Technische Universiteit **Eindhoven** University of Technology

Department of Mechanical Engineering
Dynamics and Control

An application of pattern generation in diffusive networks: The linear peristaltic pump

4Z670 - Internship Thesis

R.J.R. van Kampen 0919572
r.j.r.v.kampen@student.tue.nl

Supervisor:
dr. A.Y. Pogromsky

Version 2

DC 2017.019

Eindhoven, January 2017

Abstract

The goal of the project is to find an application of pattern generation in diffusive networks within linear peristaltic pumping.

The most designs of linear peristaltic pumps have multiple actuation points which are driven by a cam. The phase difference between the actuation points is fixed. The current state of the art designs have multiple individually controlled actuators. Pattern generation in diffusive networks can be used to create a control or reference signal for each individual actuator. For this setup it is required that the following parameters of the network can be tuned or estimated:

- direction of the wave;
- phase difference between the output signals;
- the oscillation frequency of the wave;
- the amplitude of the wave.

Different network topologies are simulated and analyzed. From this study it follows that the unidirectional ring network is the most suitable within application. By using an unidirectional ring, the direction of the wave is in the same direction as the coupling is applied. Hence the direction of the wave can be chosen independently from the initial conditions.

The number of nodes k within the unidirectional ring must be odd. When k is odd, the phase difference between the output signal of node j and $j + 2$ and between the output signal of node 2 and k is $\frac{2\pi}{k}$.

The oscillation frequency of the system can be predicted when the coupling strength is chosen such that the closed loop linear system is close to the Hopf bifurcation.

The amplitude of the wave cannot be predicted, since it is depending on different parameters such as the coupling strength and the dynamics of the individual nodes. However normalizing the output and multiplying it with a gain factor will give the possibility to control the amplitude.

The shape of the generated wave-like pattern is mostly depending on the dynamics of the nodes. The effect of pole and zero placement of the linear feedback part has been examined. However it did not result in any design rules regarding shape of the wave.

The diffusive network used to control the linear peristaltic pump can be realized in two different ways. The first option is to use a real-time simulator with hardware in the loop to create the output or reference signal for the actuators. This option is the most suitable for prototyping, due to the possibility to change the dynamics of the network easily. Changing the dynamics of the nodes can be important for a study on the performance of a linear peristaltic pump.

When the required dynamics for an optimal performance are known, the network can be created by using electronics. The nodes can be made by using an electrical circuit with one or more feedback loops. The circuit of the actuator can be implemented within the dynamics of a single node, creating a more sophisticated design suitable for industrial applications.

Preface

When I received my bachelor in mechanical engineering at the Rotterdam University of applied science I was not satisfied. I learned a lot about heavy offshore equipment, but I wanted to learn more about robotics and control theory. Therefore I decided to go to Eindhoven University of Technology, which is in the "brainport" of the Netherlands, one of the leading technology regions.

Within my premaster, all my elective courses were related to robotics and control. Unfortunately I could not do my premaster thesis within the Dynamics and Control or Control System Technology research groups. However, now for my internship within my master program, it was possible to do it within the Dynamics and Control research group. This was a great pleasure.

After a few meetings, Sasha Pogromsky had a very interesting subject for my internship where theory and practice could meet. I really want to thank Sasha Pogromsky for giving me the opportunity to work on this interesting subject and his guidance during the internship. I hope this report will be a good contribution to further work of Sasha Pogromsky and the Dynamics and Control research group.

I hope you enjoy your reading.

Ricky van Kampen

Eindhoven, December 2016

Contents

List of Figures

List of Tables

1 Introduction

Synchronization of coupled dynamical systems is an important field of research, because synchronization is common in nature. Animals have good locomotor skills to move efficiently in complex environments. Their neural networks produce rhythmic coordinated patterns to achieve the locomotion. The interesting part is that the rhythmic coordinated patterns are caused without any rhythmic input. Different studies have shown that for lamprey and leeches the central pattern generators are distributed networks made of multiple coupled oscillatory centers. A review on neurobiological observations concerning locomotor pattern generators with high-dimensional rhythmic output signals while only receiving simple input signals, is made in the study of Ijspeert [1]. This study also covers some applications of central pattern generators for locomotion control in robotics. These examples are mainly complex nature like robotics like a salamander, snake or even a humanoid.

Another promising application for central pattern generators out of the scope of Ijspeert, is linear peristaltic pumping. Within nature the peristaltic motion is very common, for example peristaltic motions are used in the intestine to propel food. The advantage of a peristaltic pump is that the channel wall is the only part which makes contact with the fluid. Therefore it can be used to transport acid which would normally affect the impeller of a pump. Furthermore it can be used for applications with high requirements on hygienics. The wall of the channel could be easily cleaned or replaced with respect to the parts of a centrifugal pump.

The neural network can be used to control different actuators used for the compression and/or retraction of the (flexible) channel wall. By creating a wave-like motion with the actuators, the fluid inside the tube will move in the direction of the wave. By using the neural network to control the actuators, the design is easily scalable and adjustable since a decentralized controller can be used.

When applying the neural network in an application it is important to guarantee a certain behavior. In the study by Pogromsky et al. [2] the existence and stability in a network of diffusively coupled identical dynamical systems is examined. The study provides criteria on stability where subsystems without interconnections are globally asymptotically stable and the oscillatory behavior is forced via diffusive coupling. These criteria will be used to design a network where stability and oscillatory behavior can be guaranteed.

This report will first cover the current state of the art within peristaltic pumping. Than the diffusive networks and their requirements will be discussed to create a pattern. Furthermore a study on the effect of different network topologies on the output signal will be executed. After the effect of the different network topologies is known, a study is performed on the effect of the node dynamics on the output. With the knowledge about the possibilities to create different patterns, two designs of a peristaltic pump and a controller are proposed. One design for prototyping and a design which can be used in future applications. Furthermore an outlook of the project is given.

2 Current state of the art in linear peristaltic pumping

Trough the ages different types of linear peristaltic pumps are developed. Two different types of designs can be distinguished, designs whether the phase difference between the actuation points is fixed or it can be influenced.

2.1 Linear peristaltic pumps with a fixed phase difference between actuation points

The most peristaltic pumps with fixed phase actuation found in literature are patented, but a common thing among this patents is the use of a camshaft. These designs can be split up into two different types, where the camshaft is in direct contact with the tube and the designs where an intermediate body is involved.

A simple design where the camshaft is directly connected to the channel wall, is in patent [3]. The design is schematically shown in Figure 1. Multiple cams are mounted on the shaft. The cams are slightly twisted with respect to each other such that they are not completely compressing the channel at the same time. Rotating the shaft will result in the cams pressing against the channel wall at different times causing a wave-like motion. Since the cams are fixed to the shaft, the phase difference between the actuation points is also fixed. The frequency of the waveform can be changed by in- or decreasing the speed of the motor.

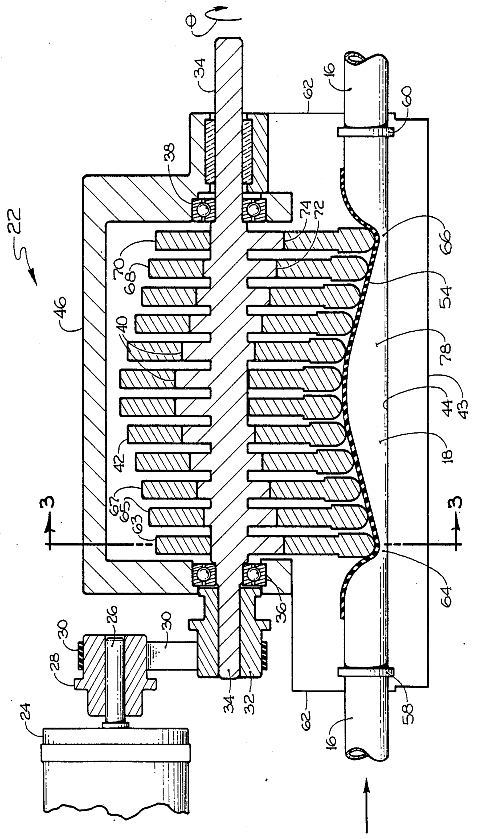


Figure 1: A design a of linear peristaltic pump with a fixed phase difference between the actuation points. The shaft (34) with the cams (67-69) is directly connected to the channel (78) [3].

Other designs where the cams are directly pressing against the channel are very similar to this design. A variation is in [4] and [5] where the massive cams are replaced by eccentrically mounted ball bearings to reduce the friction.

The other type of designs with fixed phase actuation are using intermediate bodies, which are driven by cams. The advantage of the intermediate bodies is that the face in contact with the channel wall can be optimized for pumping. In patent [6, 7, 8, 9, 10] and article [11] are different designs using intermediate bodies driven by camshafts.

Another design using camshafts with intermediate bodies is in the design of the RTU concrete mechanics laboratory [12]. Within this design a flexible tube goes through different chambers. Each chamber is filled with a fluid or gas and can be compressed by using cylinders. The intermediate bodies (pistons) are used to compress the medium. Compressing the medium in the chamber will result in compression of the channel (tube). By compressing the chambers after each other, peristaltic pumping can be achieved. A visual representation of this design is given in Figure 2.

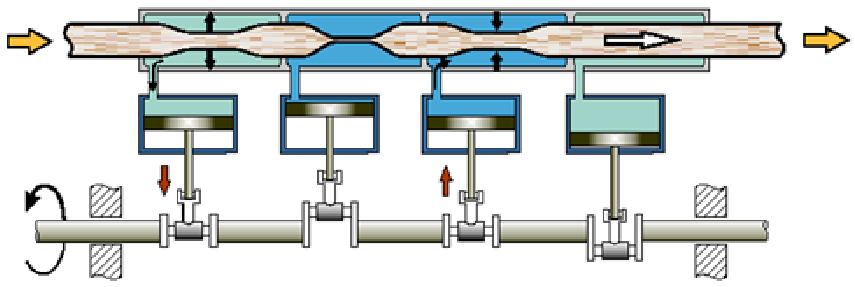


Figure 2: A design of a linear peristaltic pump with pneumatic or hydraulic actuation [13].

The design in patent [14] differs from the previous designs. This design does not use a camshaft, but gears instead. A shoe is connected via a link to a line of gears. One of the gears is driven by an electric motor causing them all to rotate. The mounting position of the link on the gear determines the phase difference between the shoes. In Figure 3, a schematic representation of the design is given.

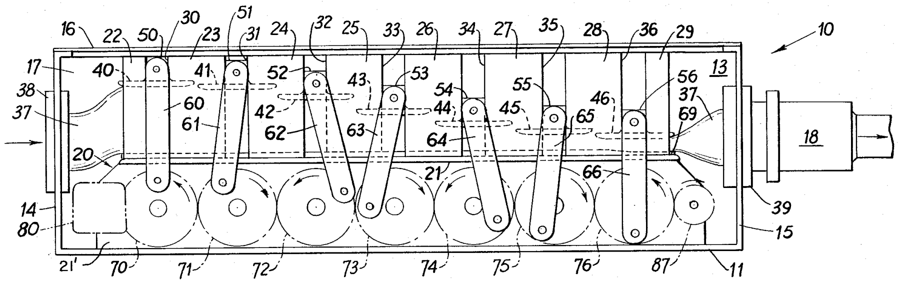


Figure 3: A linear peristaltic pump design where the shoes (40-56) are used to compress the tube (37). The shoes linked (61-66) and driven by gears (70-67) which are all driven by one gear connected to an electric motor (87) [14].

2.2 Linear peristaltic pumps with a variable phase difference between actuation points

A characteristic of linear peristaltic pumps with a variable phase difference between the actuation points is that they have multiple actuators which can be manipulated individually. By manipulating the actuators individually the phase difference between the actuators can be controlled. Since there is a great diversity of actuators, many different designs are possible.

The design in patent [15] uses pneumatic cylinders. The end of each cylinder is connected to a shoe which presses on the channel. The cylinder is either *in* or *out*, which makes the design suitable for binary control.

The patented design from NASA [16] is also suitable for binary control. Separate series of electrically conductive strips are overlying a microchannel, whose top surfaces are covered with electrically insulating material. The channel is covered by an electrically conductive flexible membrane. When no voltage is applied, the membrane is linear and lies over the channel. By applying a voltage between strips and the membrane, the membrane is electrostatically pulled into the microchannel. Alternating the applied voltage on the strips, a peristaltic motion of the membrane can be achieved. An overview of the micropump is given in Figure 4 (on the next page).

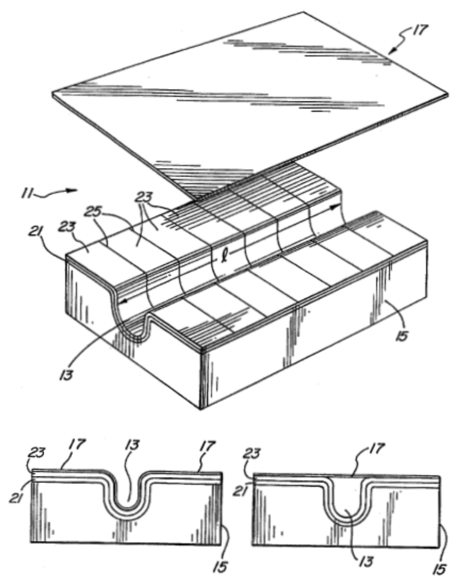


Figure 4: The linear peristaltic micropump from NASA. By applying a voltage on the conductive membrane (17) and the conductive strips (21), the membrane is pulled into the microchannel (13) pressing the substrate away. Between the two conductive layers there is a electrically insulting layer (23) [16].

A third design for binary control in literature can be found in the articles of Maddoui et al. [17] and Lee et al. [18]. Within both articles linear peristaltic pumps are actuated with electromagnetic actuators. In Figure 5 the design of Lee et al. [18] is shown. The design uses a parallel plate actuator. The channel is between the fixed and movable electrode. The spring represents the elasticity of the film. When applying a voltage to both electrodes in the right direction, the movable electrode film will be attracted to the fixed electrode. Therefore the fluid displaced by the plate will move away, but the direction of the fluid motion is unknown.

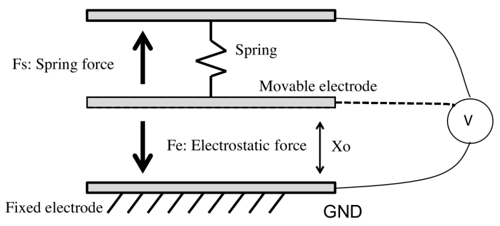


Figure 5: The structure of the parallel plate actuator [18].

2.3 Linear peristaltic pump controllers

The designs with variable phase differences are all using binary control strategies. The trajectory of the actuator to their final position is not controlled. A signal is send to the actuator such that it goes to their zero or end position. The direction of the medium by using binary control, can be controlled by eliminating all other flow directions. In case of the peristaltic pump it is done by holding the previous actuator down until the next one is also down. When there are more than 2 actuators, different actuating schemes are possible. In the paper of Lee et al. [18], four different actuating schemes are presented and evaluated for a gas as fluid. The flow-rate of the fluid is depending on the actuating scheme and actuating frequency. For certain cases the flow-rate can be easily approximated by using the volume displacement of the actuator.

The actuating scheme is a logical pattern of binary signals. Within the patent of NASA [16] a controller is proposed which consist out of an oscillator and shift registers. The output of the shift register is a clocked bit stream of 0's and 1's which apply no voltage or a voltage to the actuators. Within this patent the actuation is not discussed, but it is assumable that the frequency can be changed by changing the oscillator and the actuating scheme can be set in the shift registers.

Most designs mentioned in Chapter 2.2 are controlled binary by applying certain voltages to the electromagnetic actuators. When using an analog control signal the trajectory of the actuators can be controlled. By using an analog control signal a more natural way of peristaltic pumping can be achieved such as in the intestines. However no analog controller for peristaltic pumps are found in literature.

A possible method to control a peristaltic pump analog, is by creating multiple wave-like signals with phase difference. The wave-like signals can be used as reference signal or direct input for the actuators. A method to create wave-like patterns, where the nodes have a certain phase difference is by using two oscillators. The output signals of the oscillators are connected to the ground and to a potentiometer. The resulting signal U_3 is a summation of both signals. It is required that the oscillators must have the same frequency and be in anti-phase. The drawing of this electrical circuit is in Figure 6.

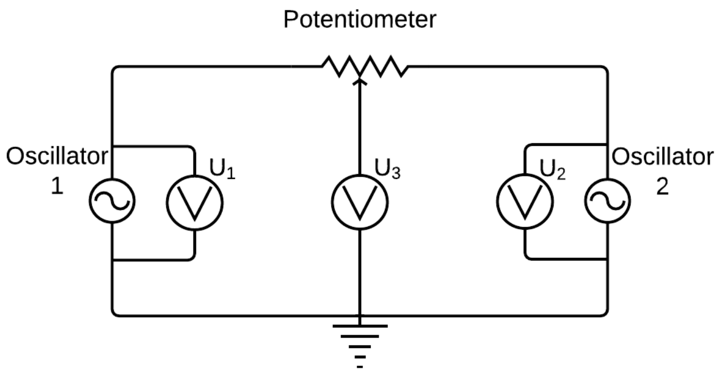


Figure 6: Electrical circuit with two oscillators to create a signal U_3 with a phase difference between 0 and π.

By changing the contribution of the oscillators by using the potentiometer, the phase of the output signal can be controlled from 0 to π. A side effect is that the amplitude of the signal will also change. However, in practice it is really hard to have two individual oscillators with exact the same frequency and required phase difference. When the frequency is slightly off, this will result in a wave form with extra peaks.

Therefore another possibility to create a phase difference between some signals is by applying a filter. The filter can be designed such that the phase-difference and amplitude meat the requirements for a specific frequency. However changing the frequency of the oscillator will cause the filters to create a signal with a different phase shift and amplitude.

Wave-like signals can also be generated by using Pulse Width Modulation (PWM) and applying a passive low-pass filter which filters the higher PWM frequencies out of the signal. However the reference signal for the PWM output must be generated somewhere.

A possible way to generate such a signal is by using a neural network. Such networks have multiple outputs and can generate different patterns, therefore it is also called a central pattern generator. The oscillation profile is related to the coupling matrix that specifies the network topology and the coupling strength. The frequency, amplitudes and phases are essentially encoded in terms of a pair of eigenvalue and eigenvector. This can be used to estimate the oscillation profile or to design a central pattern generator [19]. A topology which creates a wave-like motion, is given in the article of Pogromsky et al. [2]. The produced signals could be used as reference or actuation signal for a linear peristaltic pump.

3 Diffusive networks

The goal is to control a linear peristaltic pump with a neural network as central pattern generator. A diffusive cellular network will be used to develop a central pattern generator with the required wave-like pattern as output.

3.1 Notations of a diffusive cellular network

A diffusive cellular network is a network composed of identical dynamical systems coupled through diffusive coupling that cannot be decomposed into smaller disconnected networks. The k identical system are of the form

$$\dot{x}_j = f(x_j) + Bu_j, \quad y_j = Cx_j \tag{1}$$

where f is a smooth vector field, $j = 1, \ldots, k$, $x_j \in \mathbb{R}^n$ is the state, $u_j(t) \in \mathbb{R}^m$ is the input and $y_j(t) \in \mathbb{R}^m$ is the output of j-th system. Here B and C are constant matrices of appropriate dimension. The systems are coupled through mutual linear output coupling

$$u_j = -\gamma_{j_1}(y_j - y_1) - \gamma_{j2}(y_j - y_2) - \ldots - \gamma_{jk}(y_j - y_k) \tag{2}$$

with $\gamma_{ij} \geq 0$. Furthermore, it is assumed that matrix CB is a positive definite matrix. The coupling matrix is defined as

$$\Gamma = \begin{pmatrix} \sum_k^{i=2} \gamma_{1i} & -\gamma_{12} & \cdots & -\gamma_{1k} \\ -\gamma_{21} & \sum_k^{i=1, i\neq 2} \gamma_{2i} & \cdots & -\gamma_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{k1} & -\gamma_{k2} & \cdots & \sum_{k-1}^{i=1} \gamma_{1i} \end{pmatrix} \tag{3}$$

where the sum of all rows is zero. The network is called a diffusive cellular network when Γ is symmetric and only has one zero eigenvalue. The k identical systems (1) in the network with feedback (2) can be written as

$$\begin{cases} \dot{x} & = F(x) + (I_k \otimes B)u \\ y & = (I_k \otimes C)x \end{cases} \tag{4}$$

with the feedback

$$u = -(\Gamma \otimes I_m)y, \tag{5}$$

where $\otimes$ stands for the Kronecker product, $x = col(x_1, \ldots, x_k)$, $F(x) = col(f(x_1), \ldots, f(x_k) \in \mathbb{R}^{kn}$, $y = col(y_1, \ldots, y_k)$ and $u = col(u_1, \ldots, u_k) \in \mathbb{R}^{km}$.

3.2 Requirements on the diffusive network

To use the diffusive network as a central pattern generator two requirements are posed on the network. To create a infinite long pattern it is necessary that a compact (sub)set is ultimately bounded, but that the origin of this set is unstable such that it is oscillatory in the sense of Yakubovich.

3.2.1 Asymptotic Stability

The paper of Pogromsky et al. [2] shows that a network of diffusively coupled identical systems contains a globally asymptotically stable compact subset of the set $\ker(I_{kn} - \Pi \otimes I_n)$ when

i) The system (4, 5) is ultimately bounded

ii) There exists a positive definite matrix P and positive ϵ such that for all $x \in \mathbb{R}^n$ the following inequality

$$P\frac{\partial f(x)}{\partial x} + \left(\frac{\partial f(x)}{\partial x}\right)^\top P \leq -\epsilon I_n \tag{6}$$

holds.

iii) There is a $k \times k$ permutation Π which commutes with Γ: $\Pi\Gamma = \Gamma\Pi$.

Let λ' be the largest eigenvalue of Γ under the restriction that the corresponding eigenvector lies in the range of $(I_k - \Pi)$, then there exists a positive $\bar{\lambda}$ such that if $\lambda' < \bar{\lambda}$ the set $\ker(I_{kn} - \Pi \otimes I_n)$ contains a globally asymptotically stable compact subset.

The maximum value for γ, for which the system is stable can be computed by finding the smallest λ of the following problem

$$(I_k - \Pi)^\top \Gamma (I_k - \Pi) \leq \lambda (I_k - \Pi)^\top (I_k - \Pi). \tag{7}$$

3.2.2 Oscillatory behaviour

The second requirement posed on the network is that the network is oscillatory in the sense of Yakubovich. A system is oscillatory in the sense of Yakubovich if it is ultimately bounded and for almost all initial conditions there is no limit $\lim_{t\to\infty} y(t)$. According to the paper of Pogromsky et al. [2], the following conditions must hold for the network in (1, 2) with

$$f(x_j) = Ax_j - B\phi(z_j). \tag{8}$$

Furthermore all requirements of Chapter 3.2.1 must also hold.

i) The matrix A is Hurwitz, so there is a positive definite matrix $P = P^\top$, so that $A^\top P + PA < 0$.

ii) $z_j = Zx_j$, where $Z^\top = PB$ with the matrix P as in i.

iii) ϕ is an odd smooth strictly increasing function with the following property:

$$\forall C > 0 \quad \exists \sigma > 0 \quad \forall z > \sigma \quad \phi(z) > Cz. \tag{9}$$

iv) Let $Wy(s)$ be the transfer function of the linear part from u_j to $y_j = Cx$ taking $\phi(z_j) = 0$: $W_y(s) = C(sI - A)^{-1}B$. Then $W_y(s)$ is nondegenerate, it has relative degree one with even number of zeros with positive real part and $W_y(0) > 0$.

v) Let $W_z(s)$ be the transfer function $W_z(s) = (sI - A)^{-1}B$. Then $W_z(0) > 0$.

Then there is a number $\bar{\lambda} > 0$ so that if the largest eigenvalue of Γ exceeds $\bar{\lambda}$ then the network is oscillatory in the sense of Yakubovich. When all conditions hold it is sufficient prove the instability of the closed loop linear system

$$\dot{\xi} = (A - \lambda BC)\xi, \tag{10}$$

where $\xi := \Pi \otimes I_n x$. Here $\bar{\lambda}$ is the Hopf bifurcation point of (10), with λ the largest eigenvalue of Γ.

3.3 The required extensions of the requirements on diffusive networks

Besides the symmetric coupling matrices it is possible to use an asymmetric coupling matrix. By using an asymmetric coupling matrix, the coupling is applied in one direction instead of in two directions. To use a network as central pattern generator it is required that the system is oscillatory in the sense of Yakubovich. The prove of this oscillatory behavior can be divided in three smaller problems. The two properties need to be proven to call the system oscillatory in the sense of Yakubovich. With an additional third property, the uniqueness of the equilibrium point, the analysis can be simplified.

Ultimate Boundedness

To show that the system solutions do not blow up, the system needs to be ultimately bounded. This means that the system will converge to a certain subset as long as it is in a certain set. Ultimate Boundedness can be proven by using a Lyapunov function

$$V(x) = x^\top P x \tag{11}$$

where there is a number $\mathcal{C}$ such that $V(x) > \mathcal{C}$ implies that $\dot{V}(x) < 0$.

Uniqueness of the equilibrium point

Assume that the set for which the system is ultimately bounded contains (multiple) equilibrium points. Suppose that all equilibria are unstable, than there is a possibility that the system is going to oscillate. Uniqueness of the equilibrium point will simplify further analysis.

Unstable equilibrium points

The last requirement which need to be proven is the instability of the equilibrium point(s) for a sufficiently large coupling strength. This instability will provide the last requirement to call the system oscillatory in the sense of Yakubovich.

4 Network topologies and their output characteristics

Within the previous chapter it is shown that a neural network can create an infinite long pattern when fulfilling the conditions in 3.2.1 and 3.2.2. The pattern generated by the neural network is depending on the subsystems and the diffusive coupling. Based on the theory in the previous chapter, different types of coupling are possible and will be studied.

Within the article of Pogromsky et al. [2] an example is given. The example network fulfills all conditions mentioned in the previous chapter and generates an infinite long pattern. The used network topology is a ring of four diffusively coupled nodes, where the output of this network is a wave-like pattern. The output signals of node 1 and 3 are in-phase; node 2 and 4 are in-phase; and node 1 and 2 are in anti-phase.

Changing the network topology or using a different number of nodes will affect the output signals. In this chapter the different topologies and their output characteristics are covered. For the analysis of the different topologies, the following system is used

$$\begin{aligned} \dot{x}_j &= Ax_j + B(u_j - z_j^3), \quad j = 1, \dots, k, \quad x(t) \in \mathbb{R}^3 \\ z_j &= Zx, \quad y_j = Cx, \quad u = -\Gamma y \end{aligned} \tag{12}$$

where

$$A = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ -4 & 2 & -3 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top, \quad C = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \quad Z = B^\top C$$

and P is the solution of the following Lyapunov equation

$$A^\top P + PA = -I_3.$$

This network is the same as the example in the article of Pogromsky et al. [2]. The given network satisfies all mentioned conditions in 3.2.1 and 3.2.2.

The Hopf bifurcation of the closed loop system (10) depends on the largest eigenvalue of Γ. The Hopf bifurcation for this system is

$$\lambda = \frac{-1 + \sqrt{13}}{2} \tag{13}$$

where λ is the larges eigenvalue of Γ. The system is oscillatory in the sense of Yakubovich when $\lambda > \frac{-1+\sqrt{13}}{2}$. For the sake of simplicity all connections have the same coupling strength γ. Hence the coupling matrix can be expressed as $\Gamma = \gamma\ \Gamma_0$, where Γ_0 contains all the information on the topology. The system is oscillatory in the sense of Yakubovich when

$$\gamma \geq \frac{a}{\lambda}, \quad a = \frac{-1 + \sqrt{13}}{2},$$

where λ is the largest eigenvalue of Γ_0. The coupling strength for all simulations is $\gamma = 1.2\frac{a}{\lambda}$ unless indicated otherwise. The script used for simulations of only one topology is in Appendix A, the script used for simulations with combined topologies is in Appendix B. The custom functions used in the simulation scripts are all in Appendix C.

4.1 Bidirectional ring network

The first network being analyzed is a bidirectional ring network with k nodes. A visual representation is given in Figure 7 (on the next page). Two different cases are distinguished, when k is even and when k is odd. When k is even, the even numbered nodes are all in-phase and the odd numbered nodes are also in-phase. The odd numbered nodes are in anti-phase with respect to the even numbered nodes.

The network is simulated with MATLAB for $k \in \{4,\ 6,\ 10,\ 40,\ 200\}$ and different initial conditions, where $x_j(0) = 0$, $y_j(0) \neq 0$ and $y_j \neq y_{j+1}$. The simulation results are visualized by using animations showing the output y. The higher the number of nodes, the longer it takes to synchronizes, but the system will go the same periodic solution for different initial conditions. However, the initial conditions affect the rate of convergence to the periodic solution. The links to the animations of the periodic solutions for different number of nodes are in Table 1. To give more insight in the periodic solution, the output trajectory of a bidirectional ring with 6 nodes is given in Figure 8a.

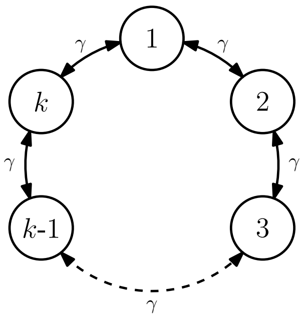


Figure 7: A bidirectional k node ring network.

For a certain set of initial conditions it takes very long for the network to converge to the periodic solution. Different patterns are generated before reaching the periodic solution. The links to the animations on different time intervals with different patterns are in Table 2.

Table 1: Links to the animations and information of the periodic solutions of a bidirectional ring network with even numbered nodes.

Nodes	Animation link
4	https://youtu.be/FOOd-cIlsTk
6	https://youtu.be/JSOYBa6xl90
10	https://youtu.be/3twnQwErbgQ
40	https://youtu.be/xrZCEY0PciY
200	https://youtu.be/n-74adNt7hs

Table 2: Links to the animations of the different patterns in the bidirectional network with 200 nodes, before reaching the periodic solution.

Time interval	Animation link
0 - 100	https://youtu.be/a4h3eWWvPxg
10000 - 10100	https://youtu.be/vM15DcQ4HQY
20000 - 20100	https://youtu.be/ur95se1P4m4
30000 - 30100	https://youtu.be/ZYUu5t2iQ1c
40000 - 40100	https://youtu.be/E8plar3-gdY
50000 - 50100	https://youtu.be/hDMZl0sALuY

The bidirectional ring network is also simulated for an odd number of nodes, $k \in \{3,\ 5,\ 9,\ 39,\ 199\}$. The simulation shows that the system will also converge to a periodic solution. The periodic solution for a systems with an odd number of nodes differs form the periodic solution of a network with an even number of nodes. Within a network with an even number of nodes, the nodes are either in-phase or anti-phase (phase difference of 0 or π). However, within the network with an odd number of nodes, the phase difference is equally shifted between certain the nodes. The phase difference between y_j, y_{j+2} and between y_{k-1}, y_1 is $\frac{2\pi}{k}$. The links to the animations of the simulation results can be found in Table 3 (on the next page). To give a better insight, the output trajectory of a bidirectional ring with 5 nodes is given in Figure 8b. The phase difference between the nodes is computed by

$$\varphi = \arccos\left(\frac{y_j \cdot y_{j+2}}{||y_j||||y_{j+2}||}\right). \tag{14}$$

Furthermore, the expected oscillation frequency on the bifurcation point of a bidirectional ring network is 0.130Hz and the oscillation frequency obtained from the simulation with the fast Fourier transform is 0.137Hz, for all networks. The output amplitude of the network differs slightly, depending on the number of nodes.

Table 3: Links to the animations of the periodic solutions of a bidirectional ring network with odd numbered nodes.

Nodes	**Animation link**
3	https://youtu.be/pyVYzhaHqoA
5	https://youtu.be/tfNdjuuEOlU
9	https://youtu.be/YC3iqh5yuTk
15	https://youtu.be/tXfsHMnJQ3w
39	https://youtu.be/EtX50axxzG4
199	https://youtu.be/pWemkUb6gIw

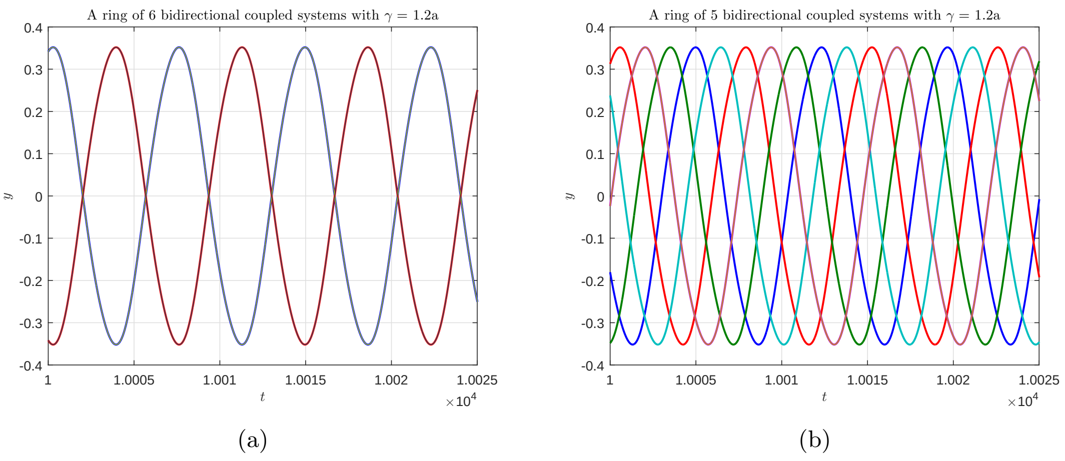


Figure 8: The output trajectory for a network with 6 nodes (a) and 5 nodes (b) connected in a bidirectional ring. When the number of nodes is even, the nodes are either in phase or in anti-phase. For an odd number of nodes, the phase is shifted equally between the nodes.

4.2 Bidirectional line network

The second network which is examined is a bidirectional line network. The hypothesis is that the line network will behave the same as an infinite long bidirectional ring network. Hence the network will converge to a periodic solution, where the nodes are close to the in-phase and anti-phase mode for all k.

Simulations of the network show that for small k the periodic solution of the bidirectional line network is almost the same as the periodic solution of the bidirectional ring. The difference with the bidirectional ring network is that the ouptut amplitudes of the first and last nodes are much smaller. The amplitude of the other nodes is close to the output amplitudes of the bidirectional ring network.

For higher numbers of k the network goes to the same periodic solution as the bidirectional network, but converges much slower. A plot of the output trajectory for $t \in [10000,\ 10020]$ with $k = 6$ is in Figure 9. Here it can be seen that the trajectory is close to the in-phase and anti-phase mode. When time increases, the trajectory converges to the in-phase and anti-phase periodic solution.

For very high numbers of k the network converges so slow that it cannot be confirmed by using simulations that the system reaches the in-phase and anti-phase mode. Due to the large number of data for the larger simulation times the computer runs into memory problems. However, it is very assumable since the observed patterns obtained after a certain time t are very similar to the patterns generated in the bidirectional network before reaching the periodic solution. The links to the animations of the simulation results are in Table 4.

The expected oscillation frequency on the bifurcation point of the bidirectional line is also 0.130Hz. The oscillation frequency obtained form the simulations is 0.137Hz. These frequencies are the same as in the bidirectional ring network.

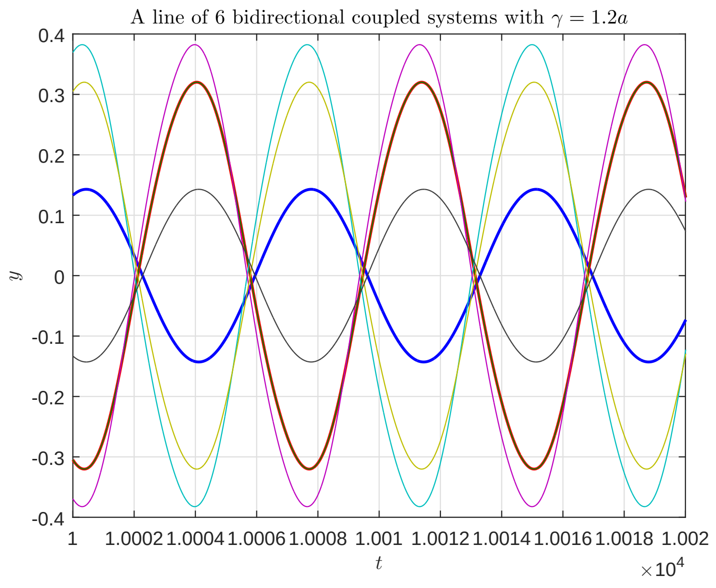


Figure 9: The output trajectory of the bidirectional line network for $t \in [10000,\ 10020]$ and $k = 6$.

Table 4: Links to the animations of the periodic solutions of a bidirectional line network.

Nodes	**Animation link**
3	https://youtu.be/TCAmTqY9Ch8
4	https://youtu.be/cUjHOf2nkoA
5	https://youtu.be/XkcOWgZ8Ufk
6	https://youtu.be/-kN_1k3vJx0
9	https://youtu.be/XMJS92r2GRc
10	https://youtu.be/YEH-wIf0cf0
15	https://youtu.be/JJufssCbRbI
39	https://youtu.be/UEh4hUKUTHo
40	https://youtu.be/fH9-nYawP7M
199	https://youtu.be/fF71aUk1g2E
200	https://youtu.be/z8YLGuHMjWA

4.3 Unidirectional ring network

Within the bidirectional ring and line network a waveform is generated, but the direction of the wave is depending on the initial conditions. A possible way to influence this behavior is by changing the coupling within the network. The coupling between nodes can be changed into a one sided connection (see Figure 10). The direction in which the nodes in Figure 10 are coupled is called right.

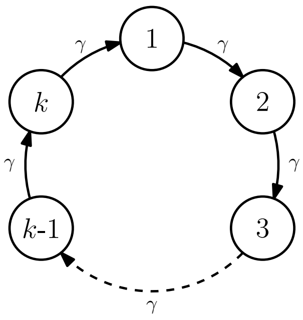
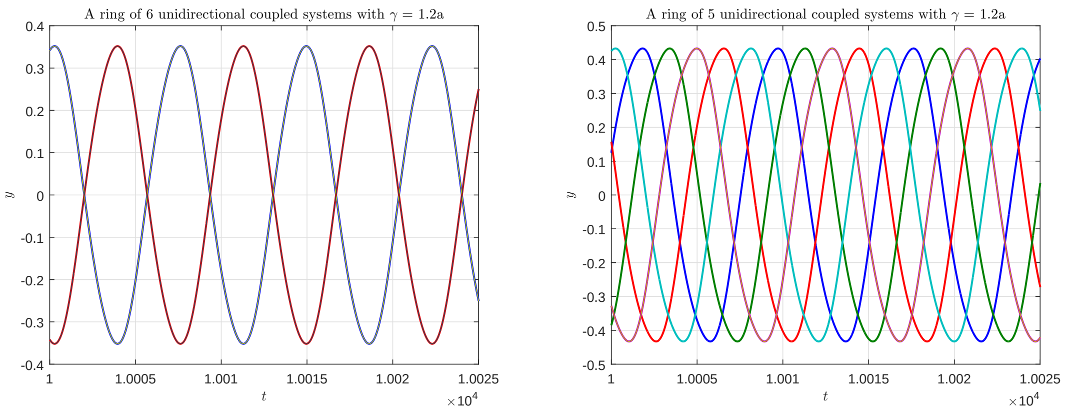


Figure 10: An unidirectional k node ring network, where the coupling direction of the nodes is called right.

By applying this kind of coupling, the network is not a cellular network anymore. The coupling matrix Γ_0 is not symmetric and also has complex eigenvalues. Therefore the theorems proposed by Pogromsky et al. [2] do not hold, hence the system cannot be guaranteed to be stable or oscillatory. However it can be proven that the origin is unstable. Due to the complex eigenvalues of Γ_0 the simulations are performed close to the bifurcation point with $\gamma = \frac{1.2}{\mathrm{Re}(\lambda)}a$.

The simulations of the unidirectional ring network gives the expected results. A unidirectional ring network with an even number of nodes, has the same periodic solutions as the bidirectional ring network (see Figure 11). When changing the unidirectional network to an odd number of nodes, the phase difference is divided equally and the direction of the wave is the desired direction. The direction of the wave can be changed by changing the direction of the coupling (taking the transposed of the coupling matrix Γ_0).



(a)

(b)

Figure 11: The output trajectory for a network with 6 nodes (a) and 5 nodes (b) connected in an unidirectional ring. When the number of nodes is even, the nodes are either in phase or in anti-phase. For an odd number of nodes, the phase is shifted equally between the nodes. The results are the same as in a bidirectional ring network.

The simulation results are in Table 5. The simulation for $k = 199$ did not reach its periodic solution within $t = 10000$. Hence, the system will converge slower to the periodic solution than the bidirectional ring network. However after $t = 50000$ the system has reaches its periodic solution and the phase difference between certain nodes is $\frac{2\pi}{k}$.

The expected oscillation frequency of the unidirectional ring is 0.131Hz and the oscillation frequency within the simulation is 0.137Hz for even number of k, which is the same as within the bidirectional ring. For odd numbers of k the oscillation frequency differs. For $k = 3$, $f_{\mathrm{sim}} = 0.122$Hz,

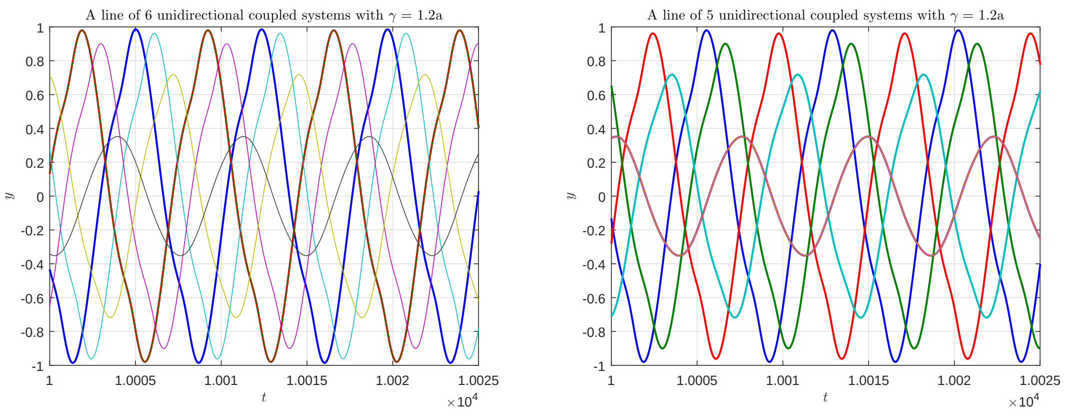
$k \in \{5,\ 9,\ 15\}$, $f_{\text{sim}} = 0.130$Hz and $k \in \{39,\ 199\}$ gives $f_{\text{sim}} = 0.137$Hz. For an odd number of nodes k, the output amplitude changes slightly. This is not the case for even numbers of nodes k.

Table 5: Links to the animations of the periodic solutions of an unidirectional line network.

Nodes	**Animation link**
3	https://youtu.be/EnEly9GvRGU
4	https://youtu.be/lYjUfX7hQNE
5	https://youtu.be/gQjUVRveZ4c
6	https://youtu.be/gdCEHRUDZ3Q
9	https://youtu.be/19H341hU11I
10	https://youtu.be/kLAq3hWV5Ck
15	https://youtu.be/Es_AzCl3SLY
39	https://youtu.be/vXtBTStNl2I
40	https://youtu.be/8IOf7lsvAz8
199	https://youtu.be/G6Cg6ICvGyI
200	https://youtu.be/WCvh0lfKUpQ

4.4 Unidirectional line network

The unidirectional line network also has an asymmetric coupling matrix with complex eigenvalues. Simulations on this network were also performed and gave a surprising outcome. The phase difference between all the odd nodes is $\approx \frac{2\pi}{6.7}$, the phase difference between the even nodes is also $\approx \frac{2\pi}{6.7}$ and the phase difference between the nodes $k-1$ and 1 nodes varies. Furthermore, the amplitude of the output signal is twice as big as in the other networks.



(a) (b)

Figure 12: The output trajectory for a network with 6 nodes (a) and 5 nodes (b) connected in an unidirectional line. There is almost no difference between an even or odd number of nodes.

The expected oscillation frequency is 0.133Hz. The oscillation frequency obtained from the simulations is 0.137Hz, which is the same as in the original bidirectional ring network. The links to the animations of the simulations are in Table 6, the output trajectory is shown in Figure 12.

Table 6: Links to the animations of the periodic solutions of an unidirectional line network.

Nodes	Animation link
3	https://youtu.be/9bLbzpF5BCw
4	https://youtu.be/EDrTis_DecQ
5	https://youtu.be/-0K9KHbHjHA
6	https://youtu.be/cpRSE82oLlw
9	https://youtu.be/liX6Ktm-8U0
10	https://youtu.be/PPFRMRF0Rqo
15	https://youtu.be/934GzzPRb88
39	https://youtu.be/iaomMRtIuZ4
40	https://youtu.be/knNXxgWb660
199	https://youtu.be/kRdb2x0FTEc
200	https://youtu.be/t2brf9TLOTc

4.5 Combining different network topologies

The above networks can be combined to create bigger and more complex networks. For example, multiple unidirectional ring networks can be coupled in a bidirectional line, see Figure 13.

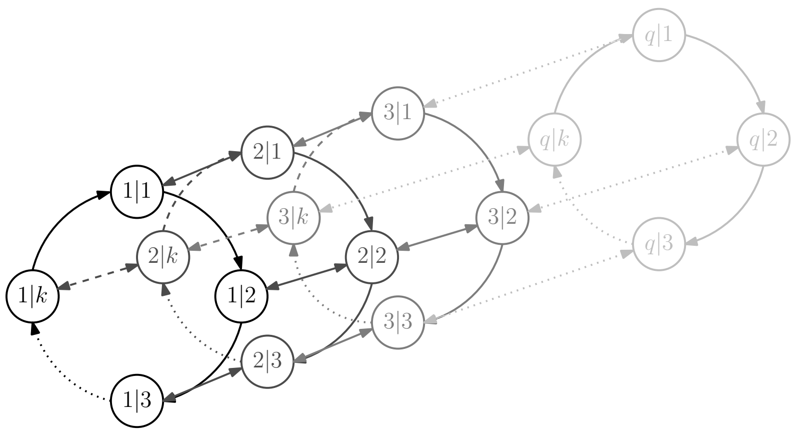


Figure 13: An unidirectional ring network with k-nodes substituted in a bidirectional line network with q repetitions.

When coupling the networks, the hypothesis is that the combined network takes the characteristic of the original network in the direction it is applied. For example an unidirectional ring network with an odd number of nodes k should create a wave-like pattern in the desired direction. The phase difference between the nodes $i|j$ and $i|j+2$ in the ring should be shifted equally as in the regular unidirectional ring network with an odd number of nodes. Multiple unidirectional rings can be connected to each other as a bidirectional line. The direction where the bidirectional line is created is the direction where the nodes $i|j$ and $i+1|j$ should be in anti-phase and the nodes $i|j$ and $i+2|j$ in-phase.

Simulations of four unidirectional ring networks with $k \in \{5,\ 9,\ 15,\ 19\}$ which are coupled in a bidirectional ring are executed. The links tot the animations of the simulation results can be found in Table 7. A representation for six unidirectional ring networks with 5 nodes is in figure 14. Comparing the systems on the vertical axis shows that they are in-phase. While comparing the systems on the horizontal axis it can be seen that they are in anti-phase.
Here it can be seen that the network output is as expected, which can be seen most clearly in

Table 7: Links to the animations of a combined network topology with four unidirectional rings with k nodes connected as a bidirectional ring.

Nodes in Unidirectional Ring	Animation link
5	https://youtu.be/X-8KTaSv7cM
9	https://youtu.be/EDwjBJ8ceKg
15	https://youtu.be/JOn3fBPURjs
19	https://youtu.be/xU_3Lukod9I

the animation with 19 nodes. Note that the amplitude of the output is four times larger than the output of the original unidirectional ring.

The same is done for four unidirectional ring with 19 nodes in a bidirectional line. The animation of the simulation can be found at https://youtu.be/BS-_jt_aYrA. The animation shows that the network does not provide the expected behavior. However, a part of the wave-like pattern can be distinguished within the animation. It is very assumable that the network has not reached its periodic solution yet. Unfortunately the computer runs into memory problems when simulating for a larger time interval.

Systems with an odd number of repetitions are also simulated, these simulations takes a lot more computer memory. Therefore smaller numbers of k must be used for these simulations. The animation of the simulation for a system with three repetitions of a unidirectional ring with 5 nodes, coupled as bidirectional ring is at https://youtu.be/yqCW_4krkTw. The animation for 5 repetitions is at https://youtu.be/yqCW_4krkTw. Here it can be seen that the phase difference between $i|j$ and $i|j+2$ is $\frac{2\pi}{5}$ and the phase difference between $i|j$ and $i+2|j$ is also $\frac{2\pi}{5}$ when $k = 5$.

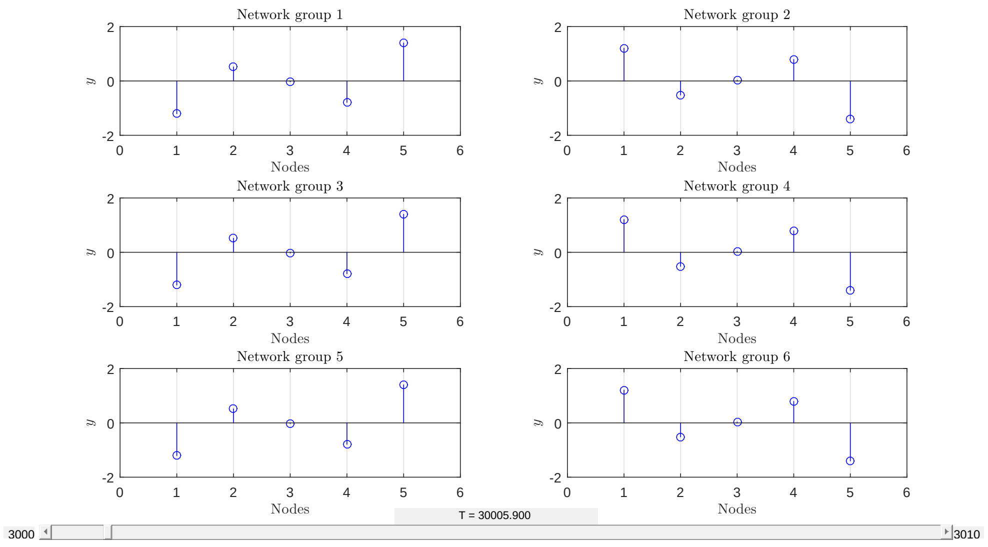


Figure 14: Six unidirectional ring networks with 5 nodes coupled in a bidirectional ring. Vertically the systems are in phase and looking horizontal shows that the systems are in anti-phase.

5 Dynamics of the nodes and their influences on the output

In the previous chapter the effect of the network topologies on the output is discussed. Besides the topology of the network, the dynamics of the nodes can also influence the output. In Chapter 3.2 the requirements on the dynamics are listed. The conditions are easily fulfilled when the dynamics of a node has a certain structure. The used structure to describe the dynamics of the nodes is in (12) with

$$A = \begin{pmatrix} a & -b & 1 \\ 1 & 0 & 0 \\ -e & f & -g \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \quad Z = B^{\top} P$$

where P is a positive definite matrix and the solution to $A^{\top} P + PA = -I_3$. The corresponding transfer function from u_j to y_j is

$$W_y(s) = \frac{s^2 - as + b}{s^3 + (e-a)s^2 + (b + c - ae)s + be - d}. \tag{15}$$

With use of this equation, the poles and zeros can be placed anywhere to investigate the influence on the output signal. The focus of the analysis will be on the amplitude and the shape of the output signal with respect to the pole and zero placement. The oscillation frequency is not interesting because the simulations will be performed close to the bifurcation point. Therefore the oscillation frequency of the network is close to the largest imaginary value of the eigenvalues of the closed loop system as described in (10). The oscillation frequency can be scaled by multiplying the A and B matrix with a certain factor α, since

$$\begin{aligned} \det\left(\lambda I - \alpha(A - \lambda' BC)\right) &= \\ \det\left(\frac{\lambda}{\alpha} I - (A - \lambda' BC)\right) &= \\ \det\left(\bar{\lambda} - (A - \lambda' BC)\right) &= 0 \end{aligned} \tag{16}$$

where λ' is the largest eigenvalue of Γ and $\bar{\lambda}$ is the new scaled eigenvalue. Hence the oscillation frequency is scaled with a factor α and can be arbitrary placed without changing the poles or zeros.

For the analysis the poles will be placed in the open left half plane, therefore the A matrix is Hurwitz and the required conditions are fulfilled. By placing the zeros in the open right half plane, all conditions of Chapter 3.2 are satisfied, including the conditions on $W_z(s)$. Here $W_z(s) = Z(sI - A)^{-1}B$ is the linear part from u_j to z_j.

When placing the poles and zeros, four different cases can be distinguished. The first case is real valued poles and real valued zeros; the second case is real valued poles and complex valued zeros; third case is complex valued poles and real valued zeros; finally the fourth case is complex valued poles and complex valued zeros. For all cases the influences on the output is studied. To have a good comparison the coupling strength γ is chosen to be $\gamma = 1.2\frac{a}{\lambda}$, where λ is the largest eigenvalue of Γ_0 and a is the Hopf bifurcation point for the closed loop system as described in (10). The network used for the analyses is a bidirectional ring network with 5 nodes.

5.1 Real valued poles and real valued zeros

The first case is where both, the poles and the zeros, only have a real part. First all poles are placed on -1 and the zeros will be changed. The placement of the zeros and the corresponding amplitude of the waveform are in Table 8 as well as the references to the figure numbers. The reference signal is the signal where both zeros are located at 1 (see Figure 15). The other waveforms are shown in Figure 16.

Table 8: The effects of different real valued zeros on the amplitude and their figure number.

Poles	**Zeros**	$\|y\|$	**Figure**
$-1,\ -1,\ -1$	1, 1	0.7186	15
$-1,\ -1,\ -1$	2, 2	0.1016	16a
$-1,\ -1,\ -1$	3, 3	0.02035	16b
$-1,\ -1,\ -1$	$^1/_2,\ ^1/_2$	0.7848	16c
$-1,\ -1,\ -1$	$^1/_4,\ ^1/_4$	0.3837	16d
$-1,\ -1,\ -1$	1, 3	0.1411	16e
$-1,\ -1,\ -1$	$1,\ ^1/_4$	0.1411	16f

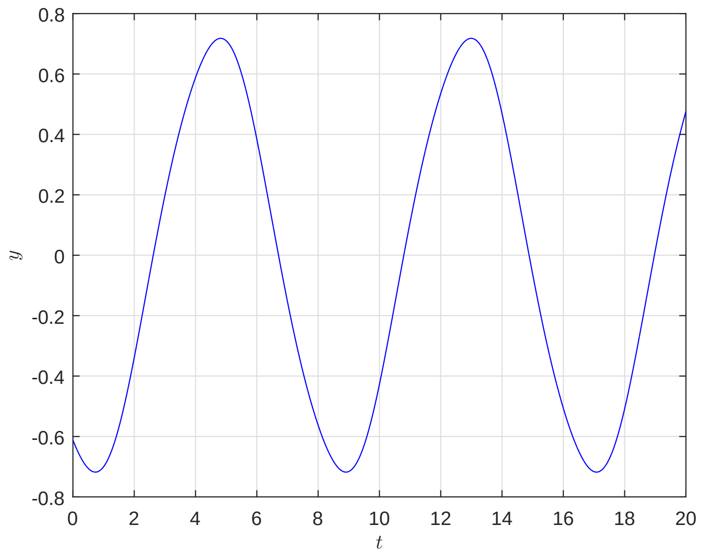


Figure 15: Reference waveform with all poles of $W_y(s)$ on -1 and the zeros on 1.

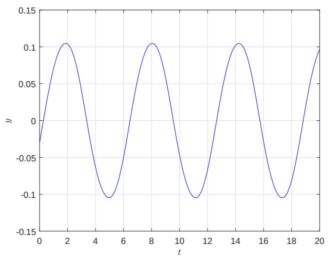
It can be seen that the placement of the poles influences the amplitude and the sharpness of the waveform. Larger valued zeros imply smoother waveforms and smaller amplitudes.

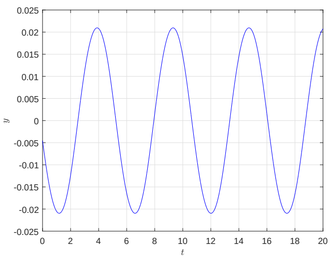
Now the effect of the poles can be studied. In Table 9 the placement of the poles and their influence on the amplitude can be found, as well as the references to the figure numbers. In Figure 17 the other waveforms are shown.

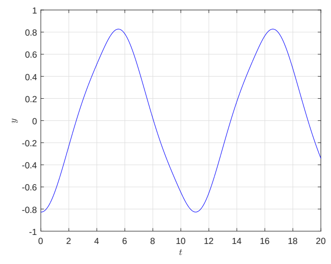
Table 9: The effects of different real valued poles on the amplitude and their figure number.

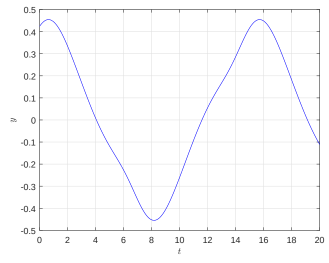
Poles	**Zeros**	$\|y\|$	**Figure**
$-1,\ -1,\ -1$	1, 1	0.7186	15
$-2,\ -2,\ -2$	1, 1	22.11	17a
$-3,\ -3,\ -3$	1, 1	81.96	17b
$-^1/_2,\ -^1/_2,\ -^1/_2$	1, 1	0.009711	17c
$-^1/_4,\ -^1/_4,\ -^1/_4$	1, 1	0.0001234	17d
$-1,\ -1,\ -3$	1, 1	6.086	17e
$-1,\ -1,\ -^1/_4$	1, 1	0.04832	17f

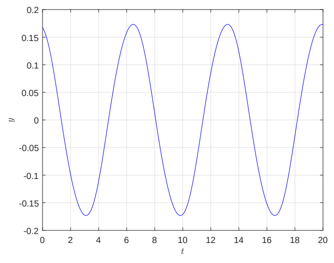
Larger values for the poles create larger amplitudes, furthermore the waveform is smoother with


(a)


(b)


(c)


(d)


(e)

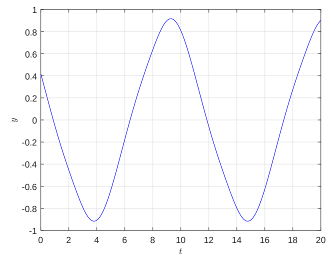
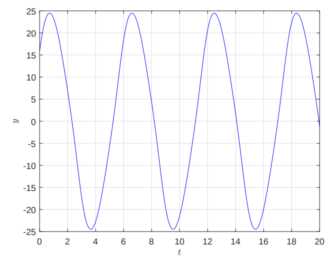
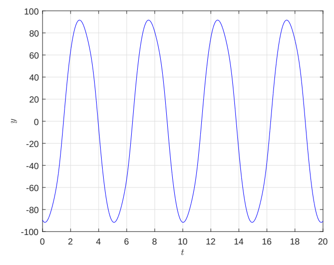
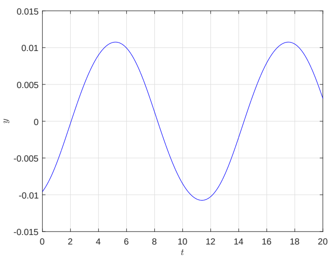
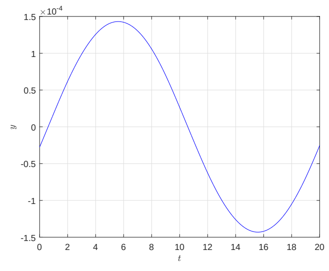

(f)

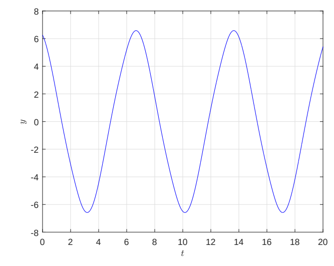
Figure 16: The effect of the zero placement on the waveform.


(a)


(b)


(c)


(d)


(e)

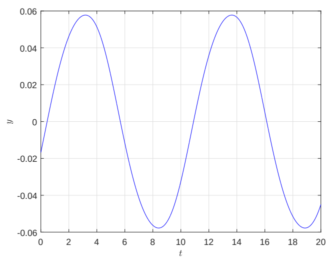

(f)

Figure 17: The effect of the pole placement on the waveform.

respect to the ones where the zeros were changed. This can be clearly seen when comparing the waveforms in Figure 16d and Figure 17d. Hence the real values of poles and zeros influences the amplitude of the wave-form and the sharpness of the shape.

5.2 Real valued poles and complex valued zeros

Since it is known what kind of effect the real values have on the output of the network, the next step is to investigate the effect of complex zeros on the output. The poles and real value of the zeros is constant while the complex value is changed. The results are in Table 10 and in Figure 18.

Table 10: The effects of different complex valued zeros on the amplitude and their figure number.

Poles	**Zeros**	$\|y\|$	**Figure**
$-1, -1, -1$	$1, 1$	0.7186	18a
$-1, -1, -1$	$1 \pm 1j$	0.3273	18b
$-1, -1, -1$	$1 \pm 2j$	0.0536	18c
$-1, -1, -1$	$1 \pm 3j$	0.009528	18d
$-1, -1, -1$	$1 \pm \frac{1}{2}j$	0.5911	18e
$-1, -1, -1$	$1 \pm \frac{1}{4}j$	0.62456	18f

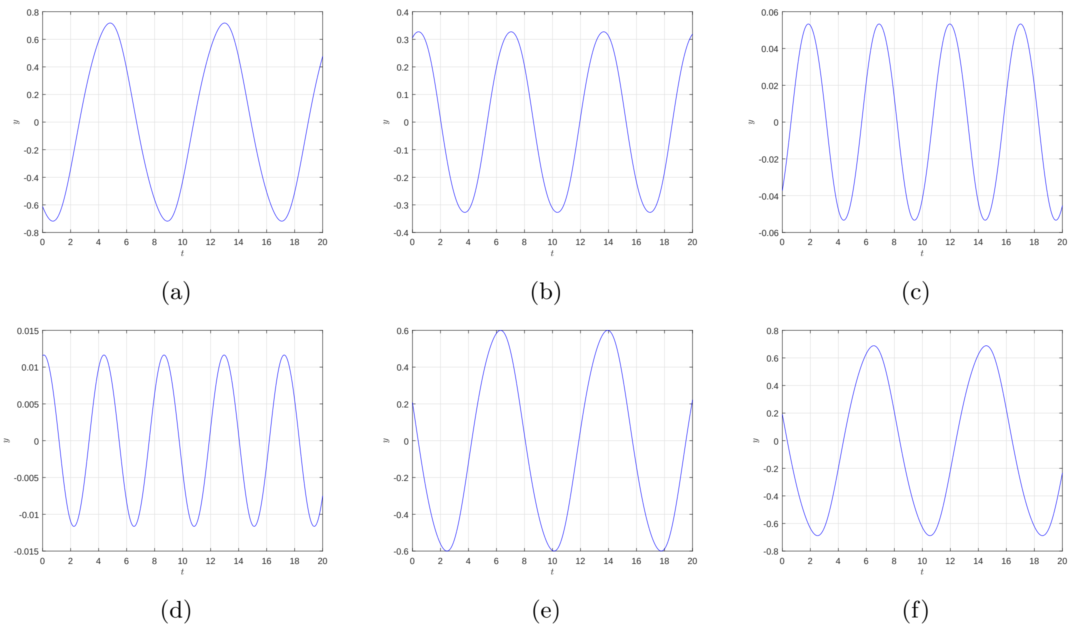


Figure 18: The effect of complex valued zeros on the waveform.

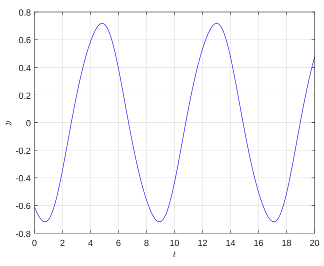
Increasing the complex part of the poles gives a smaller amplitude, but does not have a great effect on the shape of the output.

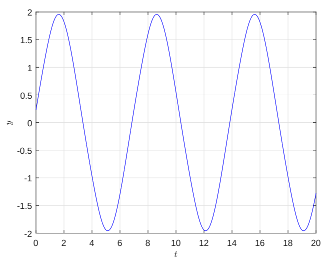
5.3 Complex valued poles and real valued zeros

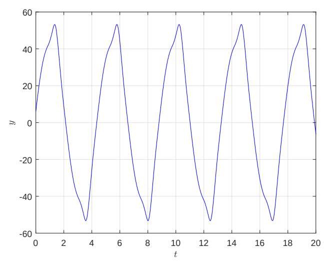
The only thing left to study is the effects of the complex poles. The results are in Table 11 and in Figure 19.

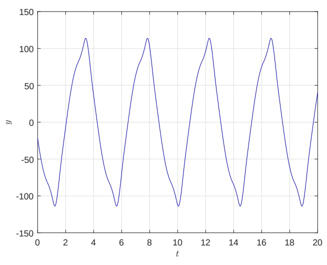
Table 11: The effects of different complex valued zeros on the amplitude and their figure number.

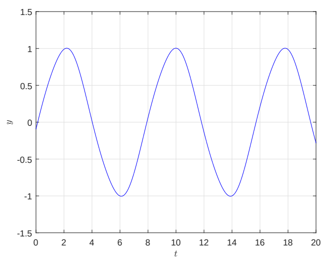
Poles	**Zeros**	$\|y\|$	**Figure**
$-1, -1, -1$	$1, 1$	0.7186	19a
$-1, -1 \pm j$	$1, 1$	1.783	19b
$-1, -1 \pm 10j$	$1, 1$	38.92	19c
$-1, -1 \pm 20j$	$1, 1$	94.08	19d
$-1, -1 \pm \frac{1}{2}j$	$1, 1$	0.9529	19e
$-1, -1 \pm \frac{1}{4}j$	$1, 1$	0.7377	19f


(a)


(b)


(c)


(d)


(e)

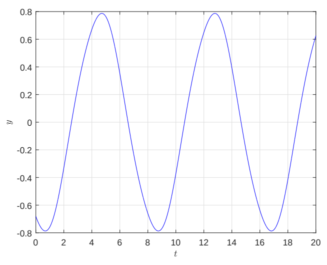

(f)

Figure 19: The effect of complex valued zeros on the waveform.

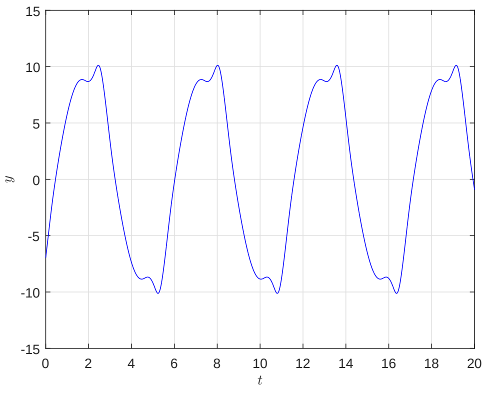
It can be seen that for high imaginary values the top of the waveform shows an additional peak. From the Fourier transform it is clearly visible that for higher imaginary values there are more sinusoidal frequencies in the signal than before. For smaller imaginary values, no difference can be observed within the waveform or the Fourier transform.

5.4 Complex valued poles and complex valued zeros

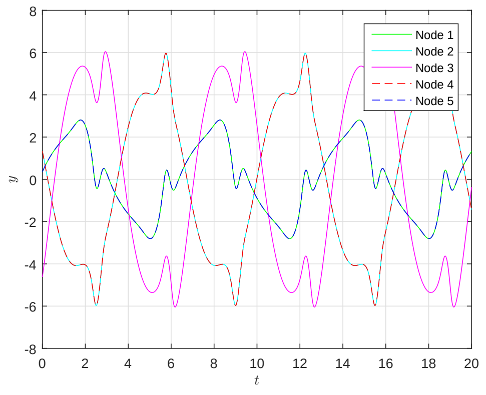
Simulations with complex poles and complex zeros did at first sight not show any new or unexpected behavior. Earlier it was observed that complex zeros do not have an observable influence on the waveform. When combining complex poles and complex valued zeros, the waveform is mostly the same as with only complex poles. However a major difference is observed when the real part of the poles and zeros is small and the imaginary part of the poles is large. An example of such a waveform is given in Figure 20a where the poles are placed on $-\frac{1}{2}$ and $-\frac{1}{2} \pm 5j$ and the zeros on $\frac{1}{2} \pm j$.

When creating an even bigger difference between the real and imaginary values of the poles (poles on $-\frac{1}{4}$, $-\frac{1}{4} \pm 10j$ and zeros on $\frac{1}{4} \pm j$), the system does not fully synchronizes (see Figure 20b). Three different waveforms are observed and two times there are two nodes which have the same waveform and are in phase.

When both zeros and poles are complex, the real part of the poles and zeros is small and the complex part of the poles is large, strange waveforms can appear and the desired synchronization can be lost.



(a)



(b)

Figure 20: Waveforms created where poles have a small real part and a large imaginary part.

6 Proposed pump design of a linear peristaltic pump

To apply a central pattern generator on a linear peristaltic pump as a controller for the peristaltic motion, a design is required. Within Chapter 2 the current state of linear peristaltic pumps is covered. Most of the current designs have a fixed phase difference between the actuation points and only one control input for changing the frequency of the peristaltic motion. However a central pattern generator has multiple output signals and can generate different waveforms and phase difference between the nodes. Therefore the linear peristaltic pump needs to have multiple individual actuation points on the channel.

Within Chapter 2 also some pumps with variable phase difference between the actuation points are covered. Those pumps have multiple actuators which can be controlled individually, but they are either *in* or *out*. A square waveform is introduced in to the system and the flow-rate can be approximated by using the volume displacement of an actuator. The advantage of the central pattern generator is that it creates smoother waveforms, which can mimic a more natural peristaltic motion.

It is not yet known which kind of waveform creates the most optimal peristaltic motion. Therefore two different designs will be proposed. The first design is a prototype where different waveforms can be tested and verified with theory. The second design will be more focused on industrial applications, where the dynamics for the waveform is known.

6.1 The central pattern generator

The linear peristaltic pump is controlled by a central pattern generator and generates either the control or reference signal. The required dynamics of the central pattern generator are not known (yet). The desired waveform and therefore dynamics can depend on a different parameters, such as the fluid and the channel dimensions. To study the desired waveform, the prototype of the linear peristaltic pump needs to be versatile. The dynamics of the network can be realized on two manners, namely by hardware or via software.

The dynamics of a node can be created by using an electric circuit with feedback loops. It is assumable that there is a possibility to change the dynamics slightly, however lager changes require a new circuit.

Another possibility to create the dynamics of the network is by using a real-time simulator. The real-time simulation will compute the output of the dynamical system at the same time as it would take the real physical system to obtain the value. The computed output signal can be created by using hardware. This is called hardware-in-the-loop (HIL) real-time simulation. According to [20] there is a great variety of different real-time simulators, but only a few are capable of simulating large systems. The other real-time simulators are suitable for small systems or can be used as a real-time controller. The focus of the article is on real-time simulations of power systems, which contains discrete time events. Therefore it is not fully representable for simulating a central pattern generator, however it gives insight in the existence of real-time simulations and their possibilities. Hence there are real-time simulators available with hardware connectivity, but there are only a few real-time simulators available to simulate larger systems.

The real-time simulator can simulate every possible network as long as the computing power of the simulator is sufficient. However real-time simulators are more expensive than an electrical circuit on a printed circuit board (PCB). Larger networks also require more computational power, therefore the price of a real-time simulator will increase. When using a PCB it only cost an additional PCB, which is assumed to be cheaper.

Therefore the most suitable option for the prototype is a real-time simulator with hardware in the loop to test different dynamics. For the more industrial linear peristaltic pump an individually designed PCB creating the desired dynamics is advised, the driver for the actuator could also be directly implemented in the PCB.

6.2 The actuators

The purpose of the central pattern generator is to create an actuator input or reference signal such that peristaltic pumping is achieved. The peristaltic motion can be created by using individually controlled actuators to deform the channel wall. Within different studies [21, 22, 23] on peristaltic motion, the dynamics of the fluid are expressed as an function of the local pressure or channel height. Assumed is that the required pressure or deformation profile is known and the central pattern generator creates the corresponding signal. The actuators need to create the given profile by applying a force on the channel wall.

A Lorentz actuator has a linear relation between their current and force, therefore they could be suitable for the job. They are also available in many different sizes which is a great advantage, since the dimensions of the pump are not known yet. The Lorentz actuator can be driven by using Pulse Width Modulation (PWM) since their inner circuit, a coil and a resistor (RL-circuit), acts as a low-pass filter

$$\frac{I(s)}{V(s)} = \frac{1}{Ls+R}. \tag{17}$$

With L the inductance in Henry, R the resistance in Ohm, $I(s)$ the current in Ampere and $V(s)$ the electric potential in Volt. The low-pass filter, filters the higher harmonics ($\omega >> {}^{R}/_{L}$) out of the signal and only the average value remains. Furthermore the position of the Lorentz actuator can be measured to create a closed loop feedback controller, but due to the linear dynamics it is often controlled open-loop.

The mechanical part of the differential equation describing a Lorentz actuator with mass m, viscous damping b and stiffness k, driven by a magnetic force $F(t) = B\ell i(t)$ which depends on the current $i(t)$. Is given by

$$m\ddot{x} + b\dot{x} + kx = B\ell i(t), \tag{18}$$

where B is the strength of the magnetic field and ℓ the effective wire length through the magnetic field. The electrical part including back EMF can be described by

$$L\frac{di}{dt}(t) + Ri(t) = V - B\ell\dot{x} \tag{19}$$

with inductance L and resistance R. Combing those equations and taking the Laplace transform gives

$$\frac{X(s)}{V(s)} = \frac{B\ell}{mLs^3 + (bL + mR)s^2 + (kL + bR + (B\ell)^2)s + kR} \tag{20}$$

The Lorentz actuator itself is thus a third order differential equation and could be used (partially) to created the desired node dynamics. However the transfer function has three poles and no zeros, the zeros could be added by using a feedback loop.

6.3 The channel

The connection between the actuators and the channel is of high importance, since it determines the pressure or deformation profile of the channel. In Figure 21 two different connection types are shown. The first connection (see Figure 21a) is as in [17, 18], where a large part of the channel is compressed by the actuators. These system are mostly controlled by turning the actuators *on* or *off*, therefore the flow rate can be approximated by using the volume displacement and the switching frequency. The second connection type (see Figure 21b) has the advantage that the displacement of the wall behaves more like a smooth function. With this setup the shape of the channel can be controlled with higher precision. Within literature it is also more common to study the peristaltic motions by using a smooth curved profile. Therefore it is recommended to use the connection type from Figure 21b, such that the performance can be predicted and confirmed with theory.

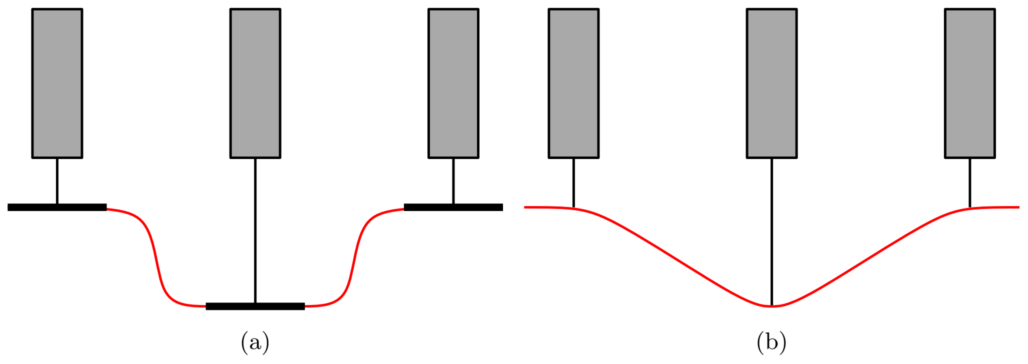


Figure 21: Two different connection types from the actuator (gray) to the channel wall (red).

The analysis of peristaltic flow is done in many different ways with respect to the fluid, channel deformation and channel lay-out. Most studies start with the analysis of a fluid in a two dimension plane, with long wavelengths and low Reynolds numbers [22, 23]. Analysis done in the two dimensional plane means that the channel has a characteristic length and width, but an infinite depth [21]. In this way the properties of the system will not depend on the third dimension. Furthermore, most studies assume symmetric deformation of the channel to simplify the problem and corresponding equations. Another common lay-out is an axi-symmetric channel, but in practice it is hard to control the deformation in all directions and therefore not recommended.

Another boundary conditions which is often used, is the boundary condition that the channel is sufficiently elastic such that there is no movement of the channel in the longitudinal direction.

The proposed channel has a characteristic length and height, the depth of the channel must be very large with respect to the length and height such that the effect of this dimension can be neglected. Furthermore the channel is actuated symmetrical and is composed out of elastic material such that movements in the longitudinal direction can be neglected.

6.4 Architecture Design

All components of the linear peristaltic pump are covered individually, combing the different components gives the proposed architecture designs.

The proposed design for the prototype is in Figure 22. The network is simulated with a real-time simulator with hardware in the loop. By using this solution the dynamics of the network can be easily changed. However there are restrictions on the dynamics due to the limitation of the real-time simulator. The simulation performed on networks with low oscillation frequencies are a lot faster with respect to higher oscillation frequencies. The size of the network also plays a significant role, there will be a trade of between the oscillation frequency and network size. However the most analysis of peristaltic motion use low frequencies, hence larger networks can be used. The required computation time of all networks used in this report is smaller than the time for which it is simulated. Only with larger networks or higher frequency the computer runs into memory problems. The computer used for these simulations runs on Windows 8.1, has 8GB of RAM and uses an Intel i7-4700MQ processor running at 2.40GHz. The simulations are performed in MATLAB R2015a using the ode45 solver.

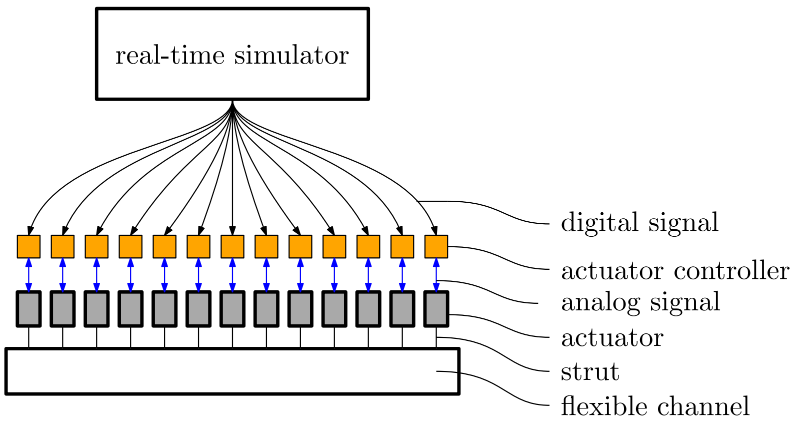


Figure 22: The architecture design of the prototype linear peristaltic pump.

The output of the real-time simulator goes directly to the actuator controller. Which replicates the digital signal to an electric signal which is send to the actuator. The electric signal can be created by using pulse width modulation when a passive low-pass filter is used. When using an electromagnetic actuator with an inductor and resistor, the filter is already in the circuit. The only restriction is that the frequency of the PWM signal is sufficiently high.

A Lorentz actuator is recommended since the dynamics are very linear and it contains the passive low-pass filter. Due to the linear dynamics a closed-loop feedback controller is not necessary. The Lorentz type actuators can be controlled open-loop.

The actuators are connected via struts to the flexible wall of the channel, which has a negligible amount of movement in the longitudinal direction. The length of the channel must at least be long enough to obtain one complete waveform. The depth of the channel is very large with respect to the height such that it can be neglected. By using this channel dimensions, the performance of the pump is easier to compare with theory. However different dimensions can also be used when te focus is on performance and not on theory.

The proposed final design does not have a real-time simulator due to the high cost. The network is created with electronics and implemented in the circuit which drives the actuator. The proposed architecture design is in Figure 23.

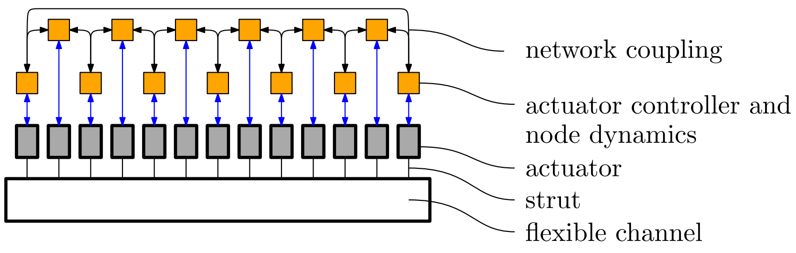


Figure 23: The architecture design of the linear peristaltic pump

The dimensions and shape of the channel are depending on the test results with the prototype. The used network topology for the controller will be covered in the next chapter.

7 Proposed controller design

The next step is to choose the network topology such that the linear peristaltic pump creates the desired motion.

In the previous chapters it is observed that the shape of waveform does not only depend on the chosen topologies, but mostly on the dynamics of the nodes. Assumed is that the dynamics of the nodes are chosen such that it fulfills all conditions in Chapter 3.2. For the prototype it is assumed that the time can be scaled via parameter α such that

$$\begin{aligned} \frac{dx_j}{dt} &= \alpha\left(Ax_j + B(u_j - z_j^3)\right) \\ \frac{1}{\alpha}\frac{dx_j}{dt} &= Ax_j + B(u_j - z_j^3) \\ \frac{dx_j}{d\tau} &= Ax_j + B(u_j - z_j^3) \end{aligned} \tag{21}$$

the oscillation frequency can be changed. Hence the new time scale τ is expressed as $\tau = \alpha t$. For the final design the α parameter could be fixed such that the dynamics can be created with electronic hardware. The oscillation frequency could still be changed a little by changing the coupling strength.

7.1 Proposed network topology

In Chapter 4 it is observed that when k nodes are coupled in a bidirectional ring, where k is an odd number, the phase difference between the output of certain nodes is equally shifted $\frac{2\pi}{k}$. Hence the desired phase difference between the actuators can be controlled by choosing the number of nodes within the ring, where the only restriction is that k is odd and $k \geq 3$.

The direction of the wave in a ring network is still unknown since it depends on the initial condition. However it is observed that the direction can be influenced by changing the topology from a bidirectional network to an unidirectional network. The direction of the coupling between the nodes determines the direction of the waveform. The direction of the wave can be easily changed by taking the transposed of the coupling matrix Γ_0. With an unidirectional network stability is not guaranteed, since it is not proven. However the behavior of the network is very similar to a bidirectional network. It is assumed that it can be proven to be ultimately bounded by using the same or maybe some additional conditions as in Chapter 3.2.

When it is desired that there is more than one waveform present in the linear peristaltic pump, more outputs than nodes are required. There are two options to solve this problem. Multiple topologies could be combined as described in Chapter 4.5. The disadvantage is that it takes longer before the networks reaches its periodic solution. Furthermore the proposed prototype of the pump uses a real-time simulator. Simulating a larger network will also effect the required computational power. The second option is duplicating the output signals of the network as reference or control signal for the other actuators, which requires no extra computational power of the real-time simulator. Therefore duplicating the outputs of the network is the proposed solution for the prototype. This solution could also be used in the final design. However when modularity is desired and slow convergence to the periodic solution is allowed. Than the combined network with multiple unidirectional rings coupled in a bidirectional ring can be used. The number of repetitions of the unidirectional ring must be two times the required number of present waveforms. Only the rings which are inphase with the first ring can be used to control the actuators.

The use of line networks is discouraged since the phase difference between nodes is not uniformly spread. When the phase difference between all nodes is the same, the output of the network can be directly used to drive actuators equally spaced along the channel. When the phase difference is not equal, it is harder to create the peristaltic motion and therefore not recommended.

7.2 Proposed output configuration

As mentioned before, the phase difference between certain nodes in a ring network can be determined by selecting the number of nodes in the network. The phase difference between node j and $j+2$ equals $\frac{2\pi}{k}$, which also holds for the nodes $k-1$ and 1. Therefore the first actuator is linked to the first node of the ring and the second actuator to the third node and so on, until all the odd nodes are linked to the actuators. Then the next actuator is linked to the second node, the following actuator to the fourth node and so on until all even nodes are linked. When all nodes are linked to the actuators the signal can be duplicated and distributed over the remaining actuators by using the same pattern. A visual representation for a network with $k = 5$ is given in Figure 24. The other solution is by creating a combined network with an even number of repetitions. The distribution of the signals from the nodes to the actuators is the same. However only the nodes in the rings which are in-phase with the first ring are used.

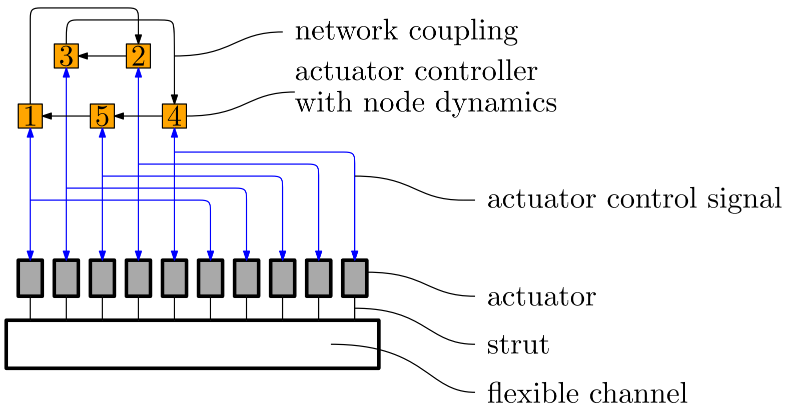


Figure 24: The architecture design of the linear peristaltic pump with the control network.

The outputs of the nodes are linked to the actuators, but the signal is not useful yet. The maximum amplitude of the output is depending on many different variables, such as the topology, the distance to the bifurcation point and the dynamics of the network. Therefore the output should be normalized such that the amplitude of the wave is always the same.

It is assumed that the network creates a trajectory for the wall or a pressure profile. The most convenient is to only compress the channel, where 0 means that it is in rest and a positive value means compression. The current output signals switches sign, while it is desired that there is only a positive value. The bias of the signal is 0, therefore the output is given a certain offset.

To study the flow of the linear peristaltic pump it can be useful to change the compression rate. Therefore the most idealized output from the network is $0 \leq y_j \leq 1$, such that it can be multiplied with the desired compression rate c. Therefore the new output signal $\bar{y}_j$ can be expressed as

$$\bar{y}_j = \frac{c}{2}\left(\frac{y_j}{\max(y_j)} + 1\right). \tag{22}$$

An example of the signal can be viewed at https://youtu.be/m4yFdW7tmxs, where the signal is also mirrored to give a better insight in the channel deformation. Another possibility is to take the absolute value of the signal instead of adding an offset.

$$\bar{y}_j = c\left|\frac{y_j}{\max(y_j)}\right| \tag{23}$$

An example of the signal is at https://youtu.be/D8OMzF7Thxg, which creates another type of compression and doubles the frequency.

The recommended signal modification is depending on the desired profile and needs further research.

8 Outlook

8.1 Conclusion

The goal of the project was to find an application of pattern generation in diffusive networks within linear peristaltic pumping.

8.1.1 The control parameters of the diffusive network

The desired control parameters for a linear peristaltic pump are the direction of the wave; phase difference between the actuation points; the actuation frequency; and the amplitude. The used topology to create the reference or control signal is an unidirectional ring with an odd number of nodes.

The direction of the wave is the same direction as the applied coupling within the unidirectional ring topology. Therefore the direction can be changed by changing the coupling direction, which is the same as taking the transposed of the coupling matrix Γ.

The phase difference between the actuation points can be controlled by changing the number of nodes k. The phase difference between the output signals of node j and $j+2$ and between node k and 2 is $\frac{2\pi}{k}$. The only restriction to create this phase difference is that the number of nodes k is an odd number. Choosing an even number, will give a phase difference of 0 between the output signals of node j and $j+2$ and a phase difference of π between the output of node j and $j+1$.

The actuation frequency can be estimated when the coupling strength γ is chosen close to the Hopf bifurcation of the closed loop linear system.

The amplitude of the signal is depending on many different parameters such as the coupling strength and the node dynamics. To control this parameter the output should be normalized and multiplied with a gain factor such that the desired amplitude is achieved.

The only disadvantage of the unidirectional ring network topology is, that it has not been proven to be oscillatory in the sense of Yakubovich. However it is assumed that it is possible to prove since the outcome of the simulations are very similar to the outcomes of the bidirectional ring, which are proven to be oscillatory in the sense of Yakubovich for a sufficiently large coupling strength.

8.1.2 The design of the linear peristaltic pump

The diffusive network as a controller or reference generator for the linear peristaltic pump, can be realized in two different ways depending on the application. For prototyping proposes the network outputs can be created by using a real-time simulation with hardware in the loop. With this solution different dynamics can be easily used to test or validate the effect on the flow rate of the pump.

The second solution, with the focus on more industrial applications, is to create the network by using electronics. The circuit with the node dynamics could be implemented within the actuator circuit.

8.2 Recommendations

It is shown that an unidirectional ring network creates a wave-like pattern and how it can be influenced by changing the topology or coupling direction. However the shape of the wave-like pattern is mainly depending on the dynamics of the nodes. The effect of pole and zero placement of the linear feedback is studied within this report. The effect of the nonlinear feedback is not examined, but can have an additional influence on the wave-form. Therefore it is recommended to do further investigation on the effect of the node dynamics on the waveform since it is not yet known how to create the desired waveform.

Furthermore it is recommended to build a prototype by using the real-time simulator, such that it is possible to test the effect of different network dynamics on the flow rate easily.

References

[1] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, (Special Issue):642–653, March 2008.

[2] A. Pogromsky, N. Kuznetsov, and G. Leonov. Pattern generation in diffusive networks: how do those brainless centipedes walk? *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, December 2011.

[3] D.E. Kaplan, D. Burkett, and L. Warden. Linear peristaltic pump, March 20 1990. US Patent 4,909,710.

[4] A.S. Borsanyi and D.E. Bobo. Linear peristaltic pumping apparatus and disposable casette therefor, January 15 1985. US Patent 4,493,706.

[5] A.S. Borsanyi. Collapsible conduit for linear peristaltic pump and method of making the same, August 8 1989. US Patent 4,854,836.

[6] P. Manella. Peristaltic pump with means compressing its tube in two directions, November 24 1981. US Patent 4,302,164.

[7] J. Bradley. Linear peristaltic pump, December 31 1985. US Patent 4,561,830.

[8] R.D. Irvin, D. Burkett, D.E. Kaplan, and R.J. Harvey. Peristaltic pump with mechanism for maintaining linear flow, September 4 1990. US Patent 4,954,046.

[9] A. Tsoukalis. Linear peristaltic pump, November 9 1999. US Patent 5,980,490.

[10] R.J. Hill, J.H. Monti, J.A. Oliver, G. Lindemann, and H.C. Copp. Linear peristaltic pump with reshaping fingers interdigitated with pumping elements, May 22 2001. US Patent 6,234,773.

[11] Jiwen Xiang, Ziliang Cai, Yong Zhang, and Wanjun Wang. A micro-cam actuated linear peristaltic pump for microfluidic applications. *Sensors and Actuators A: Physical*, 251:20 – 25, 2016.

[12] V. Lapsa and A. Krasnikovs. Linear peristaltic pump, December 12 2011. LV Patent 14389.

[13] RTU Concrete Mechanics Laboratory. Concrete mechanics laboratory peristaltic pump. `http://www.bml.rtu.lv/en/page/28/65/72`, October 7 2016.

[14] D.L. Cummins. Linear peristaltic pump, February 19 1985. US Patent 4,500,266.

[15] I.J. Phallen and D.N. Vogt. Linear peristaltic pump, April 10 2001. US Patent 6,213,739.

[16] F.T. Hartley. Micromachined peristaltic pump, January 6 1998. US Patent 5,705,018.

[17] L. Maddoui, F.Z. Kadid, and R. Abdessemed. Linear peristaltic pump based on electromagnetic actuators. *Serbian Journal of Electrical Engineering*, 11(3):457–464, October 2014.

[18] Insoo Lee, Pyohwan Hong, Chanseob Cho, Byeungleul Lee, Kyunghan Chun, and Bonghwan Kim. Four-electrode micropump with peristaltic motion. *Sensors and Actuators A: Physical*, 245:19 – 25, 2016.

[19] Tetsuya Iwasaki. Multivariable harmonic balance for central pattern generators. *Automatica*, (44):3061–3069, November 2008.

[20] M. D. Omar Faruque, T. Strasser, G. Lauss, V. Jalili-Marandi, P. Forsyth, C. Dufour, V. Dinavahi, A. Monti, P. Kotsampopoulos, J. A. Martinez, K. Strunz, M. Saeedifard, Xiaoyu Wang, D. Shearer, and M. Paolone. Real-time simulation technologies for power systems design, testing, and analysis. *IEEE Power and Energy Technology Systems Journal*, 2(2):63–73, June 2015.

[21] Thomas Walker Latham. Fluid motions in a peristaltic pump. Master's thesis, Massachusetts Institue of Technology, June 1964.

[22] M. Y. Jaffrin and A. H. Shaprino. Peristaltic pumping. *Annual Review of Fluid Mechanics*, 3:13–37, January 1971.

[23] K. Thanesh Kumar and A. Kavitha. A review report on recent developments in peristaltic transport of physiological fluids. *international Journal of Pharmacy & Technology*, 8(2):4105–4120, June 2016.

A MATLAB script for the simulation of a network with one topology

```
%% General Information
%  Title:   Simulation and Analysis of Simple Networks
%  Author:  R.J.R. van Kampen
%  Date:    23-09-2016
clear all; clc;
%% General User Settings
% Number of nodes k
k = 19;
% Poles and Zeros of the Matrix A (if A is not standard)
Poles = [-1, -1+1j, -1-1j];
Zeros = [1+j, 1+j];

% Network Topology:
    % 1) Bidirectional Ring
    % 2) Bidirectional Line
    % 3) Unidirectional Ring
    % 4) Unidirectional Line
Topology = 3;
% Unidirectional Direction ('r' or 'l')
dir = 'r';
% Distance from bifurcation point in terms of a.
Efac = 1.2;
% Oscilation Frequency Multiplier
Kf = 1;

% Simulation Timespan
TspanSim = [0 10.1e3];

% Visualisation Options (off = 0, on = 1)
Plt.ClosePrevPlts  = 1;
Plt.Frequency      = 1;
Plt.Normal_Full    = 1;
Plt.Normal_Short   = 1;
Plt.Interactive    = 0;
Plt.Animation      = 0;

% Analysis Timespan
TspanAn = [max(TspanSim)-100,max(TspanSim)];
% Interpolation stepsize
fs_interp = 1e-3;
% Interactive Visualisation Stepsize
fs_vis    = 1e-3;

% Animation Settings:
fps = 30;
Speed = 2.5;
%% Create All System Parameters
% System Parameters
A = [1 -1 1; 1 0 0; -4 2 -3];            % Standard System
% [A, bif] = PZPofA(Poles,Zeros);
A = Kf*A;
B = Kf*[0 0 1].';
C = [0 0 1];
P = lyap(A.',eye(size(A)));
Z = B.'*P;

% Create Coupling Matrix
G = CreateSimpleCouplingMatrix(k,Topology,dir);
% Compute eigenvalues and correspoding eigenvectors
[eigV,eigL] = eig(G);
eigL = diag(eigL);
```

```

% Compute Bifurcation Point
a = (sqrt(13)-1)/2;                            % Standard System
% a = bif;
min_gamma = a/max(real(eigL));
f_osc_ther = max(imag(eig(A-min_gamma*B*C)))/2/pi;
%% Simulation
x0 = [];
for ii = 1:k
    x0 = [x0; [0 0 (-1)^(ii-1)*0.1].'];
end

% Solving ODE:
Options = odeset('RelTol',1e-4,'AbsTol',1e-6);
Gsim = Efac*min_gamma*G;
tic
[T,X] = ode45(@(t,x)dxSimpleNetwork(t,x,A,B,C,Z,Gsim,k),TspanSim, x0, Options);
Y = (kron(eye(k),C)*X.').';
toc
%% Post Processing
% Interpolation
Tn = min(TspanAn):fs_interp:max(TspanAn);
samplePoints = {T,1:k};
F = griddedInterpolant(samplePoints,Y,'spline');
queryPoints = {Tn,1:k};
Yn = F(queryPoints);

% Frequency Analysis
nfft = 2^nextpow2(length(Tn));
Freq = (1/fs_interp)/2*linspace(0,1,nfft/2+1);

Yf = fft(Yn,nfft)/length(Tn);

abs_Yf = 2*abs(Yf(1:nfft/2+1,:));
[f_osc_sim,idx] = max(abs_Yf);
f_osc_sim = Freq(idx(1));

fprintf('----------------------------------------------------------------\n');
fprintf('The expected oscilation frequency: \t\t%0.3f Hz\n',f_osc_ther);
if max(Freq(idx))-min(Freq(idx)) < 1e-15
    fprintf('The simulated oscilation frequency: \t%0.3f Hz\n',f_osc_sim);
else
    fprintf('There is a difference between the osccilation frequencies\n');
    fprintf('The ossciliation Frequencies in Hz are');
    disp(Freq(idx));
end
fprintf('----------------------------------------------------------------\n');

% Phase difference Analysis
order = [1:2:k,2:2:k];
phase_diff2 = angle(Yn(end,:))/pi;
for ii = 1:k-1
    phase_diff(ii) =...
        acos(dot(Yn(:,order(ii)),Yn(:,order(ii+1)))/(norm(Yn(:,order(ii)))...
        *norm(Yn(:,order(ii+1))))))./pi;
    fprintf('The phase difference between node %d and %d is: \t 2*pi/ %0.4f\n',...
        [order(ii),order(ii+1),2/phase_diff(ii)])
end
fprintf('----------------------------------------------------------------\n');

% Plots
if Plt.ClosePrevPlts == 1
    close all;
end
if Plt.Frequency == 1
    figure('name','Single-sided amplitude spectrum of signal y','numbertitle','off')
    semilogx(Freq,abs_Yf); grid on;
    title('Single-Sided Amplitude Spectrum of $y_i(t)$','Interpreter','latex');
    xlabel('Frequency [Hz]','Interpreter','Latex');
    ylabel('$|Y(f)|$','Interpreter','Latex');
end
if Plt.Normal_Full == 1
    figure('name','Simulation result from begin to end','numbertitle','off')
    low_prim = min(factor(k));
    plot(T,Y(:,1:low_prim),'linewidth',1.5); hold on;
    plot(T,Y(:,low_prim:end)); grid on;
    xlabel('$t$','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    if Topology == 1
        title(['A ring of ',num2str(k),' bidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 2
        title(['A line of ',num2str(k),' bidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 3
        title(['A ring of ',num2str(k),' unidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 4
        title(['A line of ',num2str(k),' unidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    end
end
if Plt.Normal_Short == 1
    figure('name','Simulation result on analysis interval','numbertitle','off')
    low_prim = min(factor(k));
    plot(Tn,Yn(:,1:low_prim),'linewidth',1.5); hold on;
    plot(Tn,Yn(:,low_prim:end)); grid on;
    xlabel('$t$','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    if Topology == 1
        title(['A ring of ',num2str(k),' bidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 2
        title(['A line of ',num2str(k),' bidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 3
        title(['A ring of ',num2str(k),' unidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    elseif Topology == 4
        title(['A line of ',num2str(k),' unidirectional coupled systems with ...
            $\gamma =$ ',num2str(Efac),'a'],'Interpreter','latex')
    end
end
if Plt.Interactive == 1
    Interactive_Network_Plotter(T,Y,Efac*min_gamma,TspanAn,fs_vis)
    Interactive_Network_Plotter_FINAL(T,Y,Efac*min_gamma,TspanAn,fs_vis)
end
if Plt.Animation == 1
    NNRP(T,Y,Efac,TspanAn,Speed,fps,1,Topology)
    NNRP_FINAL(T,Y,Efac,TspanAn,Speed,fps)
end
```

B MATLAB script for the simulation of a network with combined topologies

```
%% General Information
%  Title:   Simulation and Analysis of Advanced Networks
%  Author:  R.J.R. van Kampen
%  Date:    26-09-2016
clear all; clc;
%% General User Settings
% Number of nodes k
k = 5;
% Number of Ring Repetitions
q = 4;
% Poles and Zeros of the Matrix A (if A is not standard)
Poles = [-1, -1+1j, -1-1j];
Zeros = [1+j, 1+j];

% Network Topology:
    % 1) Bidirectional Rings coupled in a Bidirectional Ring
    % 2) Bidirectional Rings coupled in a Bidirectional Line
    % 3) Unidirectional Rings coupled in a Bidirectional Ring
    % 4) Unidirectional Rings coupled in a Bidirectional Line
Topology = 3;
% Unidirectional Direction ('r' or 'l')
dir = 'r';
% Distance from bifurcation point in terms of a.
Efac = 1.2;
% Oscilation Frequency Multiplier
Kf = 1;

% Simulation Timespan
TspanSim = [0 30.1e3];

% Visualisation Options (off = 0, on = 1)
Plt.ClosePrevPlts  = 1;
Plt.Frequency      = 1;
Plt.Normal_Full    = 1;
Plt.Normal_Short   = 1;
Plt.Interactive    = 0;
Plt.Animation      = 0;

% Analysis Timespan Ow
TspanAn = [max(TspanSim)-100,max(TspanSim)];
% Interpolation stepsize
fs_interp = 1e-3;
% Interactive Visualisation Stepsize
fs_vis    = 1e-3;

% Animation Settings:
fps = 30;
Speed = 2.5;
%% Create All System Parameters
% System Parameters
A = [1 -1 1; 1 0 0; -4 2 -3];            % Standard System
% [A, bif] = PZPofA(Poles,Zeros);
A = Kf*A;
B = Kf*[0 0 1].';
C = [0 0 1];
P = lyap(A.',eye(size(A)));
Z = B.'*P;

% Create Coupling Matrix
[Gs, Gc]  = CreateAdvancedCouplingMatrix(k,q,Topology,dir);
% Compute eigenvalues and correspoding eigenvectors
```

```
[eigV,eigL] = eig(Gs);
eigL = diag(eigL);

% Compute Bifurcation Point
a = (sqrt(13)-1)/2;                             % Standard System
% a = bif;
min_gamma = a/max(real(eigL));
f_osc_ther = max(imag(eig(A-min_gamma*B*C)))/2/pi;
%% Simulation
x0 = [];
for ii = 1:k*q
    x0 = [x0; [0 0 (-1)^(ii-1)*0.1].'];
end

% Solving ODE:
Options = odeset('RelTol',1e-4,'AbsTol',1e-6);
Gsim = Efac*min_gamma*Gc;
tic
[T,X] = ode45(@(t,x)dxSimpleNetwork(t,x,A,B,C,Z,Gsim,k*q),TspanSim, x0, Options);
Y = (kron(eye(k*q),C)*X.').';
toc
%% Post Processing
% Interpolation
Tn = min(TspanAn):fs_interp:max(TspanAn);
samplePoints = {T,1:k*q};
F = griddedInterpolant(samplePoints,Y,'spline');
queryPoints = {Tn,1:k*q};
Yn = F(queryPoints);

% Frequency Analysis
nfft = 2^nextpow2(length(Tn));
Freq = (1/fs_interp)/2*linspace(0,1,nfft/2+1);

Yf = fft(Yn,nfft)/length(Tn);

abs_Yf = 2*abs(Yf(1:nfft/2+1,:));
[f_osc_sim,idx] = max(abs_Yf);
f_osc_sim = Freq(idx(1));

fprintf('-------------------------------------------------\n');
fprintf('The expected oscilation frequency: \t\t%0.3f Hz\n',f_osc_ther);
if max(Freq(idx))-min(Freq(idx)) < 1e-15
    fprintf('The simulated oscilation frequency: \t%0.3f Hz\n',f_osc_sim);
else
    fprintf('There is a difference between the osccilation frequencies\n');
    fprintf('The osscillation Frequencies in Hz are');
    disp(Freq(idx));
end
fprintf('-------------------------------------------------\n');

% FFT plot
if Plt.ClosePrevPlts == 1
    close all;
end
if Plt.Frequency == 1
    figure('name','Single-sided amplitude spectrum of signal y','numbertitle','off')
    semilogx(Freq,abs_Yf); grid on;
    title('Single-Sided Amplitude Spectrum of $y_i(t)$','Interpreter','latex');
    xlabel('Frequency [Hz]','Interpreter','Latex');
    ylabel('$|Y(f)|$','Interpreter','Latex');
end
if Plt.Normal_Full == 1
    figure('name','Simulation result from begin to end','numbertitle','off')
    low_prim = min(factor(k));
    plot(T,Y(:,1:low_prim),'linewidth',1.5); hold on;
    plot(T,Y(:,low_prim:end)); grid on;
    xlabel('$t$','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    switch Topology
        case 1
            title({[num2str(q),' Bidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Ring',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 2
            title({[num2str(q),' Bidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Line',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 3
            title({[num2str(q),' Unidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Ring',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 4
            title({[num2str(q),' Unidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Line',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
    end
end
if Plt.Normal_Short == 1
    figure('name','Simulation result on analysis interval','numbertitle','off')
    low_prim = min(factor(k));
    plot(Tn,Yn(:,1:low_prim),'linewidth',1.5); hold on;
    plot(Tn,Yn(:,low_prim:end)); grid on;
    xlabel('$t$','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    switch Topology
        case 1
            title({[num2str(q),' Bidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Ring',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 2
            title({[num2str(q),' Bidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Line',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 3
            title({[num2str(q),' Unidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Ring',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
        case 4
            title({[num2str(q),' Unidirectional Rings with ', num2str(k),'-nodes ...
                coupled in a Bidirectional Line',...
                    ',\quad$\gamma =$ ',num2str(Efac),'a']},'Interpreter','latex');
    end
end

if Plt.Interactive == 1
    Interactive_Advanced_Network_Plotter(T,Y,k,q,Efac*min_gamma,TspanAn,fs_vis);
    Interactive_Network_Plotter(T,Y,Efac,TspanAn,fs_vis)
    Interactive_Network_Plotter_FINAL(T,Y,Efac*min_gamma,TspanAn,fs_vis)
end
if Plt.Animation == 1
    NNRP(T,Y,Efac,TspanAn,Speed,fps,2,[Topology,k,q])
    NNRP_FINAL(T,Y,Efac,TspanAn,Speed,fps)
end
```

C   MATLAB functions used in the simulations scripts

C.1   Create Advanced Coupling Matrix

```
%% General Information
%  Title:   Create Advanced Copuling Matrix
%  Author:  R.J.R. van Kampen
%  Date:    26-09-2016
%  Info:    k: Number of nodes in the network
%           T: Topology of the Network
%              1) Bidirectional Rings coupled in a Bidirectional Ring
%              2) Bidirectional Rings coupled in a Bidirectional Line
%              3) Unidirectional Rings coupled in a Bidirectional Ring
%              4) Unidirectional Rings coupled in a Bidirectional Line
%%
function [Gs, Gc]  = CreateAdvancedCouplingMatrix(k,q,T,dir)
    if nargin ≤ 3
        dir = 'r';
    end
    % Bidirectional Rings coupled in a Bidirectional Ring
    if T == 1
        if k == 2
            Gs = eye(k,k);
        else
            Gs = 2*eye(k,k);
        end
        Gs(k,1) = -1;
        Gs(1,k) = -1;
        for ii = 1:k-1
            Gs(ii+1,ii) = -1;
            Gs(ii,ii+1) = -1;
        end
        Gc = zeros(q*k);
        for ii = 1:q
            row = ((ii-1)*k+1):(ii*k);
            col = ((ii-1)*k+1):(ii*k);
            if ii  / q
                Gc(row,col) = Gs + 2*eye(k);
                Gc(row+k,col) = -eye(k);
                Gc(row,col+k) = -eye(k);
            else
               Gc(row,col) = Gs + 2*eye(k);
               Gc(1:k,col) = -eye(k);
               Gc(row,1:k) = -eye(k);
            end
        end
    % Bidirectional Rings coupled in a Bidirectional Line
    elseif T == 2
        if k == 2
            Gs = eye(k,k);
        else
            Gs = 2*eye(k,k);
        end
        Gs(k,1) = -1;
        Gs(1,k) = -1;
        for ii = 1:k-1
            Gs(ii+1,ii) = -1;
            Gs(ii,ii+1) = -1;
        end
        Gc = zeros(q*k);
        for ii = 1:q
            row = ((ii-1)*k+1):(ii*k);
            col = ((ii-1)*k+1):(ii*k);
            if ii == 1 || ii == q
                Gc(row,col) = Gs + eye(k);
                if ii  /q
                    Gc(row+k,col) = -eye(k);
                    Gc(row,col+k) = -eye(k);
                end
            else
                Gc(row,col) = Gs + 2*eye(k);
                Gc(row+k,col) = -eye(k);
                Gc(row,col+k) = -eye(k);
            end
        end
    % Unidirectional Rings coupled in a Bidirectional Ring
    elseif T == 3
        Gs = eye(k,k);
        Gs(k,1) = -1;
        for ii = 1:k-1
            Gs(ii,ii+1) = -1;
        end
        if dir == 'l'
            Gs = Gs.';
        end
        Gc = zeros(q*k);
        for ii = 1:q
            row = ((ii-1)*k+1):(ii*k);
            col = ((ii-1)*k+1):(ii*k);
            if ii  /q
                Gc(row,col) = Gs + 2*eye(k);
                Gc(row+k,col) = -eye(k);
                Gc(row,col+k) = -eye(k);
            else
               Gc(row,col) = Gs + 2*eye(k);
               Gc(1:k,col) = -eye(k);
               Gc(row,1:k) = -eye(k);
            end
        end
    % Unidirectional Rings coupled in a Bidirectional Line
    elseif T == 4
        Gs = eye(k,k);
        Gs(k,1) = -1;
        for ii = 1:k-1
            Gs(ii+1,ii) = -1;
            Gs(ii,ii+1) = -1;
        end
        if dir == 'l'
            Gs = Gs.';
        end
        Gc = zeros(q*k);
        for ii = 1:q
            row = ((ii-1)*k+1):(ii*k);
            col = ((ii-1)*k+1):(ii*k);
            if ii == 1 || ii == q
                Gc(row,col) = Gs + eye(k);
                if ii  /q
                    Gc(row+k,col) = -eye(k);
                    Gc(row,col+k) = -eye(k);
                end
            else
                Gc(row,col) = Gs + 2*eye(k);
                Gc(row+k,col) = -eye(k);
                Gc(row,col+k) = -eye(k);
            end
        end
    end
end
```

C.2 Create Simple Coupling Matrix

```
%% General Information
%  Title:   Create Simple Copuling Matrix
%  Author:  R.J.R. van Kampen
%  Date:    23-09-2016
%  Info:    k: Number of nodes in the network
%           T: Topology of the Network
%              1) Bidirectional Ring
%              2) Bidirectional Line
%              3) Unidirectional Ring
%%
function G = CreateSimpleCouplingMatrix(k,T,dir)
if nargin < 3
    dir = 'r';
end
    % Bidirectional Ring
    if T == 1
        if k == 2
            G = eye(k,k);
        else
            G = 2*eye(k,k);
        end
        G(k,1) = -1;
        G(1,k) = -1;
        for ii = 1:k-1
            G(ii+1,ii) = -1;
            G(ii,ii+1) = -1;
        end
    % Bidirecitonal Line
    elseif T == 2
        G = 2*eye(k,k);
        G(1,1) = 1; G(k,k) = 1;
        for ii = 1:k
            if ((ii ⁄ 1) && (ii ⁄ k))
                G(ii,ii-1) = -1;
                G(ii,ii+1) = -1;
            elseif ii (ii == 1)
                    G(ii,ii+1) = -1;
            elseif  ii (ii == k)
                    G(ii,ii-1) = -1;
            end
        end
    % Unidirectional Ring
    elseif T == 3
        G = eye(k,k);
        G(k,1) = -1;
        for ii = 1:k-1
           G(ii,ii+1) = -1;
        end
        if dir == 'l'
            G = G.';
        end
    % Unidirectional Line
    elseif T == 4
        G = eye(k,k);
        for ii = 1:k-1
           G(ii,ii+1) = -1;
        end
        if dir == 'l'
            G = G.';
        end
end
```

C.3 dxSimpleNetwork

```matlab
%% General Information
%  Title:   dx Simple Network
%  Author:  R.J.R. van Kampen
%  Date:    23-09-2016
%%
function dx = dxSimpleNetwork(t,x,A,B,C,Z,G,k);
    if k < 50   % Is faster for smaller networks
        I = eye(k);
        dx = kron(I,A)*x + kron(I,B)*(((-G)*kron(I,C)*x)-(kron(I,Z)*x).^3);
    else
        y = [];
        for jj = 0:k-1
            xj{jj+1} = x((jj*3+1):(jj*3+3));
            zj{jj+1} = Z*xj{jj+1};
            y        = [y; C*xj{jj+1}];
        end
        u = -G*y;
        dx = [];
        for jj = 1:k
            dx = [dx; A*xj{jj}+B*(u(jj)-zj{jj}^3)];
        end
end
```

C.4 Interactive Advanced Network Plotter

```
%% General Information
%  Title:   Interactive Advance Network Plotter
%  Author:  R.J.R. van Kampen
%  Date:    26-09-2016
%%
function Interactive_Advanced_Network_Plotter(T,Y,k,q,gamma,Tspan,StepSize)
    %% Input error Messages
    if max(Tspan) > max(T)
         error('MyComponent:InvalidInput','Error. \nVideo time is longer then ...
             simulation time.');
    end
    if min(Tspan) < 0
        error('MyComponent:InvalidInput','Error. \nTime cannot be negative.');
    end
    %% Interpolation
    % Set new time vector
    Tn = (min(Tspan):StepSize:max(Tspan)).';
    % Interpolation of signals
    for ii = 1:k*q
        Yn(:,ii) = interp1(T,Y(:,ii),Tn);
    end
    %% Create Figure Layout
    fig = figure('name','Interactive Network Plotter',...
                 'numbertitle','off',...
                 'Position',[100, 100, 1280, 720],...
                 'DockControls','on',...
                 'MenuBar','none',...
                 'Toolbar','figure',...
                 'Units','normalized',...
                 'Resize','on',...
                 'Visible','on');

    % find number of subplots
    row = 1; col = 1;
    while (row*col < q)
        if row <= col
            row = row + 1;
        else
            col = col + 1;
        end
    end
    for ii=0:q-1
        subplot(row,col,ii+1)
        graph{ii+1} = stem(Yn(1,ii*k+1:(ii+1)*k));
        xlim([0, k+1]); ylim([floor(min(min(Yn))),ceil(max(max(Yn)))]);
        xlabel('Nodes','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
        grid on;
        title(['Network group ',num2str(ii+1)],...
                'Interpreter','Latex');
    end
    slid = uicontrol('Style','slider',...
                     'Units','normalized',...
                     'Position',[0.05 0.02 0.9 0.025],...
                     'Value',min(Tn),...
                     'Min',min(Tn),...
                     'Max',max(Tn),...
                     'SliderStep',[StepSize,StepSize*10],...
                     'Callback',@SliderValueChanged);

    txtl = uicontrol('Style','text',...
                     'Units','normalized',...
                     'Position',[0.015 0.0225 0.03 0.02],...
                     'FontSize',9,...
                     'HorizontalAlignment','right',...
```

```
                        'String',num2str(min(Tn)));
    txtr = uicontrol('Style','text',...
                        'Units','normalized',...
                        'Position',[0.95 0.0225 0.03 0.02],...
                        'FontSize',9,...
                        'HorizontalAlignment','left',...
                        'String',num2str(max(Tn)));
    txtc = uicontrol('Style','text',...
                        'Units','normalized',...
                        'Position',[0.40 0.045 0.2 0.03],...
                        'FontUnits','normalized',...
                        'FontSize',0.65,...
                        'HorizontalAlignment','center',...
                        'String',['T = ',num2str(min(Tn),'%10.3f')]);
    uistack(txtc,'bottom');

    function SliderValueChanged(source,callbackdata)
        [mi idx] = min(abs(Tn-source.Value));
        for ii = 0:q-1
            set(graph{ii+1},'YData',Yn(idx,ii*k+1:(ii+1)*k));
            drawnow;
        end
        set(txtc,'String',['T = ',num2str(min(Tn(idx)),'%10.3f')])
    end
end
```

C.5 Interactive Network Plotter

```
%% General Information
%  Title:   Interactive Network Plotter
%  Author:  R.J.R. van Kampen
%  Date:    23-09-2016
%%
function Interactive_Network_Plotter(T,Y,gamma,Tspan,StepSize)
    k = size(Y,2);
    %% Input error Messages
    if max(Tspan) > max(T)
         error('MyComponent:InvalidInput','Error. \nVideo time is longer then ...
             simulation time.');
    end
    if min(Tspan) < 0
        error('MyComponent:InvalidInput','Error. \nTime cannot be negative.');
    end
    %% Interpolation
    % Set new time vector
    Tn = (min(Tspan):StepSize:max(Tspan)).';
    % Interpolation of signals
    for ii = 1:k
        Yn(:,ii) = interp1(T,Y(:,ii),Tn);
    end
    %% Create Figure Layout
    fig = figure('name','Interactive Network Plotter',...
                 'numbertitle','off',...
                 'Position',[100, 100, 1280, 720],...
                 'DockControls','on',...
                 'MenuBar','none',...
                 'Toolbar','figure',...
                 'Units','normalized',...
                 'Resize','on',...
                 'Visible','on');

    ax = axes('Position',[0.05 0.20 0.9 0.75],...
              'Units','normalized',...
              'Box','on');

    graph = stem(Yn(1,:));
    xlim([0, k+1]); ylim([floor(min(min(Yn))),ceil(max(max(Yn)))]);
    xlabel('Nodes','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    grid on;
    title(['A network of ',num2str(k),' diffusively coupled system with $\gamma ...
        = $',num2str(gamma)],...
           'Interpreter','latex')

    slid = uicontrol('Style','slider',...
                     'Units','normalized',...
                     'Position',[0.05 0.02 0.9 0.025],...
                     'Value',min(Tn),...
                     'Min',min(Tn),...
                     'Max',max(Tn),...
                     'SliderStep',[StepSize,StepSize*10],...
                     'Callback',@SliderValueChanged);

    txtl = uicontrol('Style','text',...
                     'Units','normalized',...
                     'Position',[0.008 0.0225 0.04 0.02],...
                     'FontSize',9,...
                     'HorizontalAlignment','right',...
                     'String',num2str(min(Tn)));
    txtr = uicontrol('Style','text',...
                     'Units','normalized',...
                     'Position',[0.95 0.0225 0.05 0.02],...
                     'FontSize',9,...
                     'HorizontalAlignment','left',...
                     'String',num2str(max(Tn)));
    txtc = uicontrol('Style','text',...
                     'Units','normalized',...
                     'Position',[0.05 0.04 0.9 0.05],...
                     'FontSize',12,...
                     'HorizontalAlignment','center',...
                     'String',['T = ',num2str(min(Tn),'%10.3f')]);


    function SliderValueChanged(source,callbackdata)
        [mi idx] = min(abs(Tn-source.Value));
        set(graph,'YData',Yn(idx,:));
        set(txtc,'String',['T = ',num2str(min(Tn(idx)),'%10.3f')])
        drawnow;
    end
end
```

C.6 Final Interactive Network Plotter

```
%% General Information
%  Title:   Interactive Network Plotter
%  Author:  R.J.R. van Kampen
%  Date:    23-09-2016
%%
function Interactive_Network_Plotter_FINAL(T,Y,gamma,Tspan,StepSize)
    k = size(Y,2);
    %% Input error Messages
    if max(Tspan) > max(T)
         error('MyComponent:InvalidInput','Error. \nVideo time is longer then ...
             simulation time.');
    end
    if min(Tspan) < 0
        error('MyComponent:InvalidInput','Error. \nTime cannot be negative.');
    end
    %% Interpolation
    % Set new time vector
    Tn = (min(Tspan):StepSize:max(Tspan)).';
    % Interpolation of signals
    for idx = 1:k
        Yn(:,idx) = interp1(T,Y(:,idx),Tn);
    end
    %% Create Figure Layout
    fig = figure('name','Interactive Network Plotter',...
                 'numbertitle','off',...
                 'Position',[100, 100, 1280, 720],...
                 'DockControls','on',...
                 'MenuBar','none',...
                 'Toolbar','figure',...
                 'Units','normalized',...
                 'Resize','on',...
                 'Visible','on');

    ax = axes('Position',[0.05 0.20 0.9 0.75],...
              'Units','normalized',...
              'Box','on');

    graph(1) = stem(0.5*[Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end)]/max(Yn(:,1))+0.5,'b'); hold on;
    graph(2) = stem(-0.5*[Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end),...
        Yn(1,1:2:end),Yn(1,2:2:end)]/max(Yn(:,1))-0.5,'r'); hold off;
    xlim([0, ceil(4*k+1)]); ylim([-1,1]);
    grid on;
    xlabel('Nodes','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
    title(['A network of ',num2str(k),' diffusively coupled system with $\gamma ...
        = $',num2str(gamma),'a, duplicate 4 times'],...
           'Interpreter','latex')

    slid = uicontrol('Style','slider',...
                     'Units','normalized',...
                     'Position',[0.05 0.02 0.9 0.025],...
                     'Value',min(Tn),...
                     'Min',min(Tn),...
                     'Max',max(Tn),...
                     'SliderStep',[StepSize,StepSize*10],...
                     'Callback',@SliderValueChanged);

    txt1 = uicontrol('Style','text',...
                     'Units','normalized',...
                     'Position',[0.008 0.0225 0.04 0.02],...
```

```
                        'FontSize',9,...
                        'HorizontalAlignment','right',...
                        'String',num2str(min(Tn)));
    txtr = uicontrol('Style','text',...
                        'Units','normalized',...
                        'Position',[0.95 0.0225 0.05 0.02],...
                        'FontSize',9,...
                        'HorizontalAlignment','left',...
                        'String',num2str(max(Tn)));
    txtc = uicontrol('Style','text',...
                        'Units','normalized',...
                        'Position',[0.05 0.04 0.9 0.05],...
                        'FontSize',12,...
                        'HorizontalAlignment','center',...
                        'String',['T = ',num2str(min(Tn),'%10.3f')]);


    function SliderValueChanged(source,callbackdata)
        [mi idx] = min(abs(Tn-source.Value));
        set(graph(1),'YData',0.5*[Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end)]./max(Yn(:,1))+0.5);
        set(graph(2),'Ydata',-0.5*[Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end),...
            Yn(idx,1:2:end),Yn(idx,2:2:end)]./max(Yn(:,1))-0.5);
        set(txtc,'String',['T = ',num2str(min(Tn(idx)),'%10.3f')])
        drawnow;
    end
end
```

C.7 Node Network Response Plotter

```
%% General Information
%  Title:   NNRP (Node Network Response Plotter)
%  Author:  R.J.R. van Kampen
%  Date:    02-09-2016
%% Function
function NNRP(T,Y,gamma,Tspan,Speed,fps,Type1,Type2)
tic
%% Settings and variables
% Number of nodes
k = size(Y,2);
% Default intputs
switch nargin
    case 2
        Tspan = [min(T),max(T)];
        Speed = 5;
        fps   = 30;
    case 3
        Speed = 5;
        fps   = 30;
    case 4
        fps   = 30;
    case 5
        Type1 = 1;
    case 6
        Type2 = k;
end
%% Input error Messages
if max(Tspan) > max(T)
     error('MyComponent:InvalidInput','Error. \nVideo time is longer then ...
         simulation time.');
end
if min(Tspan) < 0
    error('MyComponent:InvalidInput','Error. \nTime cannot be negative.');
end
%% Interpolation
% Set new time vector
Tn = (min(Tspan):Speed/fps:max(Tspan)).';
% Interpolation of signals
for ii = 1:k
    Yn(:,ii) = interp1(T,Y(:,ii),Tn);
end
%% Create images
if Type1 == 1
       switch Type2
           case 1
               Vid = VideoWriter(['Bidirectional_Ring_Network_with_',num2str(k),...
                   '-nodes_', 'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 2
               Vid = VideoWriter(['Bidirectional_Line_Network_with_',num2str(k),...
                   '-nodes_', 'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 3
               Vid = VideoWriter(['Unidirectional_Ring_Network_with_',num2str(k),...
                   '-nodes_', 'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 4
                Vid = ...
                    VideoWriter(['Unidirectional_Line_Network_with_',num2str(k),...
                    '-nodes_', 'gamma=',num2str(gamma),'a2'],'Motion JPEG AVI');
       end
    elseif Type1 == 2
        switch Type2(1)
           case 1
               Vid = VideoWriter([num2str(Type2(3)),'-Bidirectional_Rings_with_',...
                   num2str(Type2(2)),'-nodes_coupled_in_a_Bidirectional_Ring_',...
                   'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 2
                Vid = ...
                    VideoWriter([num2str(Type2(3)),'-Bidirectional␣Rings␣with␣',...
                    num2str(Type2(2)),'-nodes␣coupled␣in␣a␣Bidirectional␣Line␣',...
                   'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 3
               Vid = ...
                   VideoWriter([num2str(Type2(3)),'-Unidirectional␣Rings␣with␣',...
                   num2str(Type2(2)),'-nodes coupled␣in␣a␣Bidirectional␣Ring␣',...
                   'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
           case 4
               Vid = ...
                   VideoWriter([num2str(Type2(3)),'-Unidirectional␣Rings␣with␣',...
                   num2str(Type2(2)),'-nodes␣coupled␣in␣a␣Bidirectional␣Line␣',...
                   'gamma=',num2str(gamma),'a'],'Motion JPEG AVI');
        end
end

Vid.FrameRate = fps;
Vid.Quality = 75;
open(Vid);
figure('name','Simulation of a diffusively coupled ...
    network','numbertitle','off','Position',[100, 100, 1280, 720])
stem(Yn(ii,:)); grid on;
xlim([0, k+1]); ylim([floor(min(min(Yn))),ceil(max(max(Yn)))]);

if k ≤ 50
    Nstep = 1;
elseif k > 50 & k < 100
    Nstep = 5;
elseif k ≥ 100 & k < 200
    Nstep = 10;
elseif k ≥ 200
    Nstep = 25;
end
set(gca,'xtick',0:Nstep:k);

xlabel('Nodes','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
set(gca,'nextplot','replacechildren');
for ii = 1:length(Tn)
    stem(Yn(ii,:));
    if Type1 == 1
       switch Type2
           case 1
               title({['Bidirectional Ring Network with ',num2str(k),'-nodes'];...
                   ['$\gamma =$ ',num2str(gamma),'a\quad speed = ...
                       ',num2str(Speed),'x\quad $t = $ ...
                       ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
           case 2
               title({['Bidirectional Line Network with ',num2str(k),'-nodes'];...
                   ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                       ',num2str(Speed),'x\quad $t = $ ...
                       ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
           case 3
               title({['Unidirectional Ring Network with ',num2str(k),'-nodes'];...
                   ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                       ',num2str(Speed),'x\quad $t = $ ...
                       ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
           case 4
                title({['Unidirectional Line Network with ',num2str(k),'-nodes'];...
                   ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                       ',num2str(Speed),'x\quad $t = $ ...
                       ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
       end
    elseif Type1 == 2
        switch Type2(1)
           case 1
```

```
                title({[num2str(Type2(3)),' Bidirectional Rings with ', ...
                    num2str(Type2(2)),'-nodes coupled in a Bidirectional Ring'];...
                    ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                        ',num2str(Speed),'x\quad $t = $ ...
                        ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
            case 2
                 title({[num2str(Type2(3)),' Bidirectional Rings with ', ...
                     num2str(Type2(2)),'-nodes coupled in a Bidirectional Line'];...
                    ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                        ',num2str(Speed),'x\quad $t = $ ...
                        ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
            case 3
                title({[num2str(Type2(3)),' Unidirectional Rings with ', ...
                    num2str(Type2(2)),'-nodes coupled in a Bidirectional Ring'];...
                    ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                        ',num2str(Speed),'x\quad $t = $ ...
                        ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
            case 4
                title({[num2str(Type2(3)),' Unidirectional Rings with ', ...
                    num2str(Type2(2)),'-nodes coupled in a Bidirectional Line'];...
                    ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                        ',num2str(Speed),'x\quad $t = $ ...
                        ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
        end
    end
    Frame = getframe(gcf);
    writeVideo(Vid,Frame);
end
close(Vid);
toc
end
```

C.8 Final Node Network Response Plotter

```
%% General Information
%  Title:   NNRP (Node Network Response Plotter)
%  Author:  R.J.R. van Kampen
%  Date:    02-09-2016
%% Function
function NNRP_FINAL(T,Y,gamma,Tspan,Speed,fps)
tic
%% Settings and variables
% Number of nodes
k = size(Y,2);
% Default intputs
switch nargin
    case 2
        Tspan = [min(T),max(T)];
        Speed = 5;
        fps  = 30;
    case 3
        Speed = 5;
        fps  = 30;
    case 4
        fps  = 30;
end
%% Input error Messages
if max(Tspan) > max(T)
     error('MyComponent:InvalidInput','Error. \nVideo time is longer then ...
         simulation time.');
end
if min(Tspan) < 0
    error('MyComponent:InvalidInput','Error. \nTime cannot be negative.');
end
%% Interpolation
% Set new time vector
Tn = (min(Tspan):Speed/fps:max(Tspan)).';
% Interpolation of signals
for ii = 1:k
    Yn(:,ii) = interp1(T,Y(:,ii),Tn);
end
%% Create images
Vid = VideoWriter(['NNRP_FINAL'],'Motion JPEG AVI');

Vid.FrameRate = fps;
Vid.Quality = 75;
open(Vid);
figure('name','Simulation of a diffusively coupled ...
    network','numbertitle','off','Position',[100, 100, 1280, 720])
stem(0.5*[Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end)]/max(Yn(:,1))+0.5,'b'); hold on;
stem(-0.5*[Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end),...
    Yn(1,1:2:end),Yn(1,2:2:end)]/max(Yn(:,1))-0.5,'r'); hold off;
xlim([0, ceil(4*k+1)]); ylim([-1,1]);
grid on;

if k ≤ 50
    Nstep = 1;
elseif k > 50 & k < 100
    Nstep = 5;
elseif k ≥ 100 & k < 200
    Nstep = 10;
elseif k ≥ 200
    Nstep = 25;
```

```
end
set(gca,'xtick',0:Nstep:k);

xlabel('Nodes','Interpreter','latex'); ylabel('$y$','Interpreter','latex');
set(gca,'nextplot','replacechildren');
for ii = 1:length(Tn)
    stem(0.5*[Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end)]./max(Yn(:,1))+0.5,'b'); hold on;
    stem(-0.5*[Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end),...
        Yn(ii,1:2:end),Yn(ii,2:2:end)]./max(Yn(:,1))-0.5,'r'); hold off;
    xlim([0, ceil(4*k+1)]); ylim([-1,1]);
    grid on;
    title({['Unidirectional Ring Network with ',num2str(k),'-nodes, and 4 output ...
        repetitions'];...
                   ['$\gamma =$ ',num2str(gamma),'a \quad speed = ...
                       ',num2str(Speed),'x\quad $t = $ ...
                       ',num2str(Tn(ii),'%10.3f')]},'Interpreter','latex');
    Frame = getframe(gcf);
    writeVideo(Vid,Frame);
end
close(Vid);
toc
end
```

C.9 Poles and Zero Placement of A

```matlab
%% General Information
%  Title:   PZPofA (Pole, Zero placement of A)
%  Author:  R.J.R. van Kampen
%  Date:    22-11-2016
%%
function [A, bif] = PZPofA(poles,zeros)
if length(poles) ≠ 3
   error('MyComponent:InvalidInput','Error. \nThree poles are required.');
elseif length(zeros) ≠ 2
    error('MyComponent:InvalidInput','Error. \nTwo poles are required.');
elseif imag(poles(1)) ≠0
    error('MyComponent:InvalidInput','Error. \nOnly the seoncd and third pole ...
        can be complex.');
end

a = abs(real(zeros(1)))+abs(real(zeros(2)));
b = abs(real(zeros(1)))*abs(real(zeros(2)))+abs(imag(zeros(1)))*abs(imag(zeros(2)));
f = abs(real(poles(1))) + abs(real(poles(2))) + abs(real(poles(3)))+a;
c = abs(real(poles(1)))*abs(real(poles(2))) + ...
    abs(real(poles(1)))*abs(real(poles(3))) +...
    abs(real(poles(2)))*abs(real(poles(3))) + ...
        abs(imag(poles(2)))*abs(imag(poles(3))) + a*f-b;
e = b*f -(abs(real(poles(1)))*(abs(real(poles(2)))*abs(real(poles(3))) + ...
    abs(imag(poles(2)))*abs(imag(poles(3)))));


A = [a -b 1; 1 0 0; -c e -f];

a_abc = a;
b_abc = (2*a*f-c-a^2);
c_abc = (a*f^2+a*b+a*c-c*f-a^2*f-e);

bif = (-b_abc+sqrt(b_abc^2-4*a_abc*c_abc))/(2*a_abc);
end
```

D The eigenvalue analysis of the bidirectional ring example network

```
%% General Information
%  Title:   Example Network bRing Eigenvalue Analysis
%  Author:  R.J.R. van Kampen
%  Date:    01-09-2016
clear all; close all; clc;
%% Analysis settings
% Number of Nodes in the network
Nodes = [2,3,4,5,6,8,15];
%% Create coupling and permutation matrices
for ii = 1:length(Nodes)
    N = Nodes(ii);
    if N == 2
        G{ii} = eye(N,N);
    else
        G{ii} = 2*eye(N,N);
    end
    G{ii}(N,1) = -1;
    G{ii}(1,N) = -1;
    for jj = 1:N-1
        G{ii}(jj+1,jj) = -1;
        G{ii}(jj,jj+1) = -1;
        Pi{ii}{jj} = zeros(N,N);
        for kk = 1:N
            row = kk;
            col = kk+1+(jj-1);
            if col > N
                col = col - N;
            end
            Pi{ii}{jj}(row,col) = 1;
        end
    end
    if N  / 2
        for jj = 1:N
            for kk = 1:N
                row = kk;
                col = kk+1+(jj-1);
                if col > N
                    col = col - N;
                end
                Pi{ii}{N+(jj-1)}(row,N+1-col) = 1;
            end
        end
    end
end
%% Check permutation matrices
for ii = 1:length(Nodes)
    for jj = 1:length(Pi{ii})
       if Pi{ii}{jj}*G{ii} == G{ii}*Pi{ii}{jj}
           fprintf('For N = %d, Permutation matrix %d is true\n',[Nodes(ii),jj]);
       else
           fprintf('For N = %d, Permutation matrix %d is false\n',[Nodes(ii),jj]);
       end
    end
    fprintf('----------------------------------------\n');
end
%% Calculate and print the eigenvalues and eigenvectors
syms gamma
for ii = 1:length(Nodes)
    [U{ii},L{ii}] = eig(G{ii});
    Lambda{ii} = diag(L{ii});
    fprintf('Eigenvectors of G for N = %d\n',Nodes(ii));
    disp(U{ii});
    fprintf('Eigenvalues of G for N = %d\n',Nodes(ii));
    disp(gamma*Lambda{ii}.')
    fprintf('------------------------------------------------------------------\n');
end
%% Range etc.
error = 1e-15;
for ii = 1:length(Nodes)
    fprintf('For N = %d:\n',Nodes(ii));
    for jj = 1:length(Pi{ii})
        for kk = 1:length(U{ii})
            check1 = 0; check2 = 0;
            if abs(pinv(orth(eye(Nodes(ii))-Pi{ii}{jj})) * ...
                orth(eye(Nodes(ii))-Pi{ii}{jj}) - ...
                eye(rank(eye(Nodes(ii))-Pi{ii}{jj}))) < error
                if abs(U{ii}(:,kk) - orth(eye(Nodes(ii))-Pi{ii}{jj}) * ...
                    (pinv(orth(eye(Nodes(ii))-Pi{ii}{jj}))*U{ii}(:,kk))) < error
                    fprintf('v_%d is in the range of I-Pi_%d\n',[kk,jj]);
                    check1 = 1;
                end
                if abs(U{ii}(:,kk) - orth(eye(Nodes(ii))+Pi{ii}{jj}) * ...
                    (pinv(orth(eye(Nodes(ii))+Pi{ii}{jj}))*U{ii}(:,kk))) < error
                    fprintf('v_%d is in the range of I+Pi_%d\n',[kk,jj]);
                    check2 = 1;
                end
                if check1 == 1 && check2 == 1
                    fprintf('v_%d is in both ranges Pi_%d!!\n',[kk,jj]);
                end
            else
                warning('The tollerance on the pseudo inverse has been exceeded');
            end
        end
    end
    fprintf('-----------------------------------------\n');
end
```