University of Passau

Faculty of Computer Science and Mathematics

Chair of Dynamical Systems

Master Thesis in Mobile and Embedded Systems

Machine Learning Control applied in Synchronization of Kuramoto Oscillators

Submitted by

Richa Chauhan

1. Examiner: Prof. Dr. Fabian Wirth
2. Examiner: Prof. Dr. Dirk Sudholt
Supervisor: Prof. Dr. Fabian Wirth
Date: March 22, 2022

Contents

Abstract

In this thesis, we will be discussing techniques from machine learning. We will explore how genetic programming is used as an effective method to find control laws. We will focus on how Machine Learning Control works and see an example of implementing genetic programming for the control problem of synchronization in the networks of Kuramoto oscillators.

We will be studying how the formation of synchronization patterns is characterized in a network of Kuramoto oscillators, and cluster synchronization in networks of Kuramoto oscillators, necessary and sufficient conditions on the interconnection weights of network and on the natural frequencies of oscillators will be derived to ensure that the phases of groups of oscillators develop cohesively with each other, yet independently from the phases of oscillators in different clusters. Additionally, a control mechanism is developed to modify the edges of a network to ensure the formation of desired synchronization clusters. As the control mechanism finds the smallest perturbation (in the Frobenius norm) for a desired synchronization pattern that is agreeable with a fixed set of structural restrictions, so we will observe optimal control mechanism. The powerful methods from machine learning are used to find effective control laws. Machine learning is used to create models of a system from data, these models should improve with more data.

Machine learning control (MLC) is motivated by issues in complex control tasks where it can be impossible or difficult to model the system and generate a useful control law. Alternatively, collected data and measurements are used to learn controllers directly. Hence, MLC directly gains knowledge of control laws of the system. MLC technique minimizes a given cost function to generate associated control laws. Genetic Programming is a flexible machine learning algorithm, which we will explore to find control laws in MLC.

We will implement MLC as a control mechanism on the control problem of synchronization in the networks of Kuramoto oscillators. We will setup the experiments used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators to find control law and after applying the GP (used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators, we will observe the results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized.

Keywords - Machine Learning Control, Genetic Programming, Synchronization of Kuramoto Oscillators, Cluster synchronization conditions, control mechanism.

Acknowledgments

I am incredibly grateful to Prof. Dr Fabian Wirth for his support, knowledge, and positive feedback during my Master thesis. Furthermore, he not only suggested the idea of my Master's Thesis but also provided invaluable guidance and continuous assistance throughout the entire process. Additionally, I am grateful to the University of Passau. It has been a pleasure spending time in Germany because of their kind assistance and support. Finally, I want to thank my parents, teachers, and friends for their encouragement, because without their support, I would not have been able to complete my studies.

List of mathematical symbols

$\|\Delta\|_F^2$ Frobenius norm of the perturbation matrix. 26, 27

$\circ$ Hadamard Product. 7, 25, 27, 52, 53, 54, 58, 61, 62, 64

$\imath$ Imaginary unit. 10, 11, 55

Δ Perturbation matrix. 26, 27, 28

$\oslash$ Element-wise division. 27

γ Penalization coefficient. 40, 49, 50, 55, 64, 67, 79

List of Figures

List of Tables

1 Introduction

In this Chapter, we will discuss the two important topics synchronization patterns in the networks of Kuramoto oscillators and Machine Learning Control (MLC) that we will discuss in detail in this thesis. We will discuss the synchronization of oscillators, the formation of synchronization patterns is characterized in a network of Kuramoto oscillators, and we will present conditions for cluster synchronization. We will describe the mechanism of Machine Learning Control to explore about the working of Genetic Programming (GP) further used in this thesis and will implement this control technique to the control problem of synchronization patterns in networks of Kuramoto oscillators in our thesis. Firstly, we will state the motivation for the thesis in this Chapter, then we will describe the main research goals of the thesis, and finally, we will describe the outline of this thesis in detail.

1.1 Motivation

1.1.1 Synchronization patterns in networks of Kuramoto Oscillators

Synchronization of oscillators is vital for the right functionality of numerous regular and man-made complex systems. There are systems that require complete synchronization among all the parts for proper functioning and some systems depend on the cluster or partial synchronization. In cluster synchronization, the subsets of nodes show coherent behaviors that remain independent from the evolution of other oscillators in the network. In [Tib+17], the complete focus is on networks of Kuramoto oscillators that are described in [Kur75], and intrinsic and topological conditions are characterized that confirms the formation of desired clusters of oscillators. The formation of synchronization patterns is characterized in a network of Kuramoto oscillators. Particularly, we will learn the conditions on the weights and structure of the network, and on the natural frequencies of oscillators that allow the phases of a group of oscillators to develop cohesively, still independently from the phases of oscillators in different clusters. These conditions are applicable to weighted networks of heterogeneous oscillators. Surprisingly, although the oscillators show nonlinear dynamics, the approach depends completely on tools from linear algebra and graph theory. Further, a control mechanism is developed to find the smallest (in the Frobenius norm) network perturbation to make sure the generation of the desired synchronization pattern. A procedure is provided, we restrict the set of edges that can be modified, consequently enforcing the sparsity structure of the network perturbation. The outcomes in [Tib+17] are verified through a set of numerical examples.

1.1.2 Machine Learning Control (MLC)

In feedback control systems, we have noticed a few challenges. Machine Learning Control can tackle these problems. MLC is a technique that learns by trail and error to find an effective control law. Sometimes, there are systems in which it is really impossible to predict control policy effect. However, it can be comparatively easy to test the control policy effect in a system. Then it is possible to evolve the control policy by testing, exploring alternative control policies and exploiting good control policies. Evolutionary strategies in design problems of fluid mechanics follow these principles. We can see *Machine Learning Control* as a strategy using any of the data-driven regression techniques like evolutionary programming (EP), genetic algorithms (GA), genetic programming (GP), and other techniques (decision trees, support vector machines (SVM), and neural networks) to discover effective control laws.

Machine learning control gives us a powerful new framework to control complex dynamical systems that are as of now beyond the capacity of existing techniques in control theory. The physical system is modified by actuators i.e input u through a control law trained by sensor measurements of the system i.e output y. Machine learning use data to generate models of the system, then these models improve with more data. We use machine learning algorithms in a complex system to find an effective control law - $u = K(y)$, that maps the output (sensors, y) to the input (actuators, u) of the system. Machine learning control is motivated by problems in complex control tasks where it might be impossible to model the system and discover useful control law. As an alternative, we directly use experience and data to gain knowledge of effective control laws. From [DBN16, chapter 2], we can claim that in the classic framework, the machine learning has been used to model the input–output dynamics of a system. Then based on these models controllers are designed by using traditional methods. Machine learning control avoids this process and directly gain knowledge of effective control laws without needing a model of the system. Figure 1 shows machine learning control architecture. MLC aims to minimize cost function J and find an effective control law. The learning loop trains controller by providing experienced data.

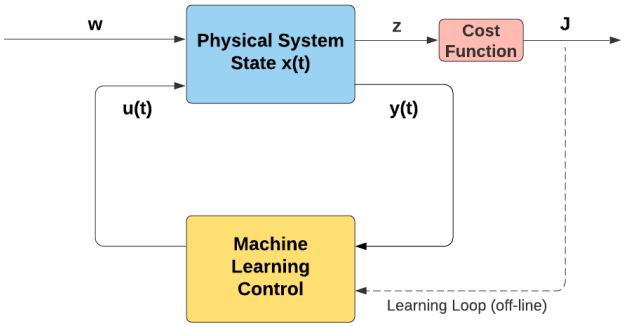

Figure 1.1: Schematic of Machine Learning Control

1.2 Main research goals

In the thesis, we will describe the mechanism of Machine Learning Control, explore about the working of Genetic Programming (GP) and will implement this control technique to the control problem of synchronization patterns in networks of Kuramoto oscillators. We will analyse about the conditions for cluster synchronization, understand and setup problem of control method used to modify the desired synchronizaton method and use Machine Learning Control technique as control mechanism. Based on our research, in our thesis, we will be understanding the working of the these methods in detail and implement them. We will be identifying the following concrete research questions by observing the results after implementation.
Finally, the expected results of the thesis will be as follows:

- We will be studying the synchronization of oscillators, the formation of synchronization patterns is characterized in a network of Kuramoto oscillators, and the conditions for the cluster synchronization. We will also study the methods and implementation of machine learning control (MLC) in detail to understand the mechanism and use it further on the system of the networks of Kuramoto oscillators
- We will apply MLC technique as the control mechanism to the control problem used to ensure the synchronization patterns in the desynchronized system of the networks of Kuramoto oscillators.
- We will observe the results after implementing MLC to synchronization control problem of the desynchronized system of the networks of Kuramoto oscillators to know whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized.

1.3 Outline of the Thesis

The thesis begins by stating the introduction to the synchronization patterns in networks of Kuromto Oscillators and Machine Learning control 1. Chapter 2 describes the background knowledge required for this thesis, Graph Theory and Matrix Analysis are explained which are tools for the networks of the Kuromoto oscillators.

In Chapter 3, we will see the overview of Kuramoto's classic coupled oscillators. We will discuss the strength of coupling necessary between each oscillator for the oscillators to converge to a common phase and frequency. The main objective of this chapter is to understand the synchronization and cluster synchronization of Kuramoto oscillators and their conditions. And finally, we will study the control mechanism for forming the desired synchronization pattern by the minimization of the problem.

In Chapter 4, we will thoroughly discuss *Machine Learning Control*, which is a data-driven regression strategy that uses evolutionary programming (EP), genetic algorithms (GA), genetic programming (GP), and other techniques (decision trees, support vector machines (SVM), and neural networks) to discover effective control laws. Firstly, we will describe feedback control as MLC uses feedback control and highlight the concepts from machine learning with a focus on evolutionary algorithms. Furthermore, in this Chapter, Machine learning will be

mentioned as a tool for designing control laws. Machine learning control (MLC) offers a robust new approach for controlling the complex dynamical systems that can be controlled more effectively than any other currently existing methods in control. Further in this Chapter, the use of GP as a search algorithm to find control laws in MLC. The method of machine learning control (MLC) is described in detail. In addition to this, we will analyse the use of genetic programming as an effective method to discover control laws. Finally, we will provide implementation and analyse illustrative example to better understand these concepts.

In Chapter 5, we will be discussing the baseline of our thesis - MLC applied in the networks of Kuramoto oscillators. For this, firstly, we will setup our system and describe the control mechanism used to modify the desired synchronization patterns in networks of Kuramoto oscillators along with the cost function to apply the Genetic Programming (GP used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators. This Chapter will be describing the development of the system the networks of Kuramoto oscillators used for the task of applying MLC and then will discuss the desynchronization regime for the system so that system becomes desynchronized to apply control input to control the system to ensure the synchronization of the system. The control mechanism for the desynchronized system of the networks of Kuramoto oscillators is explained further in this Chapter. Finally in the last of this Chapter, the setup MLC to apply it on the desynchronized system of the networks of Kuarmoto oscillators instead of normal controller to ensure the synchronization of the system is provided.

Chapter 6 will discuss the experiments with the results based on the methods described in Chapter 5 in detail. The simulations of the networks of Kuramoto oscillators are produced in this Chapter and we will desynchronize the system of the networks of Kuramoto oscillators by changing the system parameters. We will make the coupling strength weak to make the system desynchronized. We will add control input to the Kuramoto model equation to introduce controller on this uncontrolled desynchronized system of Kuramoto oscillators to achieve the synchronization patterns. We will also discuss the experimental setup used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators to find best control law. After applying the GP (used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators, we will observe the results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized.

In the last Chapter 7, we will conclude the thesis and talk about the possible future explorations in the related topics.

2 Preliminaries

The tools from Graph theory and linear algebra are used for the networks of Kuramoto oscillators (Chapter 3). In this Chapter, we are going to understand the tools from Graph theory and Matrix analysis in Section 2.1 and Section 2.2, respectively. Deep insight of Graph Theory and Matrix analysis is provided in [GR01] and [HJ85], respectively. In Section 2.1, we will describe the concepts from the Graph theory that are directed graph, undirected graph, and matrices related to graph like adjacency matrix, degree matrix and Laplacian matrix. In Section 2.2, we will define the concepts from Matrix Analysis that are used this thesis that are the image of matrix A and Hadamard product.

2.1 Graph Theory

Graph theory is a useful tool to describe the topology of a network. A graph is an ordered pair $\mathcal{G} = (\nu, \epsilon)$ of a set of nodes or vertices $\nu = \{1, 2, \ldots, N\}$ and a set of links or edges $\epsilon \subseteq \nu \times \nu$. Each element of ϵ is an ordered pair of different vertices and not the self-loop of vertices. If the edge $(j, i) \in \epsilon$ for all $(i, j) \in \epsilon$ then the graph is undirected, else it is directed as for the edge $(i, j) \in \epsilon$ there will be a fixed direction from j to i and the reverse direction may not be possible. If the nodes are connected from j to i, then the edge (i, j) represents the influence from node i to node j, while if the graph is undirected, then the edge represents the mutual influence between the two vertices.

Graphs are described by two matrices, the adjacency matrix and the Laplacian matrix. An undirected graph's adjacency matrix A is defined as follows: if there exists a link between node i and j, i.e., if $(i, j) \in \epsilon$ then $a_{ij} = a_{ji} = 1$; otherwise $a_{ij} = a_{ji} = 0$. Here, we consider simple graphs, for which the diagonal terms occur as $a_{ii} = 0$ for all $i = 1, \ldots, N$. This indicates that there are no self-loops. A directed graph, on the other hand, takes into consideration the direction of interaction, with $a_{ij} = 1$ if $(i, j) \in \epsilon$. Accordingly, an adjacency matrix is not symmetric for a directed graph.

In the case of undirected graphs, the degree d_i of vertex i is given as the number of connections incident to node i, i.e., $d_i = \sum_{j=1, j\neq i}^{N} a_{ij} = \sum_{j=1, j\neq i}^{N} a_{ji}$ for $i = 1, 2, \ldots, N$. In the case of a directed graph, rather, in-degree and out-degree can be determined by taking into the account the links from the node: $d_i^{out} = \sum_{j=1, j\neq i}^{N} a_{ij}$ and $d_i^{in} = \sum_{j=1, j\neq i}^{N} a_{ji}$.

In the case of graph $\mathcal{G}$, the Laplacian matrix L is defined as $L_{ij} = -a_{ij}$ if $i \neq j$, and $L_{ii} = d_i^{out}$. Both Laplacian L and adjacency matrices A are $N \times N$ matrices; which are connected as $L = D - A$, in this case, D represents a diagonal matrix that includes the node out-degrees d_i^{out} , i.e., $D = \operatorname{diag}\left\{d_1^{out}, d_2^{out} \ldots d_N^{out}\right\}$.

As we have seen Laplacian matrix of the graph is the difference of the adjacency matrix and degree matrix of the particular graph, this matrix is very helpful for the in-depth under-

standing of the particular graph topology. The adjacency matrix A describes how an agent is physically connected to its neighbours, while the degree matrix D of matrix A indicates the number of neighbours connected along a diagonal.

In the following, we give an example of Laplacian matrix for the graph shown in Figure 2.1.

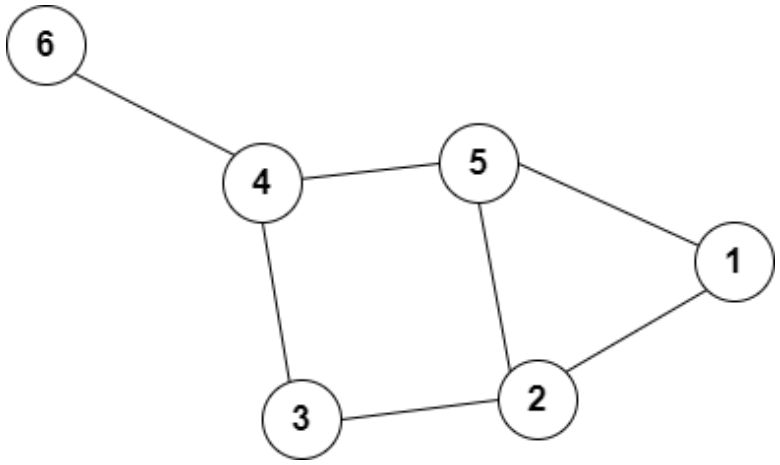

Figure 2.1: Undirected Graph for the example of Laplacian graph which is constructed from its adjacency matrix and degree matrix

Laplacian matrix L for graph is constructed from graph's adjacency matrix A and degree matrix D. Let A be the adjacency matrix and D be the degree matrix, they are given below:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

then L, Laplacian matrix is:

$$L = D - A = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

A graph is weighted if each edge is associated with a weight (a real number). Weighted graphs consist of a set of edges, a set of vertices, and a set of weights assigned to each edge. It is also possible to represent these graphs using an adjacency matrix in which the coefficient a_{ij} is equal to the edge weight of the edge (i, j).

Given a graph $\mathcal{G} = (\nu, \epsilon)$, a path from a vertex i to a vertex j is a sequence of links

$(i_1, j_1), \ldots, (i_k, j_k)$ such that $i_1 = i$,$j_k = j$ and $j_m = i_{m+1}$ for all $m = 1, \ldots, k-1$. More concretely, it is a sequence of links starting in i, ending in j and so that the end of one link is always the start of the next one.

A strongly connected graph is a directed graph $\mathcal{G}$ so that for each pair i, j of vertices of $\mathcal{G}$, there exists a path from i to j. We will use strongly connected graphs for the networks used in the modelling of Kuramoto oscillators explained in the Chapter 3.

2.2 Matrix Analysis

Matrix theory and linear algebra have long been essential tools for the Graph theory. Matrix analysis is discuused in [HJ85], where classic results and recent ones are presented that have proved to be useful in applied mathematics. In concrete, we will need the following two concepts that are used in Chapter 3 for the networks of Kuramoto oscillators. First concept is the image of a matrix A, it is the space (set of all the linear combinations) of its column. Hence, called as column space of a matrix A.

Second concept is Hadamard product, let A and B be the matrices of order $m \times n$ (m rows and n columns) with entries in $\mathbf{C}$. The Hadamard product '$\circ$' of A and B is defined by:

$$[A \circ B]_{ij} = [A]_{ij} [B]_{ij}$$

for all $1 \leq i \leq m,\ 1 \leq j \leq n$.

Hadamard's product is simple entrywise multiplication. Due to this, the Hadamard product inherits the same advantages (and limitations) as multiplication in $\mathbf{C}$. Also note that A and B must be of the same size, but are not necessarily square.

3 Synchronization patterns in networks of Kuramoto Oscillators

Kuramoto Oscillators are employed to model the system dynamics of numerous oscillators that are initially out of phase and frequency synchronization. In this Chapter, we will study the concept of synchronization in Section 3.1. In Section 3.2, we will provide the overview of Kuramoto's classic coupled oscillators and discuss some insight of preliminary notions necessary to appropriately use this model and discuss the strength of coupling necessary between each oscillator for the oscillators to converge to a common frequency. In Section 3.3, we will describe the networks of Kuramoto oscillators. The purpose and objective of this chapter is to understand the synchronization and cluster synchronization of Kuramoto oscillators and their conditions which described in Section 3.4. Finally in Section 3.5, we will study the control mechanism for forming the desired synchronization pattern. The thesis lays the framework for applying the machine learning control technique to ensure the synchronization of networks of Kuramoto oscillators.

3.1 Synchronization

Synchronization can be recognized and leveraged in computer science and engineering systems [PRK03]. For example, in data mining, which is the process of searching for patterns in massive amounts of data to extract meaning and knowledge, computer scientists encode data vectors representative of a database as frequencies of oscillators. The hope is that synchronization of the different frequencies will group similar data together and hence categorically give meaning to a heterogeneous space of complex data. This work is present in [Are+08].

There are plenty of examples of synchronization to be found throughout history presented in [PRK03]. In addition, several examples of synchronization provide a broad recognition that synchronization can be used to model any number of events that exhibits a state transition from complex heterogenic systems to those of homogenous systems [Are+08], be it through spatial and/or temporal changes. From [Are+08], we gained a deep understanding of synchronization phenomena when oscillating elements are deprived of interaction in complex network topology. This paper also presents analytical approaches to the problem, extensive numerical work and a review on several examples of synchronization in complex networks.

A synchronization network is basically a network of coupled dynamical systems. It consist of connected oscillators that forms a network, where oscillators are nodes that produce a signal with almost similar frequency, and also have ability of receiving a signal. Synchronization is the phase change where the almost complete network of oscillators starts pulsing at the same frequency. There are mainly two properties of synchronizations that are frequency synchronization and phase synchronization. These both synchronizations are seen as follows:

1. **Frequency synchronization:** A frequency synchronization is achieved when all the frequencies merge to a standard constant frequency (i.e. let us say $\omega_{sync} \in \mathbb{R}$) as time $t \to \infty$.
2. **Phase synchronization:** A phase synchronization is achieved when all the phases converge to the same value as time $t \to \infty$.

3.2 Kuramoto Model

The presentation in this section is based on [KA75], [Win67], [Str00], and [DB14]. Here, we review basics of the theory of Kuramoto oscillators.

Overview of "Kuramoto Oscillators"

In a 1975 contribution [KA75], Yoshiki Kuramoto first proposed his model of complex oscillators and how best to model their behaviour towards in and out of phase synchrony. However, Kuramoto was not the first to produce a mathematical model describing coupled oscillators. Arthur Winfree is associated with setting down the framework for modelling the behaviour of large interacting oscillators. Although his model is not often used, his assumptions and insight into the correct implementation of the model still stand today.

Winfree proposed that for the oscillators to achieve synchrony in [Win67], oscillators first need to be almost identical and secondly that their coupling is weak. These two assumptions were necessary to decrease the complexity of the mathematics, describing a large network of complex coupled oscillators that would otherwise have many state variables. In addition, Winfree's assumptions allowed the model to be reduced to representing each oscillator's effect on the group at large through their phase relationships to each other.

Kuramoto took Winfree's assumptions and further expanded on them by allowing Winfree's model to be generalized among any networks of oscillators. Kuramoto also provided weighting factors to the phase relationship between each oscillator to better model the bounds necessary for synchronization in [Str00]. It was these contributions that enabled Kuramoto's oscillator model to gain academic reputation.

Kuramoto's equation is given as:

$$\dot{\theta}_i = \omega_i + \sum_{j=1}^{N} a_{ij} \sin(\theta_j - \theta_i), \qquad i = 1, \cdots, N \tag{3.1}$$

Or for the special case of $a_{ij} = \frac{K}{N}, \quad 1 \leq j \leq N$

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^{N} \sin(\theta_j - \theta_i), \qquad i = 1, \cdots, N \tag{3.2}$$

where N is the total number of networked oscillators, $a_{ij} \in \mathbb{R}$ or $K \in \mathbb{R}$ are the coupling parameters, i i is an index. This is already obvious from equation 3.1. $\theta_i \in \mathbb{S}^1$ is the phase angle of the ith oscillator and $\omega_i \in \mathbb{R}$ is the natural frequency of the ith oscillator.

For uniformity in modelling, it is usually best practice to establish the initial frequencies of ω_i normally distributed about a centre controlling the mean frequency of the modeller's choice. As mentioned previously, the main draw from researchers to the Kuramoto model is in the K constant, K is known as the coupling constant described in [DB14]. There is a particular value of the coupling constant, which is called the critical coupling constant. Namely, above the critical coupling constant synchronization occurs and below it it does not. Kuramoto [KA75] proved that if K exceeds a particular threshold constant (this threshold is the critical coupling constant) with the distribution of the natural frequencies, then synchronization is certain.

3.2.1 Importance of the Critical Coupling Constant

Kuramoto [KA75] determined that there is a value $K_{critical}$, if $K < K_{critical}$ then a system of networked oscillators does not achieve synchrony. However, when the coupling constant exceeds ($K > K_{critical}$) the critical coupling constant, then individual oscillators achieve synchronization.

An advantageous way to visually describe Kuramoto's model is through a set of nodes on a unit circle distributed randomly around the circle initially and given random initial frequencies. Kuramoto modelled these nodes on a unit circle to help understand the critical coupling constant. This new performance measure introduced is invented as the performance measure of phase cohesiveness known as the complex order parameter. Kuramoto defined this order parameter as:

$$re^{\imath\phi} = \frac{1}{N}\sum_{j=1}^{N} e^{\imath\theta_j} \tag{3.3}$$

where $\imath$ is now the imaginary unit, r is the modulus of the order operator or measure of synchronization and ϕ is the argument of the order. Another way to view the "order operator" is to say that it represents the mean collective behaviour of all the networked coupled oscillators. In other words, when $r = 0$, the system is said to be completely unsynchronized, whereas when $r = 1$, the system is said to be in complete synchrony.

This order parameter r is seen as the vector in the complex plane pointing out towards the edge of the circle with its magnitude representing the synchrony of the system, as shown in Figure 3.1.

It is seen from Figure 3.1, the length of r increases as the oscillators converge together in both phase and frequency synchronization.

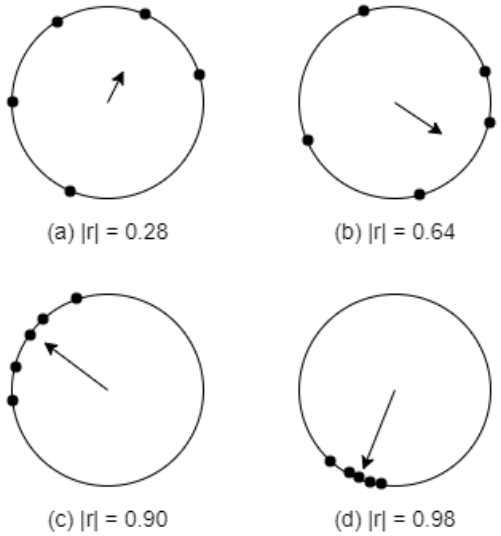

Figure 3.1: Order Parameter r denoting the cohesiveness of a particular system

3.2.2 Deriving the Coupling Constant's Relationship to the Modulus Order Parameter

It is beneficial to relate the coupling constant K to the modulus order parameter r to understand better how the modulus parameter, which is a metric of synchrony, relates to the strength of coupling K.

We begin by multiplying both sides of equation (3.3) by $e^{-\imath\theta_i}$:

$$re^{\imath(\phi-\theta_i)} = \frac{1}{N}\sum_{j=1}^{N} e^{\imath(\theta_j-\theta_i)}, \qquad i = 1,\ldots,N \tag{3.4}$$

considering only the imaginary parts we are left with

$$r\sin(\phi-\theta_i) = \frac{1}{N}\sum_{j=1}^{N} \sin(\theta_j-\theta_i), \qquad i = 1,\ldots,N \tag{3.5}$$

Equation (3.5) is now substituted into equation (3.2) to arrive at a form that represents the

Kuramoto model in terms of the modulus parameter and the coupling constant:

$$\dot{\theta}_i = \omega_i + Kr\sin(\phi - \theta_i), \qquad i = 1, \ldots, N \tag{3.6}$$

Equation (3.6) offers us helpful insight into the synchronization of complex oscillators that was not obvious from equation (3.2). If the oscillators synchronize, then they move towards a mean order parameter $re^{\imath\phi}$. It is also clear that with small coupling constants of K that each oscillator will adhere to its natural frequency ω_i. On the other hand, with a significant value of K, each oscillator "feels" the influence of its neighbours either "pushing" or "pulling" and will entrain together toward the mean order parameter $re^{\imath\phi}$.

3.2.3 Determination of the Critical Coupling Constant

Determining the order parameter related to the coupling constant provides us with intrinsic dynamics about a particular system. For example, when we observe the order parameter, the measure of synchrony against a varying coupling constant, it allows us to identify the critical coupling constant that K must be greater than, to guarantee synchronization. This implementation is based on the code [Mun10].

For a complete phase and frequency synchronization, the modulus order parameter r must be equal to one. In reality, r never reach one but rather approaches one as $K \to \infty$ as is suggested by figure 3.2.

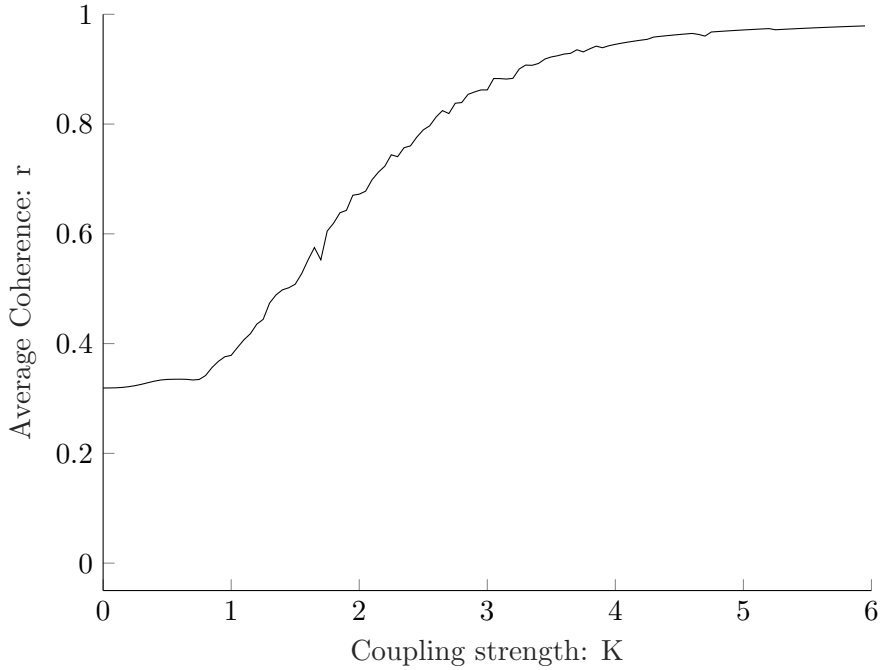

Figure 3.2: Order parameter as a function of coupling strength for 10 oscillators

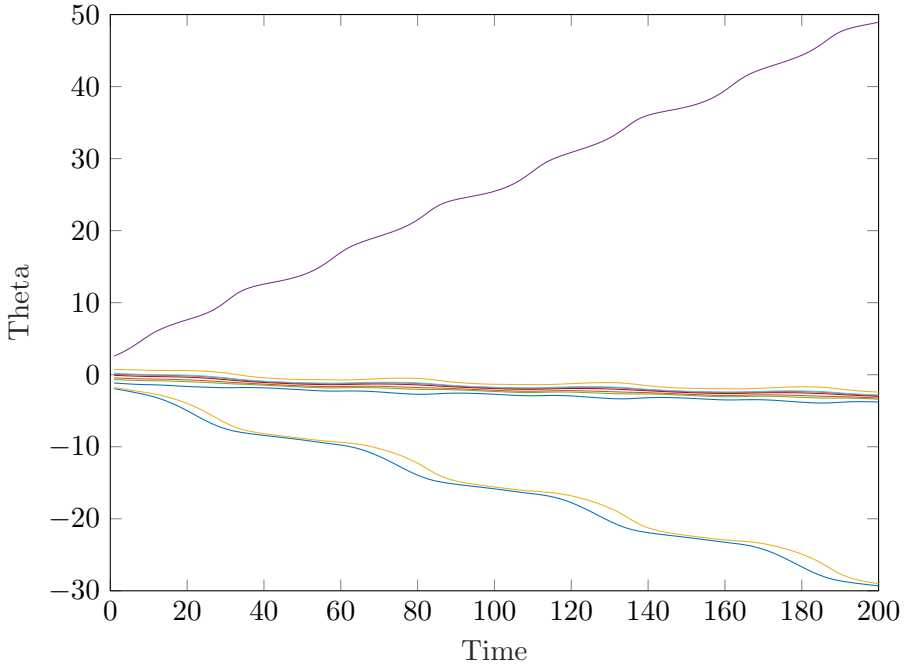

Figure 3.3: Phase relationships of 10 Kuramoto oscillators for $K = 2$

From Figure 3.2, it is clearly visible that for $0 < K < 3.5$ synchronization does not takes place as order parameter r starts to approach one after $K = 3$ or $K = 3.5$.

To portray this, we consider a system with 10 oscillators distributed randomly both in their initial phase and frequency. Figure 3.3, 3.4, and 3.5 illustrates the phase relationship of the 10 oscillators through time with a varying coupling constant $K = 2$, $K = 3.5$, and $K = 6$, respectively.

Figure 3.6, 3.7, and 3.8 illustrates the frequency relationship of the 10 oscillators through time with a varying coupling constant $K = 2$, $K = 3.5$, and $K = 6$, respectively.

As figure 3.4 shows there is relatively good phase synchronization of 8 of the 10 oscillators. The two oscillators that have drifted away from the group have synchronized in frequency, as shown as in figure 3.7. The previous these two figures illustrate that phase synchronization and frequency synchronization are independent of each other. While the 10 oscillators did achieve frequency synchronization, they did not achieve phase synchronism.

With only 10 Kuramoto oscillators, it has become evident that excellent simultaneous phase and frequency synchronization may not be possible depending on each oscillator's initial phase and frequency. However, the power of this analysis is evident in that we now determined that the oscillators tend to be more synchronized at coupling constants of 3 or greater.

Expansion on Kuramoto Oscillators

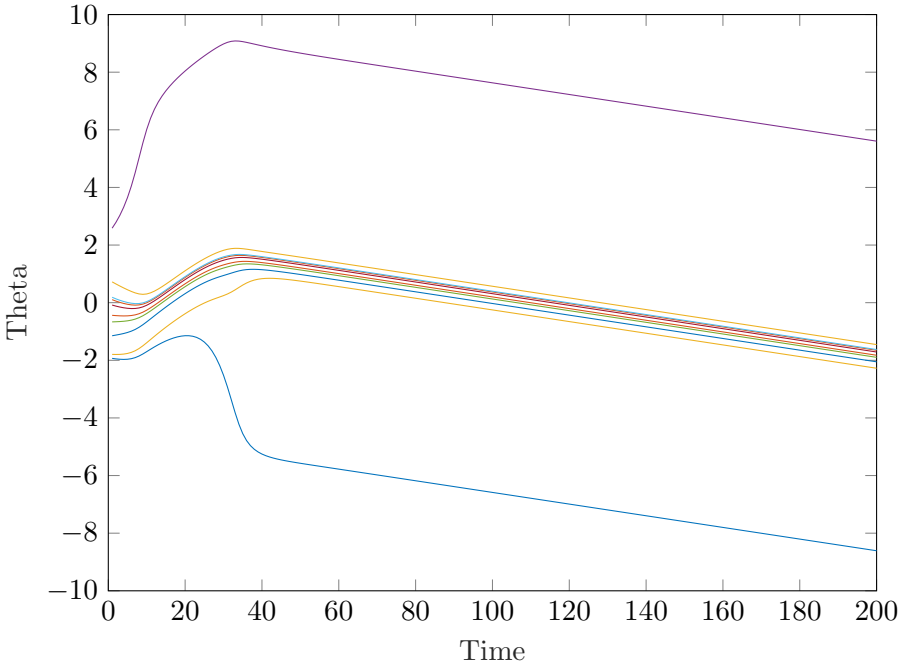

Figure 3.4: Phase relationships of 10 Kuramoto oscillators for $K = 3.5$

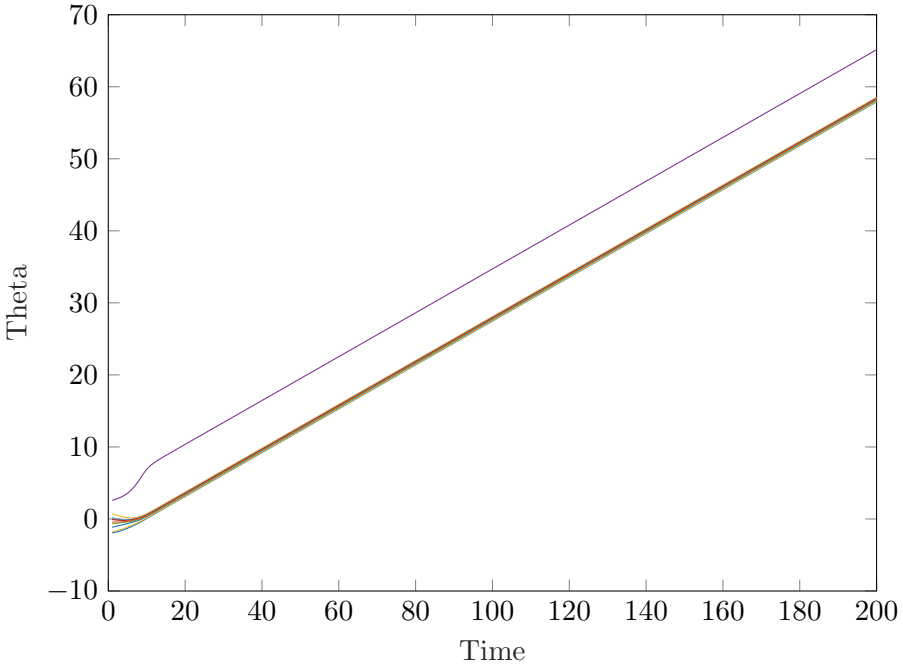

Figure 3.5: Phase relationships of 10 Kuramoto oscillators for $K = 6$

As shown in Figures 3.4 and 3.7, we can see that the "drifting" oscillators are almost mirrored images of each other and therefore cancel each other out in their contribution to the modulus order parameter r. From statistics, we know that outliers are eventually "phased" out as we

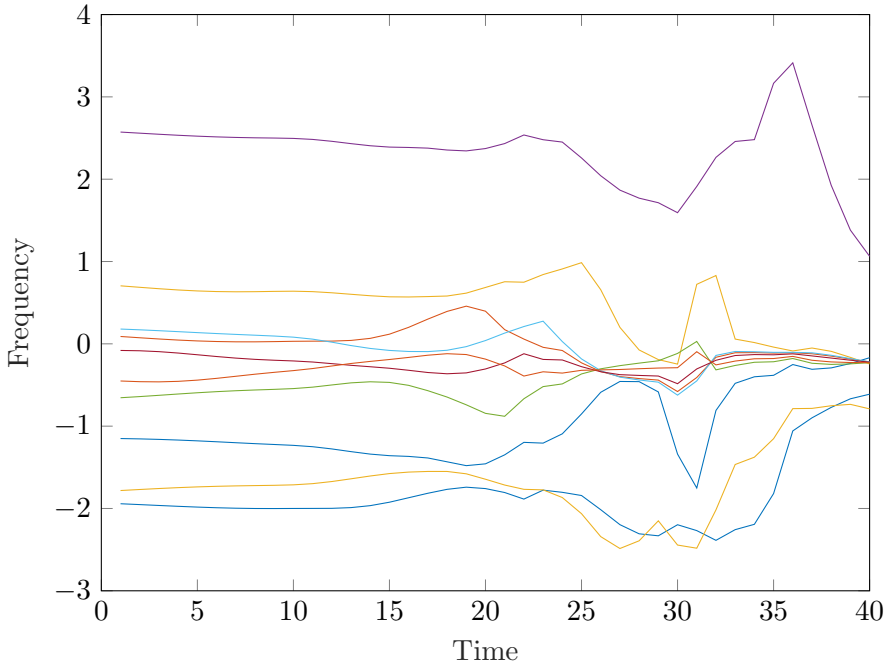

Figure 3.6: Frequency relationships of 10 Kuramoto oscillators for $K = 2$

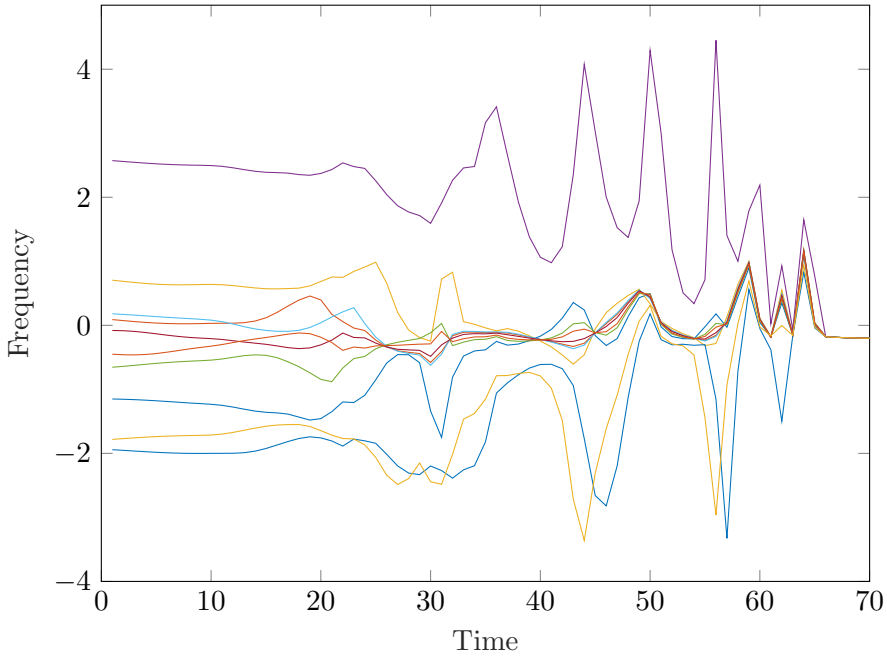

Figure 3.7: Frequency relationships of 10 Kuramoto oscillators for $K = 3.5$

increase the number of samples taken from a population. This phenomenon is well known as the Central Limit Theorem. In much the same way, Kuramoto Oscillators exhibit this same behaviour as more identical oscillators are introduced to the complex networked system. Figures 3.9 and 3.10 demonstrate how increasing the number of oscillators in a network has

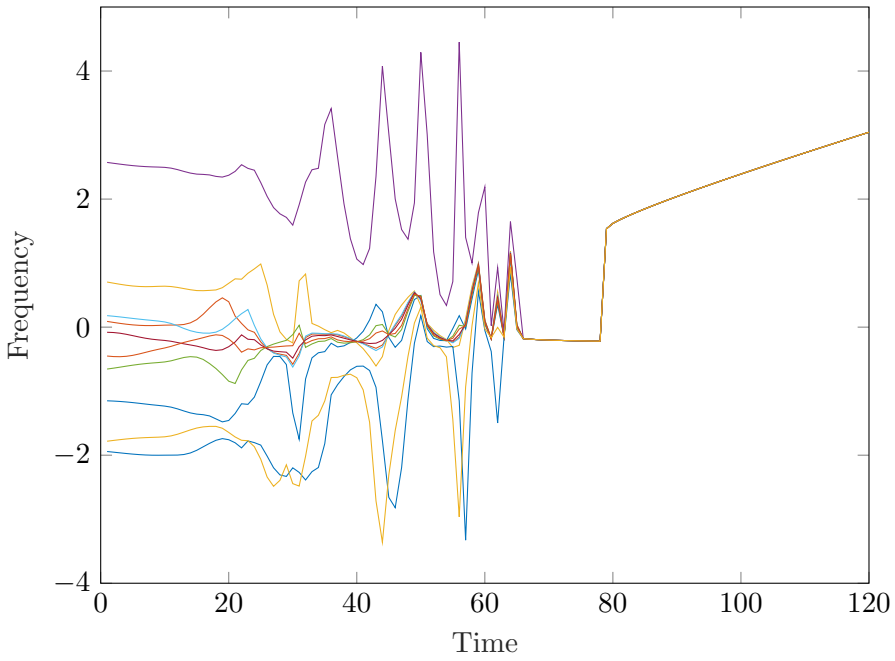

Figure 3.8: Frequency relationships of 10 Kuramoto oscillators for $K = 6$

the effect of improving dynamic non-linear behaviour by increasing the tightness of each of the oscillators phase and frequency relationships to the group as a whole.

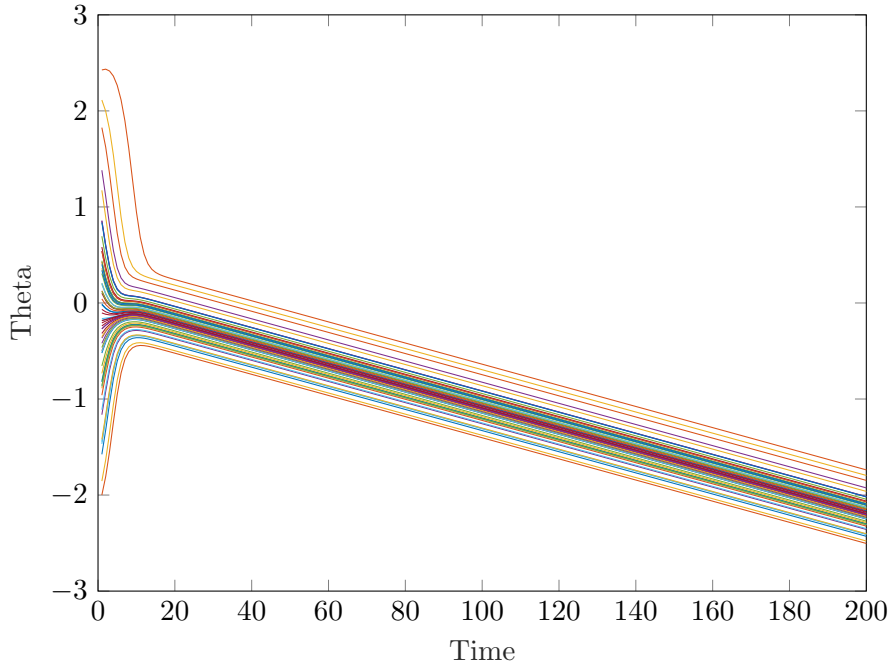

Figure 3.9: Phase relationships of 60 Kuramoto oscillators for $K = 6$

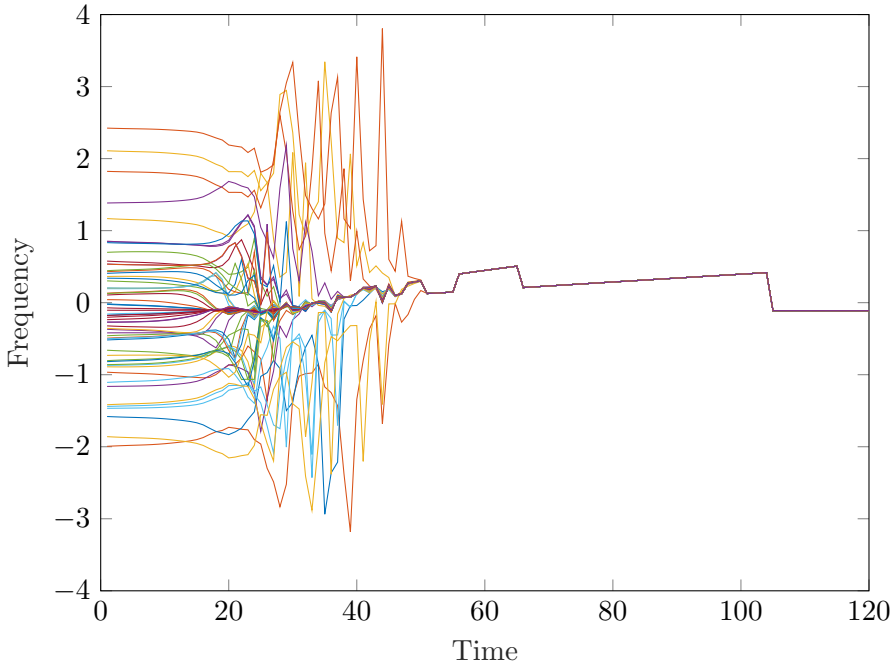

Figure 3.10: Frequency relationships of 60 Kuramoto oscillators for $K = 6$

The critical coupling constant for this particular system is still three. Still, it is clear that the effect of many more oscillators added to a specific system has the effect of smoothing the coupling strength contribution to the average coherence of the system, as well as tightening the phase relationships between all the oscillators.

3.3 Networks of Kuramoto Oscillators

A network consists of connected oscillators, where oscillators are nodes. Nodes are connected by their edges/links and produce a signal with almost similar frequency and also have the ability to receive a signal. Graph theory is a valuable tool to describe the topology of a network, and graph theory is previously discussed in detail in Section 2.1 of Chapter 2.

Each element within the graph $\mathcal{G} = (\nu, \epsilon)$ represents a set of agents/vertices/nodes $\nu = \{1, \ldots, N\}$ and a set of edges/links denoted as $\epsilon \subseteq \nu \times \nu$. A set of edges describes the connectivity of graph. All the edges that connect to a particular node and its neighbors (when nodes are directly connected) is given as:

$$N_i = \{j \in \nu : (i, j) \in \epsilon\}$$

In this section, we will discuss the setup for the networks of Kuramoto oscillators, describe a few synchronization notions and metrics for the networks of Kuramoto oscillators like definitions of frequency synchronizability, phase synchronizability and characteristic matrix along with an example showing how the setup and these definitions work.

3.3.1 Problem Setup and Definitions

In this section, we want to setup a problem of the network of Kuramoto oscillators and later define the terms important for the cluster synchronization of the network of Kuramoto oscillators. This problem received attention from the authors and here I want to explain the work of other authors.

[GMA07; ZSL14] describes a complete synchronization in the networks of Kuramoto oscillators and shows that synchronization of all different nodes appears when the coupling strength between the nodes is larger than the heterogeneity of natural frequencies of the oscillators. Pattern formation and Partial synchronization are focused less in the literature, which comprises only a few recent works. In [Pec+14], the connection between network symmetries and the formation of clusters is shown. Furthermore, this work describes how the symmetry among the interconnections leads to partial synchronization. From [Sor+16], it is clear that graph symmetry methods are used to find all the existing clusters in the networks of Laplacian-coupled oscillators. The relationship between network topology and clusterization has been analyzed in [LLC10] for unweighted interconnections in the network. The occurrence and stability of groups of nodes that are synchronized within a network for distinct dynamics classes, like neuronal spiking models and delay-coupled laser models, is described in the work of [DLS12]. Here, the master stability approach is used where the master stability function indicates a discrete rotational symmetry dependent on the number of groups. In [Fav+17; FCP17], the concept of cluster synchronization is put forward to study the idea of cluster synchronization in the networks of Kuramoto oscillators through linear systems theory tools. These studies represent a quantitative approach to know that cluster synchronization depends on weak inter-cluster connections and strong intracluster. And also define the similarity for the oscillators' natural frequencies within each cluster and the heterogeneity of the coupled oscillators' natural frequencies belonging to different subnetworks. The work of [Sch+16] shows that the cluster synchronization is related to the idea of an equitable external partition in the network's graph, and it is closer to [Tib+17]. In fact in [Tib+17], the idea of an equitable external partition is analyzed in terms of the network adjacency matrix's invariant subspaces. However, the study in [Sch+16] is moved forward with undirected and unweighted networks. The study of [Tib+17] shows that the conditions discussed in [Sch+16] may not be essential when dealing with weighted and directed networks. Therefore, in [Tib+17], the approach depends on simple notions from linear algebra, directing the control algorithm's development to form desired patterns. In [Tib+17], the complete focus is on networks of Kuramoto oscillators, and intrinsic and topological conditions are characterized that confirms the formation of desired clusters of oscillators. The formation of patterns of synchronization is characterized in a network of Kuramoto oscillators.

Now, we follow the setup of the problem which is taken from [Tib+17]. We have taken Kuramoto oscillators that can be **homogeneous and heterogeneous**; In homogeneous Kuramoto oscillators, oscillators have similar properties, while in heterogeneous Kuramoto oscillators, oscillators have different properties.

Consider a network of N heterogeneous Kuramoto oscillators represented by the digraph $\mathcal{G} = (\nu, \epsilon)$, where $\nu = \{1, \ldots, N\}$ is the set of oscillators and $\epsilon \subseteq \nu \times \nu$ is the set of edges (describing connectivity of the network).

Let the matrix $A = [a_{ij}]$ be the weighted adjacency matrix of $\mathcal{G}$, where $a_{ij} \in \mathbb{R}$ if $a_{ij} \in \epsilon$ and

$a_{ij} = 0$ otherwise. Assume that $\mathcal{G}$ is strongly connected and $\mathbb{S}^1$ is the unit-circle. Let $\theta_i \in \mathbb{S}^1$ stand for the phase angle of the i-th oscillator, whose dynamics is described as a generalized version of heterogeneous Kuramoto system as shown in equation (3.1) of Section 3.2:

$$\dot{\theta}_i = \omega_i + \sum_{j=1}^{N} a_{ij} \sin(\theta_j - \theta_i), \qquad i = 1, \cdots, N$$

where $\omega_i \in \mathbb{R}$ denotes the natural frequency of the i-th oscillator. Various oscillatory patterns are possible dependent on the interconnection graph $\mathcal{G}$, the oscillator natural frequencies ω_i, and the weighted adjacency matrix $A = [a_{ij}]$. These different patterns can be in partial or cluster synchronization states (as we will define in 3.3.2). We are specifically interested in the case of cluster synchronization, where phases of groups of oscillators develop cohesively within each group; however, it is independent of the phases of oscillators in distinct groups.

For this purpose, $P = \{P_1, \ldots, P_M\}$ is taken as partition of ν, the sets P_i, $i \in \{1, \ldots, M\}$ are nonempty, pairwise disjoint subsets of $\{1, \ldots, M\}$ with the property that $\nu = \bigcup_{i=1}^{M} P_i$. In [Tib+17], it is assumed that the oscillators are labeled so that:

$$P_i = \left\{ \sum_{j=1}^{i-1} |P_j| + 1, \ldots, \sum_{j=1}^{i} |P_j| \right\}$$

where $|P_j|$ denotes the cardinality of the set P_j.

3.3.2 Synchronization Notions and Metrics

Synchronization of oscillators is vital for the proper functioning of numerous regular and artificial complex systems. Network of Kuramoto oscillators can show complete, partial and cluster synchronization. For these networks, complete synchronization occurs when all the oscillators are identically synchronized, and partial synchronization arises when a few oscillators but not all the oscillators are identically synchronized. When similar groups of oscillators identically synchronize in partial synchronization, no synchronization exists among the different groups of oscillators; this is called cluster synchronization. In addition, some systems need complete synchronization among all the system parts for proper functioning, and some systems depend on partial and cluster synchronization. There exist different notions of synchronization; we understand the following definitions:

Definition 1 (**Frequency Synchronizability** [Tib+17])**.** Let the network of oscillators be $O = (\mathcal{G}, \omega, A)$ where $\mathcal{G} = (\nu, \epsilon)$, ω is a set of natural frequencies of oscillators, and $A = [a_{ij}]$ is the weighted adjacency matrix, then the partition $P = \{P_1, \ldots, P_M\}$ of ν is frequency synchronizable if it holds:

$$\dot{\theta}_i(t) = \dot{\theta}_j(t)$$

for some initial phases $\theta_1(0), \ldots, \theta_N(0)$, for all times $t \in \mathbb{R}_{\geq 0}$ and all $k \in \{1, \ldots, M\}$, with all $i, j \in P_k$.

□

Definition 2 (**Phase Synchronizability** [Tib+17])**.** Let the network of oscillators be $O = (\mathcal{G}, \omega, A)$ where $\mathcal{G} = (\nu, \epsilon)$, ω is a set of natural frequencies of oscillators, and $A = [a_{ij}]$ is the weighted adjacency matrix, then the partition $P = \{P_1, \ldots, P_M\}$ of ν is phase synchronizable if it holds:

$$\theta_i(t) = \theta_j(t)$$

for some initial phases $\theta_1(0), \ldots, \theta_N(0)$, for all times $t \in \mathbb{R}_{\geq 0}$ and all $k \in \{1, \ldots, M\}$, with all $i, j \in P_k$.

□

Undoubtedly, phase synchronization implies frequency synchronization, while the opposite statement is not true in general.

The characteristic matrix correlated with a partition P of the network nodes is defined, used to obtain our synchronization conditions in Section 3.4.

Definition 3 (**Characteristic matrix** [Tib+17])**.** Let the graph of the network of oscillators be $\mathcal{G} = (\nu, \epsilon)$ and the partition $P = \{P_1, \ldots, P_M\}$ of ν, then the characteristic matrix of P is $V_P \in \mathbb{R}^{N \times M}$ where,

$$V_P = [v_1 \; v_2 \; \ldots v_M],$$

and

$$v_{ij}^T = [\underbrace{0\;0\;\ldots 0}_{\sum_{j=1}^{i-1}|P_j|} \; \underbrace{1\;1\;\ldots 1}_{|P_i|} \; \underbrace{0\;0\;\ldots 0}_{\sum_{j=i+1}^{N}|P_j|}]$$

i.e.

$$v_{ij}^T = \begin{cases} 1 & \text{if } j \in P_i \\ 0 & \text{if } j \notin P_i \end{cases}$$

here, $|P_j|$ is cardinality of the set P_j. □

The following Example 1 is similar to the example present in [Tib+17], this example shows how the setup and Definition 1, 2 and 3 works.

Example 1. *Let us take an example, consider the network of Kuramoto oscillators* $O = (\mathcal{G}, \omega, A)$ *as shown in Figure 3.11, with graph* $\mathcal{G} = (\nu, \epsilon)$*, where set of nodes* $\nu = \{1, 2, 3, 4, 5, 6\}$ *and with partition* $P = \{P_1, P_2\}$*, where* $P_1 = \{1, 2, 3\}$ *and* $P_2 = \{4, 5, 6\}$*.*

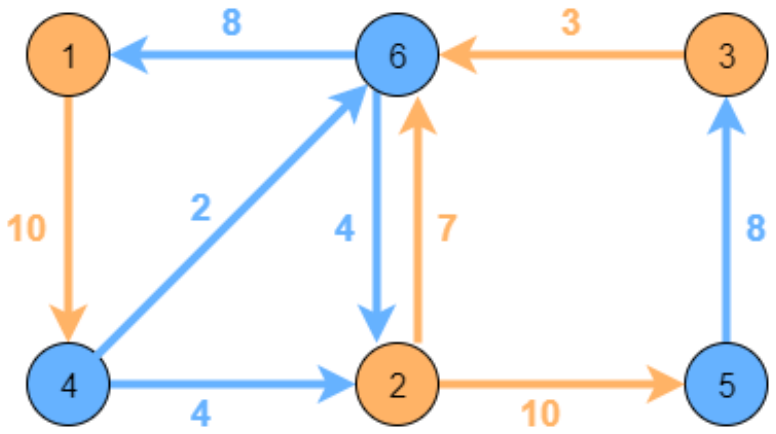

Figure 3.11: A network of oscillators with partitions $P_1 = \{1, 2, 3\}$ and $P_2 = \{4, 5, 6\}$.

The graph $\mathcal{G}$ *and the partition* P *are depicted by the weighted adjacency matrix* A *and the characteristic matrix* V_P *as given below respectively:*

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 & 8 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 7 & 3 & 2 & 0 & 0 \end{bmatrix}$$

and

$$V_P = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Figure 3.11 depicts a network of oscillators with partition sets $P_1 = \{1, 2, 3\}$ *and* $P_2 = \{4, 5, 6\}$*. The sum of all edges'* (i, j) *weights is equal for each node* i *of* P_1 *(respectively* P_2*) incoming from* $j \in P_2$ *(respectively* $j \in P_1$*). We will see this condition in Section 3.4 as this is a necessary condition in the network of Kuramoto oscillators for phase synchronization of the partition* P *of* ν*.*

3.4 Cluster synchronization

In cluster synchronization, the subsets of nodes show coherent behaviours independent from the evolution of other oscillators in the network.

Let the geodesic distance between the two angles $\theta_i, \theta_j \in \mathbb{S}^1$ be $|\theta_j - \theta i|$. In this context, the following definition is introduced here from [[FCP17], Definition 1]:

Definition 4 (Cluster of oscillators). The set of oscillators $\mathcal{C} \subseteq \nu$ is a cluster if there exists an angle $0 \leq \gamma \leq \frac{\pi}{2}$ such that, if $|\theta_j(0) - \theta_i(0)| \leq \gamma$, then $|\theta_j(t) - \theta_i(t)| \leq \gamma$, for all $i, j \in \mathcal{C}$ and at all times $t \geq 0$. □

Figure 3.12 shows a network of 9 oscillators with 3 colour-coded clusters. The phases of the oscillators within each cluster develop cohesively, as defined in the above definition and displayed in Figure 3.12. The frequencies don't have to be equal for the nodes i.e. oscillators to belong to the same cluster.

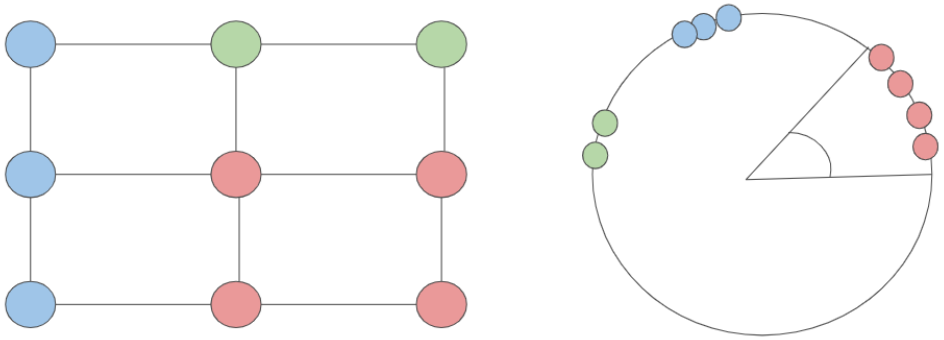

Figure 3.12: A network of 9 oscillators with 3 colour-coded clusters

After all the analysis, we focused on a fundamental notion of cluster synchronization. First, we identified the necessary and sufficient conditions on weights of the network of oscillators and the natural frequencies of oscillators to emerge the desired synchronization pattern. Then, we used our study to develop a structural control mechanism to form the desired synchronization pattern.

3.4.1 Conditions for Cluster synchronization

Unlike the complete synchronization, the technique allowing cluster synchronization in the networks of Kuramoto oscillators have not been properly characterized in [FCP17]. In [Tib+17], the conditions for cluster synchronization are studied, which will be discussed further in this section. A idea of exact cluster synchronization is considered where the oscillators' phases remain equal within each cluster over the period of time and distinct from the oscillators' phases in different clusters over the period of time. The necessary and sufficient

conditions are derived to form a given pattern of synchronization in weighted and directed graph of the networks of Kuramoto oscillators.

Specifically, it is shown that cluster synchronization can occur if and only if:

1. The natural frequencies of oscillators within each cluster are equal,
2. The sum of the weights of the edges incoming from every separate group of nodes for each node in a cluster is equal for all nodes in that cluster.

In this section, the necessary and sufficient conditions are derived to ensure phase and frequency synchronization of a partition P of oscillators. Particularly, it is shown that how synchronization of a partition depends on the weights and structure interconnection and the oscillators' natural frequencies. The following technical assumption (A1) is considered:

Assumption 1 (A1)**.** *For the partition $P = \{P_1, \dots, P_M\}$ of ν, there exists clusters' ordering P_i and the time interval $[t_1, t_2]$, with $t_2 > t_1$, such that for all times $t \in [t_1, t_2]$:*

$$\max_{i \in P_1} \dot{\theta}_i > \max_{i \in P_2} \dot{\theta}_i > \cdots > \max_{i \in P_M} \dot{\theta}_i$$

Assumption (A1) needs the phases of the oscillators in separate clusters to evolve with separate frequencies in some time interval. This assumption is not restrictive, as this is commonly the case when the networks' oscillators belonging to separate clusters have different natural frequencies.

As the study follows if it is possible to synchronize pattern in each partition P set, here now the conditions of cluster synchronization are described in Theorem 1 from [Tib+17].

Theorem 1 (**Cluster synchronization**)**.** *For the oscillators' network $O = (\mathcal{G}, \omega, A)$ where $\mathcal{G} = (\nu, \epsilon)$, $\nu = \{1, \dots, N\}$ and $\epsilon \subseteq \nu \times \nu$, the partition $P = \{P_1, \dots, P_M\}$ of ν is phase synchronizable if and only if both the conditions mentioned below satisfy together:*

(a) The network weights satisfy $\sum_{k \in P_z} a_{ik} - a_{jk} = 0$ for all $i, j \in P_l$ and for all $z, l \in \{1, \dots, M\}$, with $z \neq l$;

(b) The natural frequencies satisfy $\omega_i = \omega_j$ for all $k \in \{1, \dots, M\}$ and for all $i, j \in P_k$.

Proof. (If) Let $\theta_i = \theta_j$ for all $i, j \in P_k$, with all $k \in \{1, \dots, M\}$.

Let $i, j \in P_l$ and observe that:

$$\begin{aligned} \dot{\theta}_i - \dot{\theta}_j &= \sum_{z \neq l} \sum_{k \in P_z} [a_{ik} \sin(\theta_k - \theta_i) - a_{jk} \sin(\theta_k - \theta_j)] \\ &= \sum_{z \neq l} s_{zl} \sum_{k \in P_z} [a_{ik} - a_{jk}] = 0 \end{aligned} \tag{3.7}$$

here both the conditions (a) and (b) are considered, and where z and l are clusters and $s_{zl} = \sin(\theta_z - \theta_l)$ is dependent on z and l clusters but not dependent on i, j and k.

Hence, when both the conditions (a) and (b) satisfies, then $\theta \in Im(V_P)$ signifies $\dot{\theta} \in Im(V_P)$, the image of V_P. $Im(V_P)$ is uniform and the network of oscillators is phase synchronizable. Image of matrix is defined in Section 2.2 of Chapter 2.

(Only if) Firstly it is shown that condition (a) is necessary in the network of oscillators for phase synchronization. Let $i, j \in P_l$ and the network is phase synchronized. For all times, it must hold:

$$\begin{aligned} 0 = \ddot{\theta}_i - \ddot{\theta}_j &= \sum_{z \neq l} \sum_{k \in P_z} a_{ik} \cos(\theta_k - \theta_i)(\dot{\theta}_k - \dot{\theta}_i) - \sum_{z \neq l} \sum_{k \in P_z} a_{jk} \cos(\theta_k - \theta_j)(\dot{\theta}_k - \dot{\theta}_i) \\ &= \sum_{z \neq l} c_{zl} u_{zl} \underbrace{\sum_{k \in P_z} [a_{ik} - a_{jk}]}_{d_z} \end{aligned} \tag{3.8}$$

where $u_{zl} = (\dot{\theta}_k - \dot{\theta}_i)$ and $c_{zl} = \cos(\theta_k - \theta_i)$ are dependent on z and l clusters but not dependent on i, j and k.

From (A1), after the clusters' reordering, in some significant time interval, we have:

$$\max_{i \in P_1} \dot{\theta}_i > \max_{i \in P_2} \dot{\theta}_i > \cdots > \max_{i \in P_M} \dot{\theta}_i$$

Thus, (3.8) implies that, either $d_z = 0$ for all z (therefore indicating condition (a)), or the functions $c_{zl}u_{zl}$ must be linearly dependent at all times in the time interval. So let us assume that the functions $c_{zl}u_{zl}$ are linearly dependent at all times in the above time interval. Then it must hold:

$$\sum_{z \neq l} d_z \frac{\mathrm{d}^n}{\mathrm{d}t^n} c_{zl} u_{zl} = 0,$$

where $\frac{\mathrm{d}^n}{\mathrm{d}t^n}$ denotes n-times differentiation and n is positive integer. Therefore, not only the functions $c_{zl}u_{zl}$ must be linearly dependent at some times in the above time interval, but also all their derivatives must be linearly dependent at some times in the above time interval.

Let $d_1 \neq 0$ (if there is $d_1 = 0$, then let us simply select the first coefficient which is nonzero), and $i, j \notin P_1$. An integer n exists such that:

$$\sum_{z \neq l} d_1 \frac{\mathrm{d}^n}{\mathrm{d}t^n} c_{1l} u_{1l} \gg \sum_{z \neq l} d_z \frac{\mathrm{d}^n}{\mathrm{d}t^n} c_{zl} u_{zl},$$

for all $z \neq 1$, because of assumption (A1). Therefore, the functions $c_{zl}u_{zl}$ cannot be linearly dependent at all times in the time interval. It can be concluded that statement (a) is important for phase synchronization in the network of oscillators.

Now, it is proved that statement (a) implies statement (b) when the network is phase synchronized. This indicates that statement (b) is necessary for phase synchronization.

Let network of Kuramoto oscillators $O = (\mathcal{G}, \omega, A)$ be phase synchronized, and $i, j \in P_l$. We have:

$$0 = \dot{\theta}_i - \dot{\theta}_j = (\omega_i - \omega_j) + \sum_{z \neq l} s_{zl} \underbrace{\sum_{k \in P_z} [a_{ik} - a_{jk}]}_{=0}$$

where $s_{zl} = \sin(\theta_z - \theta_l)$ is not dependent on i, j and k as mentioned above but dependent on clusters z and l, and where we have used that statement (a) is necessary for phase synchronization. For the conclusion, $\omega_i = \omega_j$ at all times, and statement (b) is also necessary for the network of the oscillators to be phase synchronized.

The necessity of assumption A1: Consider a network of Kuramoto oscillators $O = (\mathcal{G}, \omega, A)$ with adjacency matrix A:

$$A = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ a_{21} & 0 & a_{23} & 0 \\ 0 & a_{32} & 0 & a_{34} \\ 0 & 0 & a_{43} & 0 \end{bmatrix}$$

and the natural frequencies be $\omega_i = \bar{\omega}$ at all times and all $i \in \{1, 2, 3, 4\}$. Consider that condition (a) in Theorem 1 is not satisfied. Then, let $\theta_1(0) = \theta_2(0)$ and $\theta_3(0) = \theta_4(0) = \theta_1(0) + \pi$, and observe that $\dot{\theta}_i = \bar{\omega}$ at all times and for all $i \in \{1, 2, 3, 4\}$, here assumption A1 is not satisfying as all the values of $\dot{\theta}_i$ are equal for all the times i.e. $\bar{\omega}$ and there will be no maximum value among the partition sets. The partition $P = \{P_1, P_2\}$ of ν with partition sets $P_1 = \{1, 2\}$ and $P_2 = \{3, 4\}$ is phase synchronized, independent of the interconnection weights between the oscillators. Therefore, this implies that in Theorem 1 the condition (a) is not necessary when Assumption A1 is not satisfied.

□

Now, let $A \circ B$ denotes the Hadamard product between A and B, and $Im(V_P)^{\perp}$ the orthogonal subspace to $Im(V_P)$.

Corollary 1.1 (**Matrix condition for synchronization**). *Condition (a) in Theorem 1 is equivalent to $\bar{V_P}^T \bar{A} V_P = 0$, where $\bar{V_P} \in \mathbb{R}^{N \times (N-M)}$ satisfies $Im(\bar{V_P}) = Im(V_P)^{\perp}$, and*

$$\bar{A} = A - A \circ V_P V_P^T \tag{3.9}$$

Proof. Let $\bar{A} = [\bar{a}_{ij}]$ and $A = [a_{ij}]$. Observe that if i and j belongs to the same cluster then $\bar{a}_{ij} = 0$ and if i and j belongs to different clusters then $\bar{a}_{ij} = a_{ij}$. Therefore,

$$\left[\bar{A}V_P\right]_{ij} = \begin{cases} \sum_{k\in P_j} a_{ik}, & \text{if } i \notin P_j \\ 0, & \text{if } i \in P_j \end{cases}$$

Take $\bar{V}_P$ so that $\bar{V}_P = [\bar{v}_1....\bar{v}_{N-M}]$ and $\bar{v}_i^T x = x_r - x_s$, with a vector x of compatible dimension for $r, s \in P_l$. Then,

$$\left[\bar{V}_P^T \bar{A} V_P\right]_{ij} = \begin{cases} \sum_{k\in P_j} a_{rk} - a_{sk}, & r, s \notin P_j \\ 0, & r, s \in P_j \end{cases}$$

where s and r are the nonzero elements of $\bar{v}_i$.

□

3.5 Control of Cluster synchronization

A control mechanism of cluster synchronization modifies the weights of the network of oscillators to ensure the formation of the desired synchronization pattern. The control method is optimal as it finds the smallest network perturbation for a given synchronization pattern in the network and ensures the transformation of only a desired subset of the network's edge weights. Here, the Frobenius norm is used to measure the smallest network perturbation for a given synchronization pattern in the network.

The conditions on the network of oscillators are derived in section 3.4.1 to guarantee phase and frequency synchronization. These conditions are slightly rigid and are generally not satisfied for interconnection weights and arbitrary partitions.

Now, we will set up a control problem and develop a control mechanism to modify the interconnection edge weights of oscillators to ensure the synchronization of a given partition in the network of oscillators. Specifically, the following minimization problem is studied:

$$\min_{\Delta} \qquad \|\Delta\|_F^2 \tag{3.10}$$

$$s.t. \qquad \bar{V}_P^T \left[\bar{A} + \Delta\right] V_P = 0 \tag{3.11}$$

$$\Delta \in \mathcal{H} \tag{3.12}$$

where,

- $\|\Delta\|_F$ is the Frobenius norm of the matrix Δ.

- The characteristic matrix of partition P is $V_P \in \mathbb{R}^{N\times M}$ where,

$$V_P = [v_1\ v_2\ \dots v_M]$$

and

$$v_{ij}^T = [\underbrace{0\ 0\ \dots 0}_{\sum_{j=1}^{i-1}|P_j|}\ \underbrace{1\ 1\ \dots 1}_{|P_i|}\ \underbrace{0\ 0\ \dots 0}_{\sum_{j=i+1}^{n}|P_j|}]$$

here, $|P_j|$ denotes the cardinality of the set P_j.

- Now, $\bar{V}_P \in \mathbb{R}^{N\times(N-M)}$, ($A \circ B$ denotes the Hadamard product between A and B defined in Section 2.2 of Chapter 2), and for a given weighted adjacency matrix A, we can define the following operation:

$$\bar{A} = A - A \circ V_P V_P^T$$

We denote $A = [a_{ij}]$ and $\bar{A} = [\bar{a}_{ij}]$ so that if i and j belongs to the same cluster then $\bar{a}_{ij} = 0$ and if i and j belongs to different clusters then $\bar{a}_{ij} = a_{ij}$, as mentioned in Corollary 1.1.

- $\mathcal{H}$ encodes a desired sparsity pattern of the perturbation matrix Δ. We can say that $\mathcal{H}$ may represent the set of matrices compatible with the graph $\mathcal{G} = (\nu, \epsilon)$, that is, $\mathcal{H} = \{M : M \in \mathbb{R}^{|\nu|\times|\nu|}\ and\ m_{ij} = 0\ if\ \ (i,j) \notin \epsilon\}$.

To solve the minimization problem (3.10), we define the minimization problem mentioned below by including the sparsity constraints (3.12) into the cost function, here $\oslash$ denotes element-wise division,

$$\min_{\Delta} \qquad \|\Delta \oslash H\|_F^2 \tag{3.13}$$

$$s.t. \qquad \bar{V}_P^T \left[\bar{A} + \Delta\right] V_P = 0 \tag{3.14}$$

where H satisfies $h_{ij} = 1$ if there exists a matrix $M \in \mathcal{H}$ such that $m_{ij} \neq 0$, and $h_{ij} = 0$ otherwise. The minimization problems (3.10) and (3.13) are equivalent, in the sense that Δ^* is a (possible) solution to (3.10) if and only if it has finite cost in (3.13).

Theorem 2 (**Synchronization via structured perturbation**). *Consider* $T = \left[V_P \bar{V}_P\right]$, *and let*

$$\begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix} = T^{-1}\bar{A}T$$

The minimization problem (3.10) *has a solution if and only if there is a* Λ *matrix satisfying:*

$$X = \bar{V}_P \Lambda V_P^T \circ H,$$

and

$$\tilde{A}_{21} = \bar{V}_P^T X V_P$$

Furthermore, if there exists, a solution Δ^* *to* (3.10) *is:*

$$\Delta^* = \begin{bmatrix} \tilde{\Delta}_{11}^* & \tilde{\Delta}_{12}^* \\ \tilde{\Delta}_{21}^* & \tilde{\Delta}_{22}^* \end{bmatrix} T^{-1}$$

where, $\tilde{\Delta}_{11}^* = -V_P^T X V_P, \tilde{\Delta}_{12}^* = -V_P^T X \bar{V}_P, \tilde{\Delta}_{21}^* = -\tilde{A}_{21}$ *and* $\tilde{\Delta}_{22}^* = -\bar{V}_P^T X \bar{V}_P$.

Corollary 2.1 (**Unconstrained minimization problem**). *Let* $\mathcal{H} = \{M : m_{ij} \neq 0\}$ *for all* i *and* j *. The minimization problem* (3.10) *is always possible, and its solution is:*

$$\Delta^* = -\bar{V}_P \bar{V}_P^T \bar{A} V_P V_P^T$$

In our thesis, we set up this control problem used to achieve the desired synchronization in networks of Kuramoto oscillators and apply Genetic Programming (GP used for MLC) as control mechanism for synchronization of Kuramoto oscillators.

4 Machine Learning Control (MLC)

Machine Learning Control is a data-driven regression strategy that uses evolutionary programming (EP), genetic algorithms (GA), genetic programming (GP), and other techniques (decision trees, support vector machines (SVM), and neural networks) to discover effective control laws.

In this Chapter, Machine learning is mentioned as a tool for designing control laws. This machine learning control (MLC) offers a robust new approach for controlling the complex dynamical systems that can be controlled more effectively than any other currently existing methods in control. Further in this chapter, the use of GP as a search algorithm to find control laws in MLC as shown in Figure 4.1 is discussed.

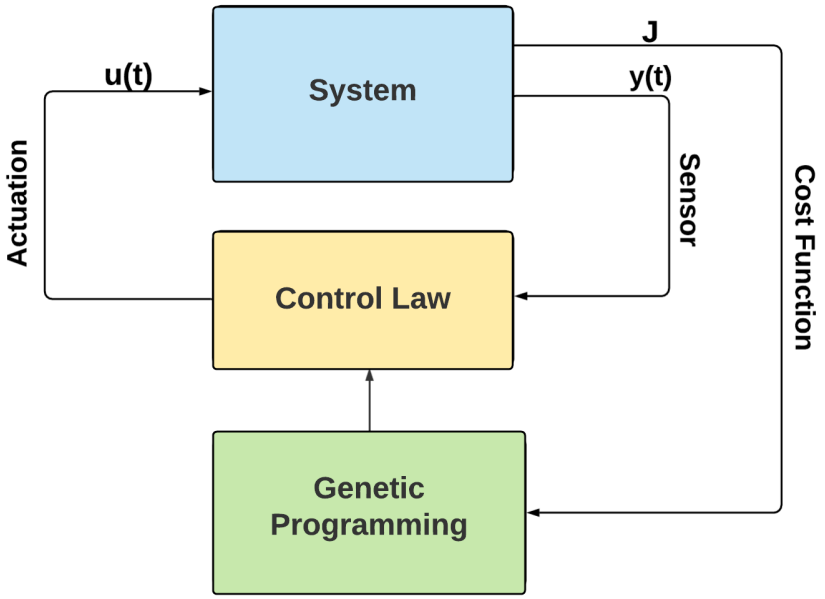

Figure 4.1: Genetic Programming (GP) in MLC

Firstly, feedback control is described in Section 4.1 as MLC uses feedback control. In Section 4.2, we highlight concepts from machine learning with a focus on evolutionary algorithms. Section 4.3 describes the method of machine learning control (MLC) in detail. In Section 4.4, we analyse the use of genetic programming as an effective method to discover control laws. Finally, in Section 4.5, we provided implementation and analysed illustrative example to better understand these concepts.

4.1 Feedback Control

Feedback appears when the output affects the input within the same system. Feedback control is a method of making such a loop, which modifies the behaviour of a dynamical system based on measurements made during the loop.

Figure 4.2 shows a feedback control system. The blue box shows the physical system (also known as the plant). The system is modified by actuators (inputs,u) through a control law shown in the yellow box informed by sensor measurements (outputs, y) of the system. The control logic is developed to create a closed-loop response to disturbances w in relation to a high-level goal cost function J. The control should also be optimized in terms of a cost function J.

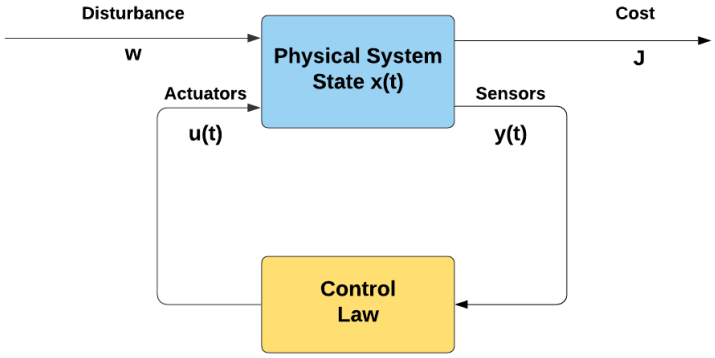

Figure 4.2: Framework for general optimization of feedback control

4.1.1 Benefits of Feedback Control

There are various examples of optimization tasks - like noise reduction, drag reduction, mixing increase, and lift increase. An airplane needs to maintain a well-defined level of thrust and lift to keep it on the desired trajectory. In this context, the control task is seen as a reference tracking problem, in which a reference force or other quantity is controlled. When this happens, the actuation level is penalized by the cost function along with the deviation from the desired state.

It is natural to first think of the open-loop control architecture as shown in Figure 4.3 for the case of the reference tracking problem. In this architecture, the actuation signal u is selected based on knowledge of the system to deliver the desired output that fits the commanded reference signal. In the same way, toasters operate by turning on the heating element for a specified amount of time, depending on the desired temperature.

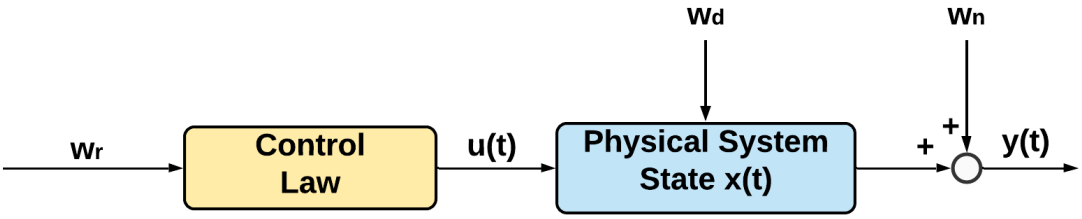

Figure 4.3: Open-loop Control Architecture

Figure 4.3 shows that Open-loop control is based on a reference signal w_r that specifies an actuation signal $u(t)$. As a result, a system's performance is degraded by external disturbances w_d, sensor noise w_n, and unmodeled dynamics and uncertainty of the system $x(t)$.

Open-loop control, however, is unable to stabilize an unstable system, like an inverted pendulum, because the plant model should be known perfectly, without disturbances or uncertainty. Open-loop control cannot compensate for system disturbances with adjustments to the actuation signal.

By feeding back sensor measurements of system output, it is possible to close the loop instead of making control decisions only based on the desired reference, as in open-loop control. In this case, the controller determines whether the target has been achieved. The closed-loop control architecture is depicted in Figure 4.4. Using sensors for feedback can solve the problems associated with open-loop control. For example, a sensor-feedback system can usually stabilize an unstable system, but an open-loop system cannot. Moreover, closed-loop control compensates for model uncertainties and external disturbances, measured in the sensor output.

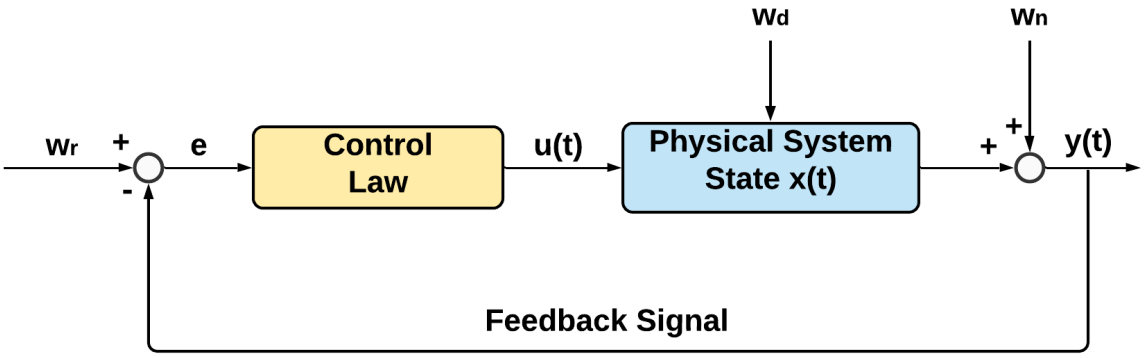

Figure 4.4: Closed-loop Control Architecture

The sensor signal $y(t)$ is fed back to the reference signal w_r and is subtracted from the reference signal w_r. Finally, the controller utilizes the resulting error e to determine the actuation signal $u(t)$. Generally, feedback stabilizes unstable system $x(t)$ dynamics while effectively attenuating sensor noise w_n and rejecting external disturbances w_d.

To summarize, feedback control is important for the tasks like:

- Optimize output or state according to the given cost function;
- Stabilizing an unstable system;
- External disturbances and model uncertainties both are measured in the sensor output, feedback control is able to compensate for both;
- Attenuating sensor noise w_n and rejecting external disturbances w_d.

4.1.2 Mathematical Formulation

There is an influential theory of feedback control depending on dynamical systems. Here, in this framework, the system is modeled by an input-output system:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = F(x(t), u(t), w_d) \tag{4.1}$$

$$y(t) = G(x(t), u(t), w_n) \tag{4.2}$$

having coupled system of possibly nonlinear differential equations in a *state variable* $x(t) \in \mathbb{R}^{N_x}$, N_x is the state's dimension. The *actuation input* $u(t) \in \mathbb{R}^{N_u}$ directly affects the state dynamics in (4.1), with disturbances w_d. The *sensor measurements output* $y(t) \in \mathbb{R}^{N_y}$ may be nonlinear functions of the state $x(t)$, the control $u(t)$ and noise w_n.

The control task given is to construct a controller:

$$u(t) = K(y(t), w_r) \tag{4.3}$$

so that the closed-loop system includes desirable properties in terms of stability, rejection of disturbances, attenuation of noise, and good reference tracking characteristics. In (4.3), w_r is commanded reference signal. All these above mentioned factors are encoded in the cost function J which is a function of the actuation input, the sensor output, and external signals w_d, w_r, and w_n.

By properly designing a sensor-based feedback control law, it is usually possible to get a closed-loop system that performs optimally for the selected cost function and is powerful to model external disturbances, uncertainty, and sensor noise. In fact, most current problems of control theory are posed in terms of optimization through cost minimization. The perspective of the book [DBN16] is that machine learning gives a powerful and efficient set of techniques to get high-performance control laws even for incredibly complex systems with non-convex[1] optimization cost functions.

[1]For any two points on the curve, there is no intersection with any other points for a convex function; but for a non-convex function, there is at least one intersection. The cost function with a convex type is always guaranteed to produce a global minimum, while the cost function with a non-convex only has local minima.

4.1.3 Challenges of Feedback Control

Feedback control problems are - System dynamics can be massively nonlinear, so sometimes it will be not possible to easily linearize the system to apply control theory on it; System state can be so high dimensional that it results in complex control tasks; If the system is high dimensional then there will be limited sensor measurements; Unknown dynamics. Therefore, feedback turbulence control is a significant challenge problem.

Machine learning techniques are utilised to solve these problems as they are helpful in such scenarios, which are discussed in the below section 4.2.

4.2 Machine Learning

As we have seen challenges of feedback control, we see another way of control design: learning by trial and error. It is not possible to predict control policy effects in a system. However, it can be comparatively easy to test the control policy effect in a system. Then it is possible to evolve the control policy by testing, exploring alternative control policies and exploiting good control policies. Evolutionary strategies in design problems of fluid mechanics follow these principles, these Evolutionary strategies have been pioneered by Rechenberg [Rec73] and Schwefel [Sch65].

Biologically inspired optimization methods have become increasingly powerful in the past five decades. Fleming and Purshouse [FP02] summarize: 'The evolutionary computing (EC) area holds its origins in four evolutionary approaches: Genetic Algorithms (GA), Genetic Programming (GP), Evolution Strategies (ES), and Evolutionary Programming (EP).'

EP, GA and GP are considered as regression techniques to find the control laws (input-output maps) that minimize the cost function. Control design can also be seen as a regression task - finding input-output maps that reduce the cost function. EC is increasingly used for complex control tasks. For example - [Wah08] provides the study that EP is used for programming robot problems, GA is utilized to optimize parameters of linear control laws as described in the analysis of [Ben+15; Dra97]. And [DK97] shows that for two decades, GP has been utilized to discover optimal nonlinear control laws. GP is the most powerful regression technique among various regression techniques as it prompts analytical control laws of practically arbitrary form. Every single evolutionary technique is a part of the quickly developing field of machine learning. [Wu+08] gives knowledge that there are also other machine learning techniques to find control laws, like decision support vector machines (SVM), neural networks, decision trees and various others. It has actually been shown that the first feedback turbulence control method using machine learning is supported by a neural network in [Lee+97]. In [DBN16], machine learning control is referred to as an approach using any of the previously mentioned data-driven regression techniques to find effective control laws.

4.3 Machine Learning Control (MLC)

Machine learning control (MLC) gives us a powerful new framework to control complex dynamical systems that are as of now beyond the capacity of existing techniques in control theory. Machine learning use data to generate models of the system; then these models improve with more data. We use Machine learning algorithms around a complex system to know an effective control law:

$$u = K(y)$$

that maps the output (sensors, $y(t)$) to the input (actuators, $u(t)$) of the system. The resulting machine learning control (MLC) is motivated by problems including complex control tasks where it might be impossible to model the system and discover useful control law. As an alternative, we directly use experience and data to gain knowledge of effective control laws.

In the classical framework, machine learning has been used to model the input-output feature of a system. Then based on these models controllers are designed by using traditional methods. Machine learning control avoids this process and directly gain knowledge of effective control laws without needing a model of the system.

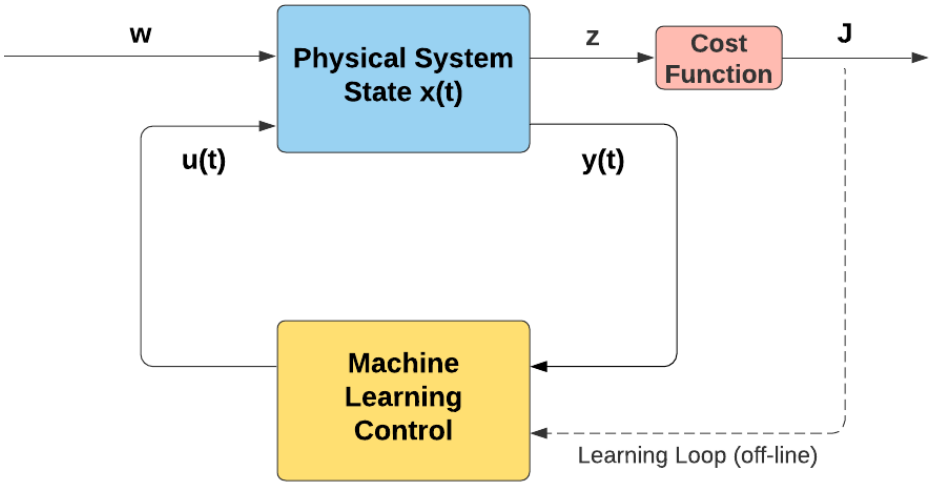

Figure 4.5: Machine Learning Control Architecture

Figure 4.5 shows machine learning control architecture. A well-defined control task is formulated to minimise a cost function J value that is evaluated based on the measured outputs z of the system. Next, the controller has an adaptable and general representation so that a search algorithm is established in the space of possible control laws. Lastly, a machine learning algorithm is selected to find the most appropriate control law by training procedures involving experiments. The offline learning loop gives experimental data to train the controller. Mainly, Genetic programming provides flexible algorithms to search for effective control laws. The vector z has complete information that might factor into the cost function value.

4.3.1 Methods of Machine Learning used for MLC

Machine learning is a rapidly advancing domain at the intersection of computer science, statistics, natural sciences, and applied mathematics. Machine learning is driven mainly by technological advances in marketing and technology and the availability of large amounts of data in nearly every field. Nevertheless, these techniques are becoming increasingly prevalent in many academic and industrial areas. They already provide insight into finance, astronomy, climate, ecology, etc. As a relatively new frontier in data-driven engineering, the application of machine learning to design feedback control laws has tremendous potential.

In this section, firstly, similarities between machine learning and classical techniques from system identification (SI) are discussed. These methods are already prominent in control design and give context for machine learning control. Evolutionary algorithms form a significant classification of machine learning methods that adjust and upgrade through a procedure impersonating natural selection. So later in this section, we will discuss the evolutionary approaches of GA and GP. Genetic programming (GP) is particularly promising for machine learning control because it can optimize both the structure and parameters of a controller. Finally, an overview of other advantageous methods from machine learning that benefit MLC efforts is briefed.

4.3.1.1 System identification as machine learning

Classical system identification is viewed as an earlier form of machine learning, where a dynamical system is represented via training data. The resulting models approximate the existing system's input-output dynamics and design controllers with the techniques defined in [DBN16, Chapter 3]. Unfortunately, most methods in system identification are developed for linear systems and provide models of questionable quality for systems with highly nonlinear dynamics. However, in [BG02; SA91], it is mentioned that there are extensions to linear parameter varying (LPV) systems, where the linear system is dependent on the time-varying parameter.

[Jua94] is a vast literature on SI with numerous methods developed to describe aerospace systems from the 1960s to the 1980s. The Observer Kalman filter identification (OKID) and Eigensystem realization algorithm (ERA) methods produce input-output models utilizing time-series data from dynamical systems; these methods are discussed better in [DBN16, Chapter 3]. On the other hand, the Singular Spectrum Analysis (SSA) delivers a similar description of a time series without developing input-output models, which is described in [BJ14]. In [237, 228, 269], we analyzed that the Dynamic Mode Decomposition (DMD) is a promising new approach for system identification that maintains powerful connections to nonlinear dynamical systems via Koopman spectral analysis and [LM13] provides the detailed study of Koopman spectral analysis. DMD has just been expanded to include control and sparse measurements which is described in detail in [PBK16] and [BPK13]. In addition, the DMD method has been applied to multiple other problems beyond fluid dynamics, where it was invented, including video processing, epidemiology, neuroscience, and robotics. Other recognized methods comprise the Auto-Regressive Moving Average models (ARMA) and their extensions. Reducing the amount of data needed for the training and implementation of a model is usually necessary when a quick decision or prediction is needed, like in turbulence

control. Machine learning and Compressed sensing have already been integrated to acquire sparse decision making, which dramatically decreases the delay in a control decision.

This paper [MAS05] focuses on the data-driven system identification of dynamical systems' nonlinear input-output mapping models. The data-driven system identification of these models uses the following tasks:

1. **Structure selection:** How to select the model structure and the order of the nonlinear static functions utilised to describe the model.
2. **Input sequence design:** Defining the input sequence, which is given into the modelled object, to develop the output sequence that is utilised for the system identification.
3. **Parameter estimation:** Analysing the model parameters from the input-output sequence.
4. **Noise modelling:** Determining the dynamic model which causes the noise.
5. **Model validation:** Comparing the modelled object's output and the model dependent on data not utilised in model development.

The assumption in most data-driven system identification algorithms is that - the structure of the model is already known, or it is determined with the help of a structure-selection algorithm. Various information-theoretic measures have been proposed for the selection of structure of linear dynamic input-output models. These classical measures are the Akaike Information Criterion (AIC) and Final Prediction-Error (FPE).

Figure 4.6 shows the process of system identification, where we see that after the system identification as machine learning applied to the model of control system and then after the analysis the controller is designed.

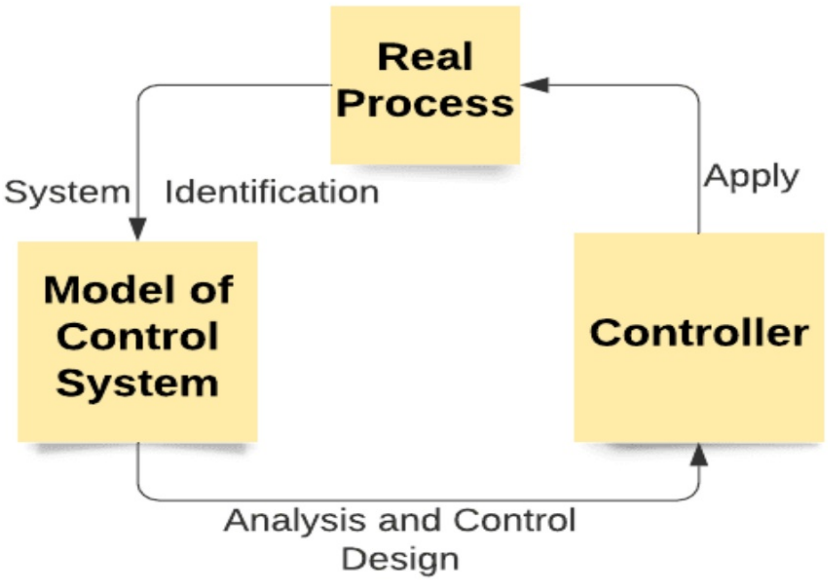

Figure 4.6: Process of System identification as machine learning

As a traditional approach, machine learning is applied to model the input-output characteristics of a system. Then, these models are used to develop controllers using traditional controller design techniques. However, effective control laws are directly learned instead of producing a system model in machine learning control. This we can see and understand by Figure 4.7, where effective control law is directly obtained without designing controller again and again.

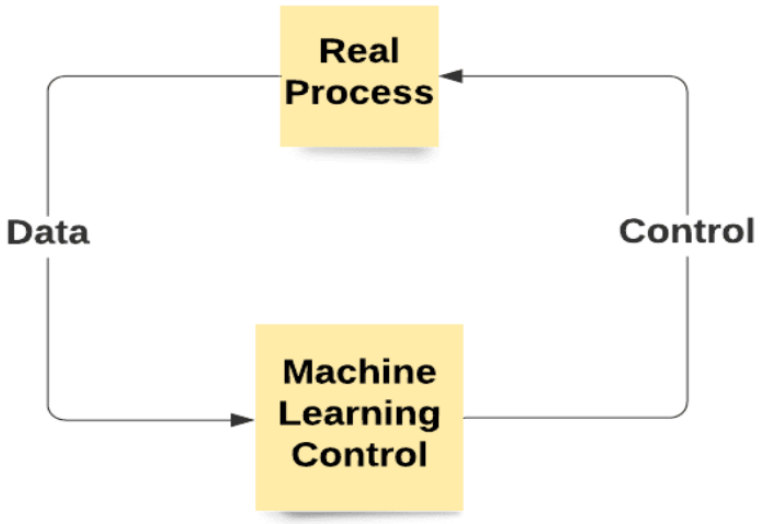

Figure 4.7: Process of Machine Learning Control

4.3.1.2 Genetic Algorithms (GA)

A population of all the individuals together is called a generation. Individuals compete at a given control task with a well-defined cost function, and there is a set of rules to propagate successful individuals to the next generation. Genetic algorithms (GA) are used to optimize and identify the parameters of an input-output map. Genetic algorithms are based on the propagation of generations of individuals through selection based on fitness. The individuals that contain a generation are first populated randomly, then every individual is evaluated, and fitness is assigned based on their performance on the evaluated cost function. For example, individuals having lower cost function J have higher fitness f and are more likely to be selected to advance.

Fitness can be given as:

$$Selection - Rate = fitness(f) = \frac{1}{J}$$

If $J = 0$, then the individual will have the highest fitness among all. An individual in a genetic algorithm relates to a set of parameter values in a parameterized model that has to be

optimized. There are a set of rules, or genetic operations, to advance successful individuals from one generation j to the next generation $j+1$.

There are four genetic operations which are as follows:

1. ***Elitism:*** The most fit individuals advance directly to the next generation.
2. ***Replication:*** Individuals propagate directly to the next generation with a probability based on fitness.
3. ***Crossover:*** Two individuals are selected according to their fitness and random parts of their parameters are exchanged. These two individuals advance to the next generation with the exchanged information.
4. ***Mutation:*** Individuals propagate with random parts of their parameter replaced with random new values.

Crossover exploits successful structures and optimises locally, while mutation is used to explore the search space and find global minima. These above mentioned four genetic operations advance successful individuals from one generation to the next generation. New individuals are added in every generation for variety. This is shown in Figure 4.8. Generations are moved forward until the performance converges to the desired stopping criterion. Regarding control, genetic algorithms are used to tune the parameters of a control law having a predefined structure.

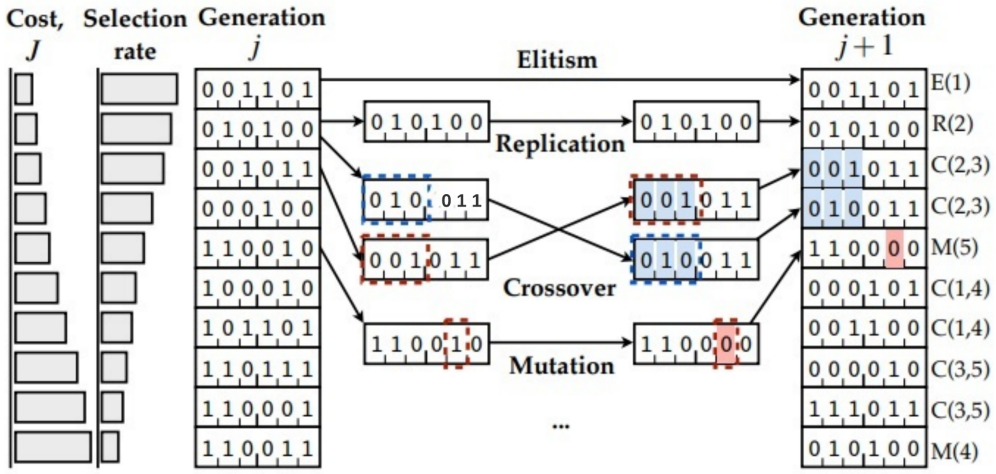

Figure 4.8: Genetic operations that are used in genetic algorithm to advance parameters of one generation to the next generation. Modified from Brunton and Noack [DBN16, Chapter 2]

However, evolutionary algorithms have been successful in diverse applications even when they may not converge to global minima. Several parameters can be tuned in evolutionary algorithms to improve their performance, such as the number of individuals in a generation, the number of generations, and the relative probability of every genetic operation.

Using genetic algorithms, parameters of an existing control law can be tuned to fit an existing structure. For instance, GA might be used to tune the gains of a proportional-integral-derivative (PID) control law.

4.3.1.3 Genetic Programming (GP)

Genetic Programming (GP) is an evolutionary algorithm that optimizes the structure as well as parameters of input-output mapping. GP is used to iteratively learn and refine control laws, which are seen as nonlinear mappings from the outputs(sensors, y) to the inputs(actuators, u) in a system to minimize a given cost function value associated with the control task.

The input-output mapping of GP is represented as recursive function tree for the Control Law, as depicted in Figure 4.9:

$$u = K(y)$$

The representation of recursive function tree - The root of the tree is output variable, in the case of MLC root is the actuation signal; The leaves of the tree are input variables and constants, in the case of MLC the inputs are sensor measurements; Each branching point is mathematical operation $(+, -, \div, \times)$; Each branch contain functions.

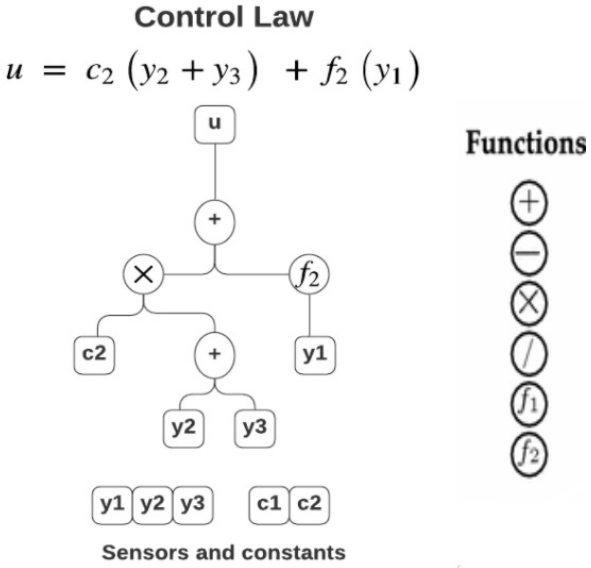

Figure 4.9: Representation of an individual as recursive function tree used in GP.

GP uses same genetic operations which are used in genetic algorithms (Like - Replication, Crossover, and Mutation), to advance individuals across generations - In crossover, random branches of two individuals are exchanged; In mutation, branch is randomly selected and replaced by other randomly generated branch; In replication, the individual is directly copied from one generation to next generation. All these operations are discussed in detail in the coming below Section 4.4.3.4.

Further in this chapter, the use of GP is analysed for closed-loop feedback control.

4.4 MLC with genetic programming (GP)

Now, we will be seeing GP as a search algorithm to find control laws in MLC. GP gives a flexible algorithm to search effective control laws. Steps for using GP in MLC are described in Figure 4.10 and these steps are discussed further.

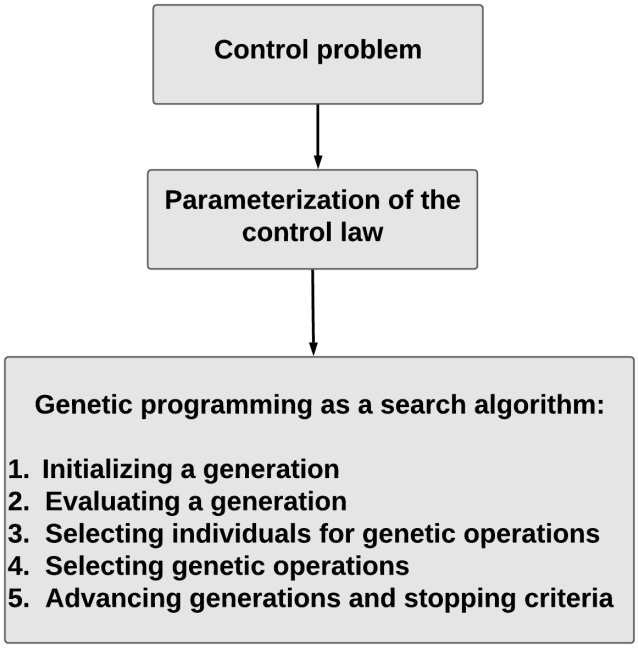

Figure 4.10: Steps for using genetic programming (GP) in MLC

4.4.1 Control problem

It is necessary to present the control problem as a well-defined cost function minimization, before implementing a machine learning technique. The performance of given control law is decided based on the value of a cost function J value, and the machine learning algorithms will aim to minimize this cost function. Let us take simplified cost function:

$$J(x(t), u(t))$$

that depends on the state $x(t)$ and the actuation $u(t)$. The assumption that the state and actuation on the cost are separable:

$$J(x, u) = J_x + \gamma J_u$$

where, J_x is the performance measure on the state of the system. J_u is the value associated with the cost of actuation. $\gamma \in \mathbb{R}$ is the penalization parameter that gives an explicit tuning knob to provide priority to either the state cost (γ small) or the actuation cost (γ large).

More goals are added to the ***cost function*** J by including norms on the various transfer functions from inputs to outputs and the complexity of the controller K is penalized to avoid overfitting.

4.4.2 Parameterization of the control law

The control laws are represented as recursive expression trees (also called function trees) while using GP for MLC. Control laws are the individuals that populate a generation in GP.

- Expression trees are typically built from various of elementary functions that can take any number of arguments yet return a single value; for instance function nodes are $+, -, \div, \times$. The arguments of functions might be sub-trees or leaves.
- In MLC, the root of the tree is the input u (actuation signal) and the leaves are elements of the sensor vector y (outputs) or constants.

There are several ways to represent. We choose to use a tree-like representation. The two main advantages of this representation are - Expression trees are interpretable; Easily synthesized and manipulated computationally using languages such as LISP[2] or Scheme[3].

In Figure 4.11, the tree is representing the controller function u.

$$u = K(y_1, y_2) = \cos(y_1) + \tanh\left(\frac{(y_1 \times y_2)}{0.21}\right) - 2.32$$

Above equation can be written as the LISP string in parenthesized Polish prefix notation -

$$(-(+(\cos y_1)(\tanh(/(\times y_1 y_2)0.21)))2.32)$$

As it is less readable than the expression tree, recursive algorithms can generate, manipulate, and evaluate these expressions. The generation procedure of any individual control law begins at the root. After that, a first component is selected from the pool of acceptable basis operators or functions. If a basis operator or function is selected, new components are added as arguments, and the procedure is iterated to incorporate their arguments until all branches have leaves.

[2]The Lisp language is essentially typeless, but originally consisted of two kinds of data structures: atoms and lists. Lisp stands for "LISt Processing" described in [BB66]

[3]The Scheme programming language is a high-level language that supports both structured and more traditional data. Strings, lists, and vectors can, for example, be used with Scheme as well as numbers and characters. While Scheme is typically associated with symbolic applications, it is a flexible, powerful language with a wide range of data types and control structures, it is described in [Dyb09]

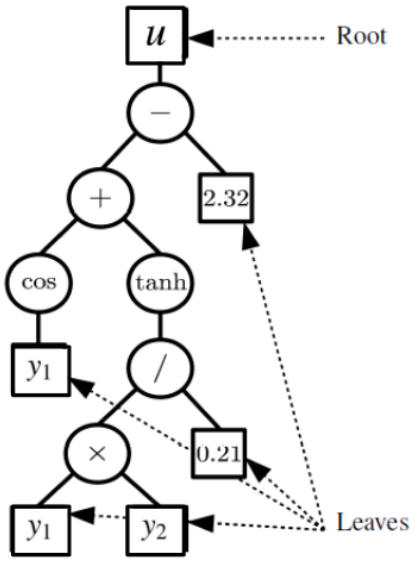

Figure 4.11: Expression Tree representing controller function u

4.4.3 Genetic programming as a search algorithm

A generation of N_i control laws (individuals) is evaluated based on cost function J. Successful individuals are selected to advance to the next generation and evolved by genetic operations like elitism, replication, crossover and mutation. This procedure is repeated until we get convergence or stopping criterion as shown in Figure 4.12 of flowchart for the genetic programming (GP) algorithm.

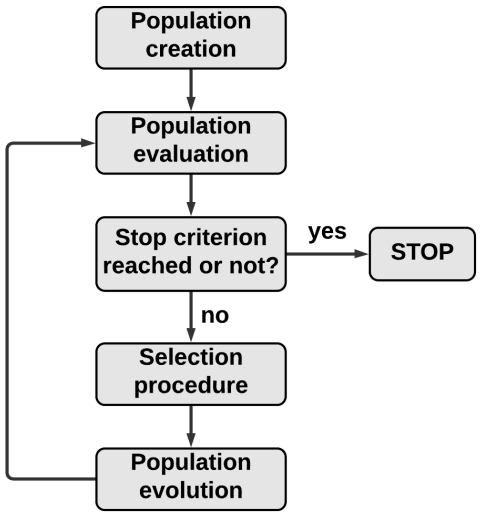

Figure 4.12: Flowchart for the genetic programming (GP) algorithm.

In Figure 4.13, the application of GP as a search algorithm for MLC is shown. In this schematic, control laws are recursive trees that take the sensor outputs y of a system and synthesize the inputs u of actuation. The genetic programming algorithm optimizes these controller individuals.

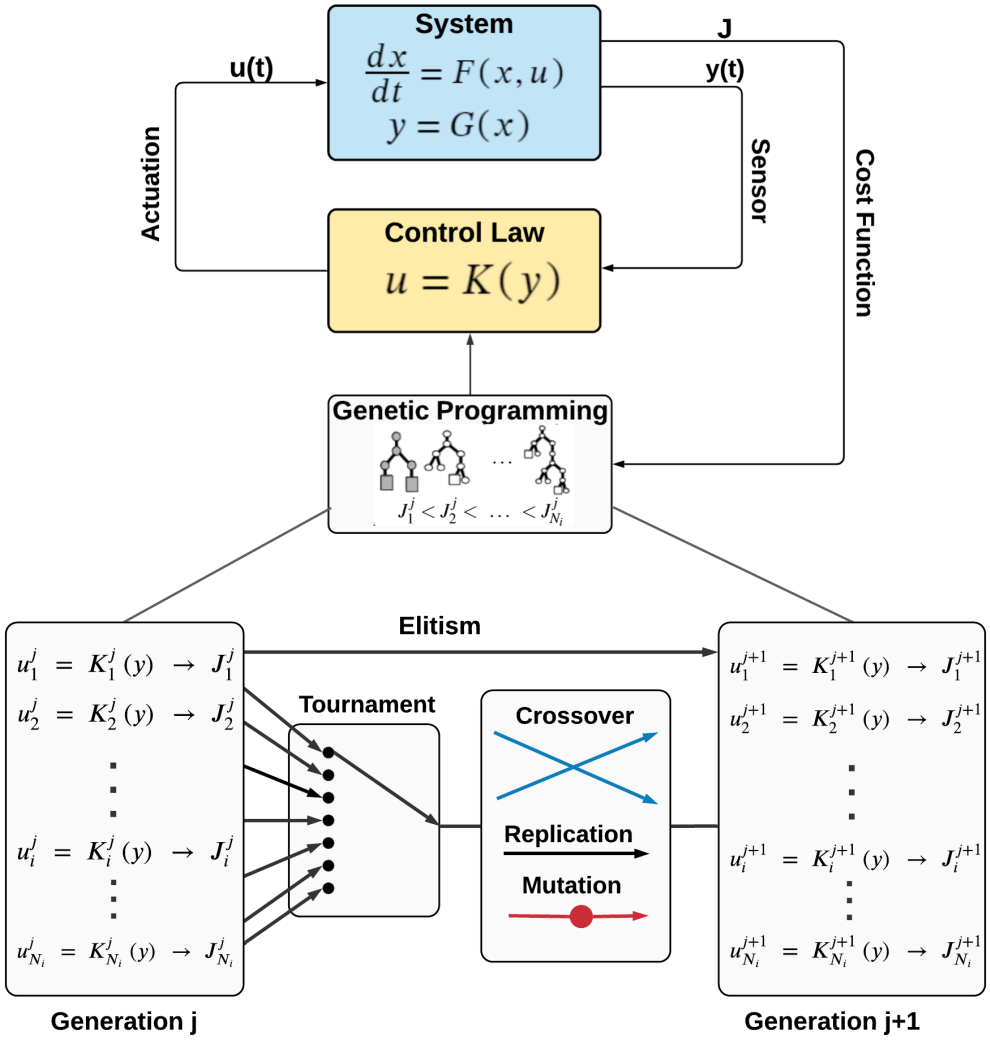

Figure 4.13: Model-free control design using GP for MLC

During the learning stage, every control law individual is evaluated by the system. This procedure is iterated over many generations of individuals. At the time of convergence, the best individual with the minimum cost function value is used for real-time control.

4.4.3.1 Initializing a generation

The population of individuals forms a generation, and these individuals compete to advance to future generations. N_i individuals must be initialized. The initialization algorithm works from the root to the trees to form the expression tree. The principle is to work from one seed or N_u seeds (if the actuation input u has multiple components), decide on a node (function, operation, or leaf), add as many new seeds as necessary (if this is a function or operation) and repetitively call back the function as many time as new seeds have been generated. The process stops when all seeds are replaced by leaves. Those leaves are selected among randomly generated constants and one of the N_y sensors in y.

4.4.3.2 Evaluating a generation

When the new generation is formed, every individual must be evaluated with respect to their performance based on the regression problem. The cost function $J(x(t), u(t))$ is computed for each individual. In MLC, this evaluation is related to the individual that is used as the control law for the dynamical system. All individuals are evaluated again, even if they have already been evaluated in a previous generation. By default in OpenMLC (Matlab toolbox), the five best individuals are re-evaluated five times, and their cost function is averaged. This procedure makes sure that the best performing individuals are more carefully ranked for a good search process.

4.4.3.3 Selecting individuals for genetic operations

The population evolution starts after the evaluation of each individual. Genetic operations (see next part) are performed on selected individuals to fill the next generation.

The process of selection employed is a tournament.

- N_p unique individuals are randomly selected from the past generation to get into the one-round tournament.
- The individual with the smallest cost function is selected.
- N_i (fixed population size) selection tournaments are run every time a new generation created. The population is ranked in order of decreasing cost function value.
- Consider $N_i \gg N_p > 1$, the probability of i individual winning a tournament is -

$$\left(\frac{(N_i - i)}{(N_i - 1)}\right)^{N_p - 1}$$

- Each individual will get into N_p tournaments.
- Each individual is sorted by their fitness, so that individual i has the i-th lowest cost function value; For each individual, we define -

$$x = \frac{i}{N_i}$$

so that $x \in (0, 1]$, where $x = 1$ is the worst individual and $x = 0$ is the best individual.

4.4.3.4 Selecting genetic operations

There are four types genetic operations that are implemented: elitism, replication, mutation and crossover.

1. ***Elitism:*** From the evaluated population, the N_e best individuals are copied directly to the next generation. This operation does not go through a selection process and makes sure that the best control laws remain in the population. Once this operation is finished, $N_i - N_e$ individuals are generated through replication, mutation, and crossover. The probability of replication, mutation, and crossover operations are P_r, P_m and P_c, respectively, with $P_r + P_m + P_c = 1$.
2. ***Replication:*** The selected individual propagates directly to the next generation.
3. ***Mutation:*** Individuals propagate with random parts of their parameter replaced with random new values. Figure 4.14 replication and mutation operation.

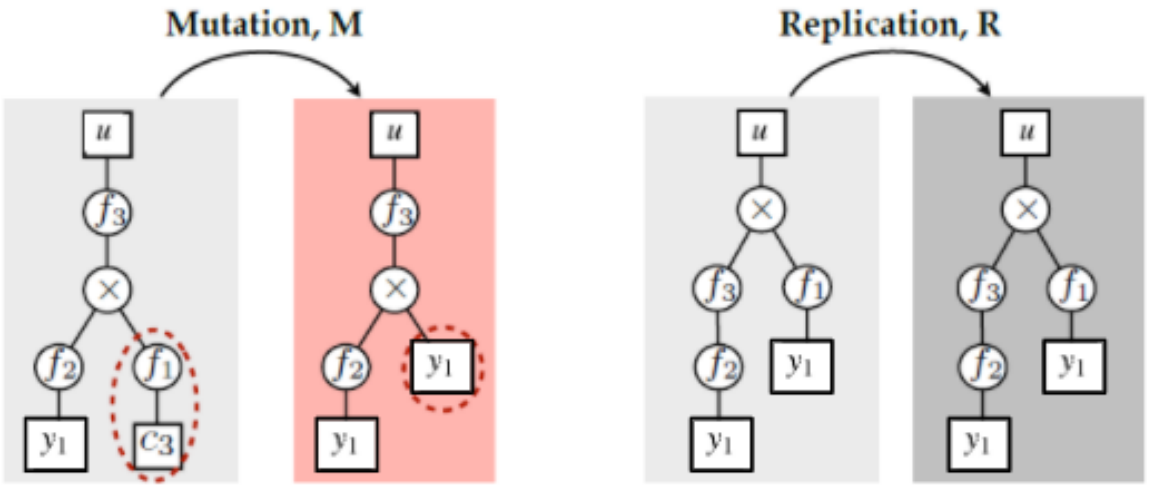

Figure 4.14: Mutation and Replication Genetic Operation

There are four mutation operations as follows:

a) **Cut and grow:** replaces a chosen sub-tree by a randomly generated new sub-tree as shown in Figure 4.15.

b) **Shrink:** replaces an entire randomly selected sub-tree by a randomly selected leaf (constant or sensor) as visible in Figure 4.16.

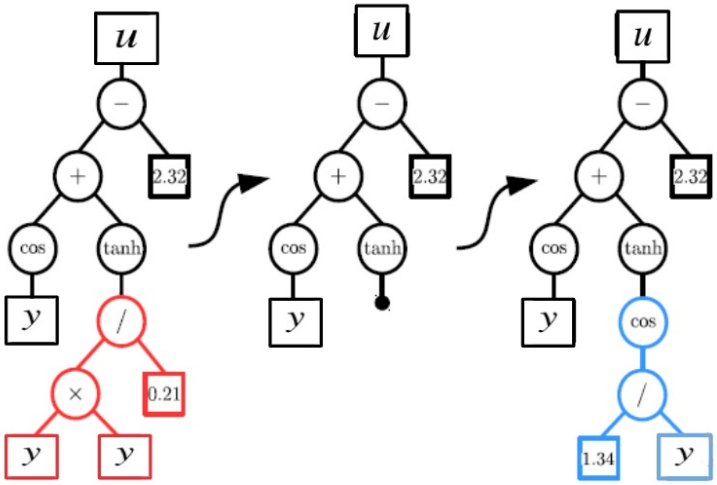

Figure 4.15: Cut and grow Mutation Operation

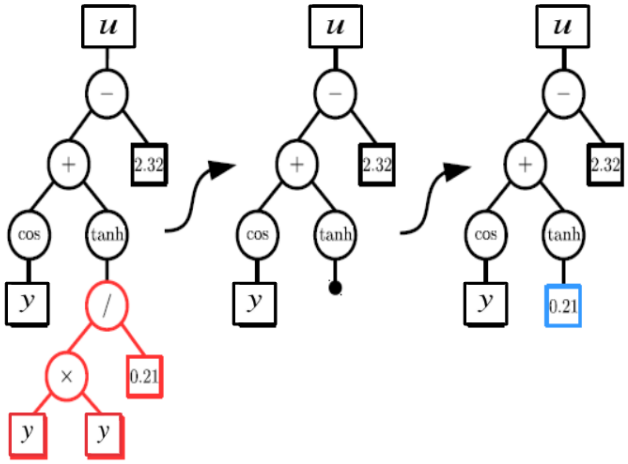

Figure 4.16: Shrink Mutation Operation

c) **Hoist:** replaces the tree by a randomly selected sub-tree as shown in Figure 4.17.

d) **Reparameterization:** sets a chance for each constant to be randomly replaced with a new value as shown in Figure 4.18.

4. ***Crossover:*** Two individuals are selected and random parts of their parameters are exchanged. These two individuals advance to the next generation with the exchanged information as shown in Figure 4.19.

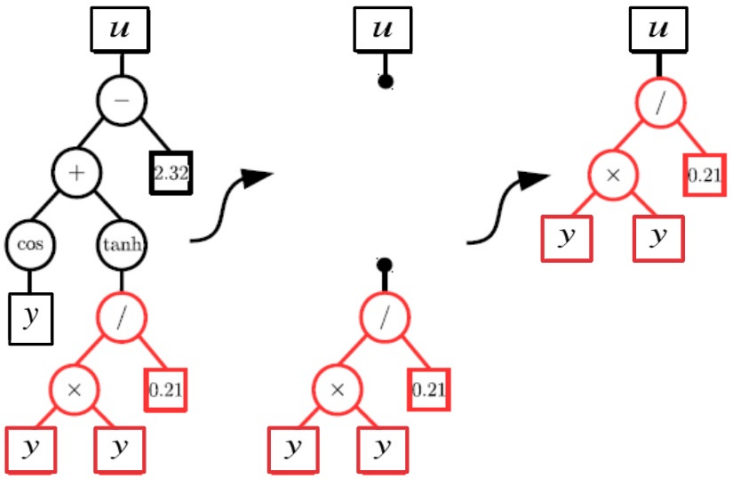

Figure 4.17: Hoist Mutation Operation

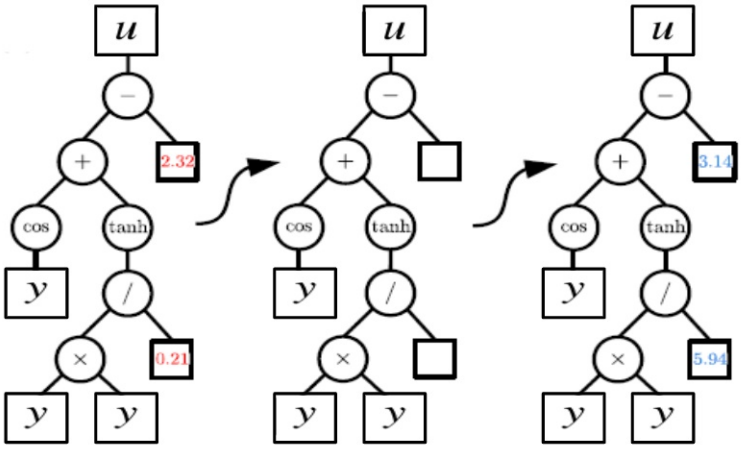

Figure 4.18: Reparameterization Mutation Operation

4.4.3.5 Advancing generations and stopping criteria

There are no general rules to select optimal parameters for evolutionary algorithms. A practice is to check the optimality of the solution by reproducing the process a number of times using various sets of parameters, by this way guided statistics can be achieved. In turbulence control, main interest is in finding an effective control law than in discovering an optimal search algorithm. The important impact of modifying parameters is on the ratio of exploitation (convergence) and exploration of the search space. The best way to fine-tune the MLC process is by monitoring the evolution of the evaluated populations.

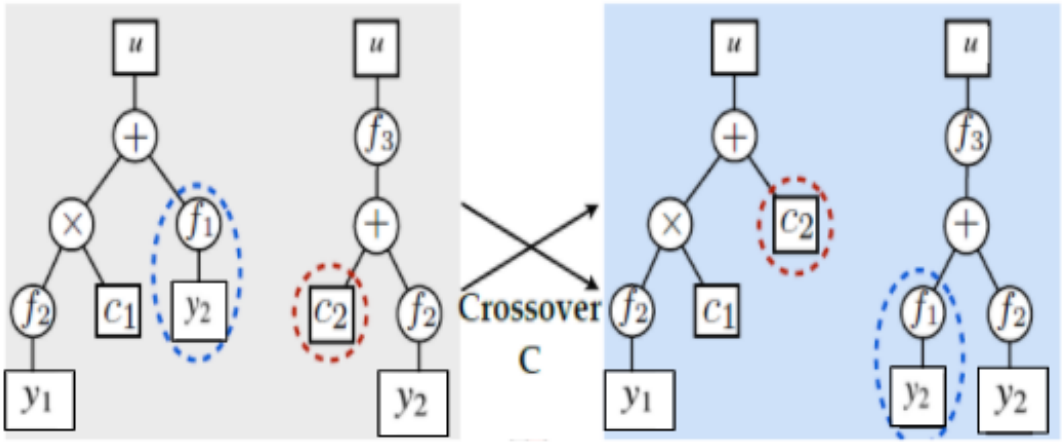

Figure 4.19: Crossover Genetic Operation

4.5 Example for implementation

Machine Learning Control (MLC) creates control design as a regression problem that finds a law that minimizes a given cost function. Genetic Programming (GP) is a regression tool for MLC.

Now, we illustrated genetic programming for a control problem. We used OpenMLC, a freely available Matlab® toolbox designed for MLC, to ease the replication of the results.

4.5.1 Problem formulation

For instance, consider control design of a noise-free, one dimensional ordinary differential equation:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x\left(\frac{x}{10} - \frac{x^2}{1000}\right) + u \tag{4.4}$$

$$y = x \tag{4.5}$$

$$u = K(y) \tag{4.6}$$

Now, we can also write by equation (4.5) and (4.6) as $u = K(x)$.

To find fixed points, put u = o and $\frac{\mathrm{d}x}{\mathrm{d}t} = 0$ in equation (4.4). Then,

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x\left(\frac{x}{10} - \frac{x^2}{1000}\right) + 0 = 0 \tag{4.7}$$

By this equation we will get two fixed points, $x^* = 0$ and $x^{**} = 100$.

The cost function to be minimized penalizes a deviation from desired state $x^* = 0$ and actuation,

$$J = \frac{1}{T}\int_0^T \left[(x(t) - 0)^2 + \gamma u^2(t)\right] dt$$

ie,

$$J = \frac{1}{T}\int_0^T \left[x^2(t) + \gamma u^2(t)\right] dt$$

The cost function to be minimized penalizes a deviation from desired state $x^{**} = 100$ and actuation,

$$J = \frac{1}{T}\int_0^T \left[(x(t) - 100)^2 + \gamma u^2(t)\right] dt$$

where T is evaluation time and γ is penalization coefficient for the cost of the input (actuation).

We set up problem in a script and give associated parameters as well as problem variables. The associated parameters can be represented as instance by calling this script by a command. This command executes the new control regression problem:

$$K(y) = argmin_{K'(y)} J\left[K'(y)\right]$$

This finds a control law that minimizes the cost function value.

The list of parameters used for MLC with genetic programming for control design is given below in Table 4.1:

Parameter	Value
N_i	50
P_m	0.4
P_r	0.1
P_c	0.5
N_p	7
N_e	10

Table 4.1: Table showing parameter value used for MLC with GP for control design.

4.5.1.1 Problem Implementation

The evaluation function for this system control problem should be provided, defining - cost function, variables, plots in it. We have used **OpenMLC[4] toolbox** to practice and learn more to implement different examples. In OpenMLC toolbox, the evaluation function is under unidim_DS_evaluator.m, we implemented our example by changing cost function, variables and plots.

4.5.2 Results

We get best individuals and best cost function value. The state, the control and the cost function integration are plotted against time.

We see the graphs defined in the evaluation function for the best individual for the particular cost function.

For now, take cost function:

$$J = \frac{1}{T}\int_0^T \left[(x(t) - 100)^2 + \gamma u^2(t)\right] dt$$

We selected 15 generations to solve the control problem. Figure 4.20 shows graph having the state, the control and the cost function integration are plotted against time defined in the evaluation function for the above cost function.

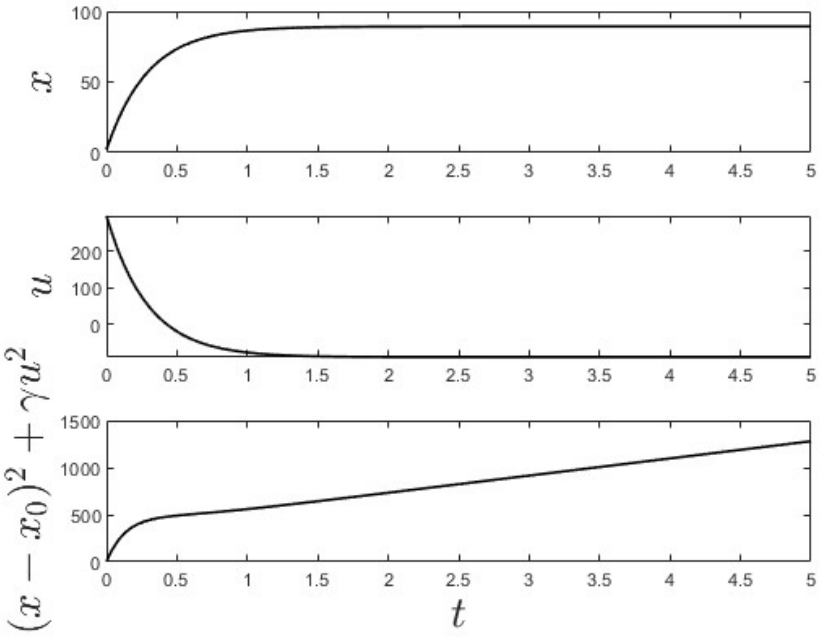

Figure 4.20: Best individual example after 15 generations

[4] https://github.com/MachineLearningControl/OpenMLC-Matlab

5 MLC applied in Synchronization of Networks of Kuramoto Oscillators

In this Chapter, we are going to discuss the thesis baseline based on topics synchronization in the networks of Kuramoto oscillators and Machine Learning Control (MLC) which we have already described in detail in Chapters 3 and 4, respectively. The thesis baseline is - MLC applied in the networks of Kuramoto oscillators. For this task, first we setup our system and describe the control mechanism used to modify the desired synchronization in networks of Kuramoto oscillators along with the cost function to apply the Genetic Programming (GP used for MLC) explained in Chapter 4 as a control mechanism for the synchronization of Kuramoto oscillators. Given a system $\dot{x} = f(x(t))$, some measurements $y(t)$ and a control law $u(y(t))$ and we control the system by adding the strength of actuation $u(y(t))$ to the system, so that the system becomes:

$$\dot{x} = f(x(t), u(y(t))$$

This architecture for the thesis baseline is shown Figure 5.1, where the system (shown in the blue box) is the network of Kuramoto Oscillators.

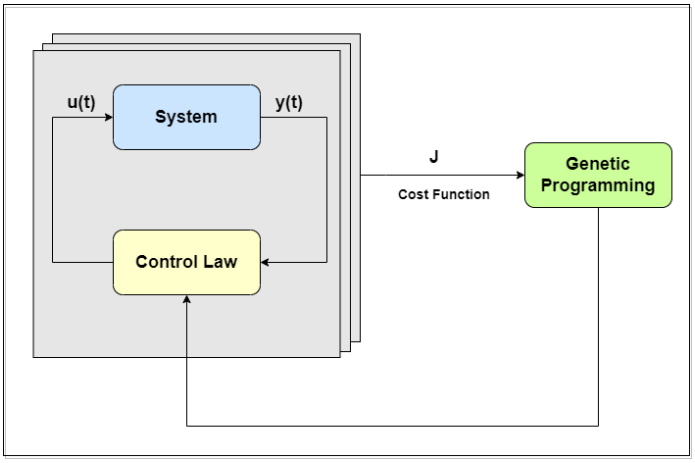

Figure 5.1: Architecture for the Thesis Baseline

In Section 5.1 of this Chapter, we will be describing the development of the system the networks of Kuramoto oscillators used for the task of MLC and then will discuss the desynchronization regime for the system so that system becomes desynchronized to apply control input to control the system to ensure the synchronization of the system. The control mechanism for the desynchronized system of the networks of Kuramoto oscillators is explained in Section

5.2. In Section 5.3, we will setup MLC, using the OpenMLC matlab toolbox mentioned in [DBN16] which we have already discusse in Chapter 4, to apply it on the desynchronized system of the networks of Kuarmoto oscillators instead of normal controller to ensure the synchronization of the system.

5.1 Developing System of Networks of Kuramoto Oscillators

In this thesis, the system is the network of Kuramoto oscillators. A network of N Kuramoto oscillators is developed. Let the matrix $A = [a_{ij}]$ be the weighted adjacency matrix of graph $\mathcal{G}$, where $a_{ij} \in \mathbb{R}$ if $a_{ij} \in \epsilon$ and $a_{ij} = 0$ otherwise. Assume that $\mathcal{G}$ is strongly connected and $\mathbb{S}^1$ is the unit-circle. Let $\theta_i \in \mathbb{S}^1$ stand for the phase angle of the i-th oscillator, whose dynamics is described as a generalized version of Kuramoto system as given below in the equation:

$$\dot{\theta}_i = \omega_i + \sum_{j=1}^{N} a_{ij} \sin(\theta_j - \theta_i), \qquad i = 1, \cdots, N \tag{5.1}$$

We can also define the dynamics of Kuramoto system as given below in the equation:

$$\dot{\theta}_i = \omega_i + \frac{1}{N} \sum_{j=1}^{N} (A \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N \tag{5.2}$$

where N is the total number of networked oscillators, $a_{ij} \in \mathbb{R}$ is the coupling parameter, i is the index of the oscillator connected to sinusoidal phase relationship of other oscillators, $\theta_i \in \mathbb{S}^1$ is the phase angle of the ith oscillator and $\omega_i \in \mathbb{R}$ is the natural frequency of the ith oscillator.

Firstly, we will develop the system of the networks of Kuramoto oscillators and obtain the simulations of the networks of Kuramoto oscillators, which we will discuss in Chapter 6.

The system setup for synchronization of the network of Kuramoto oscillators is presented and the system parameters like the distribution of natural frequencies ω and coupling strength are selected accordingly for the purpose of control so that the uncontrolled system follows a de-synchronization regime as the degree of synchronization is controlled by these parameters. It is carefully checked that the starting system state really lies in the desynchronized regime. We will fix the natural frequencies ω and make the coupling strength weak to make the system desynchronized.

Then, on this uncontrolled desynchronized system of Kuramoto oscillators, we will add control input to the Kuramoto model equation to introduce controller for the particular desynchronized system to achieve the synchronization. In particular, the following system of the networks of Kuramoto oscillators with controller input u is considered:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + u, \qquad i = 1, \cdots, N$$

For this, we will be introducing random value matrix K of the same order as of matrix A, i.e, $(N \times N)$. We will also define matrix $F = [f_{ij}]$ as hadamard product matrix A weighted adjacency matrix of above system and matrix K, i.e., $F = A \circ K$. We will setup the control input u as:

$$u = \frac{1}{N}\sum_{j=1}^{N}(F \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N$$

Then, the above system with control input becomes:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + \frac{1}{N}\sum_{j=1}^{N}(F \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N$$

The control input controls the above desynchronized system to ensure the synchronization of the Kuramoto oscillators. We will obtain the simulations of the networks of Kuramoto oscillators, which we will discuss in Chapter 6. After using random matrix K, we will use optimal K using MLC. We will take the parameters of desynchronized system again to apply MLC.

5.2 Control Mechanism for the System to apply MLC

A control mechanism modifies the weights of the network of oscillators to ensure the formation of the desired synchronization pattern. The control method is optimal as it finds the smallest network perturbation for a given synchronization pattern in the network and ensures the transformation of only a desired subset of the network's edge weights. On the uncontrolled desynchronized system of Kuramoto oscillators, we will add control input to the Kuramoto model equation to introduce controller for the particular desynchronized system to achieve the synchronization.

The GP setup is created to control this system of networks of Kuramoto oscillators to get synchronized. First to apply GP to optimize cost value in order to find best control law, a formulation of the control and cost function is necessary.

In particular, we consider the following system of the networks of Kuramoto oscillators with controller input u:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + u, \qquad i = 1, \cdots, N \tag{5.3}$$

Synchronization must be guaranteed by adding controller input u to the system. For feedback control, the above system provides some measurements vector y:

$$y = \frac{1}{N}\sum_{j=1}^{N}(B \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N$$

Where B is just the adjacency matrix A without weights and $\sin(\theta_j - \theta_i)$ is a big matrix of order $(N \times N)$.

And we control the system by adding the strength of actuation u to the system:

$$u = K(y)$$

We will write above equation (5.3) of the networks of Kuramoto oscillators with controller input u as:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + K(y), \qquad i = 1, \cdots, N \tag{5.4}$$

Now to apply MLC to the system of networks of Kuramoto oscillators, the overall system is:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + K(y), \qquad i = 1, \cdots, N \tag{5.5}$$

$$y = \frac{1}{N}\sum_{j=1}^{N}(B \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N \tag{5.6}$$

$$u = K(y) = K(\frac{1}{N}\sum_{j=1}^{N}(B \circ \sin(\theta_j - \theta_i))), \qquad i = 1, \cdots, N \tag{5.7}$$

5.3 MLC Setup

MLC creates control design as a regression problem that finds a law that minimizes a given cost function. GP is a regression tool for MLC. The GP produces a set of control solutions u, called the population. The solutions are then evaluated in many completion of the control loop; the performance in each iteration is rated through a cost function J and fed back into the GP algorithm as a cost index. The algorithm uses the performance rating to select the

best solutions and evolve them into the next generation of solutions. This learning loop repeats until at least one adequate control law is found.

We describe GP for a control problem. We will use OpenMLC, a Matlab toolbox designed for MLC, to ease the replication of the results. We will discuss the experimental setup used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators in Chapter 6, as instead of using controller we can also apply MLC to find best control law. After applying the GP (used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators, we obtained results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized. Now, we will define the methods below to setup to apply MLC to to the desynchronized system of the networks of Kuramoto oscillators.

First, we formulate cost function. As we know from the definition of phase synchronizability that the network of oscillators $O = (\mathcal{G}, \omega, A)$ where $\mathcal{G} = (\nu, \epsilon)$, ω is a set of natural frequencies of oscillators, and $A = [a_{ij}]$ is the weighted adjacency matrix, is phase synchronizable if it holds:

$$\theta_i(t) = \theta_j(t)$$

for some initial phases $\theta_1(0), \ldots, \theta_N(0)$, for all times $t \in \mathbb{R}_{\geq 0}$ and with all $i, j = 1, \cdots, N$.

Undoubtedly, phase synchronization implies frequency synchronization, while the opposite statement is not true in general. According to our studies in this thesis, the order parameter r is the modulus of the order operator or measure of synchronization, it represents the mean collective behaviour of all the networked coupled oscillators. When $r = 0$, the system is said to be completely unsynchronized, whereas when $r = 1$, the system is said to be in complete harmony and synchrony. r is defined below where $\imath$ is the imaginary unit:

$$r = \left| \frac{1}{N} \sum_{j=1}^{N} e^{\imath \theta_j} \right| \tag{5.8}$$

We use Machine learning algorithms around a complex system to know an effective control law $u = K(y)$, the performance of given control law is decided based on the value of a cost function J, and the machine learning algorithms will aim to minimize this cost function. Let us take simplified cost function to be minimized penalizes a deviation from desired synchronization parameter $r_0 = 1$ and the strenght of actuation:

$$J = \frac{1}{T} \int_0^T \left[(r - r_0)^2 + \gamma u^2(t) \right] = \frac{1}{T} \int_0^T \left[(r - 1)^2 + \gamma u^2(t) \right] dt$$

where T is evaluation time and γ is penalization coefficient for the cost of the input (actuation).

Then, we set up problem of our system in the OpenMLC script unidim_DS_script.m and give associated parameters as well as problem variables. The associated parameters can be represented as instance by calling this script by a command:

```
mlc=MLC('unidim_DS_script');
```

This command executes the new control regression problem:

$$K(y) = argmin_{K'(y)} J\left[K'(y)\right]$$

This finds a control law that minimizes the cost function value. The list of parameters used for MLC with GP for control design can be retrieved by calling:

```
mlc.parameters
```

The important GP parameters are given below in Table 5.1 which are already discussed in Section 4.5 of Chapter 4, where N_i is number of individuals in a population, N_e are the best individuals of the evaluated population following Elitism, i.e., copied directly to the next generation. Remaining $N_i - N_e$ individuals are generated through mutation, replication, and crossover to enter the next generation so the probability of each of these operations are P_m, P_r, and P_c respectively for the generation of remaining individuals, with $P_m + P_r + P_c = 1$. N_p are unique individuals that are randomly selected from the previous generation to enter the one-round tournament.

Parameter	Value
N_i	50
N_e	10
P_m	0.4
P_r	0.1
P_c	0.5
N_p	7

Table 5.1: Table showing important parameters used for MLC with GP for control design of the system of the networks of Kuramoto oscillators

The evaluation function for the control problem of the desynchronized system of the networks of Kuramoto oscillators should be provided by defining - cost function, variables, plots in OpenMLC script unidim_DS_evaluator.m. We will implement our experiments on the system of the networks of Kuramoto oscillators discussed in Chapter 6 in detail, by changing cost function, variables and plots in the script unidim_DS_evaluator.m. We will be retrieving the parameters which we will store in this script by calling:

```
mlc.parameters.problem_variables
```

We will get best individuals and best cost function value. The system state, the control and the cost function integration will be plotted against time. We will see the graphs defined in the evaluation function for the best individual for the particular cost function J defined above. We will be selecting 15 generations to solve the control problem. After all the analysis, we will observe whether the desynchronized system of the networks of Kuramoto oscillators is synchronized or not.

6 Experiments and Results

This Chapter will discuss the experiments with the results based on the methods described in Chapter 5 in detail. We will check the simulations of the networks of Kuramoto oscillators (this topic is already mentioned in Chapter 3) and desynchronize the system of the networks of Kuramoto oscillators by changing the system parameters like the distribution of natural frequencies ω and coupling strength are selected accordingly for the purpose of control so that the uncontrolled system becomes desynchronized as the degree of synchronization is controlled by these parameters. We will keep the natural frequencies same and make the coupling strength weak to make the system desynchronized. On this uncontrolled desynchronized system of Kuramoto oscillators, we will add control input to the Kuramoto model equation to introduce controller for the particular desynchronized system to achieve the synchronization. We will also discuss the experimental setup used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators, as instead of using controller we can also apply MLC to find the control law. After applying the GP (used for MLC) described in 4 as a control mechanism for the synchronization of Kuramoto oscillators, we obtained results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized.

We divide this Chapter into two sections. The first Section 6.1 will firstly discuss the Experiment I, for the first experiment of this thesis, we developed a directed weighted network of 6 nodes of Kuramoto oscillators (similar to the graph from [Tib+17]). We will discuss experimental setup and obtain results. Then, in Section 6.2, we will describe the Experiment II of this thesis for a directed network of 12 nodes of Kuramoto oscillators, experimental setup and obtain results. At last, we also mentioned the results for the large networks after applying MLC as a control mechanism for the desynchronized system of the networks of Kuramoto oscillators.

6.1 Experiment I and Results

The weighted network of 6 nodes of Kuramoto oscillators

For the first experiment of this thesis, we created a directed weighted graph of the network of 6 nodes of Kuramoto oscillators (similar to the graph from [Tib+17]), the graph of this network is shown in Figure 6.1 below:

This graph is strongly connected graph. The weighted adjacency matrix A for this graph is

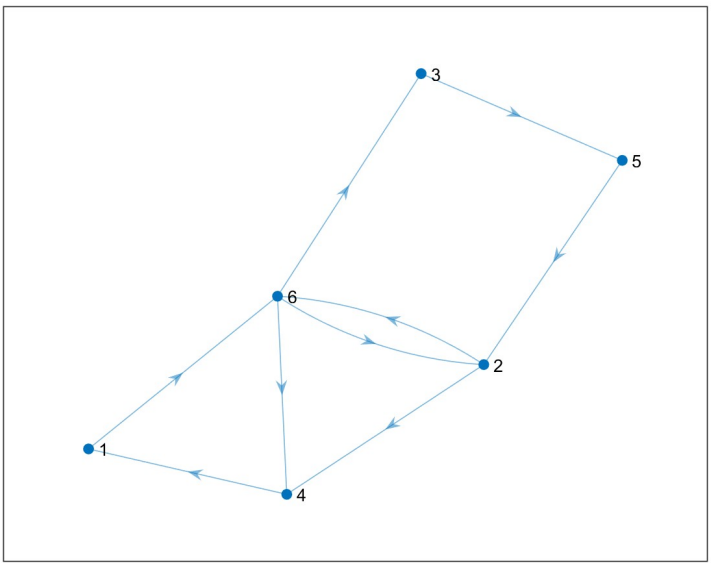

Figure 6.1: Directed Weighted Graph of the Network of 6 nodes

given as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 5 & 0 & 5 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 \\ 0 & 7 & 2 & 2 & 0 & 0 \end{bmatrix}$$

In Appendix A, the code for the weighted network of $N = 6$ nodes of Kuramoto oscillators is provided. This code also solves the differential equation of Kuramoto model:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N \tag{6.1}$$

where the matrix $A = [a_{ij}]$ of order $(N \times N)$ described above is the weighted adjacency matrix of graph $\mathcal{G}$ shown in Figure 6.1. To get the solution for above equation, we initialized the phase matrix $\theta \in \mathbb{S}^1$ (unit-circle) of order $(N \times N)$ and we also defined fixed parameter the distribution of natural frequencies vector ω of order $(N \times 1)$, here we used $\omega = [1; 1; 1; 1; 1; 1] \times \frac{1}{100}$. The simulation of this network of Kuramoto oscillators is shown in Figure 6.2 which shows the plot of $\sin(\theta)$ against time steps t. We analysed by Figure 6.2 that approximately after 5000 time steps all the 6 oscillators start synchronizing and after 10000 time steps all the 6 oscillators get completely synchronized.

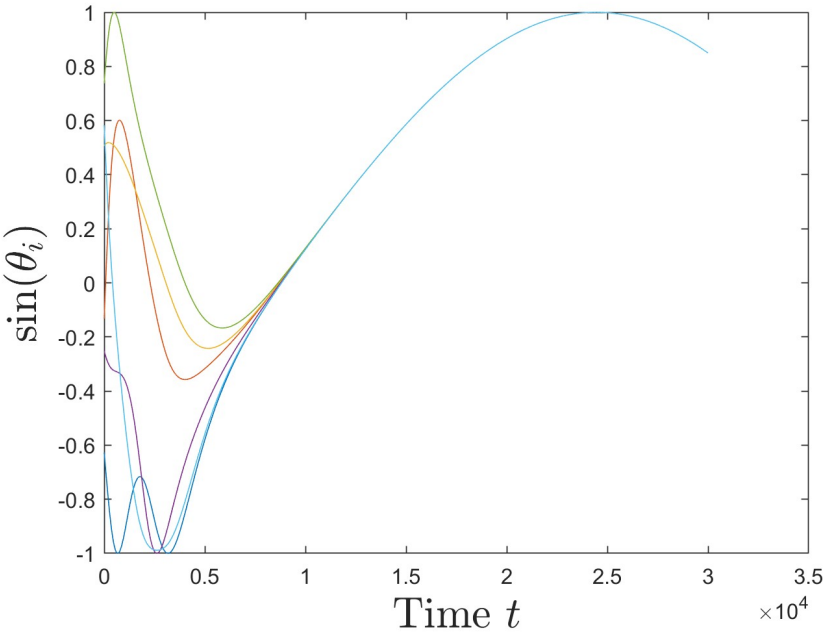

Figure 6.2: The Simulation of the Network of 6 Kuramoto Oscillators

Figure 6.3 shows the plot order parameter r for 6 Kuramoto oscillators against time steps t.

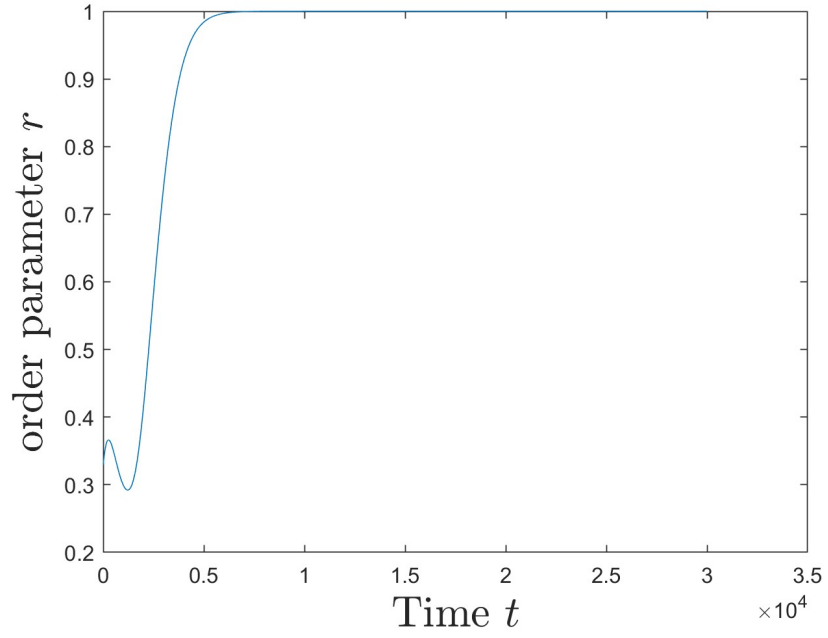

Figure 6.3: Order Parameter r for the Simulation of the Network of 6 Kuramoto Oscillators

By this plot, we analysed the same as we analysed from Figure 6.2, we already know from our study that order parameter r describes synchronization among the oscillators as when the value of r approaches to value 1, then the synchronization starts occurring. We depicted that approximately after 5000 time steps all the 6 oscillators start synchronizing and after 10000 time steps all the 6 oscillators become completely synchronized.

The system setup for synchronization of the network of Kuramoto oscillators is presented. Now, the system parameter coupling strength is selected accordingly for the purpose of control so that the uncontrolled system becomes desynchronized as the degree of synchronization is controlled by this parameter. We kept same ω and made the coupling strength weak to make the system desynchronized, for that we multiplied 0.001 with the coupling. As a result of this, we observed that all the 6 oscillators are not synchronized now which is shown in Figure 6.4.

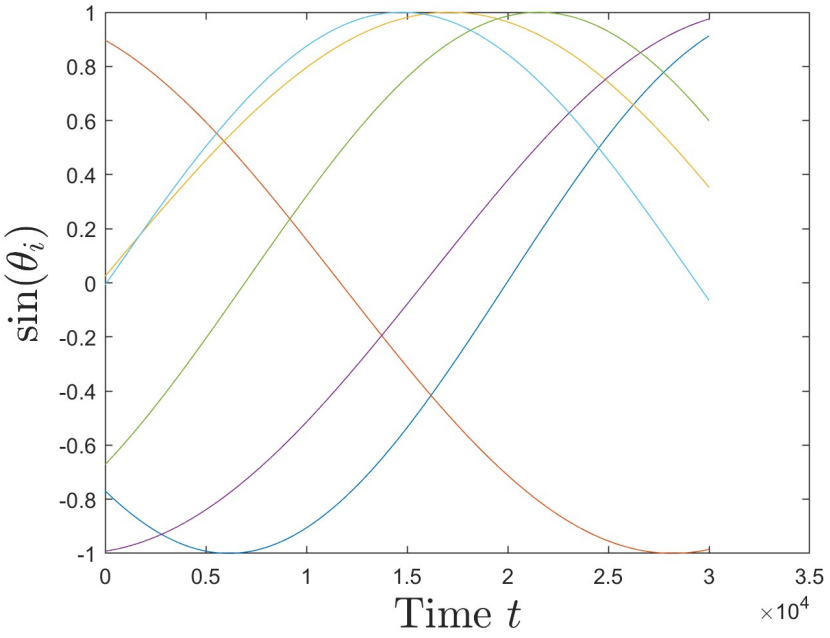

Figure 6.4: The Simulation of the Network of 6 Desynchronized Kuramoto Oscillators

Figure 6.5 also shows that the synchronization is not achieved as the value of r is not approaching to value 1. The value of r just reached approximately to 0.44.

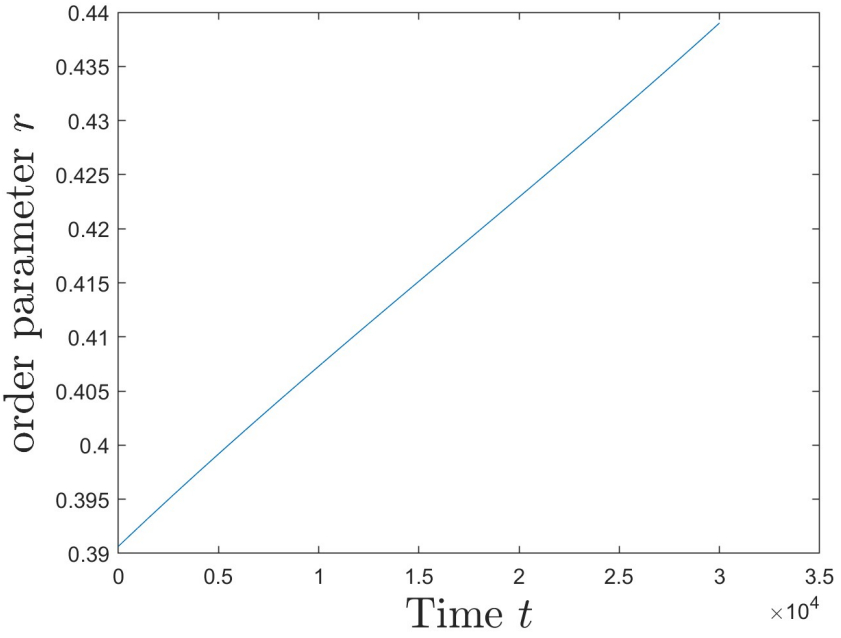

Figure 6.5: Order Parameter r for the Simulation of the Network of 6 Desynchronized Kuramoto Oscillators

The weighted network of 6 nodes of Kuramoto oscillators with Control Input to ensure the Synchronization of Desynchronized Oscillators

For the purpose of control, the uncontrolled system made desynchronized as mentioned above and now to ensure the synchronization of these desynchronized oscillators we will add control input to the system of network of Kuramoto oscillators. The equation for our system with control input is:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + u, \qquad i = 1, \cdots, N \tag{6.2}$$

where u is the control input that we setup:

$$u = \frac{1}{N}\sum_{j=1}^{N}(F \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N$$

where matrix $F = [f_{ij}]$ is hadamard product of the weighted adjacency matrix A of above system and matrix K which is a random matrix of the same order as of matrix A, i.e., $F = A \circ K$. Here, we used matrix K as:

```
K =
    0.3989   0.9527   0.9362   0.8648   0.8948   0.1495
    0.7247   0.9669   0.6160   0.8266   0.0741   0.4620
    0.7510   0.6959   0.2802   0.0034   0.0605   0.4882
    0.5645   0.8733   0.0042   0.2565   0.4232   0.6292
    0.3477   0.1604   0.8377   0.0893   0.0125   0.2515
    0.5300   0.5031   0.1709   0.9990   0.6910   0.3043
```

The control input controls the above desynchronized system to ensure the synchronization of the Kuramoto oscillators.

Appendix B represents the code for this system with control input having all the parameters same as above desynchronized system so that we can control the system by adding control input which is described in equation (6.2). In this code, we defined matrix K of order $(N \times N)$ having random values and we got the above mentioned matrix K and also defined matrix F as $F = A \circ K$, where A is the same $(N \times N)$ weighted matrix mentioned above. Finally, we added control input to the old desynchronized system. This code solves the differential equation of Kuramoto model having control input.

This system with control input ensures the synchronization of all the 6 desynchronized oscillators clearly visible from Figures 6.6 and 6.7. Initially the oscillators were not synchronized, but after applying controller oscillators start to synchronize approximately after 500 time steps and get completely synchronized after 1000 time steps as shown in Figure 6.6. Figure 6.7 shows that the value of r approaches to value 1 and after 1000 time steps the value becomes 1 as the system becomes completely synchronized.

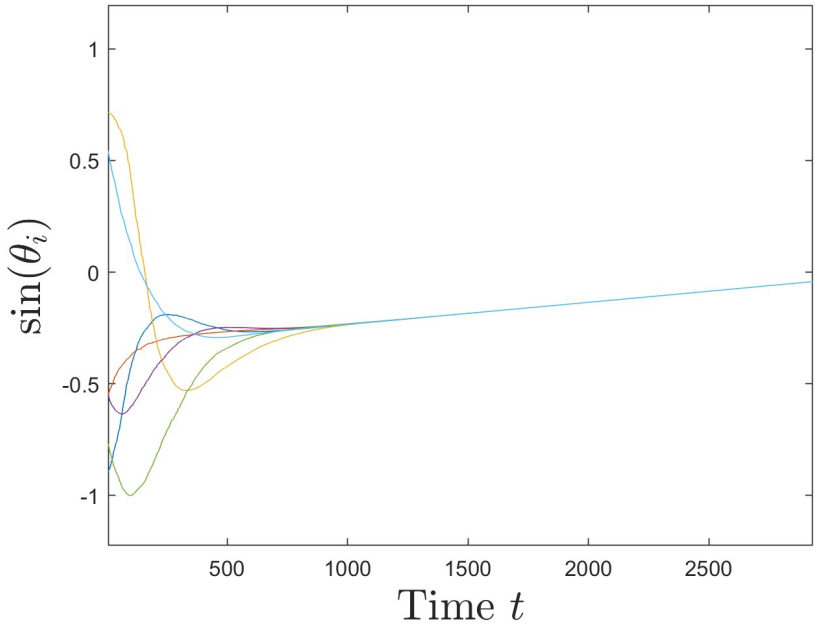

Figure 6.6: The Synchronization of the Network of 6 Kuramoto Oscillators with Control Input

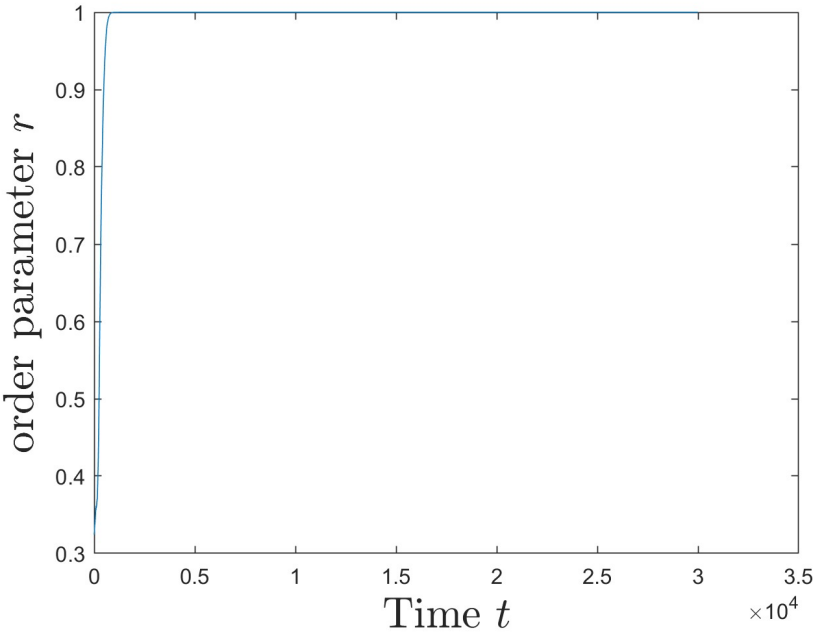

Figure 6.7: Order Parameter r for the Synchronization of the Network of 6 Kuramoto Oscillators with Control Input

For now, we have used controller to control the system to ensure the synchronization of desynchronized system. We have used random K, now we will use optimal $K(y)$ using MLC. In this thesis, we have studied that we can use MLC technique to control any system and get the control law. Now, we will use GP for our above desynchronized system to ensure the synchronization of the system.

In the below section, we will use all the parameters of our above desynchronized system to setup our system to apply GP on this desynchronized system to find optimal $K(y)$ and compare the results.

MLC applied to the weighted network of 6 nodes of Kuramoto oscillators to ensure the Synchronization of Desynchronized Oscillators

MLC creates control design as a regression problem that finds a law that minimizes a given cost function. GP is a regression tool for MLC. We used OpenMLC, a freely available Matlab toolbox designed for MLC, we have already discussed about this toolbox and GP setup in Section 4.5 of Chapter 4 and 5.3 of Chapter 5.

We applied GP to a control problem of our desynchronized system to know an effective control law: $u = K(y)$ and the performance of given control law is decided based on the value of a cost function J as GP will aim to minimize this cost function.

The system is:

$$\dot{\theta}_i = \omega_i + \frac{1}{N}\sum_{j=1}^{N}(A \circ \sin(\theta_j - \theta_i)) + K(y), \qquad i = 1, \cdots, N$$

$$y = \frac{1}{N}\sum_{j=1}^{N}(B \circ \sin(\theta_j - \theta_i)), \qquad i = 1, \cdots, N$$

$$u = K(y) = K(\frac{1}{N}\sum_{j=1}^{N}(B \circ \sin(\theta_j - \theta_i))), \qquad i = 1, \cdots, N$$

The cost function to be minimized penalizes a deviation from desired synchronization parameter $r_0 = 1$ and the strenght of actuation:

$$J = \frac{1}{T}\int_0^T \left[(r - r_0)^2 + \gamma u^2(t)\right] = \frac{1}{T}\int_0^T \left[(r-1)^2 + \gamma u^2(t)\right] dt$$

where T is evaluation time and γ is penalization coefficient for the cost of the input (actuation).

To apply MLC to our system, we defined the parameters of the above desynchronized system like weighted adjacency matrix A of the graph $\mathcal{G}$, the same vector $\omega = [1;1;1;1;1;1] \times \frac{1}{100}$, number of oscillators N, and weak coupling $kappa = 0.001$ to ensure that the system will be desynchronized without control so that the system becomes synchronized by using GP. Apart from this, we defined $(N \times N)$ matrix B which is unweighted matrix of graph $\mathcal{G}$, as this matrix B will be used for the control problem of our system. We also provided a few parameters for the cost function like $r_0 = 1$ as $objective = 1$ and $\gamma = 0.1$. We defined all these parameters in the OpenMLC script unidim_DS_script.m as shown below:

```
parameters.problem_variables.Tf = 5;
parameters.problem_variables.objective=1; % r0 value for cost
    function
parameters.problem_variables.gamma=0.1;
parameters.problem_variables.Tevmax=1;

parameters.problem_variables.N=6;

parameters.problem_variables.G=digraph([1 2 2 3 4 5 6 6 6],[6 4
    6 5 1 2 2 3 4],[10 5 5 10 9 9 7 2 2],{'1','2','3','4','5','6'
    });

parameters.problem_variables.A=adjacency(digraph([1 2 2 3 4 5 6
    6 6],[6 4 6 5 1 2 2 3 4],[10 5 5 10 9 9 7 2 2],{'1','2','3','
    4','5','6'}),'weighted');

```

```
parameters.problem_variables.omega=ones(6,1)*1/100;
parameters.problem_variables.B=adjacency(digraph([1 2 2 3 4 5 6
   6 6],[6 4 6 5 1 2 2 3 4],[10 5 5 10 9 9 7 2 2],{'1','2','3','
   4','5','6'}));
parameters.problem_variables.kappa=0.001;
```

After instantiating the associated object and obtaining full list of GP parameters by calling: mlc=MLC('unidim_DS_script'); and mlc.parameters respectively, we retrieve all the parameters which are stored as:

```
mlc.parameters.problem_variables

ans =

  struct with fields:

           Tf: 5
    objective: 1
        gamma: 0.1
       Tevmax: 1
            N: 6
            G: [1x1 digraph]
            A: [6x6 double]
        omega: [6x1 double]
            B: [6x6 double]
        kappa: 0.001
```

The evaluation function for this system control problem is provided in Appendix C by changing the OpenMLC script named as unidim_DS_evaluator.m. This implements the control regression problem to find a control law which minimizes the cost function J value.

The system is written as a actuated dynamics f defining the derivative of the state θ_i having the control law, and a OpenMLC toolbox function $testt(toc, Tevmax)$ as:

```
K=@(y)(1/N*sum(B.*sin(y-y'))');
eval(['K=@(y)(' m ');']);
f=@(t,y) (omega + kappa/N*sum(A.*sin(y-y'))' + K(y) + testt(toc,
   Tevmax));
```

When the time elapsed for this evaluation is greater than the time described in $Tevmax$ then this evaluation returns an error. This works by including the *tic* function at the starting of the evaluation function.

If we launch the problem by writing:

```
mlc.go(15,1)
```

A graphical output is be produced for the best individual of each generation. We get best individuals and best cost function value. Here, the state of the system, the control and the cost function J integration are plotted against time.

OpenMLC provides methods to analyze the whole evolution process. For example by typing:

```
mlc.show_convergence
```

This displays a succession of histograms of the cost function values for each generation. We will select 15 generations as in Figure 6.8, we see that the cost function J value remains same. We know GP minimises the cost value, so if the value of cost stable so we do not need more generations. Figure 6.8 shows the graph for the cost function J plotted against j generations.

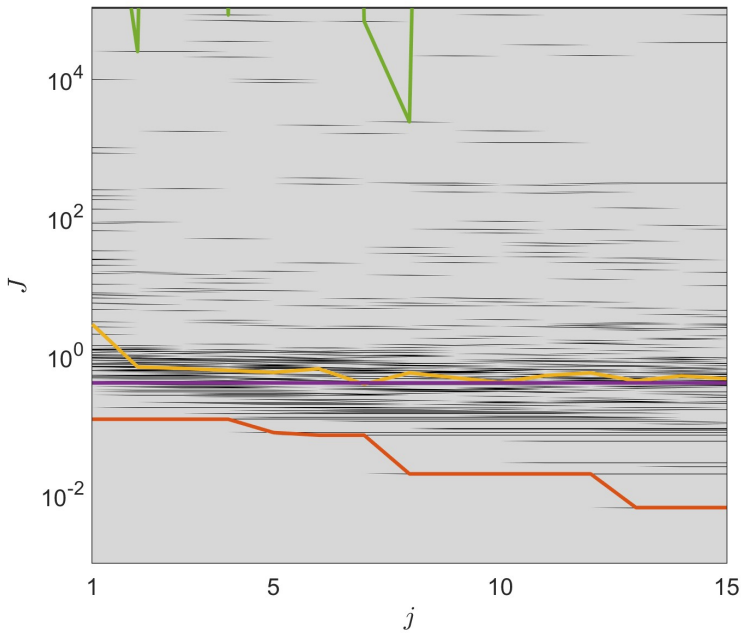

Figure 6.8: Graph for Cost function J plotted against j generations for the network of 6 Kuramoto oscillators

We selected 15 generations to solve the control problem and MLC is initiated using:

```
mlc.go(15)
```

Again by calling:

```
mlc.show_best_indiv
```

This provides the graphs described in the evaluation function for the best individual as shown in Figure 6.9. Figure 6.9 shows graph having the state of the system, the control and the cost

function integration are plotted against time defined in the evaluation function for the above mentioned cost function J.

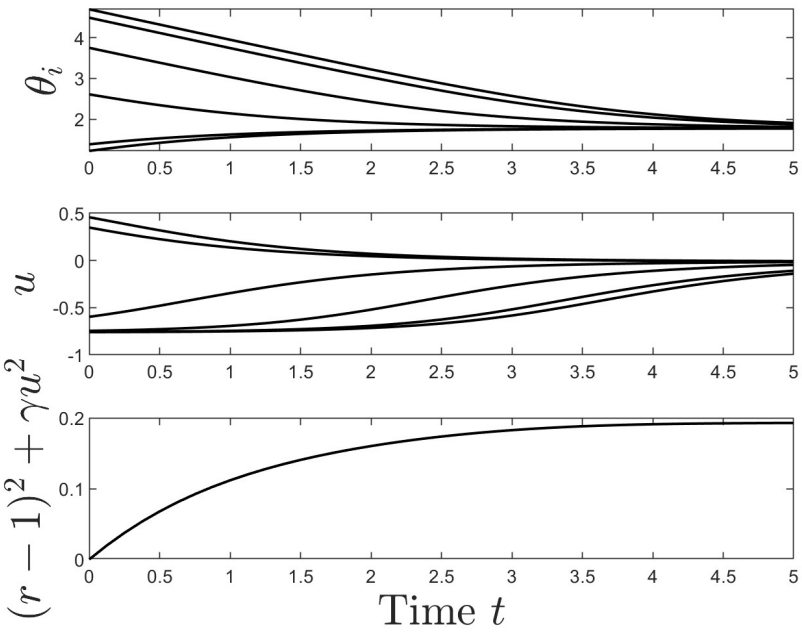

Figure 6.9: Graph for the best individual after 15 generations of the network of 6 Kuramoto oscillators, showing the state of the system, the control and the cost function integration plotted against time

We analysed by this Figure that by applying GP to a control problem of the oscillators of the desynchronized system, the system starts synchronizing after 2.5 time t and gets completely synchronized after 5 time t. The control value of each oscillator also stabilizes.

From Figures 6.10 and 6.11, we observed that the system of the network of 6 nodes of Kuramoto oscillators starts synchronizing after applying MLC to our desynchronized system of the network of 6 nodes of Kuramoto oscillators.

We also tried to run GP by changing the value of γ between 0.2 and 1. We analysed the results and found that if γ is greater than 0.2 then value of control remained zero, the reason for this is that the optimization criterion is more interested in controlling cost value rather than our synchronization criterion. So the system was not controlled as a result of it the synchronization was not achieved by the system.

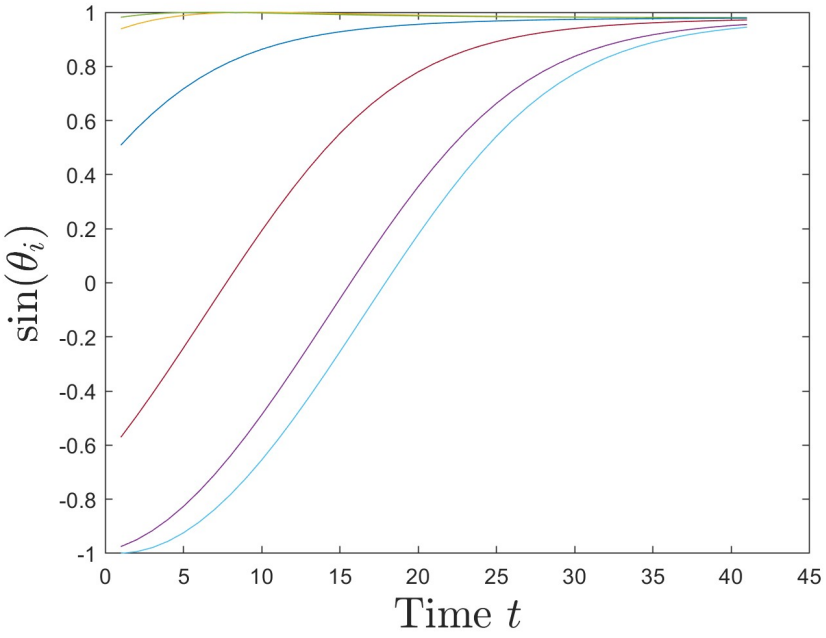

Figure 6.10: The Synchronization of the Network of 6 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

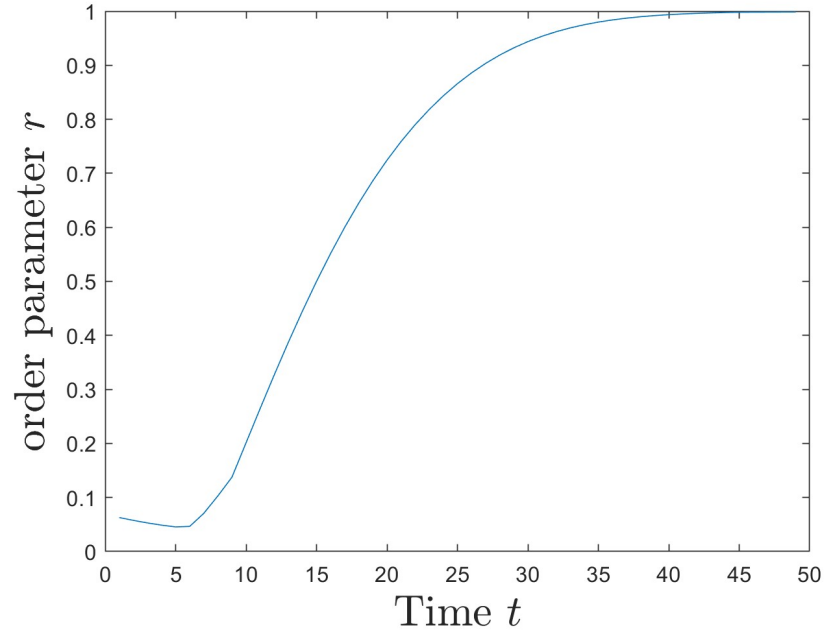

Figure 6.11: Order Parameter r for the Synchronization of the Network of 6 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

6.2 Experiment II and Results

For the second experiment of this thesis, we developed a directed graph of the network of 12 nodes of Kuramoto oscillators, the graph of this network is shown in Figure 6.12 below.

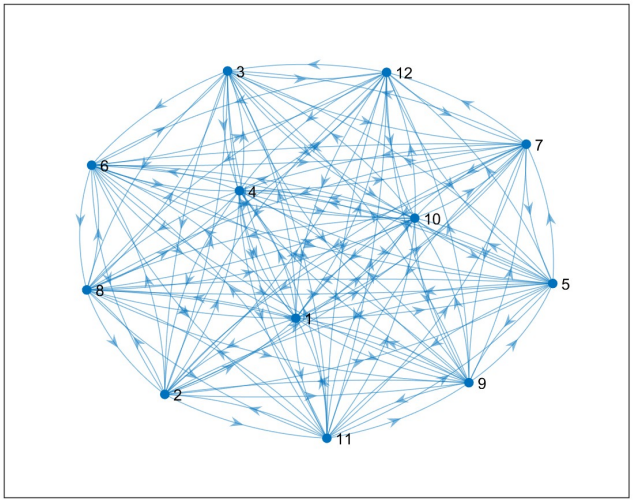

Figure 6.12: Directed Graph of the Network of 12 nodes

This graph is strongly connected graph. The adjacency matrix A of this graph has all the connected links accept self-loops, so the values in this matrix are ones accept the diagonal values which are zeros. In Appendix A, we just changed the network of Kuramoto oscillators as mentioned below and the remaining code remained the same to solve the differential equation of Kuramoto model.

```
%% Network
N = 12; %% no. of nodes
A=[ones(N/2) ones(N/2);ones(N/2) ones(N/2)] - eye(N); %%
    weighted adjacency matrix
G = digraph(A); %% directed strongly connected graph
```

The number of oscillators $N = 12$ and the matrix $A = [a_{ij}]$ of order $(N \times N)$ described above is the adjacency matrix of the above mentioned graph $\mathcal{G}$ shown in Figure 6.12. To get the solution for the Kuramoto model equation, we used remaining code of Appendix A where we initialized the phase matrix $\theta \in \mathbb{S}^1$ (unit-circle) of order $(N \times N)$ and we also defined fixed parameter the distribution of natural frequencies vector ω of order $(N \times 1)$ as a vector of (12,1) having ones and multiplied by $\frac{1}{100}$.

The simulation of this network of Kuramoto oscillators is shown in Figure 6.13 which shows the plot of $\sin(\theta)$ against time steps t. We analysed by Figure 6.13 that approximately after 10000 time steps all the 12 oscillators get completely synchronized.

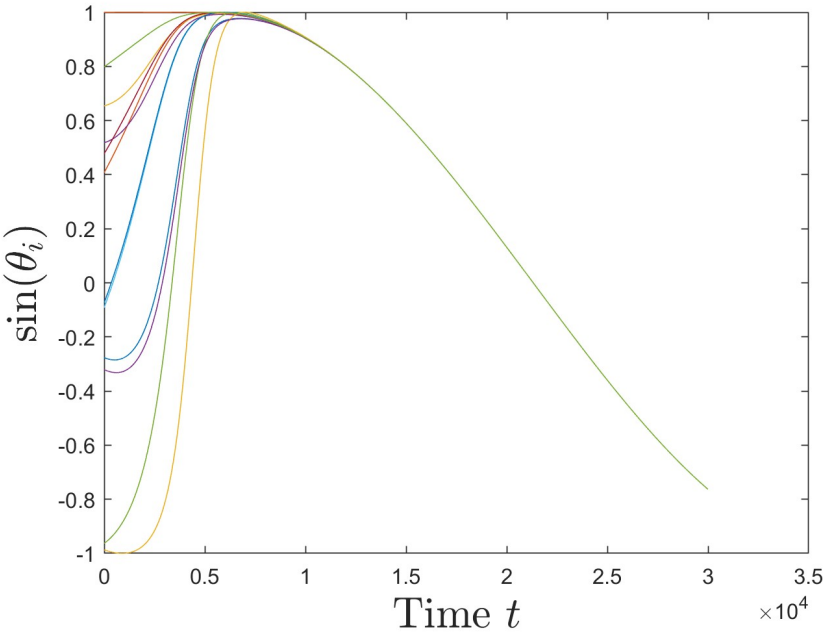

Figure 6.13: The Simulation of the Network of 12 Kuramoto Oscillators

Figure 6.14 shows the plot order parameter r for 12 Kuramoto oscillators against time steps t.

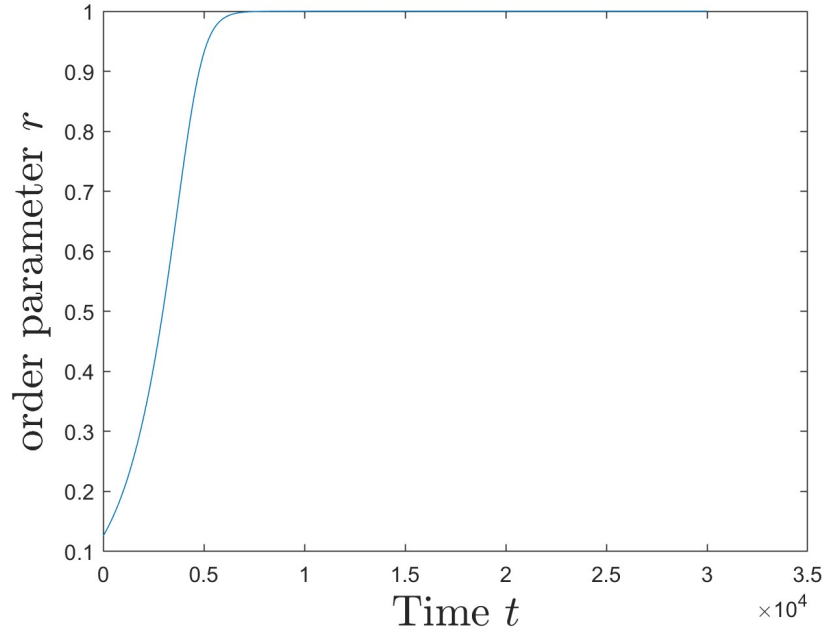

Figure 6.14: Order Parameter r for the Simulation of the Network of 12 Kuramoto Oscillators

We analysed this plot and depicted that approximately after 10000 time steps all the 12 oscillators become completely synchronized.

The system setup for synchronization of the network of Kuramoto oscillators is presented. Now, similarly like above Experiment I, the system parameter like the coupling strength are selected accordingly for the purpose of control so that the uncontrolled system becomes desynchronized as the degree of synchronization is controlled by this parameter. As a result of this, we observed that all the 12 oscillators are not synchronized now which is shown in Figure 6.15.

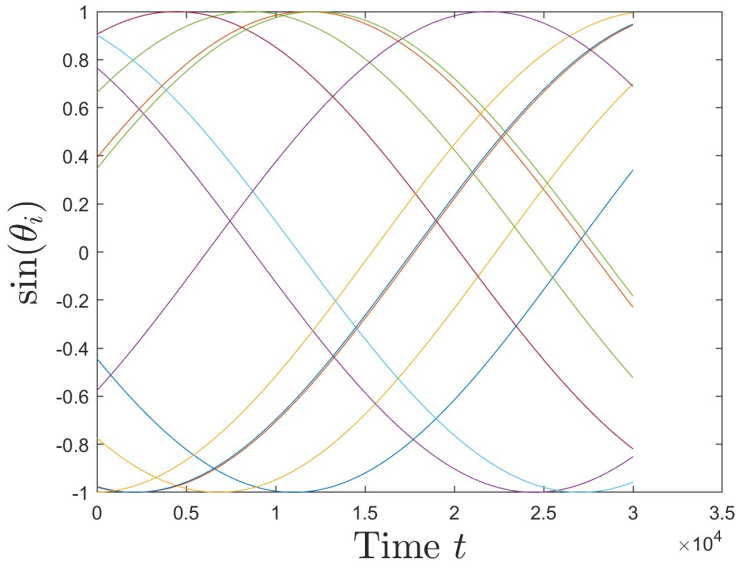

Figure 6.15: The Simulation of the Network of 12 Desynchronized Kuramoto Oscillators

For the purpose of control, the uncontrolled system made desynchronized as mentioned above and now to ensure the synchronization of these desynchronized oscillators we will add control input to the system of network of Kuramoto oscillators, in the similar way we did in the Experiment I. The control input controls the above desynchronized system to ensure the synchronization of the Kuramoto oscillators.

We used Appendix B along with the network we defined in this experiment, Appendix B represents the code for this system with control input having all the parameters same as above desynchronized system so that we can control the system by adding control input in Kuramoto model equation. This code solves the differential equation of Kuramoto model having control input. We used a fixed random matrix K, we cannot show here as it is a (12×12) order matrix.

This system with control input ensures the synchronization of all the 12 desynchronized oscillators clearly visible from Figures 6.16 and 6.17. Initially the oscillators were not synchronized, but after applying controller oscillators start to synchronize approximately after 1000 time steps and get completely synchronized after 2000 time steps as shown in Figure 6.16. Figure 6.17 shows that the value of r approaches to value 1 and after 2000 time steps

the value becomes 1 as the system becomes completely synchronized.

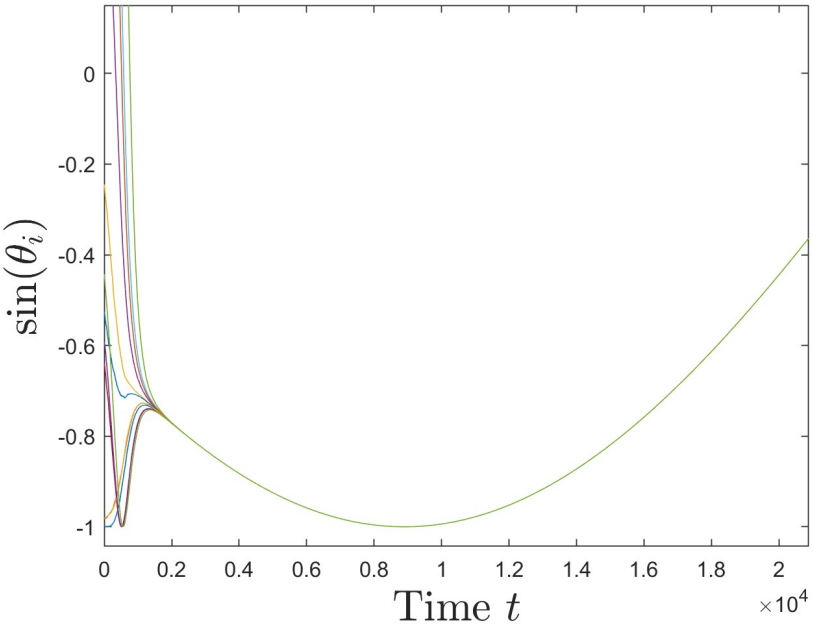

Figure 6.16: The Synchronization of the Network of 12 Kuramoto Oscillators with Control Input

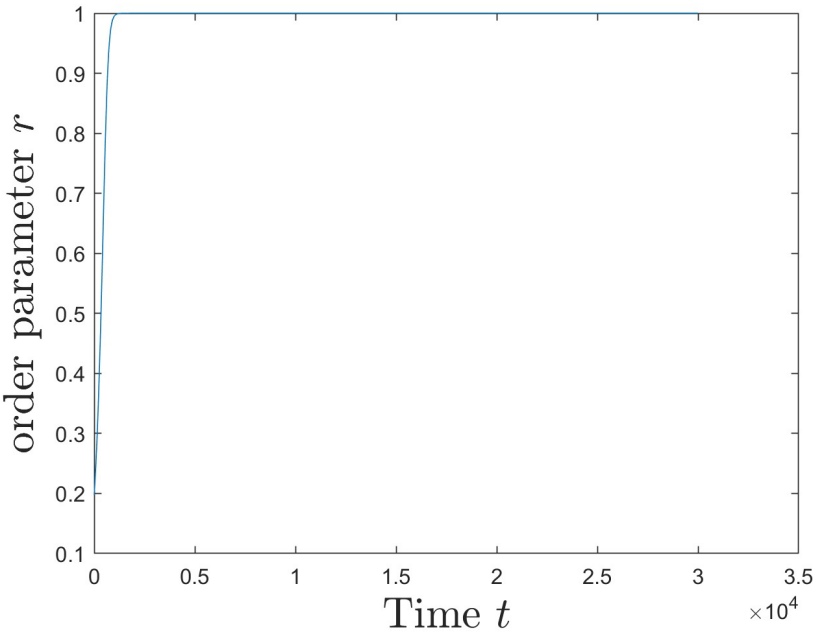

Figure 6.17: Order Parameter r for the Synchronization of the Network of 12 Kuramoto Oscillators with Control Input

For now, we have used controller by using random K to control the system to ensure the synchronization of desynchronized system but in this thesis, We have proposed to use MLC technique to control our system and get the control law. Now, we will use optimal $K(y)$ using MLC. We will use GP for our above desynchronized system to ensure the synchronization of the system. In the below section, we will use all the parameters of our above desynchronized system to setup our system to apply GP on this desynchronized system to compare the results. We will be following all the steps similar to the steps mentioned in the Experiment I for the application of GP to the system along with the setup for the network described in this experiment by defining parameters according to this network in the OpenMLC script unidim_DS_script.m except the matrix B which was defined in the Experiment I, here we can use matrix A as it is not weighted. After all the setup, we will run GP as defined in the Experiment I by using code mentioned in Appendix C.

We will select 15 generations similarly for the same reason as we have mentioned above in the Experiment I. In Figure 6.18, we see that the cost function J value remains same. Figure 6.18 shows the graph for the cost function J plotted against j generations.

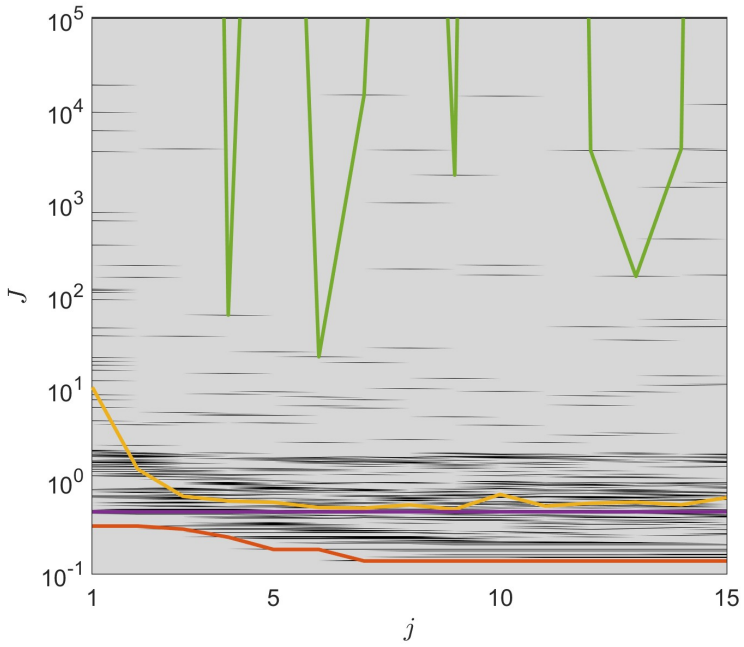

Figure 6.18: Graph for Cost function J plotted against j generations for the network of 12 Kuramoto oscillators

By calling: mlc.show_best_indiv, the graphs described in the evaluation function for the best individual will be provided as shown in Figure 6.19. Figure 6.19 shows graph having the state of the system, the control and the cost function integration are plotted against time defined in the evaluation function for the above mentioned cost function J.

We analysed by this Figure that by applying GP to a control problem of the oscillators of the desynchronized system, the system starts synchronizing and gets synchronized at some extent.

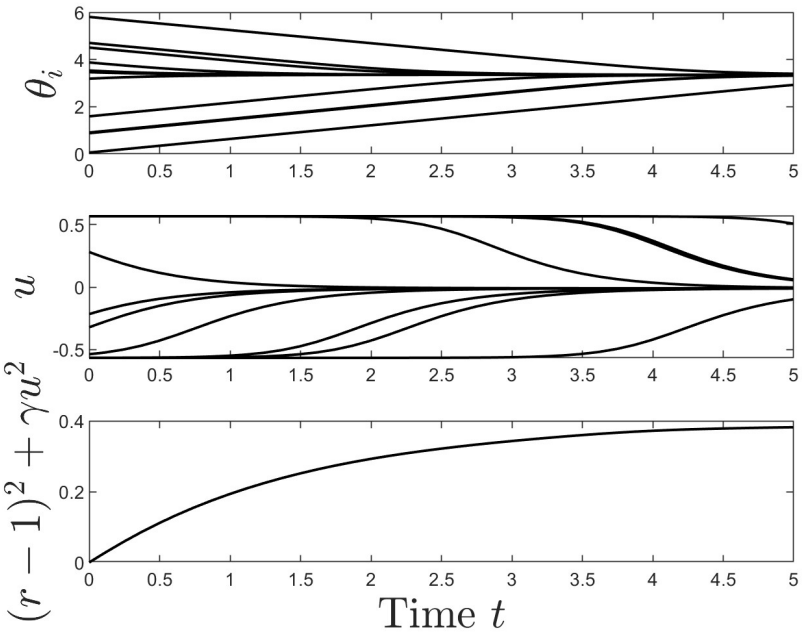

Figure 6.19: Graph for the best individual after 15 generations of the network of 12 Kuramoto oscillators, showing the state of the system, the control and the cost function integration plotted against time

The control value of each oscillator also stabilizes. From Figures 6.20 and 6.21, we observed that the system of the network of 12 nodes of Kuramoto oscillators starts synchronizing after applying MLC to our desynchronized system.

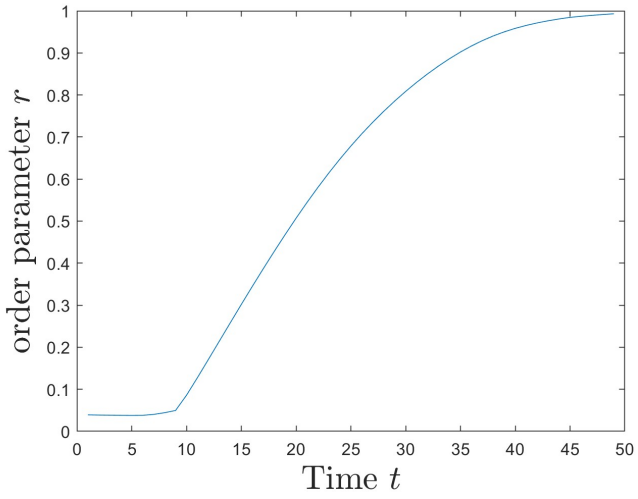

Figure 6.20: Order Parameter r for the Synchronization of the Network of 12 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

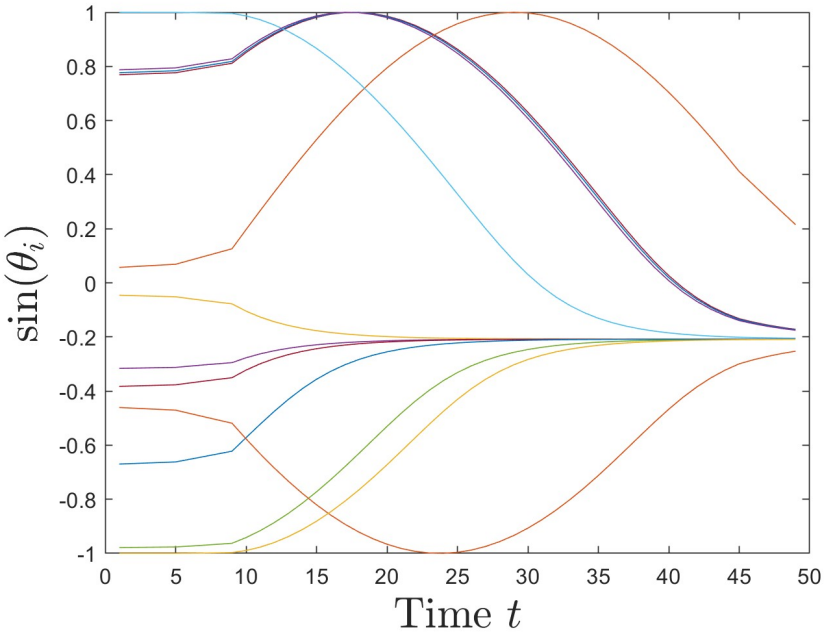

Figure 6.21: The Synchronization of the Network of 12 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

We simulated the different networks of Kuramoto oscillators (by changing the number of oscillators, $N = 16$ to 256 in the above mentioned network of this experiment) and desynchronized them to check that whether the oscillators get completely synchronized, partially synchronized or not at all synchronized after the addition of control input. We analysed that the oscillators of any size network were completely synchronized after adding control input. We also tried to run GP on the desynchronized system of the networks of Kuramoto oscillators having number of oscillators, $N = 16$ to 256, by changing the value of N in the above mentioned network of this experiment. We analysed the results and found that the synchronization was partial among the oscillators of the networks ($N = 16$ to 20). The desynchronized system of the networks of Kuramoto oscillators (greater than $N = 20$) was not at all synchronized, it remained desynchronized after the application of GP as GP was not capable enough to find out best control law for large networks. The value of control also remained zero, so the system was not controlled as a result of it the synchronization was not achieved by the system.

Here, we are going to present the results of GP applied to the the desynchronized system of the network of Kuramoto oscillators having number of oscillators, $N = 24$. Figure 6.22 shows graph having the state of the system, the control and the cost function integration are plotted against time defined in the evaluation function for the above mentioned cost function J. We analysed by this Figure that by applying GP to a control problem of the oscillators of the desynchronized system, the system not all gets synchronized and the control value of each oscillator is not stabilized.

From Figures 6.23 and 6.24, we analysed that the system of the network of 24 nodes of

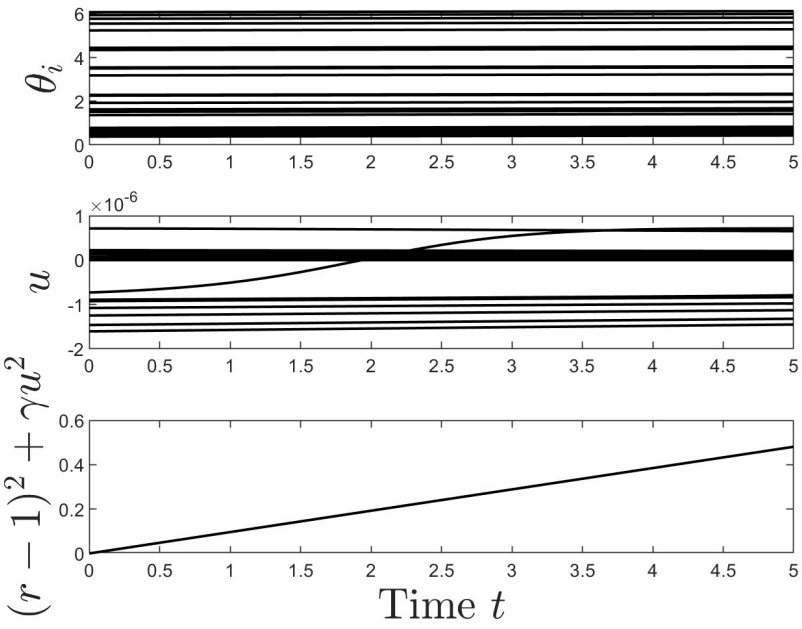

Figure 6.22: Graph for the best individual after 15 generations of the network of 24 Kuramoto oscillators, showing the state of the system, the control and the cost function integration plotted against time

Kuramoto oscillators remained desynchronizing after applying MLC to our desynchronized system of the network of 24 nodes of Kuramoto oscillators. As in Figure 6.23, we noticed that the oscillators do not came to together at one phase.

Order parameter r value is also very low as described in Figure 6.24, we clearly observed that the oscillators are not at all synchronized after the complete particular time.

When we ran GP on the desynchronized system of the networks of Kuramoto oscillators having number of oscillators, $N = 100$ to 256, then we observed that our computer was not powerful enough to evaluate such a large network as the code was crashed. It is clear that to apply MLC on such large networks, we require powerful computer for the computation of individuals and evolution of individuals to the next generation.

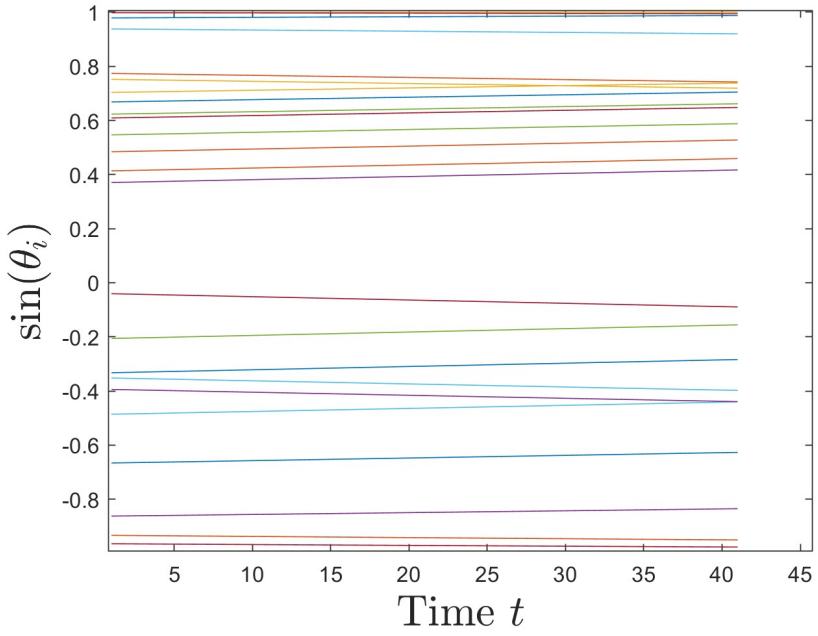

Figure 6.23: The Synchronization of the Network of 24 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

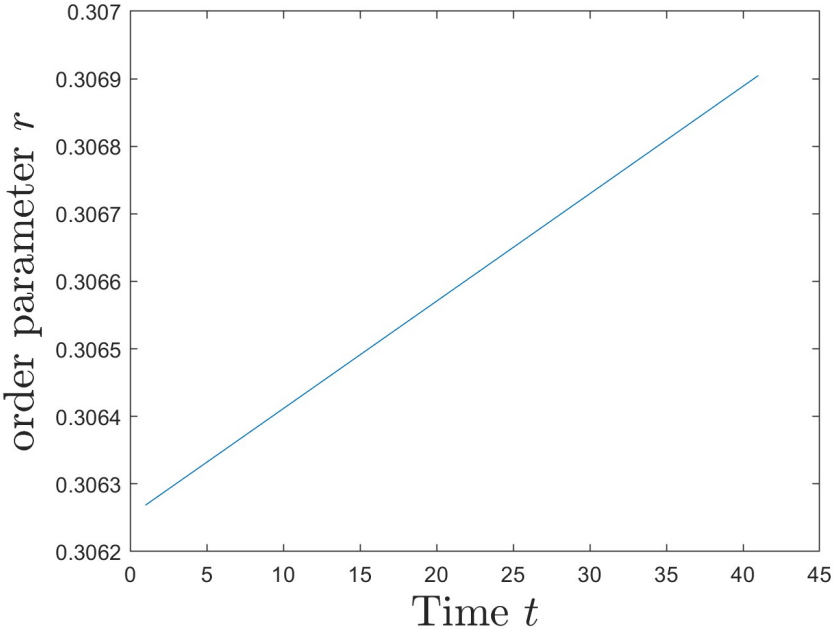

Figure 6.24: Order Parameter r for the Synchronization of the Network of 24 Kuramoto Oscillators when MLC applied to the Control Problem of the system where $\gamma = 0.1$

7 Conclusion and Future work

In thesis, we discussed the two important topics synchronization patterns in the networks of Kuramoto oscillators in Chapter 3 and Machine Learning Control (MLC) in Chapter 4. This Chapter will discuss the conclusion in Section 7.1 for both these topics and MLC applied to the networks of Kuramoto oscillators based on the methods, setup and experimental observations that we discussed in the Chapters 5 and 6, respectively. In Section 7.2, we will mention the possible future explorations in the related topics.

7.1 Conclusion

We introduced Kuramoto oscillators in their raw form discussed in Chapter 3. Through a preliminary understanding of Kuramoto oscillators we were able to show that for a certain constant coupling between all interconnected oscillators there is a value for which all oscillators synchronize – this is known as the critical coupling constant. The critical coupling constant was explored to show that even though Kuramoto oscillators model the phenomenon of natural synchronization in nature, not all oscillators in reality were able to achieve synchronization. Even if modeled oscillators in reality were able to achieve synchronization, there existed external forces that potentially disturbed the system and caused the network oscillators to diverge and lose synchronization due to the fact that the system may be near its bifurcation point. In addition, this thesis explored how the formation of synchronization patterns is characterized in a network of Kuramoto oscillators, and cluster synchronization in networks of Kuramoto oscillators, necessary and sufficient conditions on the interconnection weights of network and on the natural frequencies of oscillators were derived to ensure that the phases of groups of oscillators develop cohesively with each other, yet independently from the phases of oscillators in different clusters. Additionally, a control mechanism developed to modify the edges of a network to ensure the formation of desired synchronization clusters. As the control mechanism finds the smallest perturbation (in the Frobenius norm) for a desired synchronization pattern that is agreeable with a fixed set of structural restrictions, so we observe optimal control mechanism. The powerful methods from machine learning were used to find the controller.

We analysed that MLC technique minimizes a given cost function to generate associated control laws. Genetic Programming is a flexible machine learning algorithm, which we explored to find control laws in MLC. We discussed all this in Chapter 4. In this thesis, we discussed techniques from machine learning and explored how genetic programming is used as an effective method to find control laws. We focused on how Machine Learning Control works and saw an example of implementing genetic programming for the control problem of synchronization in the networks of Kuramoto oscillators.

The main goal of the thesis was to apply MLC to the system of the networks of Kuramoto oscillators to achieve synchronization. We implemented MLC as a control mechanism on the control problem of synchronization in networks of Kuramoto oscillators. The baseline of the thesis was explained in Chapter 5. We setup the experiments used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators to find control law and after applying the GP (used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators to observe the results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized. These experiments with the results based on the methods described in Chapter 5 were discussed in Chapter 6 in detail. We checked the simulations of the networks of Kuramoto oscillators and desynchronized the system of the networks of Kuramoto oscillators by changing the system parameters like the distribution of natural frequencies ω and coupling strength are selected accordingly for the purpose of control so that the uncontrolled system becomes desynchronized as the degree of synchronization is controlled by these parameters. We kept the natural frequencies same and made the coupling strength weak to make the system desynchronized. On this uncontrolled desynchronized system of Kuramoto oscillators, we added control input to the Kuramoto model equation to introduce controller by using used random K for the particular desynchronized system to achieve the synchronization. In this thesis, we also studied that we can use MLC technique to control any system and get the control law. Now, we will use GP for our above desynchronized system to ensure the synchronization of the system. Now after, we used controller by using used random K to control the system to ensure the synchronization of desynchronized system, we used optimal $K(y)$ by using MLC. We discussed the experimental setup used to apply MLC to the desynchronized system of the networks of Kuramoto oscillators, as instead of using controller we can also apply MLC to find the control law. After applying the GP (used for MLC) as a control mechanism for the synchronization of Kuramoto oscillators, we obtained results whether the Kuramoto oscillators are completely synchronized, partially synchronized or not at all synchronized.

In the Experiment I, we developed a directed weighted network of 6 nodes of Kuramoto oscillators, discussed experimental setup and obtained results. After applying MLC to the desynchronized system network of 6 nodes of Kuramoto oscillators, we observed complete synchronization as all the 6 oscillators where $\gamma = 0.1$ were converging to one value when we ran GP. Then, we described the Experiment II of this thesis for a directed network of 12 nodes of Kuramoto oscillators, experimental setup and obtained results. After applying MLC to the desynchronized system network of 12 nodes of Kuramoto oscillators, we observed almost complete synchronization as all the 12 oscillators $\gamma = 0.1$ were almost converging to one value when we ran GP.

We also tried to run GP by changing the value of γ between 0.2 and 1. We analysed the results and found that if γ is greater than 0.2 then value of control remained zero, the reason for this is that the optimization criterion is more interested in controlling cost value rather than our synchronization criterion. So the system was not controlled as a result of it the synchronization was not achieved by the system.

We also ran GP for the large networks after applying MLC as a control mechanism for the desynchronized system of the networks of Kuramoto oscillators. We analysed the results and found that the synchronization was partial among the oscillators of the networks for N between 16 to 20. When N was greater than 20 then the desynchronized system of the

networks of Kuramoto oscillators was not at all synchronized, it remained desynchronized after the application of GP as GP was not capable enough to find out control law for large networks. The value of control also remained zero. We also observed that we require powerful computer for the computation of individuals and evolution of individuals to the next generation as when we ran GP on the desynchronized system of the networks of Kuramoto oscillators for number of oscillators, $N = 100$ to 256, then the code was crashed.

7.2 Future work

In this section, we will discuss the future work and exploration that would continue to expand our learning in these topics. For now, we have just focused on the oscillators which we same properties and observed synchronization of them. We can also form the network of Kuramoto oscillators having different properties as various oscillatory patterns are possible dependent on the interconnection graph $\mathcal{G} = (\nu, \epsilon)$ (where ν is a set of vertices and ϵ is a sets of edges), the oscillator natural frequencies ω and the adjacency matrix $A = [a_{ij}]$. These different patterns can be in partial or cluster synchronization states (as we will define in Section 3.3.2 of Chapter 3). We will specifically focus in the case of cluster synchronization, where phases of groups of oscillators develop cohesively within each group; however, it is independent of the phases of oscillators in distinct groups. For this purpose, we will be making partition of ν (set of vertices).

We will be implementing the formation of synchronization patterns is characterized in a network of Kuramoto oscillators, and cluster synchronization in networks of Kuramoto oscillators as we have already studied necessary and sufficient conditions on the interconnection weights of network and on the natural frequencies of oscillators derived to ensure that the phases of groups of oscillators develop cohesively with each other, yet independently from the phases of oscillators in different clusters. We will setup and implement MLC as a control mechanism on the control problem of cluster synchronization in the desynchronized system of the networks of Kuramoto oscillators to find control law and after applying the GP (used for MLC) as a control mechanism for cluster synchronization of Kuramoto oscillators, we will observe the results whether we achieved complete cluster synchronization, partial cluster synchronization or not at all cluster synchronization of the different clusters of Kuramoto oscillators independent to the group of each other.

Bibliography

[Are+08] Alex Arenas, Albert Dıaz-Guilera, Jurgen Kurths, Yamir Moreno, and Changsong Zhou. 'Synchronization in complex networks'. In: *Physics reports* 469.3 (2008), pages 93–153 (cited on page 8).

[BG02] Bassam Bamieh and Laura Giarre. 'Identification of linear parameter varying models'. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 12.9 (2002), pages 841–853 (cited on page 35).

[Ben+15] Nicolas Benard, Jordi Pons-Prat, Jacques Periaux, Gabriel Bugeda, Jean-Paul Bonnet, and Eric Moreau. 'Multi-input genetic algorithm for experimental optimization of the reattachment downstream of a backward-facing-step with surface plasma actuator'. In: *46th AIAA Plasmadynamics and lasers conference*. 2015, page 2957 (cited on page 33).

[BJ14] DS Beoomhead and Roger Jones. 'Time-series analysis'. In: *Fractals in the Natural Sciences*. Princeton University Press, 2014, pages 103–122 (cited on page 35).

[BB66] Edmund Callis Berkeley and Daniel Gureasko Bobrow. *The programming language LISP: Its operation and applications*. Citeseer, 1966 (cited on page 41).

[BPK13] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 'Compressive sampling and dynamic mode decomposition'. In: *arXiv preprint arXiv:1312.5186* (2013) (cited on page 35).

[DLS12] Thomas Dahms, Judith Lehnert, and Eckehard Schöll. 'Cluster and group synchronization in delay-coupled networks'. In: *Physical Review E* 86.1 (2012), page 016202 (cited on page 18).

[DB14] Florian Dörfler and Francesco Bullo. 'Synchronization in complex networks of phase oscillators: A survey'. In: *Automatica* 50.6 (2014), pages 1539–1564 (cited on pages 9, 10).

[DK97] Dimitris C Dracopoulos and Simon Kent. 'Genetic programming for prediction and control'. In: *Neural Computing & Applications* 6.4 (1997), pages 214–228 (cited on page 33).

[Dra97] Dimitris C. Dracopoulos. 'Evolutionary learning algorithms for neural adaptive control'. In: *Perspectives in neural computing*. Springer-Verlag, London, 1997 (cited on page 33).

[DBN16] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control: Taming Nonlinear Dynamics and Turbulence*. Springer, 2016 (cited on pages 2, 32, 33, 35, 38, 52).

[Dyb09] R Kent Dybvig. *The Scheme programming language*. Mit Press, 2009 (cited on page 41).

[Fav+17] C. Favaretto, D. S. Bassett, A. Cenedese, and F. Pasqualetti. *Bode meets Kuramoto: Synchronized clusters in oscillatory networks*. Seattle, WA, USA: In IEEE American Control Conference, May 2017, pages 2799–2804 (cited on page 18).

[FCP17] C. Favaretto, A. Cenedese, and F. Pasqualetti. *Cluster synchronization in networks of Kuramoto oscillators*. Volume 50(1). In IFAC (International Federation of Automatic Control) PapersOnLine, July 2017, pages 2433–2438 (cited on pages 18, 22).

[FP02] Peter J Fleming and Robin C Purshouse. 'Evolutionary algorithms in control systems engineering: a survey'. In: *Control engineering practice* 10.11 (2002), pages 1223–1241 (cited on page 33).

[GR01] Chris Godsil and Gordon F. Royle. *Algebraic graph theory*. Volume of Graduate Texts in Mathematics. Springer Science & Business Media, 2001 (cited on page 5).

[GMA07] Jesús Gómez-Gardeñes, Yamir Moreno, and Alex Arenas. 'Synchronizability determined by coupling strengths and topology on complex networks'. In: *Physical Review E* 75.6 (2007), page 066106 (cited on page 18).

[HJ85] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985 (cited on pages 5, 7).

[Jua94] Jer-Nan Juang. *Applied system identification*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1994 (cited on page 35).

[Kur75] Y. Kuramoto. *Self-entrainment of a population of coupled non-linear oscillators. In H. Araki, editor, International Symposium on Mathematical Problems in Theoretical Physics*. Volume 39 of lecture notes in physics. Berlin, Heidelberg: Springer, 1975, pages 420–422 (cited on page 1).

[KA75] Y. Kuramoto and H. Araki. *Lecture Note in Physics, International Symposium on Mathematical Problems in Theoretical Physics*. New York: Springer, 1975 (cited on pages 9, 10).

[LM13] Yueheng Lan and Igor Mezić. 'Linearization in the large of nonlinear systems and Koopman operator spectrum'. In: *Physica D: Nonlinear Phenomena* 242.1 (2013), pages 42–53 (cited on page 35).

[Lee+97] Changhoon Lee, John Kim, David Babcock, and Rodney Goodman. 'Application of neural networks to turbulence control for drag reduction'. In: *Physics of Fluids* 9.6 (1997), pages 1740–1747 (cited on page 33).

[LLC10] Wenlian Lu, Bo Liu, and Tianping Chen. *Cluster synchronization in networks of coupled nonidentical dynamical systems*. Volume 20(1): 013120. Chaos: An Interdisciplinary Journal of Nonlinear Science, 2010 (cited on page 18).

[MAS05] János Madár, János Abonyi, and Ferenc Szeifert. 'Genetic programming for the identification of nonlinear input- output models'. In: *Industrial & engineering chemistry research* 44.9 (2005), pages 3178–3186 (cited on page 36).

[Mun10] Joakim Munkhammar. *Simulation of Kuramoto's model*. 2010. URL: `http://www.collective-behavior.com/Simulations/Ch6BoxA.m` (cited on page 12).

[Pec+14] Louis M Pecora, Francesco Sorrentino, Aaron M Hagerstrom, Thomas E Murphy, and Rajarshi Roy. 'Cluster synchronization and isolated desynchronization in complex networks with symmetries'. In: *Nature communications* 5.1 (2014), pages 1–8 (cited on page 18).

[PRK03] Arkady Pikovsky, Michael Rosenblum, and Jürgen Kurths. 'Synchronization: A universal concept in nonlinear sciences'. In: (2003) (cited on page 8).

[PBK16] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. 'Dynamic mode decomposition with control'. In: *SIAM Journal on Applied Dynamical Systems* 15.1 (2016), pages 142–161 (cited on page 35).

[Rec73] Ingo Rechenberg. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. 15. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973 (cited on page 33).

[Sch+16] M. T. Schaub, N. O'Clery, Y. N. Billeh, J. Delvenne, R. Lambiotte, and M. Barahona. *Graph partitions and cluster synchronization in networks of oscillators*. Volume 26(9): 094821. Chaos, 2016 (cited on page 18).

[Sch65] H-P Schwefel. 'Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik'. In: *Diploma thesis, Technical University of Berlin* (1965) (cited on page 33).

[SA91] Jeff S Shamma and Michael Athans. 'Guaranteed properties of gain scheduled control for linear parameter-varying plants'. In: *Automatica* 27.3 (1991), pages 559–564 (cited on page 35).

[Sor+16] Francesco Sorrentino, Louis M Pecora, Aaron M Hagerstrom, Thomas E Murphy, and Rajarshi Roy. 'Complete characterization of the stability of cluster synchronization in complex dynamical networks'. In: *Science advances* 2.4 (2016), e1501737 (cited on page 18).

[Str00] Steven H. Strogatz. 'From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators'. In: *Physica D: Nonlinear Phenomena* 143.1-4 (2000), pages 1–20 (cited on page 9).

[Tib+17] Lorenzo Tiberi, Chiara Favaretto, Mario Innocenti, Danielle S Bassett, and Fabio Pasqualetti. 'Synchronization patterns in networks of Kuramoto oscillators: A geometric approach for analysis and control'. In: *IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. Melbourne, Australia, 2017, pages 481–486 (cited on pages 1, 18–20, 22, 23, 57).

[Wah08] Mattias Wahde. *Biologically inspired optimization methods: an introduction*. WIT press, 2008 (cited on page 33).

[Win67] Arthur T. Winfree. 'Biological rhythms and the behavior of populations of coupled oscillators'. In: *Journal of Theoretical Biology* 16.1 (1967), pages 15–42. URL: `https://www.sciencedirect.com/science/article/pii/0022519367900513` (cited on page 9).

[Wu+08] Xindong Wu et al. 'Top 10 algorithms in data mining'. In: *Knowledge and information systems* 14.1 (2008), pages 1–37 (cited on page 33).

[ZSL14] Zhifei Zhang, Alain Sarlette, and Zhihao Ling. 'Synchronization of Kuramoto oscillators with non-identical natural frequencies: a quantum dynamical decoupling approach'. In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pages 4585–4590 (cited on page 18).

A Code for the Weighted Network of Kuramoto Oscillators without Control input

```
%% Code for the weighted network of 6 nodes of Kuramoto
   oscillators with Kuramoto equation solutions showing
   simulations of the network.

clear all
clc

%% Network
N = 6; %% no. of nodes
s = [1 2 2 3 4 5 6 6 6];
t = [6 4 6 5 1 2 2 3 4];
weights = [10 5 5 10 9 9 7 2 2];
names = {'1','2','3','4','5','6'};
G = digraph(s,t,weights,names); %% directed strongly connected
   graph
A = adjacency(G,'weighted');
A = full(A); %% weighted adjacency matrix

%% Network
%N = 12; %% no. of nodes
%A=[ones(N/2) ones(N/2);ones(N/2) ones(N/2)] - eye(N); %%
   weighted adjacency matrix
%G = digraph(A); %% directed strongly connected graph

%% integration parameters
dt = 0.01;
maximum_step = 300/dt; %1000/dt

%% parameters
omega = ones(N,1)*1/100; % natural frequency parameters

%% initialization
theta = zeros (N, maximum_step); % phase initial state
theta(:,1) = 2* pi* rand (N,1);
dthetadt = zeros (N,1);
```

```
32
33 for t = 1:maximum_step
34
35     %% Main Integration
36     coup = zeros (N,1);
37
38     for i = 1:N
39         coup (i) = sum(A(:,i).* sin(theta(:,t) - theta(i,t)));
40         %coup (i) = 0.001*sum(A(:,i).* sin(theta(:,t) - theta(i,
               t))); % make weak coupling for desynchronization
41         dthetadt (i) = omega(i) + 1/N*coup (i);
42         theta (i,t+1) = mod(theta(i,t) + dt*(dthetadt(i)), 2* pi
               ) ;
43     end
44 end
45
46 %% order parameter
47 r = abs( mean ( exp(j*(theta(1:N,:))) ,1));
48
49
50 %% plots
51 figure (1), plot (r)
52 xlabel('Time $t$','interpreter','latex','fontsize',20)
53 ylabel('order parameter $r$','interpreter','latex','fontsize'
       ,20)
54
55 figure(2), for i = 1:N
56             plot(sin(theta(i,:))); hold on;
57             xlabel('Time $t$','interpreter','latex','fontsize'
                   ,20')
58             ylabel('$\sin(\theta_{i})$','interpreter','latex','
                   fontsize',20)
59             end
```

B Code for the Weighted Network of Kuramoto Oscillators with Control input to ensure the synchronization of desynchronized oscillators.

```
%% Code for the weighted network of 6 nodes of Kuramoto
   oscillators with Kuramoto equation solutions showing
   simulations of the network having control input to ensure the
    synchronization of desynchronized oscillators.

clear all
clc

%% Network
N = 6; %% no. of nodes
s = [1 2 2 3 4 5 6 6 6];
t = [6 4 6 5 1 2 2 3 4];
weights = [10 5 5 10 9 9 7 2 2];
names = {'1','2','3','4','5','6'};
G = digraph(s,t,weights,names); %% directed strongly connected
   graph
A = adjacency(G,'weighted');
A = full(A); %% weighted adjacency matrix

%% Network
%N = 12; %% no. of nodes
%A=[ones(N/2) ones(N/2);ones(N/2) ones(N/2)] - eye(N); %%
   weighted adjacency matrix
%G = digraph(A); %% directed strongly connected graph

%% integration parameters
dt = 0.01;
max_step = 300/dt; %1000/dt

%% parameters
omega = ones(N,1)*1/100; % natural frequency parameters

%% initialization
```

```
29 theta = zeros (N, max_step); % phase initial state
30 theta(:,1) = 2* pi* rand (N,1);
31 dthetadt = zeros (N,1);
32
33
34 %% control
35 for t = 1:max_step
36
37     K = rand(N,N); %% control input
38     %K = zeros(N,N); %% zero control input to check without
          control mechanism
39     F = A.*K;
40
41     %% Main Integration
42     coup = zeros (N,1);
43
44    for i = 1:N
45         coup (i) = 0.001*sum(A(i,:)'.* sin(theta(:,t) - theta(i,
              t)));
46         control (i) = sum(F(i,:)'.* sin(theta(:,t) - theta(i,t))
              );
47         dthetadt (i) = omega(i) + 1/N*coup (i) + 1/N*control (i)
              ;
48         theta (i,t+1) = mod(theta(i,t) + dt*(dthetadt(i)), 2* pi
              ) ;
49     end
50 end
51
52 %% order parameter
53 r = abs( mean ( exp(j*(theta(1:N,:))) ,1));
54
55 %% plots
56 figure (1), plot (r)
57 xlabel('Time $t$','interpreter','latex','fontsize',20)
58 ylabel('order parameter $r$','interpreter','latex','fontsize'
       ,20)
59
60 figure(2), for i = 1:N
61            plot(sin(theta(i,:))); hold on;
62            xlabel('Time $t$','interpreter','latex','fontsize'
                  ,20')
63            ylabel('$\sin(\theta_{i})$','interpreter','latex','
                  fontsize',20)
64            end
```

C Code for GP applied to the Weighted Network of Kuramoto Oscillators with Control input to ensure the synchronization of desynchronized oscillators.

```
%% %% Code for GP applied to the Weighted Network of Kuramoto
   Oscillators.

function J=unidim_DS_evaluator(ind,mlc_parameters,i,fig)
%% Obtaining parameters from MLC object.
Tf=mlc_parameters.problem_variables.Tf;
objective=mlc_parameters.problem_variables.objective;
gamma=mlc_parameters.problem_variables.gamma;
Tevmax=mlc_parameters.problem_variables.Tevmax;

N=mlc_parameters.problem_variables.N;
A=mlc_parameters.problem_variables.A;
omega=mlc_parameters.problem_variables.omega;
kappa=mlc_parameters.problem_variables.kappa;
B=mlc_parameters.problem_variables.B;

%% Interpret individual.
m=readmylisp_to_formal_MLC(ind);
m=strrep(m,'S0','y');
K=@(y)(1/N*sum(B.*sin(y-y'))'); % K=@(y)(1/N*sum(A.*sin(y-y'))')
   ; when matrix A is not weighted then matrix B is not required
   .
eval(['K=@(y)(' m ');']);
f=@(t,y) (omega + kappa/N*sum(A.*sin(y-y'))' + K(y) + testt(toc,
   Tevmax));


%% Evaluation
try                         % Encapsulation in try/catch.
tic
[T,Y]=ode45(f,[0 Tf],rand(N,1)*2*pi);  % Integration.
```

```
28 if T(end)==Tf                % Check if Tf is reached.
29     b=Y*0+K(Y);              % Computes b.
30     Jt=1/Tf*cumtrapz(T,(abs( mean ( exp(j*(Y)),2))-objective).^2
           + gamma*sum(b.^2,2)); % Computes J. Mean r value should
          be one, objective = 1
31     J=Jt(end);
32 else
33     J=mlc_parameters.badvalue;  % Return high value if Tf is not
           reached.
34 end
35 catch err
36    %J=mlc_parameters.badvalue % Return high value if ode45 fails
          .
37    J = 1e+33
38 end
39
40 if nargin>3   % If a fourth argument is provided, plot the
      result
41     subplot(3,1,1)
42     plot(T,Y,'-k','linewidth',1.2)
43     ylabel('$\theta_{i}$','interpreter','latex','fontsize',20)
44     subplot(3,1,2)
45     plot(T,b,'-k','linewidth',1.2)
46     ylabel('$u$','interpreter','latex','fontsize',20)
47     subplot(3,1,3)
48     plot(T,Jt,'-k','linewidth',1.2)
49     ylabel('$(r-1)^2+\gamma u^2$','interpreter','latex','
          fontsize',20)
50     xlabel('Time $t$','interpreter','latex','fontsize',20)
51
52
53     figure (2), plot (abs( mean ( exp(j*(Y)),2)))
54     xlabel('Time $t$','interpreter','latex','fontsize',20)
55     ylabel('order parameter $r$','interpreter','latex','fontsize
          ',20)
56
57     figure(3), for i = 1:N
58                plot(sin(Y)); hold on;
59                end
60                xlabel('Time $t$','interpreter','latex','fontsize
                      ',20)
61                ylabel('$\sin(\theta_{i})$','interpreter','latex'
                      ,'fontsize',20)
62 end
```

D USB-stick Content

The USB-stick content is submitted in the form of zip file to Prof. Dr Fabian Wirth by an email containing:

- This work in PDF file – in the folder *PDF*
- The source code of the implementation – in the folder *CODE*
- The LaTeXsource code – in the folder *LATEX*

Declaration of Academic Integrity / Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind. Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten vom 31. Juli 2008 (vABlUP Seite 283) bin ich vertraut. Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

Passau, 22. März 2022

_____

Firstname Lastname

I hereby confirm that I have composed this scientific work independently without anybody else's assistance and utilising no sources or resources other than those specified. I certify that any content adopted literally or in substance has been properly identified. I have familiarised myself with the University of Passau's most recent Guidelines for Good Scientific Practice and Scientific Misconduct Ramifications from 31 July 2008 (vABlUP Seite 283). I declare my consent to the use of third-party services (e.g., anti-plagiarism software) for the examination of my work to verify the absence of impermissible representation of adopted content without adequate designation violating the intellectual property rights of others by claiming ownership of somebody else's work, scientific findings, hypotheses, teachings or research approaches.

Passau, 22. März 2022

_____

Firstname Lastname