A STUDY OF
SYNCHRONISATION IN COMPLEX NETWORKS

A PROJECT REPORT

submitted by

SHIVAPRASAD V

for the partial fulfilment of the degree

of

MASTER OF SCIENCE
IN PHYSICS

DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

CERTIFICATE

This is to certify that the report titled **A STUDY OF SYNCHRONISATION IN COMPLEX NETWORKS**, submitted by **Shivaprasad V**, to the Indian Institute of Technology Madras, for the award of the degree of **Master of Science in Physics**, is a bona fide record of the project work done by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

The project work has been carried out at IIT Madras.

Place : Chennai
Date : 28th June 2020

Dr Neelima M. Gupte
Professor
Dept. of Physics
IIT Madras, 600036

ACKNOWLEDGEMENTS

First and foremost, I sincerely thank my project guide Dr Neelima M. Gupte for her initiation, advice and guidance throughout the course of the project. Her directions prompted me to explore the problem flexibly with an open mind and an inquisitive spirit.

I express my gratitude to Ms Malayaja Chutani, Research Scholar, Dynamical Systems Research Group, Dept. of Physics, IIT Madras for the kind help and support she provided while overseeing the project work. Her instructions, assessment and feedback at every step were immensely valuable and is appreciated.

I thank Mr Rabindev Bishal, Research Scholar, Dynamical Systems Research Group, Dept. of Physics for the technical help he provided during the course of the project.

I would also like to thank other Research Scholars from the Dynamical Systems Research Group for their cooperation in presentations and seminars. I sincerely thank my friends and fellows from MSc Physics, IIT Madras for their tips and feedback to the project. I also thank my faculty advisor Dr C. V. Krishnamurthy for his empathy and kind words while discussing my choice of the project.

Finally, I am grateful to all those who had a direct or an indirect contribution to the successful completion of my project.

ABSTRACT

KEYWORDS: Synchronisation; Complex networks; Simplicial complexes; Non-linear dynamics; Computation.

Synchronisation is a field of study in non-linear dynamics that focus on harmony and rhythm found in the Universe. The modern theory of complex networks has helped to sharpen our understanding of synchronisation processes in populations of locally interacting elements. The current focus on non-linearities added by higher-order interactions encoded in simplicial complexes promises to take the research in the field to new directions.

In this project, we study the phenomenon of synchronisation of a large population of phase-coupled oscillators constrained to interact via a complex network topology by computation.

TABLE OF CONTENTS

Table of contents (continued)

LIST OF FIGURES

Chapter 4:

List of figures (continued)

CHAPTER 1

INTRODUCTION

The advent of computers revolutionised the field of study of Dynamical systems. Ever since Edward Lorenz "accidentally" discovered Chaos in 1963 while simulating weather systems on a Royal McBee LGP-30 digital computer[1], many more breakthroughs have been made in this branch of mathematical physics, and also in its several sub-branches. Most of the studies focussed on the exciting developments in Chaos theory; the seeming randomness, unpredictability in spite being completely deterministic, visually beautiful strange attractors and many more fascinating features gave an ethereal attraction to the phenomenon. The quest has now reached Quantum Chaos.

During the same era, silent work was going on in a phenomenon pole opposite to the disarray and frenzy of Chaos, which was one of deep, spontaneous order emerging out of the din of disorder, that of synchronisation. Synchronisation means that two things happen simultaneously. Even though many synchronous phenomena are rhythmic, they need not be periodic; they can even be chaotic.[2] Some examples found in nature are the neurons in the brain and pacemaker cells in the heart firing together, swarms of fireflies flashing and crickets chirping in tandem. The idea of synchronisation as a dynamic phenomenon goes as far back as 1665 when "C. Huygens discovered an odd 'kind of sympathy' in two pendulum clocks suspended side by side".[3] Modern study on the subject began around the same time as Lorenz's Chaos; starting in the 1960s and 1970s, the work in the field advanced through pioneers like Wiener, Winfree, Kuramoto, Peskin, Josephson and Strogatz.

Synchronisation mostly involves oscillators. The Harmonic oscillator has become the bedrock of much of Physics ranging from the simple pendulum to Quantum Optics. Nevertheless, it is a purely linear, non-chaotic system, a highly idealised abstraction seldom found in nature. The simple pendulum can become chaotic when driven; coupling two such pendula by say, a spring, gives us the simplest example where synchronisation can be seen, a version of which Huygens observed. The interaction through the spring forces the oscillators to fall in step and follow the lead of each other. Synchronisation involves at least two such elements in interaction, and the dynamics of a few interacting oscillators has been studied and well established before. However, the phenomenon of synchronisation of large populations is a different problem that requires a different model, with a different approach altogether.[3]

Kuramoto introduced a model based on the premise that all oscillators interact with everyone else in the collection.[4] While this mathematically tractable model was undoubtedly a breakthrough, such all-to-all connectivity is infeasible in large-population systems, as it becomes hard to maintain due to physical constraints in the real world.[3] What we usually find in such populations is specific linkages, forming networks of nodes. It was Watts and Strogatz that first presented a simple model of network structure for the interaction, which eventually became the seed for the modern theory of complex networks.[5] A complex network, as opposed to a lattice, has an inherent lack of symmetry, isotropy and homogeneity, which makes a direct mathematical analysis difficult and in most cases, there is a necessity of numerical computations and simulations. Once provided with the physics of the basic constituents and their interactions, computers can simulate a complicated model and help us visualise and analyse the dynamical behaviour, and also in subsequent mathematical treatment.

Finally, with the increasing complexity and scale of the system that we intend to replicate in a computer, there is an increasing need for speed and efficiency of the programming techniques used. Striving to minimise the

utilisation of computational resources is as important in a computational project as the achievement of results. Care needs to be given to this end, starting from the choice of the programming language used, the program design and the algorithms or libraries employed.

Synchronisation is being pursued with great zeal by researches in a wide variety of fields, as it is a truly multi-disciplinary subject. It also has an aesthetic appeal that captures the human imagination. It is ubiquitous in the cosmos and nature. It has also found useful applications in many different fields including, but not limited to, biology, ecology, climatology, sociology and technology.[3]

CHAPTER 2

BACKGROUND

Many excellent reviews exist that summarises the developments in the field of synchronisation in complex networks so far. We base our study on the review by Alex Arenas et al., 2008[3], which is cited extensively in this section.

2.1 The Kuramoto Model

Many real-world systems that display self-organisation by synchronisation are collections of interacting elements with a periodic, mostly oscillatory, action. A good example would be the pacemaker cells in the sinoatrial node of the heart that sets the rhythm of the heartbeat. We model them as collections of coupled phase-oscillators, called so because we neglect the amplitude and focus only on the phase of oscillation which is paramount to synchronisation. The Kuramoto model puts forward an all-to-all, purely sinusoidal coupling with uniform strength across the collection, giving rise to the governing equation:

$$\dot{\theta}_i = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin\big(\theta_j - \theta_i\big) \;\; (i = 1, 2, ..., N) \tag{1}$$

where θ_i and ω_i are the phase and natural frequency of the i^{th} oscillator respectively, K is the coupling constant, and the overdot stands for time derivative. The $1/N$ factor is to ensure convergence in the thermodynamics limit ($N \to \infty$). The frequencies ω_i need not be the same for all i, as oscillators in a natural collection are never identical; they are usually distributed according to some function g(ω) symmetric about its mean frequency. Due to the rotational symmetry of the model, we can use a rotating frame and hence set

the mean frequency to zero without any loss of generality. We can also always observe the dynamics from a frame co-rotating with the phase of any of the oscillators.

The presence of the sine term in the interaction makes it a non-linear model. It is this non-linearity behind the self-organising properties of the model, such as synchronisation.

2.2 Measuring synchronisation

The synchronisation in the system is quantified by the average of the complex phases ($e^{i\theta_j}$), called the *macroscopic complex order parameter*,

$$\tilde{r}(t) = |r(t)|e^{i\phi(t)} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j(t)} \tag{2}$$

where the modulus $|r(t)|$ gives the level of synchronisation in the system and $\phi(t)$ is the average phase. If the N oscillators are nearly in phase, the complex phases add up on the RHS of (2) resulting in $|r| \approx 1$, signifying synchronisation. On the other hand, in a desynchronised state the phases are at random, with $|r| \approx 0$ as the complex phases cancel out.

A rigorous mathematical treatment, following the mean-field approach, shows that this model shows a dynamic analogue of an equilibrium phase transition.[3] It can also be demonstrated by computing $|r|$ (by using Eq. (2)) and plotting it as a function of K.

2.3 Complex networks

As already mentioned in the introduction, the all-to-all connectivity of the Kuramoto model is replaced by a complex-network in more realistic models. Mathematically, a complex network is a graph of N nodes, interconnected via M links. It is usually represented using a matrix $\mathbb{A}$, called the Adjacency matrix, whose element $a_{ij} = 1$ if a directed link exists from node j to node i,

and zero otherwise. In the case of a weighted network, the weights σ_{ij} that represent the strength of the link a_{ij} can be entries of a separate matrix W, or be incorporated in the matrix A itself. Other representations exist, which could be favourable from a programming perspective (see Ref. [6]).

If the adjacency matrix A is symmetric, i.e. $a_{ij} = a_{ji} \forall i, j$, a link from j to i implies that a link from i to j also exists, and the network is called an *Undirected network*. However, if A is not symmetric, then such reciprocating relationships do not exist between the nodes, and the network is called *Directed*. Even though the links are the same in both cases, the directional property makes them very different topologies, especially in the context where they are applied to the Kuramoto model. In this case, a link directed from j to i denotes the influence of j on the dynamics of i, which is physically distinct from the reverse case. Consequently, we would study them separately.

The number of links a node i has is called its *degree* k_i. The probability of a node in the network to have a degree k, denoted as $P(k)$, is called the *degree distribution* of the network. It is one of the characterising properties of a complex network and is used to classify them. In the case of a directed network, each node has a distinct in-degree and out-degree, which is the number of incoming and outgoing links of the node respectively. Thus, in a directed network, the in-degree defines the number of nodes influencing the receiver, while the out-degree denotes the amount influence the node has on others.

2.4 Extended Kuramoto model

With the definition of complex networks in place, we can extend the Kuramoto model to incorporate a connectivity scheme. We need to reformulate Eq. (1) by replacing the universal coupling constant K with a connectivity matrix as:

$$\dot{\theta}_i = \omega_i + \sum_{j=1}^{N} \sigma_{ij} a_{ij} \sin\big(\theta_j - \theta_i\big) \quad (i = 1, 2, ..., N) \tag{3}$$

where σ_{ij} and a_{ij} are as defined in Sec. 2.3. The original Kuramoto model can be considered as a special case of the more general Eq. (3), with $a_{ij} = 1 \ \forall \ i \neq j$ and $\sigma_{ij} = K / N \ \forall \ i, j$. The order parameter (Eq. (2)), being the measure of synchronisation regardless of the connectivity, need not be redefined. We would be using only equations (2) and (3) in all the simulations performed.

Note that we can still have a phase transition by varying the coupling strength; however, since the weights σ_{ij} need to remain in the same ratio, all of them should be varied by the same factor at a time.

2.5 Beyond pairwise interaction

Recently there has been a focus on extending the Kuramoto model further, by incorporating higher-order interactions between the oscillators, e.g. three- or four-way interactions, in addition to the pairwise interaction given by Eq.(3). They add non-linearities that are shown to significantly alter the dynamics, resulting in novel properties like abrupt synchronisation switching.[7] Such interactions are encoded in higher-order structures called simplicial complexes present in a network.[8]

A *simplex* (also called a *clique*) is a complete subgraph within a larger graph; i.e., a subset of the nodes of a complex network with all-to-all connectivity. It can be considered as "a generalisation of the notion of a triangle or tetrahedron to arbitrary dimensions".[9] A triangle is called a 2-simplex, a tetrahedron is a 3-simplex, and so on. A clique which does not exist within the vertex set a larger clique (e.g. a triangle that is not one of the faces of a tetrahedron) is called a *maximal clique*. A structure constituted by simplexes (that may be of different dimensions) is called a *simplicial complex*.

The cliques of a simplicial complex can facilitate higher-order interaction terms as they bring together more than two nodes, all connected to each other. E.g. a network with 2-simplexes (triangles) formed by at least some of the pairwise links (edges) can support the following extension to the Kuramoto model:

$$\dot{\theta}_i = \dot{\theta}_i^K + \sum_j \sum_k \gamma_{ijk} b_{ijk} \sin\big(2\theta_j - \theta_k - \theta_i\big) \quad (i = 1, \ldots, N) \tag{4}$$

where $\dot{\theta}_i^K$ is the pairwise Kuramoto term (RHS of Eq. (3)) and the second term denotes a higher-order interaction, as it contains a sinusoidal coupling that involves the phase of three oscillators (i, j and k) simultaneously. Here b_{ijk} is called a 2-simplex adjacency tensor, analogous to the adjacency matrix defined in Sec. 2.3, and γ_{ijk} is the weight of the corresponding coupling. If all the elements of these tensors are set to zero, we retrieve our original network-Kuramoto model.

We would not study the dynamics of such higher-order terms here, as we are yet to introduce such terms in the model under consideration. Still, it would be instructive to look out for cliques in the network as they can be the basis of such interactions, if they were to be introduced in future. Given the edges, the maximal cliques in an undirected network can be found out using the Bron-Kerbosch (BK) algorithm.[10] In case of the directed network, the direction of edges an be assigned after the cliques are found out from the corresponding undirected network.

2.6 Programming decisions

In order to ensure efficiency, the author decided to simulate the model by writing the programs instead of using any software packages. The C programming language was chosen for its speed and performance, and familiarity as well. Mathematically, the nodes are represented using natural numbers, which we term as their index (or id). The same notation was adopted in the programs. The ids were also used to index the double-precision floating-

point arrays holding the phase values (after subtracting one, as C arrays are indexed from zero). Linked lists, a commonly used data structure in C programs, were utilised extensively to store, access and manipulate the network, its nodes and the cliques.

We avoided using external libraries and wrote all the code from scratch so that anyone can execute them without worrying about dependencies. A custom implementation (based on [11]) of the N-dimensional 4th order Runge-Kutta algorithm was used to integrate the governing equation (3) in all the simulations. A custom implementation of the BK algorithm was also written from scratch. The relevant portions of the code are attached in the Appendix; the full version is available on GitHub (see Ref. [12]).

CHAPTER 3

OBJECTIVES

We choose a specific, new, artificially generated network as the basis of our exploration. The author did not generate it; it was obtained from an external source.[13] We purposefully refrain from knowing more details about the network, including the method of generation, from its source. By doing so, we avoid the pitfall of jumping into conclusions or having preconceived notions about the network's dynamics. We aim to explore the network freely, without any prejudices or biases that might creep in from knowing the background of the network. The only information we have about the network, apart from the fact that it is artificially generated, is the name of the file listing its edges: `dclnet5_ni5_p07.edgelist` . It could be a code name; for our convenience, we shall call it *dclnet* or even merely *the network* henceforth.

The central aim of the project is to study the dynamic behaviour of phase-coupled Kuramoto oscillators, interconnected and interacting via the complex-network topology of *dclnet*, focussing on the emergent phenomenon of synchronisation, through numerical simulations and computations.

The broad objectives of our study can be summarised in the following five points:

1. To characterise the static properties of the network and identify the main topological features;
2. To simulate the dynamics of the oscillators placed at its nodes and study their collective behaviour;
3. To look for patterns in the resulting dynamic fingerprint, including that of Chaos;

4. To infer the results and to try and relate them with existing Physics of similar systems; and
5. To extrapolate the results methodologically and suggest ways forward for further exploration.

With these longterm targets in mind, we adopted a strategy of pursuing smaller, short-term goals, periodically reviewing the progress. After completing each goal, the results were analysed and discussed, taking hints to evolve the strategy forward and set new goals. The results of this step-by-step exploration are reported here in more or less the chronological order in which they were obtained.

Before starting on the objectives, the concepts and phenomenology were gathered by an extensive literature survey, the relevant parts of which were summarised in Sec. 2. We then moved on to perform the simulations.

CHAPTER 4

COMPUTATIONS

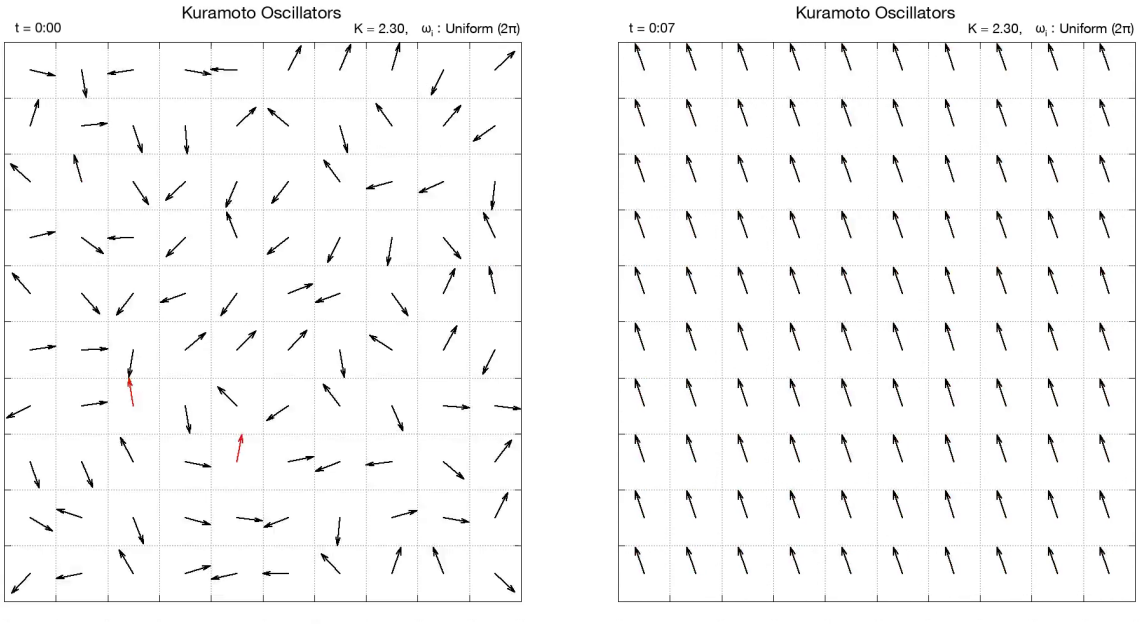
4.1 PRELIMINARIES

The network under consideration is vast and complex, with 1000 nodes and 3981 links. It would be unwise to attempt to start directly simulating a system of such a massive scale. It is a common practice in programming to first start with a smaller, manageable version of the problem which is easier to test and analyse. After making sure the code runs correctly, it can be scaled up to handle the actual complex problem. More importantly, this helps us develop a robust computational framework to build upon which becomes the groundwork for the simulations later on. From a physical point of view, this approach enables us to get over the novelty of the phenomenon and establish familiarity with the basic concepts involved. It gives us a flair for the subject.

4.1.1 Simulating the Kuramoto Model

The first thing to attempt was simulating the generic, all-to-all Kuramoto model on a small grid of phase oscillators. The evolution was qualitatively studied by visualising the oscillators using a rotating arrow to represent the phase of each oscillator. The phases at each instant are obtained by simultaneously integrating the governing equations (3) using an N-dimensional 4th order Runge-Kutta (RK4) algorithm (see Appendix). A plot of the phases (using arrows) at a given time provides a snapshot, and a series of such snapshots are combined to obtain a video. The initial and final frames of the first simulation are given in Fig. 4.1. In this case, for simplicity, all oscillators were given the same natural frequency of 2π. **Note:** the oscillators

are placed on a grid only for the convenience of plotting; it does not pertain to any physical 2D planar arrangement.

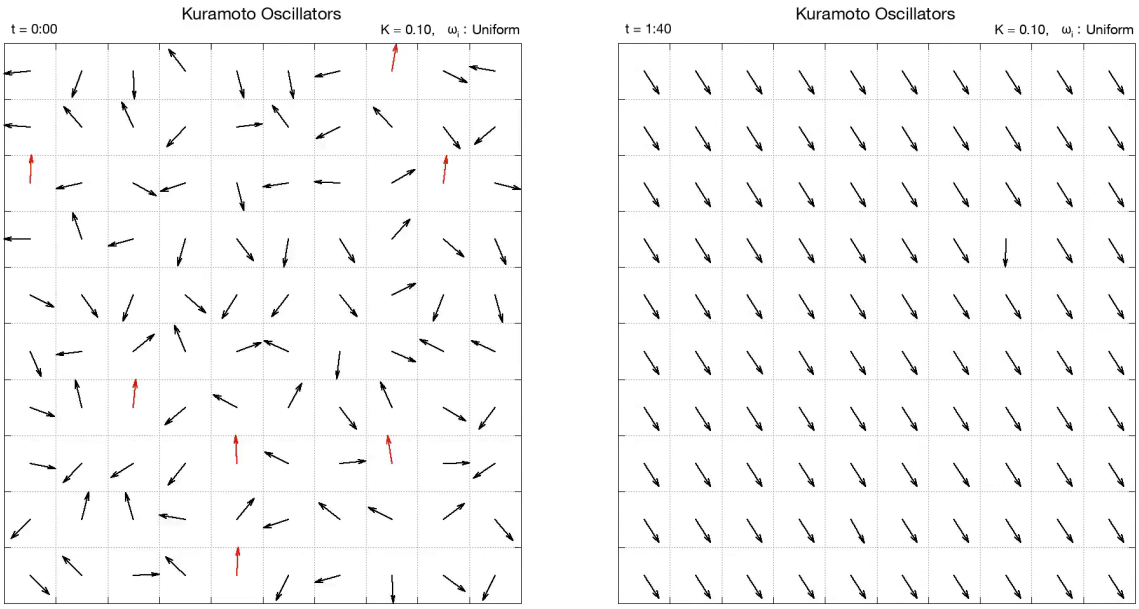


(a) (b)

Figure 4.1: All-to-all Kuramoto model simulation. (a) first frame (at t = 0s) and (b) last frame (at t = 7s) of the video simulation.

At the beginning of the simulation, each oscillator is given a random phase drawn from a uniform distribution $\mathbb{U}: (0, 2\pi)$, which is displayed by the random direction of arrows in the initial state (Fig. 4.1 (a)). Within a few seconds after starting the simulation, owing to the mutual interactions, all the oscillators tune in to the same phase resulting in a final state of perfect synchronisation (Fig. 4.1 (b)). It was also observed that such a configuration leads to synchronisation even for a small coupling strength ($K = 0.1$), given enough time (c.f. Fig. 4.2).

These conditions are but ideal, as, in a real natural system, no two such oscillators will be perfectly identical. There is always a small spread in the natural frequencies of a population of oscillators, distributed according to some function g(ω), typically chosen to be a Gaussian ($\mathbb{G}(\mu, \sigma)$).



(a) (b)

Figure 4.2: Kuramoto model with small K. (a) first frame (at t = 0s); (b) synchronisation is achieved only after a long time (t = 1m 40s).

4.1.2 Phase transition in the Kuramoto Model

To quantify the level of synchronisation in the population, we need to calculate the macroscopic (global) order parameter (Eq. (2)) and take its modulus $|r|$. It is usually a quantity that fluctuates with time, so a time-averaged value is taken after the initial transients die out and the system gets synchronised. The time-averaged $|r|$ is computed for various values of the coupling strength K ranging from 0 to 1. The resulting plot of $\langle|r|\rangle$ v/s K is a phase-transition diagram, as shown in Fig. 4.3 for different grid sizes and measurement times.

More simulations were carried out, experimenting with the effects of using different distributions for the initial phases ($\theta_i(0)$) and oscillator frequencies (ω_i), as well as changing the number of oscillators. Their results are not included here for brevity.

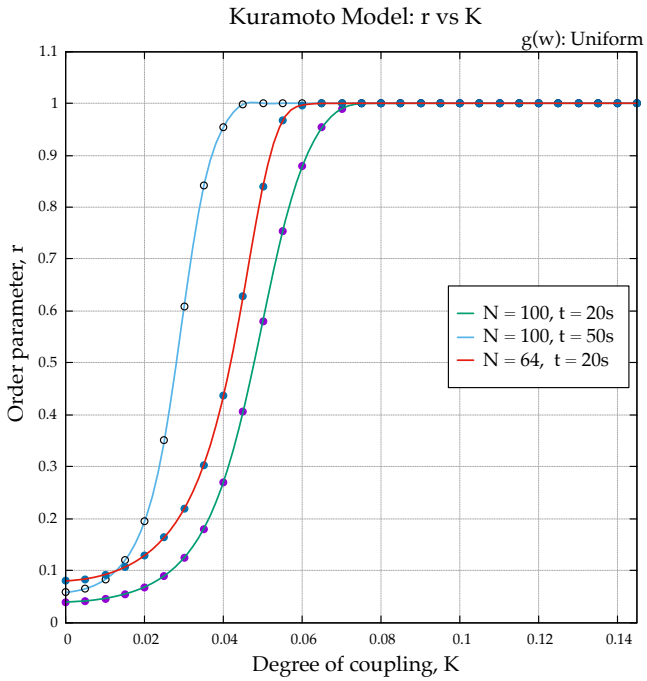


Figure 4.3: Phase transition in the Kuramoto model.

4.1.3 Introducing the Complex Networks

The next part of the preliminary work involved implementing the complex networks. One of the earliest design choices to make is the type of data structure to use for storing and manipulating such networks. Several different techniques exist[6], each with its own merits and demerits. After studying them all, Adjacency Lists were found to be most suitable for our purpose. It involves storing all the adjacent nodes ("neighbours") of a given node as an entry in a lookup table, which drastically reduces the loop required to execute the summation in Eq. (3). For every node, we can directly iterate over the adjacent nodes, instead of looping over all nodes in the network.

To study complex networks of smaller scale, we need to generate them in the first place. One of the simplest of all is the Watts-Strogatz (WS) network, which can be generated using a simple algorithm.[5][14] After developing the necessary data structures, a WS network with 50 nodes was generated, with a phase oscillator placed at each of its nodes. Once again, simulations were

focussed on visualising the time evolution of the phases using arrows. This time, when the oscillators complete a cycle (chosen as when the arrows point straight up) they were also made to flash a different colour, much like fireflies in a swarm flash their light periodically, in response to their internal clocks.

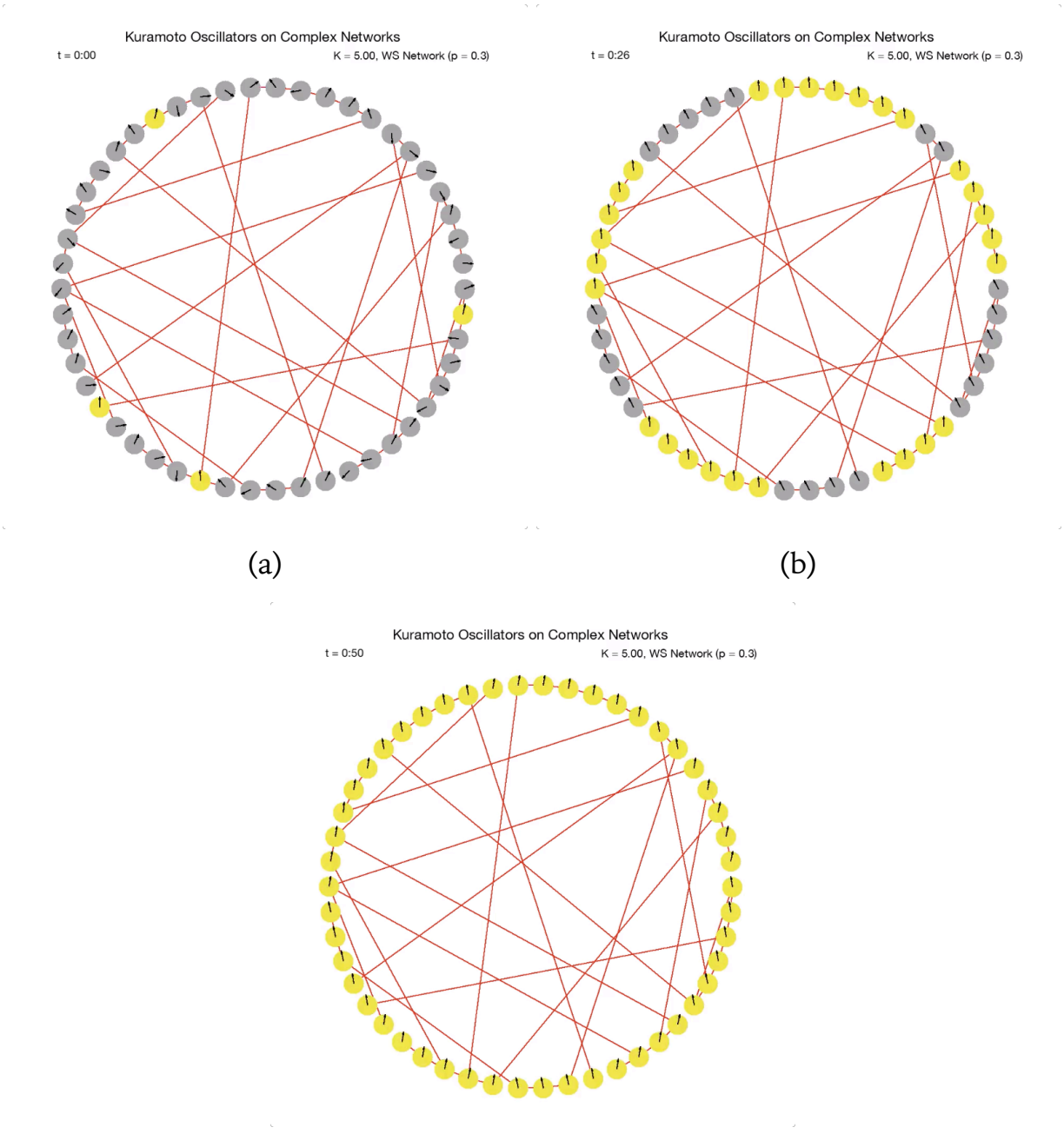


Figure 4.4: Simulation of Kuramoto model on a WS network with 50 nodes. The red lines depict the links. (a), (b) and (c) are snapshots from the video at instants t = 0s, t = 26s and t = 50s respectively.

The result produced some aesthetically pleasing visuals, which very much resembles the actual phenomenon observed in swarms of tropical fireflies.[15] As before, the first and last frames of the video are as shown in Fig 4.4, along with an intermediate frame (Fig. 4.4 (b)). It shows small clusters of sync forming among interconnected nodes which gradually coalesce and result in global synchronisation towards the end (Fig. 4.4 (c)).

4.1.4 Summary

With the above preliminary groundwork, we succeeded in achieving the following two things:

1. Get an intuition of the underlying Physics of synchronisation in a population of interacting phase oscillators; and
2. Know the basics of complex networks and put in place the computational framework necessary to handle them.

Armed with this basic knowledge, we then proceed to the actual problem.

4.2 DCLNET: Static properties

Immediately after obtaining the network edge-list, the first step was to study its static properties like degrees of the nodes and the degree distribution. It was done to identify any topological feature that might have some bearing on the dynamics. The first thing to notice was that the network contains single- and double-weighted links. At first, the network was treated as undirected. The degree of each node was calculated and plotted against the index of the nodes (Fig. 4.5 (a)). The degree distribution was also calculated from the edge list and plotted (Fig. 4.5 (b)).

As evident from Fig. 4.5 (a), the node indexed 1 has the highest number of links. Apart from node 1, a few other nodes are also densely connected (called *hubs* henceforth) while the majority of the nodes are sparsely connected. We notice that nodes with smaller indices ($\lesssim 50$) tend to have the

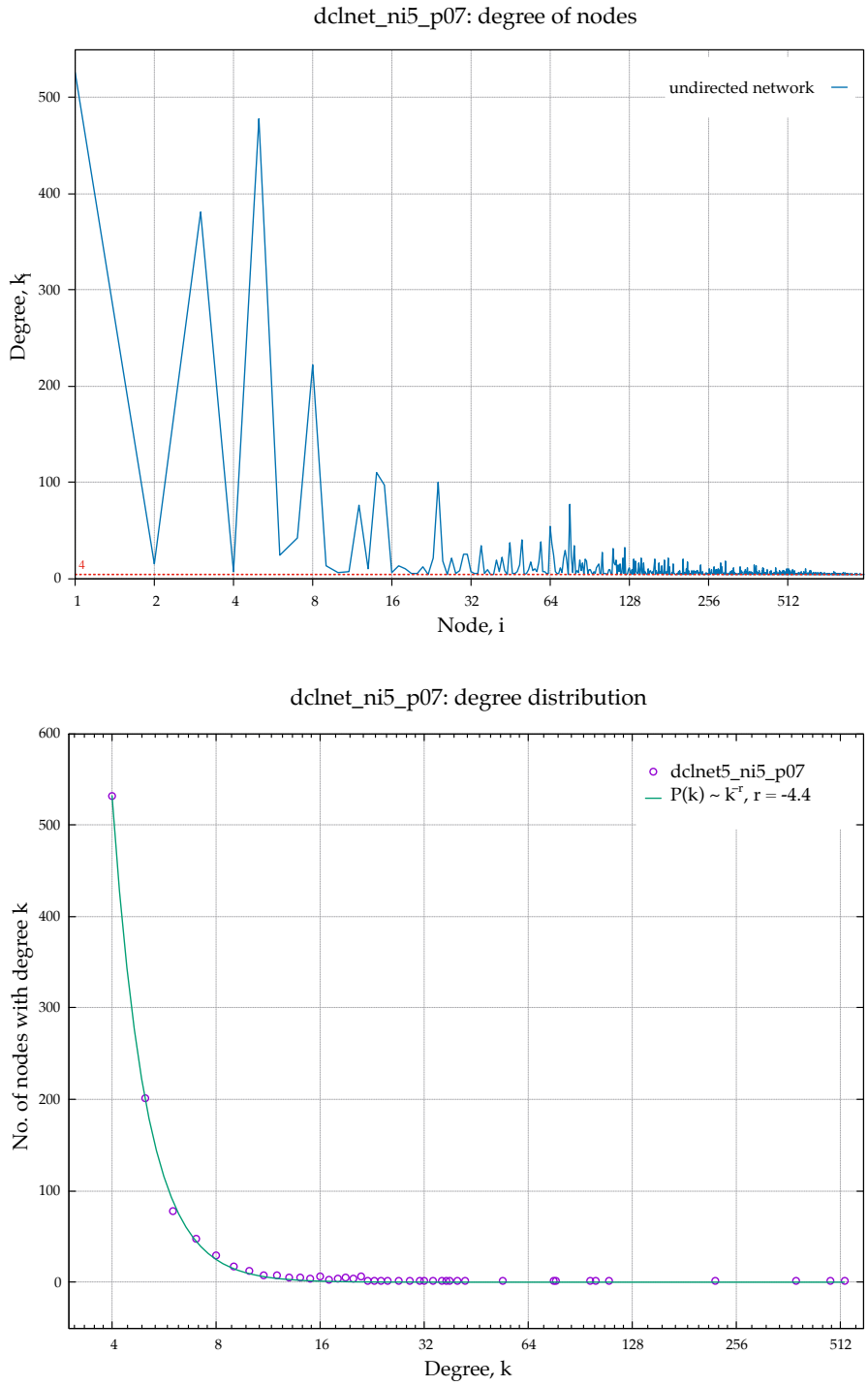


Figure 4.5: Static properties of *dclnet*. Top: (a) degree of nodes and bottom: (b) degree distribution.

most number of connections. Further, no node has a degree less than four, i.e. all nodes have at least four links. Moreover, it is clear from the degree distribution (Fig. 4.5 (b)) that four is the most probable degree in the network.

The degree distribution is also seen to obey a power law, which is a characteristic of the so-called Scale-Free (SF) networks (e.g. the Barabási-Albert model).[16] Knowledge of the actual process of generation is required to confirm this speculation.

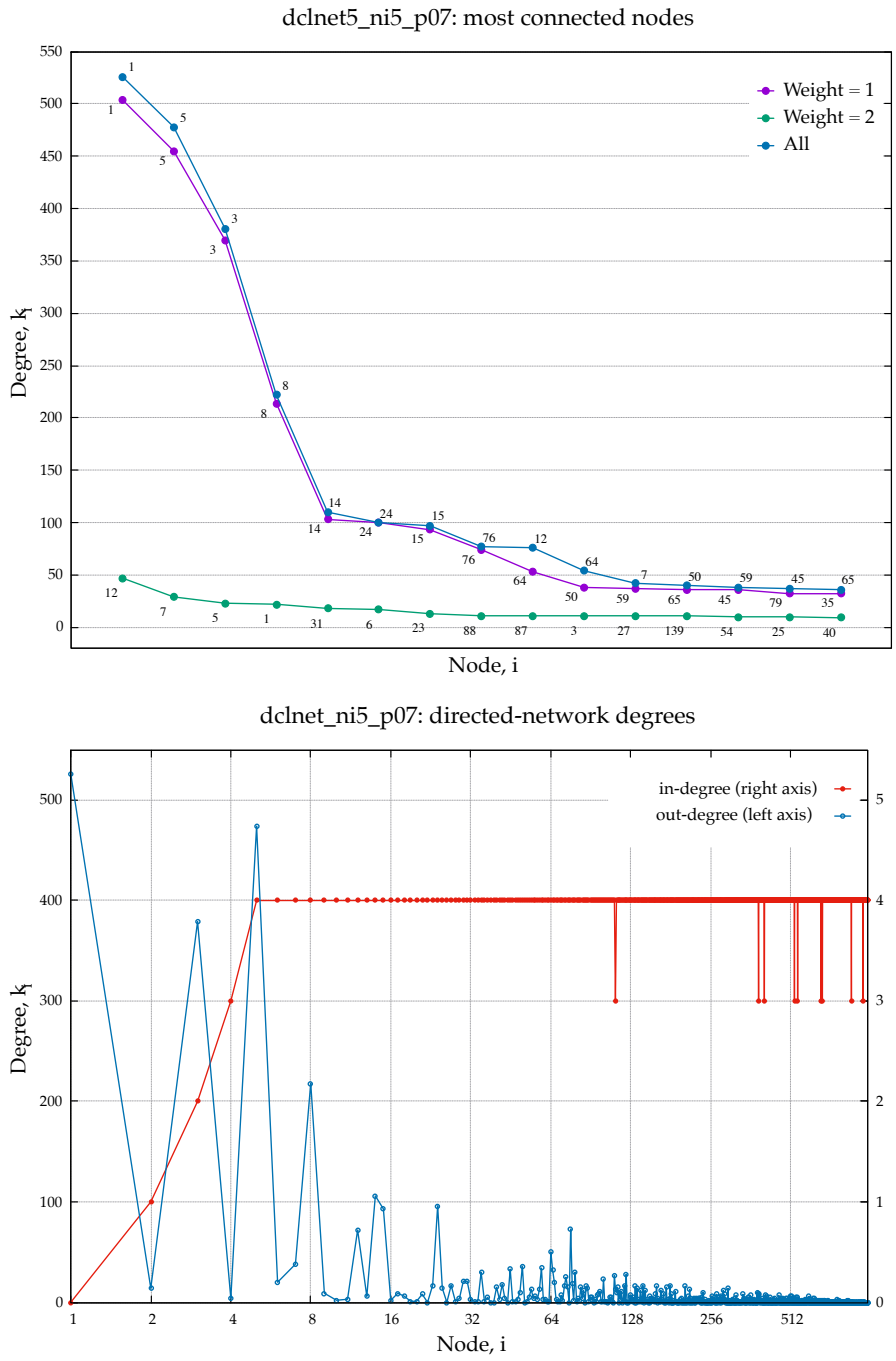


Figure 4.6: Top: (a) Hubs in the undirected *dclnet*;
bottom: (b) in-degree and out-degree of nodes in the directed *dclnet*.

The first few hubs are identified and plotted in Fig 4.6 (a). It shows that nodes indexed 1, 5, 3 and 8 are the main hubs. The nodes 12 and 7 are hubs for the double-weighted links alone.

We can also treat the network as directed, the direction of links taken as being from the nodes in the first column of the edge list to the corresponding nodes in the second. This treatment breaks the symmetry of the links: we can now identify the in-degree (number of incoming links) and out-degree (number of outgoing links) of the nodes and plot them separately, as shown in Fig. 4.6 (b). One of the striking features of the network becomes immediately evident from Fig. 4.6 (b):

The hub node 1 has no incoming links but has the most number of outgoing links.

This property of the network when treated as directed readily establishes the following:

1. Node 1 is not influenced by any other nodes in the network; so it will merely oscillate at its natural frequency throughout the simulation.

2. Node 1 also influences more than half of all the nodes in the network. This fact suggests that the dynamics of the entire network is bound to depend strongly on the behaviour of node 1 in relation to the behaviour of the rest.

These properties place Node 1 in the position of a leader/commander of the entire network. Interestingly, this is the antithesis of the primary egalitarian character usually found in synchronising systems[2] that they synchronise without any cue from a specific leader. Presence of hubs, in general, brings in a bias in the system. These observations were reinforced by evidence generated by simulations carried out throughout the rest of the project.

4.3 DCLNET: Dynamic properties

4.3.1 Undirected network

The next step was plainly to put the Kuramoto oscillators on the nodes and simulate their dynamic evolution. To calculate the order parameter $|r|$ after every RK4 step is unnecessary and also increases computations; instead, it was calculated at regular intervals (after 100 RK4 steps) and plotted against time. Simulations on the undirected network were carried out first, with random initial phases ($\theta_i \in \mathbb{U}(0, 2\pi)$) and frequencies ($\omega \in \mathbb{G}(2\pi, \pi/3)$). The system synchronises smoothly. There is no sign of Chaos (Fig. 4.7).

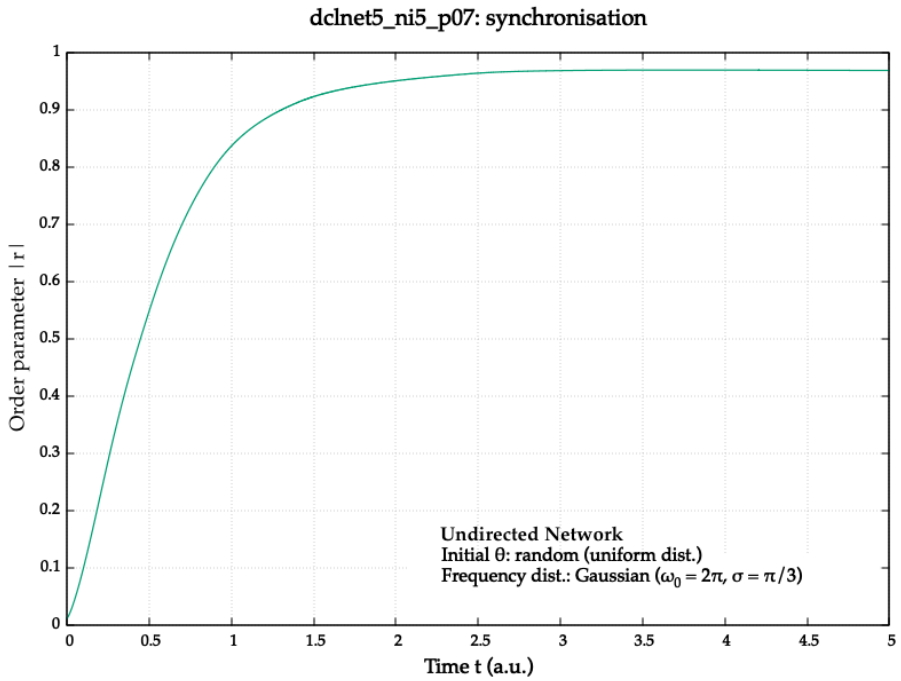


Figure 4.7: Time evolution of order parameter in the Kuramoto simulation of undirected *dclnet* showing smooth synchronisation.

By varying the coupling strength K, a phase transition diagram (Sec. 4.1.2) was produced next (Fig. 4.8 (a)). It shows some small fluctuations near the critical coupling strength K_c. The root-mean-square (RMS) dispersion in $|r|$ about the mean can be a good measure of these fluctuations. However, as we

are not interested in a quantitative description of the same, it is more convenient to just plot all values of $|r|$ over time along the y-axis for a given K on the x-axis (like a Bifurcation diagram[17]). This alternative reduces computation and shows the maximum extent of fluctuation in $|r|$ for a given K.

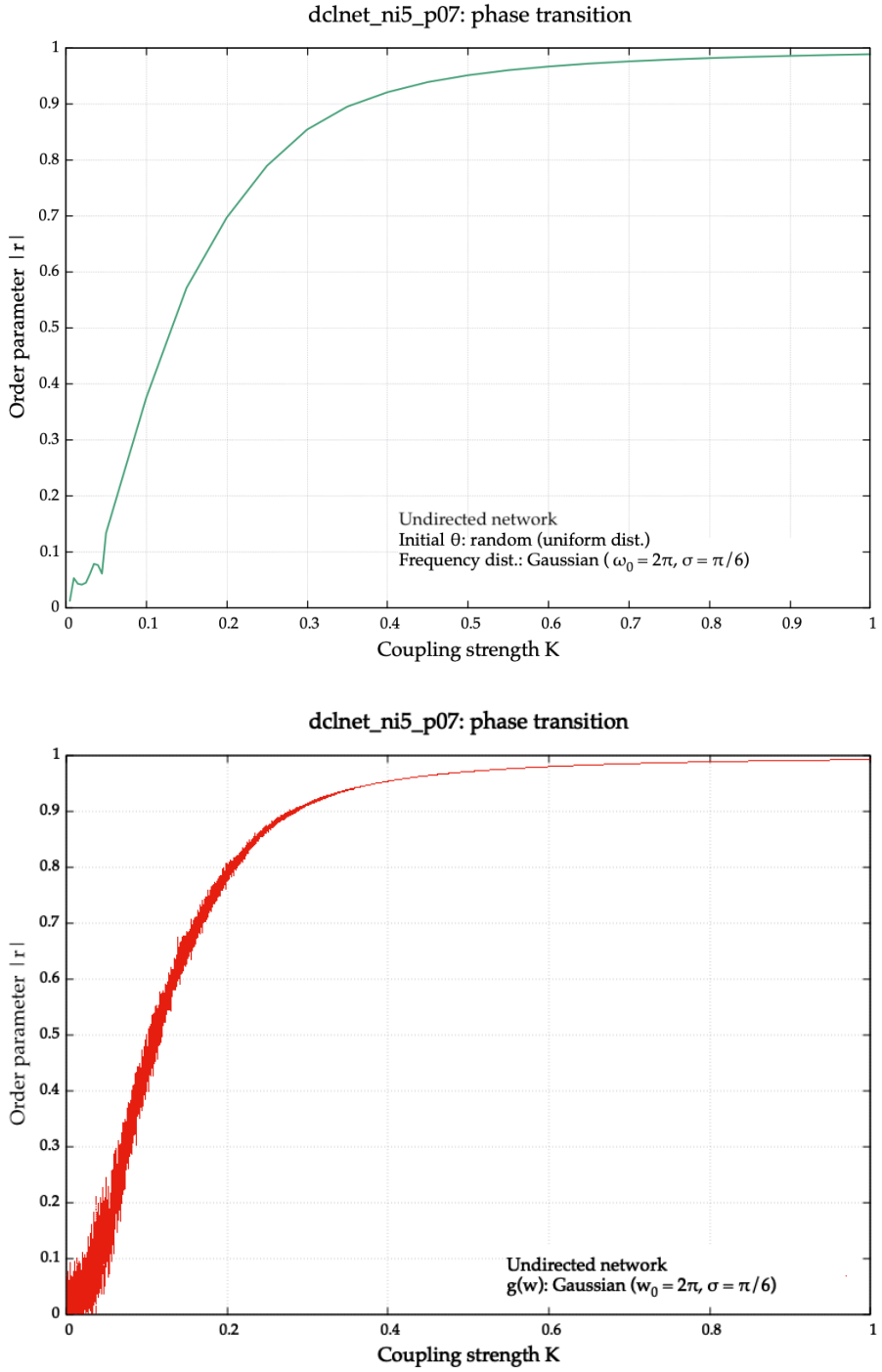


Figure 4.8: Phase transition in the undirected *dclnet* showing on the y-axis: (a) time-averaged $|r|$ (top) and (b) all values of $|r|(t)$ (bottom).

The modified plot (Fig. 4.8 (b)) more pronouncedly shows the fluctuations in $|r|$ in the transition region. These fluctuations are a manifestation of the critical-slowdown problem where, in the transition region, the simulation leads to the formation of clusters of nodes with similar phases, and the existence of multiple such clusters results in large fluctuations in $|r|$ over time. It is an issue that generally occurs in Monte-Carlo methods (see Metropolis algorithm[18]); curiously it appears here as well, despite being a fully deterministic simulation.

4.3.2 Directed network

The attention was now diverted to the directed network. Interestingly, in the time evolution of $|r|$ for maximum coupling ($K = 1$), new oscillatory patterns are seen for some values of σ (Fig. 4.9 (a)). As K is far from the critical coupling, this oscillation cannot be associated with the critical-slowdown fluctuations mentioned in Sec. 4.3.1. We are led to accept that the oscillation is characteristic to the synchronised state.

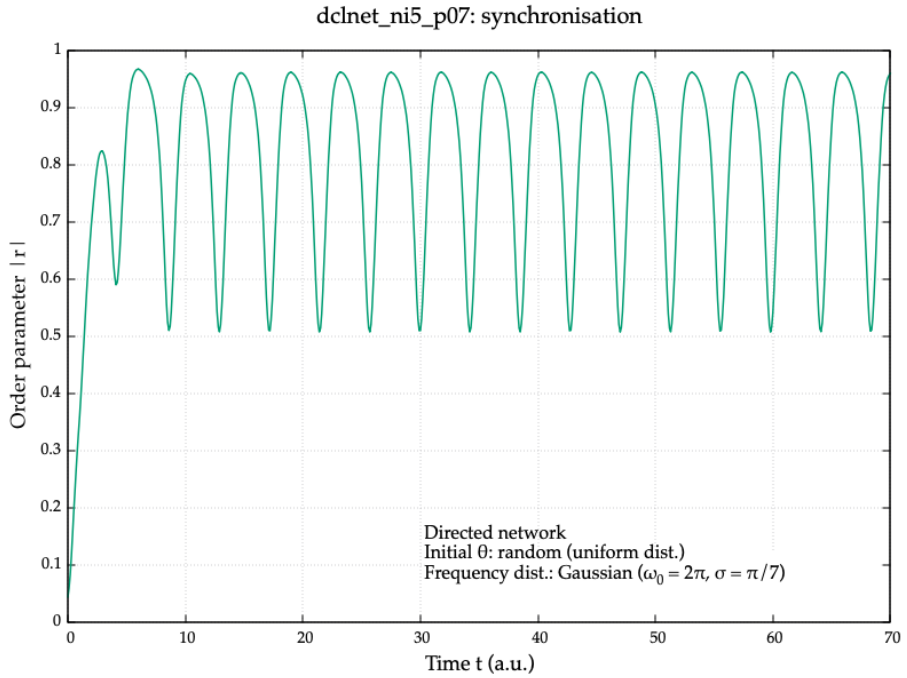


Figure 4.9: (a) Time evolution of order parameter in the directed *dclnet* showing oscillatory patterns in the synchronised state (breathing modes)

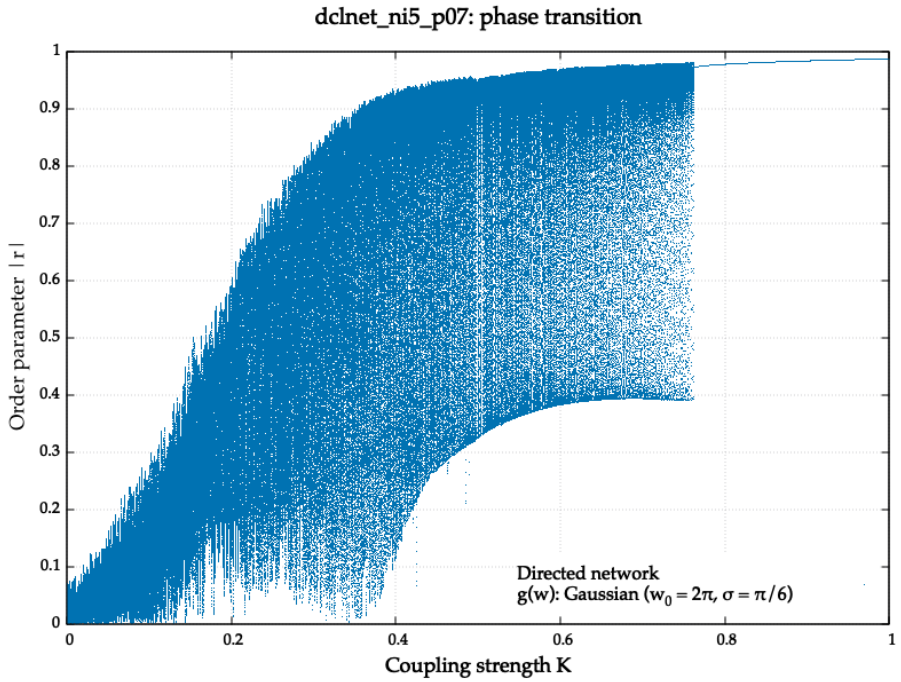


Figure 4.9 (cont.): (b) phase transition in the directed *dclnet* showing the same.

The oscillation is also seen to be periodic, with identical, repetitive patterns. Hence it is not a chaotic attractor, and the trajectory in phase space is a closed-loop. Such oscillations were almost absent in the undirected case.

The phase-transition diagram (Fig. 4.9 (b)) of this configuration (using the modified method) also shows very wide oscillations in $|r|$ away from the transition region, confirming that it is not a critical-slowdown effect. To see at what values of σ they are present, the simulation was repeated while varying σ. It turns out they are absent for comparatively narrower ($\sigma \lesssim \pi/3$) frequency distributions; the results are summarised in Fig. 4.10.

Another feature to note in Fig. 4.10 is that for enormous values of σ, the oscillation becomes aperiodic with a low overall level of synchronisation. It possibly indicates a chaotic regime. In order to verify this, the dependence of the dynamics on the initial conditions was studied for $\sigma = \pi/2$. Two runs were conducted with the same frequency distribution, but with different $\theta_i(0)$ realisations which produced two completely different oscillatory patterns in $|r|$

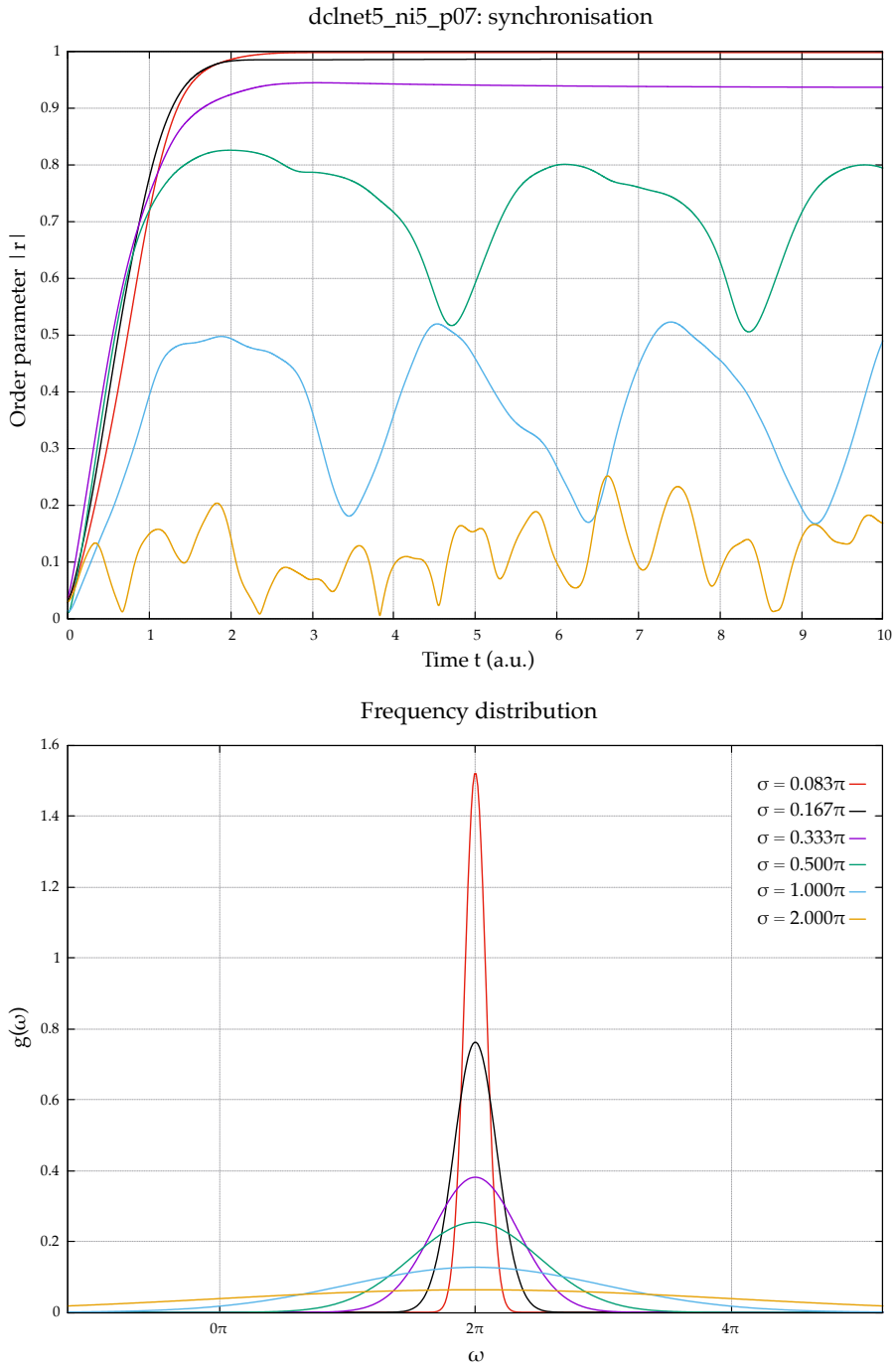


Figure 4.10: (top) Difference in the fluctuations in $|r|$ for different widths of the frequency distribution (bottom) for directed *dclnet*.

as shown in Fig. 4.11. These runs demonstrate sensitive dependence on initial conditions, the famous *butterfly effect*[19], a signature of chaos in non-linear systems. The evolution in the undirected network under similar circumstances are included in the plots for comparison. It can be seen that apart from the

transients, they are nearly identical in the two cases.

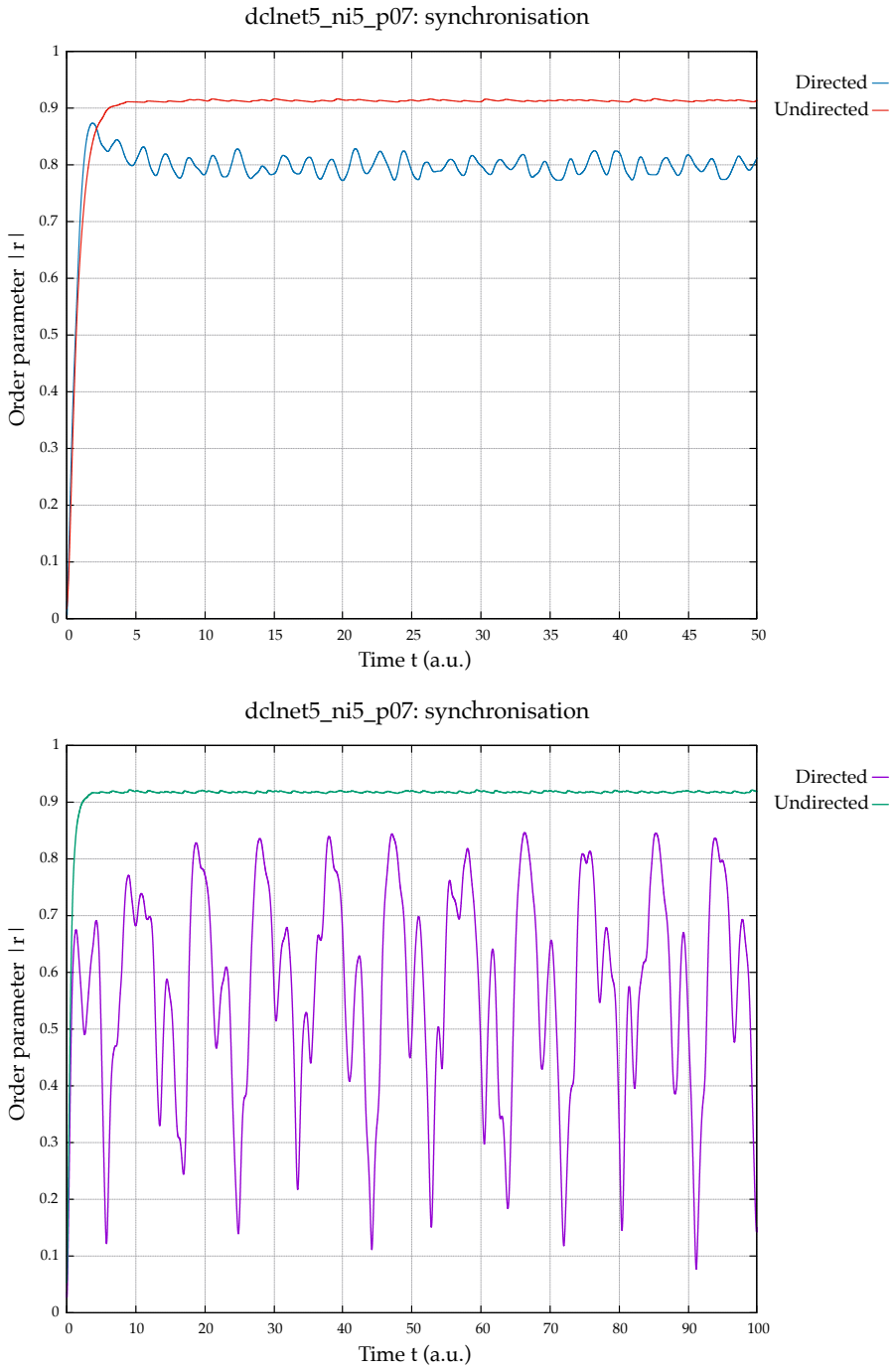


Figure 4.11: Sensitive dependence of the breathing modes in directed *dclnet* on initial conditions for a wide frequency distribution ($\sigma = \pi/2$). Same simulation for undirected *dclnet* plotted for comparison.

This result strongly indicates that the oscillators behave chaotically if their natural drives are distributed very widely. However, most natural

collections of oscillators do not display such high degrees of disparity in their intrinsic drives. It must be remembered that, in the Kuramoto model, the oscillators are assumed to be nearly identical. The provision for variation in the intrinsic frequency of the oscillators is to account for the natural noise. So in this project, such broad frequency distributions and their effects are not pursued further, and we limit our focus to cases of moderate-to-small spreads in the frequencies, for which the oscillation in $|r|$, if any, is periodic.

Thus, while dclnet shows plain, predictable behaviour when treated as undirected, the same network, when treated as directed, shows unusual patterns of oscillation which warrant further investigation.

4.3.3 Effect of frequency distribution

In order to better correlate the magnitude of fluctuations in the global order parameter with the distribution of natural frequencies of the oscillators, we need to carry out simulations while varying the later, but keeping the set of initial phases fixed in all cases. Since σ is only a measure of the dispersion in the frequencies, and the dynamics might depend on the actual values ω_i, we repeat the simulations for multiple realisations of ω_i for a given σ value and look at an averaged picture. In the present case, they were repeated for 400 different random realisations of ω_i for the same σ, and the variation in the average extent of fluctuation of $|r|$ across nine different values of σ in the range $\left(\frac{\pi}{4}, \frac{\pi}{12}\right)$ were compared. The obtained plots are shown in Fig 4.12.

From the figure, it is apparent that the extend of fluctuation varies even for the same σ, i.e. it indeed depends on the actual realisation of ω_i values. Although, it must be emphasised that this is not the sensitive dependence characteristic of chaos, as the ω_i are not initial conditions but permanent parameters of the system. We can also see in Fig. 4.12 that the fluctuations reduce in occurrence and extent as we progressively move towards smaller values of σ, i.e. narrower spread, finally getting eliminated by $\sigma = \pi/12$. At

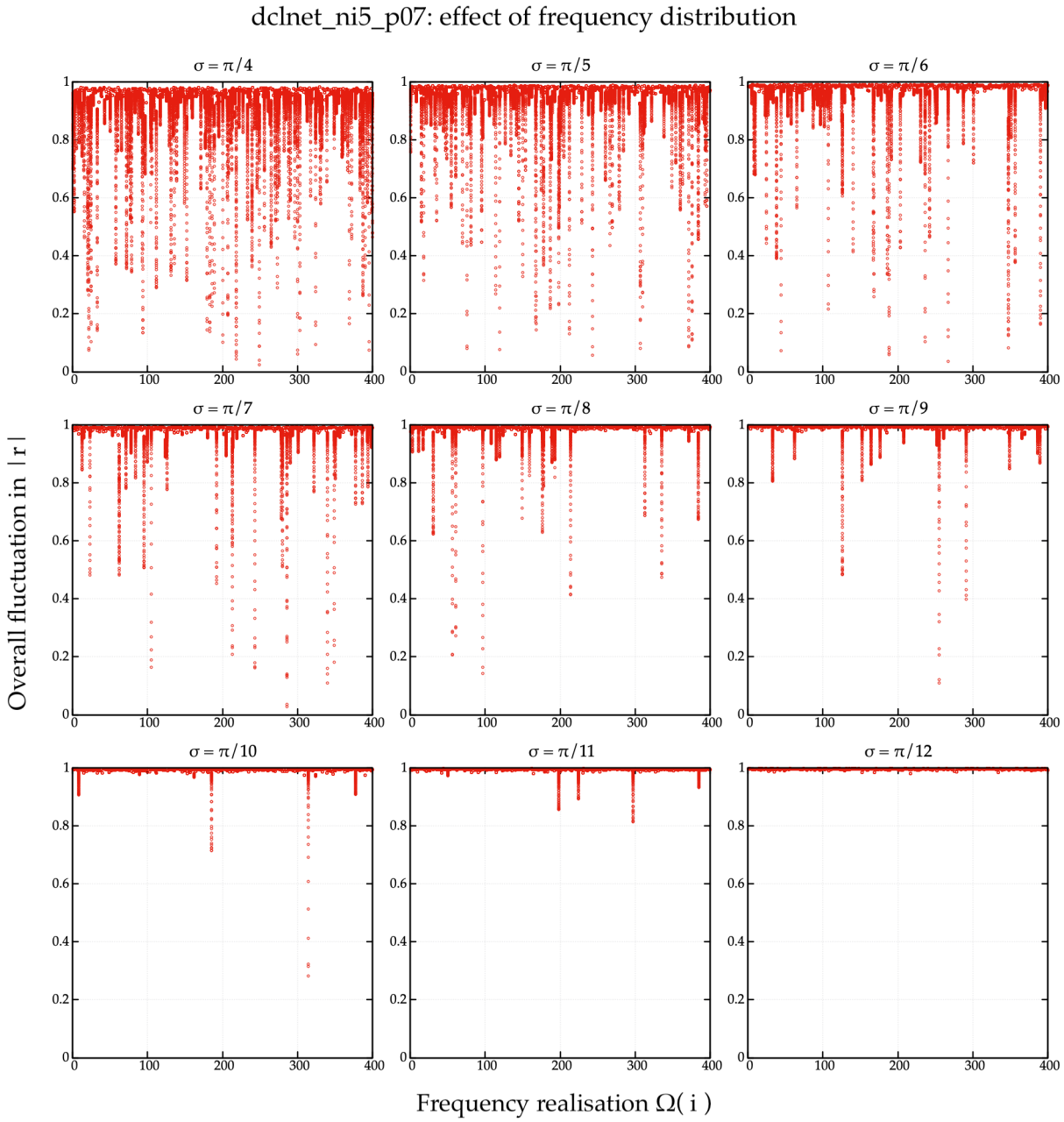


Figure 4.12: Overall fluctuations in order parameter in the directed *dclnet* for different widths of the frequency distribution for multiple ω-realisations.

such a narrow Gaussian, the oscillations in $|r|$ are virtually absent in all the 400 random realisations, suggesting a high level of synchronisation among the nodes. This behaviour is expected, as the tendency to synchronise increases as the oscillators become more and more identical. The maximum $|r|$ attained also increases gradually with decreasing σ, apparent from the plots in Fig. 4.12.

4.3.4 Video simulations

In order to figure out the exact reason behind these characteristic oscillations in $|r|$, and to try and relate it to the structural properties of the network, we need to plot and study the actual time evolution of the phases of the Kuramoto oscillators. It takes us back to the method employed during the preliminary work, of using video plots (Sec. 4.1.1). There, rotating arrows were used to create simple visualisations; but now a more quantitative representation is needed. The standard way is to plot the oscillators as dots on a circular ring, with the angle they subtend at the centre (relative to the positive x–axis) being the phases of the oscillators. A single rotating arrow of time-varying length, which depicts the complex global order parameter, is also usually placed at the centre of the circle (ref. Fig 4.13).

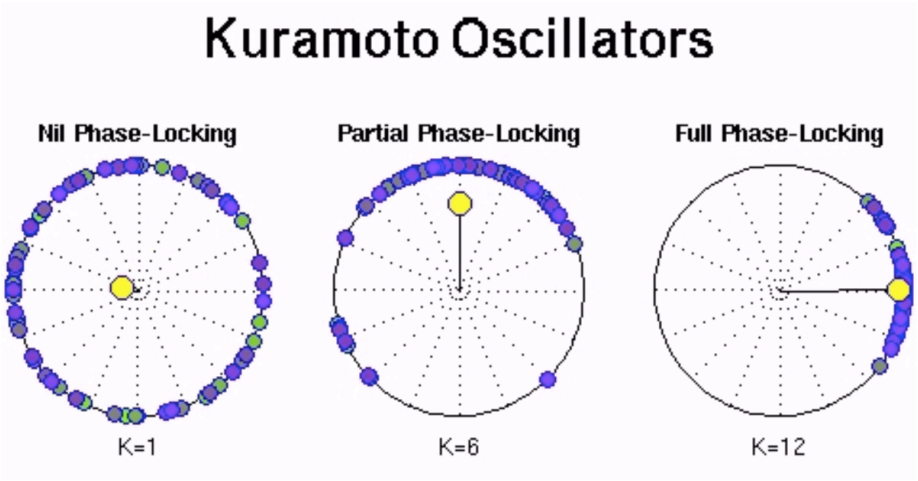


Figure 4.13: Common representation of the phases of Kuramoto oscillators.[20]

Unfortunately, this representation lacks a crucial feature: the easy identification of which node has a particular phase. A large number of revolving dots makes any labelling of the nodes nearly impossible. This identification assumes significance because we would need to discern at least some nodes, especially the hubs, among the dots. Hence, a flat representation was chosen, with phases $\theta_i \in (0, 2\pi)$ on y–axis. Therefore a dot on the graph shows the

phase (ordinate) of the n^{th} node (abscissa) and moves upward or downward as the phase evolves. As the vertical interval is circular, the top and bottom boundaries wrap around, i.e. the dots emerge from the opposite side when they cross the top or the bottom edges.

Of the 400 realisations each for the different σ values, a few cases were handpicked, based on the extent of fluctuation in $|r|$, for individual study. The choices themselves were arbitrary; nevertheless, care was made that these cases encompassed a range of small-to-large variation in $|r|$. The plots of the time evolution of $|r|$ for the selected cases (Fig. 4.14) should make this clear. As we are looking at cases that do not depend on the initial phases, an arbitrary realisation of random phases ($\in \mathbb{U}$) was used for all following simulations.

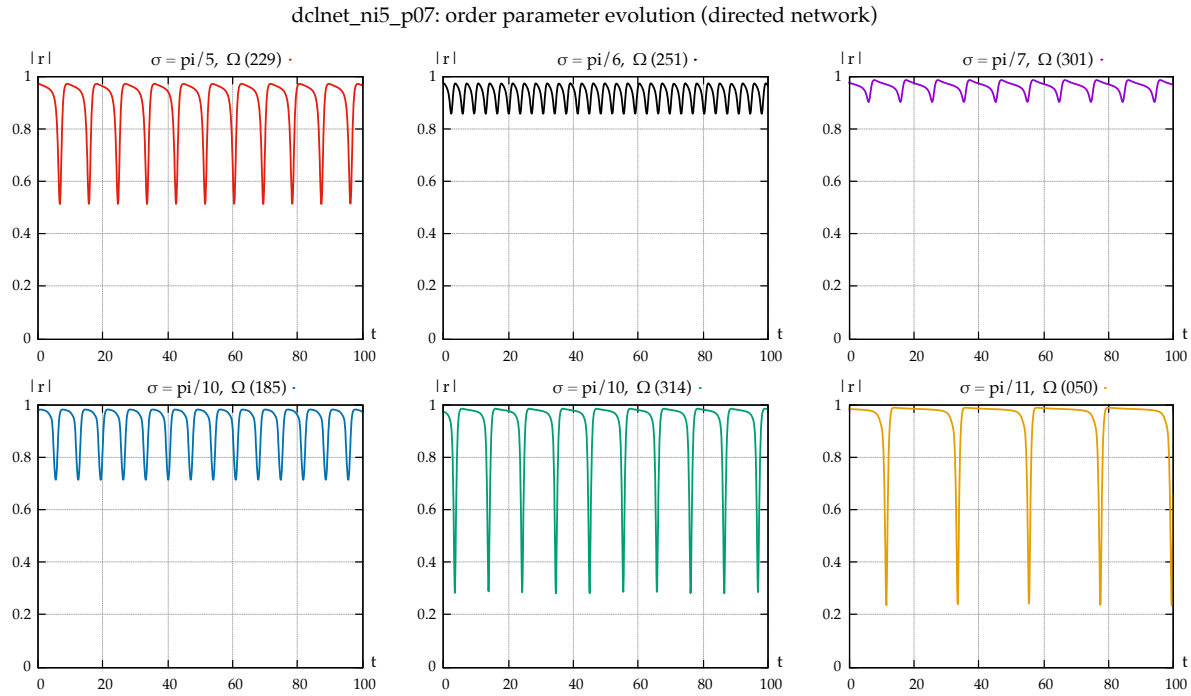


Figure 4.14: Order-parameter evolution in directed *dclnet* for some hand-picked cases of frequency realisations. $\Omega(i)$ denotes the i^{th} realisation for a given σ.

4.3.5 Discovery of the Leader

The video simulations immediately confirmed one of the earliest

speculations: the independence of node 1 when the network is treated as directed, and its leading influence over the rest of the network. Some frames extracted from the video of a particular case at specific time intervals are shown in Fig. 4.15. As expected, node 1 oscillates at its natural frequency unaffected, influencing over half of all the nodes it is directly connected to, and also the rest of the network indirectly. It is precisely this influence, and the resulting attempts of the nodes to follow or disregard the phase of node 1 that leads to the peculiar modes in the synchronised state.

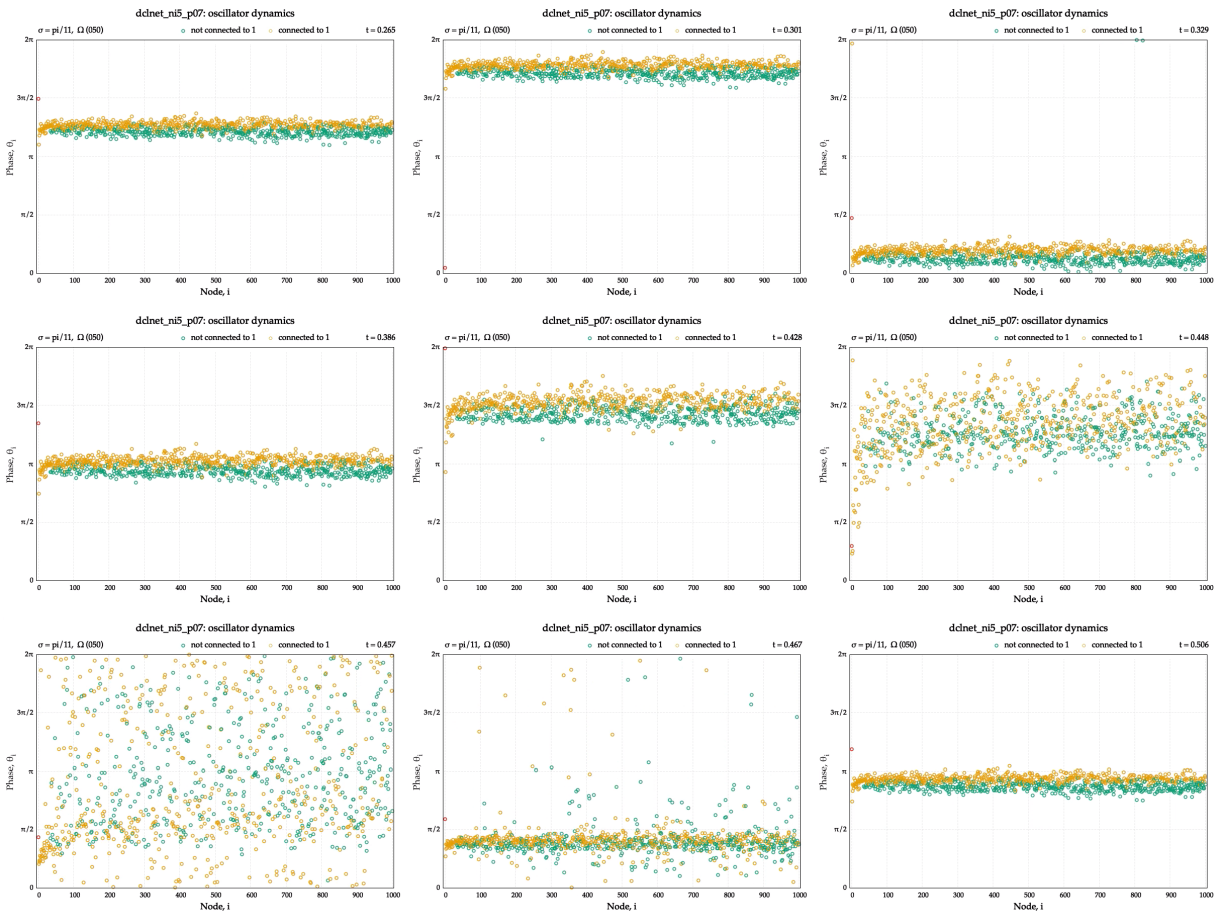


Figure 4.15: Frames extracted from a video simulation of phase evolution in directed *dclnet* showing the mechanism behind breathing modes.

This defining aspect of the network warrants the introduction of the following terminologies, which will assist in elaborating the exact mechanism behind the oscillating modes:

1. *The Leader*: the independent node indexed 1 that exerts influence on most of the network;
2. *Followers*: the nodes which are directly linked to the Leader and hence influenced by it; and
3. *Non-followers*: the nodes which are not linked to the Leader. Still, they are linked to some of the Followers, thus indirectly influenced by the Leader.

In the video simulations, the Leader, the Followers and the Non-followers are plotted in different colours for easy identification.

4.3.6 Breathing modes

After the initial transients die out, the nodes fall into a rhythmic cycle, shown in Fig. 4.15, giving rise to the oscillatory modes. The cycle broadly consists of the following:

1. The Leader oscillates at its natural frequency, unaffected by any of the other nodes;
2. The Followers catch up with the Leader, and the Non-followers with the Followers, resulting in synchronisation; this is a state of minimal interaction, as the phases are close together;
3. In the synchronised state, the Leader leads; the Followers try to follow, but the Non-followers drag them back, creating a tension;
4. Phase differences between Followers and Non-followers gradually increase;
5. Finally the synchronisation breaks: all nodes reorganise to equilibrate again, and the entire cycle repeats.

As the level of synchronisation varies throughout the cycle, it results in periodic oscillations in $|r|$. Such an oscillatory mode of a network is termed as a *Breathing mode*. The occurrence and period of such breathing modes depend on the distribution of the natural frequencies of the phase oscillators, as observed in Sec. 4.3.3. This fact can be explained now as follows. The closeness of the Followers' frequency with the Leader's determines their ability to catch

up with the latter and maintain a synchronised state. Further, the differences in the natural frequency of the Followers and Non-followers determine their ability to retain mutual coherence. Thus, lesser the spread of the frequencies, more the chance of a stable, sustained synchronised state and hence the absence of breathing modes. The actual frequency realisation determines, in a non-trivial way, the amount of spread in the oscillator phases during the disruption and reorganisation stage of the breathing mode, which in turn decides the extent of oscillation in $|r|$. It is why the latter vary across realisations of the same Gaussian distribution (ref. Fig. 4.12).

To summarise, the dynamics in the directed network can be viewed as the result of an interplay between two forces in the network, viz. the attraction between the Leader and Followers and between the Followers and Non-followers, which can be mutually supporting or opposing at times. An analogy can be drawn with a system of three blocks connected by two (non-identical) springs laid horizontally on a smooth table (Fig. 4.16). From basic mechanics, we know such a system exhibits oscillations which are superpositions of its natural modes. Similarly, the breathing modes can simply be considered as superpositions of the natural modes of the network, which is fundamentally a system of coupled oscillators, albeit with a large number of degrees of freedom.

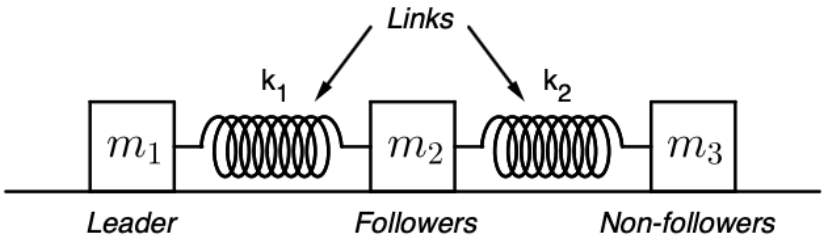


Figure 4.16: Analogy of the topology and dynamics of directed *dclnet* with a mechanical system of three blocks coupled by two springs.

In the spring system, it is the fixed properties of the system, viz. the masses of the blocks, the spring constants of the springs and the configuration of interconnections, that fully determine the dynamics of the system. Likewise, it is the static properties of the network, viz. its topological configuration along

with the natural frequencies of the individual oscillators at its nodes, that determine its dynamics.

That it is the topology that governs the dynamics to a large extent is readily established: adding even one incoming link from any other node to the Leader, thus altering the topology, makes the breathing modes disappear (Fig. 4.17). In this case, the lack of independence of the Leader forces it to fall in step and maintain a constant phase relation with the Followers and Non-followers, thus sustaining the sync.

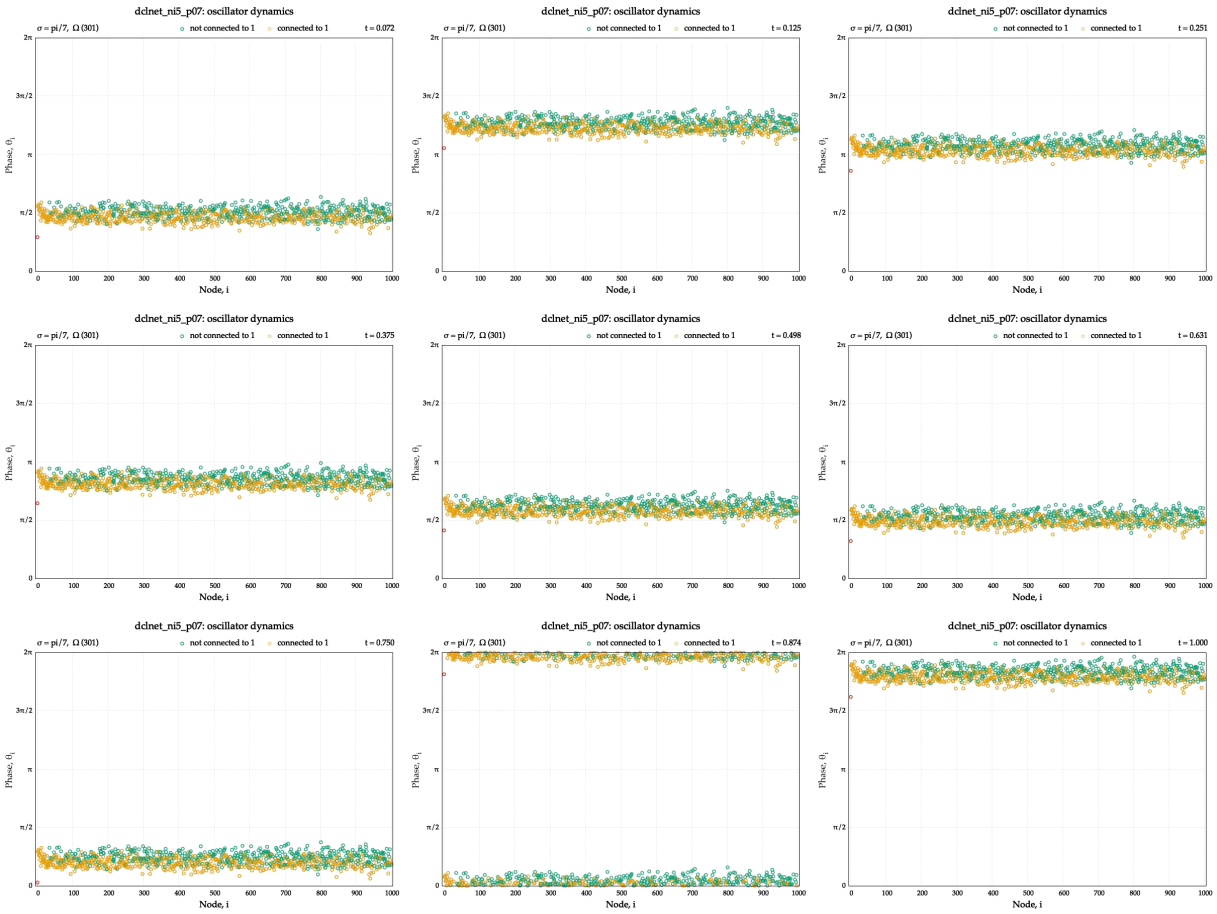


Figure 4.17: The result of adding an incoming link to the *Leader* in a directed *dclnet* configuration which otherwise shows a breathing mode.

4.3.7 Influence of the Leader

Similar video simulations were carried out for the other handpicked cases. The same leader-follower behaviour giving rise to the breathing modes of

synchronisation was observed in all of them. It shows that this is a general feature of the network dynamics. In all of these cases, the randomly generated natural frequency of the Leader (ω_1) was seen to be either very large or very small than the mean of the frequency distribution. To better quantify the effect of this disparity, ω_1 was varied in steps through $(-\pi, \pi)$ while keeping the ω of all other oscillators fixed, measuring the amplitude of the breathing mode in each case. The resulting plot is as shown in Fig. 4.18.

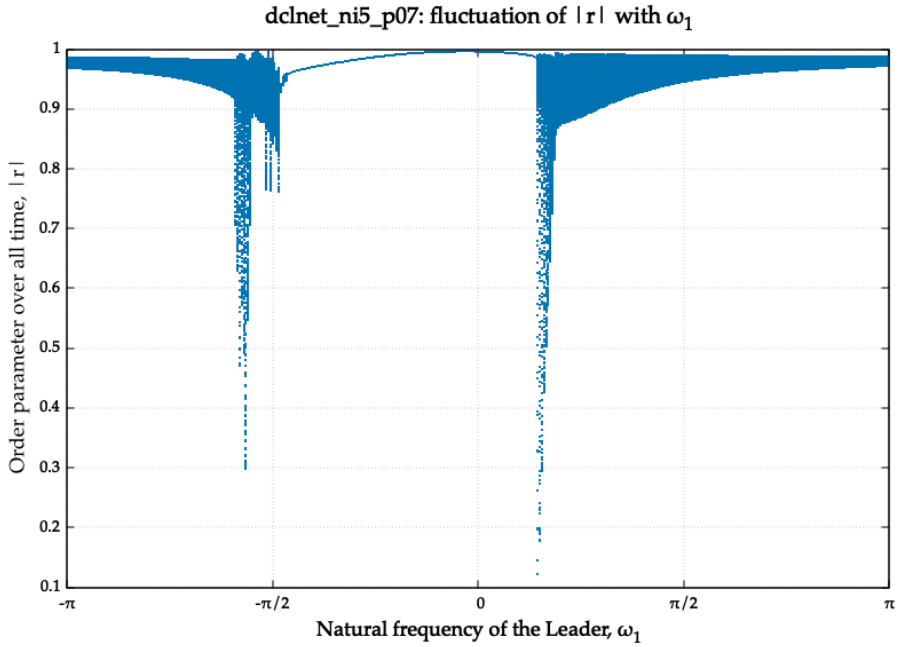


Figure 4.18: Variation in the overall fluctuations of the order parameter with the Leader's frequency in an arbitrary case.

There is a small 'window' of natural frequencies of the Leader close to 0, which is the mean of the Gaussian distribution from which the frequencies are drawn, for which breathing modes are absent and $|r|$ remains constant. This region is where the Leader's frequency is comparable to that of the other nodes. If the Leader oscillates very fast compared to the rest, $|r|$ oscillates by only a small amount, as seen towards both ends of Fig. 4.18. It is due to the breathing modes being short and frequent, owing to the Leader's high frequency of oscillation. The maximum fluctuations occur when ω_1 is moderately larger than

the mean. Naturally, the effect is not symmetric about the mean as well.

4.3.8 Influence of other hubs

The Leader node has such a dramatic influence on the entire network not just because of its independence, but rather because it is the most connected node in the network, i.e. the densest hub. However, node 1 is not the only hub; Fig. 4.6 shows the other hubs like 5, 3 and 8. These lesser connected hubs are also bound to influence the breathing-mode dynamics, though to a lesser extent than the Leader.

For demonstrating this, the same simulation of Sec. 4.3.7 was repeated, varying the natural frequency of the 2^{nd} most connected node (5) this time, while keeping the other ω's fixed (including ω_1). The outcome is shown in Fig. 4.19. The plot is similar to Fig. 4.18, with a window of reduced fluctuations in $|r|$; but unlike the previous case, the breathing modes are nowhere fully eliminated, which shows that the effect of ω_5 is lesser compared to that of ω_1.

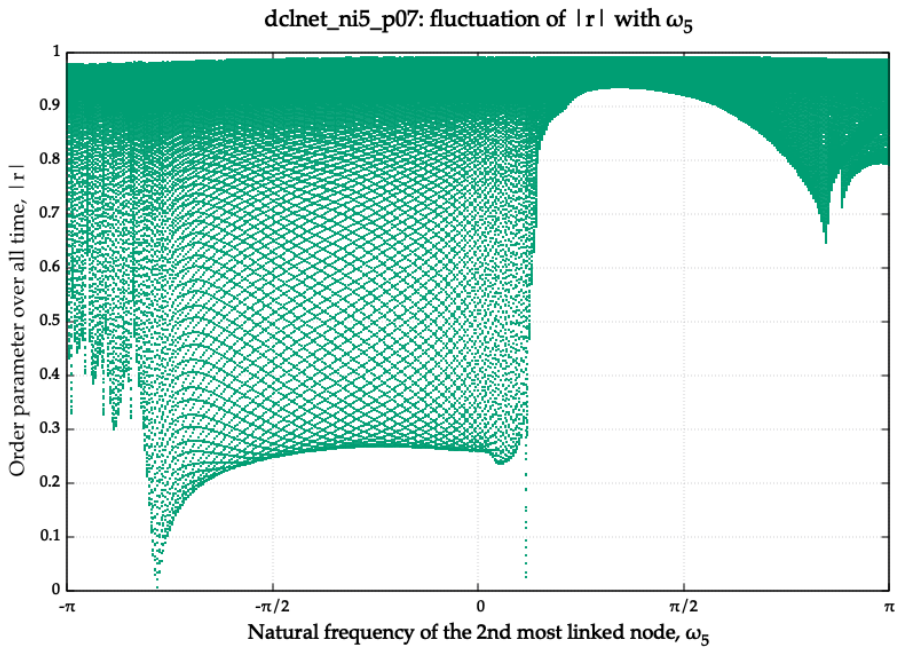


Figure 4.19: Variation in the order-parameter fluctuations with the frequency of node 5 in the same case as that of 4.3.7.

The same could be repeated for the other hubs as well, but such plots are neither particularly informative nor reveal any new phenomenon.

4.4 HIGHER ORDER STRUCTURES IN DCLNET

4.4.1 Maximal cliques

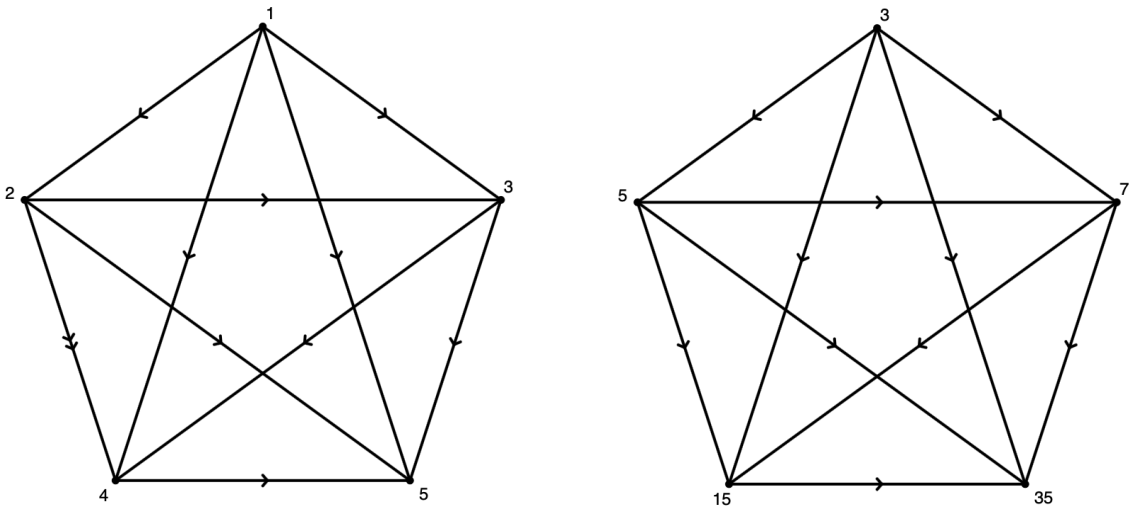
So far, we have been discussing the dynamic behaviour of the Kuramoto oscillators interacting through the network substrate of dclnet. Kuramoto model is a one-to-one, pairwise interaction model. However, it is not the only mode of interaction that might be present in complex networks. As mentioned in the Introduction section, higher-order interactions that involve more than two nodes simultaneously are supposed to exist in various natural systems, and the pairwise Kuramoto model has been extended to incorporate such interaction terms in applicable cases. It is therefore advisable to look for appropriate topological structures that may lend themselves into higher-order interactions, which are nothing but the maximal cliques. Their presence, if any, in the dclnet network would indicate the applicability of the higher-order terms.

The Bron-Kerbosch (BK) algorithm, mentioned in Sec. 2.5, is used to find the maximal cliques in a network, given all the pairwise links in the network. A basic intuitive understanding of the algorithm enabled the author to write a custom code-implementation of it that blended in with the existing code framework, eliminating the need for any external-library implementations. A benchmarking test showed that it ran sufficiently fast for our purposes.

Application of the BK algorithm to dclnet revealed more striking static properties of the network:

1. The dclnet network is entirely made up of 4-dimensional cliques (also called 5-cells[21]), i.e. all-to-all connected subgraphs containing five nodes each. There are precisely 987 such 5-cells in the network. Maximal cliques of no other dimension exist in the network.

2. 520 of the 987 cliques contain node 1 (the Leader) as a member, while the remaining 467 cliques do not.
3. In all cliques, the direction of flow (when we treat the network as directed) is from smaller-indexed nodes to larger-indexed nodes (see example in Fig. 4.20 (a)). So none of them is a *circular* flow.
4. A clique contains **at most** one double-weighted link ($2 \twoheadrightarrow 4$ in the example just mentioned). There are 682 such links distributed (seemingly randomly) amongst the 987 cliques.



(a) (b)

Figure 4.20: Two cliques in the *dclnet* network (a) with and (b) without a double-weighted link.

This clique structure immediately explains why the nodes in the network have a minimum degree of four (as clearly noted in Sec. 4.2.): each node exists belonging to a 5-cell and hence have four neighbours at the least. Hubs like node 1 and 5 are shared by more than one clique, which makes their degree larger than four. Meanwhile, the flow of control clearly depicts the hierarchy from small-indexed nodes down to the large-indexed ones, in agreement with earlier observations.

This finding is significant because it reveals an organised pattern of

interconnected, uniform-dimensional cliques in what was previously seen as a randomly linked collection of nodes. It is too non-trivial to be a coincidence, indicating the network was generated precisely this way. It also invites us to look at the cliques, more than the individual oscillators, as the possible fundamental units of interaction in the network. It makes sense to study how synchronisation evolves within the cliques separately, what influences their behaviour and how they interact with each other, analogous to the work done so far for the individual oscillators. It could also prepare the ground for future work intending to model a higher-order interaction term on the cliques.

4.4.2 Clique dynamics

The measure of synchronisation within individual cliques is quantified by the local order parameter, calculated by applying equation (2) to only the nodes forming a clique, as opposed to the global order parameter used for the whole network. As there are 987 cliques in the network, this gives us 987 parameter values evolving in time. Hence, in the same fashion as done in Sec. 4.3.4, we use video plots to study the evolution, with each dot in the plot now representing the local order parameter value $|r_i|$ of one of the 987 cliques. Since $|r_i|$ is bound to $(0, 1)$, the dots do not cross the horizontal edges as before but stays within the plots. The frames from one such video simulation are compiled in Fig. 4.21.

In these plots, we distinguish the cliques that either do or do not contain the Leader by the colour of the dots, (see legends). By extending our earlier nomenclature, we shall call them *Follower cliques* and *Non-follower cliques* respectively. Clearly, from Fig. 4.21, the two categories of cliques are seen to segregate into two distinct groups and remain so for a large part of the breathing-mode cycle. The disruptive phase of the cycle is long and protracted for the Follower cliques, while it is short and rapid for the Non-follower cliques. Hence, the latter remains mostly in synchronisation while the former goes through lower levels of synchronisation for the most part of the cycle.

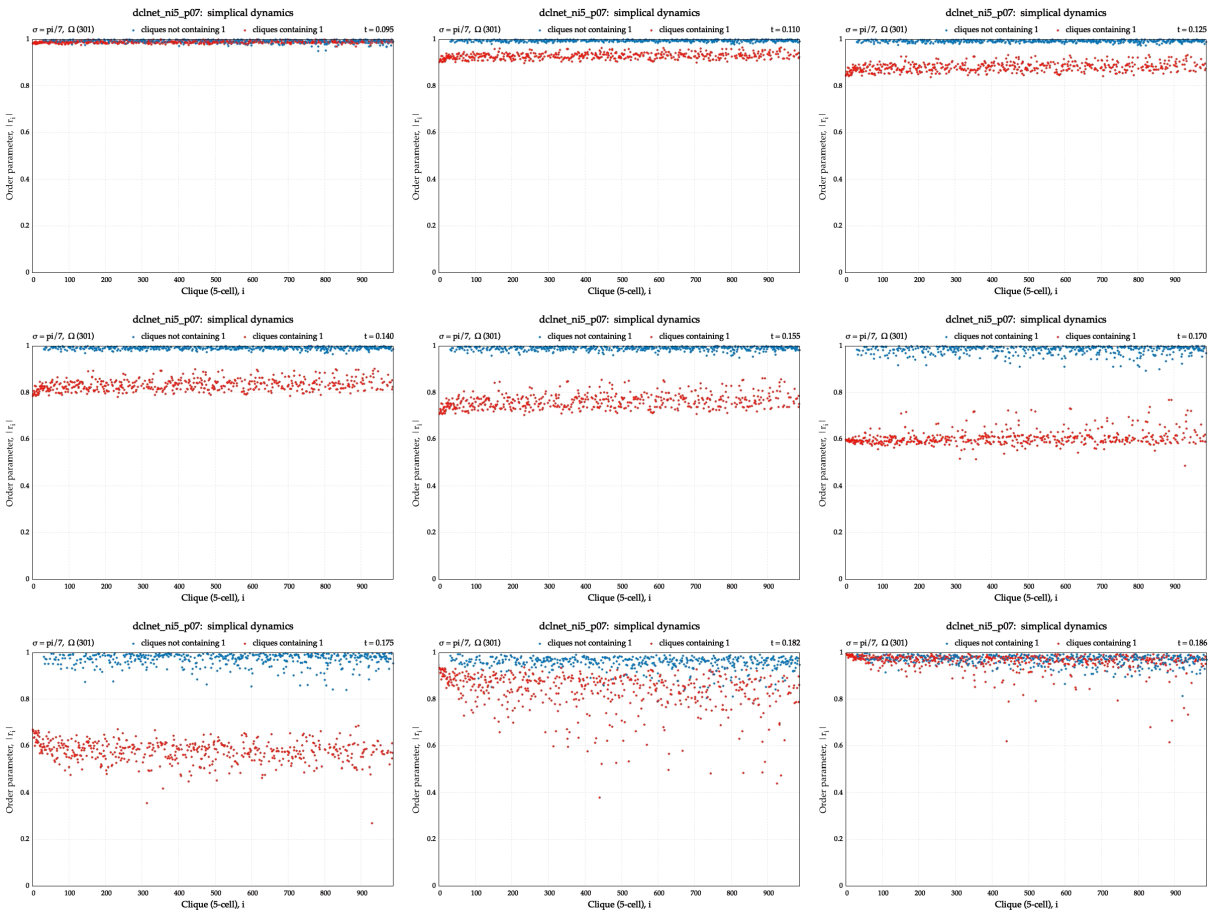


Figure 4.21: Frames extracted from a video simulation showing the time-evolution of the local order parameter of the cliques during a breathing mode.

This difference can be readily explained by comparing the corresponding frames of oscillator-wise and clique-wise simulations for the same configuration, especially at the instances minimum and maximum synchronisation, as shown in Fig. 4.22. Unsurprisingly, at the instance of maximum synchronisation ($t = 0.131$), all nodes are nearly in phase meaning all cliques have $|r_i| \approx 1$. Though, at $t = 0.249$ which is an instant of minimum synchronisation, we see that while the Followers and Non-followers are still mostly synchronised, they are nearly completely out of phase (by $\approx 180°$) with the Leader. Therefore the complex amplitude ($e^{i\theta_j}$) of the Leader almost cancels out the amplitude of one member in each clique to which it belongs. The coherence of the other three nodes of the clique can result in a maximum $|r_i|$

value of $\approx 0.6\,(3/5)$, which is close to the values observed (see lower-right plot in Fig. 4.22). On the other hand, the complex amplitudes of the oscillators in Non-follower cliques add up to give an $|r_i|$ value close to 1, as they do not suffer this cancellation. Of course, both of this is disrupted during the decoherence and reorganisation phase of the breathing mode.

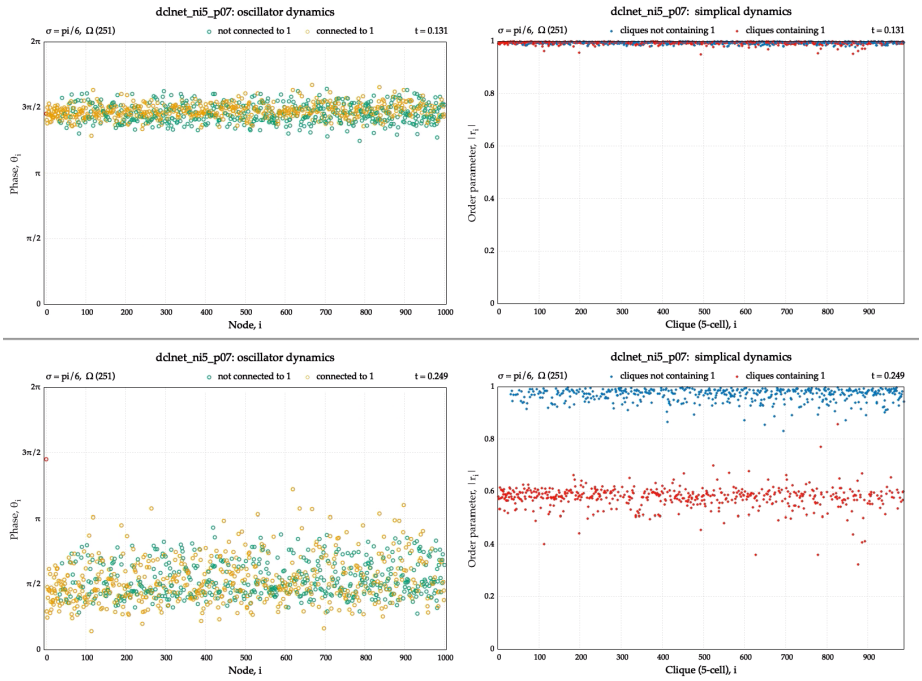


Figure 4.22: Side-by-side comparison of frames from the oscillator-wise and clique-wise simulations of a breathing mode, showing instants of maximum (top) and minimum (bottom) order parameter $|r|$.

The clique-wise simulations were repeated for a couple of other previously handpicked configurations a well, as for different coupling strength factors. All of them displayed more or less similar dynamics. These final set of simulations further underline the dramatic role of the topology in general, and the Leader in particular, on the dynamics of the network. All these characteristics in the behaviour stem from the two fundamental properties, viz. presence of hubs and the independence of the Leader, when we treat the network as a directed network.

CHAPTER 5

RESULTS AND DISCUSSION

All the simulation studies conducted as part of this project exhibited the dominant hand of topology in determining the dynamics of the network's phase oscillators. An analysis of the network structure of dclnet revealed an intriguing topology. We saw the presence of hubs and a degree distribution peaked at $k = 4$. It was later revealed to be due to the clique structure. We then found the most crucial feature of the network when treated as directed, which is the independence and influence of the node indexed 1. This pivotal position motivated the naming of node 1 as *Leader*. It is atypical as it violates one of the basic tenets of most synchronous phenomena, which is that spontaneous synchronisation emerges in them without a cue or command from any specific leader figure.

Nevertheless, this configuration was given the attention because of the appearance of "breathing" sync states in the form of periodic variations in the evolving order parameter. We saw that such variations were absent from the dynamics of the undirected network. We then introduced a novel way of representation of the oscillator phases to produce video simulations of the phase evolution. The video simulations at once confirmed the correlation between breathing modes and dynamic interactions between the Leader, its Followers and the Non-followers. We explored the drastic effect altering the topology even slightly had on the dynamics. We were also able to explain why the extent of variations in the order parameter depends on the frequency distribution, especially on its broadness. We also discovered a possibly chaotic regime of frequency distributions but did not pursue it further, citing a lack of relevance to real-world scenarios.

We then discovered the underlying higher-order structure, viz. the 4-simplexes making up the network, using the Bron-Kerbosch algorithm. This discovery explained most of the earlier observations made on the static properties of the network. A study of the dynamics of the network with cliques taken as the basic units showed the distinct status of the Leader in even better contrast. We reasoned the dynamic bifurcation of the Follower- and Non-follower cliques based on their phase relations existing during the breathing modes. More tests could be conducted to examine the influence of the Leader further; a quick suggestion would be to *drive* the independent Leader with an external function and watch how the network responds.

To summarise, the network code-named `dclne5_ni5_p07` has a very modular, homogenous organisation of nodes. The undirected dclnet network has a hub-oriented structure, typical of SF networks, with dense connections and tight couplings leading to extremely stable synchronisation states. The directed network, while still hub-centric, can be considered to have a biased and asymmetric topology. Speaking in more human terms, it has a very undemocratic character due to the unilateral influence of the hubs, with an autocratic Leader on the top. A small hierarchy of nodes, beginning with the Leader, holds the sway over virtually the whole network, making it a highly oligarchic order. Such an arrangement, and the dynamics arising out of it, can find parallels in both Nature and human societies.

CHAPTER 6

CONCLUSION

This draws us to the conclusion of the project. Much information has been generated about the network studied, which will hopefully become useful in future research into the same. We finish in order, leaving both the space for more investigation and a strong foundation to build upon. An interesting topic to study next would be the role of the double-weighted links in the dynamics, which has been left mostly unexplored by the present study. Another matter worthy of investigation is the occurrence of Chaotic configurations discovered in Sec. 4.3.2, but not pursued to the end. Finally, the clique-wise results could become the springboard for further developments.

The discovered simplicial structure is still only static; the interactions simulated throughout were of the pairwise network-Kuramoto model. As we already mentioned, this simplicial structure could be a basis for higher-order interactions that could be incorporated in a future study. Such higher-order terms would, of course, alter the currently observed dynamics. We hope the results we have obtained here would help in arriving at an interaction term that leads to a particular behaviour of choice. We hope whatever we discovered would be an asset in furthering the intentions behind generating this network with precisely this structure.

We have, from the beginning, stressed upon the necessity of making the simulations efficient. Numerical integration of differential equations like the Kuramoto equation (3) we studied is notorious for being computationally expensive and time-consuming. On the other hand, for difference equations (also called *maps*, e.g. the Logistic map) the computation is much simpler and

much faster (by a factor of up to 1000) so that more simulations can be performed in the same duration. So it would be advantageous to cast the problem considered, if possible, to a study of a plane mapping (for an example see Ref. [22]). The challenge here is to devise a map that replicates the dynamical behaviour observed. This project's work could be an aid in that direction.

On the whole, the stated aim of freely exploring the network and hence unravel its properties can be considered to have been successful. Personally for the author, this project provided an immense opportunity to study, understand and assimilate a lot of important concepts and aspects of Physics with significance in numerous prospective research fields. It has undoubtfully paved a strong foundation to choose for a career in research in this promising new field of science.

REFERENCES

[1] Lorenz, Edward N. (1963). *Deterministic non-periodic flow*. J. Atmos. Sci. (1963) **20** (2), 130–141.

[2] Steven Strogatz. 2003. *Sync: The Emerging Science of Spontaneous Order*. Penguin Books.

[3] Arenas, A., Diaz-Guilera, A., Kurths, J., Moreno, Y., & Zhou, C. (2008). *Synchronization in complex networks*. Physics Reports, **469** (3), 93-153.

[4] Kuramoto, Y. *Self-entrainment of a population of coupled nonlinear oscillators*. International Symposium on Mathematical Problems in Theoretical Physics, Lecture Notes in Physics, vol. 39, Springer, NY, USA, 1975, pp. 420–422.

[5] D.J. Watts, S.H. Strogatz. *Collective dynamics of 'small-world' networks*. Nature (London) **393** (1998) 440.

[6] Graph (abstract data type) - Wikipedia: Representations. `https://en.wikipedia.org/wiki/Graph_(abstract_data_type)#Representations`

[7] P.S. Skardal and A. Arenas. *Higher-order interactions in complex networks of phase oscillators promote abrupt synchronization switching*. arXiv:1909.08057, 2019.

[8] V. Salnikov, D. Cassese, R. Lambiotte. *Simplicial complexes and complex systems*. Eur. J. Phys. **40**, 014001 (2019).

[9] Simplex - Wikipedia. `https://en.wikipedia.org/wiki/Simplex`

[10] Bron, Coen; Kerbosch, Joep (1973). *Algorithm 457: finding all cliques of an undirected graph*, Commun. ACM, ACM, **16** (9), 575–577

References (continued)

[11] William H. Press et al. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, 1992, 712-714.

[12] Link to GitHub repository hosting full code: `https://github.com/shivaprsdv/msc-project` .

[13] Bosiljka Tadić, Jozef Stefan Institute, Ljubljana. Private communication.

[14] Liu, R., Beus, P., Madler, S., & Bush, B. (2015). *Analysis of Watts-Strogatz Networks*. p 1.

[15] Video clip showing fireflies flashing in sync: `https://youtu.be/sR0KYelaWbo` .

[16] A.L. Barabási, R. Albert, *Emergence of scaling in random networks*, Science **286** (1999) 509–512.

[17] Bifurcation diagram - Wikipedia. `https://en.wikipedia.org/wiki/Bifurcation_diagram` .

[18] Metropolis, et al. 1953, Journal of Chemical Physics, **21**, pp. 1087–1092.

[19] Butterfly effect - Wikipedia. `https://en.wikipedia.org/wiki/Butterfly_effect` .

[20] Screenshot taken from video file: *Phase locking in a network of all-to-all coupled Kuramoto oscillators*, by Stewart Heitmann, Wikimedia Commons. `https://en.wikipedia.org/wiki/File:KuramotoModelPhaseLocking.ogv`

[21] 5-cell - Wikipedia. `https://en.wikipedia.org/wiki/5-cell` .

[22] M. Hénon and C. Heiles, *The applicability of the third integral of motion: Some numerical experiments*, Astron. J. **69**, 73 (1964), pp. 77-78.

APPENDIX

These are the relevant parts of the C programs written to do the simulations. The full code is available on the author's GitHub repository (Ref. [12]).

kuramoto.c

```
#include "network.h"
#include "util.h"
#define N 1000

/* Data storage */
double omega[N];
struct adjnode *adjlist[N] = { NULL };

/* Function to execute the governing equation */
void kuramoto(double t, double theta[N], double thdot[N])
{
        int i;
        struct adjnode *ap;

        for (i = 0; i < N; ++i) {
                thdot[i] = omega[i];
                for (ap = adjlist[i]; ap != NULL; ap = ap->next)
                        thdot[i] += ap->wt * sin(theta[ap->id - 1] - theta[i]);
        }
}

/* Function to calculate the global order parameter */
double ord_param(double *theta, unsigned int n)
{
        int i;
        double sr, si;

        sr = si = 0;
        for (i = 0; i < n; ++i) {
                sr += cos(theta[i]);
                si += sin(theta[i]);
        }
        return sqrt(sr * sr + si * si) / n;
}

/* Function to calculate the local order parameter of cliques */
double clq_ord(struct clique *clq, double *theta)
{
        double sr, si, th;
        struct adjnode *ap;
```

```
        sr = si = 0;
        for (ap = clq->nodes; ap != NULL; ap = ap->next) {
                th = theta[ap->id - 1];
                sr += cos(th);
                si += sin(th);
        }
        return sqrt(sr * sr + si * si) / clq->dim;
}
...
/* Main program for the last simulation done */
int main(int argc, char *argv[])
{
        int t, i;
        const int T = 1e5;
        double r, h, theta[N];
        FILE *fp;
        struct adjnode *p = NULL;
        struct clique *c, *clq = NULL;

        fp = fopen(argv[argc - 2], "r");
        readadjl(fp, adjlist, N, UNDIR);
        for (i = 1; i <= N; ++i)
                addnode(&p, i, 1);
        bk(NULL, p, NULL, adjlist, N, &clq);
        freeadjl(adjlist, N);
        rewind(fp);
        readadjl(fp, adjlist, N, DIR);
        fclose(fp);

        fp = fopen(argv[argc - 1], "r");
        for (i = 0; i < N; ++i)
                fscanf(fp, "%le\t%le\n", &theta[i], &omega[i]);
        fclose(fp);
        h = 1e-3;
        for (t = 0; t < T; ++t) {
                if (!(t % 100)) {
                        for (c = clq; c != NULL; c = c->next)
                                printf("%le\t%d\n", clq_ord(c, theta),
                                        (has_node(c->nodes, 1) ? 7 : 6));
                        printf("\n\n");
                        fflush(stdout);
                }
                rk4(t * h, theta, N, kuramoto, h, theta);
        }
        return 0;
}
```

network.c

```
#include "network.h"
#include "util.h"
#define HASH(n) (n - 1)
```

```
/* Function to read nodes from an edgelist file into adjacency list */
int readadjl(FILE *fp, struct adjnode **adjlist, int lim, enum nwtype t)
{
        int n1, n2, w, edgc;

        edgc = 0;
        while (fscanf(fp, "%d %d %d", &n1, &n2, &w) != EOF) {
                if (n1 > lim || n2 > lim) {
                        fprintf(stderr, "Node %d is out of limits!\n",
                                        (n1 > lim) ? n1 : n2);
                        return -1;
                }
                ++edgc;
                if ((t == UNDIR && addnode(&adjlist[HASH(n1)], n2, w) == NULL)
                                || addnode(&adjlist[HASH(n2)], n1, w) == NULL) {
                        fprintf(stderr, "Memory allocation error!\n");
                        return -1;
                }
        }
        return edgc;
}

/* Function to calculate and output the degrees of nodes */
void degdist(struct adjnode **adjlist, int lim)
{
        struct adjnode *ap;
        int i, d;

        for (i = 0; i < lim; ++i) {
                for (d = 0, ap = adjlist[i]; ap != NULL; ap = ap->next, ++d)
                        ;
                printf("%d %d\n", i + 1, d);
        }
}

/* Bron-Kerbosch algorithm to find the maximal cliques in the network */
void bk(struct adjnode *r, struct adjnode *p, struct adjnode *x, struct adjnode
        **adjlist, int lim, struct clique **clqs)
{
        int n;
        struct adjnode *v, *pnew, *xnew;

        if (p == NULL && x == NULL) {
                v = NULL;
                if ((n = append(r, &v)) != -1)
                        if (addclq(clqs, v, n) == NULL)
                                fprintf(stderr, "Memory allocation error!\n");
                return;
        }
        while (p != NULL) {
                v = p;
                if ((n = v->id) > lim) {
                        fprintf(stderr, "Node %d is out of limits!\n", n);
                        return;
                }
                p = p->next;
```

```
                v->next = r;
                pnew = xnew = NULL;
                intersect(p, adjlist[HASH(n)], &pnew);
                intersect(x, adjlist[HASH(n)], &xnew);

                bk(v, pnew, xnew, adjlist, lim, clqs);
                v->next = x;
                x = v;
        }
}
```

util.c

```
#include <math.h>
#include <time.h>
#include "util.h"

/* Return a random number from the Gaussian distribution centred at <mu> having
 * a std. deviation <sig>. Employs the Marsaliga / Box-Muller polar method. */
double gauss(double mu, double sig)
{
        double u, s;
        static double v;
        static enum bool gen = false;
        gen = !gen;                     /* generate afresh every other call */

        if (!gen)
                return v * sig + mu;    /* return spare */
        /* Get a random point (u, v) within the unit circle centred at origin */
        do {
                u = frand(-1, 1);
                v = frand(-1, 1);
                s = u * u + v * v;
        } while (s >= 1.0 || s == 0.0);
        s = sqrt(-2 * log(s) / s);      /* polar scaling */
        v *= s;                         /* store spare & return the other */
        return u * s * sig + mu;
}

/* n dimensional RK4 algorithm for calculating a single step.
 * This implementation is loosely based on the one given in Numerical Recipes.
 * As we cannot multiply arrays with numbers in C (without using a loop), an
 * approach different from the traditional way (k1, k2, ...) has been followed.
 */
void rk4(double x, double y[], int n, deriv_func f, double h, double yout[])
{
        int i;
        double hh, xh, *dy, *dys, *yt;

        yt = vector(n);
        dy = vector(n);         /* derivatives of y1, y2, ... */
        dys = vector(n);        /* sum of derivatives */
        hh = 0.5 * h;
        xh = x + hh;
```

```
        f(x, y, dy);                            /* first step */
        for (i = 0; i < n; ++i) {
                yt[i] = y[i] + hh * dy[i];      /* y + k1 / 2 */
                dys[i] = dy[i];
        }
        f(xh, yt, dy);                          /* second step */
        for (i = 0; i < n; ++i) {
                yt[i] = y[i] + hh * dy[i];      /* y + k2 / 2 */
                dys[i] += 2 * dy[i];
        }
        f(xh, yt, dy);                          /* third step */
        for (i = 0; i < n; ++i) {
                yt[i] = y[i] + h * dy[i];       /* y + k3 */
                dys[i] += 2 * dy[i];
        }
        f(x + h, yt, dy);                       /* fourth step */
        for (i = 0; i < n; ++i)
                yout[i] = y[i] + h / 6.0 * (dys[i] + dy[i]);    /* final y */

        free(yt);
        free(dy);
        free(dys);
}
```

* * *