# Synchronization of Metronomes and Chimera States

Shreesh Kulkarni
2017B5A70279G
In collaboration with Abhishek Patwardhan
Under the guidance of Dr. Gaurav Dar


May 2020

# Abstract

Building phase models is a powerful technique to study systems involving various coupled oscillators. In this study oriented project, our aim was to first learn the full set of numerical techniques involved in building phase models, reproduce previously found results, then apply the techniques to systems with metronomes.
We first reproduced the phase models in Izhikevich's book [Izh07] for the Andronov-Hopf and the van der Pol oscillators. Secondly, we built the phase model for the experimental system described by Pantaleone [Pan02]. Our results agreed with the experiment to a large extent. We were able to accurately predict the stability of the in-phase and the anti-phase synchronization states as a function of the coupling parameter.
Thirdly, we attempted to generate chimera states using the system studied experimentally by Martens et al. [al13]. While we were not able to generate them consistently, we did find that the oscillators in the system are strongly coupled, as a result we doubt if a phase model can be built for them.

# Acknowledgement

I thank my advisor Dr. Gaurav Dar for helping and guiding me throughout the process of working on this project. I also thank Abhishek, my collaborator for helping me with the coding aspects of the project, and also for verifying my results. I thank the engineers at MATLAB for providing me with a great coding experience. Finally, I thank my family for supporting me throughout this venture.

# Contents

# Chapter 1

# Introduction to the concepts

Synchronization of periodic systems is an important part of understanding systems involving coupled oscillators, like the beating heart, circadian rhythms in the brain, transient weather patterns and chemical oscillations, to name a few. To analyze such systems, we set up experiments and build mathematical techniques that aim to explain their results. One such mathematical technique is called the Phase Model.

## 1.1 Phase

The 'phase' in the phase model forms the basis of the whole technique. The oscillators we are studying all have one important property: given any non-trivial initial condition, the oscillator tends to follow a fixed path after evolving for a long time. In the oscillator's phase space, such a path is defined by a closed curve which we call the attractive limit cycle of the oscillator. Here is one such limit cycle, for the van der Pol oscillator:

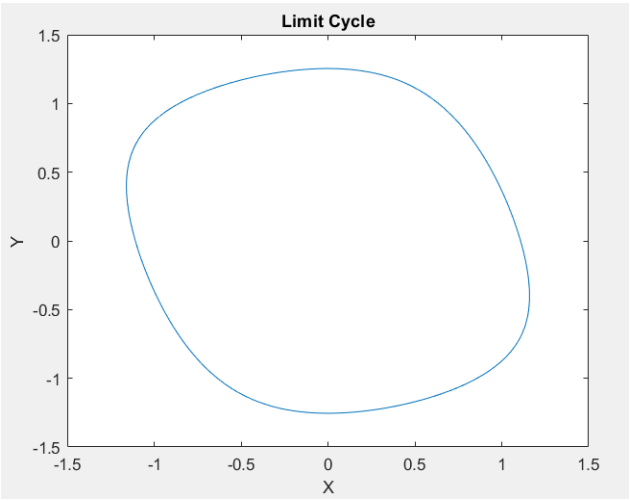


Figure 1.1: Limit Cycle

Such fixed paths loop back over themselves. Since these paths are topologically equivalent to a circle, we can assign each point on them an angle, with one point having angle 0, and every other point having an angle in the range of $(0, 2\pi)$. This "angle" is what we call the phase of every point on the limit cycle. Since $(0, T)$ maps onto $(0, 2\pi)$, (where T is the time period of the oscillator), we can use time and phase as independent parameters interchangeably in the course of this discussion.

## 1.2 Isochrons

The concept of phase can be extended to points that do not lie on the limit cycle. We first note the fact that every point on the 2D plane converges onto the limit cycle.

The locus of all points that converge onto the same point on a limit cycle with a given phase is called an isochron having the phase of said point. In essence, after a sufficiently long time, all those points behave in the same manner as the point on the limit cycle they converge to; hence they can be said to have the same phase as the point they converge to. By defining isochrons, we have extended the idea of phase to the rest of the 2D space.

Here is an example of isochrons near the limit cycle shown above. It is not fully accurate, and only serves to show their qualitative nature.

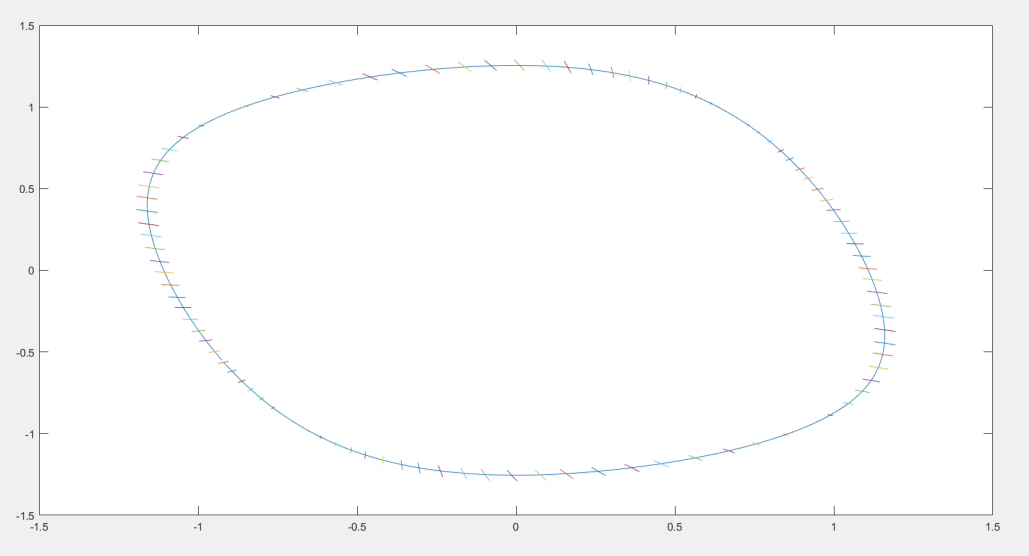


Figure 1.2: Isochrons

## 1.3 Phase Response Curves

Coupled oscillators continuously influence and perturb each other as long as they are moving. We can gain an understanding of the effects of such perturbations by building Phase Response Curves (PRCs).

At each point on the limit cycle (i.e. for each phase), we perturb the system along the x and the y axes of the phase space by a given amount, and let it evolve for a long time. Once a sufficient amount of time has passed, we check the new phase of the perturbed point. The difference between the final and initial phases is the Phase Response. Plotting this response against the initial phase of each point gives us the Phase Response Curve.

Shown below is the Phase Response Curve for an Andronov Hopf oscillator.

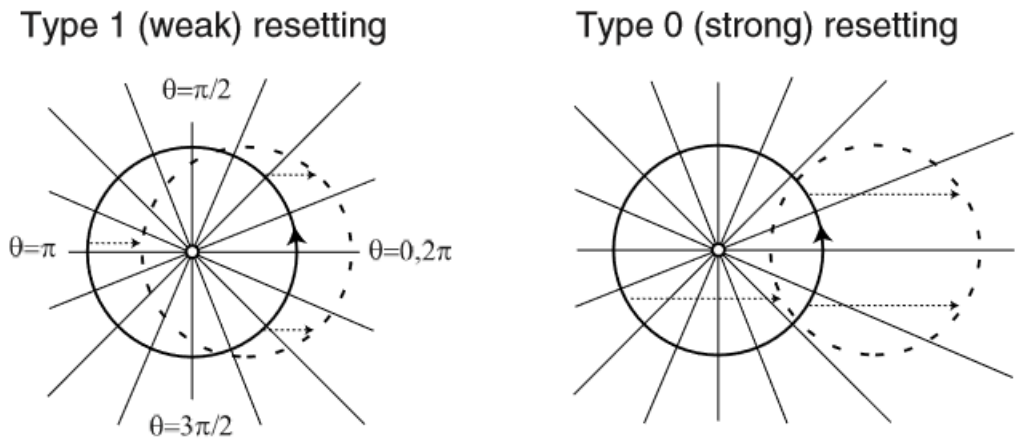


Figure 1.3: PRCs for small and large perturbations

The two types of PRCs shown here are caused by perturbations small and large as compared to the span of the limit cycle. We will only be dealing with the Type 1 PRCs, since we will not be considering systems with large perturbations.

Now that we are familiar with the concepts, we can move forward to the various methods that are used to build Phase Models.

# Chapter 2

# Phase Model

We are studying weakly coupled systems of the type:

$$\dot{x} = f(x) + \varepsilon\, p(t) \tag{2.1}$$

where $\dot{x} = f(x)$ is the equation for a free-running oscillator, and $\varepsilon p(t)$ is a comparatively small interaction term. $\varepsilon$ is a positive parameter that measures the overall strength of the coupling.

The solutions to such equations can be obtained by solving a differential equation of the form:

$$\dot{\theta} = 1 + \varepsilon\, PRC(\theta)\, p(t) + o(\varepsilon)$$

Which gives us the behavior of the phase, which is defined all over the phase space. The $o(\varepsilon)$ term is the error in the first order approximation involving $\varepsilon$. The $PRC(\theta)$ is the infinitesimal phase response.

## 2.1 Winfree's Approach

Winfree's approach assumes that the isochrons near the limit cycle are all parallel. Hence any perturbation to a point on the limit cycle results in a phase shift proportional to the perturbation.

$$PRC(\theta, A) = Z(\theta)A$$

where $Z(\theta) = \partial PRC(\theta, A)/\partial A$ at $A = 0$ is the linear response, or the *infinitesimal phase response* near the limit cycle.

## 2.2 Kuramoto's Approach

Kuramoto's approach uses the equality:

$$grad\,(\theta) \cdot f(x) = 1$$

to give the phase model:

$$\dot{\theta} = 1 + \varepsilon\, grad\,(\theta) \cdot p(t)$$

Kuramoto's approach is equivalent to Winfree's, but we will not use it due to the difficulty in calculating the gradient of phase. Instead, we have used the Malkin's approach.

## 2.3 Malkin's Approach

*Malkin's Theorem:* Consider an unperturbed oscillator having a limit cycle of period $T$. Its phase is given by equation:

$$\dot{\theta} = 1 + \varepsilon\, Q(\theta) \cdot p(t) \tag{2.2}$$

where $Q(\theta)$ is the solution to the linear adjoint equation:

$$\dot{Q} = -\{Df(x(t))\}^T Q \qquad with \quad Q(0) \cdot f(x(0)) = 1 \tag{2.3}$$

where $Df(x(t))$ is the Jacobian of $f$ at the point $x(t)$ on the limit cycle. The constant of motion in this case is the dot product $Q(t)\,.\,f(x(t))$, which stays equal to 1.

We will be using this method to find the $Q(t)$, for it is very easy to apply it numerically in MATLAB.

Now that we have seen a numerical method of building the phase model, we can simplify the calculation by expressing the above equation (2.2) in terms of phase difference between interacting oscillators. Such an expression tells us the synchronization state of the system in a clearer manner.

## 2.4 Phase Model of Coupled Oscillators

Consider $n$ weakly coupled oscillators whose equations are given by:

$$\dot{x_i} = f_i(x_i) + \varepsilon \sum_{j=1}^{n} g_{ij}(x_i(\theta_i), x_j(\theta_j))$$

The phase model for this set of equations is given by:

$$\dot{\theta_i} = 1 + \varepsilon\, Q_i(\theta_i) \cdot \sum_{j=1}^{n} g_{ij}(x_i(\theta_i), x_j(\theta_j))$$

This kind of system has two modes of evolution of phase: a fast evolution and a slow evolution. To separate out fast and slow variations in phase, we represent the phases as:

$$\theta_i(t) = t + \phi_i,$$

Where $\phi_i$ is the deviation of each oscillator from its free running phase due to the coupling. Using this substitution and then applying the method of averaging, we get:

$$\dot{\phi_i} = \varepsilon\, \omega_i + \varepsilon \sum_{j \neq i}^{n} H_{ij}(\phi_j - \phi_i) \tag{2.4}$$

Where,

$$H_{ij}(\phi_j - \phi_i) = \frac{1}{T} \int_0^T Q_i(t) \cdot g_{ij}(x_i(t), x_j(t + \phi_j - \phi_i))dt \tag{2.5}$$

where $T$ is the time period of the free running oscillator, and $\omega_i = H_{ii}(\phi_i - \phi_i) = H_{ii}(0)$. Calculating the $H_{ij}$'s numerically from the $Q_i$'s and the interaction terms builds the phase model for us, which can be then solved to understand the evolution of the phase differences of oscillators with time.

## 2.5 Two Identical Oscillators

Consider a system with $n = 2$, i.e. two coupled oscillators. We work with "slow" time, $\tau = \varepsilon t$, which transforms the equation (2.4) into:

$$\phi_1' = \omega_1 + H_1(\phi_2 - \phi_1)$$
$$\phi_2' = \omega_2 + H_2(\phi_1 - \phi_2)$$

Where $' = d/d\tau$ is the derivative with respect to slow time. Let $\chi = \phi_2 - \phi_1$ be the phase difference between the two oscillators. Since the two oscillators are identical, we have $\omega_1 = \omega_2$ and $H_1(\chi) = H_2(\chi) = H(\chi)$. The set of equations above is transformed to:

$$\chi' = G(\chi) \tag{2.6}$$

where $G(\chi) = H(-\chi) - H(\chi)$. The roots of $G(\chi)$ are the fixed points of the phase difference. The slope at these fixed points tells us their stability. Hence by numerically calculating $G(\chi)$, we can get a qualitative understanding of the system and predict the state of synchronization of the two oscillators.

# Chapter 3

# Numerical Simulations and the Methodology used to obtain Results

There are three parts to our results: first checking that our numerical methods are correct by comparing our results with those of *Izhikevich* [Izh07], and then applying said numerical methods to two systems, first to the system of two coupled metronomes [Pan02], then to the system of two coupled populations of metronomes [al13].

Numerical calculations have been done using *MATLAB, Python* and *Julia*. The plots were made using *MATLAB*'s inbuilt plotter, and *Python*'s *matplotlib* library. We have used a standard version of the *Runge-Kutta 4 (RK4)* method to solve ODEs, and the *Simpson's method* for solving integrals (and not any inbuilt functions). Wherever a Jacobian calculation is involved, we have found it by hand (i.e. we have plugged in the explicit expression), as opposed to *Izhikevich's* use of *Euler's method*, although we have checked that the latter method also gives very accurate results. The time steps that we have used range from $10^{-4}$ to $5 \cdot 10^{-3}$, sometimes even as high as $10^{-2}$, but the latter gives fairly accurate results, and reducing the time steps does not affect the qualitative nature of the plots much.

# Chapter 4

# Comparison of results with *Izhikevich*

We have built the phase models for two simple limit cycle oscillators: van der Pol and Andronov-Hopf. The equations for the free running oscillators are:

$$\begin{aligned} &\dot{x} = x - x^3 - y \qquad : \quad van\ der\ Pol \\ &\dot{y} = x \end{aligned}$$

$$\dot{z} = (1+i)z - z|z|^2,\ z\ \epsilon\ \mathbb{C} \qquad : \quad Andronov - Hopf$$

The interaction term for each oscillator is:

$$g_{ij}(x_i, x_j) = (x_{j1} - x_{i1}, 0),$$

where $x_{i1}$ is the first component of the phase space vector of a given oscillator.

## 4.1 Van der Pol Oscillator

The $Q(t)$ and $H_{ij}(\phi_j - \phi_i)$ for the van der Pol oscillator that we found numerically are given below.

## 4.2 Andronov-Hopf Oscillator

The $Q(t)$ and $H_{ij}(\phi_j - \phi_i)$ for the Andronov-Hopf oscillator that we found numerically are given below.

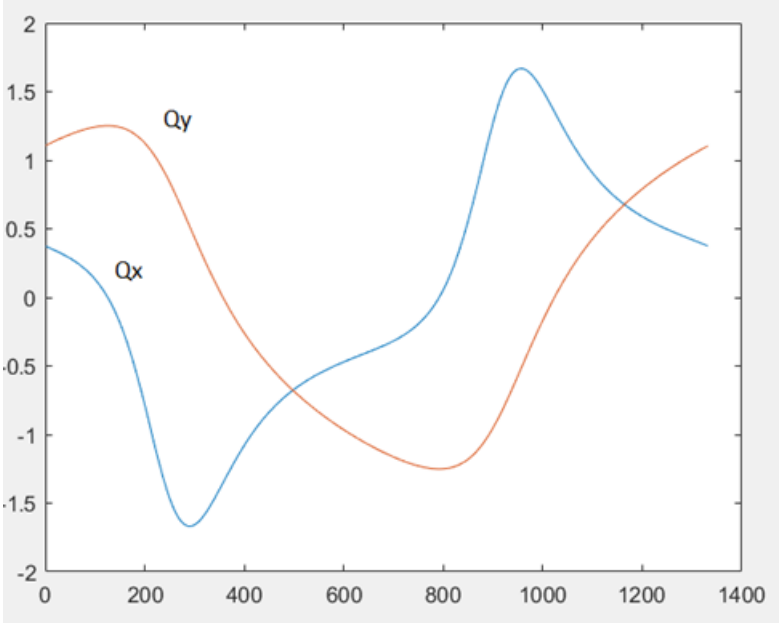


Figure 4.1: $Q(t)$ Result

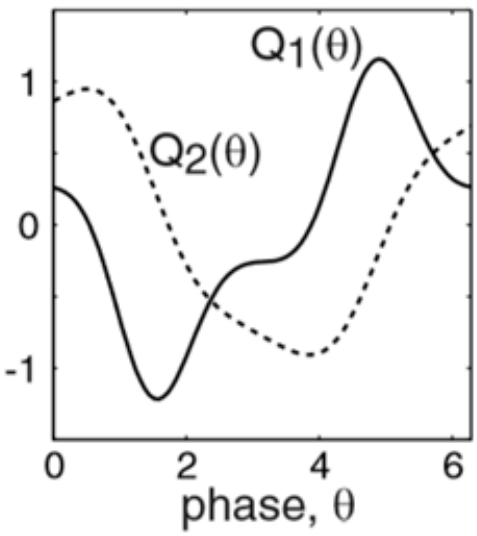


Figure 4.2: $Q(t)$ Result from Izhikevich

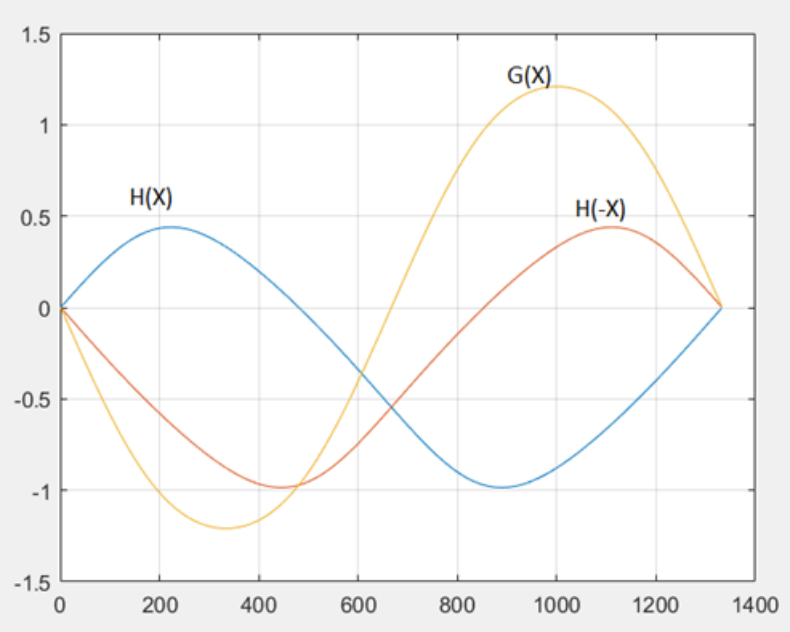


Figure 4.3: $H(\chi)$ Result

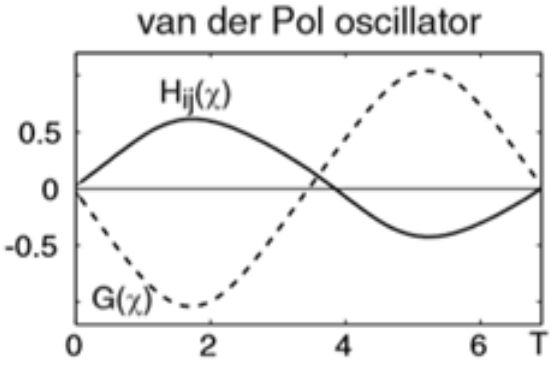


Figure 4.4: $H(\chi)$ Result from Izhikevich

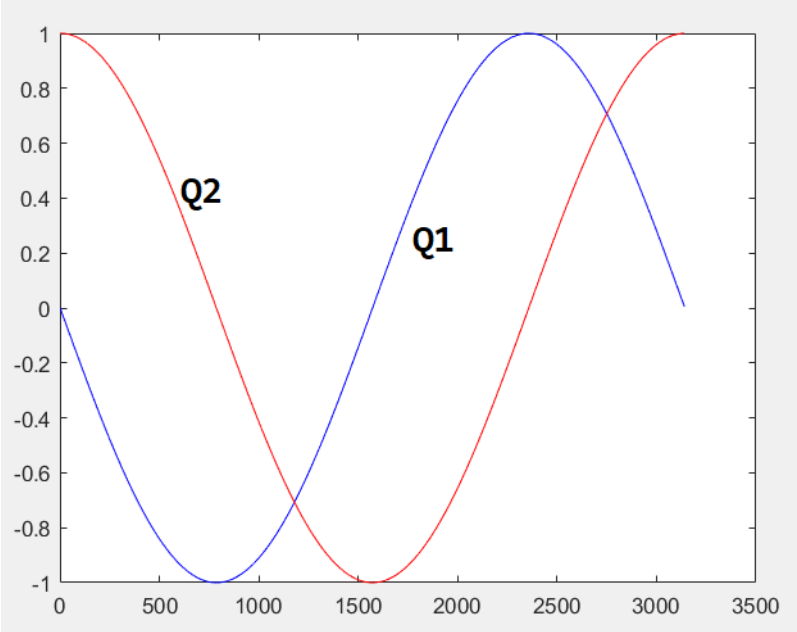


Figure 4.5: $Q(t)$ Result

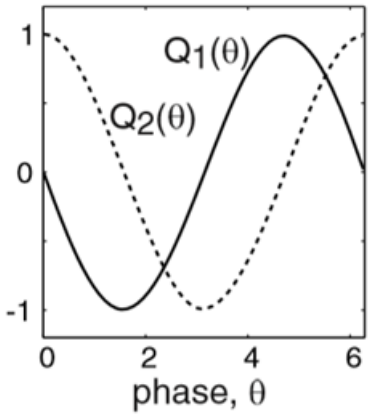


Figure 4.6: $Q(t)$ Result from Izhikevich

## 4.3 Inferences

As can be seen, the results are very close to the results published in *Izhikevich*. Wherever possible, we have tried to reconcile differences between the two set of results by using lower time steps and more accurate numerical techniques. After getting these results, we gained confidence in our methods and with the help of our advisor Dr. Dar, we decided to apply them to the next problem at hand.

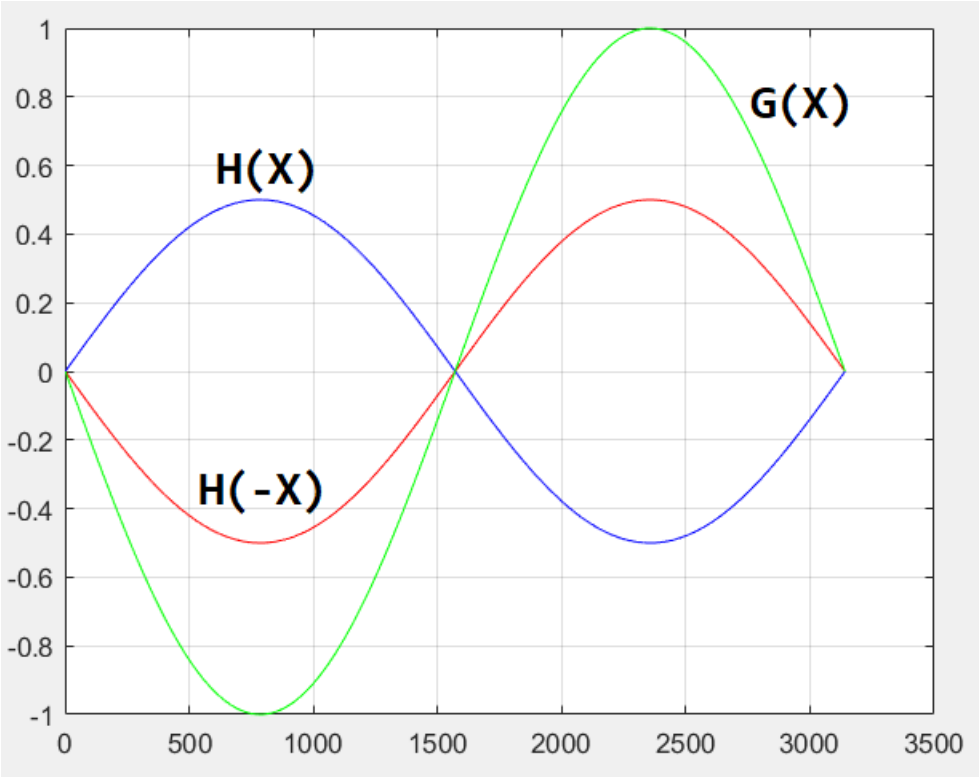


Figure 4.7: $H(\chi)$ Result

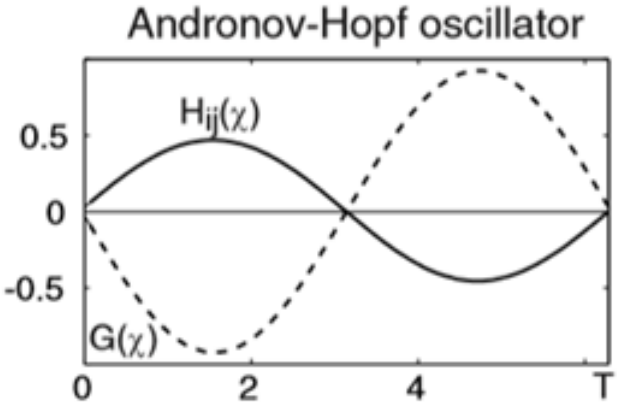


Figure 4.8: $H(\chi)$ Result from Izhikevich

# Chapter 5

# Two Coupled Metronomes

The system we are studying has two metronomes sitting on a wooden plank that is supported by two soda cans. The metronomes are non-linear limit cycle oscillators of the *van der Pol* type. We have ignored the damping in their oscillations as their springs uncoil. The wooden plank couples the two metronomes, which facilitates their synchronization. The equations of motion are given by: (from Pantaleone [Pan02])

$$\frac{d^2\theta_1}{d\tau^2} + (1+\Delta)\, sin\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\frac{d\theta_1}{d\tau} - \beta\, cos\,\theta_1 \frac{d^2}{d\tau^2}\left(sin\,\theta_1 + sin\,\theta_2\right) = 0 \tag{5.1}$$

$$\frac{d^2\theta_2}{d\tau^2} + (1-\Delta)\, sin\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\frac{d\theta_2}{d\tau} - \beta\, cos\,\theta_2 \frac{d^2}{d\tau^2}\left(sin\,\theta_1 + sin\,\theta_2\right) = 0 \tag{5.2}$$

The free running oscillator is of the form:

$$\frac{d^2\theta}{d\tau^2} + sin\,\theta + \mu\left[\left(\frac{\theta}{\theta_0}\right)^2 - 1\right]\frac{d\theta}{d\tau} - \beta\, cos\,\theta \frac{d^2}{d\tau^2} sin\,\theta = 0, \tag{5.3}$$

which is a limit cycle oscillator. Below is the limit cycle in the phase space of $\theta$ *vs.* $\omega$:
Here the time is "slow", i.e $\tau = \omega t$, where $\omega = 10.9Hz$ is the average frequency of the free running oscillator with small oscillations. $\beta$ is the dimensionless coupling parameter, and depends on the nature of the metronomes, as well as the mass of the wooden plank. For the experimental setup, $\beta = 0.011$. $\Delta = (\omega_1 - \omega_2)/\omega$ is the frequency difference between the two metronomes relative to their free-running frequency. In the experiment, it is of the order $10^{-3}$ due to non-zero physical tolerances We have taken it to be zero, i.e. we only consider perfectly identical metronomes. $\mu$ is the van der Pol parameter, which is taken to be 0.01 by the author to match their experimental observations. $\theta_0$ is half the maximum amplitude of the metronomes, which is taken to be 0.39rad, or 45 degrees.

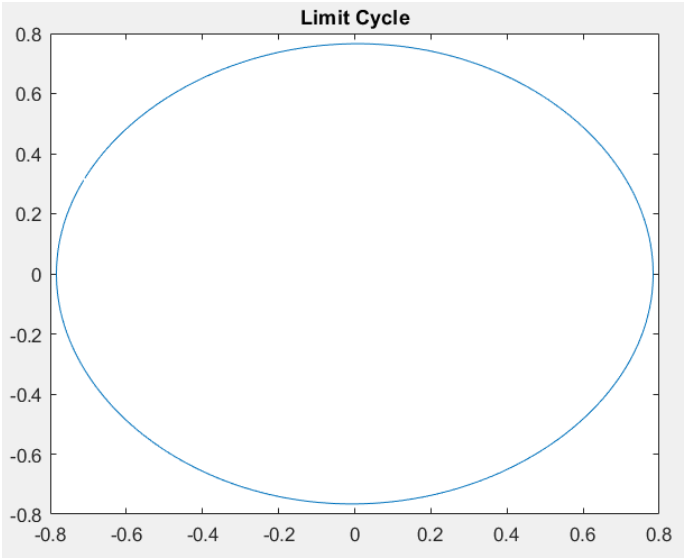


Figure 5.1: Limit Cycle of a Free Running Metronome

## 5.1 Explicit Equations

The above equations (3.1), (3.2) in explicit column-vector form are:

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}' = \begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix} \tag{5.4}$$

$$\begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix}' = \frac{-1}{1-\beta(cos^2\theta_1 + cos^2\theta_2)} \begin{bmatrix}(1-\beta cos^2\theta_2) & \beta\cos\theta_1\cos\theta_2\\ \beta\cos\theta_1\cos\theta_2 & (1-\beta\cos^2\theta_1)\end{bmatrix} \cdot \begin{bmatrix}k_1\\ k_2\end{bmatrix} \tag{5.5}$$

where

$$k_1 = (1+\Delta)\,sin\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\omega_1 + \beta\cos\theta_1\,(\omega_1^2\,sin\,\theta_1 + \omega_2^2\,sin\,\theta_2)$$

$$k_2 = (1-\Delta)\,sin\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\omega_2 + \beta\cos\theta_2\,(\omega_1^2\,sin\,\theta_1 + \omega_2^2\,sin\,\theta_2)$$

Here $' = d/d\tau$ is the derivative with respect to slow time.

The equations for the free-running metronome are:

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \omega_1\\ \omega_2\end{bmatrix}' = \begin{bmatrix}\omega_1\\ \omega_2\\ \frac{1}{1-\beta\,cos^2\theta_1}\,C_1\\ \frac{1}{1-\beta\,cos^2\theta_2}\,C_2\end{bmatrix} \tag{5.6}$$

The interaction terms for $\theta$ are both zero. The interaction terms for $\omega$ are:

$$\omega_1 : \frac{\beta^2 \, cos^2\,\theta_1 \, cos^2\,\theta_2}{(1-\beta\, cos^2\,\theta_1)\,[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\, C_1 - \frac{1-\beta\, cos^2\,\theta_2}{[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\,\beta\, cos\,\theta_1\, sin\,\theta_2\,\omega_2^2 + \frac{\beta\, cos\,\theta_1\, cos\,\theta_2}{[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\,(C_2 - \beta\, cos\,\theta_2\, sin\,\theta_1\,\omega_1^2)$$

$$\omega_2 : \frac{\beta^2 \, cos^2\,\theta_1 \, cos^2\,\theta_2}{(1-\beta\, cos^2\,\theta_1)\,[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\, C_2 - \frac{1-\beta\, cos^2\,\theta_1}{[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\,\beta\, cos\,\theta_2\, sin\,\theta_1\,\omega_1^2 + \frac{\beta\, cos\,\theta_1\, cos\,\theta_2}{[1-\beta\,(cos^2\,\theta_1 + cos^2\,\theta_2)]}\,(C_1 - \beta\, cos\,\theta_1\, sin\,\theta_2\,\omega_2^2)$$

where

$$C_1 = -\left\{(1+\Delta)\, sin\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\omega_1 + \left(\frac{\beta}{2}\right) sin\,(2\theta_1)\,\omega_1^2\right\}$$

$$C_2 = -\left\{(1-\Delta)\, sin\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\omega_2 + \left(\frac{\beta}{2}\right) sin\,(2\theta_2)\,\omega_2^2\right\}$$

## 5.2 Small Angle Approximation

To simplify the above equations, we first apply the small angle approximation: $\theta_1, \theta_2 \approx 0$, which gives us $sin\,\theta \approx 0$ and $cos\,\theta \approx 1$. The resulting equations are:

$$\theta_1' = \omega_1$$

$$\omega_1' = -\frac{1-\beta}{1-2\beta}\left\{(1+\Delta)\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\omega_1\right\} - \frac{\beta}{1-2\beta}\,\theta_1\,\omega_1^2 - \frac{\beta}{1-2\beta}\left\{(1-\Delta)\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\omega_2\right\} - \frac{\beta}{1-2\beta}\,\theta_2\,\omega_2^2$$

$$\theta_2' = \omega_2$$

$$\omega_2' = -\frac{1-\beta}{1-2\beta}\left\{(1-\Delta)\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\omega_2\right\} - \frac{\beta}{1-2\beta}\,\theta_2\,\omega_2^2 - \frac{\beta}{1-2\beta}\left\{(1+\Delta)\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\omega_1\right\} - \frac{\beta}{1-2\beta}\,\theta_1\,\omega_1^2$$

The free running oscillator terms are:

$$\begin{pmatrix}\theta_1\\ \omega_1\\ \theta_2\\ \omega_2\end{pmatrix}' = \begin{pmatrix}\omega_1\\ -\frac{1-\beta}{1-2\beta}\left\{(1+\Delta)\,\theta_1 + \mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2 - 1\right]\omega_1\right\} - \frac{\beta}{1-2\beta}\,\theta_1\,\omega_1^2\\ \omega_2\\ -\frac{1-\beta}{1-2\beta}\left\{(1-\Delta)\,\theta_2 + \mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2 - 1\right]\omega_2\right\} - \frac{\beta}{1-2\beta}\,\theta_2\,\omega_2^2\end{pmatrix} \tag{5.7}$$

The interaction terms are:

$$\begin{pmatrix} 0 \\ -\frac{\beta}{1-2\beta}\left\{(1-\Delta)\,\theta_2+\mu\left[\left(\frac{\theta_2}{\theta_0}\right)^2-1\right]\omega_2\right\}-\frac{\beta}{1-2\beta}\,\theta_2\,\omega_2^2 \\ 0 \\ -\frac{\beta}{1-2\beta}\left\{(1+\Delta)\,\theta_1+\mu\left[\left(\frac{\theta_1}{\theta_0}\right)^2-1\right]\omega_1\right\}-\frac{\beta}{1-2\beta}\,\theta_1\,\omega_1^2 \end{pmatrix} \tag{5.8}$$

## 5.3 Solving the Exact Equations

Using MATLAB, we solved the equations (3.4),(3.5) numerically for the parameters: $\beta = 0.011$, $\Delta = 0$, $\mu = 0.01$, $\theta_0 = 0.39$. The time step used was $5 \cdot 10^{-3}s$, and we evolved it for up to $1500 - 3000s$. These time values are in slow time, hence they correspond to real time steps of about $5 \cdot 10^{-4}$ and evolution time of around $150 - 300s$. The fourth figure uses a value of $\beta = 0.15$, the reason for doing so has been mentioned just after it.

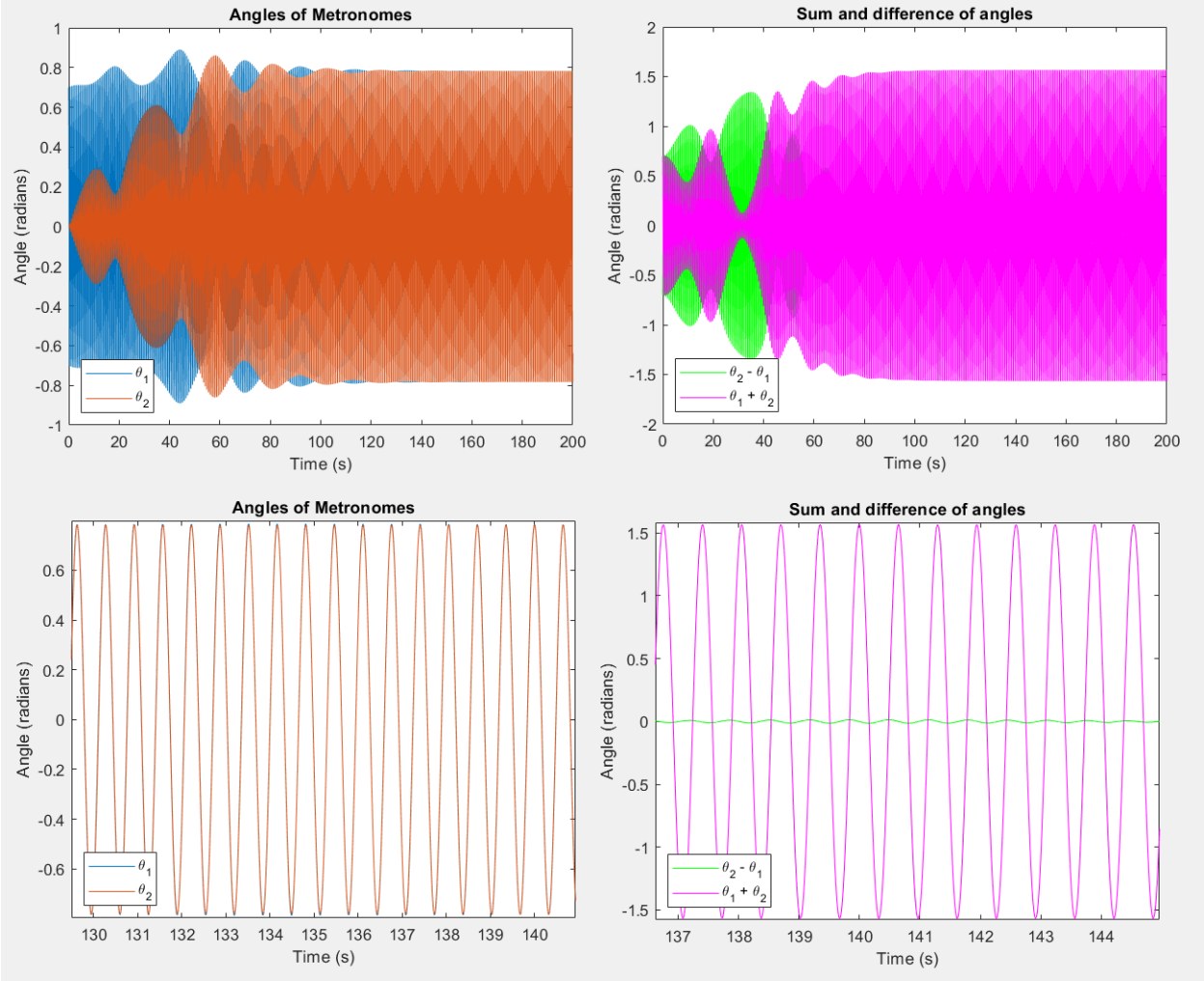


Figure 5.2: Initial phase difference of around $\pi/2$

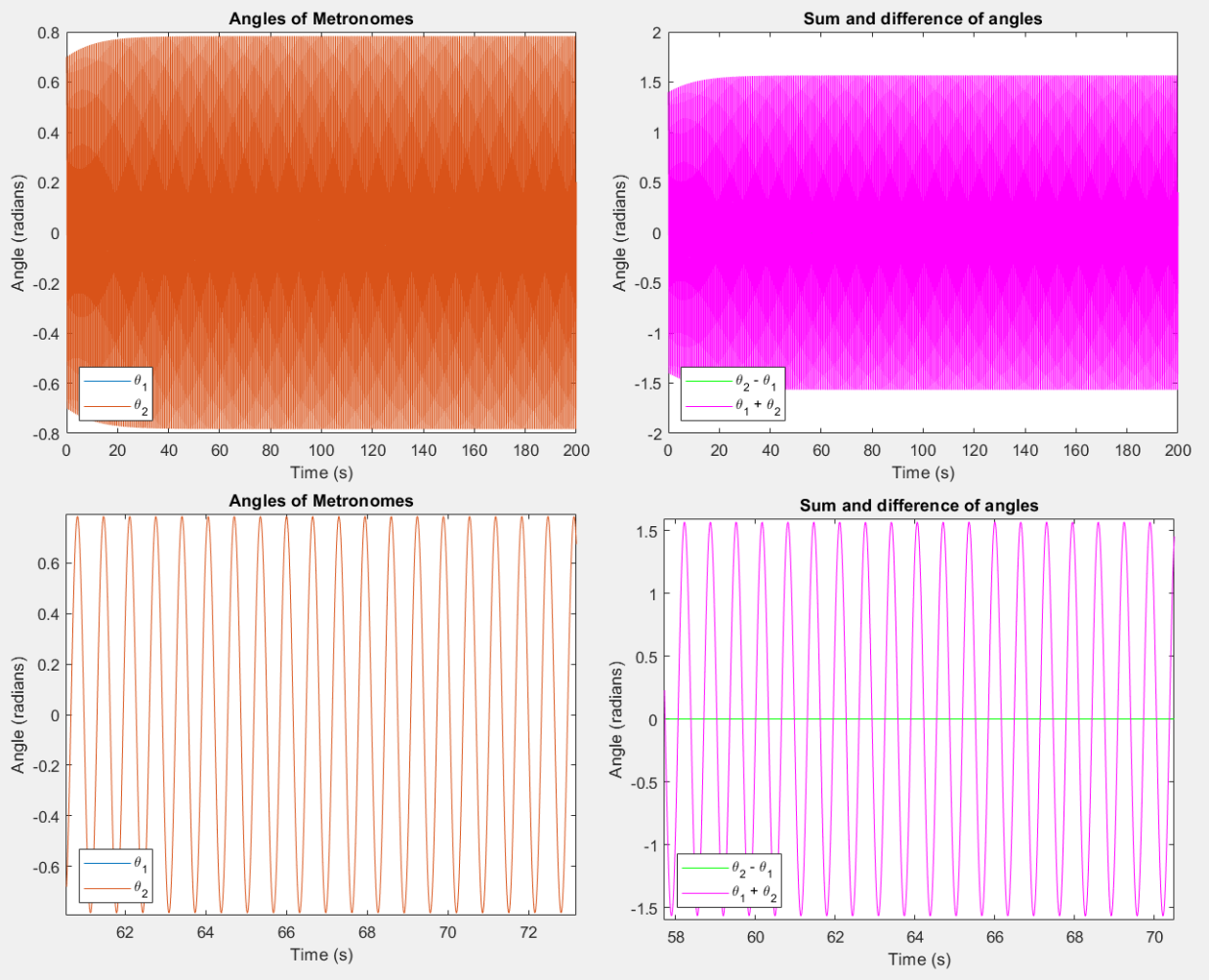


Figure 5.3: Initial phase difference of 0

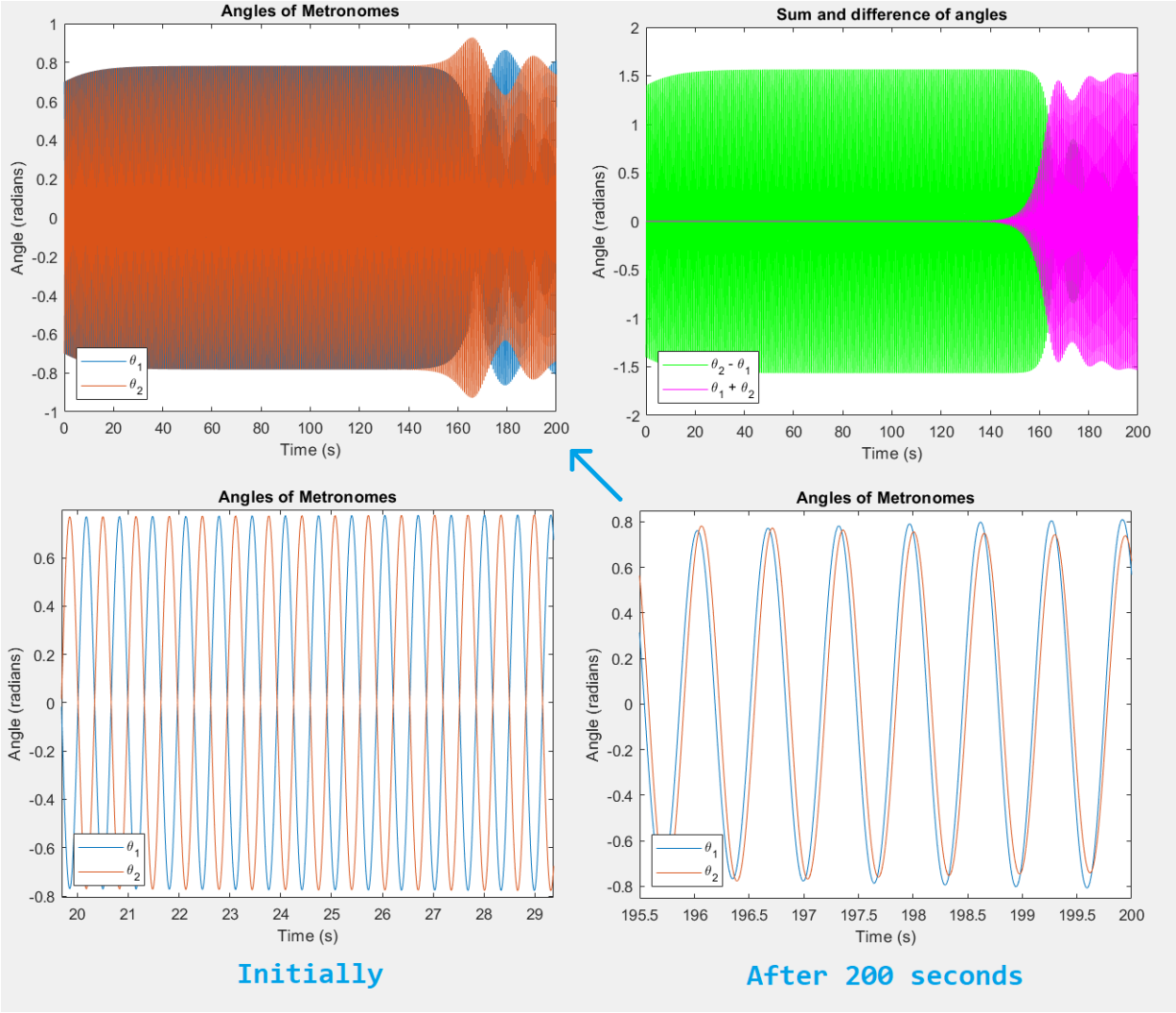


Figure 5.4: Initial phase difference of $2\pi$

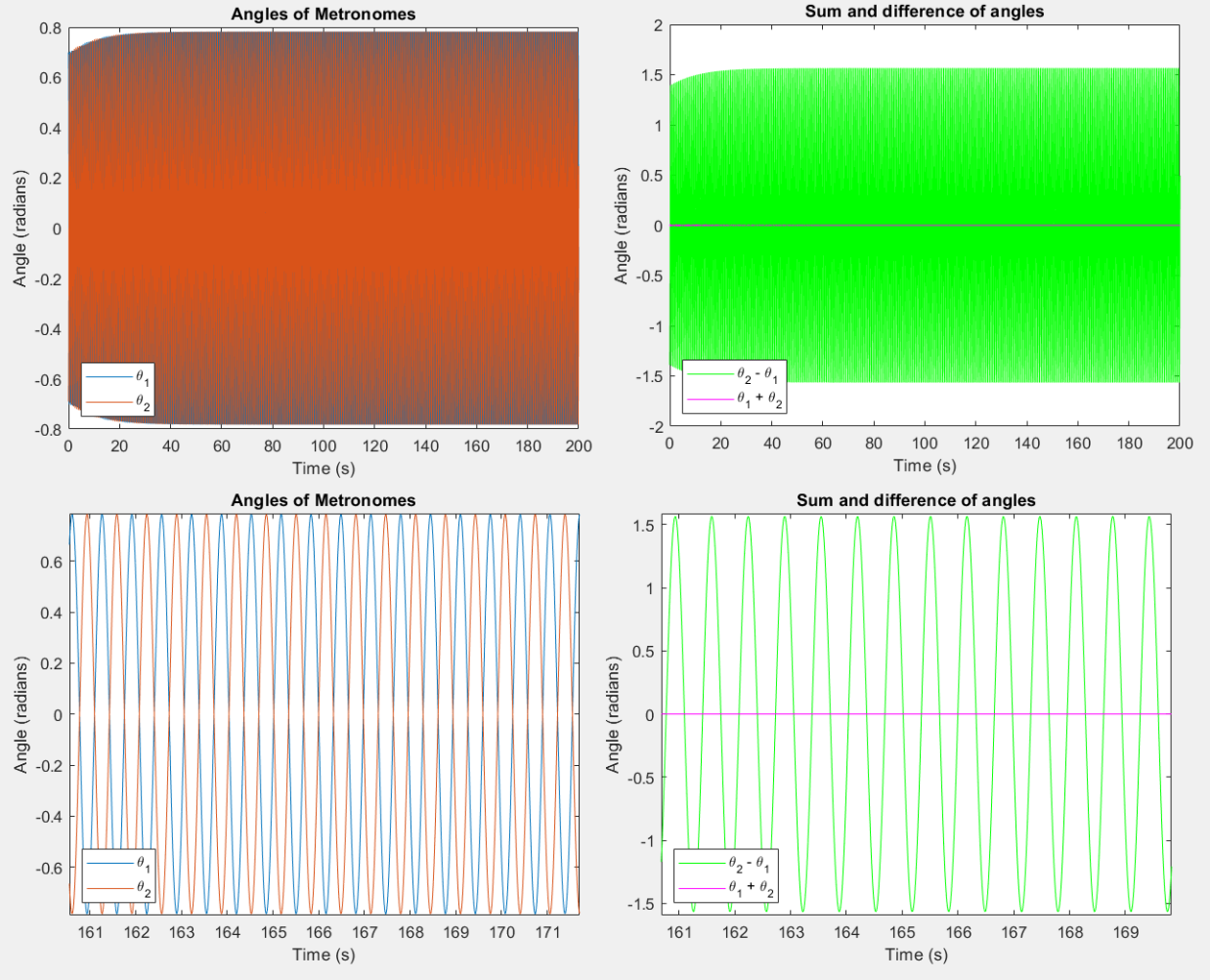


Figure 5.5: Initial phase difference of $2\pi$, $\beta = \mathbf{0.15}$

As we can see from the three figures, the in-phase synchronization state is a stable fixed point of the system variables. Figure (3.12) also shows that anti-phase synchronization is unstable for the given parameters, and tends to devolve into the in-phase state given long enough time.

Figure (3.13) also confirms the experimental observation that the anti-phase state becomes stable for high $\beta = 0.15$, which in this case is more than ten times as large as the previous $\beta = 0.011$. However, we will not be dealing with this set of parameters, as it is a case of strong coupling. The phase model result of Eq. (2.6) only works with weak coupling.

With these exact results, we go on to solve for the phase models for the approximate equations (3.7, 3.8) and the exact set of equations (3.4, 3.5), and then compare their results with the exact results.

## 5.4 Phase Model with Small Angle Approximation

We use the equation (3.6) to find the $Q(t)$ for the free running oscillator using Malkin's Approach (2.2). Then, we obtain the interaction terms from (3.7) and find the $H_{ij}(\phi_j - \phi_i)$'s for the two oscillators using (2.5), thus building the phase model. Finally, we use the $H_{ij}(\phi_j - \phi_i)$'s to find $G(\chi)$ from (2.6), then find its fixed points.

*Since the angles are assumed to be small, we have made an important change in the parameter* $\theta_0$. Instead of the experimental value of 0.39, we have assumed it to be 0.1rad. This is because the maximum amplitude of the metronomes is equal to $2\theta_0$ for small $\mu$. If $\theta_0$ were to be 0.39rad, then the max amplitude would be 0.78rad, thus nullifying our small angle approximation.

The reason why I am taking $\Delta = 0$ in this section, as well as in the next section, needs to be made clear in the context of building a phase model. The original reason of assuming perfectly identical metronomes still stands. Another more important reason is *the difficulty in calculating* $G(\chi)$ *for two non-identical metronomes.* Since their time periods are different, their limit cycles are represented by matrices of differing length, and consequently so do $H_{21}(\chi)$ and $H_{12}(-\chi)$ have different matrix lengths. Scaling the matrices to the same lengths (necessary to subtract them and find $G(\chi)$) is an arduous task, hence I have decided to only consider the case for $\Delta = 0$.

Considering the assumptions made above, the set of parameters used for building this phase model are:

$$\begin{aligned} \beta &= 0.011 \\ \Delta &= 0 \\ \mu &= 0.01 \\ \theta_0 &= 0.1 \end{aligned}$$

and $\theta_1,\ \theta_2 \approx 0$. The time step used was $5 \cdot 10^{-3}$s in slow time, hence $5 \cdot 10^{-4}$s in real time. We also found that the time period of the limit cycle is 6.225s in slow time, i.e. 0.6225s in real time.

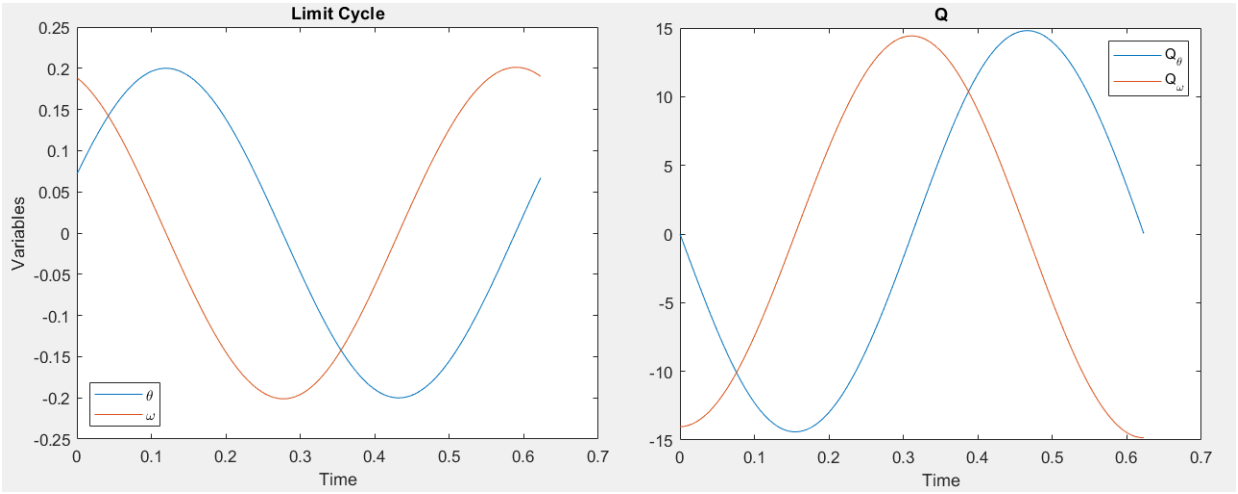


Figure 5.6: Limit Cycle and Q(t)

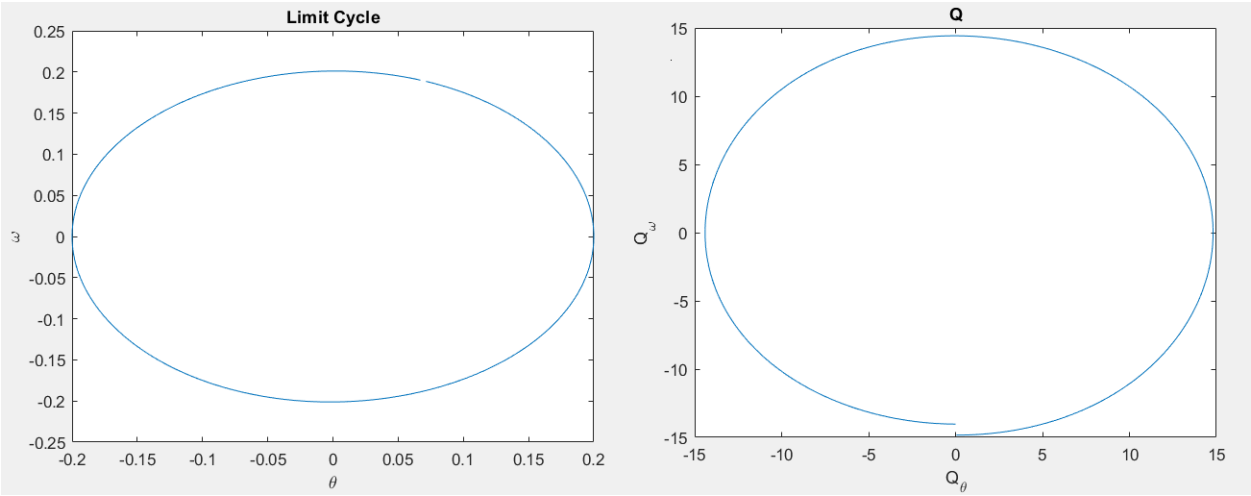


Figure 5.7: Limit Cycle and Q(t)

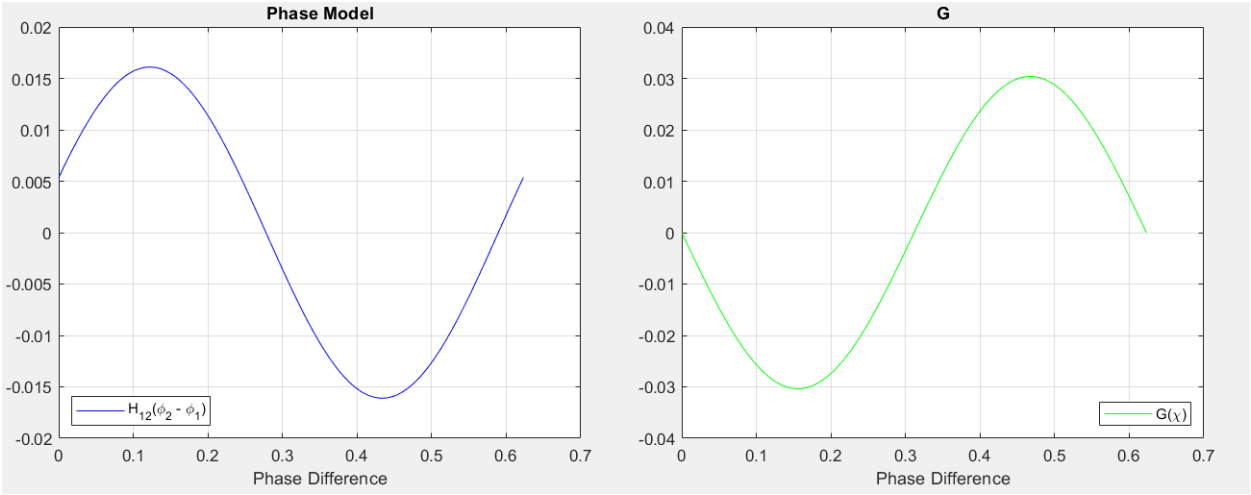


Figure 5.8: Phase Model

## 5.5 Phase Model with Exact Equations

We are using Eq.(3.6) for the free running oscillator equations. These equations are exact in nature. Using them, we find $Q(t)$ with Malkin's Approach. Then we build the phase model by considering the interaction terms. Finally, we get $G(\chi)$ and find its fixed points, as well as their stability.

Since there is no small angle approximation in this case, there is no need to have $\theta_0 = 0.1$. Yet, we cannot have $\theta_0 = 0.39$; *this causes the $Q(t)$ to become highly non-periodic.* This non-periodicity does not go away even after reducing the time steps, or by using higher order Runge-Kutta methods. This behavior can be attributed to the nature of the equations themselves, as the non-periodicity tends to increase as $\theta_0$ increases.

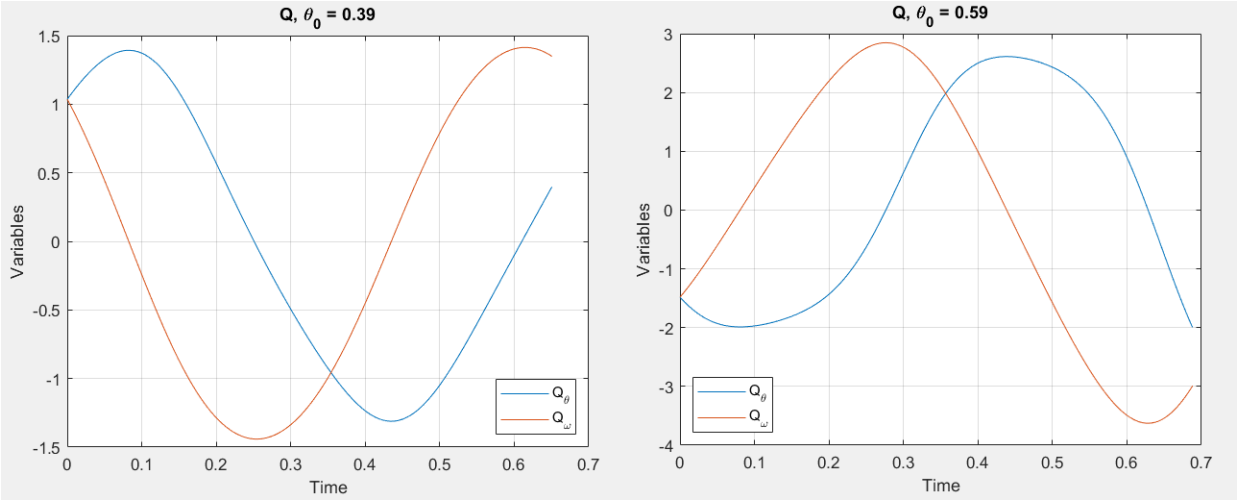


Figure 5.9: Non-periodic $Q(t)$ for large values of $\theta_0$

With the help of our advisor Dr. Dar, we found a work-around to keep the non-periodicity below a threshold, which is to keep $\theta_0$ in a range that is not very large. A value of 0.2rad works best in keeping the non-periodicity low, while also being comparable to the value used in the physical experiment (0.39rad).

Considering the assumptions made above, the set of parameters used here are:

$$\begin{aligned} \beta &= 0.011 \\ \Delta &= 0 \\ \mu &= 0.01 \\ \theta_0 &= 0.2 \end{aligned}$$

The time step used was $5 \cdot 10^{-3}$s in slow time, hence $5 \cdot 10^{-4}$s in real time. We also found that the time period of the limit cycle is 6.3s in slow time, hence 0.63s in real time.

## 5.6 Inferences

Based on the results of the phase models built in section (3.4) and (3.5), we can see that there are two fixed points for $\chi' = G(\chi)$, one stable fixed point at $\chi = 0$, and one unstable fixed point at

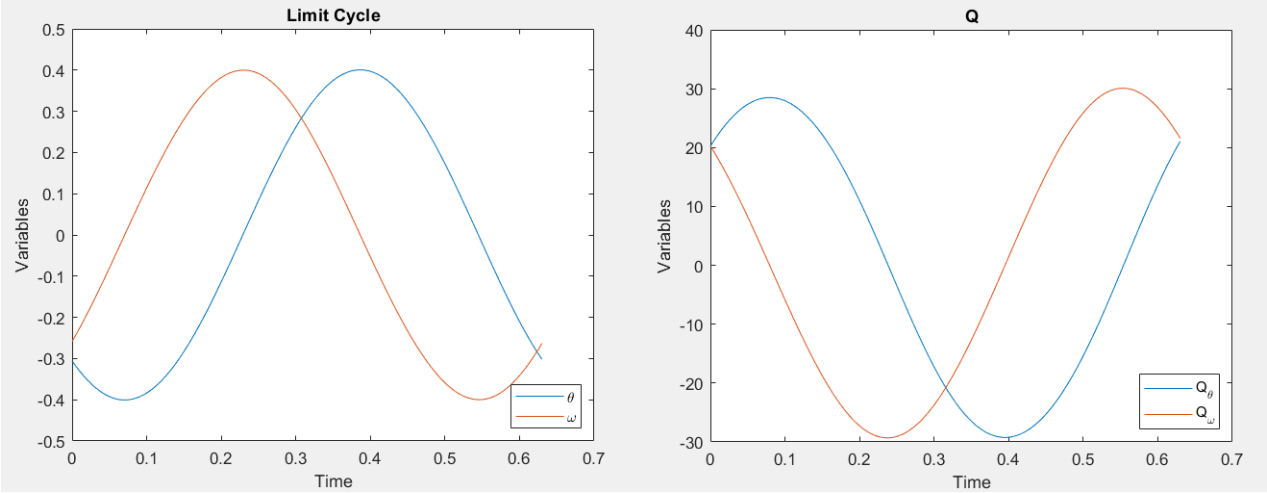


Figure 5.10: Limit cycle and $Q(t)$

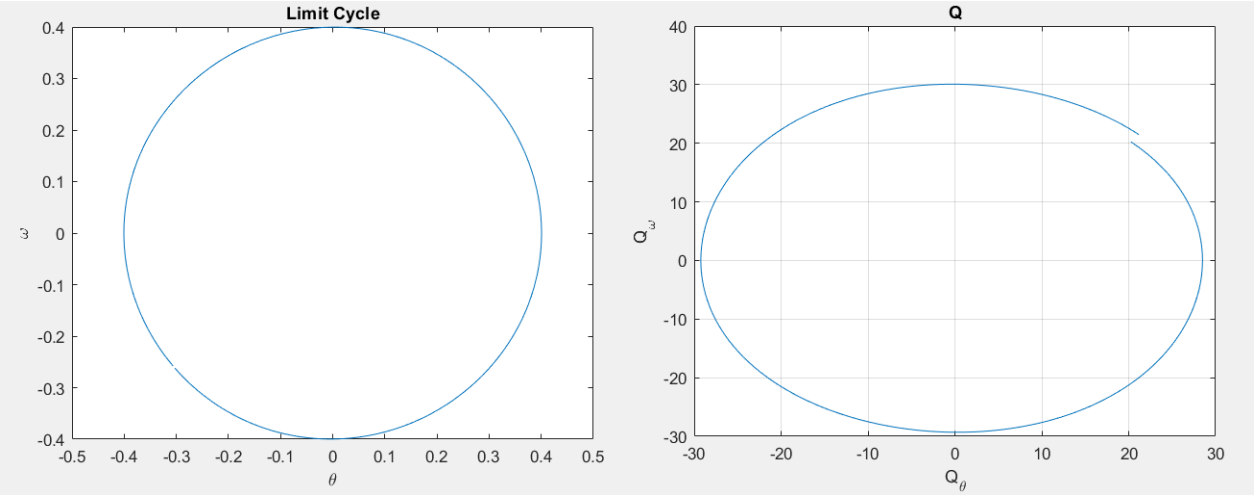


Figure 5.11: Limit cycle and $Q(t)$

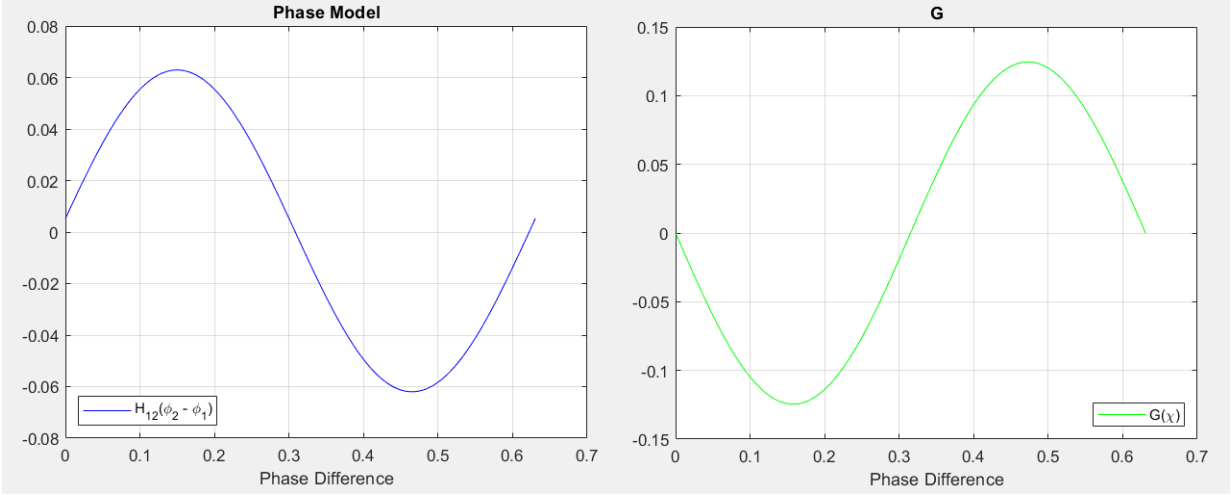


Figure 5.12: Phase Model and $G(\chi)$

$\chi = \pi$. Applying a small angle approximation to the equations does not change these results (as expected). It means that the phase difference between the two metronomes will, given enough time, come down to 0, i.e. they will synchronize. Even an initial phase difference of $\pi$ will reduce down to zero due to small perturbations or errors in calculation. This is in full agreement with Pantaleone's experimental results for weak coupling of two metronomes, as well as with the solutions to the explicit equations found in section (3.3).

The only unexpected outcome was the non-periodicity of $Q(t)$ for large values of $\theta_0$. In this specific case, the predictive power of the Phase Model fails. One reason for this failure might be the increase in the maximum magnitude of the interaction terms due to the increase in the amplitudes of the metronomes; we can see that an interaction term for the metronome: $\mu[(\frac{\theta}{\theta_0})^2 - 1]\omega$ is a term linear in $\omega$, whose maximum magnitude grows as $\theta_0$ grows. To reconcile these differences, a lower value of $\theta_0$ was used in our phase model.

# Chapter 6

# Studying Chimera States

The final aim of this project was to study chimera states occurring in a system of metronomes and swings. We dedicate this chapter to all the work we have done on trying to understand this system through numerical methods.

The hastily drawn diagram below shows the basic elements of the system. Two swings, $S1$ and $S2$ are connected together with a spring of spring constant $k$. On each swing, four metronomes are placed ($M11 - M14$ and $M21 - M24$). So in this case, $N = 4$ for the system, i.e. four metronomes on each swing. The coupling between metronomes kept on the same swing is stronger compared to the coupling between metronomes kept on different swings.

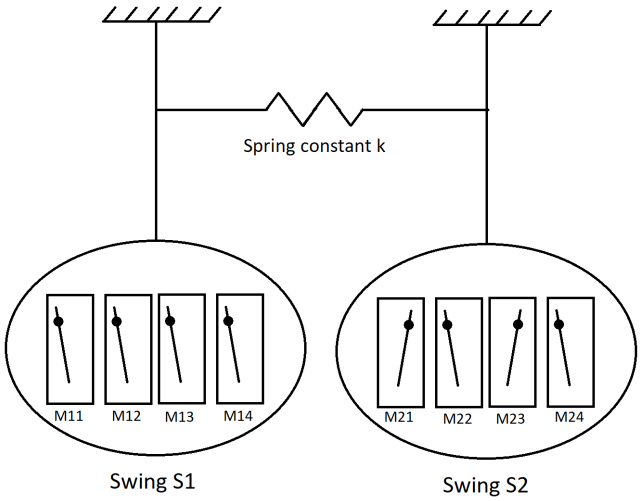


Figure 6.1: The system we are dealing with

This kind of coupling results in interesting synchronization states that could not be observed in the previous system. Apart from simple in-phase and anti-phase synchronization states of the two populations, we are also able to experimentally achieve metastable in-sync and de-sync or de-sync and in-sync states of the two populations, called *chimera states*.

To study this system numerically, we first need to evolve the equations of motion of the system numerically, and make sure that the results are in line with the results mentioned in Martens.

## 6.1 System of Equations

We will be using the system of equations provided by Martens. A small angle approximation is imposed on the swing angles and angular velocities to simplify the algebra. Some derived physical constants are used to find the exact metronome angular frequency and the sweep of the metronome arm. Finally, we will be working in slow time, just like before, and will be using derived non-dimensional parameters in the equations.

The system of equations is:

$$\partial_\tau^2 \tilde{\Phi} = \chi \left( \tilde{\Psi} - \tilde{\Phi} \right) - \omega_r^2 \, \tilde{\Phi} - \mu_s \, \partial_\tau \tilde{\Phi} - \sum_{j=1}^{N} \partial_\tau^2 \, sin \, \phi_j \tag{6.1}$$

$$\partial_\tau^2 \tilde{\Psi} = \chi \left( \tilde{\Phi} - \tilde{\Psi} \right) - \omega_r^2 \, \tilde{\Psi} - \mu_s \, \partial_\tau \tilde{\Psi} - \sum_{j=1}^{N} \partial_\tau^2 \, sin \, \psi_j \tag{6.2}$$

$$\partial_\tau^2 \, \phi_i = -sin \, \phi_i - \mu_m \, \partial_\tau \phi_i \left[ \left( \frac{\phi_i}{\theta_0} \right)^2 - 1 \right] - \beta \, cos \, \phi_i \, \partial_\tau^2 \, \tilde{\Phi} \tag{6.3}$$

$$\partial_\tau^2 \, \psi_i = -sin \, \psi_i - \mu_m \, \partial_\tau \psi_i \left[ \left( \frac{\psi_i}{\theta_0} \right)^2 - 1 \right] - \beta \, cos \, \psi_i \, \partial_\tau^2 \, \tilde{\Psi} \tag{6.4}$$

Here, the partial derivatives are with respect to slow time, i.e. $\tau = \omega \, t$, where $t$ is time. $\Phi$ and $\Psi$ are the swing angles of the left and right swings respectively, whose equilibrium values are $\Phi^*$ and $\Psi^*$, with $\Phi^* = -\Psi^*$. $\Phi'$ and $\Psi'$ are the deviations from the equilibrium angles; $\Phi(t) = \Phi^* + \Phi'(t)$ and $\Psi(t) = \Psi^* + \Psi'(t)$. $\tilde{\Phi}$ and $\tilde{\Psi}$ are the rescaled versions of these deviations, with $\tilde{\Phi}(t) = (L/x_0) \, \Phi'(t)$ and $\tilde{\Psi}(t) = (L/x_0) \, \Psi'(t)$. $\phi_i(t)$'s and $\psi_i(t)$'s represent the metronome oscillation angles of the left and right populations respectively.

The non-dimensional parameters are:

$$\beta = \frac{x_0 \, \omega^2}{g}$$

$$\chi = \frac{\kappa}{\omega^2}$$

$$\omega_r^2 = \frac{\Omega^2}{\omega^2}$$

The derived physical parameters are:

- $\omega^2 = (m \, g \, r_{cm}) / \, I$, the angular frequency of a free-running metronome;
- $\Omega^2 = g/L$, the angular frequency of the swings;
- $\kappa = (k \, l^2) / \, (ML^2)$, a scaled version of the spring constant;
- $x_0 = (m \, r_{cm}) \, / M$, the distance scale of the metronome swing motion;
- $\mu_s = 0.00016$, the damping of the swings;
- $\mu_m = 0.011$, the non-linear damping of the metronomes, the same parameter we have worked with before

These physical parameters are derived from the fundamental parameters:

- $g = 9.81\, m/s^2$, the acceleration due to gravity
- $M = 2.31\, kg$, the total mass of a swing and $N$ metronomes;
- $m = 0.028\, kg$, the mass of the moving parts of a metronome;
- $L = 0.22\, m$, the length of the swing's pendulum arm;
- $l = 0.15\, m$, the length away from the swing's fixed point where the spring is attached;
- $r_{cm}$, the center of mass of the metronome pendulum;
- $I$, the moment of inertia of the metronome pendulum;
- $\theta_0 = 0.33 rad$, the same metronome parameter we have worked with earlier

Finally, the two parameters $r_{cm}$ and $I$ are a function of the metronome bob position, which is moved around to adjust its frequency. To calculate these parameters, we use an intermediate physical parameter called $l_{bob}$:

$$\begin{aligned} l_{bob} &\approx 7.3 \times 10^{-2}\, m - 2.2 \times 10^{-4} \frac{m}{bpm} \times f[bpm] \\ r_{cm} &\approx |-0.0121\, m + 0.178 \cdot l_{bob}\, [m]| \\ I &= 1.29 \times 10^{-5}\, kg \cdot m^2 + 5 \times 10^{-3}\, kg \cdot l_{bob}^2\, [m^2] \end{aligned}$$

Where $f$ is the desired frequency of the metronome in units of $bpm$.

## 6.2 Numerical Simulations

Once we have dealt with all these parameters, we can move onto coding the system of equations.

### 6.2.1 Transforming the equations into ODEs

The first thing we notice is that we need to transform the Differential-Algebraic Equations (DAEs) into a system of ODEs. Once transformed, we have:

$$\dot{\phi}_k = \omega_k^{\phi} \tag{6.5}$$
$$\dot{\psi}_k = \omega_k^{\psi} \tag{6.6}$$
$$\dot{\Phi} = \omega^{\Phi} \tag{6.7}$$
$$\dot{\Psi} = \omega^{\Psi}, \tag{6.8}$$

for the derivatives of the oscillator angles,

$$\dot{\omega_k}^{\phi} = -\left[\text{क}_k + \beta\, cos\, \phi_k\, \partial_\tau^2\, \tilde{\Phi}\right] \tag{6.9}$$
$$\dot{\omega_k}^{\psi} = -\left[\text{छ}_k + \beta\, cos\, \psi_k\, \partial_\tau^2\, \tilde{\Psi}\right], \tag{6.10}$$

for the derivatives of the angular velocities of the $2N$ metronomes, (here $k$ varies from 1 to $N$)

$$\dot{\omega}^{\Phi} = (1 - \beta \cdot \text{अ})^{-1} \left\{ -\left(\chi + \omega_r^2\right) \tilde{\Phi} + \chi \, \tilde{\Psi} - \mu_s \, \omega^{\Phi} + \sum_{i=1}^{N} \left[ \left(\omega_i^{\phi}\right)^2 sin\, \phi_i + cos\, \phi_i \, \text{क}_i \right] \right\} \tag{6.11}$$

$$\dot{\omega}^{\Psi} = (1 - \beta \cdot \text{इ})^{-1} \left\{ -\left(\chi + \omega_r^2\right) \tilde{\Psi} + \chi \, \tilde{\Phi} - \mu_s \, \omega^{\Psi} + \sum_{i=1}^{N} \left[ \left(\omega_i^{\psi}\right)^2 sin\, \psi_i + cos\, \psi_i \, \text{छ}_i \right] \right\} \tag{6.12}$$

for the derivatives of the angular velocities of the two swings.

The abbreviations used are:

$$\text{क}_k = sin\, \phi_k + \mu_m \left[ \left( \frac{\phi_k}{\theta_0} \right)^2 - 1 \right] \omega_k^{\phi} \tag{6.13}$$

$$\text{छ}_k = sin\, \psi_k + \mu_m \left[ \left( \frac{\psi_k}{\theta_0} \right)^2 - 1 \right] \omega_k^{\psi} \tag{6.14}$$

$$\text{अ} = \sum_{k=1}^{N} cos^2\, \phi_k \tag{6.15}$$

$$\text{इ} = \sum_{k=1}^{N} cos^2\, \psi_k \tag{6.16}$$

### 6.2.2 Specifics of the numerical methods used

This set of ODEs was solved separately using three languages: MATLAB (by my collaborator Abhishek), Python and Julia (by me). The method used to solve the ODEs is RK4 in each language. The final results were found by me using Julia, owing to its speed.

The parameter values used in the simulations are all given above. We are able to vary the free-running frequency of the metronomes $f$ (measured in bpm), as well as the scaled spring constant $\kappa$.

We used the RK4 method to solve the system of ODEs, with a timestep of either 0.005s or 0.01s in slow time $\tau$. The maximum time we evolved it to was from 500s-2000s. This corresponds to approximately 159-637 oscillations of the metronomes, since we have their time periods $T = 2\pi$ in slow time.

Preliminary tests were done by selecting a frequency of 160bpm, 138bpm and 200bpm. The final frequency was chosen to match the one recommended by Martens. The initial conditions used to hopefully generate chimera states were:

$$\begin{aligned}
\phi_i(0) &= 2\theta_0 \\
\partial_\tau \phi_i(0) &= 0 \\
\psi_i(0) &= 2\theta_0(r_i - 1/2) \\
\partial_\tau \psi_i(0) &= 2\theta_0(r_i - 1/2), \qquad i = 1...N \\
\tilde{\Phi}(0) &= \tilde{\Psi}(0) = 0 \\
\partial_\tau \tilde{\Phi}(0) &= \partial_\tau \tilde{\Psi}(0) = 0
\end{aligned}$$

Here, $r_i$ represents a (pseudo) random number in $[0, 1]$. The right population (of the $\Phi$ swing) starts out being in-sync, whereas the left population (of the $\Psi$ swing) starts out fully out of sync, i.e. with random initial conditions. This population will hopefully remain out of sync throughout the duration of the simulation. Both the swings start from rest.

## 6.3 Results of Solving the ODEs

We first used the parameter values suggested by Martens:

$$\begin{aligned} N &= 15 \\ f &= 160\, bpm \\ k &= 68\, N/m \\ l &= 0.15\, m \\ L &= 0.22\, m \end{aligned}$$

The derived physical parameters are:

$$\begin{aligned} \omega &= 8.58\, rad/s \\ x_0 &= 6.51 \times 10^{-5}\, m \\ \kappa &= 13.68\, N/m/kg \end{aligned}$$

Which give the non-dimensional parameter values:

$$\begin{aligned} \beta &= 4.89 \times 10^{-4} \\ \omega_r^2 &= 0.606 \\ \chi &= 0.186 \\ \mu_s &= 0.00016 \\ \mu_m &= 0.011 \end{aligned}$$

The random initial conditions used for the metronome angles for all the following simulations are:

0.14157382, 0.2614215, -0.29608024, -0.24948547, -0.07842186, -0.31545333, 0.06066683, 0.27495881, 0.32353985, -0.19375554, 0.07076737, 0.12151782, -0.28229673, 0.14350184, 0.03906534

and the ICs for the metronome angular velocities are:

0.04105613, -0.22835415, 0.05830795, -0.20626723, -0.15800162, -0.21524905,0.18515062,0.05557701 -0.13233109, -0.31297807, -0.01390028, 0.28235838, 0.0213475 , -0.20703707, -0.04162385

We have evolved the system for a total time of 1500s in slow time, with a time step of 0.005s. The figures below are for $\kappa = 13.68$ and $f = 160 bpm$

### 6.3.1 With parameter values suggested by Martens et. al

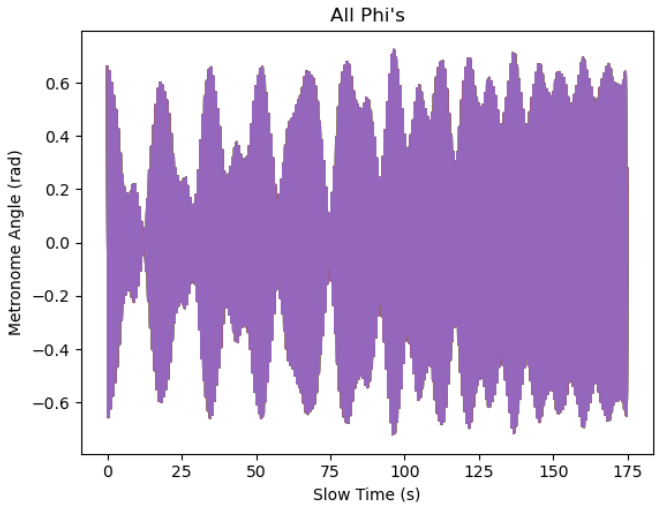


Figure 6.2: Right (In-sync) Population

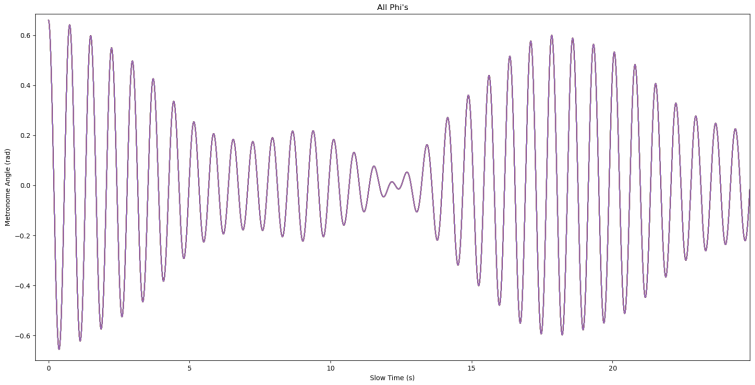


Figure 6.3: First 25s of Phi Population

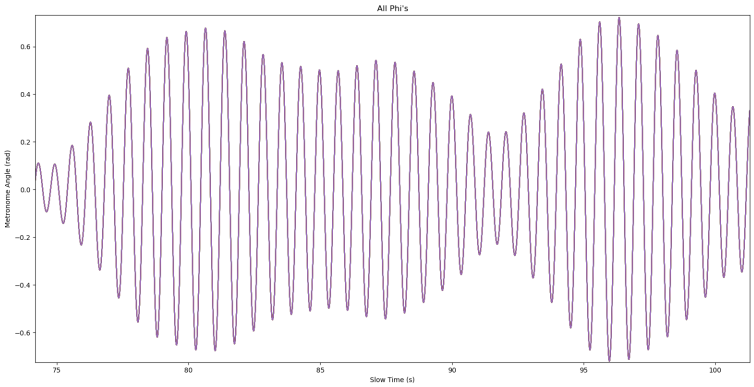


Figure 6.4: 75-100s of Phi Population

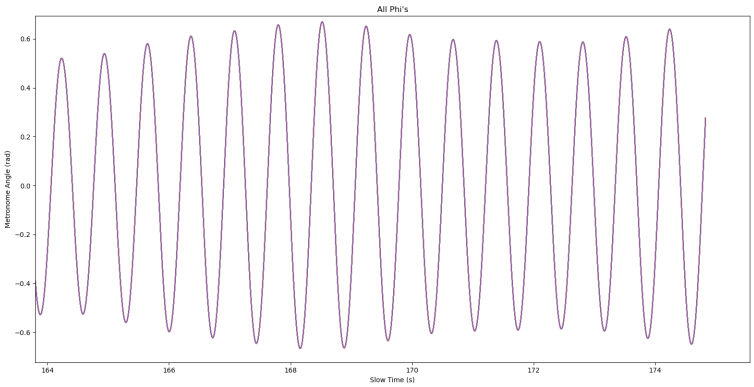


Figure 6.5: Last 10s of Phi population

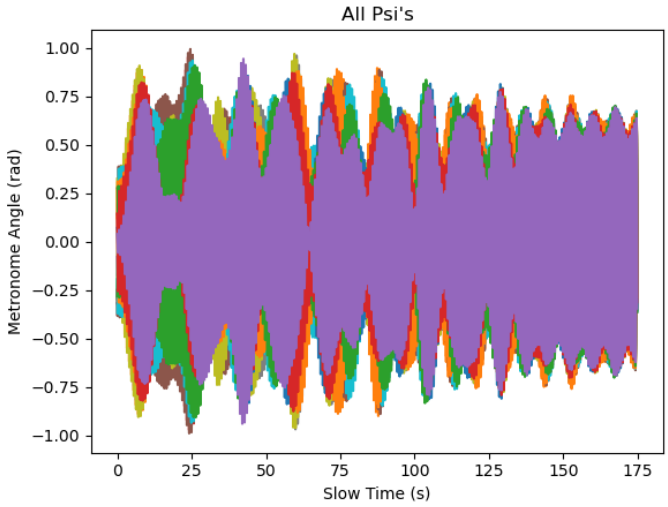


Figure 6.6: Left (de-sync) Population

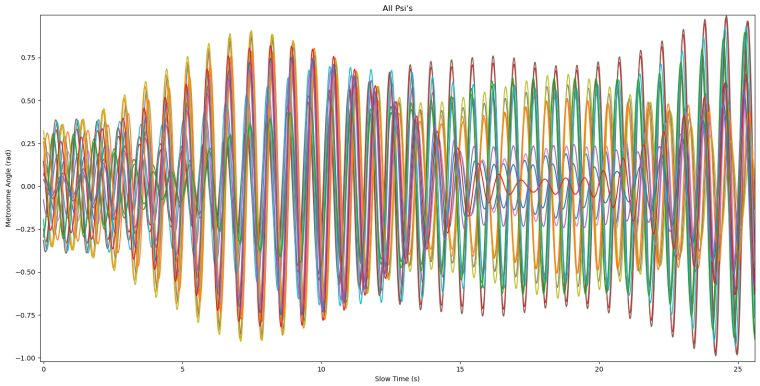


Figure 6.7: First 25s of Psi Population

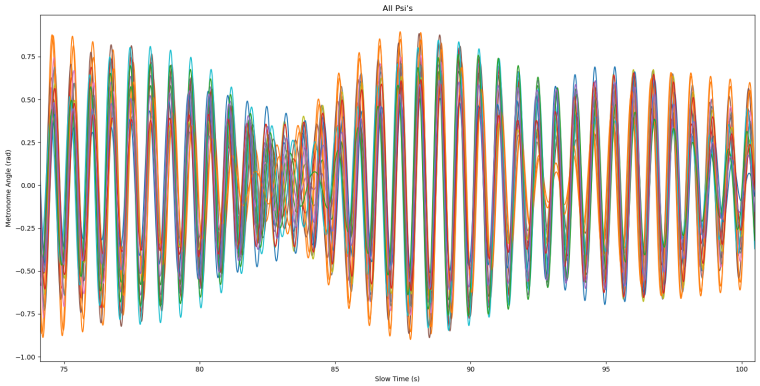


Figure 6.8: 75-100s of Psi Population

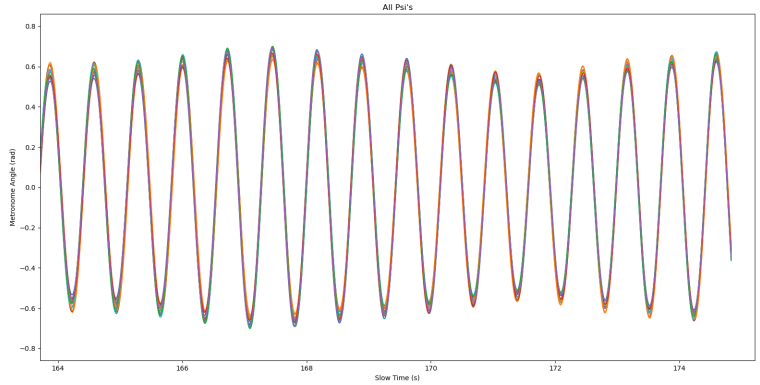


Figure 6.9: Last 10s of Psi population

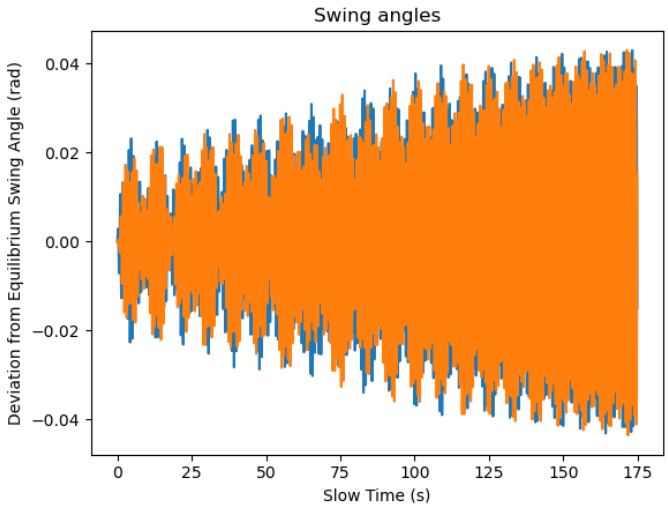


Figure 6.10: Swing Oscillations (deviation from Equ. Position)

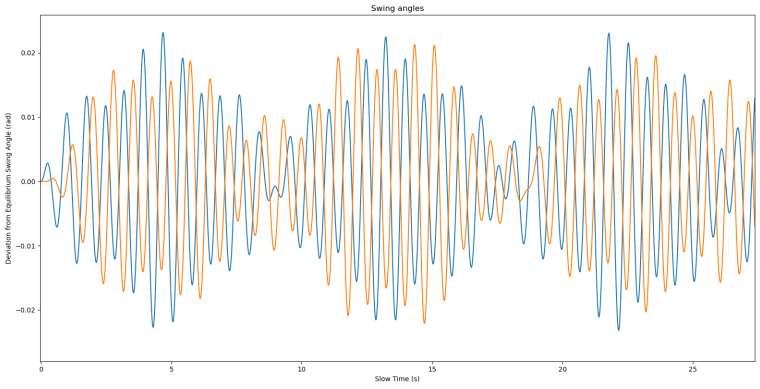


Figure 6.11: Swing Oscillations till 25s

### 6.3.2 With $\kappa = 5$

To change the value of $\kappa$, we change the value of $l = 0.15m$ to $l = 0.09m$.

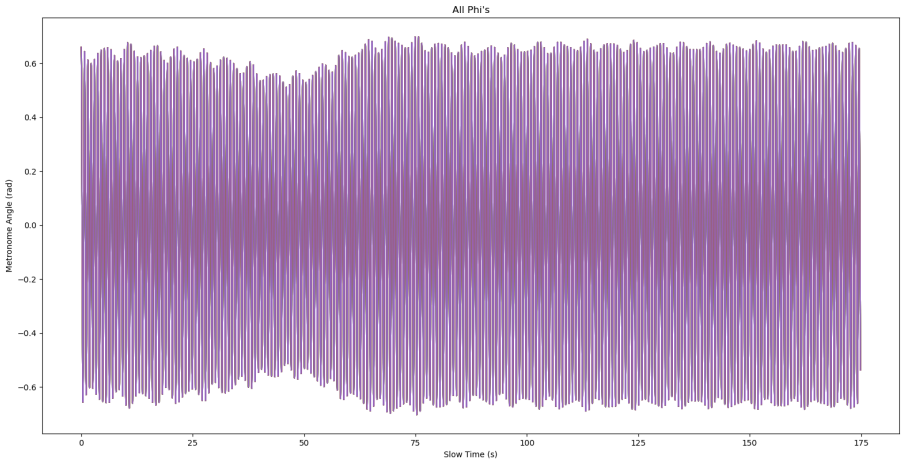


Figure 6.12: Right (In-sync) Population

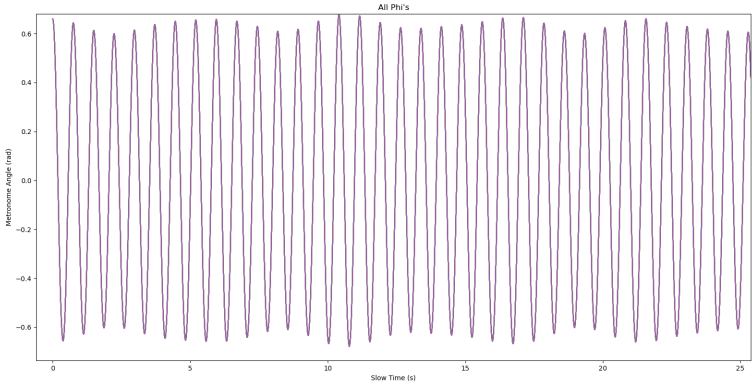


Figure 6.13: First 25s of Phi Population

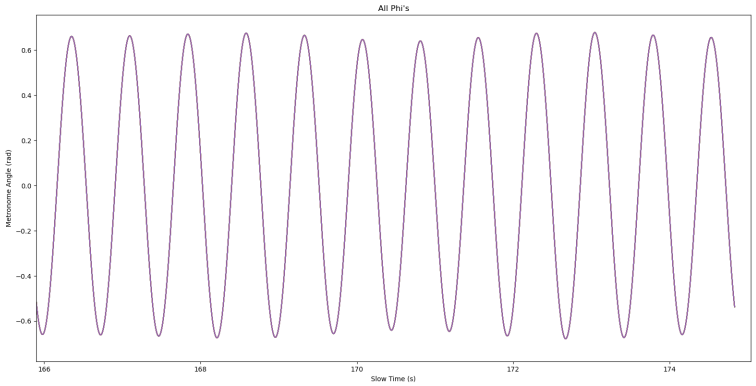


Figure 6.14: Last 10s of Phi population

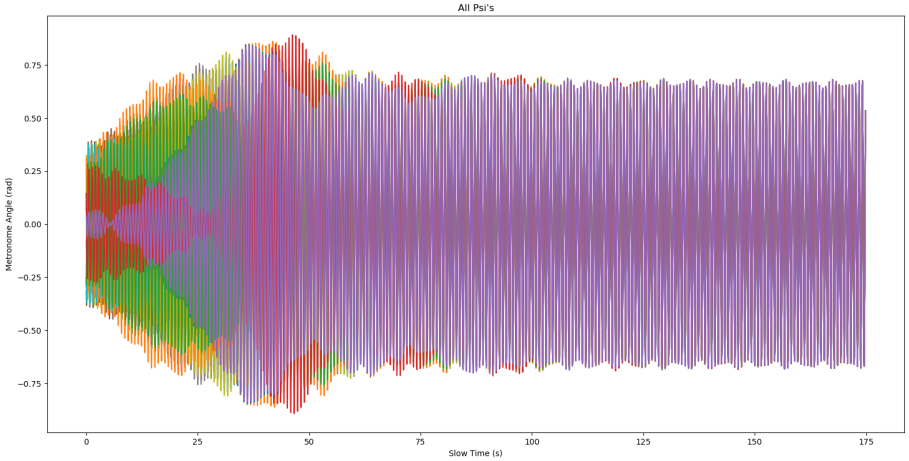


Figure 6.15: Left (de-sync) Population

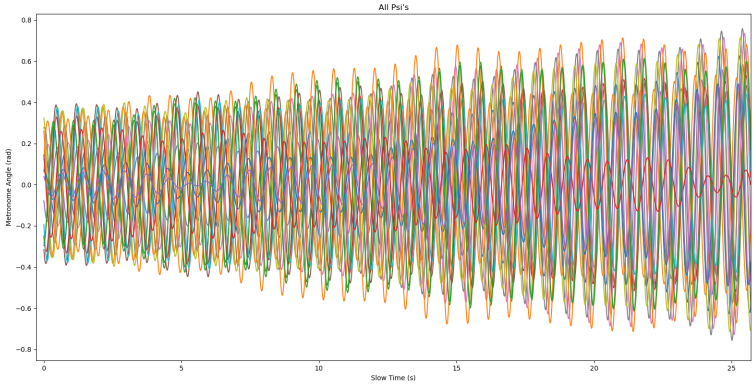


Figure 6.16: First 25s of Psi Population

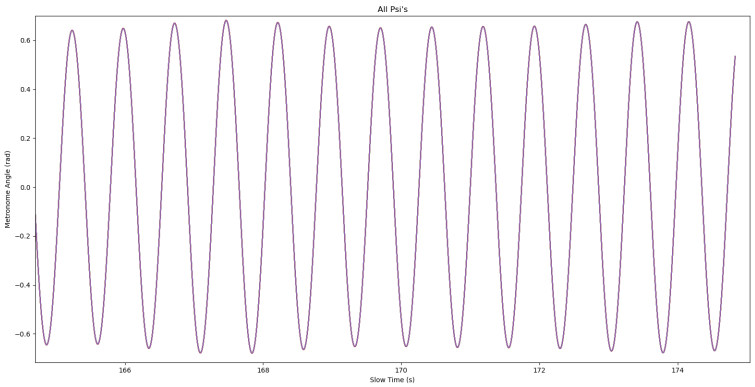


Figure 6.17: Last 10s of Psi population

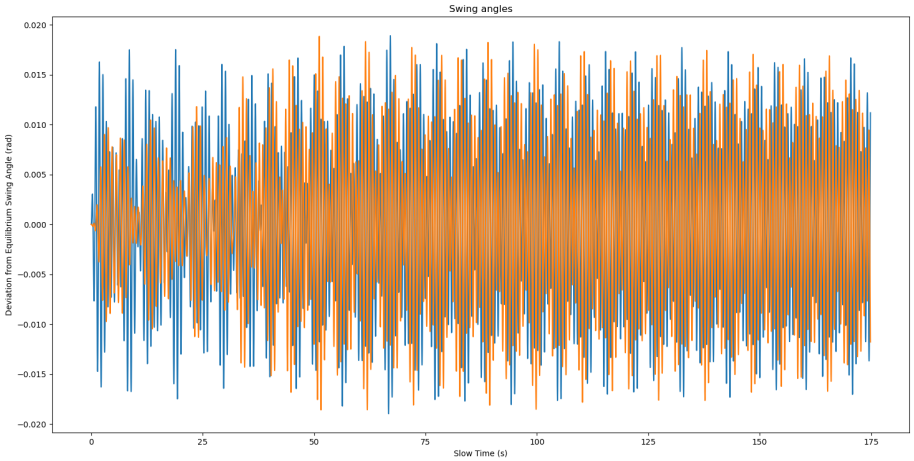


Figure 6.18: Swing Oscillations (deviation from Equ. Position)

### 6.3.3 With $\kappa = 20$

To change the value of $\kappa$, we change the value of $l = 0.15m$ to $l = 0.18m$.

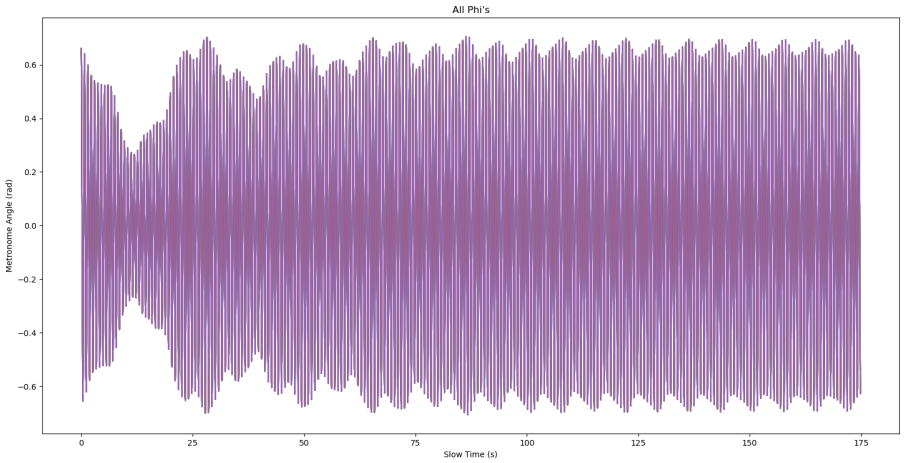


Figure 6.19: Right (In-sync) Population

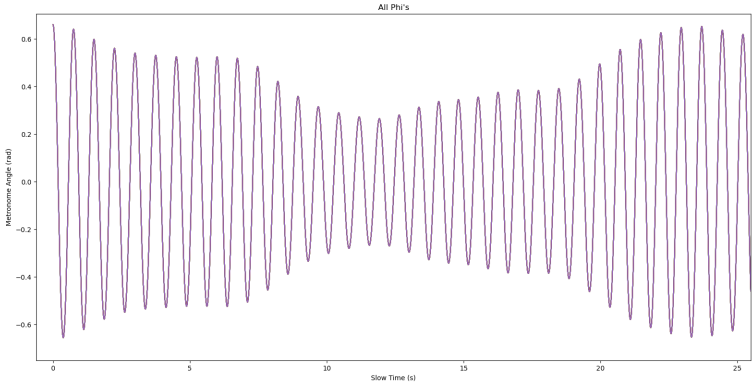


Figure 6.20: First 25s of Phi Population

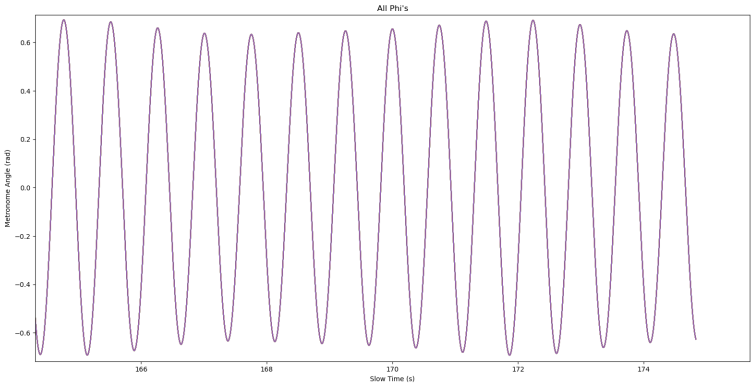


Figure 6.21: Last 10s of Phi population

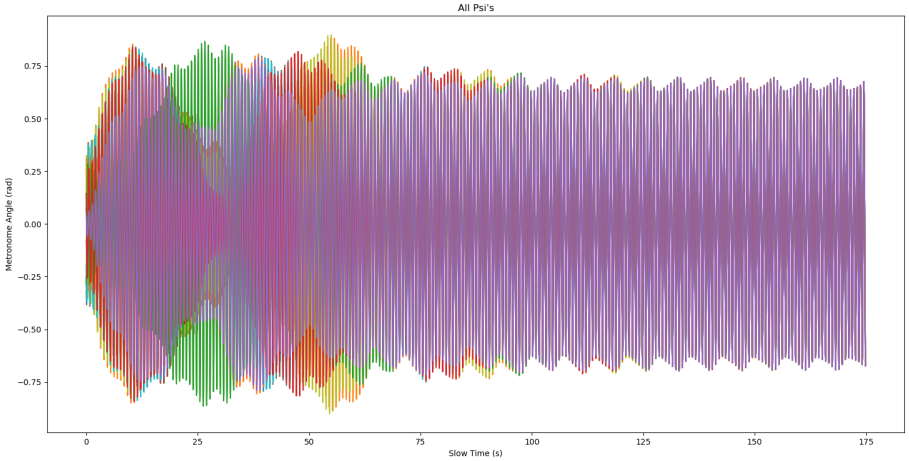


Figure 6.22: Left (de-sync) Population

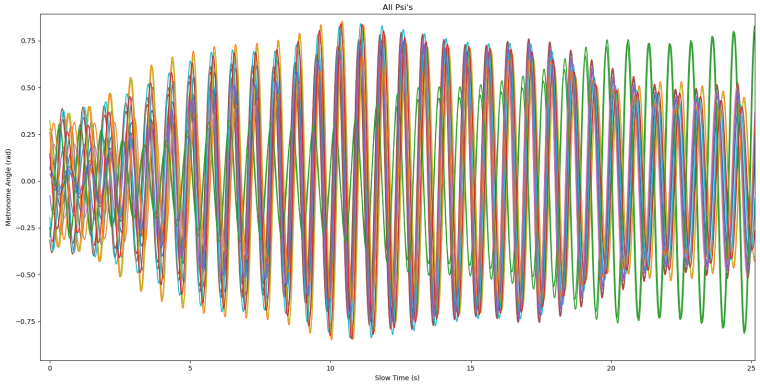


Figure 6.23: First 25s of Psi Population

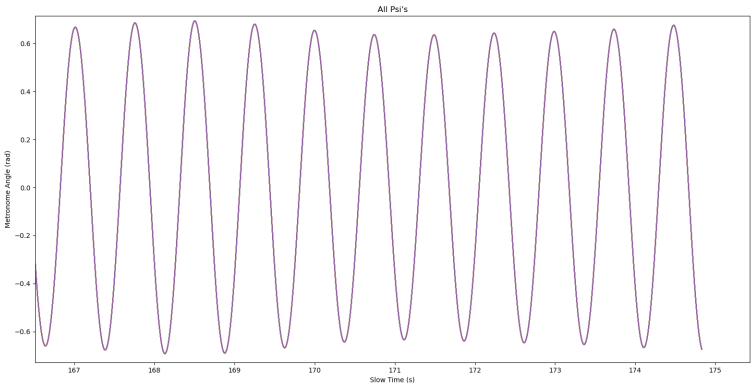


Figure 6.24: Last 10s of Psi population

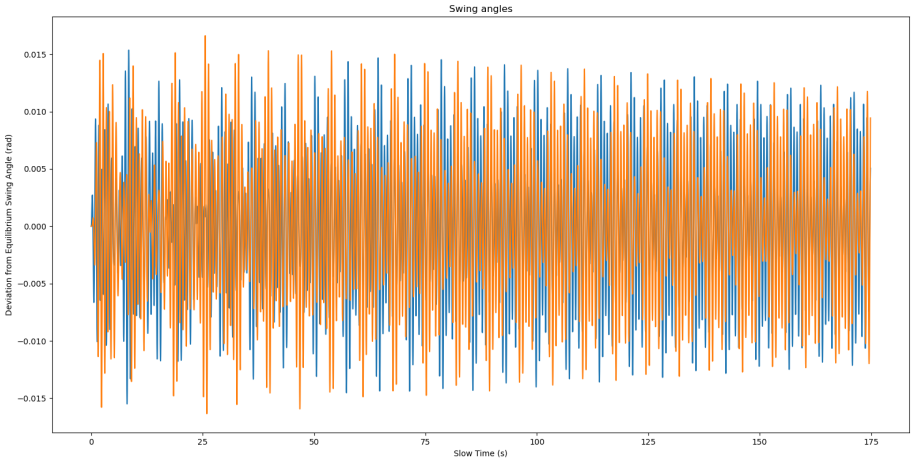


Figure 6.25: Swing Oscillations (deviation from Equ. Position)

### 6.3.4 With reduced $x_0$

While varying the parameters I noticed a peculiar behavior of the de-synced population when the value of $x_0$ is below a threshold value of $10^{-6}$. This can be achieved by dividing the current value of $x_0$ by 100, ie $x_0^{new} = 6.51 \times 10^{-7}$. The value of $\kappa = 20$ is the only other parameter changed.

With this new value of $x_0$, the $\psi$ population of metronomes syncs up to a state not seen before; one where the phase differences between each pair of metronomes varies very slowly with time. As can be seen in the plots below, it looks almost as if all metronomes are phase-locked. The reason for mentioning this behavior in the report is to bring notice to the fact that not only do the metronomes have (in-sync, in-sync), (in-sync, de-sync), (de-sync, in-sync) states, *they also have phase locked states, at least according to numerical simulations.* More research could be conducted to bring out such states experimentally, but one wonders if reducing the value of $x_0$ by a factor of 100 is physically possible, since $x_0 = (mr_{cm})/M$.

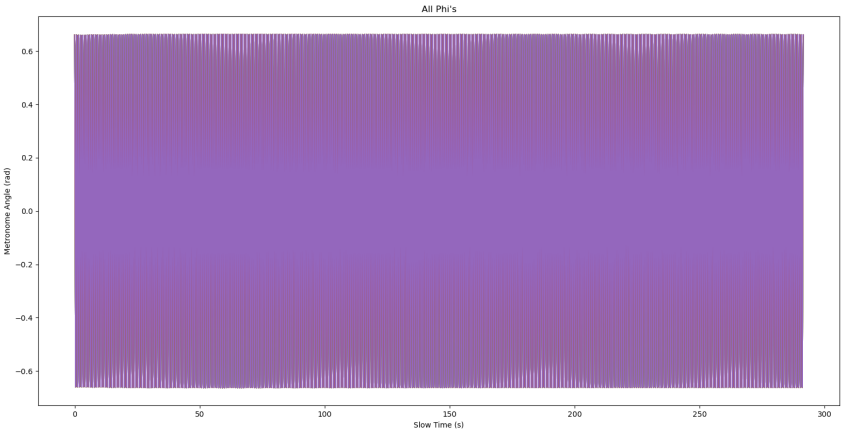


Figure 6.26: Right (In-sync) Population

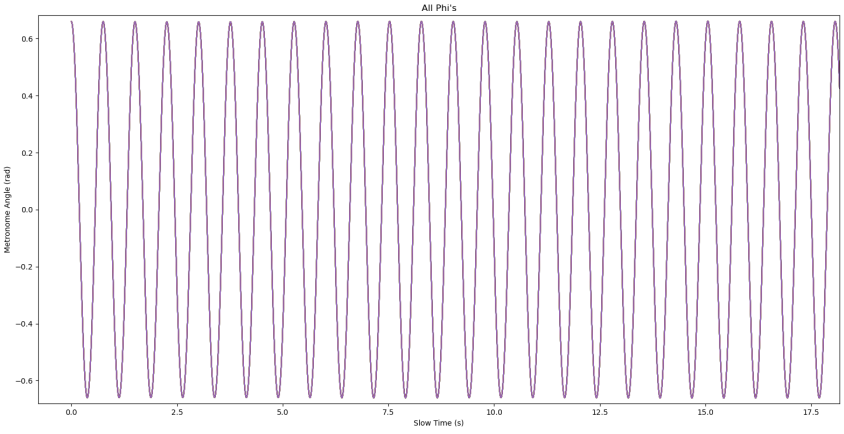


Figure 6.27: First few seconds of Phi population

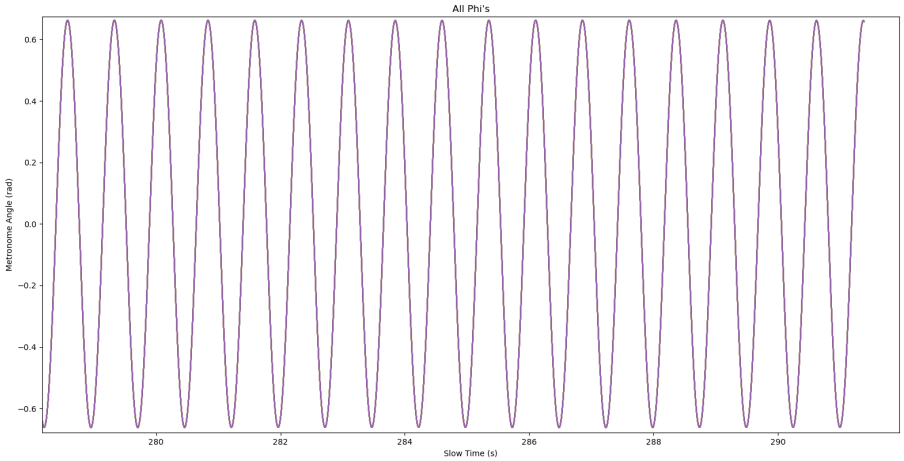


Figure 6.28: Last few seconds of Phi population

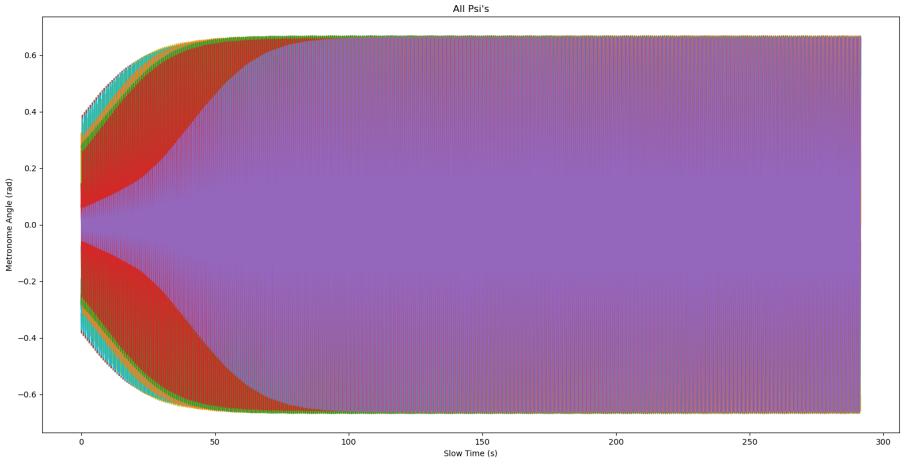


Figure 6.29: Left (De-sync) Population

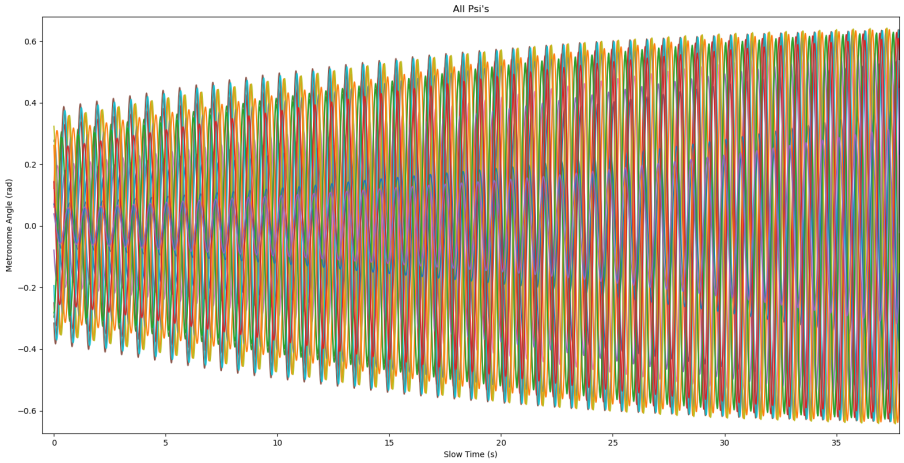


Figure 6.30: First few seconds of Psi population

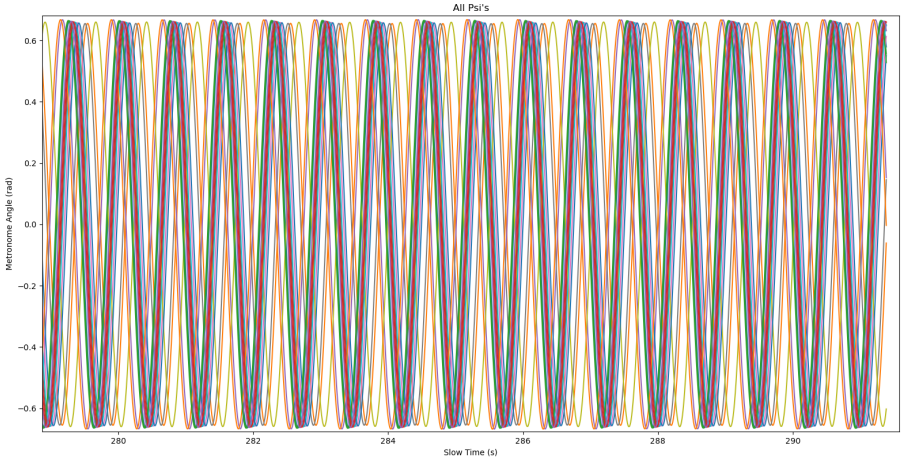


Figure 6.31: Last few seconds of Psi population

## 6.4 Inferences

Up till now, we have seen the results for three values of $\kappa = 5, 13.68, 20$. For all the values, the $\phi$ population always stays synced up. For both the first and last value, we notice that the $\psi$ population syncs up in 1500s. When $\kappa = 20$, the average value of the $\psi$ angles is in sync with the $\phi$ angles, ie. $avg(\psi_i(1500)) \approx phi_1(1500)$. When $\kappa = 5$, the average value of $\psi$ angles is anti-sync with the $\phi$ angles. These are the two extremes that Martens et. al. [al13] has mentioned. For the intermediate value of $\kappa = 13.68$, Martens et. al. were able to generate chimera states by solving the equations numerically, as well as experimentally. As can be seen in the plots above, we were *not* able to do so.

Upon comparing my results with Abhishek's, we found that while the qualitative nature of the results was similar, the exact vectors were not matching up for each time step. We were simulating with different software and were using slightly different ways of evaluating the ODE functions (my code uses direct evaluation, while his uses anonymous functions). The difference in our coding may be giving rise to the discrepancy in the results. We tried our best to narrow down the cause of the discrepancy, but to no avail. To try to narrow down the scope of the problem and find discrepancies there, we decided to work with a highly simplified version of the system, consisting of a swing and a metronome; a new kind of limit cycle oscillator. The next section deals with our findings with this system and the insights it has given us into the nature of the full system.

# Chapter 7

# Single Swing-Metronome System

For N metronomes on a swing, we have a total of 2N+2 oscillators. Out of these, 2N are metronomes and 2 are swings. The 2N metronomes are limit cycle oscillators, while the swings are not. Phase models can only be built with limit cycle oscillators, so it is necessary to somehow change the nature of the swing oscillators. The way we do it is by combining a metronome and a swing and calling it a new limit cycle oscillator. Keep in mind that for this new "oscillator" to actually be a limit cycle oscillator, the swing needs non-zero damping, else the system will be a conservative one, and the swing will be able to oscillate at any given amplitude.

The system of equations for the swing-metronome system are given below:

$$\begin{aligned}
\dot{\phi} &= \omega^{\phi} \\
\dot{\Phi} &= \omega^{\Phi} \\
\dot{\omega}^{\Phi} &= \left(1 - \beta\, cos^2\phi\right)^{-1} \left[-\omega_r^2 \Phi - \mu_s\, \omega^{\Phi} + \text{क}\, cos\,\phi + sin\,\phi\, \left(\omega^{\phi}\right)^2\right] \\
\dot{\omega}^{\phi} &= -\left\{\text{क} + \beta\, cos\,\phi\, \partial_\tau^2\, \Phi\right\}
\end{aligned}$$

The abbreviation used is:

$$\text{क} = sin\,\phi + \mu_m \left[\left(\frac{\phi}{\theta_0}\right)^2 - 1\right] \omega^{\phi}$$

Using an initial condition of $\phi(0) = 2\theta_0$ (rest of the values being zero), I evolved this set of ODEs for 100,000s and a time step of 0.01s. Once the transients died down, I had a single point on the limit cycle, which I evolved for 10 more seconds, with a time step of 0.00001s. This allowed me to find the full limit cycle. The time period of said limit cycle is 6.455s in slow time, or 0.6115s in real time. For the plots below, $f = 200bpm$. Rest of the parameters are the same as before.

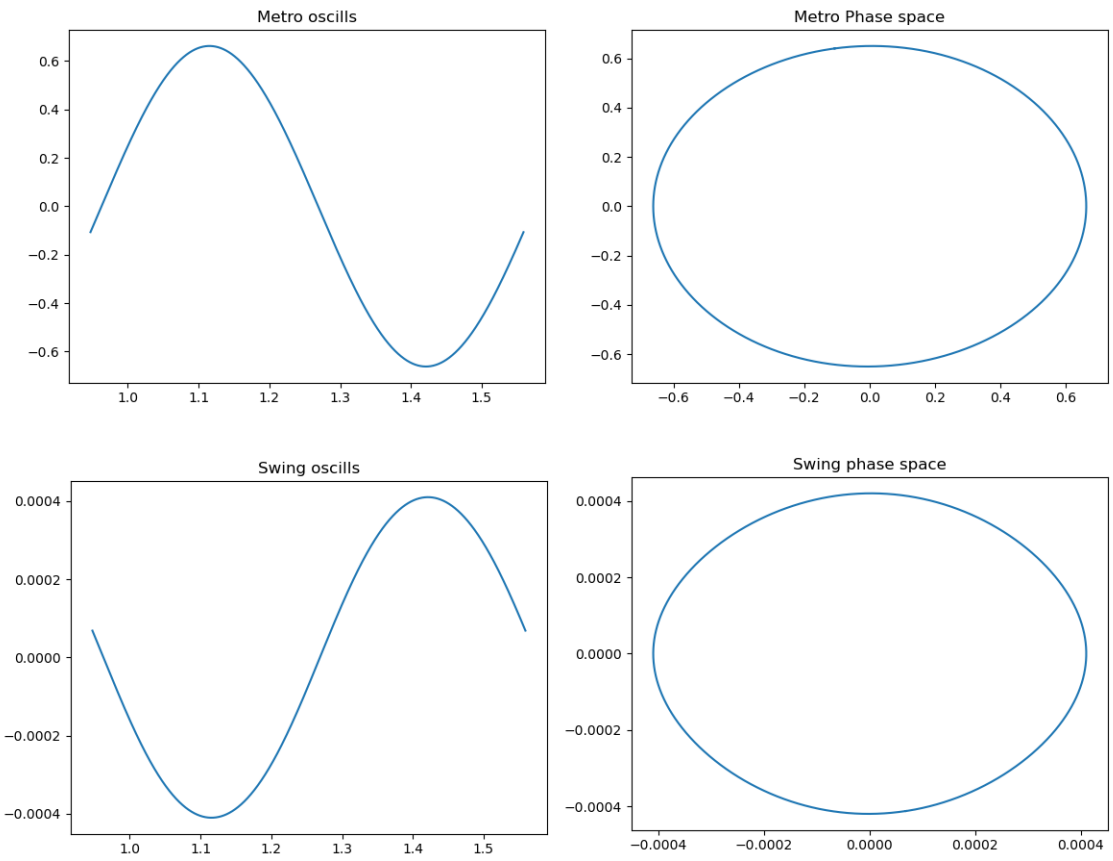


Figure 7.1: Limit cycle for the single metronome-swing system

Now, while building the phase model, we have 2N-2 free metronomes, and 2 swing-metronome oscillators. We can find the interaction terms between these 2N limit cycle oscillators, then simply build the phase model just like we have before. The question we need to ask ourselves before doing so is *what is the kind of coupling between the swing and the metronomes?*. The answer, if we want to move forward with the phase model, is obviously weak coupling. We probed into this by doing a few simulations.

## 7.1 Increasing the number of metronomes to 2

The equations for a system of one swing and two metronomes can be obtained by using the equations for the original system, then putting $N = 2$ and $\kappa = 0$, and only considering the right swing. To find the limit cycle, I first evolved the equations for 100,000s in slow time with a time step of 0.01s, then when the transients had died down, I evolved the equations for 10 more seconds with a time step of 0.00001s. This time, both the metronomes were started with an initial angle of $\phi_1(0) = \phi_2(0) = 2\theta_0$. The time period was found to be approximately equal to the time period found before, 6.451s in slow time and 0.6112s in real time. For the plots below, $f = 200bpm$, and the rest of the parameters used are the same as before. Note that in the metronome plots, the lines for the two metronomes are overlapping each other.

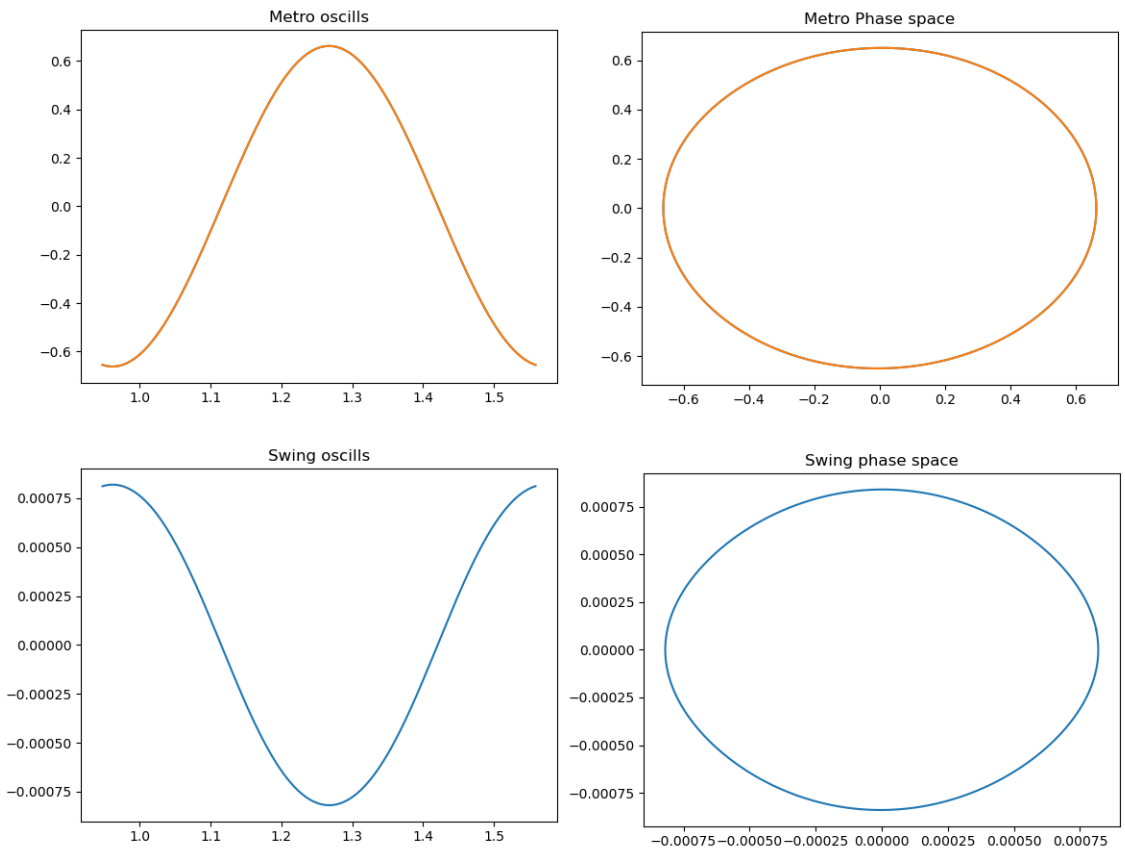


Figure 7.2: Limit cycle for the single metronome-swing system

We can clearly see some differences above. Notably, the limit cycles of the metronome has remained unchanged in both the cases. Surprisingly, the limit cycle of the swing has changed considerably. We can see that for $N = 1$ metronomes, the amplitude of the swing's oscillations is about 0.0004 rad, while for $N = 2$ metronomes, the amplitude is about 0.0008 rad, almost double the previous. (In both these cases, the equilibrium position of the swing is 0 rad.)

This drastic change in the limit cycle of the system due to the addition of a metronome points to the fact that the metronomes and swings *are coupled strongly*. To corroborate these findings, Abhishek and I made sure to replicate the results, which we were able to. Both of us generated the same limit cycles using the same initial points lying on the limit cycle.

## 7.2 Inferences

Our findings about the swing-metronome system point to the fact that a phase model cannot be built. Had the coupling been weak, we would have started out with a swing-metronome oscillator, then added $N - 1$ metronomes to this oscillator, then another swing-metronome oscillator, then another $N - 1$ to the second swing-metronome oscillator. Doing so would add interaction terms to the free-running oscillator equations of the 2N oscillators. These interaction terms would be used to finally build the phase model.

The finding that the metronomes and swings are coupled strongly defied all our expectations. It either means that a phase model cannot be built, or that some new technique must be found to incorporate a phase model.

# Chapter 8

# Final Inferences

Based on the results of the phase models built in section (3.4) and (3.5), we can see that there are two fixed points for $\chi' = G(\chi)$, one stable fixed point at $\chi = 0$, and one unstable fixed point at $\chi = \pi$. Applying a small angle approximation to the equations does not change these results (as expected). It means that the phase difference between the two metronomes will, given enough time, come down to 0, i.e. they will synchronize. Even an initial phase difference of $\pi$ will reduce down to zero due to small perturbations or errors in calculation. This is in full agreement with Pantaleone's experimental results for weak coupling of two metronomes, as well as with the solutions to the explicit equations found in section (3.3).

An unexpected outcome was the non-periodicity of $Q(t)$ for large values of $\theta_0$. In this specific case, the predictive power of the Phase Model fails. One reason for this failure might be the increase in the maximum magnitude of the interaction terms due to the increase in the amplitudes of the metronomes; we can see that an interaction term for the metronome: $\mu[(\frac{\theta}{\theta_0})^2 - 1]\omega$ is a term linear in $\omega$, whose maximum magnitude grows as $\theta_0$ grows. To reconcile these differences, a lower value of $\theta_0$ was used in our phase model.

Our inability to properly generate chimera states in the metronome-swing system numerically lead us to a very unexpected result, that the metronomes are strongly coupled to the swing they are placed on. Meanwhile, we found that the system is very sensitive to parameter values, and changing them gives very versatile results. One such unexplored state the system could be in is a phase-locked state.

# Chapter 9

# Conclusion

From the results we have seen that the Phase Model is a powerful technique for studying synchronization of coupled oscillators. It has worked well for toy examples like the ones mentioned in *Izhikevich*. With a little bit of tweaking, it also works very well to model real world systems being studied experimentally.

The main aim of this project was to learn the numerical technique of the Phase Model and to apply it in an academically relevant context. This has been accomplished. Apart from just learning the methods, we also learned to troubleshoot problems, such as the ones related to stability of solutions. Whenever faced with unstable solutions to differential equations (i.e. when they are blowing up), we tried integrating backward in time so the errors reduce instead of increase whenever an RK4 step is taken. This worked very well for Malkin's approach in finding $Q(t)$. We also learned to solve certain hard problems from the ground-up, as in when Abhishek's and my solutions were neither matching with each other, nor with those in Martens.

The unviability of the phase model technique for the swing-metronome system came as a blow to all of us involved. I hope that anyone after me pursuing this endeavor will be able to probe into it better, and come up with better results.

# Bibliography

[Pan02] James Pantaleone. "Synchronization of Metronomes". In: *American Journal of Physics* 70.992 (2002). DOI: 10.1119/1.1501118.

[Izh07] Eugene Izhikevich. "Dynamical Systems in Neuroscience". In: 2007. Chap. 10.

[al13] Martens et al. "Chimera states in mechanical Oscillator networks". In: *PNAS* 110.26 (2013). DOI: 10.1073.

# List of Figures

# Chapter 10

# Code Base

All the MATLAB code used by me to generate plots is given below.

## RK4 Differential Equation Solver

```
function V = rk42d(F,t,IC)
    siz = size(IC);
    tstep = t(2)-t(1);
    if(siz(1)==1)
        V = zeros(length(t),length(IC));
        V(1,:) = IC;
        for i = 1:1:length(t)-1
            k1 = F(t(i),V(i,:));
            k2 = F(t(i)+tstep/2,V(i,:)+k1*tstep/2);
            k3 = F(t(i)+tstep/2,V(i,:)+k2*tstep/2);
            k4 = F(t(i)+tstep,V(i,:)+k3*tstep);

            V(i+1,:) = V(i,:) + (tstep/6)*(k1+2*(k2+k3)+k4);
        end
    else
        if(siz(2)==1)
            V = zeros(length(IC),length(t));
            V(:,1) = IC;
            for i = 1:1:length(t)-1
                k1 = F(t(i),V(:,i));
                k2 = F(t(i)+tstep/2,V(:,i)+k1*tstep/2);
                k3 = F(t(i)+tstep/2,V(:,i)+k2*tstep/2);
                k4 = F(t(i)+tstep,V(:,i)+k3*tstep);

                V(:,i+1) = V(:,i) + (tstep/6)*(k1+2*(k2+k3)+k4);
            end
        end
    end
end
```

## Compound Simpson's Method Integrator

```
function sum = simpson(func)
    count = length(func);
    sum = (func(1)+func(count));
    for time = 2:1:count-1
        if(mod(time,2)==0)
            sum = sum+4*func(time);
        else
            sum = sum+2*func(time);
        end
    end
    sum = sum/3; %multiplication with timestep is expected to be done in the
                 %scope where the function is called
end
```

## Limit Cycle Generator

This function assumes that the phase space vector is a 2x1 column vector of length 2. Using a row vector will throw an error.

```
function [IC,count] = findLimitCycle(F,timestep)
    time = 0:timestep:1200;
    IC = [0;1];
    X = rk42d(F,time,IC);
    count = 1;
    distprev = 0;
    distcurr = 0;
    tolerance = timestep^2;
    len = length(time);
    IC = X(:,end);
    for c = len-1:-1:0
        distprev = distcurr;
        distcurr = ((X(1,c)-IC(1))^2+(X(2,c)-IC(2))^2);
        if(distcurr>=tolerance)           %when far away from endpoint,
            count = count+1;              %just increment count of points
        else
            if(distcurr>distprev)         %when nearby, check if approaching
                count = count+1;          %the endpoint or going away from it
            else                          %if approaching, then break
                break;
            end
        end
    end
end
```

# VDP and Andronov Hopf oscills. : Finding $Q(t)$

```
function XQ = malkin(timestep)
    %For van der Pol oscillator
    F1 = @(t,V)[V(1)-V(1)^3-V(2); V(1)];
    F2 = @(t,V1,V2)[(1-3*(V1(1)^2))*V2(1)+V2(2); -V2(1)];

%     for Andronov Hopf oscillator
%     F1 = @(t,V)[(V(1)-V(2))-V(1)*(V(1)^2+V(2)^2);
%                 (V(1)+V(2))-V(2)*(V(1)^2+V(2)^2)];
%     F2 = @(t,V1,V2)[(1-V1(2)^2-3*V1(1)^2)*V2(1) + (1-2*V1(1)*V1(2))*V2(2);
%                     -(1+2*V1(1)*V1(2))*V2(1) + (1-V1(1)^2-3*V1(2)^2)*V2(2)];

    %finding a point on the limit cycle and
    %how many timesteps are needed to evolve it
    [IC,count] = fullMet1Limit(F1,timestep);

    F = @(t,V)[F1(t,V(1:2));F2(t,V(1:2),V(3:4))];
    time = 0:timestep:(timestep*count);

    %for van der Pol oscillator
    Qic = [-0.371195702701162;0.264733342201252];
    norm = Qic'*F1(0,IC);
    Qic = Qic/norm;
    ICV = [IC;Qic];

    %for Andronov Hopf oscillator
    %ICV = [1;0;0;1];

    XQ = rk42d(F,time,ICV);
    XQ(3:4,:) = flip(XQ(3:4,:));

    plot(time,XQ(1,:))
    hold on
    plot(time,XQ(2,:));
    hold off
    title('Limit Cycle');
    xlabel('Time')
    ylabel('Variables')
    legend({'x','y'},'location','southeast')

    figure(2)
    plot(time,XQ(3,:));
    hold on
    plot(time,XQ(4,:));
    hold off
    title('Q')
```

```
    xlabel('Time')
    ylabel('Variables')
    legend({'Q_x','Q_y'},'location','southwest')
end
```

## VDP and A-H oscills. : Building the Phase Model

```
function findH
    timestep = 0.005;
    XQ = malkin(timestep);
    siz = size(XQ);
    count = siz(2);
    H = zeros(count+1,1);
    H1 = zeros(count+1,1);
    G = zeros(count+1,1);
    func = zeros(count+1,1);
    time = 0:timestep:(timestep*count);
    length(time)

    for phasediff = 0:1:count
        for t = 1:1:count
            offset = t+phasediff;
            while(offset>count)
                offset = offset - count;
            end
            func(t) = XQ(3,t)*(XQ(1,offset)-XQ(1,t));
        end
        sum1 = simpson(func);
        H(phasediff+1) = sum1/count;
    end
    H1 = flip(H);
    G = H1-H;

    figure(3)
    plot(time,H,'b');
    title('Phase Model')
    xlabel('Phase Difference')
    ylabel('H_\chi')
    grid on

    figure(4)
    plot(time,G,'g');
    title('G')
    xlabel('Phase Difference')
    ylabel('G_\chi')
    grid on
end
```

# Metronome Equations Solver

```
function metSolver2
    format long
    %all the constants
        bet = 0.011;
        del = 0;
        mu = 0.01;
        thet0 = 0.39;
    timestep = 0.005;
    time = 0:timestep:2000;
    M = @(t,V)[1-bet*(cos(V(2))^2), bet*cos(V(1))*cos(V(2));
               bet*cos(V(1))*cos(V(2)), 1-bet*(cos(V(1))^2)]
               *
    [(1+del)*sin(V(1)) + mu*((V(1)/thet0)^2-1)*V(3)
        + bet*cos(V(1))*((V(3)^2)*sin(V(1)) + (V(4)^2)*sin(V(2)));
    (1-del)*sin(V(2)) + mu*((V(2)/thet0)^2-1)*V(4)
        + bet*cos(V(2))*((V(3)^2)*sin(V(1)) + (V(4)^2)*sin(V(2)))];

    F = @(t,V)[V(3:4);
               M(t,V)/(bet*(cos(V(1))^2 + cos(V(2))^2) - 1)];

    IC = [0.7;-0.69;0;0];
    X = rk42d(F,time,IC);

    plot(time/10,X(1,:));
    hold on
    plot(time/10,X(2,:));
    hold off
    title('Angles of Metronomes');
    xlabel('Time (s)');
    ylabel('Angle (radians)');
    legend({'\theta_1','\theta_2'},'location','southwest');

    figure(2)
    plot(time/10,X(2,:)-X(1,:),'g');
    hold on
    plot(time/10,X(2,:)+X(1,:),'m');
    hold off
    xlabel('Time (s)')
    ylabel('Angle (radians)')
    title('Sum and difference of angles');
    legend({'\theta_2 - \theta_1','\theta_1 + \theta_2'},'location','southwest');
end
```

## Small Angle Approximation: $Q(t)$

```
function XQ = metMalkin(timestep,bet,del,mu,thet0)
    format long
    F1 = @(t,V)[V(2);
        -((1-bet)/(1-2*bet))*((1+del)*V(1) + mu*((V(1)/thet0)^2-1)*V(2))
            - ((bet)/(1-2*bet))*V(1)*(V(2)^2)];
    [IC,count] = findLimitCycle(F1,timestep);
    count*timestep
    time = 0:timestep:(count*timestep);

    %This is the Exact Jacobian
    F2 = @(t,V1,V2)-[0, ((bet-1)/(1-2*bet)) *
                            ( 1+del + (2*mu*V1(1)*V1(2))/(thet0^2) )
                            - (bet*(V1(2)^2))/(1-2*bet);

                     1, ((bet-1)*mu*((V1(1)/thet0)^2-1))/(1-2*bet)
                        - (2*bet*V1(1)*V1(2))/(1-2*bet)]
                        * V2;

    F = @(t,V)[F1(t,V(1:2));F2(t,V(1:2),V(3:4))];

    Qic = [0;1];
    norm = (Qic')*F1(0,IC);
    Qic = Qic/norm;
    ICV = [IC;Qic];
    XQ = rk42d(F,time,ICV);

    plot(time/10,XQ(1,:));
    hold on
    plot(time/10,XQ(2,:));
    hold off
    title('Limit Cycle');
    xlabel('Time')
    ylabel('Variables')
    legend({'\theta','\omega'},'location','southwest')

    figure(2)
    plot(time/10,XQ(3,:));
    hold on
    plot(time/10,XQ(4,:));
    hold off
    title('Q')
    xlabel('Time')
    legend({'Q_\theta','Q_\omega'},'location','northeast')
end
```

## Exact Equations: $Q(t)$

```
function XQ = fullMetMalkin(timestep,bet,del,mu,thet0)
    F1 = @(t,V)[V(2);
                ((1+del)*sin(V(1)) + mu*((V(1)/thet0)^2-1)*V(2)
                    - (bet/2)*sin(2*V(1))*(V(2)^2))/(bet*(cos(V(1))^2) - 1)];
    F2 = @(t,V)-[0, ((1+del)*((bet*(cos(V(1))^2)-1)*cos(V(1))
                        + bet*sin(V(1))*sin(2*V(1)))
                        + mu*V(2)*(2*V(1)*(bet*(cos(V(1))^2)-1)
                        + bet*(V(1)^2)*sin(2*V(1)))/(thet0^2)
                        + mu*V(2)*bet*sin(2*V(1))
                        - (bet*(V(2)^2)/2)*(2*(bet*(cos(V(1))^2)-1)*cos(2*V(1))
                        + bet*(sin(2*V(1))^2)))/((bet*(cos(V(1))^2)-1)^2);
                1, (mu*((V(1)/thet0)^2 - 1)
                        - bet*V(2)*sin(2*V(1)))/(bet*cos(V(1)^2)-1)]
                *V(3:4);
    F = @(t,V)[F1(t,V(1:2)); F2(t,V)];

    [IC,count] = findLimitCycle(F1,timestep);
    time = 0:timestep:(count*timestep);

    Qic = [1;1];
    norm = (Qic')*F1(0,IC);
    Qic = Qic/norm;
    ICV = [IC;Qic];
    XQ = rk42d(F,time,ICV);

    plot(time/10,XQ(1,:))
    hold on
    plot(time/10,XQ(2,:));
    hold off
    title('Limit Cycle');
    xlabel('Time')
    ylabel('Variables')
    legend({'\theta','\omega'},'location','southeast')

    figure(2)
    plot(time/10,XQ(3,:));
    hold on
    plot(time/10,XQ(4,:));
    hold off
    title('Q, \theta_0 = 0.59')
    xlabel('Time')
    ylabel('Variables')
    legend({'Q_{\theta}','Q_{\omega}'},'location','southwest')
    grid on
end
```

# Phase Model of Metronomes

```
ffunction metH(timestep)
    %all the constants
        bet = 0.011;
        del = 0;
        mu = 0.01;
        thet0 = 0.2;

    %XQ = metMalkin(timestep,bet,del,mu,thet0); %for small angle eqs
    XQ = fullMetMalkin(timestep,bet,del,mu,thet0); %for full eqs

    sizXQ = size(XQ);
    count = sizXQ(2);
    timevect = 0:timestep:(count*timestep);
    func1 = zeros(count+1,1);
    H = zeros(count+1,1);
    H1 = zeros(count+1,1);

    %interaction terms for small angle approx
    %F = @(V)-((bet)/(1-2*bet))*(( (1+del)*V(1) + mu*((V(1)/thet0)^2-1)*V(2))
    %          - V(1)*V(2)^2);

    %interaction terms for full set of eqs
    %D is the full set of equations for both the oscills
    %Subtracting D from the expression for a free running oscill
    %   gives us the interaction terms

    D = @(V1,V2)[1-bet*(cos(V2(1))^2), bet*cos(V1(1))*cos(V2(1))]
                *
                [(1+del)*sin(V1(1)) + mu*((V1(1)/thet0)^2-1)*V1(2)
                    + bet*cos(V1(1))*((V1(2)^2)*sin(V1(1))
                    + (V2(2)^2)*sin(V2(1)));
                (1-del)*sin(V2(1)) + mu*((V2(1)/thet0)^2-1)*V2(2)
                    + bet*cos(V2(1))*((V1(2)^2)*sin(V1(1))
                    + (V2(2)^2)*sin(V2(1)))];
    F = @(V1,V2)((1+del)*sin(V1(1)) + mu*((V1(1)/thet0)^2-1)*V1(2)
                    - (bet/2)*sin(2*V1(1))*(V1(2)^2))/(1-bet*(cos(V1(1))^2))
                    - D(V1,V2)/(1-bet*(cos(V1(1))^2 + cos(V2(1))^2));

    for phasediff = 0:1:count
        for phase = 1:1:count
            offset = phase+phasediff;
            while(offset>count)
                offset = offset - count;
            end
            %for full equations
```

```
            func1(phase) = XQ(4,phase)*F(XQ(1:2,phase),XQ(1:2,offset));
            %for small angle eqs
            %func1(phase) = XQ(4,phase)*F(XQ(1:2,offset));
        end
        func1(count+1) = func1(1);
        sum1 = simpson(func1);
        H(phasediff+1) = sum1/(count);
    end
    H1 = flip(H);
    G = H1-H;
    figure(3)
    plot(timevect/10,H,'b');
    title('Phase Model')
    xlabel('Phase Difference')
    ylabel('H')
    legend('H_{12}(\phi_2 - \phi_1)','location','southwest')
    grid on

    figure(4)
    plot(timevect/10,G,'g');
    title('G')
    xlabel('Phase Difference')
    ylabel('G')
    legend('G(\chi)','location','southeast')
    grid on
end
```

# Chimera States generator

The script below is written in the Julia language.

```
using LinearAlgebra
using PyPlot

#all physical params

N = 15;
freqBpm = 160;
m = 0.028;
M = 2.31;
l = 0.15;
L = 0.22;
k = 68;
g = 9.81;

l_bob = 0.073 - 0.00022*freqBpm;
r_cm = abs(0.178*l_bob - 0.0121);
I = 0.0000129 + 0.005*(l_bob^2);

#derived params
x0 = (m*r_cm)/M;
omega = sqrt((m*g*r_cm)/I);
kappa = (k/M)*((l/L)^2);
Omega = sqrt(g/L);

#all the constants:

mu_m = 0.011;         #nonlinearity of metros
mu_s = 0.00016;      #damping of swings
thet0 = 0.33;

#all the params that can be customized

#kappa = 20;        #coupling of swings
#x0 /= 100;


#nondimensional parameters
beta = (x0*(omega^2))/g;
omeg_r2 = (Omega/omega)^2;
chi = kappa/(omega^2);

println("x0        : ",x0)
println("omega     : ",omega)
println("kappa     : ",kappa)
```

```
println("Beta      : ",beta)
println("Omega^2_r : ",omeg_r2)
println("Chi       : ",chi)

timestep = 0.01;
maxTime = 1000;

println("Num of oscills : ",maxTime/(2*pi));

timeVec = collect(0:timestep:maxTime);
timeCount = length(timeVec);
println("Count is: ",timeCount);
V = zeros(Float64,2*N+2,2,timeCount); #The vector used in the ODEs
R = zeros(Float64,2*N);       #To store reused values
F = zeros(Float64,2*N+2,2,4);       #To store function values
V_ = zeros(Float64,2*N+2,2);       #To store intermediate RK4 step vectors
cos_Phi_Psi = zeros(Float64,(N,1));   #To calculate sum of cos

#initializing V with ICs
V[1:N,1,1] .= 2*thet0;   #for AP IC
#V[N+1:2*N,1:2,1] = 2*thet0*rand(Float64,(N,2)) .- thet0;
V[N+1:2*N,1,1] = [0.14157382;
  0.2614215;
 -0.29608024;
 -0.24948547;
 -0.07842186;
 -0.31545333;
  0.06066683;
  0.27495881;
  0.32353985;
 -0.19375554;
  0.07076737;
  0.12151782;
 -0.28229673;
  0.14350184;
  0.03906534];
V[N+1:2*N,2,1] = [0.04105613;
 -0.22835415;
  0.05830795;
 -0.20626723;
 -0.15800162;
 -0.21524905;
  0.18515062;
  0.05557701;
 -0.13233109;
 -0.31297807;
 -0.01390028;
```

```
  0.28235838;
  0.0213475;
 -0.20703707;
 -0.04162385];

#used to evaluate the function value at a given V_
function findFunct(V_,rk)
R = sin.(V_[1:2*N,1]) .+ mu_m.*(((V_[1:2*N,1]./thet0).^2).-1).*V_[1:2*N,2]
#note this peculiarity in julia,
#sin, cos, exp etc. all these functions have to be
#dotted to be applied onto vectors
cos_Phi_Psi = cos.(V_[1:N,1])
sum1Phi = dot(cos_Phi_Psi, cos_Phi_Psi)
cos_Phi_Psi = cos.(V_[N+1:2*N,1])
sum1Psi = dot(cos_Phi_Psi, cos_Phi_Psi)

sum2Phi = sum(R[1:N].*cos.(V_[1:N,1]) .+ sin.(V_[1:N,1]).*(V_[1:N,2].^2))
sum2Psi = sum(R[N+1:2*N].*cos.(V_[N+1:2*N,1])
        .+ sin.(V_[N+1:2*N,1]).*(V_[N+1:2*N,2].^2))

F[2*N+1,2,rk] = (-(chi + omeg_r2)*V_[2*N+1,1] - mu_s*V_[2*N+1,2]
           + chi*V_[2*N+2,1] + sum2Phi)/(1 - beta*sum1Phi)
F[2*N+2,2,rk] = (-(chi + omeg_r2)*V_[2*N+2,1] - mu_s*V_[2*N+2,2]
           + chi*V_[2*N+1,1] + sum2Psi)/(1 - beta*sum1Psi)

F[1:(2*N+2),1,rk] = V_[1:2*N+2,2]
F[1:N,2,rk] = -(R[1:N] + beta.*cos.(V_[1:N,1]).*F[2*N+1,2,rk])
F[N+1:2*N,2,rk] = -(R[N+1:2*N] + beta.*cos.(V_[N+1:2*N,1]).*F[2*N+2,2,rk])

return nothing
end

#RK4 implementation
for t = 1:(timeCount-1)
V_ = V[:,:,t]
for rk = 1:4
findFunct(V_,rk)
if rk<3
V_ = V[:,:,t] + (timestep/2)*F[:,:,rk]
elseif rk==3
V_ = V[:,:,t] + timestep*F[:,:,rk]
else
V[:,:,t+1] = V[:,:,t] + (timestep/6)*(F[:,:,1] + 2*(F[:,:,2]+F[:,:,3])
                                                    + F[:,:,4])
end
end
end
```

```
V[2*N+1:2*N+2,:,:] = V[2*N+1:2*N+2,:,:].*(x0/L);

#plotting angles of metronomes and swings
figure(1)
for x = 1:N
plot(timeVec./omega,V[x,1,:])
end
title("All Phi's")
xlabel("Slow Time (s)")
ylabel("Metronome Angle (rad)")

figure(2)
for x = 1:N
plot(timeVec./omega,V[N+x,1,:])
end
title("All Psi's")
xlabel("Slow Time (s)")
ylabel("Metronome Angle (rad)")

figure(3)
plot(timeVec./omega,V[2*N+1,1,:])
plot(timeVec./omega,V[2*N+2,1,:])
title("Swing angles")
xlabel("Slow Time (s)")
ylabel("Deviation from Equilibrium Swing Angle (rad)")

show()
```

## Swing-Metronome System Solver

```
using LinearAlgebra
using PyPlot

N = 2;
freqBpm = 200;
m = 0.028;
M = 2.31;
l = 0.15;
L = 0.22;
g = 9.81;

l_bob = 0.073 - 0.00022*freqBpm;
r_cm = abs(0.178*l_bob - 0.0121);
I = 0.0000129 + 0.005*(l_bob^2);

x0 = (m*r_cm)/M;
```

```
omega = sqrt((m*g*r_cm)/I);
Omega = sqrt(g/L);

mu_m = 0.011;
mu_s = 0.00016;
thet0 = 0.33;

beta = (x0*(omega^2))/g;
omeg_r2 = (Omega/omega)^2;

println("x0        : ",x0)
println("omega     : ",omega)
println("Free TimP : ",(2*pi)/omega)
println("Beta      : ",beta)
println("Omega^2_r : ",omeg_r2)

timestep = 0.01;
maxTime = 1000;
#repe = 10; #to be able to repeat iterations of maxTime over the vector
oscill = round(maxTime/(2*pi)) + 2;
println("Num of oscills : ",maxTime/(2*pi));

timeVec = collect(0:timestep:maxTime);
timeCount = length(timeVec);
timeVec = timeVec .+ (repe*maxTime);
println("Count is: ",timeCount);
global V = zeros(Float64,N+1,2,timeCount);
R = zeros(Float64,N)
F = zeros(Float64,N+1,2,4);
global V_ = zeros(Float64,N+1,2);

V[1:N,1,1] .= 2*thet0

function findFunct(V_,rk)
R = sin.(V_[1:N,1]) .+ mu_m.*(((V_[1:N,1]./thet0).^2).-1).*V_[1:N,2];
sum1Phi = sum(cos.(V_[1:N,1]).^2);
sum2Phi = sum(R.*cos.(V_[1:N,1]) .+ sin.(V_[1:N,1]).*(V_[1:N,2].^2));

F[1:N+1,1,rk] = V_[1:N+1,2];
F[N+1,2,rk] = (-omeg_r2*V_[N+1,1] - mu_s*V_[N+1,2]
            + sum2Phi)/(1 - beta*sum1Phi);
F[1:N,2,rk] = -(R .+ beta.*cos.(V_[1:N,1]).*F[N+1,2,rk]);

return nothing;
end

#uncomment these lines to find limit cycle, put repe=10
```

```
#global vecDiff1 = 0;
#global vecDiff2 = 0;
#global timestep2 = timestep^2
#global limitCount = 0
#for i = 1:repe
for t = 1:(timeCount-1)
V_ = V[:,:,t]
for rk = 1:4
findFunct(V_,rk);
if rk<3
V_ = V[:,:,t] + (timestep/2)*F[:,:,rk]
elseif rk==3
V_ = V[:,:,t] + timestep*F[:,:,rk]
else
V[:,:,t+1] = V[:,:,t] + (timestep/6)*(F[:,:,1]
           + 2*(F[:,:,2]+F[:,:,3]) + F[:,:,4])
end
end
#uncomment these lines to find limit cycle
#global vecDiff1 = vecDiff2;
#global vecDiff2 = sum((V[1,:,t+1].-init[1,:]).^2);
#if(vecDiff2<=timestep2)
# if(vecDiff2<vecDiff1)
# global limitCount = t+1;
# break
# end
#end
end
#V[:,:,1] = V[:,:,end];
#end

V[N+1,:,:] = V[N+1,:,:]*(x0/L);

figure(1)
for i = 1:N
plot(timeVec/omega,V[i,1,:])
#plot(timeVec[1:limitCount]/omega,V[i,1,1:limitCount])
end
#plot(timeVec[1:limitCount]/omega,V[2,1,1:limitCount])
#plot(timeVec/omega,V[1,1,:])
#plot(timeVec/omega,V[2,1,:])
title("Metro oscills")

figure(2)
#plot(timeVec[1:limitCount]/omega,V[N+1,1,1:limitCount])
plot(timeVec/omega,V[N+1,1,:])
title("Swing oscills")
```

```
figure(3)
for i = 1:N
plot(V[i,1,:],V[i,2,:])
#plot(V[i,1,1:limitCount],V[i,2,1:limitCount])
end
#plot(V[1,1,:],V[1,2,:])
#plot(V[2,1,:],V[2,2,:])
title("Metro Phase space")

figure(4)
#plot(V[N+1,1,1:limitCount],V[N+1,2,1:limitCount])
plot(V[N+1,1,:],V[N+1,2,:])
title("Swing phase space")
show()
```