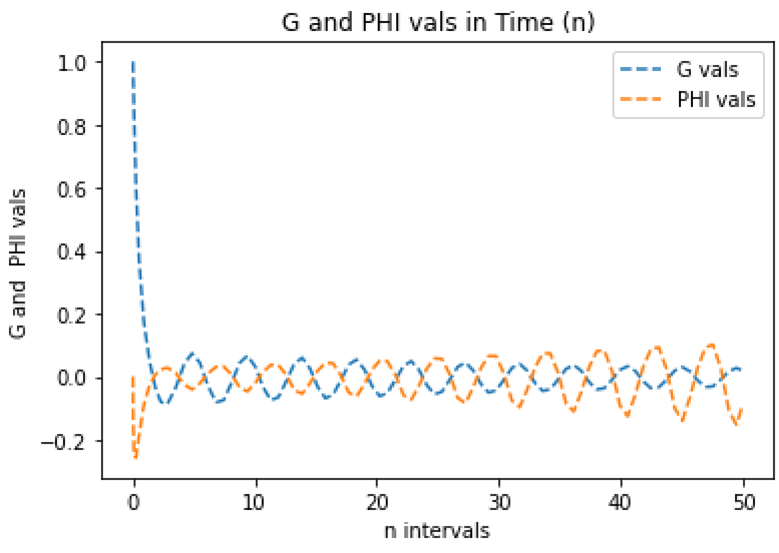
```
#importing important libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp


#creating a global variable
alpha=0
#defining a de function
def coupled_de_func(n,y):

    #defining the variables
    Sc=1
    sigma=1
    epsilion=10
    global alpha

    #assigning each DE a vector element (Conversion from scalar to vector components)
    X1=y[0]
    X2=y[1]
    X3=y[2]
    X5=y[3]
    X6=y[4]
    X7=y[5]
    X8=y[6]

    #defining the des
    dX1dn=X2
    dX2dn=X3
    dX3dn=-(X1*X2)/2
    dX5dn=X7
    dX6dn=X8
    dX7dn=sigma/2*(-X1*X7+X5*X2+2*X1*X2 *(-alpha*X1* X6*np.exp(X5*np.sqrt(X1) /(1+epsilion

    dX8dn=Sc/2*(-X1*X8+2*X1*X2* (-X1*X6*np.exp(X5*np.sqrt(X1)/(1+epsilion*X5*np.sqrt(X1)))
     (1+epsilion*X5*np.sqrt(X1))**2))

    return np.array([dX1dn,dX2dn,dX3dn,dX5dn,dX6dn,dX7dn,dX8dn])

n_span=np.array([0,50]) #--defining the time vector ranging from 0-50(can be changed based
times=np.linspace(n_span[0],n_span[1],100) #--converting the y-components into a sort of a

#lets say for instance we set the initial conditions as follows
y0=[4,1,0,0,1,1,0]

#solving the DE using solve_ivp function
solution=solve_ivp(coupled_de_func,n_span,y0,n_eval=times) #---calling the solve_ivp funct
n=solution.t
X1=solution.y[0] #storing the outputs of the solve_ivp results in the following defined ve
X2=solution.y[1]
X3=solution.y[2]
X5=solution.y[3]
X6=solution.y[4]
X7=solution.y[5]
```

```
X8=solution.y[6]


#plotting the G and PHI results
plt.plot(n,X7,'--',label='G vals')
plt.plot(n,X8, '--', label='PHI vals')
plt.title('G and PHI vals in Time (n)')
plt.xlabel('n intervals')
plt.ylabel('G and  PHI vals')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/scipy/integrate/_ivp/common.py:41: UserWarnin
  .format(", ".join("`{}`".format(x) for x in extraneous)))
```



✓ 0s completed at 12:42 PM