BT6270: Computational Neuroscience

Ayush Mukund Jamdar EE20B018

November 12, 2023

1 Assignment 3: Introduction

The assignment aims to simulate Hopf Oscillators for the following coupled cases –

- complex coupling
- power coupling

Hence, we solve the differential equations governing oscillator coupling, which stands as the building block of the much larger and involved, Oscillatory Neural Networks.

2 Coupled Hopf Oscillators

An independent single Hopf Oscillator is governed by the following set of equations

$$\dot{z} = (\mu - |z|^2)z + i\omega z$$

In polar coordinates,

$$\dot{r} = \mu r\left(1 - r^2\right)$$
$$\dot{\theta} = \omega$$

However, when two oscillators interact, we add their weighted effect into the other's governing equation.

$$\dot{z_1} = (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + w_{21}z_2$$
$$\dot{z_2} = (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + w_{12}z_1$$

2.1 Complex Coupling

In Complex Coupling, the weights are complex conjugates.

$$\dot{z_1} = (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + Ae^{i\varphi}z_2$$
$$\dot{z_2} = (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + Ae^{-i\varphi}z_1$$

In polar coordinates,

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

$$\dot{r_2} = (\mu - r_2^2)r_2 + Ar_1\cos\left(\theta_1 - \theta_2 - \varphi\right)$$
$$\dot{\theta_2} = \omega_2 + A\frac{r_1}{r_2}\sin\left(\theta_1 - \theta_2 - \varphi\right)$$

The same set of equations is simulated in Python. We use time steps of 10 milliseconds. Initial r and θ are randomized, hence two simulation runs are demonstrated. Also note that we constrain r to stay positive in code since it is the radial distance.

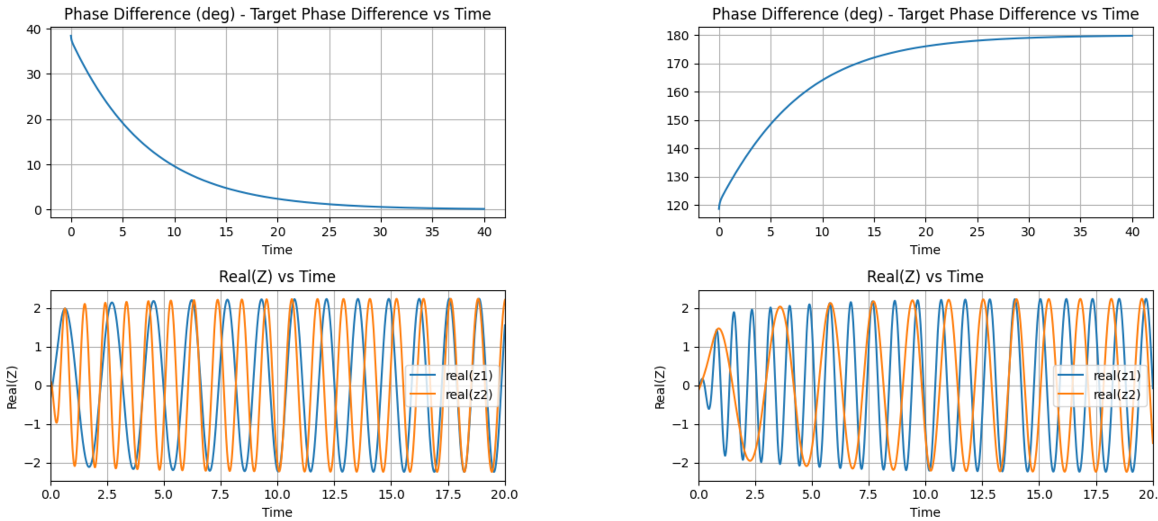
```python
def complex_coupling(t_sim, phase_diff, magnitude, omega_1, omega_2):
    """
    Phase Difference is in Degrees
    """
    mu = 1
    dt = 0.01
    time_sampled = np.arange(0, t_sim, dt)

    # initialize r and theta for both oscillators
    r_1 = np.zeros(len(time_sampled))
    r_2 = np.zeros(len(time_sampled))
    theta_1 = np.zeros(len(time_sampled))
    theta_2 = np.zeros(len(time_sampled))

    r_1[0] = np.random.uniform(0, 1)
    r_2[0] = np.random.uniform(0, 1) # to avoid 0/0

    theta_1[0] = np.random.uniform(0, 1) * 360
    theta_2[0] = np.random.uniform(0, 1) * 360

    # forward Euler method
    for t in range(1, len(time_sampled)):
        r_1_dot = (mu - r_1[t-1]**2) * r_1[t-1] + magnitude * r_2[t-1] * np.cos(np.deg2rad(theta_2[t-1] - theta_1[t-1] + phase_diff))
        r_2_dot = (mu - r_2[t-1]**2) * r_2[t-1] + magnitude * r_1[t-1] * np.cos(np.deg2rad(theta_1[t-1] - theta_2[t-1] - phase_diff))
        theta_1_dot = omega_1 + magnitude * np.sin(np.deg2rad(theta_2[t-1] - theta_1[t-1] + phase_diff)) * (r_2[t-1] / r_1[t-1])
        theta_2_dot = omega_2 + magnitude * np.sin(np.deg2rad(theta_1[t-1] - theta_2[t-1] - phase_diff)) * (r_1[t-1] / r_2[t-1])

        if r_1[t-1] + r_1_dot * dt < 0:
            r_1[t] = r_1[t-1]
        else:
            r_1[t] = r_1[t-1] + r_1_dot * dt

        if r_2[t-1] + r_2_dot * dt < 0:
            r_2[t] = r_2[t-1]
        else:
            r_2[t] = r_2[t-1] + r_2_dot * dt

        theta_1[t] = theta_1[t-1] + theta_1_dot * dt
        theta_2[t] = theta_2[t-1] + theta_2_dot * dt

    return time_sampled, r_1, r_2, theta_1, theta_2
```

The simulation results for two runs with $A = 4$, $\varphi = -47°$ are collated in Figure 1 and $\varphi = 98°$ in Figure 2. **Please note** that since trigonometric functions are involved, the phase difference settles to $n\pi$. In each of the plots, the y-axis has $(\theta_1 - \theta_2 - \varphi)\%\pi$ where % denotes the modulo operator.

Coupling Coefficients (Note that the amplitude is arbitrary, it can be shown analytically to work for any A) –

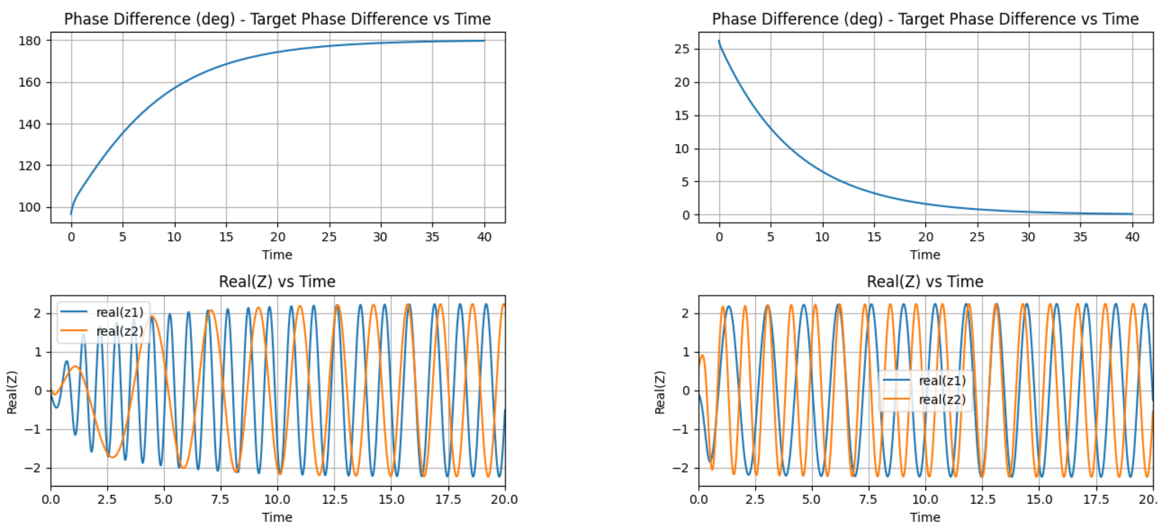
$$w_{21} = 4e^{-i47°}$$
$$w_{12} = 4e^{i47°}$$

$$w_{21} = 4e^{i98°}$$
$$w_{12} = 4e^{-i98°}$$



(a) Run-1 (b) Run-2

Figure 1: $\varphi = -47°$ for Complex Coupling. Observe how the phase difference settles after some time and the same is reflected in the waveform too.



(a) Run-1 (b) Run-2

Figure 2: $\varphi = 98°$ for Complex Coupling.

2.2 Power Coupling

Here, the governing differential equations become –

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

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

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

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

Here too, we use random initialization and hence, show two runs. The code is as follows –

```python
        def power_coupling(t_sim, phase_diff, magnitude, omega_1, omega_2):
            """
            Phase Difference is in Radians
            """
            mu = 1
            dt = 0.01
            time_sampled = np.arange(0, t_sim, dt)

            # initialize r and theta for both oscillators
            r_1 = np.zeros(len(time_sampled))
            r_2 = np.zeros(len(time_sampled))
            theta_1 = np.zeros(len(time_sampled))
            theta_2 = np.zeros(len(time_sampled))

            r_1[0] = np.random.uniform(0, 1)
            r_2[0] = np.random.uniform(0, 1) # to avoid 0/0

            theta_1[0] = np.random.uniform(0, 1) * 2 * np.pi
            theta_2[0] = np.random.uniform(0, 1) * 2 * np.pi

            # forward Euler method
            for t in range(1, len(time_sampled)):

                r_1_dot = (
                            mu - r_1[t-1]**2) * r_1[t-1] + \
                            magnitude * (r_2[t-1] ** (omega_1/omega_2)) * \
                            np.cos(omega_1 * ((theta_2[t-1] / omega_2) - \
                            (theta_1[t-1] / omega_1) + (phase_diff/(omega_1*omega_2)))
                        )

                r_2_dot = (
                            mu - r_2[t-1]**2) * r_2[t-1] + \
                            magnitude * (r_1[t-1] ** (omega_2/omega_1)) * \
                            np.cos(omega_2 * ((theta_1[t-1] / omega_1) - \
                            (theta_2[t-1] / omega_2) - (phase_diff/(omega_1*omega_2)))
                        )

                theta_1_dot = (
                            omega_1 + magnitude * \
                            np.sin(omega_1 * ((theta_2[t-1] / omega_2) - \
                            (theta_1[t-1] / omega_1) + (phase_diff/(omega_1*omega_2)))) * \
                            ((r_2[t-1] ** (omega_1/omega_2)) / r_1[t-1])
                        )

                theta_2_dot = (
                            omega_2 + magnitude * \
                            np.sin(omega_2 * ((theta_1[t-1] / omega_1) - \
                            (theta_2[t-1] / omega_2) - (phase_diff/(omega_1*omega_2)))) * \
                            ((r_1[t-1] ** (omega_2/omega_1)) / r_2[t-1])
                        )

                if r_1[t-1] + r_1_dot * dt < 0:
                    r_1[t] = r_1[t-1]
                else:
```

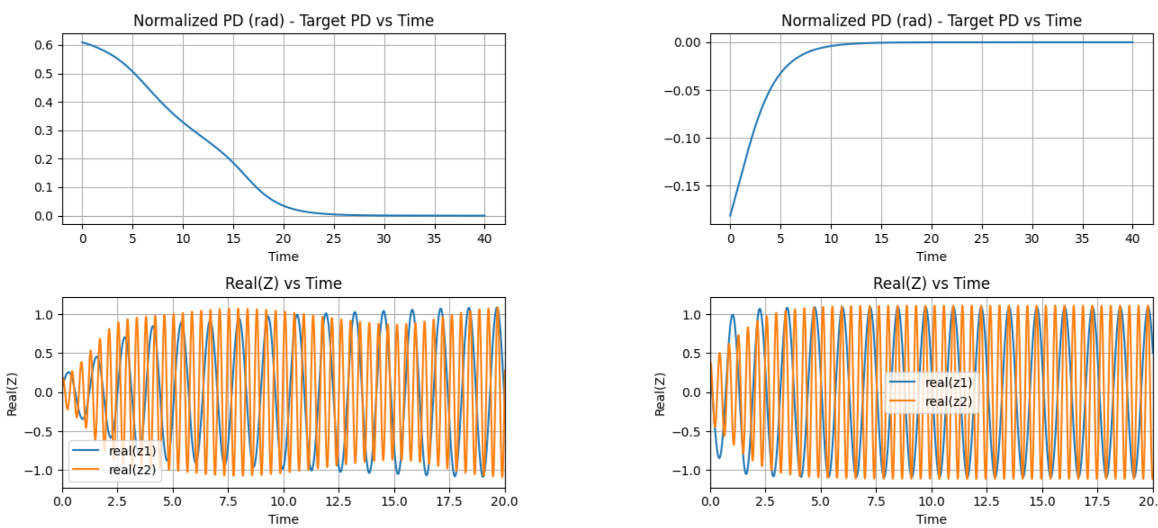
```python
                    r_1[t] = r_1[t-1] + r_1_dot * dt

                if r_2[t-1] + r_2_dot * dt < 0:
                    r_2[t] = r_2[t-1]
                else:
                    r_2[t] = r_2[t-1] + r_2_dot * dt

                theta_1[t] = (theta_1[t-1] + theta_1_dot * dt)
                theta_2[t] = (theta_2[t-1] + theta_2_dot * dt)

            return time_sampled, r_1, r_2, theta_1, theta_2
```

Finally, we plot $\left(\frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2} - \frac{\varphi}{\omega_2\omega_1}\right)$ against time, in Figures 3 and 4 for different normalized phase difference.

Coupling Coefficients (Note that the amplitude is arbitrary, it can be shown analytically to work for any A) –

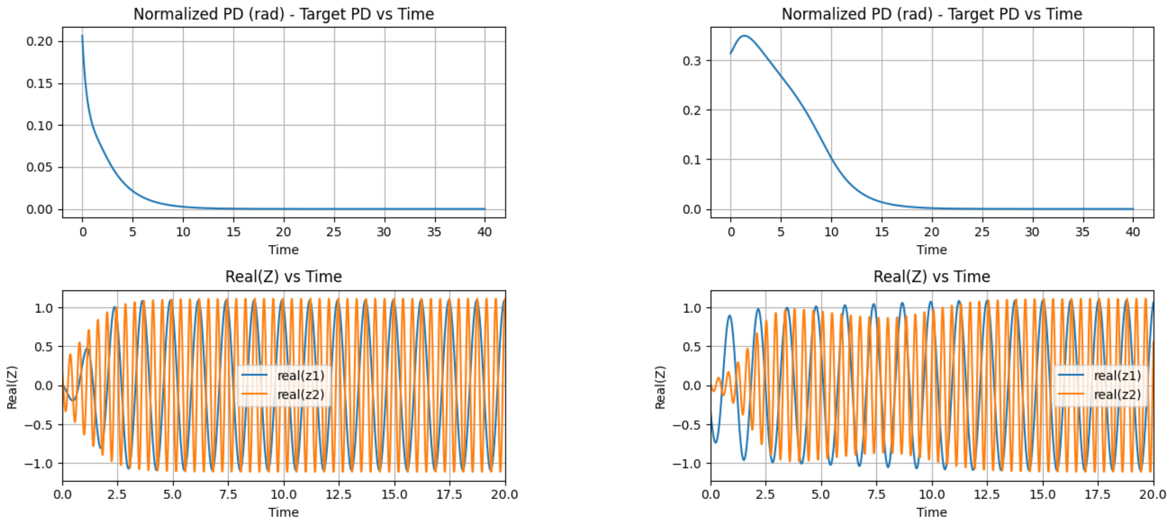
$$w_{21} = 0.2e^{-i47^\circ}$$
$$w_{12} = 0.2e^{i47^\circ}$$

$$w_{21} = 0.2e^{i98^\circ}$$
$$w_{12} = 0.2e^{-i98^\circ}$$



(a) Run-1 (b) Run-2

Figure 3: $\varphi = -47^\circ$ for Power Coupling. Observe how the phase difference settles after some time and the same is reflected in the waveform too.



(a) Run-1

(b) Run-2

Figure 4: $\varphi = 98°$ for Power Coupling.

3 Conclusion

Through this assignment, we calculated coupling coefficients between oscillators and simulated the coupled behaviour. This gave a deep insight into the system from its fundamentals.