v*rnn_pca_2[:,3]
fp_compare[:,2] = rnn_2_wstar
for i, Iext in enumerate(fp_compare[:,0]):
    fp_compare[i,3], fp_compare[i,4], _ = fhn_fixed_point(Iext)
fp_compare[:,5] = fp_compare[:,1] - fp_compare[:,3]
fp_compare_no_dup = (pd.DataFrame(fp_compare).drop_duplicates()).to_numpy()
```

In [ ]:
```
for i in range(rnn_pca_2.shape[0]):
    plt.plot(2*rnn_pca_2[i,3:], 'k-', lw = 3)
plt.xlabel('Time [s]', fontsize = 15)
plt.ylabel('$v^*$', fontsize = 15)
plt.title('Time Evolution of RNN ouput $v^*$', fontsize = 20)
plt.grid()
plt.show()
```

In [ ]:
```
time_length = 100
iteration = 20

pc_fixed_point = np.loadtxt("PCA2_fixed_points.csv",delimiter=" ", dtype=float)
pc_trajectory = np.loadtxt("PCA2_trajectory.csv",delimiter=" ", dtype=float)

for i in range(iteration):
    plt.plot(pc_trajectory[0, i * time_length:(i + 1) * time_length],
             pc_trajectory[1, i * time_length:(i + 1) * time_length], linewidth=1.1)
plt.scatter(pc_fixed_point[0], pc_fixed_point[1], color='red', marker='x', label = 'FP $x^*$')
plt.xlabel('PCA 1', fontsize = 15)
plt.ylabel('PCA 2', fontsize = 15)
plt.title('Phase Space of RNN State Trajectories with FP $x^*$', fontsize = 20)
plt.grid()
plt.legend(fontsize = 15)
plt.show()
```

In [ ]:
```
pc_fixed_point = np.loadtxt("PCA3_fixed_points.csv",delimiter=" ", dtype=float)
pc_trajectory = np.loadtxt("PCA3_trajectory.csv",delimiter=" ", dtype=float)

fig, ax = plt.subplots(1, subplot_kw={"projection": "3d"}, figsize = (10,10))
for i in range(iteration):
    ax.plot(pc_trajectory[0, i * time_length:(i + 1) * time_length],
            pc_trajectory[1, i * time_length:(i + 1) * time_length],
            pc_trajectory[2, i * time_length:(i + 1) * time_length], linewidth=1.1)
ax.scatter(pc_fixed_point[0], pc_fixed_point[1], pc_fixed_point[2], color='red', marker='x', label = '$x^*$ FP')
ax.set_xlabel('PCA 1', fontsize = 15)
ax.set_ylabel('PCA 2', fontsize = 15)
ax.set_zlabel('PAC 3', fontsize = 15)
ax.view_init(30,45)
plt.title('Phase Space of RNN State Trajectories with FP $x^*$', fontsize = 20)
plt.legend(loc = 'lower center', fontsize = 15)
plt.show()
```

In [ ]:
```
error = np.round(np.sum(fp_compare_no_dup[:,5]**2)/len(fp_compare_no_dup[:,5]), decimals = 2)

plt.plot(fp_compare_no_dup[:,0],fp_compare_no_dup[:,1], 'ro-', label = 'RNN $v^*$')
plt.plot(fp_compare_no_dup[:,0],fp_compare_no_dup[:,3], 'bo-', label = 'FHN $v^*$')
plt.xlabel('$I_{ext}$', fontsize = 20)
plt.ylabel('$v^*$', fontsize = 20)
plt.title('Comparison of $v^*(I_{ext})$ for FHN and RNN', fontsize = 20)
plt.text(1.1,-0.4, f'MSE = {error} mV', c = 'w', backgroundcolor = 'k', fontsize = 20)
plt.grid()
plt.legend(fontsize = 15)
plt.show()
```

In [ ]:
```
fig, ax = plt.subplots(1, 3, subplot_kw={"projection": "3d"}, figsize = (15,15))
for i in range(3):
    ax[i].plot(fp_fhn[1,:],fp_fhn[2,:],fp_fhn[0,:], label ='FHN (Full I range)')
    ax[i].scatter(fp_compare_no_dup[:,3],fp_compare_no_dup[:,4],fp_compare_no_dup[:,0], c = 'k', label = 'FHN')
    ax[i].scatter(fp_compare_no_dup[:,1],fp_compare_no_dup[:,2],fp_compare_no_dup[:,0], c = 'r', label = 'RNN')
    ax[i].set_xlabel('v [mV]', fontsize = 10)
    ax[i].set_xticks(np.arange(-1,1+0.5,0.5))
    ax[i].set_ylabel('w', fontsize = 10)
    if i == 0 or 2:
        ax[i].set_zlabel('$I_{ext}$ [mA]', fontsize = 10)

ax[1].set_zticks([ ])
ax[2].set_yticks([ ])
ax[2].set_ylabel(' ')
ax[1].view_init(90,-90)
ax[2].view_init(0,-90)
ax[1].set_title('Fixed Points $(v*,w*)$ for RNN and FHN', fontsize = 20)
plt.legend(loc = 'upper right', fontsize = 10)
plt.show()
```

In [ ]:
```
C = 85 ## current (I_ext) indexer

v_star, w_star, jacobian = fhn_fixed_point(I_list[C])

eig_l, eig_v = np.linalg.eig(jacobian)

if np.imag(eig_l[0]) != 0 and np.real(eig_l[0]) < 0:
    title = 'Stable Spiral'
else:
    title = 'Limit Cycle'

fig, ax = plt.subplots(1)
ax.vlines(-1, -1, 2.5, color = 'c', linestyles = '--')
ax.vlines(1, -1, 2.5, color = 'c', linestyles = '--')
ax.plot(W_nullcine[C,:], vspan, 'b',lw = 2, label = 'W nullcine')
ax.plot(vspan,V_nullcine[C,:], 'r',lw = 2, label = 'V nullcine')
ax.plot(fhn_sol[C,:,0], fhn_sol[C,:,1], 'k--',lw = 2, label = 'Solution')
ax.plot(v_star,w_star, 'm*', ms = 20, label = 'Fixed Point')
#ax.plot(fp_compare_no_dup[:,1], fp_compare_no_dup[:,2], 'go', lw = 3, label = 'RNN $(v^*,w^*)$')
ax.set_xlabel('v [mV]', fontsize = 20)
ax.set_ylabel('w', fontsize = 20)
ax.set_xlim([-2.5,2.5])
ax.set_ylim([-0.8,2.5])
ax.set_title(f'{title} of FHN Model with $I = {np.round(I_list[C], decimals = 2)}$', fontsize = 20)
ax.text(-2.3, -0.65, f'eig vals: $[{np.round(eig_l[0], decimals = 2)}, {np.round(eig_l[1], decimals = 2)}]$'
         , c = 'w', backgroundcolor = 'k', fontsize = 10)
ax.text(-0.6, 2.25, 'Limit Cycle region',
        c = 'w', backgroundcolor = 'k', fontsize = 15)
ax.legend(loc = 'lower right')
ax.grid()
plt.show()
```

In [ ]:
```
fig, ax = plt.subplots(1,2,sharey=True)
ax[0].plot(tspan, fhn_sol[C,:,0], 'k-', lw = 3)
ax[1].plot(tspan, fhn_sol[C,:,1], 'k-', lw = 3)
fig.supxlabel('t [s]')
ax[0].set_title('v [mV]')
ax[1].set_title('w')
fig.suptitle('FHN $v$ and $w$ with $I_{ext} = 1.0$', fontsize = 17)
ax[0].grid()
ax[1].grid()
plt.show()
```

In [ ]:

```python
v_star, w_star, jacobian = fhn_fixed_point(I_list[0])

eig_l, eig_v = np.linalg.eig(jacobian)

if np.imag(eig_l[0]) != 0 and np.real(eig_l[0]) < 0:
    title = 'Stable Spiral'
else:
    title = 'Limit Cycle'


rnn_fp_v_ani, rnn_fp_w_ani  = [], []

fig, ax = plt.subplots(1)
ax.vlines(-1, -1, 2.5, color = 'c', linestyles = '--')
ax.vlines(1, -1, 2.5, color = 'c', linestyles = '--')
ax.plot(W_nullcine[0,:], vspan, 'b',lw = 2, label = 'W nullcine')
vnull, = ax.plot(vspan,V_nullcine[0,:], 'r',lw = 2, label = 'V nullcine')
sol, = ax.plot([], [], 'k--',lw = 2, label = 'Solution')
fhn_fp, = ax.plot([],[], 'm*', ms = 20, label = 'FHN FP')
rnn_fp, = ax.plot([], [], 'g*', ms = 20, label = 'RNN FP')
ax.set_xlabel('v [mV]', fontsize = 20)
ax.set_ylabel('w', fontsize = 20)
ax.set_xlim([-2.5,2.5])
ax.set_ylim([-0.8,2.5])
eig_val_txt = ax.text(-2.3,-0.65,f'eig vals: $[{np.round(eig_l[0], decimals = 2)},{np.round(eig_l[1],decimals = 2)}]$'
         , c = 'w', backgroundcolor = 'k', fontsize = 10)
ax.text(-0.6, 2.25, 'Limit Cycle region',
        c = 'w', backgroundcolor = 'k', fontsize = 15)
ax.legend(loc = 'lower right')
ax.grid()

def update(i):
    v_star, w_star, jacobian = fhn_fixed_point(I_list[i])

    eig_l, eig_v = np.linalg.eig(jacobian)

    if np.imag(eig_l[0]) != 0 and np.real(eig_l[0]) < 0:
        title = 'Stable Spiral'
    else:
        title = 'Limit Cycle'

    vnull.set_data(vspan, V_nullcine[i,:])
    sol.set_data(fhn_sol[i,:,0], fhn_sol[i,:,1])
    fhn_fp.set_data(v_star,w_star)

    if len(rnn_fp_v_ani)!= 14 and np.round(I_list[i], decimals = 2) == np.round(fp_compare_no_dup[len(rnn_fp_v_ani),0],
                                                                                 decimals = 2):
        rnn_fp_v_ani.append(fp_compare_no_dup[len(rnn_fp_v_ani),1])
        rnn_fp_w_ani.append(fp_compare_no_dup[len(rnn_fp_w_ani),2])
        rnn_fp.set_data(rnn_fp_v_ani,rnn_fp_w_ani)

    elif len(rnn_fp_v_ani)== 14 and np.round(I_list[i], decimals = 2) == np.round(fp_compare_no_dup[len(rnn_fp_v_ani)-1,0],
                                                                                   decimals = 2):
        rnn_fp_v_ani.append(fp_compare_no_dup[len(rnn_fp_v_ani)-1,1])
        rnn_fp_w_ani.append(fp_compare_no_dup[len(rnn_fp_w_ani)-1,2])
        rnn_fp.set_data(rnn_fp_v_ani,rnn_fp_w_ani)

    eig_val_txt.set_text(f'eig vals: $[{np.round(eig_l[0], decimals = 2)}, {np.round(eig_l[1], decimals = 2)}]$')

    ax.set_title(f'{title} of FHN Model with $I = {np.round(I_list[i], decimals = 2)}$',loc = 'left', fontsize = 20)
    return vnull,sol,fhn_fp,rnn_fp,eig_val_txt

ani = animation.FuncAnimation(fig, update, frames = range(len(I_list)), interval = 1, blit = True)

writergif = animation.PillowWriter(fps = 24)
ani.save('FHN_fp.gif', writer = writergif)
```