BACHELOR

Synchronisation in multilayer networks of Hindmarsh-Rose neurons

van den Broek, Matthijs G.

Award date:
2024

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Dynamics and Control

Synchronisation in multilayer networks of Hindmarsh-Rose neurons

Bachelor End Project

Matthijs van den Broek

1620061

Supervisors:
Dr. Ir. Erik Steur

Eindhoven, January 2024

Abstract

Many biological and technological systems can be modelled as multilayer networks. Contrary to single-layer networks, multilayer networks can model different types of interactions in a system. This research studies synchronisation in multilayer networks of Hindmarsh-Rose oscillators through numerical simulation and by means of an experimental setup, which consists of electronic circuit board realisations of the Hindmarsh-Rose neuron.

Furthermore, the multilayer connection graph method is applied to multilayer networks of Hindmarsh-Rose oscillators with the aim to predict synchronisation based on the oscillator properties and the network structure. It is discovered that this method does not account for some of the oscillator behaviour that is shown, and thus does not always yield correct predictions. For this reason, an extension of the multilayer connection graph method is proposed which allows for a conservative but correct prediction of synchronisation in a multilayer network of Hindmarsh-Rose oscillators.

Contents

Chapter 1

Introduction

Synchronisation is a phenomenon that can be observed in many places. Examples range from fireflies flashing in unison [1], an audience all clapping in the same rhythm [2], to metronomes aligning their swing when they stand on a shared platform [3]. What these examples have in common, is that they start in a state of asynchrony, where their rhythms do not line up. However, after some time they somehow align their frequencies and start behaving in unison. This phenomenon of synchronisation is caused by a coupling between the synchronising parts. In the example of the metronomes, each metronome pushes a bit on the platform with each swing, thereby influencing the other metronomes until they have found a common rhythm.

Synchronising systems are often modelled in the form of networks, where a node in the network represents a part of the system (such as a firefly, clapping person or metronome), and a connection (or edge) between two nodes means these parts of the system are coupled. Usually research focuses on single-layer networks with only one type of coupling in the system. However, in many applications there can be multiple types of coupling in a network, such as neurons that interact via electrical as well as chemical synapses [4], or diseases spreading through different kinds of interactions between people [5]. These systems require adding other types of coupling via extra layers to the network model, thereby creating a multilayer network.

1.1 Synchronisation analysis

In single-layer networks with diffusive coupling it is understood well when synchronisation occurs: typically when the strength of the coupling in the network exceeds a threshold value [6]. This threshold value can be predicted from the dynamics of the individual nodes of the network, and the network structure (network topology). There are multiple different methods that can predict the synchronisation threshold, such as the master stability function and the connection graph method.

The master stability function, developed by Pecora and Carroll [7] in 1998, predicts synchronisation via the eigenvalues of the Laplacian matrix of the network (see section 3.1). However, for complicated networks it is often impossible to calculate the eigenvalues analytically [6]. For this reason Belykh et al. [6] developed the connection graph stability method, which predicts synchronisation based on the lengths of the paths passing through a certain edge. This method can be used analytically in more complicated networks than the master stability function, and can also be used to study networks with time-dependent coupling [6].

Although both methods are very useful in single-layer networks, their use in multilayer networks is very limited. Only in a few specific cases (for example when the Laplacian matrices of the two layers commute) can the master stability function be applied to a multilayer network [8]. To predict synchronisation in multilayer networks in general, Belykh et al. [9]

have developed the multilayer connection graph method (MCG method), an extension of the connection graph method for single-layer networks. Much like the master stability function and connection graph method, the MCG method predicts the synchronisation threshold from the individual node properties and network topology. Until now the research into the MCG method has mostly been theoretical [9] [10]. Therefore, the focus of this research is to numerically and experimentally verify the validity of the multilayer connection graph method.

1.2 Verifying the MCG method

To reach the research goal of assessing whether the MCG method can accurately predict the synchronisation threshold in multilayer networks, a combination of numerical and experimental methods is used and applied to Hindmarsh-Rose (HR) oscillators. In particular, we aim to create numerical and experimental stability diagrams, showing when a certain network of HR oscillators synchronises depending on the coupling strengths in the different layers of the network. Then, using the theory of the MCG method we try to reconstruct these stability diagrams from just the properties of the individual nodes and the network topology. Furthermore, the MCG method is applied to a new network to see how well its prediction matches the numerical simulations and experimental measurements for this network.

To this end, a multilayer network simulation environment has been created in Matlab, which simulates a given network and assesses whether or not it synchronises for a given coupling strength. Also, an existing experimental setup that consists of electronic circuit board realisations of the mathematical Hindmarsh-Rose neuron has been adjusted, such that it can be used to test multilayer networks. This included soldering new components to the circuit boards and adjusting the existing C++ code that computes the coupling functions of the circuit boards.

1.3 Report structure

In this report, it will be assessed whether the predictions of synchronisation in multilayer networks of Hindmarsh-Rose neurons via the multilayer connection graph method match well with the results obtained by numerical simulations and experimental measurements. In chapter 2, a general explanation of multilayer networks, Hindmarsh-Rose neurons and synchronisation is given, along with a supporting example that is used in the rest of the report to clarify the discussed concepts. Chapter 3 describes the development of the Matlab simulation environment, and shows the result of the numerical simulation when applied to the supporting example. The experimental setup is discussed in chapter 4, highlighting the adjustments that have been made to the setup, and the notion of practical synchronisation. This chapter also shows the result of the experiment conducted with the network from the supporting example. In chapter 5 the multilayer connection graph method is applied to Hindmarsh-Rose neurons, an extension to the multilayer connection graph method is proposed, and the multilayer connection graph method and its extension are used to generate a prediction of the stability diagram for the supporting example. It is shown that with the new extension to the multilayer connection graph method it is possible to make a conservative but correct prediction of synchronisation in a multilayer network. The application of all relevant concepts to a new, slightly more complicated network is discussed in chapter 6, where it can be seen that also for this network a conservative but correct prediction can be made. Lastly, in chapter 7 the conclusion from this research and recommendations for future steps are outlined.

Chapter 2

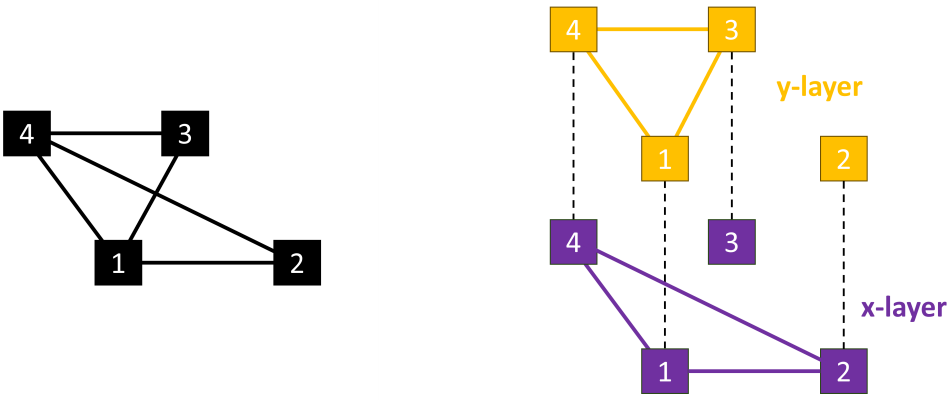
The network model

Before synchronisation in multilayer networks can be considered, first we must understand what networks are and how they are built up. This research investigates synchronisation in networks consisting of Hindmarsh-Rose oscillators that interact via diffusive coupling over multiple layers. Therefore, this chapter will give an introduction to multilayer networks, Hindmarsh-Rose oscillators and synchronisation. Also the supporting example that will be used throughout the report will be presented at the end of this chapter.

2.1 Multilayer networks

Any network consists of nodes that are interconnected by edges. Each edge connects two nodes to each other, meaning these nodes can interact. Depending on the type of network, these interactions can all be of the same type, which gives a single-layer network, or there might be multiple types of interactions, which gives a multilayer network. In the case of a network of neurons, the interactions on one layer might be electric, and on the other layer chemical [4]. An example of a single-layer network with 4 nodes and 5 edges can be seen in Figure 2.1a. A similar example of a multilayer network with 4 nodes and 6 edges can be seen in Figure 2.1b.

Different types of multilayer networks exist. In this research however, only multiplex networks are considered, meaning that the same nodes are represented in each layer, although



(a) Single-layer network

(b) Multilayer network

Figure 2.1: Two examples of networks.

the coupling between them may differ per layer [11]. The network in Figure 2.1b is an example of a multiplex network. The edges used in the networks are undirected, meaning that when node 1 is coupled to node 2, node 2 is automatically also coupled to node 1.

2.2 Hindmarsh-Rose oscillators

In this research the nodes in the network are Hindmarsh-Rose (HR) oscillators. This is a mathematical model of the action potential of a neuron, first described by J.L. Hindmarsh and R.M. Rose in 1984 [12]. As in the research by Gorissen [10], the state $p_i(t)$ of each neuron i is represented by three variables: $p_i(t) = (x_i(t), y_i(t), z_i(t))$, where $x_i(t)$ represents the membrane potential of the neuron, and $y_i(t)$ and $z_i(t)$ represent internal states [13].

The differential equations of the HR model that will be used are presented in equation (2.1). These are representative of the experimental setup that will be used later, and are identical to the model used by Steur et al. [13], except for a scaling by a factor $1/100$ which makes all timescales 100 times longer. This is done to make the differential equations less stiff and therefore computationally more feasible when solving them numerically.

$$\begin{aligned} \dot{x}_i &= -x_i^3 + 3x_i - 8 + 5y_i - z_i + E + u_{i,x} \\ \dot{y}_i &= -x_i^2 - 2x_i - y_i + u_{i,y} \\ \dot{z}_i &= 0.005(4x_i + 4.472 - z_i) \end{aligned} \tag{2.1}$$

The parameter E is a constant that defines the operating mode of the neuron without any external input (so $u_{i,x} = u_{i,y} = 0$), where the membrane potential of the neuron can be either constant, bursting (a spike followed by a period of rest), chaotic (multiple spikes followed by a period of rest), or tonic (constant spiking) [13]. Unless specified otherwise, this research uses a value of $E = 3.3$, for which the HR neuron operates in chaotic mode. $u_{i,x}$ and $u_{i,y}$ in equation (2.1) represent the coupling of the neurons via the x- or y-layer, respectively. They are defined as

$$\begin{aligned} u_{i,x} &= -\sigma_x \cdot \sum_{j=1}^{n} c_{ij}(x_i - x_j), \\ u_{i,y} &= -\sigma_y \cdot \sum_{j=1}^{n} d_{ij}(y_i - y_j), \end{aligned} \tag{2.2}$$

where σ_x and σ_y are the uniform coupling strengths in the x- and y-layer respectively. The coupling strength determines the extent to which nodes are coupled. A high coupling strength on a certain edge indicates a strong connection between the two nodes. And vice versa, when the coupling strength on an edge is zero, the two nodes are not coupled at all (this is equivalent to the edge not being there at all). In this research, the coupling strength within a layer is always uniform, meaning that the coupling strength of all edges in one layer is the same (so $\sigma_{x,k} = \sigma_x$ and $\sigma_{y,l} = \sigma_y$ for every edge k in the x-layer and every edge l in the y-layer). The uniform coupling strengths σ_x and σ_y of the x- and y-layer are not necessarily the same. Furthermore n is the number of nodes in the network, and c_{ij} and d_{ij} indicate whether or not a coupling exists between node i and node j in the x- or y-layer, respectively. When an edge is present between node i and node j in the x-layer $c_{ij} = 1$, and otherwise $c_{ij} = 0$. The same holds for d_{ij} in the y-layer.

A plot of the time trajectory of a single HR neuron with $E = 3.3$ can be seen in Figure 2.2. It can be observed that this neuron operates in chaotic mode. Since this neuron is not part of a network, there are no connections to other neurons, and thus $u_{i,x} = u_{i,y} = 0$.

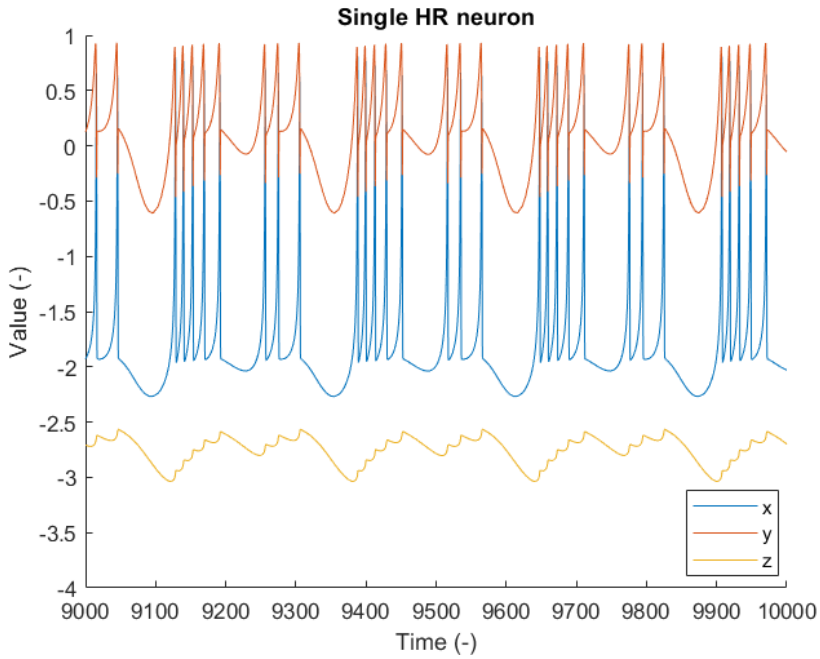


Figure 2.2: The time trajectory of a single Hindmarsh-Rose neuron. The x-component represents the membrane potential of the neuron, while the y- and z-component represent internal states.

2.2.1 Oscillator types

In single-layer networks, there are three ways in which oscillators can synchronise [9]. The class of Type I oscillators, to which Hindmarsh-Rose oscillators belong [9], synchronise for any coupling strength above a certain threshold. Type II oscillators also synchronise above a certain threshold, but eventually lose their synchronisation as the coupling gets increased further [9]. Oscillators that belong to Type III are incapable of synchronising at all [9].

2.3 Synchronisation of multilayer networks

Just like a single-layer network, a multilayer network can synchronise under certain conditions. This means that the nodes of the network will eventually oscillate in unison. Different types of synchronisation are used in literature, such as global synchronisation [9] (where each node in the network synchronises), and partial synchronisation [13] (where only part of the nodes in the network synchronise). In this research whenever *synchronisation* is mentioned, this can be understood as the asymptotic match of the states of all nodes in the network (which is a form of global synchronisation).

In general, a network must be connected to be able to synchronise, meaning that each node in the network can be reached from every other node in the network via one or more edges. After all, if a part of the network is not connected to the rest, there will be no coupling and synchronisation can never occur. The network in Figure 2.1a is an example of a connected single-layer network. Contrary to single-layer networks, the layers in a multilayer network do not have to be connected, as multiple disconnected layers can form a connected network together when all edges are projected onto one layer. For example when all edges in Figure 2.1b are projected onto a single layer, the connected network in Figure 2.1a is obtained. Although in the bottom layer (x-layer) of Figure 2.1b node 3 is not connected to any other node, the total network is still connected because this node does have connections in

the top layer (y-layer).

Also networks of Hindmarsh-Rose neurons are capable of synchronising when the coupling between them is strong enough. In this research we are interested in global synchronisation, where the states $p_i(t) = (x_i(t), y_i(t), z_i(t))$ of all HR oscillators in the network are synchronised, and become equal to the synchronous solution $s(t) = (x(t), y(t), z(t))$. The definition of synchronisation is therefore given by equation (2.3).

$$p_1(t) = p_2(t) = ... = p_n(t) = s(t) \tag{2.3}$$

From equations (2.1) and (2.2) it can be shown that the coupling of the oscillators through the terms $u_{i,x}$ and $u_{i,y}$ contributes to synchronisation. Say node i and node j are connected via the x-layer, then the difference between their derivatives $\dot{x}_i$ and $\dot{x}_j$ is given by

$$\begin{aligned}\dot{x}_i - \dot{x}_j &= -x_i^3 + x_j^3 + 3x_i - 3x_j + 5y_i - 5y_j - z_i + z_j - 2\sigma_x(x_i - x_j) \\ &= -2\sigma_x(x_i - x_j) + \Delta(x_i, x_j, y_i, y_j, z_i, z_j).\end{aligned}$$

As x, y and z remain within a certain range, the term $|\Delta(x_i, x_j, y_i, y_j, z_i, z_j)| \leq P$ where P is some finite number. For large values of σ_x, the term $-2\sigma_x(x_i - x_j)$ will dominate over $\Delta(x_i, x_j, y_i, y_j, z_i, z_j)$, so $\dot{x}_i - \dot{x}_j \approx -2\sigma_x(x_i - x_j)$. This is a differential equation for which it can be shown that $x_i - x_j \to 0$ for $t \to \infty$. In reality, $x_i - x_j$ will never completely converge to 0 due to the presence of the disturbing term $\Delta(x_i, x_j, y_i, y_j, z_i, z_j)$, but $|x_i - x_j|$ will become increasingly small for increasing σ_x. In a similar manner it can be shown that for two nodes connected via the y-layer $\lim_{t\to\infty} |y_i - y_j|$ will become increasingly small for increasing values of σ_y.

An example of synchronisation in a network of Hindmarsh-Rose oscillators is shown in section 2.4.

2.4 Supporting example

Throughout this report many different concepts are introduced. To clarify these concepts, a simple supporting example is used. This example network consists of 3 nodes, and has one x-edge between nodes 2 and 3, and one y-edge between nodes 1 and 2, see Figure 2.3. This network is a very simple example of a multilayer network where both the x- and y-layer are not connected, but the total network is. This means that coupling via both layers is necessary (i.e. $\sigma_x \neq 0 \wedge \sigma_y \neq 0$) to obtain global synchronisation in the network. For example when $\sigma_x = 0$ and $\sigma_y \neq 0$, node 1 and 2 are coupled to each other, but node 3 is not coupled to any other node (recall that when the coupling strength on a certain edge is zero, this is equivalent to the edge not being there at all). In this case, node 1 and 2 can synchronise when σ_y is high enough, but node 3 cannot. As not all nodes in the network synchronise, we do not have global synchronisation, and thus do not consider the network to be synchronised.

Figure 2.4 shows an example of synchronisation in the network of Figure 2.3 for a coupling strength $\sigma_x = \sigma_y = 0.5$. The figure shows that initially $p_1 \neq p_2 \neq p_3$, but over time the nodes synchronise until $x_1 = x_2 = x_3$, $y_1 = y_2 = y_3$ and $z_1 = z_2 = z_3$.

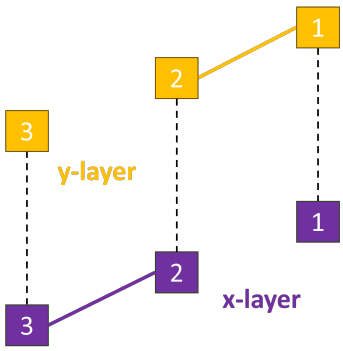


Figure 2.3: The 3 node network that is used as supporting example throughout the report. Adapted from the research by Gorissen [10].

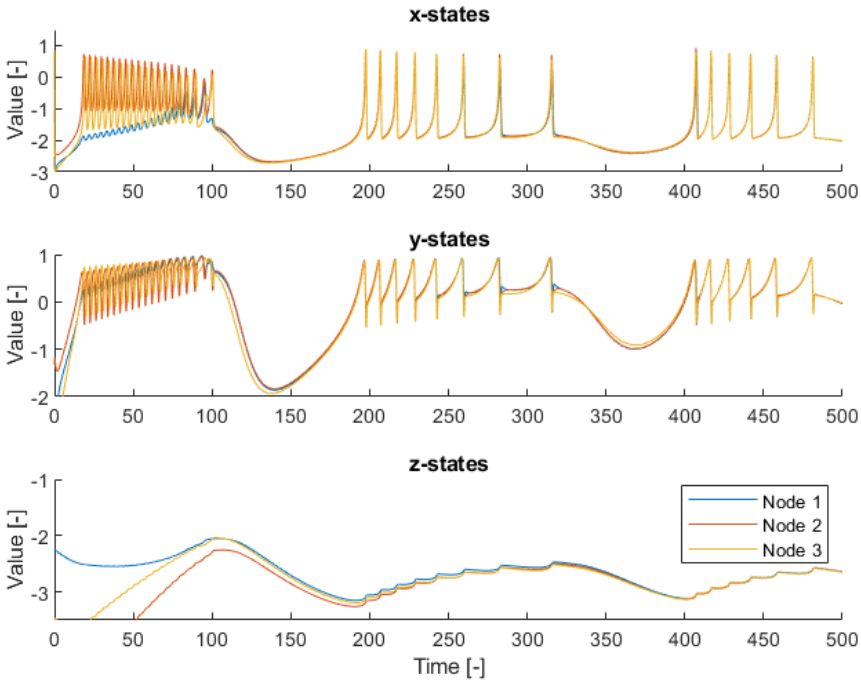


Figure 2.4: The time trajectories for the network of three Hindmarsh-Rose neurons depicted in Figure 2.3. Each plot has time on the x-axis. The x-, y- and z-component of each node synchronise over time for a coupling strength of $\sigma_x = \sigma_y = 0.5$.

Chapter 3

Numerical simulation

As we have seen in the previous chapter, each node in the multilayer network of Hindmarsh-Rose neurons is essentially a system of differential equations, given by equation (2.1). But how do we determine whether these differential equations synchronise? One of the possible answers to this question is numerical simulation in Matlab. This chapter discusses the development of the Matlab simulation environment, and shows the results of the numerical simulation of the supporting example. The concepts explained in section 3.1 are applied to the supporting example from Figure 2.3 in section 3.2.

3.1 Matlab simulation environment

The starting point for a numerical simulation is the network topology (network structure). The network topology of a layer k can be expressed by means of an adjacency matrix A_k. For a network with n nodes, A_k is an $n \times n$-matrix where the matrix element $A_{k,ij} = 1$ when node i and node j are connected and $A_{k,ij} = 0$ otherwise. Each layer in the multilayer network will have its own adjacency matrix. Since only undirected networks are considered in this research, the matrix A_k will be symmetric, as a connection between node i and j means there is also a connection between node j and i, so $A_{k,ij} = A_{k,ji}$. The elements on the diagonal are always equal to 0, as a node cannot be connected to itself.

Next, the degree matrix D_k of each layer is determined. This matrix contains the number of connections of each node on its diagonal, and thus shows the degree of connectedness for each node.

From the adjacency matrix and degree matrix the Laplacian L_k can be computed as

$$L_k = D_k - A_k. \tag{3.1}$$

The Laplacian is then used to define the external inputs u_i in equation (2.1), according to

$$\begin{aligned} \vec{u}_x &= -\sigma_x L_x \vec{x}, \\ \vec{u}_y &= -\sigma_y L_y \vec{y}, \end{aligned} \tag{3.2}$$

where $\vec{u}_x = [u_{1,x} u_{2,x} \dots u_{n,x}]^T$ and $\vec{u}_y = [u_{1,y} u_{2,y} \dots u_{n,y}]^T$ are the external input vectors, and $\vec{x} = [x_1 x_2 \dots x_n]^T$ and $\vec{y} = [y_1 y_2 \dots y_n]^T$ are the vectors that contain the current x- and y-state variables of the nodes.

With the network topology known, the network of Hindmarsh-Rose neurons can be simulated by solving the differential equation (2.1) in Matlab using ode23s. The network is simulated for 10000 time units with the default Matlab-solver error tolerances (relative error tolerance of $1 \cdot 10^{-3}$ and absolute error tolerance of $1 \cdot 10^{-6}$). Synchronisation is determined based on the behaviour in the last 1000 time units of the simulation. When the difference

between any two nodes is at some point in the last 1000 time units larger than the synchronisation tolerance, 0.06 for the x-state and 0.01 for the y- and z-state, the network is said not to synchronise. Due to the numerical nature of the simulations there will always be a small difference between the nodes, even in a synchronising network. The tolerance for the x-variable is larger than for the y- and z-variables, because the x-variable has the largest range and highest steepness. This causes a larger difference between the nodes, also in a synchronising network. The tolerances have been chosen such that they are approximately 10x larger than the difference in a very well-synchronised network.

The initial condition for each node is chosen randomly in the space ($x \in [-3,1], y \in [-6,2], z \in [-6,-1]$). This is an approximation of the positive invariant space of the Hindmarsh-Rose oscillator. As almost any network is able to synchronise when the initial conditions of the nodes are close enough together, each network is simulated 5 times for each pair of coupling strengths (σ_x, σ_y) but with different initial conditions. Only when the network synchronises all 5 times, it is said to synchronise for the pair (σ_x, σ_y).

To get a view of where in the (σ_x, σ_y)-space a network synchronises, stability diagrams are created (see for example Figure 3.1 in subsection 3.2.1). These diagrams show for which coupling strengths the network has stable synchronisation (green points in the diagram), and for which it does not (yellow points). To create a stability diagram, the (σ_x, σ_y)-space is split in a coarse grid where the synchronisation of every grid point is assessed as described above. When the four corner points of a certain element in the diagram are not of the same type (either synchronising or non-synchronising), the element is flagged and used in the next iteration. Inside the flagged elements a new finer grid is defined and the synchronisation of each point is assessed again. This method assumes that when all corner points of a certain element are of the same type, the points inside the elements are most likely of that type as well. Depending on the number of grid points in each iteration, we may choose to do an additional third iteration.

The Matlab simulation code used to determine synchronisation of multilayer networks and create stability diagrams, can be found in Appendix A.

3.2 Application to the supporting example

To clarify the concepts that have been explained in this section, they will be applied to the network from the supporting example of Figure 2.3. Since this is a 3-node network, $n = 3$ and A is a 3×3-matrix. The adjacency matrices for the x- and y-layer are given by A_x and A_y, respectively.

$$A_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad A_y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For example, the second row of matrix A_x shows the connections of node 2 via the x-layer, and it can be seen that node 2 is only connected to node 3. Since the network is undirected, node 3 is also connected to node 2, and therefore the third row also contains a 1 in the second position.

Next, the degree matrices are computed by counting the number of connections each node has. This number is put on the diagonal of the degree matrix:

$$D_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad D_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

It can be seen that on the x-layer nodes 2 and 3 have 1 connection, while node 1 has no connections.

Then, the Laplacian matrices are computed as:

$$L_x = D_x - A_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \qquad L_y = D_y - A_y = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Now according to equation (3.2), we can compute the external inputs on the x-layer $\vec{u}_x$ to be

$$\begin{bmatrix} u_{1,x} \\ u_{2,x} \\ u_{3,x} \end{bmatrix} = -\sigma_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = -\sigma_x \begin{bmatrix} 0 \\ x_2 - x_3 \\ -x_2 + x_3 \end{bmatrix}$$

This formulation is equivalent to equation (2.2) when worked out. For the external inputs $\vec{u}_y$ a similar calculation can be made.

3.2.1 Result of the numerical simulation

The resulting stability diagrams from the simulation of the 3-node supporting example network are shown in Figure 3.1. This figure shows three different stability diagrams, with a different range for σ_y. Each diagram has σ_x on the x-axis.

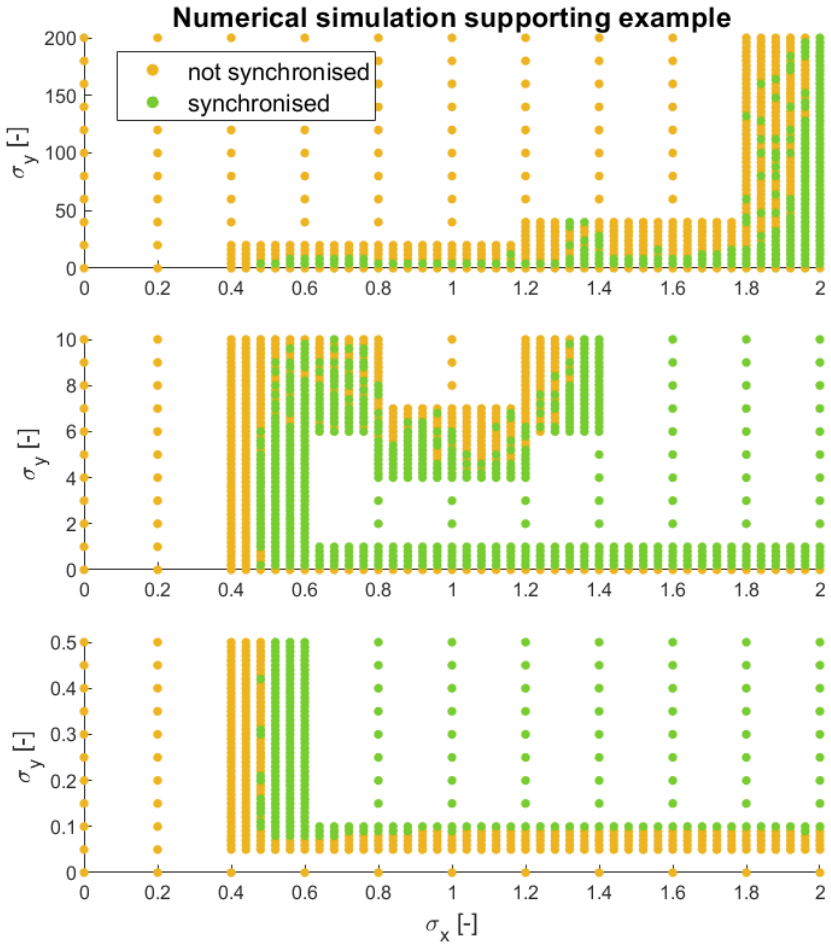


Figure 3.1: The resulting stability diagrams for the supporting example. Note that the three figures all have σ_x on the x-axis, but have different ranges for σ_y.

The stability diagrams are generated by splitting the initial domain of σ_x and σ_y into a coarse grid of 11×11 points. Then the corner points of each element are compared and flagged if they are not of the same type. The flagged elements are then used in a second iteration with grids of 6×6 points. In the top diagram, some of the bottom elements have been added manually, as the corner points are all of the same type.

From the stability diagrams it can be observed that there is an interesting interplay between the coupling strengths σ_x and σ_y. As expected, there is no synchronisation when either σ_x or σ_y is 0, because then the network is not connected. Furthermore, it stands out that a higher coupling strength does not always improve synchronisation. In the top diagram for $0.5 \leq \sigma_x \leq 1.9$, there is synchronisation for low values of σ_y, but this synchronisation breaks when σ_y becomes too large. This is reminding of the oscillator Type II behaviour, described in subsection 2.2.1. This pattern stops when σ_x becomes around 2, then a high value of σ_y does not break synchronisation anymore. Based on inspection of points outside the domain of this figure it is expected that the trends shown in the top figure continue outside of the figure (so for $\sigma_x > 2 \wedge \sigma_y > 0.1$ all points synchronise). The middle figure shows that also for σ_x, although less prevalent, a higher coupling strength is not always beneficial for synchronisation. For example for $(\sigma_x, \sigma_y) = (0.6, 7)$ there is synchronisation, but when σ_x is increased to 1.1, the synchronisation breaks. In the bottom diagram it can be seen that for $\sigma_x \leq 0.45$ or $\sigma_y \leq 0.07$ no synchronisation is possible at all.

At first sight the shape of the stability diagram may seem a bit unexpected or counter-intuitive, but in chapter 5 we explain by means of the multilayer connection graph method why the stability diagram looks like this.

3.3 Validity of numerical simulations

When interpreting the results of the numerical simulation, it is important to keep its limitations in mind. Most importantly, numerical simulations are always an approximation of the real solution with a certain error. It is believed that the used ode solver, Matlab's ode23s, is accurate enough for the purpose of this research, but nevertheless it has been observed that different ode solvers can give slightly different results. The choice for ode23s has been made because this solver seemed to give the most conservative synchronisation results (i.e. in the same network ode23s showed some non-synchronising regions where another solver already showed synchronisation), and this research aims to predict guaranteed synchronisation.

Furthermore, it can be seen in Figure 3.1 that at the boundary between the synchronised and non-synchronised region, this boundary is not always clearly defined (there are some yellow points in the green region and vice versa). This is because the points at the boundary are on the edge of synchronisation. They might synchronise for a lot of different initial conditions, but not all. These points might also be just inside the synchronisation tolerance during some time periods and be just outside the tolerance during other periods. This makes the exact boundary for synchronisation hard to distinguish, and it might also slightly differ between different simulation runs. Conclusions are therefore only drawn based on the general shape of the stability diagram, which is very consistent over multiple runs.

Chapter 4

Experimental research

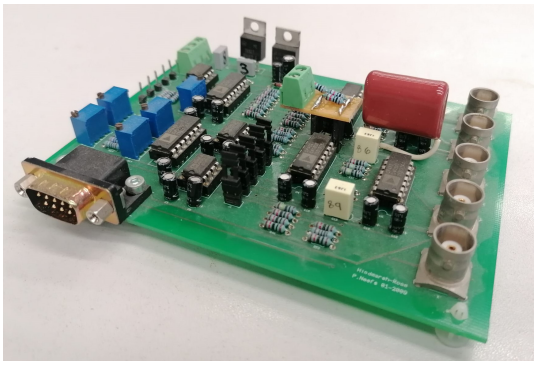
In chapter 3 it has been shown how synchronisation in a multilayer network can be numerically assessed. However, numerical simulations are only an approximation of what might happen in reality. Another option to assess synchronisation is through synchronisation experiments on a physical setup. This chapter describes this experimental setup and explains how the experiments work. Then the result of the experiment of the supporting example will be presented and it will be discussed whether it matches with the simulations. Lastly, the validity of the experiments will be considered.

4.1 Experimental setup

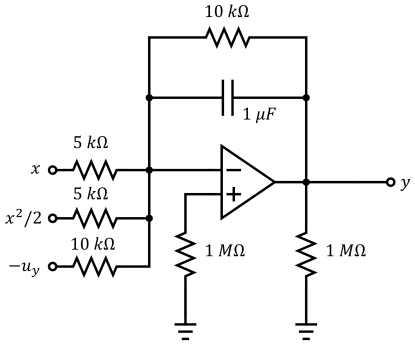
The set of differential equations presented in equation (2.1) has been derived from the set of differential equations in the research by Steur et al. [13] (shown in equation (4.1)) by scaling it by a factor $1/100$.

$$\begin{aligned} \dot{x}_i &= 100(-x_i^3 + 3x_i - 8 + 5y_i - z_i + E) + u_{i,x} \\ \dot{y}_i &= 100(-x_i^2 - 2x_i - y_i) + u_{i,y} \\ \dot{z}_i &= 0.5(4x_i + 4.472 - z_i) \end{aligned} \tag{4.1}$$

This original set of differential equations allows for an electronic circuit board realisation with off-the-shelf components [13], see Figure 4.1a.

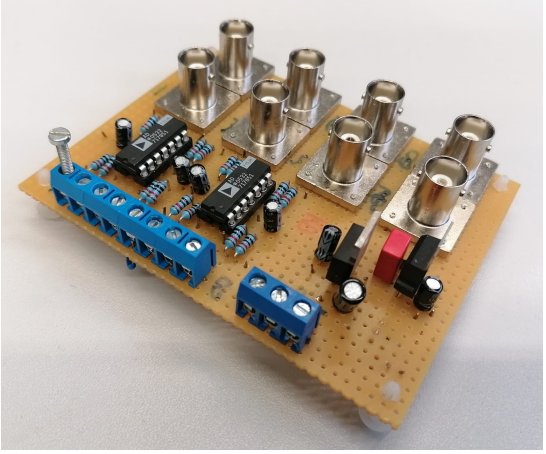


(a) An electronic circuit board realisation of the set of differential equations (4.1).

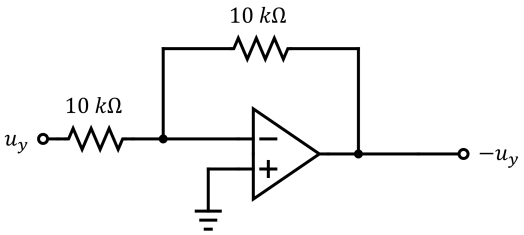


(b) The circuit diagram of the y-state. Figure adapted from Neefs et al [14].

Figure 4.1: The Hindmarsh-Rose electronic circuit board.



(a) The circuit board that allows for the external y-inputs $u_{i,y}$.



(b) The circuit diagram of the external y-input $u_{i,y}$.

Figure 4.2: The circuit board realisation of the y-coupling.

In previous research, this experimental setup has been used to study synchronisation in Hindmarsh Rose neurons in single-layer x-coupled networks. To use this setup for multilayer networks, the external y-input $u_{i,y}$ was added. The new circuit diagram for the y-state can be seen in Figure 4.1b. A total of 6 circuit boards has been adjusted, so networks of up to 6 nodes can be tested. The circuit diagrams for the x- and z-states can be found in the research by Neefs et al. [14], with as only change that the capacitance of the capacitors has been increased by a factor 10.

On the right of Figure 4.1a it can be seen that the Hindmarsh-Rose circuit board (HR circuit) has 5 coaxial ports, of which 3 are used to measure the states x, y and z. The other 2 ports are used to input the parameter E and the external x-input $u_{i,x}$. The external y-input $u_{i,y}$ is inserted via a separate circuit board, see Figure 4.2a, which is connected to the Hindmarsh-Rose circuit board. The circuit diagram for the y-input circuit board is shown in Figure 4.2b. This board can receive y-inputs for up to 8 Hindmarsh-Rose circuit boards, so the structure shown in the circuit diagram is present 8 times on the board.

The HR circuits are connected via a coupling interface. This interface receives the x- and y-states of each HR circuit via coaxial cables, and computes the external x- and y-inputs ($u_{i,x}$ and $u_{i,y}$, respectively) according to equation (2.2). The network structure is specified by the user in an adjacency matrix A_x for the x-layer and A_y for the y-layer. Also the coupling strengths σ_x and σ_y are user-specified. More details about the hardware that is used in the synchronisation experiments can be found in the research by Steur et al. [13].

4.2 Synchronisation experiments

With this experimental setup, it is possible to asses synchronisation in multilayer networks. By measuring the x- and y-states of each circuit in the network it is possible to determine whether the circuits synchronise for a pair of coupling strengths (σ_x, σ_y). However, because of the inherent imperfections in the circuits not all circuits are exactly identical, and therefore the difference between the circuits will also not be exactly 0 in a synchronised state [13]. Therefore, a definition of *practical synchronisation* is necessary. Two HR circuits are said to be practically synchronised when the Pearson correlation coefficient of their signals x_1 and x_2, and y_1 and y_2 is greater or equal to 0.99. The calculation for the correlation coefficient between x_1 and x_2 is given by

$$R_{x_1,x_2} = \frac{\sum_{i=1}^{n}(x_{1,i}-\bar{x}_1)(x_{2,i}-\bar{x}_2)}{\sqrt{\sum_{i=1}^{n}(x_{1,i}-\bar{x}_1)^2}\sqrt{\sum_{i=1}^{n}(x_{2,i}-\bar{x}_2)^2}} \geq 0.99, \tag{4.2}$$

where n is the number of samples in the signal, $x_{1,i}$ and $x_{2,i}$ are the values of the i-th sample in the signal, and $\bar{x}_1$ and $\bar{x}_2$ are the mean of the signal [15]. The calculation for R_{y_1,y_2} is similar. This definintion of practical synchronisation is taken from the research by Castanedo-Guerra et al. [15].

Synchronisation is determined based on signals obtained over a period of 10 seconds, with a sampling frequency of 1000 Hz. The measurement time of 10 seconds is comparable to the 1000 time units used to determine synchronisation in the simulations described in chapter 3. The HR circuits are allowed to settle their behaviour for 10 seconds before each measurement, to ensure no transient behaviour is recorded after adjusting the coupling strength.

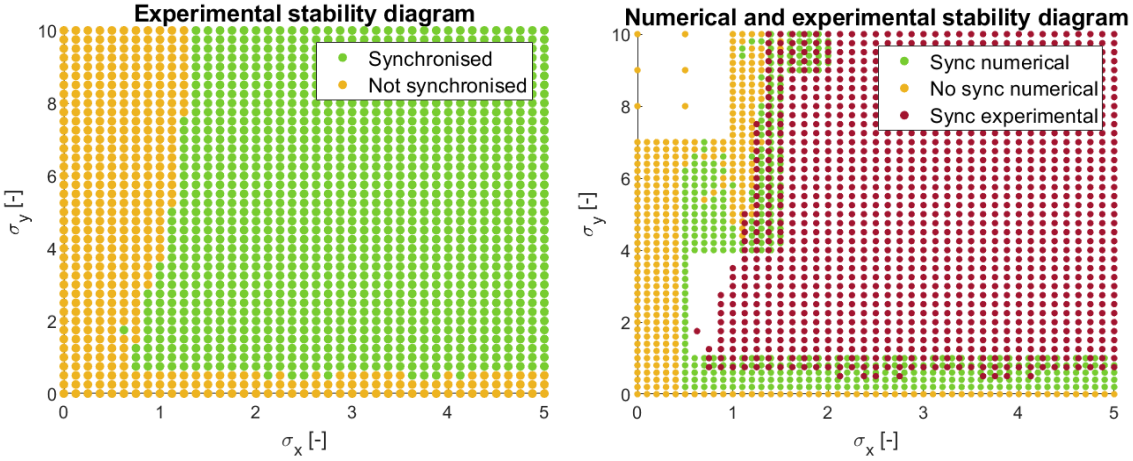
It is now possible to make stability diagrams similar to those from numerical simulation by trying a lot of different combinations for (σ_x, σ_y). To this end, a LabVIEW measurement protocol has been created that systematically tries all combinations of (σ_x, σ_y) within a user-specified range, and saves the resulting x- and y-signals of each circuit. During post-processing in Matlab it is determined when circuits synchronise and a stability diagram is created.

4.3 Application to the supporting example

The 3 node network of the supporting example from Figure 2.3 is input by specifying its adjacency matrices A_x and A_y (see section 3.2). The range for σ_x and σ_y is chosen as $\sigma_x \in [0,5], \sigma_y \in [0,10]$, with a resolution of 41×41 measurement points. The resulting experimental stability diagram is shown in Figure 4.3a.

This figure shows that also in the experimental setup we see the behaviour associated with Type II oscillators, where for example for $\sigma_x = 1$ we see synchronisation when $\sigma_y \geq 0.75$, but the synchronisation breaks when $\sigma_y \geq 3.75$. Figure 4.3b shows the result of the numerical simulation with the same range for σ_x and σ_y, and the synchronising points from the experiment are plotted in the same figure in red. Compared to the numerical stability diagram, it can be seen that in the experiment synchronisation starts from a higher coupling strength than in the simulation ($(\sigma_x, \sigma_y) = (0.75, 0.75)$ for the experiment compared to $(\sigma_x, \sigma_y) = (0.48, 0.10)$ for the simulation). This is logical, as in the simulation all nodes in the network are exactly identical while in the experiment the inherent small differences between the circuits make synchronisation more difficult.

Furthermore it stands out that in the simulated stability diagram we see a pattern around $\sigma_y = 6$, where we first have synchronisation for $0.52 \leq \sigma_x \leq 0.76$, then no synchronisation for $0.80 \leq \sigma_x \leq 1.32$, and then synchronisation again for $\sigma_x \geq 1.36$. This pattern is not visible in the experimental stability diagram. However, still the simulation has a larger synchronisation region in the coupling parameter space, as there are (almost) no points (σ_x, σ_y) for which there is synchronisation in the experimental diagram but not in the simulation diagram, as can be seen from Figure 4.3b.



(a) The experimental stability diagram for the 3 node suppporting example network shown in Figure 2.3.

(b) The synchronising points of the experimental stability diagram (red) plotted over the stability diagram of the numerical simulation.

Figure 4.3: The resulting stability diagram for the 3 node supporting example.

4.4 Validity of the experiments

Just like simulations, also experiments have their drawbacks. First of all, the HR circuits are not completely identical and consequently the resulting stability diagram depends on which circuits are used in the experiment. When three other HR circuits are used, the resulting stability diagram is slightly different. At $\sigma_y = 10$ for example, the onset of synchronisation now happens at $\sigma_x = 1.625$ instead of $\sigma_x = 1.375$.

Also, the definition of practical synchronisation that is used is very important. Another definition of practical synchronisation, used in the research by Steur et al. [13], says that two signals x_1 and x_2 are practically synchronised when for all times i after the transient

$$|x_{1,i} - x_{2,i}| \leq 0.25, \tag{4.3}$$

where $x_{1,i}$ and $x_{2,i}$ are the values of the signal at time i. When this practical synchronisation definition is used, the stability diagram in Figure 4.3a does not change too much. However, the stability diagram for the same network but with different HR circuits becomes wildly different and hardly shows any synchronisation at all.

Lastly, it has been observed that synchronisation at a certain point in the stability diagram can depend on whether or not there was synchronisation at the previous measurement point. For a certain network with $\sigma_y = 10$ the coupling strength σ_x can be increased to 1.0 without synchronisation appearing. When the coupling strength is then further increased to $\sigma_x = 2.0$ the nodes in the network synchronise. But if now the coupling strength is lowered back again to $\sigma_x = 1.0$, the synchronisation persists. This shows that for certain points around the synchronisation threshold, synchronisation can be non-deterministic. It is however believed that this phenomenon has little effect on the experimental stability diagrams, as in the experiment the coupling strength is always going from low to high.

Chapter 5

Multilayer connection graph method

In chapter 3 and chapter 4 two different approaches (numerical and experimental) have been used to create stability diagrams for synchronisation. For networks of small size these methods might be good enough, but when larger networks are considered this becomes very complex. For this reason, we would like to have a method to predict the synchronisation of a multilayer network based on the properties of the individual node and the network topology, analogous to the methods for single-layer networks. In 2019, Belykh et al. [9] developed the multilayer connection graph method (MCG method) which aims to do exactly this. The MCG method has already been applied to Hindmarsh-Rose neurons by Gorissen [10], so this research is used as a basis.

In this chapter, first the steps of the MCG method and its application to Hindmars-Rose oscillators will be explained. Then, an extension to the MCG method is proposed that predicts a maximum coupling strength for synchronisation. Lastly, the MCG method and its extension will be applied to the supporting example and the results will be presented.

5.1 The multilayer connection graph method

The multilayer connection graph method consists of four steps [9] [10], each of which is either related to the network topology, or the oscillator properties.

1. Calculate traffic loads. Traffic loads are a measure of how heavily each edge in the network is used, and therefore relates to the network topology.
2. Determine the 2-node behaviour. This step is related to the oscillator properties and determines what coupling strength is required to synchronise two nodes connected by a single x-edge, single y-edge or both.
3. Quantify the desynchronising effect of interlayer traffic. This oscillator property determines how much additional coupling strength is required when an edge is used in an interlayer path.
4. Combine all contributing factors determined in steps 1 to 3 to find a threshold coupling strength for synchronisation.

These 4 steps will each be explained in more detail in the next sections.

5.1.1 Traffic loads

Traffic loads are a measure of how heavily a certain edge in the network is used, i.e. how many nodes are connected to each other via this edge. Edges with a high traffic load are more critical for synchronisation than edges with a lower traffic load [10].

To determine the traffic load for a certain edge, we must first find the set of paths that run through the network. Any node i in the network is connected to a node j via a path P_{ij}, with $j > i$ (since all edges are undirected, this means that node j is also connected to node i). We will distinguish here between paths that go through only one layer (intralayer paths), and paths that pass through multiple layers (interlayer paths). The intralayer traffic load b_k^x of an x-edge k (b_k^y in case of a y-edge) is then determined by summing the lengths $|P_{ij}|$ of all shortest intralayer paths going through the edge k [9]:

$$\begin{aligned} b_k^x &= \sum_{j>i;k\in P_{ij}\in C}^{n} |P_{ij}|, \\ b_k^y &= \sum_{j>i;k\in P_{ij}\in D}^{n} |P_{ij}|. \end{aligned} \tag{5.1}$$

Here C and D are the set of all shortest paths that only pass through the x- or y-layer, respectively. The same goes for the interlayer traffic load b_k^{int} for an edge k. Here we sum the lengths of all paths that go through both the x- and y-layer and pass through edge k [9]:

$$b_k^{int} = \sum_{j>i;k\in P_{ij}\notin C,D}^{n} |P_{ij}|. \tag{5.2}$$

5.1.2 2-node synchronisation thresholds

Next, the oscillator behaviour in a two-node network must be characterised by finding the four constants a_x, a_y, ω_x and ω_y. This two-node behaviour is seen as representative for the behaviour of connected nodes in the full network [10]. The synchronisation threshold for the coupling strength of two nodes connected only by an x- or y-edge is denoted by σ_x^* or σ_y^*, respectively. This means that for any coupling strength $\sigma_x \geq \sigma_x^*$, the two nodes connected via an x-edge only will synchronise. The constants a_x and a_y can now be obtained by doubling the values for σ_x^* and σ_y^* (i.e. $a_x = 2\sigma_x^*$ and $a_y = 2\sigma_y^*$) [9].

To find the values for ω_x and ω_y the behaviour of two nodes coupled by both an x- and y-edge must be observed. (ω_x,ω_y) represents a pair of coupling strengths $\sigma_x = \omega_x$ and $\sigma_y = \omega_y$ for which the two nodes synchronise, with $\omega_x \leq a_x$ and $\omega_y \leq a_y$. Belykh et al. [9] mention that this choice of (ω_x,ω_y) is somewhat arbitrary as there are many combinations that will give synchronisation, but it is generally a good idea to choose ω_x and ω_y non-zero, and somewhere in the middle of their range [9].

For two coupled Hindmarsh-Rose oscillators, the stability diagram obtained using numerical simulations is shown in Figure 5.1. Along the x-axis of this figure $\sigma_y = 0$, so this is equivalent to two HR neurons coupled via an x-edge only. Along this axis it can be read that $\sigma_x^* \approx 0.5$. Along the y-axis of the figure $\sigma_x = 0$, and this is therefore equivalent to coupling via a y-edge only. Here it can be read that $\sigma_y^* \approx 0.06$, which gives that for a network of HR oscillators $a_x = 1.0$ and $a_y = 0.12$. As it is not exactly known what the influence of ω_x and ω_y is on the final result, multiple combinations of (ω_x, ω_y) within the allowed range will be tried.

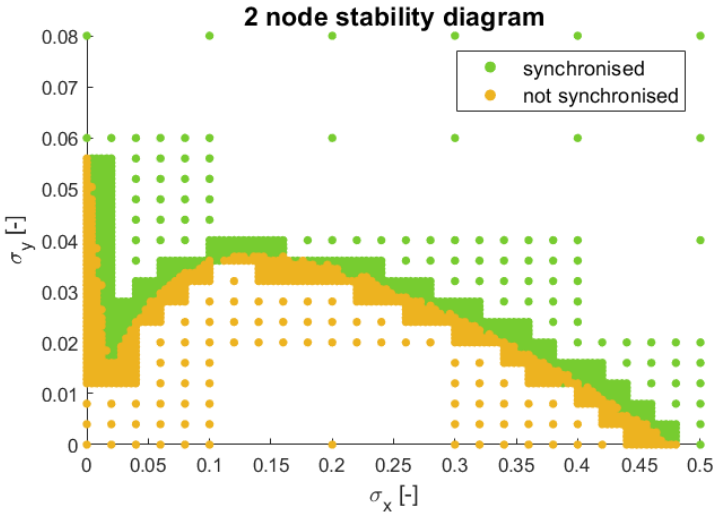


Figure 5.1: The stability diagram for a network of two Hindmarsh-Rose oscillators coupled via both an x- and y-edge.

5.1.3 Interlayer traffic desynchronisation

An important factor in determining whether a certain edge can synchronise in a multilayer network is interlayer traffic. Interlayer traffic plays a destabilising role, and if the interlayer traffic on a certain edge is too high it is possible that the edge cannot be synchronised at all (even for very high coupling strengths) [9]. To compensate for this destabilising effect, Belykh et al. [9] propose to add a term α_x to the synchronisation criterion for σ_x, and add a term α_y to the synchronisation criterion for σ_y. The magnitude of these terms α_x and α_y for a certain edge is therefore (among others) dependent on the interlayer traffic going through that edge.

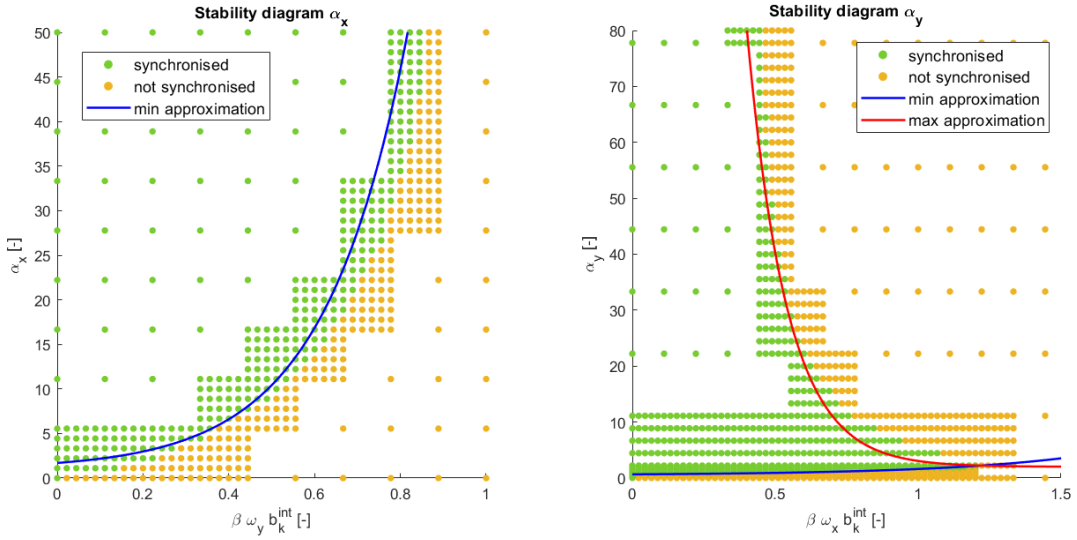
To quantify the destabilising effect of interlayer traffic in Hindmarsh-Rose neurons two auxiliary stability systems are considered. The auxiliary stability system corresponding to two nodes connected by an x-edge is given by

$$\begin{aligned}
\dot{x}_i &= -x_i^3 + 3x_i - 8 + 5y_i - z_i + E - 0.5 \cdot (a_x + \alpha_x) \cdot \sum_{j=1}^{n} c_{ij}(x_i - x_j), \\
\dot{y}_i &= -x_i^2 - 2x_i - y_i + 0.5 \cdot A \cdot \sum_{j=1}^{n} d_{ij}(y_i - y_j), \\
\dot{z}_i &= 0.005(4x_i + 4.472 - z_i),
\end{aligned} \tag{5.3}$$

and the auxiliary stability system corresponding to two nodes connected by a y-edge is given by

$$\begin{aligned}
\dot{x}_i &= -x_i^3 + 3x_i - 8 + 5y_i - z_i + E + 0.5 \cdot B \cdot \sum_{j=1}^{n} c_{ij}(x_i - x_j), \\
\dot{y}_i &= -x_i^2 - 2x_i - y_i - 0.5 \cdot (a_y + \alpha_y) \cdot \sum_{j=1}^{n} d_{ij}(y_i - y_j), \\
\dot{z}_i &= 0.005(4x_i + 4.472 - z_i),
\end{aligned} \tag{5.4}$$

where $A = \beta\omega_y b_k^{int}$ and $B = \beta\omega_x b_k^{int}$ [10]. These systems represent the HR dynamics presented in equation (2.1), but the coupling strengths are now replaced such that one of the



(a) Auxiliary stability diagram α_x. Note that the x-axis shows $A = \beta\omega_y b_k^{int}$.

(b) Auxiliary stability diagram α_y. Note that the x-axis shows $B = \beta\omega_x b_k^{int}$.

Figure 5.2: Numerically simulated stability diagrams for the auxiliary stability systems in equations (5.3) and (5.4). The blue and red lines approximate the green synchronising region in the diagram.

layers actively destabilises the system. In equation (5.3) the term $+0.5A$ replaces the coupling strength $-\sigma_y$, and because of the difference in sign this term is now destabilising instead of stabilising. This destabilisation has to be compensated for in the x-layer by the term $-0.5(a_x+\alpha_x)$. The goal here is to find how much compensation α_x in the x-layer is necessary to correct for the interlayer traffic destabilisation in the y-layer, represented by A. Essentially the same thing happens in auxiliary system (5.4), but now the x-layer is destabilised by the term $0.5B$, and this should be compensated by $-0.5(a_y+\alpha_y)$ in the y-layer.

It is possible to make numerically simulated stability diagrams for these auxiliary systems, in a similar manner as described in chapter 3. The resulting stability diagrams can be seen in Figure 5.2. These plots show the destabilising term, $A = \beta\omega_y b_k^{int}$ or $B = \beta\omega_x b_k^{int}$, plotted against the stabilising term, α_x or α_y. To be able to use the results from these diagrams in the synchronisation criteria in subsection 5.1.4, the boundary between the synchronising and non-synchronising region is approximated as

$$\alpha_x \approx 0.7e^{5.2\beta\omega_y b_k^{int}} + 1, \tag{5.5}$$

$$\begin{aligned} \alpha_{y,min} &\approx 0.15e^{2\beta\omega_x b_k^{int}} + 0.5 \qquad \text{for } \beta\omega_x b_k^{int} \leq 1.22, \\ \alpha_{y,max} &\approx 1400e^{-7.2\beta\omega_x b_k^{int}} + 2 \qquad \text{for } \beta\omega_x b_k^{int} \leq 1.22. \end{aligned} \tag{5.6}$$

5.1.4 Synchronisation criteria

With all the values determined in the steps above, everything can now be put together to determine a criterion for synchronisation. For each edge $k \in C$ (x-layer edges) and each edge $l \in D$ (y-layer edges), respectively the first or second equation [9] below is evaluated.

$$\begin{aligned} \sigma_{x,k} &> \sigma_{x,k,min} = \frac{1}{n}[\gamma_1 a_x b_k^x + \beta\omega_x b_k^{int} + \alpha_x] \\ \sigma_{y,l} &> \sigma_{y,l,min} = \frac{1}{n}[\gamma_2 a_y b_l^y + \beta\omega_y b_l^{int} + \alpha_{y,min}] \end{aligned} \tag{5.7}$$

Here, n is the number of nodes in the network, and γ_1, γ_2 and β are constants between 0 and 1, added to compensate for overestimates made in the derivation of the synchronisation criterion [9]. In this research $\gamma_1 = \gamma_2 = \beta = 1$ is used, which corresponds to a conservative estimation for the threshold coupling strength. This is to make sure that there is guaranteed synchronisation. The actual threshold for the coupling strength may therefore be lower than calculated using this criterion.

Since the coupling strength in each layer is uniform (i.e. $\sigma_{x,k} = \sigma_x$ for each edge k in the x-layer, and $\sigma_{y,l} = \sigma_y$ for each edge l in the y-layer), the uniform coupling strength in that layer must be larger than the maximum between the synchronisation thresholds of the individual edges (so $\sigma_x \geq \sigma_{x,min} = \max_k \sigma_{x,k,min}$ and $\sigma_y \geq \sigma_{y,min} = \max_l \sigma_{y,l,min}$).

5.2 Extending the MCG method

The MCG method stability criterion in equation (5.7) as proposed by Belykh et al. [9], predicts synchronisation when both σ_x and σ_y are above their calculated threshold value. That is because in its current version the MCG method is meant to be applied to Type I oscillators, which in a single-layer network synchronise once their coupling strength exceeds a threshold value. For this reason, Belykh et al. expected that also in a multilayer network there would be synchronisation once both σ_x and σ_y are larger than their threshold. However, in both the numerical simulation in chapter 3 and the experiment in chapter 4, Type II-like behaviour has been observed for low values of σ_x, where synchronisation can disappear if the coupling strength σ_y becomes too large. When σ_x eventually becomes large enough, this Type II behaviour disappears.

The cause for the observed Type II behaviour is the destabilising interlayer traffic, which has a dual role. On the one hand, it introduces the Type II behaviour which breaks synchronisation at higher coupling strengths. This behaviour can also be observed in the stability diagram for α_y in Figure 5.2b. On the other hand, the interlayer traffic ensures that synchronisation can take place at lower coupling strengths than in a single-layer network (the supporting example of Figure 2.3 would need a coupling $\sigma_x = 0.97$ if it had only x-edges and a coupling of $\sigma_y = 0.13$ if it had only y-edges, while now a coupling of $(\sigma_x, \sigma_y) = (0.52, 0.08)$ is sufficient). At higher values of σ_x the Type II behaviour disappears altogether. This can be explained according to the logic presented in section 2.3, where it is shown that two nodes connected by an x- or y-edge can always be synchronised for high values of σ_x or σ_y, respectively. If every node in the network is synchronised with the other nodes it is connected to in the network, this means that the whole network must be synchronised (provided that the network is connected).

In its current form, the multilayer connection graph method does not account for an upper bound to σ_y. This is not a problem, as long as the synchronisation prediction always lies in the Type I region of the stability diagram ($\sigma_x \geq 2$ in the case of the supporting example). However, it turns out that this is not always the case, as can be seen in section 5.3. This means that based on the criteria in equation (5.7), synchronisation cannot be guaranteed. Therefore an addition to the MCG method is proposed, that predicts an upper bound for synchronisation to $\sigma_{y,l}$ based on the auxiliary stability diagram for α_y. The new stability criterion for $\sigma_{y,l}$ is shown in equation (5.8). The stability criterion for $\sigma_{x,k}$ remains unchanged and can be found in equation (5.7).

$$\sigma_{y,l,min} < \sigma_{y,l} < \sigma_{y,l,max} = \frac{1}{n}[\gamma_2 a_y b_l^y + \beta \omega_y b_l^{int} + \alpha_{y,max}] \tag{5.8}$$

Since we have a uniform coupling strength σ_y, the maximum allowable coupling strength may not be exceeded for any of the y-edges. Therefore, the allowable range for the uniform coupling strength σ_y is now determined according to $\sigma_{y,min} \leq \sigma_y \leq \sigma_{y,max} = \min_l \sigma_{y,l,max}$.

Not for all combinations of (ω_x, ω_y) equation (5.8) yields a reliable prediction that guarantees synchronisation. Therefore, different combinations for ω_x and ω_y are tried system-

atically, and the most conservative prediction (i.e. the lowest predicted $\sigma_{y,max}$]) is then taken as predicted upper boundary. As the synchronisation criteria for σ_x and the lower bound of σ_y, presented in equation (5.7), have been proven formally [9] for any (ω_x, ω_y), the most optimistic (i.e. lowest) prediction will be taken as predicted boundary for those.

5.3 Application to the supporting example

To show how the predictions from the extended MCG method perform on the supporting example from Figure 2.3 we only still need to calculate the traffic loads in the network, as the other quantities in the synchronisation criteria have already been determined and are valid for Hindmarsh-Rose neurons in any network structure.

In this network there are 3 paths that connect each node to every other node in the network, namely P_{12}, P_{13} and P_{23}. Path P_{12} is an intralayer path over the y-layer with length 1, path P_{13} is an interlayer path with length 2, and path P_{23} is an intralayer path over the x-layer with length 1. We can now calculate the traffic loads for the two edges (edge 12 between node 1 and 2 and edge 23 between node 2 and 3) in the network to be

$$\begin{aligned} b_{12}^{y} &= |P_{12}| = 1, \\ b_{12}^{int} &= |P_{13}| = 2, \\ b_{23}^{x} &= |P_{23}| = 1, \\ b_{23}^{int} &= |P_{13}| = 2. \end{aligned}$$

Note that $b_{12}^{x} = b_{23}^{y} = 0$ because nodes 1 and 2 are not connected via the x-layer, and nodes 2 and 3 are not connected via the y-layer.

All variables in equations (5.7) and (5.8) have now been determined. For $\omega_x \in [0, a_x]$ and $\omega_y \in [0, a_y]$, a 20×20 grid is created, and for every combination of (ω_x, ω_y) that is synchronising in Figure 5.1, the stability criteria are evaluated. The result is shown in Figure 5.3.

The green and orange points in this figure are obtained by numerical simulation. The blue points are $(\sigma_{x,min}, \sigma_{y,min})$ for different combinations of (ω_x, ω_y) and are thus the result from the original MCG method stability criterion in equation (5.7). It can be seen that all blue points lie in the green synchronising region, which shows that the points from this stability criterion indeed guarantee synchronisation for every combination of ω_x and ω_y. However, the MCG method also predicts synchronisation for all $\sigma_y \geq \sigma_{y,min}$, and since the blue points lie in the Type II region it can be seen that this is not true (as there are orange non-synchronising points above the blue points). For this reason, the figure also shows the red points $(\sigma_{x,min}, \sigma_{y,max})$ which are the result from the extension of the MCG method. These points do not all lie in the synchronising region, and thus synchronisation is not guaranteed for every combination of ω_x and ω_y. The black box in the figure indicates the region where synchronisation is predicted based on the extended MCG method. It can be seen that this black predicted region lies entirely in the green simulated synchronising region. Although this prediction is correct, it is also apparent that it is a conservative prediction, meaning that the coupling strength σ_y can actually become a lot larger before synchronisation disappears.

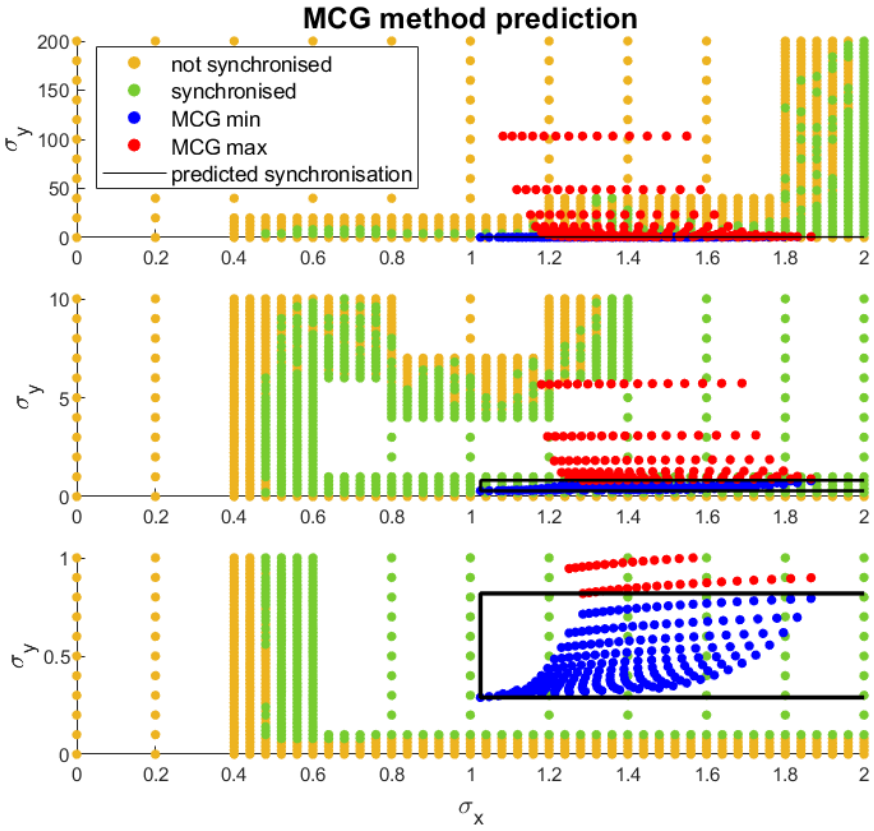


Figure 5.3: The prediction from the extended MCG method (red, blue and black) for the supporting example network from Figure 2.3, plotted over the results of numerical simulation (green and orange). The y-axis has a different scale for σ_y in each of the plots, the x-axis shows σ_x in each plot.

Chapter 6

Application to a new network

In chapter 5 it has been shown that the extended multilayer connection graph method yields a correct prediction for synchronisation in the 3 node supporting example network. To show that this method works in other cases as well, it will be applied to a new network. First, this new network will be introduced and its traffic loads will be presented. Then the result from the extended MCG method will be shown and compared to the results from numerical simulation. Unfortunately, due to time limitations no experiments could be performed for this network.

6.1 The network

So far, the extended MCG method has only been tested on the very simple supporting example network from Figure 2.3. To see how the method performs on a more complex network, it is tested on the network shown on the left of Figure 6.1. This network has been taken from the research by Gorissen [10], where this network was the result of the greedy optimisation over the y-layer.

The traffic loads in this network are calculated by means of the Matlab code written by Gorissen [10], and are as follows:

$$\begin{aligned} b^x_{12} &= 1, & b^{int}_{12} &= 4,\\ b^x_{34} &= 5, & b^{int}_{34} &= 2,\\ b^x_{36} &= 3, & b^{int}_{36} &= 2,\\ b^x_{45} &= 3, & b^{int}_{45} &= 2,\\ b^x_{56} &= 1, & b^{int}_{56} &= 2,\\ b^y_{15} &= 1, & b^{int}_{15} &= 6,\\ b^y_{23} &= 1, & b^{int}_{23} &= 6. \end{aligned}$$

The other variables in the stability criteria from equations (5.7) and (5.8) have already been determined in chapter 5 and are used in the prediction of the MCG method.

6.2 Result

The resulting prediction from the extended MCG method is presented on the right of Figure 6.1. This plot is similar to Figure 5.3, where the green and orange parts are the result of the numerical simulation, the blue and red points indicate respectively the predicted minimum and maximum values for the coupling strength, and the black region indicates the predicted region of synchronisation.

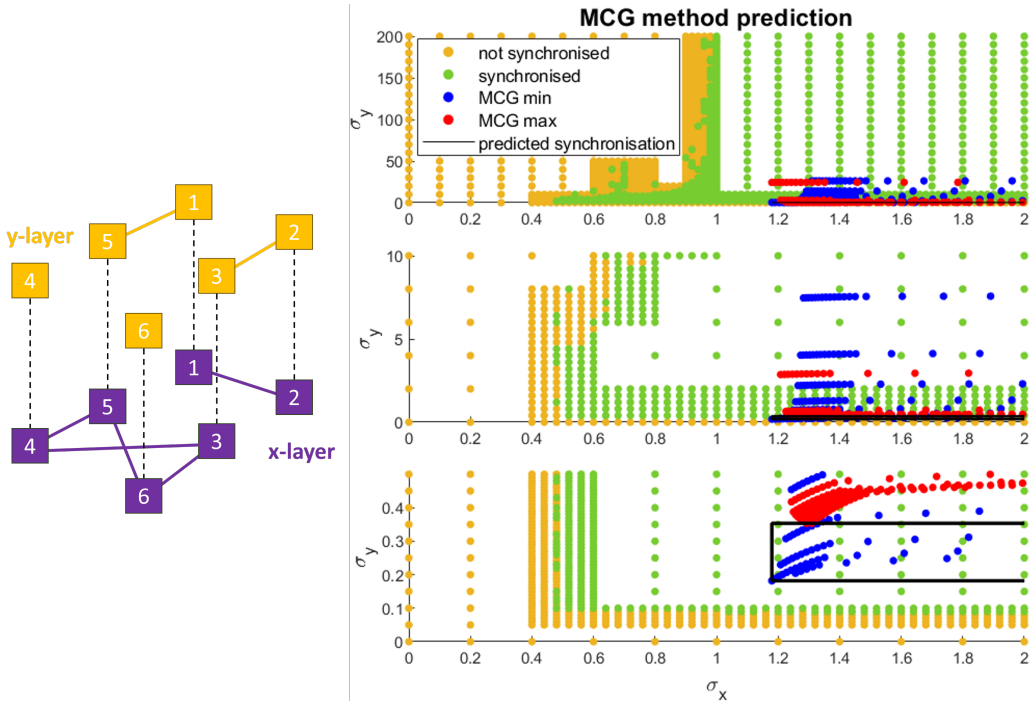


Figure 6.1: (left) The network that is used to test the performance of the extended MCG method. Figure adapted from the research by Gorissen [10]. (right) The prediction from the extended MCG method, plotted over the results from the numerical simulation. Each plot has σ_x on the x-axis, and a different range for σ_y on the y-axis.

From the figure it can be seen that also for this network, the extended MCG method gives a correct prediction that guarantees synchronisation, as all points within the black region synchronise. However, this example also shows once more that the prediction by the extended MCG method is very conservative, as there is a large region with synchronising points that is not included in the predicted region. As can be seen from the top plot in Figure 6.1, in this case it would not even be necessary to use the maximum value for σ_y, as the prediction from the MCG method lies in the Type I region, meaning that all points above the black region are also synchronised.

Chapter 7

Conclusion and recommendations

Synchronisation in multilayer networks of Hindmarsh-Rose neurons has been studied in this research. To this end, a Matlab simulation environment (provided in Appendix A) has been created that can produce stability diagrams for an x- and y-coupled multilayer network of Hindmarsh-Rose neurons. The findings from these simulations have been verified using an experimental setup consisting of electronic circuit board realisations of the Hindmarsh-Rose neuron. These circuit boards have been adjusted to allow for coupling via the y-layer, in addition to the already existing coupling via the x-layer.

The results from the simulations and experiments are quite consistent, with as main difference that synchronisation starts for slightly higher coupling strengths in the experimental setup. This finding is expected, as the imperfections in the HR circuit boards slightly hinder the onset of synchronisation compared to the idealised simulations. In both the simulations and the experiments it has been observed that the Hindmarsh-Rose oscillator, which belongs to the class of Type I oscillators, shows Type II-like behaviour when used in a multilayer network at a relatively low coupling strength σ_x. At higher coupling strengths this behaviour disappears.

To predict synchronisation in multilayer networks based on the oscillator properties and the network topology, the multilayer connection graph method [9] has been applied to networks of Hindmarsh-Rose oscillators. However, it turned out that this method does not always yield correct results that guarantee synchronisation. This is due to the Type II behaviour shown at low coupling strengths that is unaccounted for in the MCG method. In order to make correct predictions that guarantee synchronisation, an extension to the multilayer connection graph method has been developed which predicts a maximum to the coupling strength σ_y. With this extended multilayer connection graph method it is possible to make a correct but conservative prediction of synchronisation in a multilayer network.

7.1 Recommendations

Related to the synchronisation prediction by the multilayer connection graph method, there are a few things that would improve this prediction. First of all, in the current method the prediction for the maximum allowable coupling strength $\sigma_{y,max}$ is instated for all values of σ_x, while it is known that at some value of σ_x the Type II behaviour disappears. If this value for σ_x would be known, a much larger region of synchronisation can be predicted correctly as the predicted maximum to the coupling strength can be removed in this region. Secondly, it would be beneficial to know how and why the maximum to the coupling strength σ_y increases for increasing σ_x in the region showing Type II behaviour. If this is understood better, it might be possible to make a more accurate prediction in the region of the stability diagram showing Type II behaviour.

To gain a better understanding about the points mentioned, it is recommended to do more research into the dynamics of the Hindmarsh-Rose neuron when used in a multilayer network, as it is still poorly understood how it is possible that a Type I oscillator like the Hindmarsh-Rose neuron shows Type II behaviour when placed in a multilayer network. Furthermore, where this research focused mostly on the study of the supporting example from Figure 2.3, it would be recommended to expand the range of the networks studied. This might provide more insight into where the region of Type II behaviour appears, and where it converts back to Type I behaviour.

Additionally, Belykh et al. [9] mention in their paper that it is possible to extend the multilayer connection graph method such that it can be used for Type II oscillators. Although the Hindmarsh-Rose oscillator is a Type I oscillator, it could still be useful to apply this MCG method for Type II oscillators to the HR oscillator as it shows some behaviour typical for Type II oscillators. When this method is developed, it might provide a more realistic prediction of the maximum coupling strength $\sigma_{y,max}$ than presented in this research.

Related to the experimental setup, it has been observed that the definition of practical synchronisation is very important for the results of the experiment. With the definition used in the research of Steur et al. [13] the resulting stability diagrams wildly vary between different circuits, while with the definition from Castanedo et al. [15] the results are much more consistent. Although this last definition seems to give good results, it was formulated based on synchronisation in single-layer x-coupled networks. It would be a good idea to do more verification that this definition is also suitable for use in multilayer networks.

Lastly, the current research has focused on networks with idealised edges without time-delay. However, in many real-world applications there is some time-delay in the coupling between nodes. In the future, when synchronisation in multilayer networks without time-delay is understood better, time-delay coupled multilayer networks could be an interesting new research area.

Bibliography

[1] J. Buck and E. Buck, "Synchronous Fireflies," *Scientific American*, vol. 234, no. 5, pp. 74–85, 1976. 1

[2] Z. Néda, E. Ravasz, Y. Brechet, T. Vicsek, and A.-L. Barabási, "The sound of many hands clapping," *Nature*, vol. 403, no. 6772, pp. 849–850, 2000. 1

[3] J. Pantaleone, "Synchronization of metronomes," *American Journal of Physics*, vol. 70, pp. 992–1000, 10 2002. 1

[4] A. E. Pereda, "Electrical synapses and their functional interactions with chemical synapses," *Nature Reviews Neuroscience*, vol. 15, no. 4, pp. 250–263, 2014. 1, 3

[5] G. F. de Arruda, E. Cozzo, T. P. Peixoto, F. A. Rodrigues, and Y. Moreno, "Disease Localization in Multilayer Networks," *Physical Review X*, vol. 7, p. 11014, 2 2017. 1

[6] V. N. Belykh, I. V. Belykh, and M. Hasler, "Connection graph stability method for synchronized coupled chaotic systems," *Physica D: Nonlinear Phenomena*, vol. 195, no. 1, pp. 159–187, 2004. 1

[7] L. M. Pecora and T. L. Carroll, "Master Stability Functions for Synchronized Coupled Systems," tech. rep., 1998. 1

[8] F. Sorrentino, "Synchronization of hypernetworks of coupled dynamical systems," *New Journal of Physics*, vol. 14, 3 2012. 1

[9] I. Belykh, D. Carter, and R. Jeter, "Synchronization in multilayer networks: When good links go bad," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 4, pp. 2267–2302, 2019. 1, 2, 5, 16, 17, 18, 19, 20, 21, 25, 26

[10] N. Gorissen, "Greedy optimisation of multilayer networks for synchronisation of Hindmarsh-Rose neurons," tech. rep., 2023. 2, 4, 7, 16, 17, 18, 23, 24

[11] A. Aleta and Y. Moreno, "Multilayer Networks in a Nutshell," *The Annual Review of Condensed Matter Physics*, vol. 10, pp. 45–62, 2019. 4

[12] J. L. Hindmarsh and R. M. Rose, "A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations," Tech. Rep. 1222, 1984. 4

[13] E. Steur, C. Murguia, R. H. Fey, and H. Nijmeijer, "Synchronization and partial synchronization experiments with networks of time-delay coupled hindmarsh-rose neurons," *International Journal of Bifurcation and Chaos*, vol. 26, 6 2016. 4, 5, 12, 13, 15, 26

[14] P. J. Neefs, E. Steur, and H. Nijmeijer, "Network complexity and synchronous behavior an experimental approach," *International Journal of Neural Systems*, vol. 20, pp. 233–247, 6 2010. 12, 13

[15] I. Castanedo-Guerra, E. Steur, and H. Nijmeijer, "Synchronization of "light-sensitive" Hindmarsh–Rose neurons," *Communications in Nonlinear Science and Numerical Simulation*, vol. 57, pp. 322–330, 4 2018. 14, 26

Appendix A

Matlab simulation environment

The Matlab simulation environment consists of three scripts: stability_diagram.m, Sync_network.m and HR_network.m, each of which is nested inside the previous script. stability_diagram.m creates a stability diagram showing synchronisation in a given multilayer network of Hindmarsh-Rose neurons. Sync_network.m determines synchronisation at a given point (σ_x, σ_y) for this network. HR_network.m is the function providing the derivatives $\dot{x}$, $\dot{y}$ and $\dot{z}$ to the Matlab ode solver.

A.1 stability_diagram

```
% Create a stability diagram showing stable synchronisation for a
% multilayer network of Hindmarsh–Rose neurons in the (sigma_x,sigma_y)–plane
%
% User input
% Ax:           adjacency matrix of the x–layer of the network
% Ay:           adjacency matrix of the y–layer of the network
% n:            number of nodes in the network
% E:            defines operating mode of the Hindmarsh–Rose neuron
%               0 <= E <= 1.4 Resting
%               1.4 <= E <= 3.1 Bursting
%               3.1 <= E <= 3.5 Chaotic
%               E >= 3.5 Tonic
% simulation_nr: number of simulations to do per point to determine synchronisation
% filename:     location to save the sync and no_sync points
% rge_sx:       range for sigma x in the stability diagram
% rge_sy:       range for sigma y in the stability diagram
% res1:         number of grid points in each direction for the first
%               iteration. It will make a res1 x res1 grid
% res2:         number of grid points in each direction for the second
%               iteration
% iteration3:   boolean input, true will give a stability diagram with 3
%               iterations, false will give 2 iterations
%
% Output
% figure(1)     figure showing the stability diagram after one iteration
% figure(2)     figure showing the stability diagram after two iterations
% figure(3)     figure showing the stability diagram after three iterations
%               (optional)

%% Network properties

n = 3;              % Supporting example network
Ax = [0 0 0;
      0 0 1;
      0 1 0];
Ay = [0 1 0;
```

```
      1 0 0;
      0 0 0];

E = 3.3;              % Bursting mode
                      % 0 <= E <= 1.4 Resting
                      % 1.4 <= E <= 3.1 Bursting
                      % 3.1 <= E <= 3.5 Chaotic
                      % E >= 3.5 Tonic (Source: Sync. of light-sensitive
                      % HR-neurons)
simulation_nr = 5;    % Nr of simulations to do per network to determine synchronization

filename = 'C:\Users\....\Stability diagram\Data\3node_simple(5,10).mat';

% Stability diagram parameters
rge_sx = [0 5];       % Range for sigma x
rge_sy = [0 10];      % Range for sigma y
res1 = 11;            % Resolution first iteration, create a res1 x res1 grid
res2 = 6;             % Resolution second iteration, create a res2 x res2 grid on each
    subdomain
iteration3 = false;   % false will give 2 iterations, true will give 3 iterations

%% Making stability diagram first iteration

sx = linspace(rge_sx(1),rge_sx(2),res1);
sy = linspace(rge_sy(1),rge_sy(2),res1);

[Sx,Sy] = meshgrid(sx,sy);                  % Create a grid for the coupling strengths
stability = zeros(length(sy),length(sx));

no_sync = zeros(1,2);
sync = zeros(1,2);
no_sync_count = 1;
sync_count = 1;

parfor i = 1:length(sy)                     % Parallel for-loop
    stability_temp = zeros(1,length(sx));   % Temporary stability array (necessary for
        parallel computing)

    for j = 1:length(sx)
        sigma_x = Sx(i,j);
        sigma_y = Sy(i,j);

        % Determine if there is synchronisation at this point
        synchronisation = Sync_network(Ax,Ay,n,sigma_x,sigma_y,E,simulation_nr,false);

        if all(synchronisation) % Only if the point synchronises for all initial
            conditions we count it as synchronised
            stability_temp(1,j) = 2;        % 0: not simulated, 1: no synchronisation, 2:
                synchronisation
        else
            stability_temp(1,j) = 1;
        end

    end
    stability(i,:) = stability_temp;        % Put the temporary stability array in the
        general stability matrix
end

for i = 1:length(sy)        % Fill sync and no_sync with correct points
    for j = 1:length(sx)

        if stability(i,j) == 2
            sync(sync_count,1) = Sx(i,j);
            sync(sync_count,2) = Sy(i,j);
            sync_count = sync_count + 1;
        end
```

```
        if stability(i,j) == 1
            no_sync(no_sync_count,1) = Sx(i,j);
            no_sync(no_sync_count,2) = Sy(i,j);
            no_sync_count = no_sync_count + 1;
        end
    end
end

figure(1)
hold on
scatter(sync(:,1),sync(:,2),[],[0.4660 0.8 0.1880],'filled')       % Synchronized points
    are green
scatter(no_sync(:,1),no_sync(:,2),[],[0.9290 0.7 0.1250],'filled') % Non-synchronized
    points are yellow
title('Stability diagram 6-node example network')
xlabel('\sigma_x [-]')
ylabel('\sigma_y [-]')

%% Flag subdomains that need another iteration
flagged = zeros(1,4);
flag_count = 1;

for i = 1:length(sy)-1
    for j = 1:length(sx)-1
        % If all corner points are of the same type we do not need to do an
        % additional iteration
        if stability(i,j) == stability(i,j+1) && stability(i,j) == stability(i+1,j) &&
            stability(i,j) == stability(i+1,j+1)
            flagged = flagged;
        else
            flagged(flag_count,:) = [Sx(i,j), Sx(i,j+1), Sy(i,j), Sy(i+1,j)];
            flag_count = flag_count + 1;
        end
    end
end

fprintf('First iteration finished at %s \n',datestr(now,'HH:MM:SS'));

%% Second iteration of stability diagram
flagged2 = zeros(1,4);
flag2_count = 1;

for k = 1:size(flagged,1)
    rge_sx = flagged(k,1:2);       % Range for sigma x
    rge_sy = flagged(k,3:4);       % Range for sigma y

    sx = linspace(rge_sx(1),rge_sx(2),res2);
    sy = linspace(rge_sy(1),rge_sy(2),res2);

    [Sx,Sy] = meshgrid(sx,sy);                  % Create a grid of coupling strengths
    stability = zeros(length(sy),length(sx));

    parfor i = 1:length(sy)
        stability_temp = zeros(1,length(sx));
        for j = 1:length(sx)
            sigma_x = Sx(i,j);
            sigma_y = Sy(i,j);

            % Determine if there is synchronisation at this point
            synchronisation = Sync_network(Ax,Ay,n,sigma_x,sigma_y,E,simulation_nr,false
                );

            if all(synchronisation) % Only if the point synchronises for all initial
                conditions we count it as synchronised
                stability_temp(1,j) = 2;       % 0: not simulated, 1: no synchronisation
                    , 2: synchronisation
            else
```

```
                stability_temp(1,j) = 1;
            end

        end
        stability(i,:) = stability_temp;
    end

    for i = 1:length(sy)    % Fill sync and no_sync with correct points
        for j = 1:length(sx)

            if stability(i,j) == 2
                sync(sync_count,1) = Sx(i,j);
                sync(sync_count,2) = Sy(i,j);
                sync_count = sync_count + 1;
            end

            if stability(i,j) == 1
                no_sync(no_sync_count,1) = Sx(i,j);
                no_sync(no_sync_count,2) = Sy(i,j);
                no_sync_count = no_sync_count + 1;
            end
        end
    end


    for i = 1:length(sy)-1        % Flag subdomains for the third iteration
        for j = 1:length(sx)-1
            if stability(i,j) == stability(i,j+1) && stability(i,j) == stability(i+1,j)
                && stability(i,j) == stability(i+1,j+1)
                flagged2 = flagged2;
            else
                flagged2(flag2_count,:) = [Sx(i,j), Sx(i,j+1), Sy(i,j), Sy(i+1,j)];
                flag2_count = flag2_count + 1;
            end
        end
    end
    fprintf('Second iteration subdomain %d of %d checked at %s \n',k,size(flagged,1),
        datestr(now,'HH:MM:SS'));
end


%% Plot of second iteration

figure(2)
hold on
scatter(sync(:,1),sync(:,2),[],[0.4660 0.8 0.1880],'filled')       % Synchronized points
     are green
scatter(no_sync(:,1),no_sync(:,2),[],[0.9290 0.7 0.1250],'filled') % Non-synchronized
    points are yellow
title('Stability diagram after second iteration')
legend('synchronised','not synchronised')
xlabel('\sigma_x [-]')
ylabel('\sigma_y [-]')


%% Optional third iteration of stability diagram

if iteration3 == true
    for k = 1:size(flagged2,1)
        rge_sx = flagged2(k,1:2);       % Range for sigma x
        rge_sy = flagged2(k,3:4);       % Range for sigma y

        sx = linspace(rge_sx(1),rge_sx(2),6);
        sy = linspace(rge_sy(1),rge_sy(2),6);

        [Sx,Sy] = meshgrid(sx,sy);
        stability = zeros(length(sy),length(sx));
```

```
        parfor i = 1:length(sy)
            stability_temp = zeros(1,length(sx));
            for j = 1:length(sx)
                sigma_x = Sx(i,j);
                sigma_y = Sy(i,j);

                % Determine if there is synchronisation at this point
                synchronisation = Sync_network(Ax,Ay,n,sigma_x,sigma_y,E,simulation_nr,
                    false);

                if all(synchronisation) % Only if the point synchronises for all initial
                     conditions we count it as synchronised
                    stability_temp(1,j) = 2;       % 0: not simulated, 1: no
                        synchronisation, 2: synchronisation
                else
                    stability_temp(1,j) = 1;
                end

            end
            stability(i,:) = stability_temp;
        end

        for i = 1:length(sy)    % Fill sync and no_sync with correct points
            for j = 1:length(sx)

                if stability(i,j) == 2
                    sync(sync_count,1) = Sx(i,j);
                    sync(sync_count,2) = Sy(i,j);
                    sync_count = sync_count + 1;
                end

                if stability(i,j) == 1
                    no_sync(no_sync_count,1) = Sx(i,j);
                    no_sync(no_sync_count,2) = Sy(i,j);
                    no_sync_count = no_sync_count + 1;
                end
            end
        end

        fprintf('Third iteration subdomain %d of %d checked at %s \n',k,size(flagged2,1)
            ,datestr(now,'HH:MM:SS'));
    end


    %% Plot of third iteration

    figure(3)
    hold on
    scatter(sync(:,1),sync(:,2),[],[0.4660 0.8 0.1880],'filled')       % Synchronized
        points are green
    scatter(no_sync(:,1),no_sync(:,2),[],[0.9290 0.7 0.1250],'filled') % Non-
        synchronized points are yellow
    title('Stability diagram after third iteration')
    xlabel('\sigma_x [-]')
    ylabel('\sigma_y [-]')

end


%% Save sync and no_sync points to filename
if isfile(filename) % Check if the file already exists
     warning('This file already exists. File has been renamed')
     new_filename = strrep(filename,'.mat','_rename.mat');
     save(new_filename,'sync','no_sync')
else
     save(filename,'sync','no_sync')
```

```
end
```

A.2 Sync_network

```
% Determine synchronisation at a given point (sigma_x,sigma_y) for the
% multilayer network of Hindmarsh–Rose neurons
%
% Input
% Ax:               adjacency matrix of the x–layer
% Ay:               adjacency matrix of the y–layer
% n:                number of nodes in the network
% sigma_x:          coupling strength on the x–layer
% sigma_y:          coupling strenght on the y–layer
% E:                defines operating mode of the Hindmarsh–Rose neuron
% simulation_nr:    amount of times to simulate the network with different
%                   initial conditions
% make_plot:        boolean input, true will output figure(1) with a plot
%                   of the x–, y– and z–signals. false will output no
%                   figure
%
% Output
% synchronisation:  boolean array of size (1,simulation_nr). 1 (true)
%                   indicates synchronisation for that initial condition.
%                   0 (false) indicates no synchronisation

function synchronisation = Sync_network(Ax,Ay,n,sigma_x,sigma_y,E,simulation_nr,
    make_plot)

    % Check if the dimensions of the adjacency matrices correspond with the
    % number of nodes in the network
    if size(Ax,1)~=n || size(Ax,2)~=n || size(Ay,1)~=n || size(Ay,2)~=n
        error("The dimensions of Ax or Ay do not agree with n")
    end

    % Check if Ax and Ay are symmetric (as we only consider undirected
    % edges)
    if not(issymmetric(Ax)) || not(issymmetric(Ay))
        error("Matrix Ax or Ay is not symmetric")
    end

    Dx = zeros(n,n);
    Dy = zeros(n,n);
    for i = 1:n
        Dx(i,i) = sum(Ax(i,:));     % Nr of connections of each node on the diagonal
        Dy(i,i) = sum(Ay(i,:));
    end

    Lx = Dx - Ax;                   % Laplacian
    Ly = Dy - Ay;

    if rank(Lx+Ly) ~= n-1           % Check if there are no isolated parts in the
        network
        error("The network is not connected")
    end

    % Interval for initial conditions and simulation time interval
    x_range = [-3, 1];              % Interval for initial conditions
    y_range = [-6, 2];
    z_range = [-6, -1];
    tspan = [0 10000];

    synchronisation = true(1,simulation_nr);
    for k = 1:simulation_nr                     % Simulate a number of times for
        different initial conditions
```

```
        x0 = x_range(1) + (x_range(2)-x_range(1)).*rand(n,1);
        y0 = y_range(1) + (y_range(2)-y_range(1)).*rand(n,1);
        z0 = z_range(1) + (z_range(2)-z_range(1)).*rand(n,1);
        p0 = [x0; y0; z0];                      % p is a 3n by 1 vector

        % Solve ODE
        [t,p] = ode23s(@(t,p)HR_network(p,Lx,Ly,E,sigma_x,sigma_y,n),tspan,p0);

        % Check synchronization in final 1000 time units with difference matrices
        t_idx = find(t >= (tspan(2)-1000), 1);    % Find the index where the last 1000
            time units start
        x_final = p(t_idx:end,1:n);
        y_final = p(t_idx:end,n+1:2*n);
        z_final = p(t_idx:end,2*n+1:3*n);

        diff_x = zeros(n,n);                    % Top diagonal matrix with the largest
            difference between any two x-variables
        diff_y = zeros(n,n);
        diff_z = zeros(n,n);
        Ix = zeros(n,n);                        % Top diagonal matrix containing the
            index where the largest difference occurs
        Iy = zeros(n,n);
        Iz = zeros(n,n);

        for i = 1:n-1
            for j = i+1:n
                [diff_x(i,j),Ix(i,j)] = max(abs(x_final(:,i)-x_final(:,j)));
                [diff_y(i,j),Iy(i,j)] = max(abs(y_final(:,i)-y_final(:,j)));
                [diff_z(i,j),Iz(i,j)] = max(abs(z_final(:,i)-z_final(:,j)));
            end
        end

        tol_x = 0.06;       % Synchronisation tolerance for x
        tol_y = 0.01;       % Synchronisation tolerance for y
        tol_z = 0.01;       % Synchronisation tolerance for z

        if max(diff_x,[],'all') > tol_x || max(diff_y,[],'all') > tol_y || max(diff_z
            ,[],'all') > tol_z
            synchronisation(k) = false;
        end
    end

    % Plot
    if make_plot
        figure(1)

         hold on
         for i = 1:3*n
             plot(t,p(:,i))
         end
         xlim([9000 10000])
         xlabel('Time [-]')
         ylabel('Value [-]')
    end

end
```

A.3 HR_network

```
% Function that returns the derivatives of x, y and z for the
% Hindmarsh-Rose neuron to the Matlab odesolver
%
% Input
% p:        3n by 1 state vector [x1;...;xn;y1;...;yn;z1;...;zn]
```

```
% Lx:        Laplacian matrix of the x-layer
% Ly:        Laplacian matrix of the y-layer
% E:         defines operating mode of the Hindmarsh-Rose neuron
% sigma_x:   coupling strength on the x-layer
% sigma_y:   coupling strength on the y-layer
% n:         number of nodes in the network
%
% Output
% dpdt:      derivative of the state vector

function dpdt = HR_network(p,Lx,Ly,E,sigma_x,sigma_y,n)
    xn = p(1:n);
    yn = p(n+1:2*n);
    zn = p(2*n+1:3*n);

    % Hindmarsh-Rose differential equation
    xdot = (-xn.^3 + 3*xn - 8 + 5*yn - zn + E - sigma_x*Lx*xn);
    ydot = (-xn.^2 - 2*xn - yn - sigma_y*Ly*yn);
    zdot = 0.005*(4*xn + 4.472 - zn);

    dpdt = [xdot; ydot; zdot];
end
```