BT6270: Computational Neuroscience

Assignment 3

Sumedh Sanjay Kangne (BE21B040)
Department of Biotechnology
Indian Institute of Technology, Madras

A BT6270: Computational Neuroscience Assignment

November 13, 2023

1 Hopf Oscillator

In the exploration of neural coding, two prevalent approaches are the spike frequency code, emphasizing the instantaneous rate of action potentials or ”rate code,” and the spike time code, which centers on the precise timing of action potential occurrences. Neural oscillators, crucial components in modeling these phenomena, are often conceptualized as Hopf oscillators with adaptive frequencies and dynamics described in the complex domain. The incorporation of adaptive frequencies allows these models to encompass the broad spectrum of spectral components observed in brain dynamics. In particular, oscillatory networks with high-frequency accommodation become essential for capturing the intricate and diverse neural activities present in complex brain processes. This framework provides a nuanced understanding of how neural systems encode information through dynamic frequency modulation and temporal patterns, contributing to advancements in both neuroscience and computational modeling.

Problem 1

For this assignment, consider two Hopf oscillators (equation given below) coupled together. Calculate the coupling coefficients (W_{12} and W_{21}) required to achieve a given phase difference between the oscillators. Two types of coupling are to be considered –

a) Complex coupling:
The phase differences to be achieved are –47°and 98°.
Consider $\omega_1 = \omega_2 = 5$.

b) Power coupling:
The normalized phase differences to be achieved are –47°and 98°.
Consider $\omega_1 = 5$ and $\omega_2 = 15$.

1.1 Complex coupling

Complex coupling refers to the interaction or connection between two dynamical systems, such as oscillators, where the coupling strength is described using complex numbers. In the context of systems like Hopf oscillators, these complex coupling coefficients determine how one oscillator influences another in terms of both amplitude and phase.
When two Hopf oscillators, possessing identical intrinsic frequencies, are mutually linked through complex coefficients characterized by Hermitian symmetry, they have the potential to demonstrate phase-locked oscillations at a specific angle, mirroring the angle of the complex coupling coefficient.

The following differential equations allow us to look at the above behavior of Hopf Oscillators:

$$\dot{r_1} = (\mu - r_1^2)r_1 + Ar_2cos(\theta_2 - \theta_1 + \varphi)$$

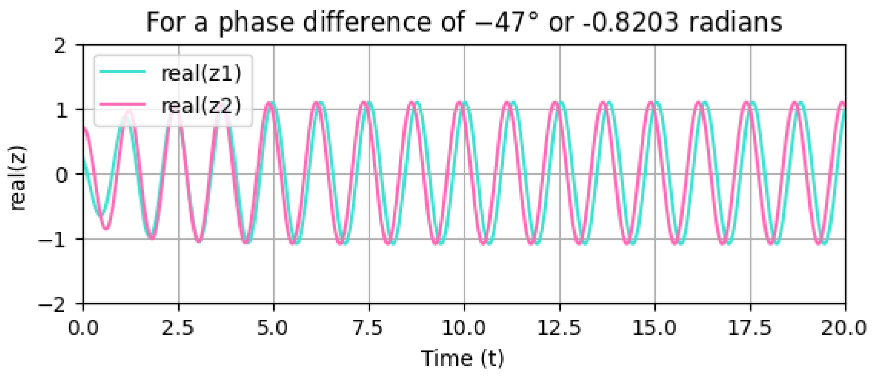
$$\dot{\theta}_1 = \omega_1 + A\frac{r_2}{r_1}sin(\theta_2 - \theta_1 + \varphi)$$

$$\dot{r_2} = (\mu - r_2^2)2_1 + Ar_1cos(\theta_1 - \theta_2 - \varphi)$$

$$\dot{\theta}_2 = \omega_2 + A\frac{r_1}{r_2}sin(\theta_1 - \theta_2 - \varphi)$$

where,
$\mu = 1$,
$\omega_1 = 5$,
$\omega_2 = 5$,
$A = 0.2$,

Plotting real values of z, i.e. r$\cos\theta$, as well as the phase difference vs time



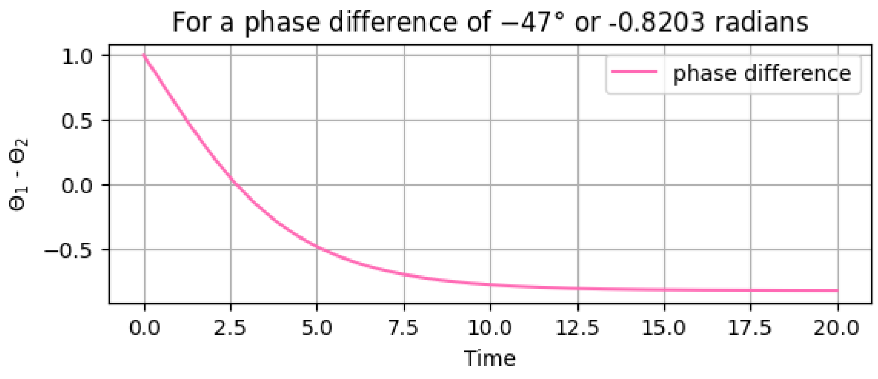


Figure 1: real(z) and $\theta_1 - \theta_2$ variation with time for a phase difference of -47°.

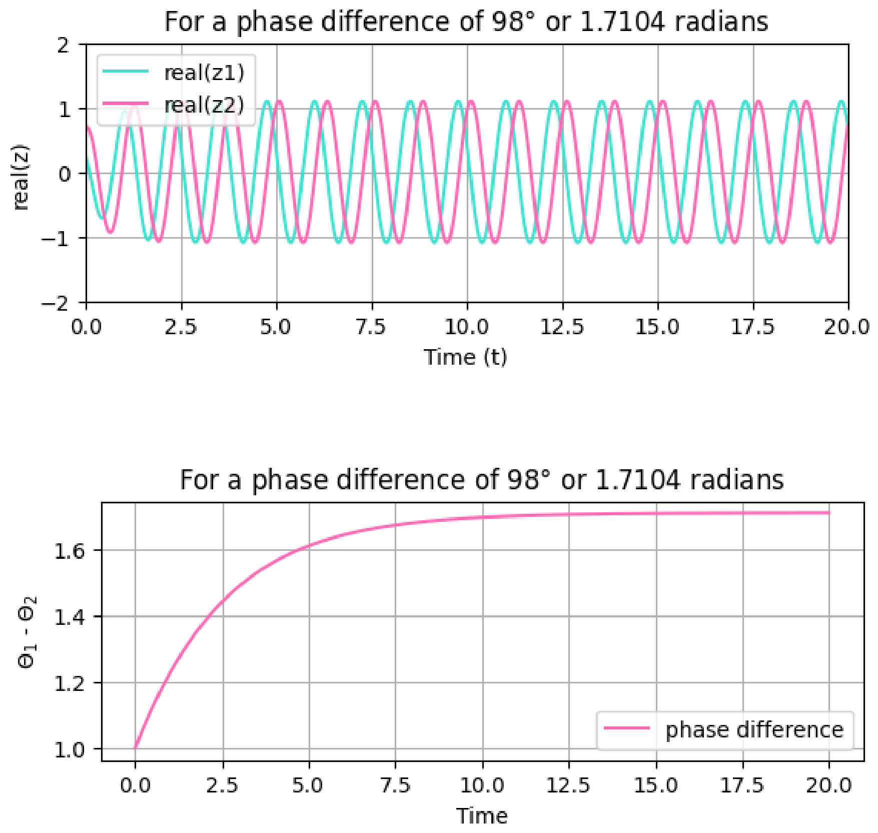


Figure 2: real(z) and $\theta_1 - \theta_2$ variation with time for a phase difference of 98°.

1.2 Power coupling

Power coupling in the context of oscillatory systems, such as sinusoidal oscillators modeled as Hopf or Kuramoto oscillators, refers to the transfer of influence and synchronization strength between two oscillators through complex coupling coefficients. These coefficients, often represented by complex numbers, encapsulate both the amplitude and phase information of the coupling. The concept of power coupling is crucial in understanding how oscillators entrain or synchronize at specific normalized phase differences. The intricacies of the coupling parameters determine the stability and nature of the entrainment, shedding light on the nuanced dynamics of coupled oscillatory systems. By exploring power coupling, researchers gain insights into the mechanisms that govern synchronization phenomena, offering valuable applications in fields ranging from physics to neuroscience and beyond.

The following differential equations allow us to look at the above behaviour of Hopf Oscillators:

$$\dot{r_1} = (\mu - r_1^2)r_1 + Ar_2^{\frac{\omega_1}{\omega_2}}cos(\frac{\theta_2}{\omega_2} - \frac{\theta_1}{\omega_1} + \frac{\varphi}{\omega_1\omega_2})$$

$$\dot{r_2} = (\mu - r_2^2)r_1 + Ar_2^{\frac{\omega_2}{\omega_1}}cos(\frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2} - \frac{\varphi}{\omega_1\omega_2})$$

$$\dot{\theta_1} = \omega_1 + A\frac{r_2^{\frac{\omega_1}{\omega_2}}}{r_1}sin(\frac{\theta_2}{\omega_2} - \frac{\theta_1}{\omega_1} + \frac{\varphi}{\omega_1\omega_2})$$

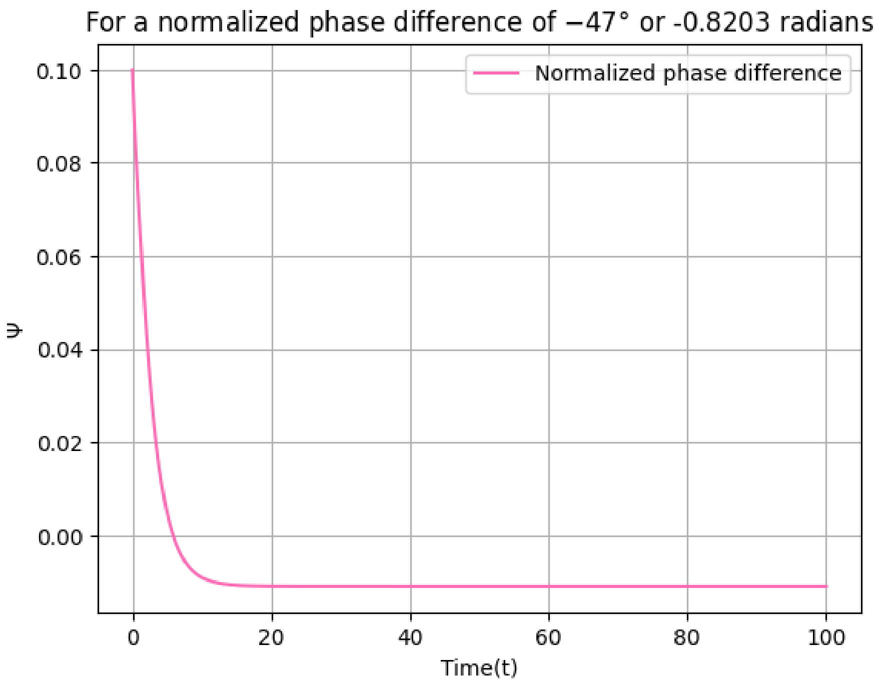
$$\dot{\theta_2} = \omega_2 + A\frac{r_1^{\frac{\omega_2}{\omega_1}}}{r_2}sin(\frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2} - \frac{\varphi}{\omega_1\omega_2})$$

$$\dot{\psi} = \frac{Ar_2^{\frac{\omega_1}{\omega_2}}}{\omega_1 r_1}sin\omega_1(\frac{\varphi}{\omega_1\omega_2} - \psi) + \frac{Ar_1^{\frac{\omega_2}{\omega_1}}}{\omega_2 r_2}sin\omega_2(\frac{\varphi}{\omega_1\omega_2} - \psi)$$

$$\sigma = \frac{\theta_1(t)}{\omega_1} - \frac{\theta_2(t)}{\omega_2} - \frac{\psi}{\omega_1\omega_2}$$

where,
$\mu = 1$,
$\omega_1 = 5$,
$\omega_2 = 15$,
$A = 0.2$,

Plotting normalized phase difference as well as σ variation with time



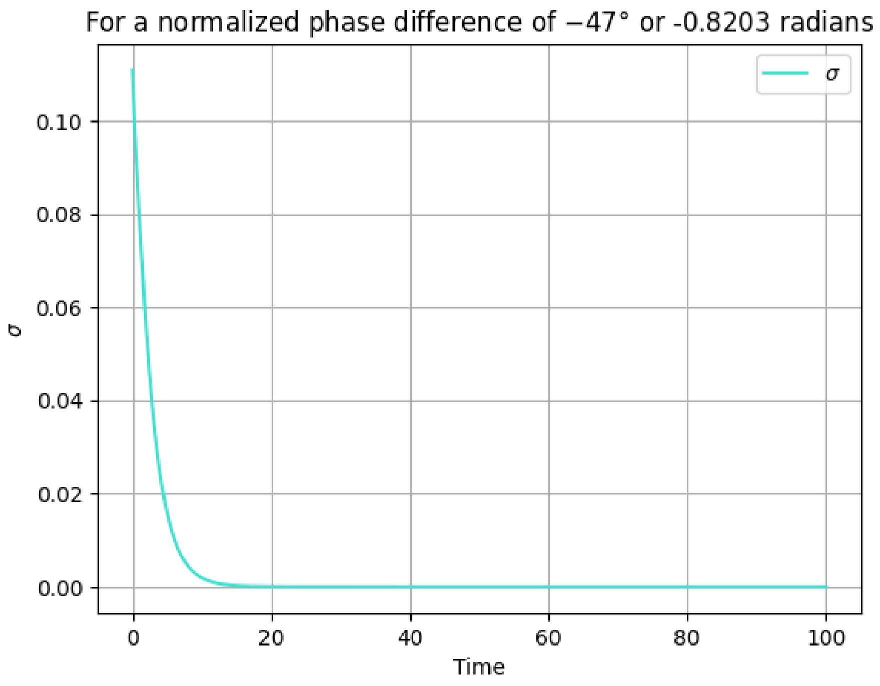
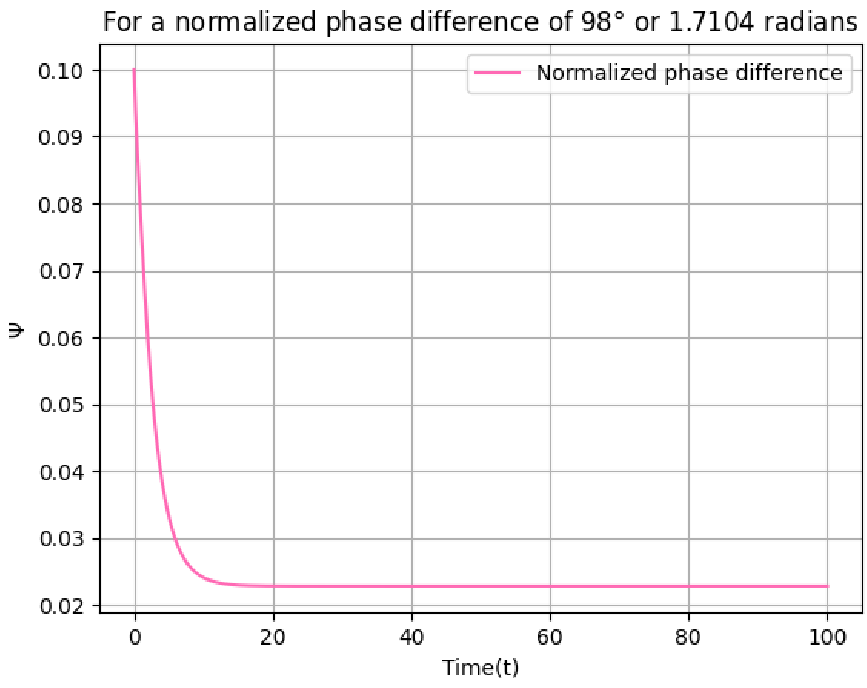


Figure 3: ψ and σ variation with time for a phase difference of -47°.



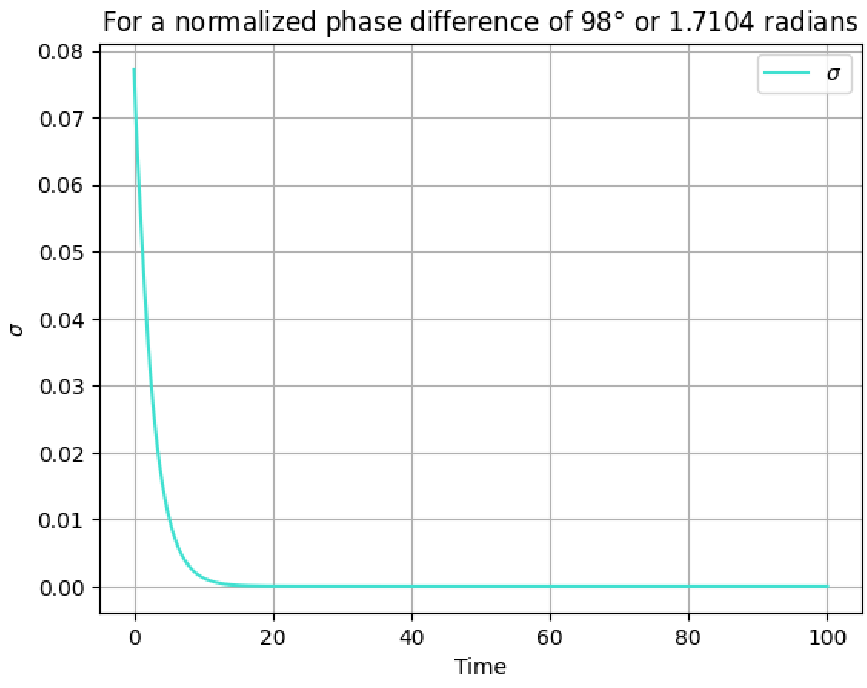


Figure 4: ψ and σ variation with time for a phase difference of 98°.

1.3 Assumptions Made:

- Pre-defined function 'odeint' from the scipy library was used to solve the system of differential equations.
- Values of μ and A were assumed as 1 and 0.2 respectively.
- Change in time dt for both the complex coupling and power coupling was taken as 0.02 and 0.1 ms respectively.

1.4 Code used for the assignment:

1. For complex coupling:

```
# -*- coding: utf-8 -*-
"""hopf.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/16oPggkZMKEFC-Y8WuAF7PwTem8dRpltM
"""

import numpy as np
import scipy
from scipy.integrate import odeint
import matplotlib.pyplot as plt

############################## Initialisation ##############################

# Defining variables
meu = 1
omega1 = 5
omega2 = 5
A = 0.2
y0 = [0.5, 1, 0.5, 0] # Initial conditions for r1, theta1, r2, theta2
phi_deg = [-47, 98]
t = np.linspace(0, 20, 1000)

############################## Function Definitions ##############################

# Defining the system of ODEs
def system(y, t, meu, omega1, omega2, A, phi):
    r1, theta1, r2, theta2 = y
    dr1dt = (meu - np.square(r1)) * r1 + A * r2 * np.cos(theta2 - theta1 + phi)
    dr2dt = (meu - np.square(r2)) * r2 + A * r1 * np.cos(theta1 - theta2 - phi)
    dtheta1dt = omega1 + (A * (r2 / r1) * (np.sin(theta2 - theta1 + phi)))
    dtheta2dt = omega2 + (A * (r1 / r2) * (np.sin(theta1 - theta2 - phi)))
    return [dr1dt, dtheta1dt, dr2dt, dtheta2dt]

# Defining function to extract solutions after solving the differential eqs
def extractSol(sol):
    r1 = sol[:, 0]
    theta1 = sol[:, 1]
    r2 = sol[:, 2]
    theta2 = sol[:, 3]
    return [r1,theta1,r2,theta2]

# Defining function to plot oscillator response over time
def plotComplexCoupling(r1,theta1,r2,theta2,deg):
    theta_diff = theta1 - theta2
    real_z1 = r1 * np.cos(theta1)
    real_z2 = r2 * np.cos(theta2)

    # Plot real(z) vs t
    fig= plt.figure()
    plt.subplot(2, 1, 1)
    plt.title(f"For a phase difference of ${phi_deg[deg]}\degree$ or {np.radians(
     phi_deg[deg]).round(4)} radians")
    plt.plot(t, real_z1, label='real(z1)',color='turquoise')
    plt.plot(t, real_z2, label='real(z2)',color='hotpink')
    plt.xlim([0, 20])
    plt.ylim([-2, 2])
    plt.xlabel('Time (t)')
    plt.ylabel('real(z)')
    plt.yticks()
    plt.xticks()
    plt.grid()
    plt.legend()
    plt.show()

    # Plot theta_diff
    plt.subplot(2, 1, 2)
    plt.plot(t, theta_diff, label='phase difference', color='hotpink')
    plt.title(f"For a phase difference of ${phi_deg[deg]}\degree$ or {np.radians(
     phi_deg[deg]).round(4)} radians")
    plt.legend()
    plt.grid()
    plt.xlabel('Time')
    plt.ylabel(r'$\Theta_1$ - $\Theta_2$')
    plt.show()

############################## Solving ODEs and ploting the oscillator response
     ##############################

# Solving the ODEs
phi1 = np.radians(phi_deg[0])
phi2 = np.radians(phi_deg[1])
sol1 = odeint(system, y0, t, args=(meu, omega1, omega2, A, phi1))
sol2 = odeint(system, y0, t, args=(meu, omega1, omega2, A, phi2))

# For Phase difference of -47 deg
r1, theta1, r2, theta2 = extractSol(sol1)
plotComplexCoupling(r1,theta1,r2,theta2,0)

# For Phase difference of 98 deg
r1, theta1, r2, theta2 = extractSol(sol2)
```

```
plotComplexCoupling(r1,theta1,r2,theta2,1)
```

2. For power coupling:

```
# -*- coding: utf-8 -*-
"""powerCoupling.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1NaBEUhcHc-Kin21KU8I7AW1o_g6jTVoN
"""

import numpy as np
import scipy
from scipy.integrate import odeint
import matplotlib.pyplot as plt

############################## Initialisation ##############################

# Defining variables
meu = 1
omega1 = 5
omega2 = 15
A = 0.2
phi_deg = [-47, 98]
t = np.linspace(0, 100, 10000)

############################## Function Definitions ##############################

# Defining the system of ODEs
def p_system(y, t, meu, omega1, omega2, A, phi):
    r1, r2, theta1, theta2, psi= y
    dr1dt = (meu - np.square(r1))*r1 + (A*(np.power(r2, (omega1/omega2))*np.cos(
    omega1*(theta2/omega2 - theta1/omega1 + phi/(omega1*omega2)))))
    dr2dt = (meu - np.square(r2))*r2 + (A*(np.power(r1, (omega2/omega1))*np.cos(
    omega2*(theta1/omega1 - theta2/omega2 - phi/(omega1*omega2)))))
    dtheta1dt = omega1 + (A*(np.power(r2, (omega1/omega2))/r1)*(np.sin(omega1*(
    theta2/omega2 - theta1/omega1 + phi/(omega1*omega2)))))
    dtheta2dt = omega2 + (A*(np.power(r1, (omega2/omega1))/r2)*(np.sin(omega2*(
    theta1/omega1 - theta2/omega2 - phi/(omega1*omega2)))))
    dpsidt = ((A*(np.power(r2, (omega1/omega2))))/(omega1*r1)) * np.sin(omega1*(
    phi/(omega1*omega2) - psi)) + ((A*(np.power(r1, (omega2/omega1))))/(omega2*r2)
    ) * np.sin(omega2*(phi/(omega1*omega2) - psi))
    return [dr1dt, dr2dt, dtheta1dt, dtheta2dt, dpsidt]

# Defining function to extract solutions after solving the differential eqs
def extractSol(sol):
    psi = sol[:, 2]/omega1 - sol[:, 3]/omega2
    psi_values = sol[:, 4]
    return [psi,psi_values]

# Defining function to plot oscillator response over time
def plotComplexCoupling(psi,psi_values,phi,deg):
    dpsidt_values = np.gradient(psi_values, t)

    # Calculate the expression for sigma, i.e. theta2/w2 - theta1/w1 + phi/(w1*w2)
    expr = psi - phi/(omega1*omega2)

    # Plot normalized phase difference
    plt.figure()
    plt.title(f"For a normalized phase difference of ${phi_deg[deg]}\degree$ or {
    np.radians(phi_deg[deg]).round(4)} radians")
    plt.plot(t, psi, label='Normalized phase difference',color='hotpink')
    plt.grid()
    plt.xlabel('Time(t)')
    plt.ylabel(chr(936))
    plt.legend()
    plt.show()

    # Plot the expression theta2/w2 - theta1/w1 + phi/(w1*w2)
    plt.figure()
    plt.title(f"For a normalized phase difference of ${phi_deg[deg]}\degree$ or {
    np.radians(phi_deg[deg]).round(4)} radians")
    plt.plot(t, expr ,label ='$\sigma$', color='turquoise')
    plt.legend()
    plt.xlabel('Time')
    plt.ylabel('$\sigma$')
    plt.grid()
    plt.show()

############################## Solving ODEs and ploting the oscillator response
    ##############################

# Solving the ODEs
phi1 = np.radians(phi_deg[0])
phi2 = np.radians(phi_deg[1])
y01 = [0.5, 1, 0.5, 0, np.round(phi1/75)]  # Initial conditions for r1, theta1, r2
    , theta2
y02 = [0.5, 1, 0.5, 0, np.round(phi2/75)]  # Initial conditions for r1, theta1, r2
    , theta2
sol1 = odeint(p_system, y01, t, args=(meu, omega1, omega2, A, phi1))
sol2 = odeint(p_system, y02, t, args=(meu, omega1, omega2, A, phi2))

# For Phase difference of -47 deg
psi,psi_values= extractSol(sol1)
plotComplexCoupling(psi,psi_values,phi1,0)

# For Phase difference of 98 deg
psi,psi_values= extractSol(sol2)
plotComplexCoupling(psi,psi_values,phi2,1)
```