Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Sharmin Akthar

29 April 2022

Reaching Consensus in Multi-agent Systems

Project Supervisor: Paolo Rapisarda
Second examiner: Mike Wald

A project report submitted for the award of
MEng Electrical and Electronic Engineering

Abstract

The consensus problem is a fundamental problem for Distributed Computing and Multi-Agent Systems (MAS). With thorough research and many different methods of approach, this report discusses a general consensus algorithm for third-order integrator agents with fixed topology. The foundation of the report is built by the background theory, including graph theory, matrix theory and control theory- covering key concepts and context in this paper - and are of interest due to their many applications and flexibility in modern engineering. Existing literature regarding the applications of consensus in Blockchain and MAS is then explored, allowing the larger context to be understood and providing motivation for research. The key findings of this research verify the consensus protocol for networks of high-order integrator agents, but also provides significant work in the case of studying variations to the nominal model with the addition of different sources (noise, sinusoidal input) and time delay effects. The parameter conditions necessary for consensus is established via the Routh-Hurwitz stability criterion, and validated via stability analysis (Root locus and Nyquist criterion). It is shown that the agents achieve consensus before and after adjustments, posing new questions and possible implications of the key findings.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

I confirm that:

I have acknowledged all sources, and identified any content taken from elsewhere.

I have used open source code from source(s) listed below. Each of these uses are explained and cited in the report.

I did all the work myself and have not helped anyone else.

The material in the report is genuine, and I have included all my data/code/designs.

Part of this report was submitted as a progress report for this project.

My work did not involve human participants, their cells or data, or animals.

Signed: Sharmin Akthar Date: 29 April 2022

Acknowledgements

I would like to express my gratitude to my supervisor, Dr Paolo Rapisarda, whose expertise and guidance was invaluable. Thank you for the support and for mentoring me throughout this project.

I'd like to thank my friends for their constant support, advice and courage. Thank you for always being there for me.

Contents

1 Notations

Symbol	Definition
$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\mathbb{Z}$	Set of integers
$\mathfrak{Re}$(z)	Real part of complex number
$\mathfrak{Im}$(z)	Imaginary part of complex number z
$z*$	Complex conjugate of complex number z
$\|z\|$	Absolute value of complex number z
$\otimes$	Kronecker product
$\forall x$	For all elements x
$x \in A$	Element x is in the set A
$A \rightarrow B$	From proposition A follows proposition B
$\dot{x}$	First derivative of x with respect to time
$\mathbf{M}^\top$	The transpose of matrix $\mathbf{M}$
I_n	$n * n$ matrix
OLHP	Open Left Hand Plane
ORHP	Open Right Hand Plane
LTI	Linear time invariant

2 Introduction

2.1 Problem

A distributed system is a group of network components that work together to act as a single entity and a Multi-Agent System (MAS) is a group of (intelligent) agents. For all components to act as a single entity they must agree on 'some' data value i.e. reach a consensus. For consensus to be achieved, algorithms, also know as consensus protocols, are used to choose an outcome that all the components of a system agree on. These protocols must be fault-tolerant, resilient, and designed to deal with a limited number of faulty processes.

There are two methods of approaching the consensus problem: the distributed method and the centralised method. Centralised solutions to decentralised systems are fragile due to single points of failure in the networks that can lead to a singular malfunction, consequently causing the whole system to fail. The drawbacks going into more detail in [4], as a result, centralised solutions are undesirable for consensus protocols as they fail to achieve the goal of high availability and reliability. In turn, motivating the interest in distributed solutions for the consensus problem.

Solving the consensus problem and working towards an ideal solution (with no failures and must tolerate process failures) is necessary due to the range of applications consensus has on networks such as autonomous flight formation, control of communication networks, distributed sensor fusion in sensor networks, swarm-based computing, etc. [4]

2.2 Goals

The objective is to implement and simulate consensus protocols for linear continuous-time multi-agent systems. The project deliverables will be MATLAB programmes and Simulink models to evaluate the performance of various consensus algorithms for systems with fixed topology, under some realistic scenarios (including disturbances and malicious attacks). The first approach to be analysed is that described in the paper by Jiang et al. [1] for higher-order integrator systems consensus. The methodology used will be a foundation for this paper as similarly the dynamics are unknown, but also using [2] and [3] for context and altering the process as best to fit the project.

2.3 Scope

This report will review the literature and relevant background theory, such as graph theory, matrix theory, and control theory, and will study the foundation of the paper [1]. The geometry of agents will be agreed upon and the configuration then simulated on MATLAB to visually represent the complex system. Simulink models will test the approaches in more realistic situations. The theoretical calculations discussed are then verified through analysis, design and implementation of the dynamics found in the de-

sired network. The project deliverables are used throughout the project to support and verify the concepts introduced and aid stability analysis of the Linear time-invariant system. To finalise, introducing an overview of the project and possible future enhancements and developments at the end of the report. Relevant extensions will then be made to further explore the network system, expanding upon the completed research and implementation.

2.4 MATLAB and Simulink

MATLAB is a high performance tool developed by 'Mathworks', primarily used for numerical computations. The software is used by engineers and scientists in industry and academia, because of its efficiency and easy-to-use environment. The implementation of algorithms, plotting of functions, and matrix manipulations (and more) are performed all in one environment. In contrast to performing mathematical operations on large matrices by hand, which is prone to errors and unproductive due to inefficiency. Alternatively, MATLAB is preferred over performing numerical calculations on math computational knowledge engines (i.e. WolframAlpha, Symbolab), or graphing calculators which are not well suited (i.e. Desmos). These listed engines are unfitting for lengthy calculations and complex functions, and as a result cannot be relied upon in comparison.

MATLAB has additional packages, one of which being 'Simulink'. It is a graphical programming environment for modelling, simulating, and analysis of dynamical systems. Simulink is as straightforward to read as a block diagram and is as advantageous because it helps visualise designed systems. Since Simulink is integrated with MATLAB, both can be used in cohesion. For example, importing data and integrating mathematical calculations within Simulink.

The features of MATLAB/Simulink are particularly beneficial, such as the 100+ toolboxes available. The control system toolbox being appropriate for this project as it provides algorithms and apps for analysis and design of linear control systems.

3 Review of Literature

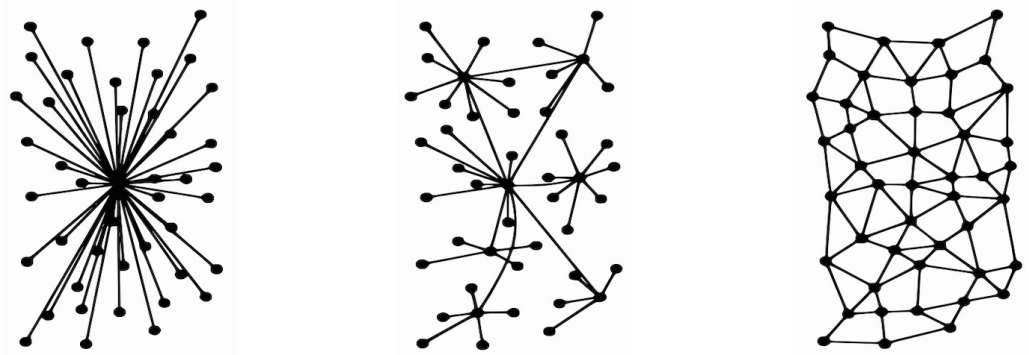
There is an extensive amount of literature regarding consensus and its applications due to its broad nature. Reviewing previous work for the topic of interest is fundamental for controlling the scope of a project and not straying from the original goals.

3.1 Consensus Problem Assumptions

The consensus problem in a distributed system refers to a group of processes that agree on a value after one or more 'nodes' propose a value. It reaches an agreement in the presence of arbitrary process failures. Some assumptions are made:

- There are N number of processes
- There is reliable communication between agents
- The system must be synchronous or asynchronous
- Process failures must be tolerated:
  - Crash failures: the process stops sending values at some point. This is difficult to detect in an asynchronous system.
  - Byzantine failures: components may fail and there is imperfect or unreliable information on whether a component has failed

3.2 Decentralised Systems



(a) Centralised system (b) Decentralised system (c) Distributed system

Figure 1: Different types of networks

A centralised network is a system where there is one central network owner/server with all users connected, and thus only one owner has access to all information. A decentralised network has multiple central owners, with multiple owners of information, and a distributed network is void of centralisation, where every user has access to all data.

Blockchain is a decentralised and distributed system of recording information in a transparent and unalterable way, consisting of several blocks. Each block is a record of transactions of specific data, e.g. cryptocurrency. In the literature, there exist several examples of consensus algorithms.
Examples of consensus algorithms on the Blockchain include [9]:

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Proof of Burn
- Proof of Capacity
- Elapsed time
- Proof of Activity

These are a few of many examples present in the literature of decentralised consensus mechanisms, with consensus algorithms being the set of rules controlling the network. Therefore, there is an agreement on the rules of a specific blockchain and how users can participate in the network. [6], [7].

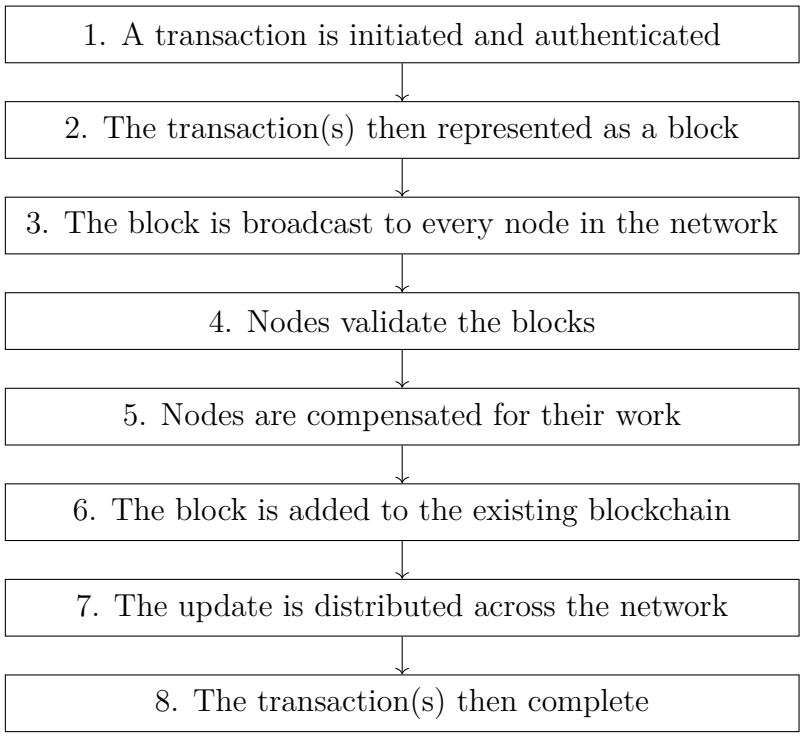


Figure 2: Blockchain transaction illustrated by a flow chart as shown in [8]

Consensus protocols occur during this process, denoted by steps 3 to 5 on Figure 2

These will not be explored due to them being decentralised solutions as mentioned in 2.1, and because the multi-agent system that will be discussed throughout this report is a distributed system (not decentralised). However, this literature is reviewed because it is beneficial to know that consensus mechanisms have many applications, providing motivation for work.

3.3 Consensus Protocol Requirements

Consensus protocols must be fault tolerant and resilient to agents that may fail or be unreliable. Thus, protocols that solve consensus problems are designed to deal with faulty processes, as they must be useful. For this to be achieved, consensus protocols must meet the following requirements [10]:

- Termination: eventually each correct process sets its value
- Agreement: all correct processes must decide on the same value
- Integrity: if all correct processes propose the same value, then any correct process that has decided must choose that value.

3.4 Real World Applications

Kia et al. [4] explains in greater detail the range of applications of dynamic average consensus in network systems. An overview of problems that use consensus algorithms will be selected to motivate research on (general) consensus algorithms and illustrate the relevance of 'consensus'.

- Distributed formation control: Autonomous networked mobile agents are necessary for coverage, surveillance, and patrol in military domains. These tasks require dynamic motion coordination and formation between each mobile agent. The consensus algorithms are the foundation in the design of formation control strategies.
- Distributed state estimation: Wireless sensors are crucial for monitoring and control, with applications such as object tracking, fire detection, etc. A disadvantage is a single fault point (recall definition in 2.1), with an alternative approach to the network system utilising the dynamic average consensus algorithm. As a result, all agents know the size of the entire network and can update the equation of the target locally.

3.5 Discussion

For those unfamiliar with Vision, Learning and Control, the research papers covering consensus are overwhelmingly extensive, covering many theories and concepts that are new. As a result, it is difficult to determine what content is necessary for understanding the general consensus problem and what information is non-essential. Jiang et al. [1] is the foundation of this report; consequently, it is important to get a grasp on the overall piece of literature, as getting a deep understanding of all subtopics discussed in the paper would pose a challenge. Alongside the necessary background theory, some subtopics mentioned in Jiang et al. were researched and reviewed further:

- Gershgorin circle theorem: Gives bounds in the complex plane on the locations of the eigenvalues of a matrix. [11]

- Routh-Hurwitz stability criterion: Determines stability of a high-order polynomial without solving for the roots directly. [12]

The decision to explore these further was made to allow extensions upon the original system and future work. The complete description of the reasons is in the respective section of each topic.

4 Graph Theory

Jiang et al. [1] frequently draws attention to the spanning tree and the relevance of the topology of a graph in relation to consensus, stressing the importance of graph theory. Simply visualising systems is beneficial in the organisation of working and thinking. Graph theory is practical for problems with a graph or network structure as it models relations between objects (nodes).

An undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ has n nodes and edges, graphically represented as circles and lines, respectively, as shown in Figure 3. $\mathcal{V}$ being the Vertex set, $\mathcal{V} = \{v_1, v_2, ...v_N\}$ and $\mathcal{E}$ being the Edge set, $\mathcal{E} = \{(v_i, v_j), \dots\}$. An edge can be described as $e_{ij} = (v_i, v_j)$, with v_i being the neighbour of v_j and the neighbour set= $\mathcal{N}_i$ being the collection of neighbours for a given node. a_{ij} represents the edge connecting i and j.

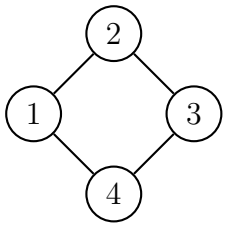


Figure 3: A graph network of 4 nodes

A graph can be mathematically expressed via the adjacency matrix $A \in \mathbb{R}^{N*N}$, with elements $[a_{ij}], \forall i, j \in \{1, 2, 3, ..., N\}$. In an unweighted graph, if an edge is present, a_{ij}=1, and in the absence of an edge, a_{ij}=0. The adjacency matrix for Figure 3 is $\mathcal{A}_1$

$$\mathcal{A}_1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

The out degree of a node, $d^{out}(i) = \sum_{j=1}^{N} a_{ij}$, and the degree matrix $\mathcal{D}^{out} = diag\{d_1, d_2, ...d_N\}$ with the entries being $d^{out}(i), i \in \forall$.

The Laplacian matrix $\mathcal{L}$ of graph $\mathcal{G}$ is calculated by $\mathcal{D} - \mathcal{A}$. Figure 3 would have the Laplacian matrix:

$$\mathcal{L} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

`Matlab: A=[0 1 0 1; 1 0 1 0; 0 1 0 1; 1 0 1 0];` creates Adjacency matrix A.
`node_names={'1','2','3','4'};`
`G= graph(A,node_names);`
`L= laplacian(G);` returns the Laplacian matrix of graph G
`plot(G);` plots the graph G representing the connections described by A.

4.1 Complete Graphs

A graph is said to be simply strong or strongly connected if there exists a path between each pair of nodes in the graph. Figure 3 being strongly connected is represented by Figure 4. Figures 4a and 4b show that a network can be drawn differently but represent the same system, both having the adjacency matrix $\mathcal{A}_2$.

(a) square graph

(b) tetrahedral graph

Figure 4: A strongly connected graph network of 4 nodes

$$\mathcal{A}_2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

4.2 Directed Graphs

A directed graph, also known as a digraph is similar to that of a regular graph. The edges are represented by arrows which can be single-headed or double-headed to show the direction of the edge. Figure 3 modified with directed edges is shown as Figure 5, as well its respective adjacency matrix $\mathcal{A}_3$. Recalling, the element a_{ij} defines if there is an edge connecting from v_i to v_j.

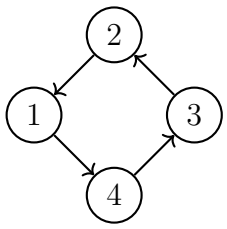


Figure 5: A digraph network of 4 nodes

$$\mathcal{A}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

4.3 Weighted Graphs

The total sum of a weighted edge determines the direction and magnitude of the edge. An adjacency matrix developed into weighted adjacency matrix means the entries of the matrix are the weighted edge between a pair of nodes. Using a modified version of Figure 3 with weighted edges, a new figure and adjacency matrix can be made.

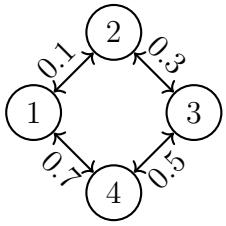


Figure 6: A weighted graph network of 4 nodes

$$\mathcal{A}_4 = \begin{bmatrix} 0 & 0.1 & 0 & 0.7 \\ 0.1 & 0 & 0.3 & 0 \\ 0 & 0.3 & 0 & 0.5 \\ 0.7 & 0 & 0.5 & 0 \end{bmatrix}$$

The graph network used to demonstrate a multi-agent system network of agents in this paper will be undirected (leaderless) and unweighted. This is a choice made in order to represent uniform communication between agents, and is the simplest way of approaching a system which can then be expanded upon for complex systems. The mathematics in the approach remain the same, however, as previously mentioned minor adjustments would be made for directed and weighted graphs regarding calculations, i.e. with $a_{ij} \neq 1$ a likely error would be transcriptional errors, consequently leading to experimental errors when calculating the dynamic of each agent (later discussed in 7).

4.4 Application: Google Maps GPS

Graphs are the backbone of many ideas, concepts and processes in everyday life. A well-known application is Google maps, which uses GPS to find the (shortest/quickest) route most beneficial to the user. Via graph algorithms, the critical path (optimal route) between two destinations (represented as nodes) is calculated. This is only possible because the data structure is represented as a graph, allowing graph and tree search algorithms to be performed and the most optimum path to being found.

5 Matrix Theory

Matrix theory is fundamental to this paper because the theories discussed overlap and link together. For example, recalling the previous section: the adjacency matrix is a basic notation of a graph; this can be seen as matrix theory applied to graph theory. Control theory uses matrices to represent systems and provides characterisations of Linear Time-Invariant (LTI) systems via matrices.

5.1 Eigenvalues and Eigenvectors

For a matrix A, the vector $\mathbf{x}$ points in the same direction as the vector A$\mathbf{x}$ when,

$$\mathrm{A}\mathbf{x} \propto \mathbf{x}$$

and so,

$$\mathrm{A}\mathbf{x}_\lambda = \lambda\mathbf{x}_\lambda$$

where

$$\begin{gathered}\text{eigenvalue } \lambda \in \mathbb{C} \\ \text{eigenvector } \mathbf{x}_\lambda \in \mathbb{C}^n\end{gathered}$$

The values of λ are the eigenvalues of A and are the roots of the characteristic polynomial:

$$\mathcal{X}_A(s) := det(sI - A)$$

Where I is the identity matrix and I_n is the n*n identity matrix.
And the set of eigenvalues of the matrix A is represented by σ_A.

`Matlab` can be used to compute the eigenvalues and the corresponding eigenvectors.
`e=eig(A)` calculates the corresponding eigenvalues of the square matrix A and stores it in `e`.
`[V,D,W]=eig(A)` calculates: `V`, whose columns are the corresponding right eigenvectors. `D`, the diagonal matrix of eigenvalues. And `W`, whose columns are the corresponding left eigenvectors.

5.2 Kronecker Product

The matrices A and B do not necessarily have to be compatible. $A = [a_{ij}] \in \mathbb{R}^{m*n}$ and $B = [b_{ij}] \in \mathbb{R}^{p*r}$ The Kronecker product of A and B is defined as

$$A \otimes B := \begin{bmatrix} a_{11}B & a_{11}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp*nr}$$

`Matlab`: `K=kron(A,B)` calculates the Kronecker tensor product of `A` and `B`, storing it in `K`.

5.3 Characteristic Polynomial

The characteristic of the matrix Ω is expressed as:

$$\prod_{i=1}^{N}(s^m + c_{m-1}s^m + ... + c_1 s + \mu_1)$$

μ_i to the ith eigenvalue of $\Upsilon\mathcal{L}$ with $\mu_1 = 0$.

$$P_\Omega(s) = (s^3 + c_2 s^2 + c_1 s + \mu_1) * (s^3 + c_2 s^2 + c_1 s + \mu_2) * \cdots * (s^3 + c_2 s^2 + c_1 s + \mu_N)$$

The eigenvalues of Ω = roots of characteristic polynomial $P_\Omega(s)$, so it can be written:

$$\sigma_\omega = det(sI - \omega) = \prod_{i=1}^{N}(s^m + c_{m-1}s^m + ... + c_1 s + \mu_1)$$

5.4 Gershgorin Circle Theorem

The Gershgorin circle theorem is a concept introduced and extensively discussed by Jiang et al. [1]. Understanding this concept isn't compulsory for Multi-Agent Systems reaching consensus, but it will be further explored due to the applications of the theorem in relative stability analysis of a system (represented by a polynomial). There are applications of the Gershgorin theorem in which it is used in reduced order modeling and stabilisation for LTI systems. This will not be further explored, but due to the wide applications and importance of this theorem for control, it will be mentioned, allowing room for future work relating to Consensus and Gershgorin. [11] [13]

A strictly diagonally dominant matrix A_{nn} is defined as strictly diagonally dominant (SDD) if:

$$|A_{ii}| > \sum_{j \neq i} |A_{ij}|$$

for $i = 1, 2, .., n$

For a 4×4 square matrix A:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

each row must satisfy the inequality above for a SDD matrix:

$$\begin{aligned} &\text{Row 1} : |a_{11}| > |a_{12}| + |a_{13}| + |a_{14}| \\ &\text{Row 2} : |a_{22}| > |a_{21}| + |a_{23}| + |a_{24}| \\ &\text{Row 3} : |a_{33}| > |a_{31}| + |a_{32}| + |a_{34}| \\ &\text{Row 4} : |a_{44}| > |a_{41}| + |a_{42}| + |a_{43}| \end{aligned}$$

Gershgorin's theorem states that every eigenvalue of matrix A_{nn} satisfies:

$$|\lambda - A_{ii}| \leqslant \sum_{j \neq i} |A_{ij}|$$

for $i \in \{1, 2, .., n\}$

Gershgorin disc:
let

$$d_i = \sum_{j \neq i} |A_{ij}|$$

The ith Gershgorin disc of the matrix A is defined as the set:

$$D_i = \{z \in \mathbb{C} : |z - A_{ii}| \leqslant d_i\}$$

Disk D_i, radius d_i, centred at $(\mathfrak{Re}(A_{ii}), \mathfrak{Im}(A_{ii}))$

6 Control Theory

6.1 Continuous Time-invariant System

6.1.1 State Representation

The general form of a linear-time invariant state system is represented in the matrix form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \vdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1r} \\ b_{21} & \cdots & b_{2r} \\ \vdots & & \vdots \\ b_{n1} & \vdots & b_{nr} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_r \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ c_{m1} & c_{m2} & \vdots & c_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} d_{11} & \cdots & d_{1r} \\ d_{21} & \cdots & d_{2r} \\ \vdots & & \vdots \\ d_{m1} & \vdots & d_{mr} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_r \end{bmatrix}$$

and in the compact form:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

where,

$$x : \mathbb{R} \to \mathbb{R}^n \text{ is the state vector}$$
$$y : \mathbb{R} \to \mathbb{R}^p \text{ is the output vector}$$
$$u : \mathbb{R} \to \mathbb{R}^m \text{ is the input vector}$$

with,

$$\text{System matrix: } A \in \mathbb{R}^{n*n}$$
$$\text{Input matrix: } B \in \mathbb{R}^{n*m}$$
$$\text{Output matrix: } C \in \mathbb{R}^{p*n}$$
$$\text{Feedthrough matrix: } D \in \mathbb{R}^{p*m}$$

`matlab`: `sys=ss(A,B,C,D)` creates the continuous time state-space model above.

6.1.2 Spectral Decomposition

A spectral decomposition of the system matrix A is performed, using the eigenvalues calculated as discussed in section 5.1.

$$A \underbrace{\begin{bmatrix} v_{\lambda_1} & v_{\lambda_2} & \cdots & v_{\lambda_n} \end{bmatrix}}_{=:V} = \underbrace{\begin{bmatrix} v_{\lambda_1} & v_{\lambda_2} & \cdots & v_{\lambda_n} \end{bmatrix}}_{=:V} \underbrace{\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}}_{=:\Delta}$$

And so, because V is a non singular matrix:

$$A = V\Delta V^{-1}$$

$$A^k = V\Delta^k V^{-1}$$

$$= V \begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^k \end{bmatrix} V^{-1}$$

Multiplying this with e^{At} gives the free response of the system, also termed the natural response / mode of the system.

$$e^{At} = V\Delta V^{-1}$$

$$= V \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix} V^{-1}$$

Free response:

$$x(t) = e^{At} * x(0)$$

$$x(t) = V \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix} V^{-1} * x(0)$$

6.1.3 Controllability and Observability

Controllability describes the ability to steer the system. The state system $\dot{\mathbf{x}} = \mathbf{Ax(t)} + \mathbf{Bu(t)}$ is controllable, if and only if the controllability matrix has full rank n (all rows/colums are linearly independent).

$$\mathcal{C}(A, B) := \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \in \mathbb{R}^{n*m \cdot n}$$

`matlab`: controllability can be checked using `O=obsv(A,B)`

Observability infers the state of the system based on the input and output. A system is said to be observable if and only if the observability matrix has full rank n.

$$\mathcal{O}(A, C) := \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \in \mathbb{R}^{n \cdot p * n}$$

A system is said to be minimal if $\mathcal{C}$ is controllable and $\mathcal{O}$ is observable.
`matlab`: observability can be checked using `O=obsv(A,C)`

6.1.4 Transfer Function

The laplace transformation of the state space system,

$$\dot{x}(t) = Ax(t) + Bu(t)$$

is applied to each component, by linearity,

$$\mathcal{L}(\dot{x}) = A \underbrace{\mathcal{L}(x)}_{=:\mathrm{X(s)}} + B \underbrace{\mathcal{L}(u)}_{=:\mathrm{U(s)}}$$

which yields,

$$sX(s) - x(0) = AX(s) + BU(s)$$

simplifying the equations for $X(s)$,

$$(sI_n - A)X(s) = x(0) + BU(s)$$

it concludes,

$$X(s) = (sI_n - A)^{-1}x(0) + (sI_n - A)^{-1}BU(s)$$

Substituting $X(s)$ back into the output equation,

$$Y(s) = CX(s) + DU(s)$$

yields

$$Y(s) = C(sI_n - A)^{-1}x(0) + [C(sI_n - A)^{-1}B + D]U(s)$$

The transfer function matrix is defined as the ratio of output to input:

$$H(s) = \frac{Y(s)}{U(s)}$$

hence, the rational matrix

$$H(s) = C(sI_n - A)^{-1}B + D$$

is the transfer function of the system.

The characteristic equation $U(s)$ is the denominator of the transfer function of $H(s)$ and the roots of $U(s)$ determine the stability of a system. If the roots of the denominator lie in the Open left half plane (OLHP), the system is said to be stable.

6.1.5 Feedback

The system as represented by a closed loop diagram in Figure 7 can be stabilised. This would be by assigning the eigenvalues of the matrix A to the open left half plane, via state feedback. Hence, multiplying the output by a matrix K and setting this as the input.

$$\begin{cases} u(t) = v(t) - K\hat{x}(t) \\ \dot{x}(t) = Ax(t) + Bu(t) \end{cases}$$

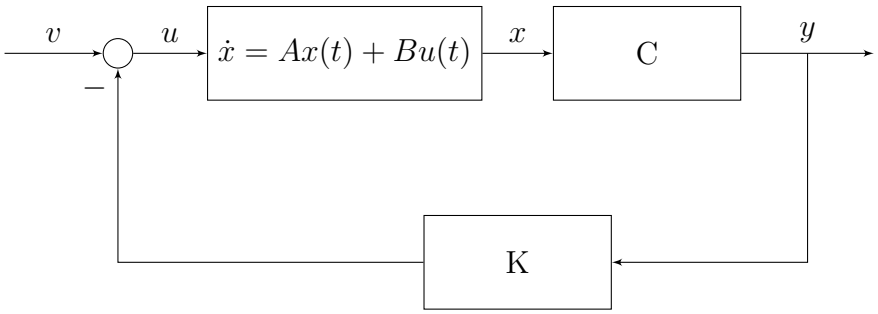


Figure 7: A closed loop system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$\dot{x}(t) = Ax(t) - BK\hat{x}(t) + Bv(t)$$

$$\therefore \dot{x}(t) = (A - BK)\hat{x}(t) + Bv(t)$$

$$y(t) = Cx(t)$$

The eigenvalues of $A-BK$ can be arbitrarily placed if and only if (A,B) is controllable. The poles of the transfer function $H(s)$ are the eigenvalues of the state matrix A

$$H(s) = C(sI_n - A)^{-1}B + D$$

6.1.6 Minimality and Stability

The state representation of H(s) is (A, B, C, D) and is classified as a minimal if the state dimension is minimal among all realisations. Single-input single-output (SISO) systems are systems in which there is a simple single variable control system with one input and one output systems. The following conditions must be met for minimality:

- $H(s) = C(sI - A)^{-1}B + D$ and so, $\mathcal{C}(A, B)$ must be controllable, and $\mathcal{O}(C, A)$ must be observable.

- For SISO systems, $H(s) = C(sI - A)^{-1}B + D$ must be irreducible, and so the numerator and denominator must be co-prime.

- For SISO systems, the dimension of matrix A must be equal to the degree of the denominator of the (irreducible) H(s).

Kalmans Theory states that minimality $\Leftrightarrow$ simultaneous controllability and observability.
For SISO systems, irreducibility $\Leftrightarrow$ minimality

If the realisation is a minimal realisation, it can be said the eigenvalues of matrix A = poles of H(s).

6.1.7 Application

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 \end{bmatrix}$$

$$H(s) = \frac{\beta_0 + \beta_1 s + \beta_2 s^2}{\alpha_0 + \alpha_1 s + \alpha_2 s^2 + s^3}$$

Controllability matrix:

$$\mathcal{C}(A,B) = \begin{bmatrix} 0 & 0 & \textcircled{1} \\ 0 & \textcircled{1} & -\alpha_2 \\ \textcircled{1} & -\alpha_2 & -\alpha_1 - \alpha_2^2 \end{bmatrix}$$

$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ B & BA & BA^2 \\ & & = BA(A) \end{array}$$

Because the antidiagonal has all elements of 1, the matrix is always non-singular, and therefore always controllable.

Observability matrix:

$$\mathcal{O}(C,A) = \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 \\ -\alpha_0\beta_2 & \beta_0 - \alpha_1\beta_2 & \beta_1 - \alpha_2\beta_2 \\ * & * & * \end{bmatrix} \begin{array}{ll} \leftarrow & C \\ \leftarrow & CA \\ \leftarrow & CA^2 = CA(A) \end{array}$$

The realisation H(s) is non-observable if C=0.
The numerator and the denominator of H (s) share a common root if there is cancellation between them. Let $\lambda_1, \lambda_2, \lambda_3$ be the roots of the characteristic polynomial $U(s) = \alpha_0 + \alpha_1 s + \alpha_2 s^2 + s^3$. Then one of such numbers λ_i must be such that $\beta_0 + \beta_1\lambda_i + \beta_2\lambda_i^2 = 0$.

6.2 Continuous Time-variant System

A system is said to be 'time-variant' if the input and output characteristics vary with time.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t)$$

For this case, the delayed output of the system is not equal to the output due to delayed input.

6.3 Discrete Time-invariant System

A system is 'time-invariant' if the way it responds to inputs does not change overtime.

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) + Du(k)$$

In this case, the time invariance property is known as shift invariance.

6.4 Discrete Time-variant System

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

$$y(k) = C(k)x(k) + D(k)u(k)$$

In this case, the output of the system is not equal to the output due to delayed input.

6.5 Routh–Hurwitz Stability Criterion

Recalling Section 6.1.4, a system being classified as stable means that all roots of the characteristic equation (denominator of the transfer function) must lie in the OLHP. Routh Array determines the number of closed-loop poles in the left-half plane. [12]
For the polynomial equation

$$a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n = 0$$

arrange the coefficients in the Routh Array table like so:

$$\begin{array}{l||llll} s^n & a_0 & a_2 & a_4 & \cdots \\ s^{n-1} & a_1 & a_3 & a_5 & \cdots \\ s^{n-2} & b_1 & b_2 & b_3 & \cdots \\ s^{n-3} & c_1 & c_2 & c_3 & \cdots \\ \vdots & \vdots & \vdots & & \\ s^2 & d_1 & d_2 & & \\ s^1 & e_1 & & & \\ s^0 & f_0 & & & \end{array}$$

with:

$$b_1 = \tfrac{a_1 a_2 - a_0 a_3}{a_1} \qquad c_1 = \tfrac{b_1 a_3 - a_1 b_2}{b_1} \qquad d_1 = \tfrac{c_1 b_2 - b_1 c_2}{c_1}$$

$$b_2 = \tfrac{a_1 a_4 - a_0 a_5}{a_1} \qquad c_2 = \tfrac{b_1 a_5 - a_1 b_3}{b_1} \qquad d_2 = \tfrac{c_1 b_3 - b_1 c_3}{c_1}$$

$$b_3 = \tfrac{a_1 a_6 - a_0 a_7}{a_1} \qquad c_3 = \tfrac{b_1 a_7 - a_1 b_4}{b_1} \qquad \cdots$$

Until the array has been completed. The first column of the array must be all of the same sign in order for the roots to lie in the OLHP. If there are sign changes, the number of changes determines how many roots are in the Open Right Hand Plane.

6.6 Application: Mass-Spring-Damper System

The control theory studied will be applied to an example. This is to emphasise the relevance and applicability of control systems theory as it is not only for network systems. The wide applications intertwine different branches of engineering with control. A real-world example of control theory for a mechanical system would be the Mass-Spring-Damper system.

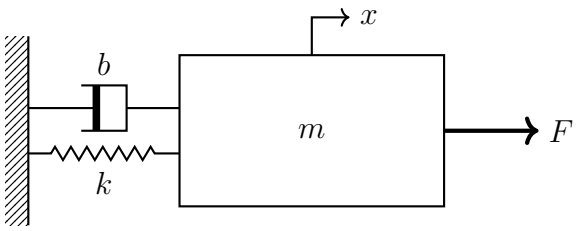


Figure 8: Free-body diagram for Mass-Spring-Damper System

Recalling Hook's law, viscous force and Newtonian laws, equations for force can be derived. With, $f_k = kx$, $f_c = c\dot{x}$, and $\Sigma F_x = m\ddot{x}$, respectively. Thus, the free-body diagram Figure 8 can be simplified to:

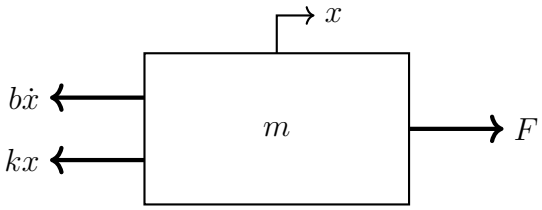


Figure 9: Simplified free-body diagram for Mass-Spring-Damper System

The characteristic of the dynamic state of the system is described by the governing equation:

$$\Sigma F_x = F(t) - b\dot{x} - kx = m\ddot{x}$$

with F(t) being an external force. The governing equation is reduced to two first-order differential equations in order for state-space representation. With state variables:

$$x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

The state equation is:

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

With output equation:

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

The laplace transform (assuming zero initial conditions):

$$ms^2X(s) + bsX(s) + kX(s) = F(s)$$

And therefore the transfer function of the mass-spring-damper system:

$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$

7 Consensus Protocol

7.1 Distributed Consensus

A graph network has N dynamic agents and the dynamic of each agent is described by,

$$\xi_i^{(m)} = u_i$$

m denoting the order of the differential equation system, $m \in \mathbb{Z}^+$. The information of agent i is denoted by $\xi_i \in \mathbb{R}$. The kth order derivative of ξ_i is denoted by $\xi_i^{(k)}$, with feedback gains of absolute information $k \in m-1$. Recall from Section 4, a_{ij} denotes the weight of an edge if present, e_{ij} denotes an edge from agent j to agent i.
The consensus protocol is denoted by u_i and is also referred to as the control input. The distributed consensus protocol for the consensus problem is:

$$u_i = \sum_{k=1}^{m-1} c_k \xi_i^{(k)} - \sum_{j \in \mathcal{N}_i} k_i a_{ij}(\xi_i - \xi_j)$$

with $c_k > 0$.
Given the consensus protocol above, the dynamics of agent i is expressed as:

$$\dot{\overline{\xi}}_i = E_m \overline{\xi}_i - \sum_{j \in \mathcal{N}_i} k_i a_{ij}(\xi_i - \xi_j)$$

With the feedback gain of relative information, $k_i > 0$.
The closed-loop network has the following dynamics:

$$\dot{\overline{\xi}} = [I_N \otimes E_m - (\Upsilon \mathcal{L}) \otimes F_m]\overline{\xi} := \Omega \overline{\xi}$$

And so,

$$\Omega = [I_N \otimes E_m - (\Upsilon \mathcal{L}) \otimes F_m]$$

With,

$$E_m = \begin{bmatrix} \mathbf{0} & I_{m-1} \\ 0 & \theta^T \end{bmatrix}, F_m = \begin{bmatrix} \mathbf{0} & 0 \\ 1 & \mathbf{0}^T \end{bmatrix}, \theta = \begin{bmatrix} -c_1 & \cdots & -c_{m-1} \end{bmatrix}^T, \mathbf{0} = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^T, \Upsilon = diag\{k_1 \cdots k_N\}$$

A general form of Ω for an 4-node system of third order ($N = 4, m = 3$),

$$\Omega = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3k_1 & -c_1 & -c_2 & k_1 & 0 & 0 & k_1 & 0 & 0 & k_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ k_2 & 0 & 0 & -k_2 & -c_1 & -c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ k_3 & 0 & 0 & 0 & 0 & 0 & -2k_3 & -c_1 & -c_2 & k_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ k_4 & 0 & 0 & 0 & 0 & 0 & k_4 & 0 & 0 & -2k_4 & -c_1 & -c_2 \end{bmatrix}$$

A method to find the dynamics of each agent i, $\dot{\overline{\xi}}_i$, can be by performing Ω (the matrix used to describe consensus of the system) * $\overline{\xi}_i$.

$$\overline{\xi}_i = \begin{bmatrix} \xi_1 \\ \dot{\xi}_1 \\ \ddot{\xi}_1 \\ \xi_2 \\ \dot{\xi}_2 \\ \ddot{\xi}_2 \\ \xi_3 \\ \dot{\xi}_3 \\ \ddot{\xi}_3 \\ \xi_4 \\ \dot{\xi}_4 \\ \ddot{\xi}_4 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{1st Agent (Node)} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{2nd Agent (Node)} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{3rd Agent (Node)} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{4th Agent (Node)} \end{matrix}$$

Recall Jiang et al. the protocol of each agent u_i is equivalent to ${\xi_i}^{(m)}$, with m being the order of integration. Therefore, in the example above ($N = 4, m = 3$) the third state component of the result $\dddot{\xi}_i = u_i$.

7.2 C_k Parameters

A Routh-array table can be constructed to examine the parameter conditions. It can be shown that

$$h_1(s) \prod_{i=1}^{N} g_{2i}(s)$$

with

$$h_{m-1}(s) = s^{m-1} + c_{m-1}s^{m-2} + \cdots + c_1$$

$$g_{mi}(s) = s^m + c_{m-1}s^{m-1} + \cdots + c_1 s + \mu_i$$

with

$$0 \neq \mu_i \in \sigma(\Upsilon\mathcal{L})$$

is Hurwitz stable if c_1 satisfies the condition of the parameter for the graph $G(A)$ with fixed topology:

$$c_1 > \max_{0 \neq \mu_i \in \sigma(\Upsilon\mathcal{L})} \frac{|\mathfrak{Im}(\mu_i)|}{\sqrt{\mathfrak{Re}(\mu_i)}}$$

7.3 Convergence

The consensus state of the system, $\chi(\overline{\xi}_0)e_1$ relates the left eigenvector of Ω, $w_l^\top$, equilibria of the system, e_1 and the initial state of the agents of the system, $\overline{\xi}_{i0}$.
The left eigenvector $w_l^\top$ of $\mathcal{L}$ associated with the zero eigenvalue is also known as the vector $w_l^\top$ such that $w_l^\top \mathcal{L} = 0$.

`matlab: [V,D,W]=eig(`$\mathcal{L}$`)` gives the vector `W`, where the first column of `W` is the eigenvector w_l and must then be transposed to give $w_l^\top$.

$$w_l = \begin{bmatrix} w_1 \cdots w_N \end{bmatrix}^\top$$

Equilibria of the system is defined as,

$$e_1 = (\frac{1}{c_1} w_l^\top \otimes \tilde{c}) \in \mathbb{R}^m$$

with

$$\tilde{c} = \begin{bmatrix} c_1 \cdots c_{m-1} 1 \end{bmatrix}^\top$$

The state of agent i at time t is defined as $\overline{\xi}_i(t)$, with the initial state of agent i being $\overline{\xi}_{i0}$.
To conclude, the consensus state is $\chi(\overline{\xi}_0)e_1$, with

$$\chi(\overline{\xi}_0) = \frac{1}{c_1} \sum_{i=1}^{N} w_i \tilde{c}^\top \overline{\xi}_{i0}$$

8 Four Agent System

A connected graph network with unweighted edges and four nodes is used to represent a multi-agent system (MAS). Using the theories and calculations discussed in Sections 4, 5 and 6, the graph network is analysed as mentioned. First, we go through the graph theory and apply the appropriate notations and calculations. As a result forming matrices which in turn produce a state space system representation of the MAS. The consensus algorithm and consensus protocol discussed in Section 7 is also applied to the graph network and the dynamics are computed for every agent.

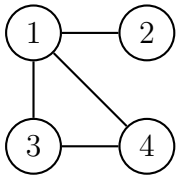


Figure 10: Desired graph network

Vertex set $\mathcal{V} = \{1, 2, 3, 4\}$

Edge set $\mathcal{E} = \{(1, 2), (1, 3), (1, 4), (3, 4)\}$

Neighbour sets:
$\mathcal{N}_1 = \{2, 3, 4\}$
$\mathcal{N}_2 = \{1\}$
$\mathcal{N}_3 = \{1, 4\}$
$\mathcal{N}_4 = \{1, 3\}$

Adjacency Matrix $\mathcal{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$

Degree Matrix $\mathcal{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$

Laplacian Matrix $\mathcal{L} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$

Eigenvalues of $\mathcal{L}$ $\mathtt{e} = \mathtt{eig}(\mathcal{L})$ gives $\mathtt{e} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 4 \end{bmatrix}$

$\therefore \lambda_1 = 0, \lambda_2 = 1, \lambda_1 = 3, \lambda_1 = 4.$

Left eigenvector $w_l^\top$ (calculated as discussed in section 7.3) `[V,D,W]=eig(`$\mathcal{L}$`)` returns

$$\mathtt{W} = \begin{bmatrix} -0.5000 & 0.0000 & 0.0000 & -0.8660 \\ -0.5000 & 0.8165 & 0.0000 & 0.2887 \\ -0.5000 & -0.4082 & -0.7071 & 0.2887 \\ -0.5000 & -0.4082 & 0.7071 & 0.2887 \end{bmatrix}$$

$$\therefore w_l^\top = \begin{bmatrix} -0.5 & -0.5 & -0.5 & -0.5 \end{bmatrix}$$

The network has N dynamic agents and the dynamic of each agent is described as mth order. Therefore, the 4 dynamic agents of third integrator have m=3.

$$\dot{\overline{\xi}} = [I_N \otimes E_m - (\Upsilon\mathcal{L}) \otimes F_m]\overline{\xi}$$
$$\Omega = [I_N \otimes E_m - (\Upsilon\mathcal{L}) \otimes F_m]$$

Using $c_1 = 2$ and $c_2 = 3$, defining $k_i = 1, i = 1...N$ and the methods described in Section 7, Ω is calculated with the defined parameters.

$$\Omega = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -2 & -3 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -2 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & -3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 & -2 & -3 \end{bmatrix}$$

Considering the system described by

$$\frac{d}{dt}x = \Omega x$$

The eigenvalues of Ω:

$$eig(\Omega) = \begin{bmatrix} -2.7963 + 0.0000i \\ -0.1018 + 1.1917i \\ -0.1018 - 1.1917i \\ -0.0000 + 0.0000i \\ -2.6717 + 0.0000i \\ -2.3247 + 0.0000i \\ -2.0000 + 0.0000i \\ -0.1642 + 1.0469i \\ -0.1642 - 1.0469i \\ -0.3376 + 0.5623i \\ -0.3376 - 0.5623i \\ -1.0000 + 0.0000i \end{bmatrix}$$

Therefore, the system is stable, as all eigenvalues lie in the OLHP. This can be shown by plotting the eigenvalues on a graph.
`matlab`: `plot(eig(`Ω`),'*')` yields:

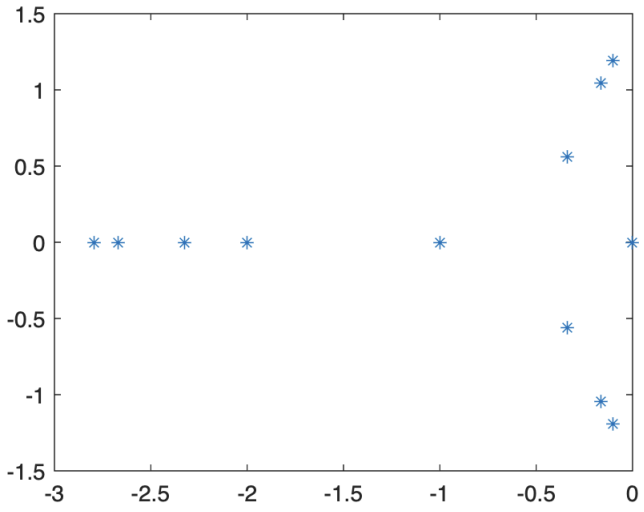


Figure 11: Eigenvalues for Ω

The Gershgorin disks for the matrix Ω calculated via `matlab` using [13] also show the eigenvalues in the OLHP.

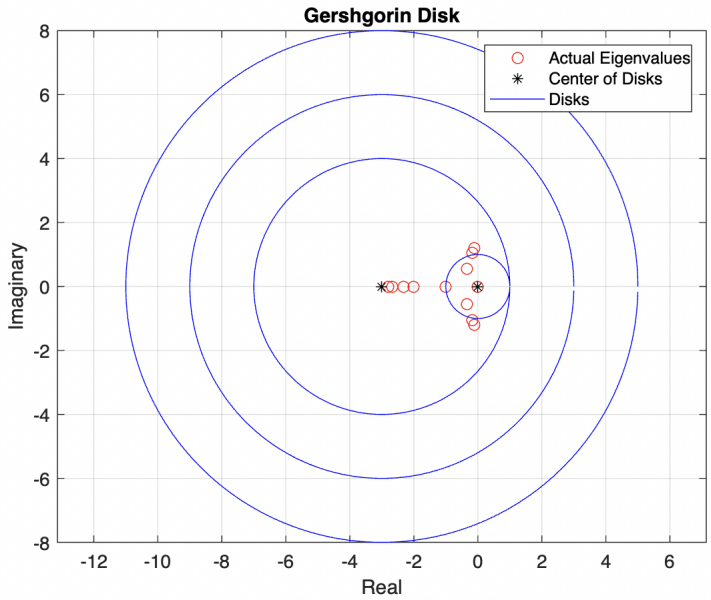


Figure 12: Gershgorin circles for Ω B

The free response of the system $\frac{d}{dt}x = \Omega x$ is

$$x(t) = e^{\Omega}t * x(0)$$

with

$$e^{\Omega}t = V * \begin{bmatrix} e^{\lambda_1 t} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{\lambda_n t} \end{bmatrix} * V^{-}1$$

and so,

$$x(t) = V * \begin{bmatrix} e^{\lambda_1 t} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{\lambda_n t} \end{bmatrix} * V^{-}1 * \bar{x}$$

All the eigenvalues of Ω are either negative real or negative complex numbers, and so the exponents in the matrix are also negative real/complex. The exponent can be rewritten as $e^{-xt \pm iyt}$. In all cases there is a '$-xt$' in the exponent, and so as $t \to \infty$, $e^{-t*x} \to 0$ and therefore, $e^{-xt \pm iyt} \to 0$.

8.1 MATLAB

8.1.1 Representation

Adjusting the values of c_k, e.g. doubling them, would increase the rate at which the plots of evolution reach a consensus value. Therefore, from hereon the parameters c_1 and c_2 will be adjusted to 4 and 6 respectively. The calculations are computed as previously,

$$E_m = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -4 & -6 \end{bmatrix}$$

And as a result,

$$\Omega = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -4 & -6 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -4 & -6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -2 & -4 & -6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 & -4 & -6 \end{bmatrix}$$

With the new eigenvalues,

$$eig(\Omega) = \begin{bmatrix} -0.3020 + 0.8063i \\ -0.3020 - 0.8063i \\ -0.0000 + 0.0000i \\ -5.3961 + 0.0000i \\ -5.3579 + 0.0000i \\ -5.2361 + 0.0000i \\ -5.2780 + 0.0000i \\ -0.7639 + 0.0000i \\ -0.3210 + 0.6759i \\ -0.3210 - 0.6759i \\ -0.3610 + 0.2432i \\ -0.3610 - 0.2432i \end{bmatrix}$$

Therefore, the transfer function of Ω:

$$s^{12}+24s^{11}+232s^{10}+1160s^{9}+3264s^{8}+5568s^{7}+6611s^{6}+5604s^{5}+3396s^{4}+1436s^{3}+376s^{2}+48s$$

Investigating this, it can be discovered that the further left eigenvalues of Ω have been pushed even further to the OLHP.

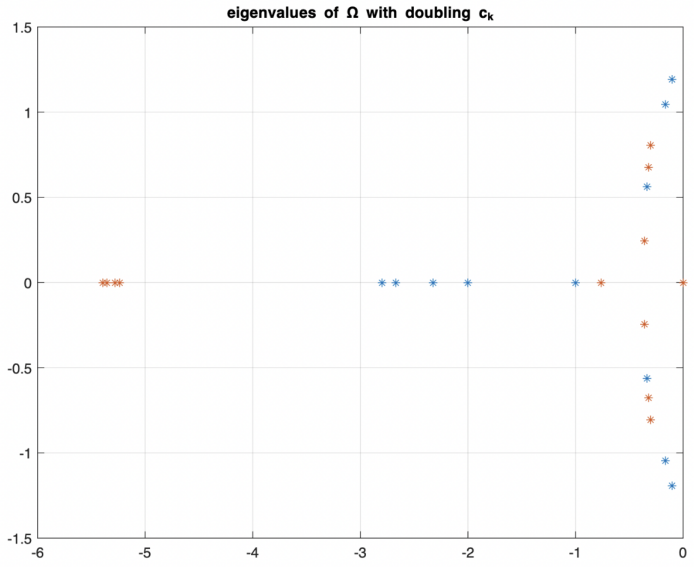
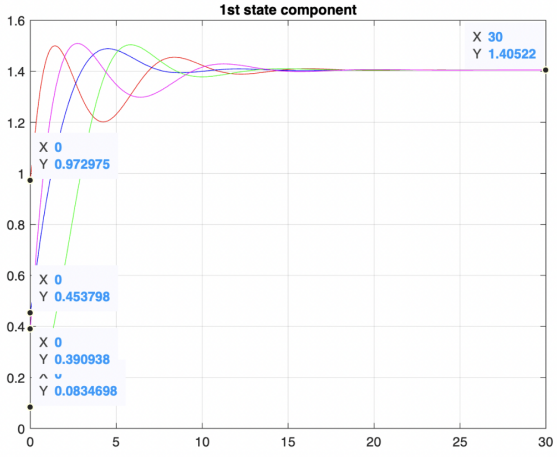


Figure 13: Eigenvalues of Ω with $c_1 = 2, c_2 = 3$(Blue), Eigenvalues of Ω with $c_1 = 4, c_2 = 6$(Orange)

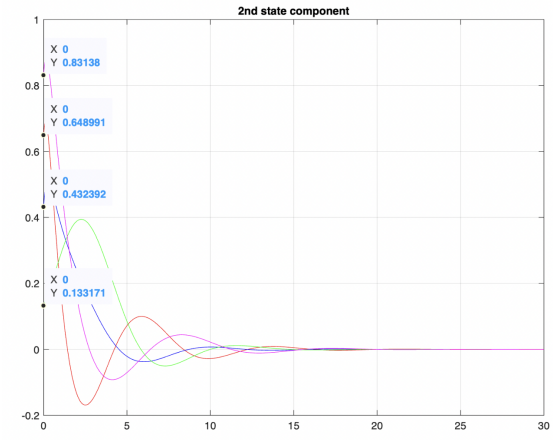
Ergo, the feedback gains (of absolute information), also known as the pole placement gains, not only is used to stabilise the system, but increases the rate at which the system of networks reaches consensus.

8.1.2 Results

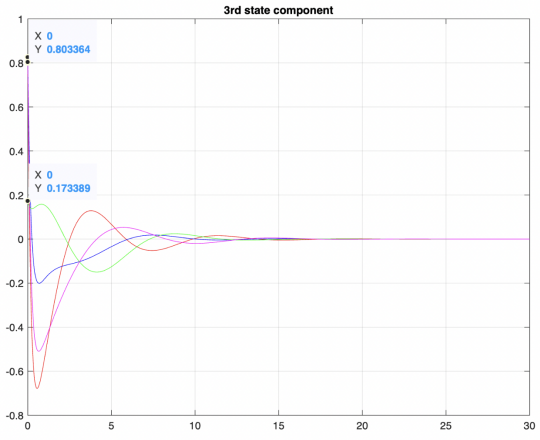
Using the functions discussed at the end of the background theory sections and the calculations from above, the MATLAB code to represent the graph network has been designed and implemented. The code B implements the whole network under the consensus protocol discussed. The initial conditions of the agents are set to be random values, with the first, second, and third state components of all agents being plotted on separate graphs.



(a) 1st state component



(b) 2nd state component



(c) 3rd state component

Figure 14: Graphs produced via MATLAB code B

The graphs of the evolution of the state variables x_k of all agents, k = 1, 2, 3 converge to a value. The relevant consensus value being the x_1 component, which converges to a value of 1.40522. The other state components (x_2, x_3) tend to zero, and hence will not be discussed.

The initial conditions of the initial states are visible on Figure 14, then recorded and used to verify the consensus state values of each agent as discussed in Section 7.3 using MATLAB.

```
%values of state component of each agent from figure 13
T1=[0.972975; 0.648991; 0.800331];
T2=[0.453798; 0.432393; 0.825314];
T3=[0.390938; 0.83138; 0.803364];
T4=[0.0834698; 0.133171; 0.173389];
%wl^T eigenvector
w=[-0.5 -0.5 -0.5 -0.5];
%~c
c=[4 6 1];
%equilibria of state system
e= kron(0.25*w,c);
c1=4;
%consensus state
X= 1/c1 * ((-0.5 * c * T1) + (-0.5 * c * T2) +  (-0.5 * c *
    T3) + (-0.5 * c * T4));
X * e
```

Figure 15: Matlab code implementing the consensus state $\chi(\overline{\xi}_0)e_1$ of Figure 14

Which yields the results:

$$\begin{bmatrix} 1.4052 & \cdots & \cdots & 1.4052 & \cdots & \cdots & 1.4052 & \cdots & \cdots & 1.4052 & \cdots & \cdots \end{bmatrix}$$

It can be seen that the outputs of the first state correlates with the evolution of the state variable x_1 as it is calculated as 1.4052. This also confirms the success of the implementation of the MATLAB code as it coincides with the theorems in Section 7 from Jiang et al. [1]

8.2 Dynamics

The closed-loop dynamics for the agents are:

Node 1:

$$\begin{aligned} \frac{d}{dt}\overline{\xi}_1 &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -c_1 & -c_2 \end{bmatrix} \overline{\xi}_1 - k_1 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \Big((\overline{\xi}_1 - \overline{\xi}_2) + (\overline{\xi}_1 - \overline{\xi}_3) + (\overline{\xi}_1 - \overline{\xi}_4) \Big) \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -c_1 & -c_2 \end{bmatrix} \overline{\xi}_1 - k_1 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \Big(3\overline{\xi}_1 - \overline{\xi}_2 - \overline{\xi}_3 - \overline{\xi}_4 \Big) \end{aligned}$$

Node 2:

$$\frac{d}{dt}\overline{\xi}_2 = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -c_1 & -c_2\end{bmatrix}\overline{\xi}_2 - k_2\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}(\overline{\xi}_2 - \overline{\xi}_1)$$

Node 3:

$$\begin{aligned}\frac{d}{dt}\overline{\xi}_3 &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -c_1 & -c_2\end{bmatrix}\overline{\xi}_3 - k_3\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}((\overline{\xi}_3 - \overline{\xi}_1) + (\overline{\xi}_3 - \overline{\xi}_4))\\ &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -c_1 & -c_2\end{bmatrix}\overline{\xi}_3 - k_3\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}(2\overline{\xi}_3 - \overline{\xi}_1 - \overline{\xi}_4)\end{aligned}$$

Node 4:

$$\begin{aligned}\frac{d}{dt}\overline{\xi}_4 &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -c_1 & -c_2\end{bmatrix}\overline{\xi}_4 - k_4\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}((\overline{\xi}_4 - \overline{\xi}_1) + (\overline{\xi}_4 - \overline{\xi}_3))\\ &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -c_1 & -c_2\end{bmatrix}\overline{\xi}_4 - k_4\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}(2\overline{\xi}_4 - \overline{\xi}_1 - \overline{\xi}_3)\end{aligned}$$

Alternatively, as mentioned in Section 7.1 the dynamics can be computing $\Omega * \overline{\xi}_i$ and substituting the defined parameters $c_1 = 4$, $c_2 = 6$ and $k_i = 1, i = 1...N$

$$\Omega * \overline{\xi}_i = \begin{bmatrix}\dot{\xi}_1\\ \ddot{\xi}_1\\ -3k_1\xi_1 - c_1\dot{\xi}_1 - c_2\ddot{\xi}_1 + k_1\xi_2 + k_1\xi_3 + k_1\xi_4\\ \dot{\xi}_2\\ \ddot{\xi}_2\\ k_2\xi_1 - k_2\xi_2 - c_1\dot{\xi}_2 - c_2\ddot{\xi}_2\\ \dot{\xi}_3\\ \ddot{\xi}_3\\ k_3\xi_1 - 2k_3\xi_3 - c_1\dot{\xi}_3 - c_2\ddot{\xi}_3 + k_3\xi_4\\ \dot{\xi}_4\\ \ddot{\xi}_4\\ k_4\xi_1 + k_4\xi_3 - 2k_4\xi_4 - c_1\dot{\xi}_4 - c_2\ddot{\xi}_4\end{bmatrix} = \begin{bmatrix}\dot{\xi}_1\\ \ddot{\xi}_1\\ -3\xi_1 - 4\dot{\xi}_1 - 6\ddot{\xi}_1 + \xi_2 + \xi_3 + \xi_4\\ \dot{\xi}_2\\ \ddot{\xi}_2\\ \xi_1 - \xi_2 - 4\dot{\xi}_2 - 6\ddot{\xi}_2\\ \dot{\xi}_3\\ \ddot{\xi}_3\\ \xi_1 - 2\xi_3 - 4\dot{\xi}_3 - 6\ddot{\xi}_3 + \xi_4\\ \dot{\xi}_4\\ \ddot{\xi}_4\\ \xi_1 + \xi_3 - 2\xi_4 - 4\dot{\xi}_4 - 6\ddot{\xi}_4\end{bmatrix}$$

.

$$\begin{aligned}\therefore u_1 &= -3\xi_1 - 4\dot{\xi}_1 - 6\ddot{\xi}_1 + \xi_2 + \xi_3 + \xi_4\\ u_2 &= \xi_1 - \xi_2 - 4\dot{\xi}_2 - 6\ddot{\xi}_2\\ u_3 &= \xi_1 - 2\xi_3 - 4\dot{\xi}_3 - 6\ddot{\xi}_3 + \xi_4\\ u_4 &= \xi_1 + \xi_3 - 2\xi_4 - 4\dot{\xi}_4 - 6\ddot{\xi}_4\end{aligned}$$

8.3 Simulink

8.3.1 Modelling

Using closed-loop dynamics for each agent (as discussed in Section 8.2), a small programme has been created to simulate each agent under the same protocol synchronously. The programme models the exemplary connection of agents that Section 7 explores.

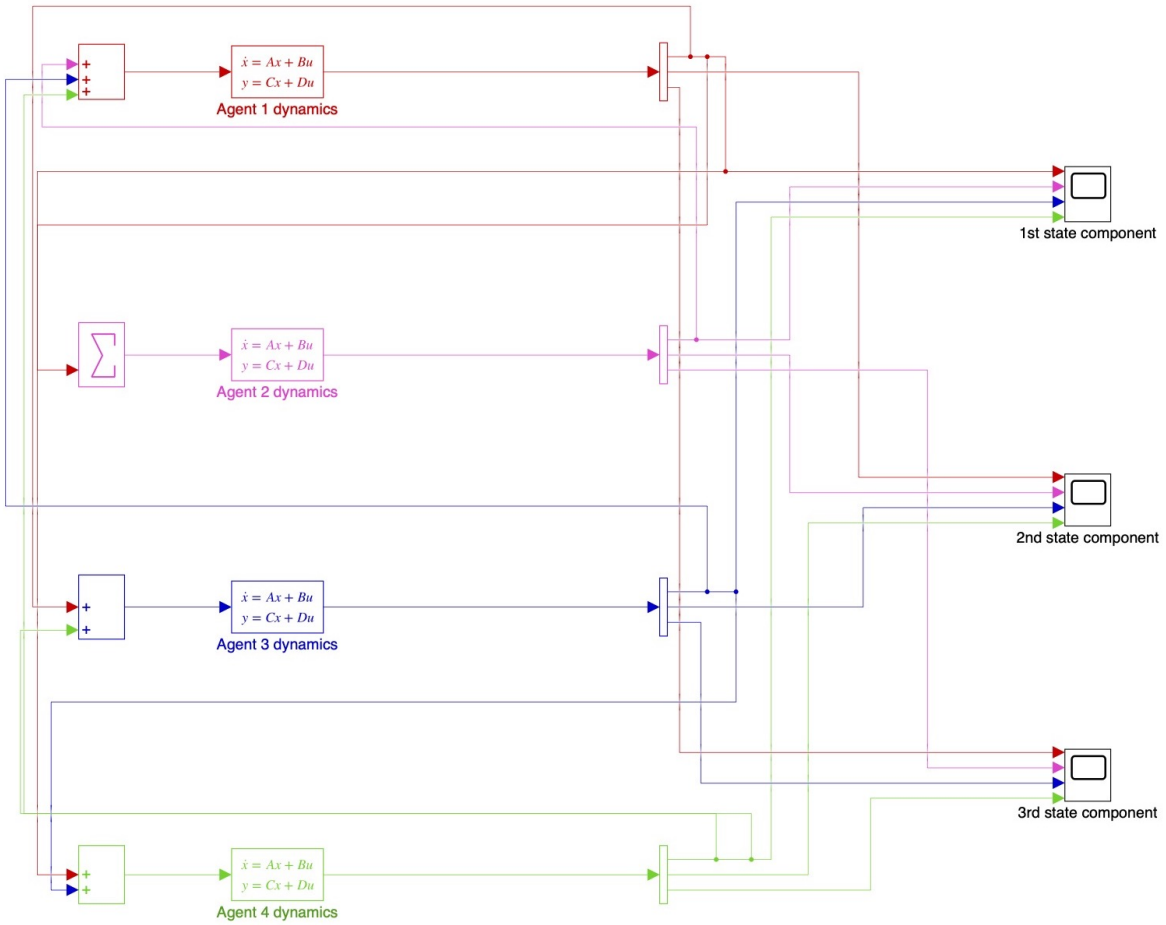
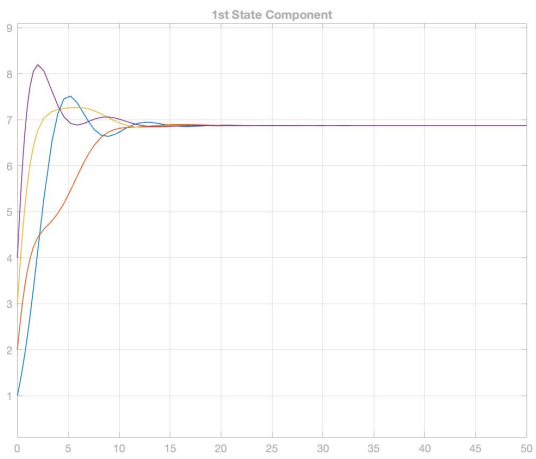


Figure 16: Simulink block diagram model for 4 agent third-order integrator system as described in Section 8

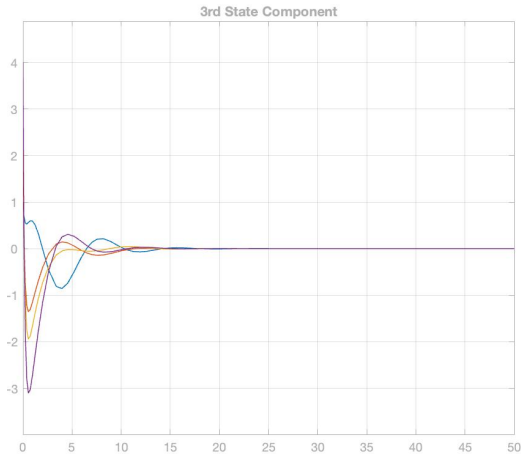
8.3.2 Results

The simulated model verifies the theoretical results for the standard assumptions. Each state component for all four agents reaches a consensus, as demonstrated in the MATLAB results. The period of the scope is extended from the default 10.0 s to 50.0 s, although consensus is reached around ≈20s. The time at which consensus is reached, i.e. the convergence rate, appears to be the ≈20s for both sets of graphs produced via the MATLAB implementation and via Simulink modelling. To conclude, Figure 16 successfully models the four-node third-order integrator system discussed and supports the MATLAB implementation.

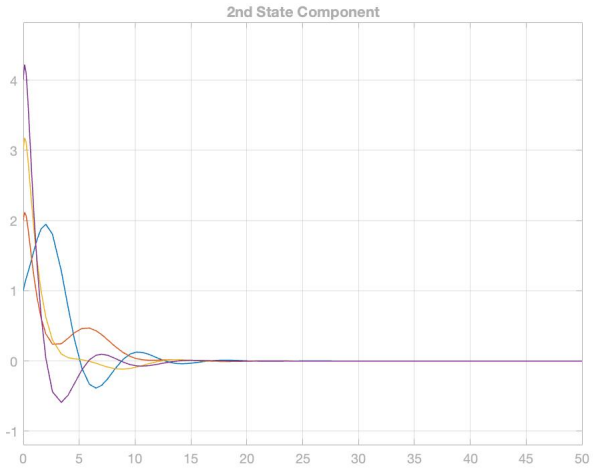
The model produces the following results:



(a) 1st state component (Scope 1)



(c) 3rd state component (Scope 3)



(b) 2nd state component (Scope 2)

Figure 17: Graphs produced from Scopes via Simulink model 16

Focusing on Figure 17a (for the same reason as mentioned in 8.1.2), the scoped graph is investigated via measurements feature (ruler icon) and the x-y components are explored as shown in Figure 18.

The consensus value was found at a time ≈ 26.3s. and holds the value of 6.875. This is approximate due to the scaling of the measurement feature, requiring one to manually scale and find the point at which the consensus value is reached. The method used was by probing a cursor to the far right and reading the correlating value and, in turn, the other cursor being manually moved and stopped (precisely) once the measurements matched.

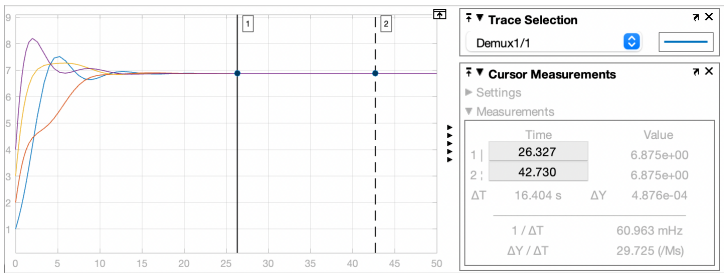


Figure 18: 1st state component (Scope 1) measurements

For the simulink model 16 the initial conditions were chosen at random for each agent's dynamics and inserted in the block parameter for each agent. To verify if the simulation is successful, Figure 15 is adjusted by inserting the inputs chosen.

```
%values of state component of each agent for figure 16 (
   chosen at random)
T1=[1; 1; 1];
T2=[2; 2; 2];
T3=[3; 3; 3];
T4=[4; 4; 4];
%wl^T eigenvector
w=[-0.5 -0.5 -0.5 -0.5];
%~c
c=[4 6 1];
%equilibria of state system
e= kron(0.25*w,c);
c1=4;
%consensus state
X= 1/c1 * ((-0.5 * c * T1) + (-0.5 * c * T2) +  (-0.5 * c *
    T3) + (-0.5 * c * T4));
X * e
```

Figure 19: Matlab code implementing the consensus state $\chi(\overline{\xi}_0)e_1$ of Figure 1617

Which yields the results:

$$\begin{bmatrix} 6.8750 & \cdots & \cdots & 6.8750 & \cdots & \cdots & 6.8750 & \cdots & \cdots & 6.8750 & \cdots & \cdots \end{bmatrix}$$

Indeed confirming the simulation is a success as the theoretical value of the consensus state (computed via matlab code) is equivalent to the simulation scope output measurements.

8.4 Scalability

Scalability is the measure of how well a system (e.g. computers, networks, algorithms) responds to changes by adding or removing resources ([16]). Scaling a network sys-

tem (by adding or removing nodes) does not prevent consensus. As the network size increases, the convergence rate of the consensus algorithm also increases. Thus, the convergence rate is proportional to the size of the network. Appendix C includes the effect of scalability on consensus for a system with $N < 4$ agents (C.1), $N > 4$ agents (C.2) and $N >> 4$ agents (C.3) to support this claim. [19] explores the effect of scalability in networks on consensus, although stressing that the limitations apply to leader-follower networks (not leaderless like the ones discussed in this project).

9 Extension: Synchronisation

The following sections will explore the addition of noise and sinusoidal input to the original 4-node agent system with third-order integrator dynamics shifting from steady-state consensus to synchronisation. It is beneficial to investigate systems with different inputs and additions (i.e. noise, malicious attacks), as exploring the generated output is practical due to the implementations it may have. This introduces possible changes and problems to the system (i.e. errors), which will then need solutions, and the system will have to be adjusted and further inspected. In conclusion, modifications will be made to the original 4-agent system using Simulink; the different inputs and what the system generates (whether this is desirable) will be studied.

9.1 Noise Input

Noise is an error/undesired random disturbance to the information signal, present in most systems. Due to its nature, the effect of noise on the four agent system discussed in section 8 will be explored. White noise is a useful theoretical approximation when the noise disturbance has a correlation time that is very small relative to the natural bandwidth of the system. External noise examples include electromagnetic interference, frequency interference, etc. The original Simulink Model 16 with the original four-agent dynamic system can be adjusted with the addition of 'Band-limited White Noise' (BLWN) to simulate noise that is present in dynamic systems, as shown in Figure 20.

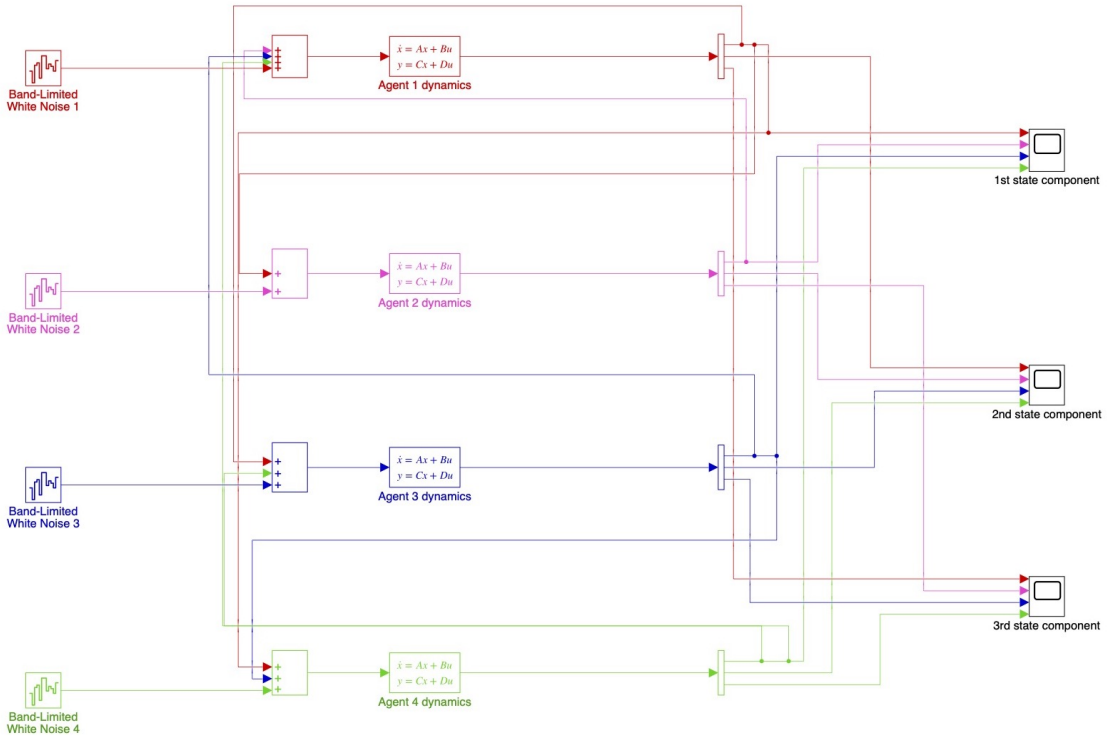
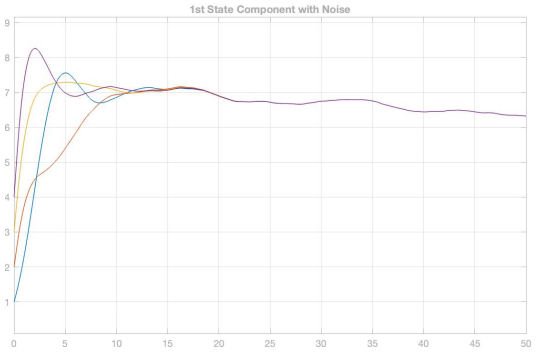


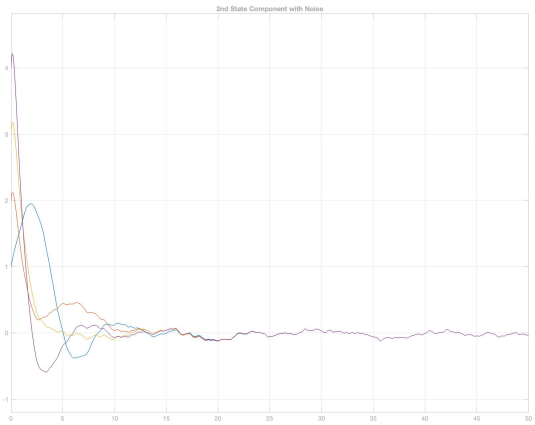
Figure 20: Simulink block diagram model for 4 agent third-order integrator system with the addition of Noise

BLWN is the only noise input available on Simulink; the block generates normally distributed random numbers suitable for continuous systems. The bandwidth of white noise is limited in practice due to finite observation capabilities. If other noise inputs were available on Simulink, it would be interesting to see if different types of noise (i.e. pink noise) inputs generate different effects on the consensus of the system.

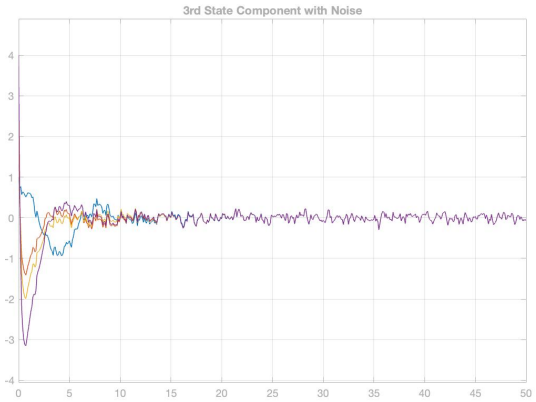
The simulated model in turn yields results as follows:



(a) 1st state component (Scope 1)



(b) 2nd state component (Scope 2)



(c) 3rd state component (Scope 3)

Figure 21: Graphs produced from Scopes via Simulink model with Noise 20

The results here are similar to that of the original Simulink model for the 4-node network, as the agents reach a consensus also at approx 20s. However, it is clear that the consensus state does not remain constant. The reason is that the noise input generates a random number, hence the generated output being (the same) random number. Thus, the consensus value becomes the varying input and demonstrates synchronisation.

9.2 Sinusoidal Input

Using a sinusoidal signal as an input signal to an LTI system has an output of a sine wave of the same period/frequency due to the property of linearity. Linearity obeys the superposition principle, meaning the output is proportional to the input. A linear dynamical system with input $u(t)$ has output $y(t)$.
For the sinusoidal input:

$$x(t) = cos(\omega t)$$

There is an output:

$$y(t) = Acos(\omega t + \phi) = acos(\omega t + \phi) + bsin(\omega t + \phi)$$

With amplitude A, phase ϕ.
The phase ϕ represents a time shift due to impedance which is a consequence of passing a sine wave.

The original Simulink Model 16 can be adjusted with the addition of 'Sine Wave' to simulate a sinusoidal input that is present in dynamic systems, as shown in Figure 22. The original four-agent geometry remains as the dynamic of the MAS, with the only alteration being the addition of the sinusoidal input.

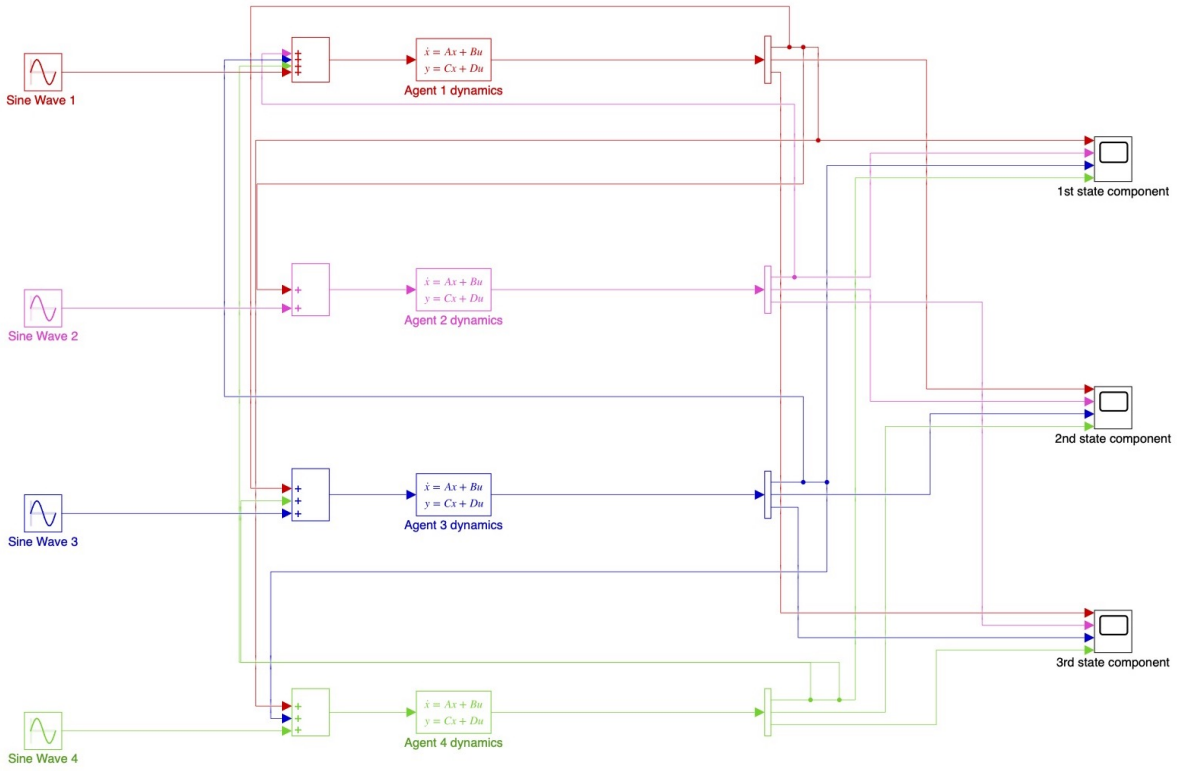
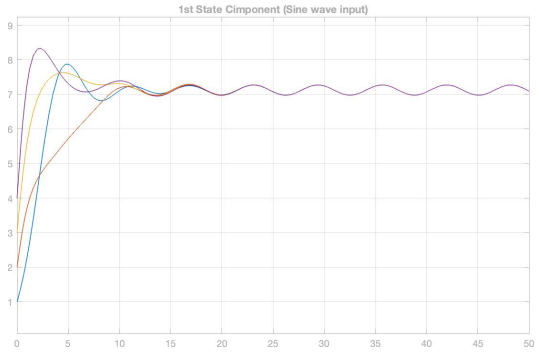
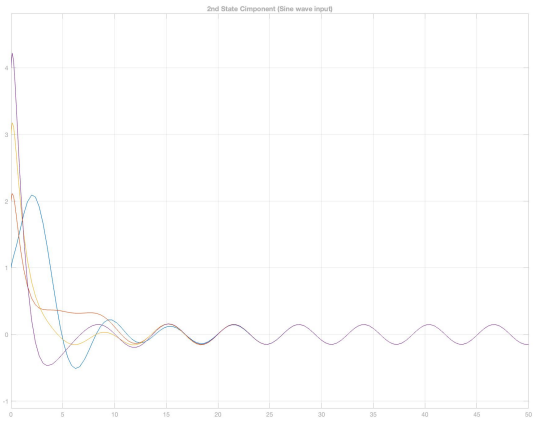


Figure 22: Simulink block diagram model for 4 agent third-order integrator system with the addition of Sinusoidal inputs

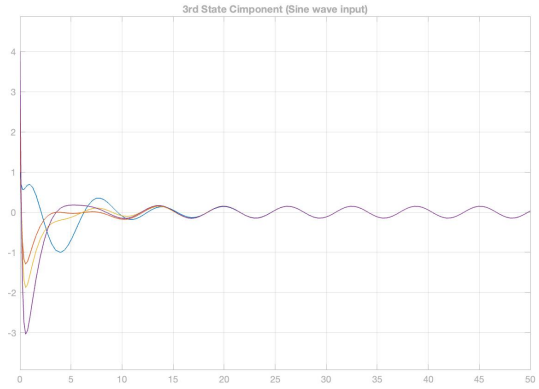
Yielding the corresponding graphs:



(a) 1st state component (Scope 1)



(b) 2nd state component (Scope 2)



(c) 3rd state component (Scope 3)

Figure 23: Graphs produced from Scopes via Simulink model with Sine wave input 22

As demonstrated by the original unaltered model, and with the addition of noise, the agents come to a general agreement with the addition of a sinusoidal input also. Although, as mentioned for Figures 21, the consensus value is not constant, and the agents reach a consensus with a non-constant input, i.e., synchronisation. [14].

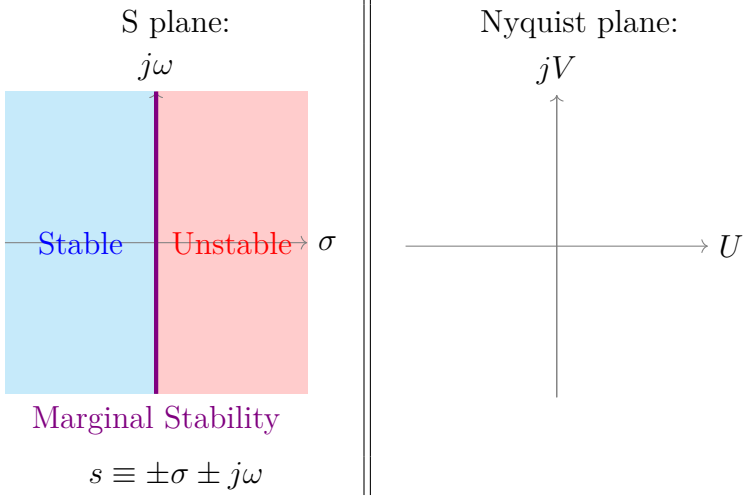
It is interesting to observe that the time at which consensus is achieved appears to be similar for the three cases simulated. If, upon further investigation, it is found to be the same time to reach consensus, researching a theoretical expression for the consensus rate would be interesting.

To conclude, synchronisation is when systems operate with all parts in synchrony, resulting in an overall system operating in unison. Investigating consensus and synchronisation is beneficial due to the ever-increasing importance of the problem in practise, e.g., based navigation systems to navigate a terrain of obstacles, large-scale wireless sensor networks.

10 Stability Analysis

For stability of a closed loop system, all the poles of the characteristic equation should lie on the left half of the S plane. There are different methods through which stability can be verified and the respective plane used for each domain:

Time Domain Methods	Frequency Domain Methods
Routh-Hurwitz Criterion	Nyquist Criterion
Root Locus Analysis	Bode Plot Analysis



10.1 Routh–Hurwitz Stability Criterion

Section 6.5 introduced the Routh-Hurwitz stability criterion and the methodology, as well as stating that stability is guaranteed if and only if the first column of the array contains elements that share the same sign. The diagram above simply demonstrates the characterisation of systems, which can be elaborated on as:

- Stable systems: have all poles in the OLHP $\Leftrightarrow$ all elements of the first column are positive
- Unstable systems: have at least one pole in the ORHP $\Leftrightarrow$ if any of the elements in the first column are negative
- Marginally stable systems: have poles on the imaginary axis. (i.e., would have sinusoidal oscillations, with a constant amplitude) $\Leftrightarrow$ if all the elements of any row of the Routh array are zero

The purpose of creating Routh array tables for high-order integrator systems is to show that a range of values for the c_k and μ parameters are necessary in order for the system to be stable, and as a result, reach consensus state. Outside of this range, the system would not be stabilisable, and the components would diverge, i.e. the system would explode ([12]). Using the method discussed in Section 6.5 Routh array tables are computed for the parameter conditions of linear systems with integrator order agents.

Second order system: m=3, $g_{2i}(s) = s^2 + c_1 s + \mu_i$

$s^2 + c_1 s + \mu_1$		
s^2	1	$\mu_1 = 0$
s^1	c_1	0
s^0	0	

(a) i=1

$s^2 + c_1 s + \mu_2$		
s^2	1	μ_2
s^1	c_1	0
s^0	$c_1\mu_2$	

(b) i=2

Table 1: Routh array tables for second order systems

Using the requirements for system stability, a range of values can be constructed from the tables. It can be seen that the first column is positive and non-zero for $c_1 > 0$ and $c_1\mu_2 > 0$ (which implies that $\mu_2 > 0$ for this to be true).

Third order system: m=4 , $g_{3i}(s) = s^3 + c_2 s^2 + c_1 s + \mu_i$

$s^3 + c_2 s^2 + c_1 s + \mu_1$		
s^3	1	c_1
s^2	c_2	$\mu_1 = 0$
s^1	$\frac{c_2 c_1}{c_2}$	
s^0	0	

(a) i=1

$s^3 + c_2 s^2 + c_1 s + \mu_2$		
s^3	1	c_1
s^2	c_2	μ_2
s^1	$\frac{c_2 c_1 - \mu_2}{c_2}$	
s^0	μ_2	

(b) i=2

Table 2: Routh array tables for third order systems

And so, from 2, it can be seen $c_2 > 0$, $c_1 > 0$ and $\mu_2 > 0$, $c_2 c_1 > \mu_2$. Therefore, it can be concluded that for the third order system to be Hurwitz stable, $\mu_2 > 0$, $c_1 > 0$, $c_2 > 0$ and $c_2 c_1 > \mu_2$.

Fourth order system: m=5, $g_{4i}(s) = s^4 + c_3 s^3 + c_2 s^2 + c_1 s + \mu_i$

$s^4 + c_3 s^3 + c_2 s^2 + c_1 s + \mu_1$			
s^4	1	c_2	$\mu_1 = 0$
s^3	c_3	c_1	0
s^2	$\frac{c_2 c_3 - c_1}{c_3}$	0	0
s^1	$\frac{(c_3 c_2 - c_1)c_1}{c_3 c_2 - c_1}$	0	0
s^0	0	0	0

(a) i=1

$s^4 + c_3 s^3 + c_2 s^2 + c_1 s + \mu_2$			
s^4	1	c_2	μ_2
s^3	c_3	c_1	0
s^2	$\frac{c_2 c_3 - c_1}{c_3}$	$\frac{c_3 \mu_2}{c_3} = \mu_2$	0
s^1	$\frac{(c_3 c_2 - c_1)c_1 - c_3^2 \mu_2}{c_3 c_2 - c_1}$	0	0
s^0	μ_2	0	0

(b) i=2

Table 3: Routh array tables for fourth order systems

Similarly, conditions of the parameters for system stability can be deduced for fourth-order systems. $c_3 > 0$, $c_2 c_3 > c_1$, $\frac{(c_3 c_2 - c_1)c_1}{c_3^2} > \mu_2$ with $\mu_2 > 0$ as before.

To summarise, using the methods discussed in the relevant sections, characteristic polynomials have been used to produce Routh array tables to propose parameters for the realisation of consensus.

10.2 Root Locus

Root Locus analysis is a graphical method for examining how the roots of a system change with variation of a certain system parameter, commonly a gain within a feedback system. It gives both absolute stability (whether a system is stable or unstable) and relative stability information (the degree of stability or how close it is to instability). Considering the system in Section 8 as a unity feedback control scheme: a closed-loop system with unity feedback and gain μ, with a plant G(s).

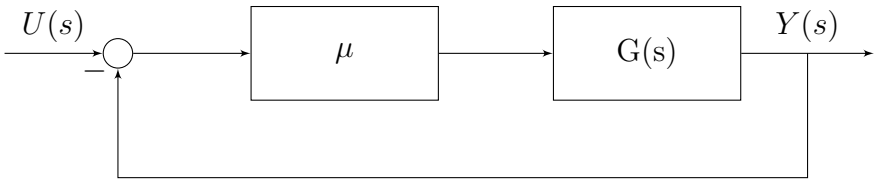


Figure 24: Unity feedback control scheme

Determining the performance of the system requires knowledge of the position of the closed-loop poles in the s-plane: the positions of the poles/eigenvalues of the closed-loop transfer function in the s-plane.
Let,

$$G(s) = \frac{1}{s^m + c_{m-1}s^{m-1} + \cdots + c_1 s}$$

Recalling the characteristic polynomial,

$$P_\Omega(s) = (s^3 + c_2 s^2 + c_1 s + \mu_1) * (s^3 + c_2 s^2 + c_1 s + \mu_2) * \cdots * (s^3 + c_2 s^2 + c_1 s + \mu_N)$$

The closed loop transfer function of Figure 24 is:

$$T(s) = \frac{\mu G(s)}{1 + \mu G(s)}$$

The characteristic equation:

$$\begin{aligned} 1 + \mu G(s) &= 1 + \frac{\mu}{s^m + c_{m-1}s^{m-1} + \cdots + c_1 s} \\ &= s^m + c_{m-1}s^{m-1} + \cdots + c_1 s + \mu_i =: g_{mi}(s) \end{aligned}$$

$0 \leq \mu_i < \infty$

For small gain μ ($\approx 0^+$), the closed loop poles (CLP) are at the same location as the open loop poles (OLP). For the plant G(s) rank n=0, as μ is increased, the CLP

move towards the open loop zeroes (OLZ). G(s) with rank n>1, n number of CLP will move towards ∞.

`matlab: sys = tf ([numerator],[denominator]);` creates a system with defined TF.
`rlocus(sys)` plots the root locus of said system.

Imaginary Axis Intercept: Routh-Hurwitz tables are used to calculate the gain value that causes marginal stability: μ_{margin}. The roots of the characteristic equation $1 + \mu G(s)$ once μ_{margin} has been substituted gives the imaginary axis crossings.

As the four-node system investigated is a third-order linear system, calculations and analysis will focus on the systems of third degree. The theory and calculations can be applied to the other m^{th}order systems previously mentioned.

Third order system:

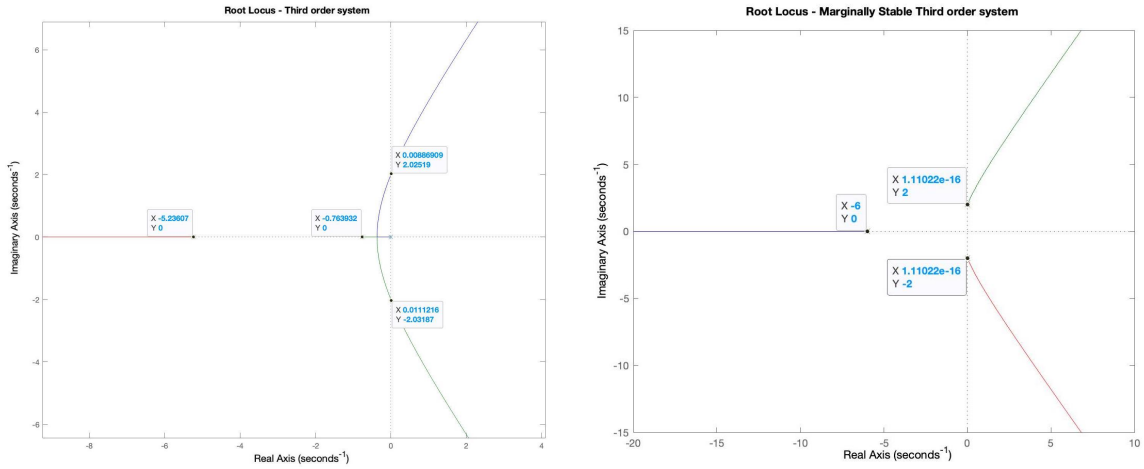
$$g_{3i}(s) = s^3 + c_2 s^2 + c_1 s + \mu_i$$

Recalling from 2, the range for gain is $0 < \mu < c_2 c_1$.
Maximum $\mu = \mu_{margin} = c_2 c_1 \Rightarrow g_{3i}(s) = s^3 + c_2 s^2 + c_1 s + c_2 c_1$
Yielding the roots/poles:

$$\underbrace{s = -c_2}_{\text{real axis intercept}}$$

$$\underbrace{s = +\sqrt{c_1}j, s = -\sqrt{c_1}j}_{\text{imaginary axis intersection points on Root Locus}}$$

Using MATLAB, the root locus graph for the third-order integrator system with defined parameters $c_1 = 4$ and $c_2 = 6$ yields:



(a) Stable system with $\mu = 0$ (b) Marginally stable system with $\mu = c_2 c_1$

Figure 25: Root Locus for Third order system with $\mu = 0$

The results demonstrated by Figure 25 show the imaginary axis intercepts indeed being $\approx 2j = \sqrt{c_1}j$ and therefore verifying the theoretical analysis. Figure 25a also has poles at $s = -3 \pm \sqrt{5}$, which verifies the calculations of the characteristic equation $g_{3i} = 0$, solved for s. Therefore, it can be concluded that μ must not exceed the limit c_2c_1, and for this system with the defined parameters: $\mu < 24$ or the system will be unstable.

10.3 Nyquist Criterion

An open-loop control system is a system where the control input is applied without influence of output or "state" of the system, in other words, a closed-loop system without state feedback.

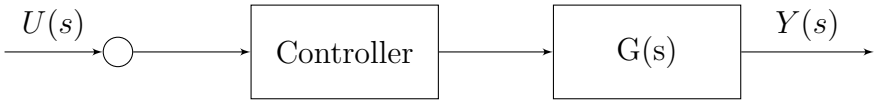
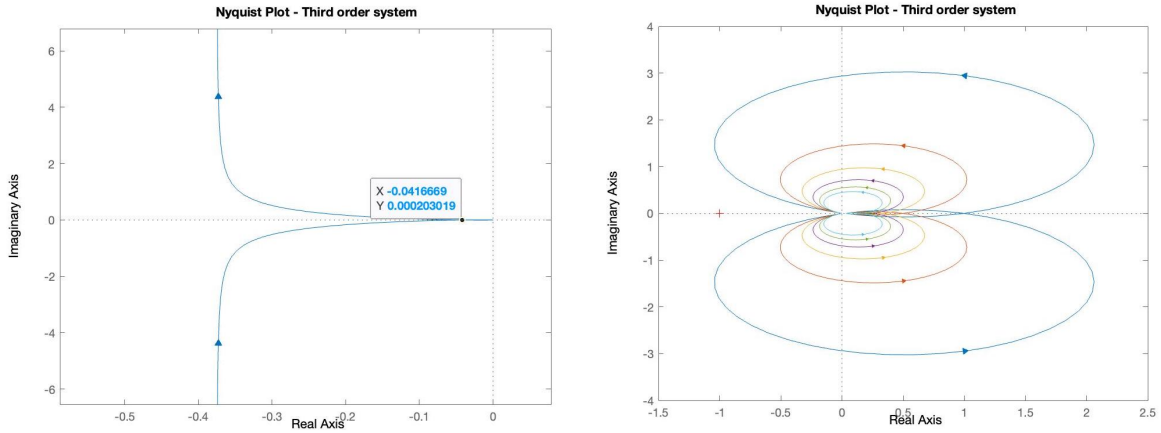


Figure 26: Open-loop control scheme

A substitution of $s = j\omega$ must be made in order to convert from the s-plane to the Nyquist plane of $jU + V$ axes. Converting the characteristic equation and separating the variables also can be used to calculate the intercepts on the nyquist plot.

$$g_{3i}(s) = s^3 + c_2s^2 + c_1s + c_2c_1 + \mu_i \xrightarrow{s=j\omega} g_{3i} = \underbrace{-6\omega^2 + \mu}_{\mathfrak{Re}} + \underbrace{j(-\omega^3 + 4\omega)}_{\mathfrak{Im}}$$

`matlab`: `nyquist(sys)` plots the nyquist plot of the system defined.



(a) Stable system with $\mu = 0$

(b) Unstable system with $\mu_{margin} > c_2c_1$

Figure 27: Nyquist Plots for Third order system

The x intercept of Figure 27a is that of $\frac{1}{GainMargin} = \frac{1}{c_1c_2}$. As shown on the figure, it is approximately $\frac{1}{24}$, therefore showing the plot has verified the calculations mentioned. Hypothetically approaching the situation as though the gain estimate is wrong,

a method is to enlarge the Nyquist plot up to the pole (-1). Figure 27b Shows multiple systems demonstrated, with the largest plot being so when the value of μ is just greater than c_1c_2. As $\mu \to \infty$, the plot gets smaller and encircles the origin.

`matlab: [Gm,Pm,Wcg,Wcp] = margin(sys)` calculates the Gain margin, Phase margin and the corresponding frequencies of the Bode response of a defined system.

This introduces the concept of robustness of a system. Robust analysis deals with the design of a model, considering the necessary assumptions. Robust stability and robust performance is defined as a systems ability to remain operational under disturbances, and is closely related to sensitivity analysis. Therefore, the loop gain $\mu G(s)$ must be high enough to guarantee robustness to parameter changes.

10.4 Extension: Time delay

Time delays do not affect the characteristics of a system but occur due to communication passing information later than expected. The presence of a time delay is beneficial to explore, as many processes involve transport delays or time lags. Root locus techniques cannot be directly used due to the form of the system not being rational-polynomial. MATLAB cannot demonstrate a system with time delay due to a lack of suitable programmes and this is further explained in 11.2 as it is a software limitation. As a result, the Padé approximations will be sufficient.

A system with time delay is of the form: [17]

$$G(s) = e^{-\tau_d s} G'(s)$$

with the delay τ_d, and non-delayed system $G'(s)$.

10.4.1 Padé approximation

Expanding the time delay via taylor series expansion yields:
First-order approximation:

$$e^{-\tau_d s} \approx \frac{1 - (\frac{\tau_d s}{2})}{1 + (\frac{\tau_d s}{2})}$$

Second-order approximation:

$$e^{-\tau_d s} \approx \frac{1 - (\frac{\tau_d s}{2}) + (\frac{(\tau_d s)^2}{12})}{1 + (\frac{\tau_d s}{2}) + (\frac{(\tau_d s)^2}{12})}$$

For small delays, a crude approximation:

$$e^{-\tau_d s} \approx \frac{1}{1 + \tau_d s}$$

10.4.2 Frequency-response design

Directly plotting root locus using phase condition, values of s for which the phase is $180° + 360°l$ must be found.
The phase of $G(s)$ is

$$\angle G(s) = \angle G'(s) - \tau_d \omega$$

for $s = \sigma + j\omega$.
Searching for locations where the phase of $G'(s)$ is

$$\angle G'(s) = 180° + \tau_d \omega + 360°l.$$

Fixing ω and finding a point on the locus, raising the value of ω, changing the target angle and repeating.

10.4.3 MATLAB Implementation

A MATLAB implementation of the Padé approximation and the corresponding root locus plot can be used to demonstrate the system with a time delay. The code below inputs the plant $G(s)$, inputting the Padé approximation with a $0.5s$ delay with 2 terms. Turning this into a transfer function called 'Delay', and using that to compute the system with the delay.

```
1  %defining the third order system with zero gain
2  sys = tf ([1],[1 6 4 0]);
3  G= sys;
4  [num,den]=pade(0.5,2);
5  Delay=tf(num,den);
6  k=logspace(-2,2,1000)';
7
8  %plot
9  rlocus(G*Delay,k)
10 title ('Pade Time-delay Root Locus - Third order system')
```

Figure 28: Matlab code implementing the Padé approximation time delay for the third order system

Yielding,

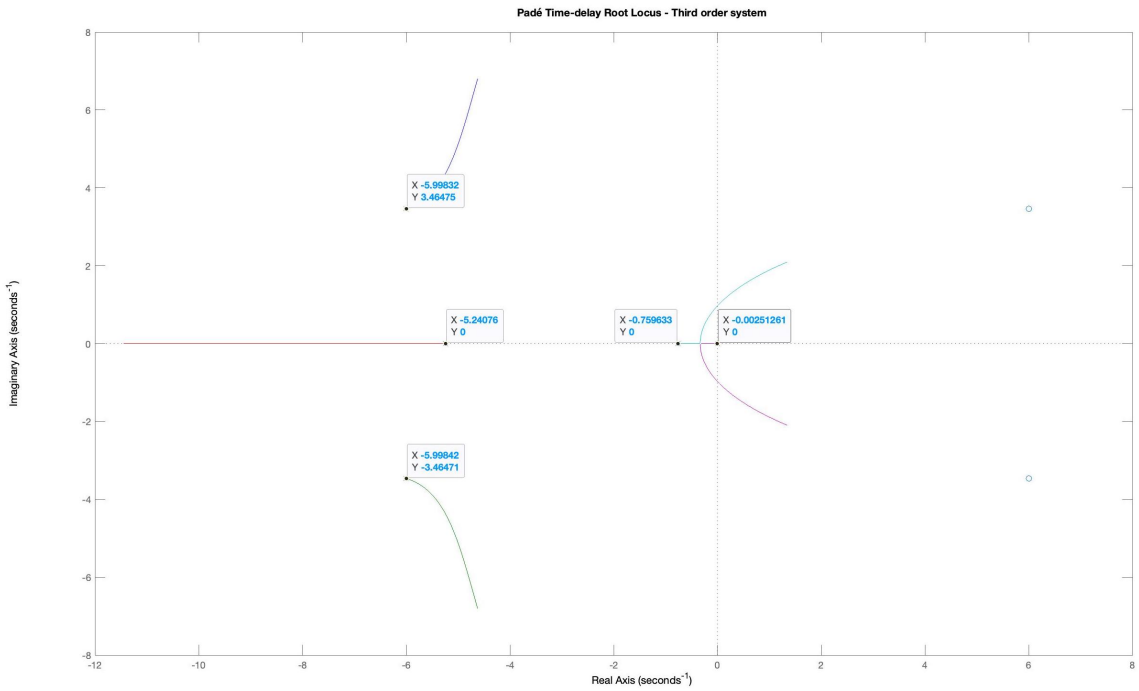


Figure 29: Root locus for Third order system with delay

The poles of the plant are $s = \{0, -3 + \sqrt{5}, -3 + \sqrt{5}\}$. The Padé approximation for delay at $\approx s = \{-6 + 3.46j, -6 - 3.46j\}$, the system with delay having two zeros at $\approx s = \{6 + 3.46j, 6 - 3.46j\}$. The two zeros are pulling the root locus to the right, destabilising the system as a result. The solutions for the system with time delay would be to find 'K' for X% overshoot, determining the crossover point to find the value(s) for s and the corresponding K. Investigating K is not necessary as the main objective for looking at time delay is to observe the effects it would have on a system, consequently whether or not the multi-agent system reaches consensus.

10.4.4 Simulink Modelling

Delays cause phase shifts, limit the control bandwidth, and affect closed-loop stability. The system with time delay is modelled by modifying the original Simulink model Figure 16 by using the 'Transport delay' block which delays the input by a given amount of time. This Simulink model is a continuation of the desired four-node graphical network, with each dynamic agent being represented by a colour and a delay applied to each state component of each agent.

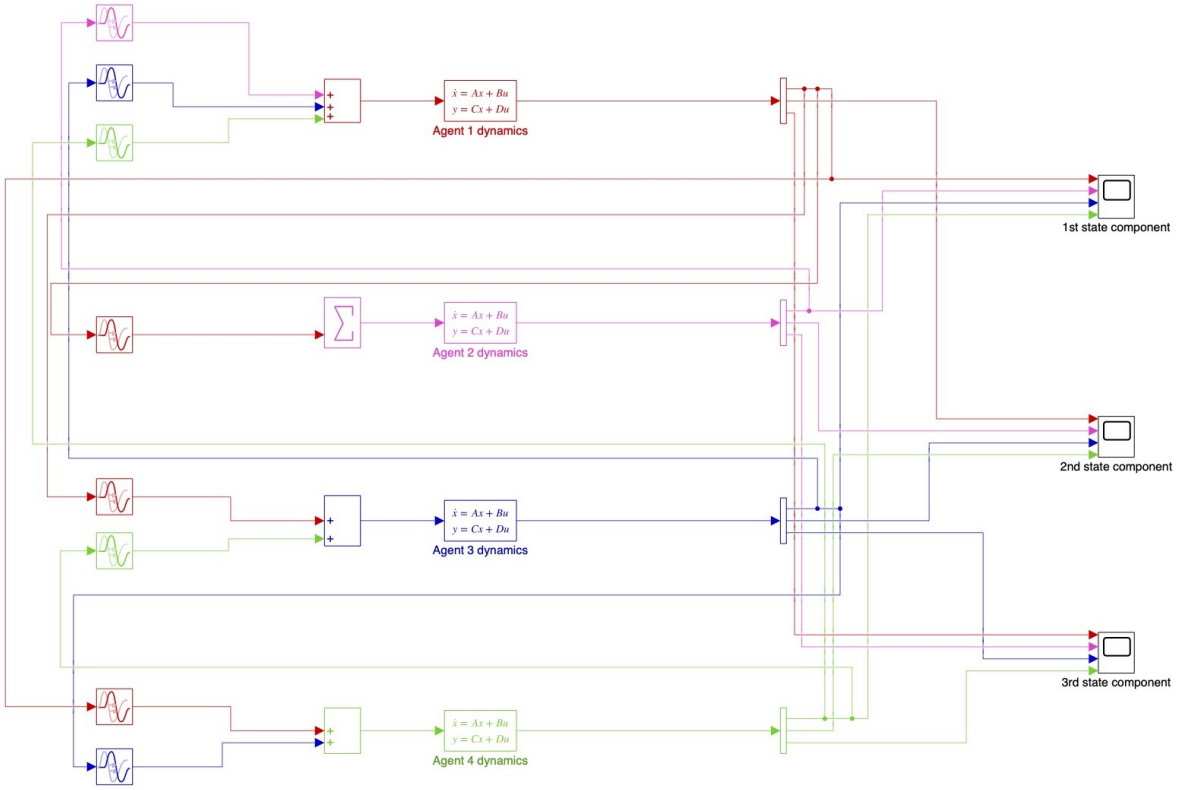
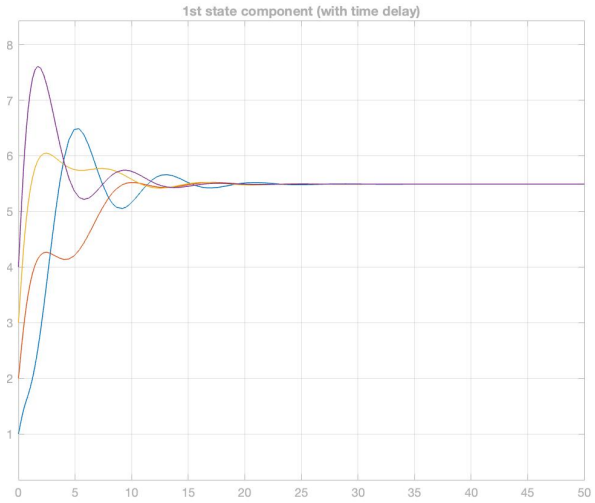
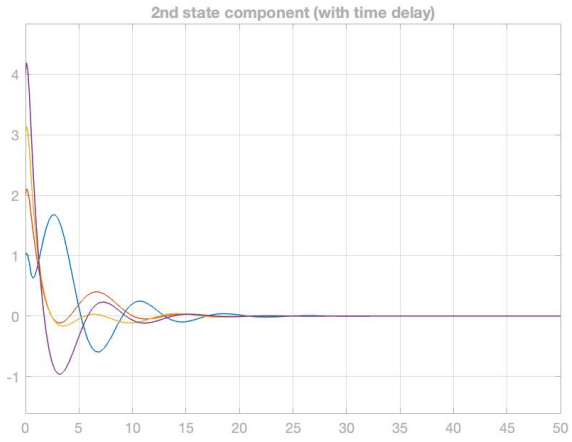


Figure 30: Simulink block diagram model for 4 agent third-order integrator system with the addition of a $0.5s$ time delay

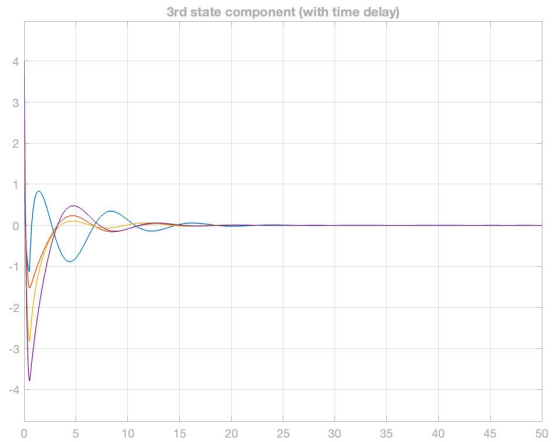
Yielding the corresponding graphs:



(a) 1st state component (Scope 1)



(b) 2nd state component (Scope 2)



(c) 3rd state component (Scope 3)

Figure 31: Graphs produced from Scopes via Simulink model with delayed input 30

The simulations show that with a time delay that affects the system inputs, the overall system succeeds in reaching consensus. Although, the difference between 31 and 17 is that there is a clear delay in which consensus is reached. Using the measurement method previously shown in Figure 18, the convergence rate is at $\approx 35s$. The 0.5s time delay has not caused the system to fail (although it tended to destabilise the system).

This poses new questions: Does a system have to be stable in order to reach consensus? What are the requirements / bounds with parameters in order for a system to (not) reach consensus? What about larger time delays (>0.5s)?

Due to delays in systems not often being known accurately, the concept of sensitivity is introduced. It is important to understand how sensitive a system is to a delay as the process characteristics may change, therefore the parameters must be chosen such that the CL system is insensitive to parameter variations in the plant.

The sensitivity function is the denominator of the closed loop transfer function described in Section 10.2:

$$T(s) = \underbrace{\frac{\mu G(s)}{1 + \mu G(s)}}_{\text{sensitivity function}}$$

11 General Design Procedure

The following chapter provides an overview and discusses the project learning process. An insight into the general design procedure of the project and justifications for considerations will be provided.

11.1 Design and Implementation

11.1.1 Desired Graph Network

The desired network being a four-node graph was a decision made due to a combination of reasons:

- Having too small of a graph network (e.g. 2 nodes) would make it difficult to see trends and possible extensions to the project. For example, in a 2 node graph network, both agents share the same dynamics. As a result, it would be a very simple network to investigate for a project.
- In a graph network with many nodes (e.g. 10), it would take a long time to compute the dynamics of each agent. It would be timely and inefficient to model because it is prone to errors as the matrices increase in dimensions proportionally to the network size.
- The concept of scalability means that the number of nodes in the graph network is not paramount to the consensus of the MAS. Therefore, the focus has moved from the size of the network to the importance of the topology of the system.

11.1.2 Consensus Algorithms

Consensus in Multi-Agent Systems has attracted a lot of attention and research, for its many applications. The consensus problem for LTI systems has been less investigated, motivating research for this paper.

A more sophisticated consensus algorithm such as H∞ was a viable choice. For example, an approach method is the H∞ consensus algorithm for a team of multi-agent systems, and solving an Algebraic Ricatti Equation; as mentioned in [21]. H∞ methods are used in control theory to achieve stabilisation and find controllers to solve a mathematical optimisation problem. The problem formulation is much more complex compared to the consensus algorithm studied. The general consensus algorithm discussed in this project is a good starting point for more sophisticated methods, allowing room for future work and expansions concerning consensus and control theory.

11.1.3 MATLAB Implementation

The calculations made in (8) were vital to implement in MATLAB. The method of approach was a direct one-to-one translation of theory/calculations into code (B):

- Lines 3-8: implements the matrices describing the desired graph network.

- Lines 9-10: initialises the position of the agents (at random, using the `rand` function).
- Lines 12-23: codes the system dynamics, Ω, and the state space system.
- Line 25: defines vector t that specifies the sample time for the simulation
- Line 26: defines input signal u, declaring it is a vector of the same size as t
- Line 29: returns the system response `y` using the `lsim` function, specifying vector x0 are the initial state values of the state space model `sys`.

Having a simple line of code to calculate Ω using the `kron` function was preferred over calculating and defining the 12×12 matrix in MATLAB by hand. The latter is prone to error (e.g. typographical); the former is advantageous if there is a change in the parameters defined. As shown in Lines 15-16, there is a change in c_1 and c_2 as the parameters were doubled (explained in 8.1.1). However, due to the structure of the code, the changes are carried through with no alterations necessary; therefore, the method used is comparatively efficient.

11.1.4 Simulink Modelling

The procedure for designing and implementing in Simulink was the same for all models (with the addition of a block for the extensions). The closed-loop dynamics of every agent was correlated to a separate 'state-space block', and was colour coded to avoid confusion (e.g. Node 1 = 'Agent 1 dynamics' block = Red).

Through expanding and liking group terms, the dynamics for each agent (8.2) can be described as a state-space model. The method for each dynamic being the same, as the block for each state space can be simplified as:

- `Parameter block A:` $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -(\texttt{Numberofconnections}) & -c_1 & -c_2 \end{bmatrix}$
- `Parameter block B:` $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- `Parameter block C: eye(3)`
- `Parameter block D: zeroes(3,1)`

Which is equivalent to

$$\dot{\overline{\xi}}_i = E_m \overline{\xi}_i - \sum_{j \in \mathcal{N}_i} k_i a_{ij} (\xi_i - \xi_j)$$

With initial conditions (randomly chosen) $\in \mathbb{R}^{1*3}$ and the overall block diagram structure being simple and straightforward control theory, as demonstrated similarly in (7).

Additional time would increase familiarity with the software, introducing more effective ways of designing and implementing the same system. With programming and modelling, there are various ways to achieve the desired result. For example, there may be functions that are better suited to the problem or just as good as each other (e.g. `lsim` or `impulse`). Similarly, `Parametric sweep` was discovered after evaluations, but could have been beneficial as the function runs simulations repeatedly and iteratively sweeps through different values of the parameter(s); this is useful for graph analysis and data extraction. Additionally, the methodology for extracting the consensus rate may not have been the most sophisticated but was the initial/intuitive way of approaching data extraction, especially in a short time frame.

11.2 Limitations of Software

Mathworks offers MATLAB/Simulink online; however, experiencing drawbacks (such as lagging performance, inconsistent saving of work on the drive, slow execution of work), the decision to use MATLAB/Simulink was a solution. This was advantageous as it made running code and simulation much smoother and was faster. Consequently, the software application required a large amount of storage and memory, a full installation taking 24-31.5GB of disk space depending on the operating system [18], which may be a deterrent when working remotely or on a personal laptop (like this project).

Mentioned in Section 10.4, MATLAB does not provide a programme to plot the root locus of systems with time delay. As a result, Padé approximates are satisfactory, making the `Pade` function on `MATLAB` sufficient. As Bode plots and Frequency responses can be plotted in MATLAB easily, [17] describes frequency-designer methods in depth if diagrams are preferred.

Lastly, there are no costs to the project because the institution provides MATLAB licences to students; otherwise, the costs would be that of the MATLAB licence. For those without a licence, a free alternative to the software would be a viable solution (e.g. Python and the Computational Engines discussed in (2.4)).

12 Conclusion, Future Works and Reflection

12.1 Conclusion

This paper has considered an analysis of the consensus protocol described in Jiang et al. [1] for a desired group network of four nodes. MATLAB and Simulink simulations illustrated the dynamics of the linear system and successfully supported and verified the theoretical calculations.

Graph theory provided a foundation for graph network topologies and introduced the relevant matrices to describe a graph: The Laplacian and Adjacency Matrix. Matrix theory was crucial for eigenvalues (which help determine the relationship between the state variables, the response of the inputs and the stability of the system), state-space system representation, and introduced new concepts: the Kronecker product and Gershgorin theorem (which is applicable for relative stability analysis). Control theory was also a foundation for the project and introduced new concepts such as the Routh-Hurwitz stability criterion; this was necessary to derive the conditions for consensus to be achieved. The four-node graph network had system dynamics described by the matrix Ω and was thoroughly evaluated via a graph of eigenvalues (all in the OLHP).

Key concepts in these theories were expanded upon in detail to allow room for extension in the work, some successfully implemented. Extensions such as shifting from consensus to synchronisation, investigating time delay, and exploring Root Locus and Nyquist stability analysis were done to explore robustness and introduce sensitivity of a system.

12.2 Future Works

12.2.1 Graphical User Interface

A practical development/extension for this project could be producing a Graphical User Interface (GUI) programme- performing all simulations and testing on a single programme. The GUI would include convergence analysis, linear system analysis, and the stability analysis discussed in this report. MATLAB provides a Graphical User Interface (GUI) development environment called 'App Designer', which uses a drag and drop functionality and an integrated editor to programme its behaviour. An advantage would be continuous improvement of the GUI as work progresses, and it would be easier to include more points of interest. e.g. directed graphs, weighted graphs, different connections, non-linear systems. A GUI programme would be more time-efficient than the method operated in this report (creating and simulating individual programmes for each design). [3] executes this and illustrates the design procedure and basics of a GUI used for consensus for linear agents with unknown parameters, which could be the foundation of this development.

12.2.2 Network Topology effects on Consensus

Investigating the physical impact of network topology on system performance could be an extension of this project. Graph theory states that there are $2^{\frac{n(n-1)}{2}}$ total ways to connect a graph of n nodes ([25]). Exploring the effect of these connections on consensus, i.e. do all 64 graph connections for a 4-node system produce identical results? Ergo, the network topology is an extra parameter for consensus and could be studied. Is there an optimal connection to reach consensus at the fastest rate? Is there a way to describe the correlation between the optimal network connection and the number of nodes? [20] evaluates the 'Impact of the Physical Topology on the Optical Network Performance' and could be a good starting point for investigating the different topologies and their effects, as demonstrated in Figure 32.

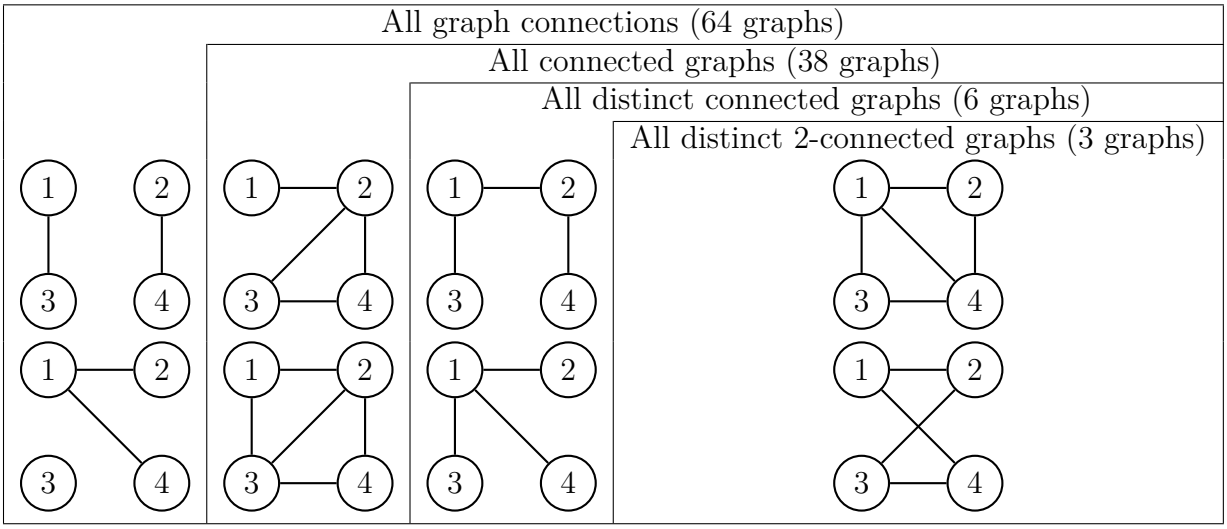


Figure 32: Possible network topologies for a 4 node graph

12.2.3 Consensus Optimisation

A further related area of interest is that of the consensus algorithms optimality: in what sense is the algorithm optimal? What is the nature of the trade-off between optimality and information sharing in networks? Designing optimal consensus control laws and adopting different techniques, such as: integral quadratic constraints and linear matrix-inequalities, as explored in [22]. A goal could be to explore the optimality properties of the consensus algorithms and obtain conditions that ensure that cost functions of the form

$$\int_0^\infty L(x,u,t)\ dt$$

are minimised. Literature tackling the optimisation problem such as the Network-Newton method [23] and the Sub-gradient-push method [24]; shift from consensus to consensus optimisation. Special attention could be devoted to uncovering the relationship between optimality, and the underlying graph structure of the network (and other parameters).

12.3 Reflection and Evaluation

After starting from scratch with zero knowledge of consensus, I am pleased with the outcome of the project. I faced personal issues early on in the academic year that created a 'snowball' effect, negatively impacting the project and progress. 'Special Considerations' and incredible support from my supervisor helped mitigate the effects, and given the circumstances, I am pleased with my accomplishments.

Regarding technical progress: this was a great learning opportunity as I'd not been confident in programming and was unfamiliar with the software deliverables. MATLAB posed a challenge but paid off upon succession.

The Gantt Chart (33) was modified as I cut out tasks that I felt strayed from the scope because the work at hand (LTI systems) alongside the concepts introduced provided much room for possibilities; the extra tasks were not as necessary in comparison. I feel as though my opinion changed over the course of the project as I understood the depth of the material and how vast it was. Overall, the pacing of the project was great, with my semester one and semester two modules complementing the project. The Gantt Chart was successfully followed, and in some instances: tasks were completed sooner than expected. This allowed room to explore and assign new tasks, and as a result, I feel more confident with the mathematical foundations of control.

If I were to start a new project or do things differently, I would draw attention to focusing on the core of the scope. Consequently, understanding the fundamentals and not tackling many tasks at once. A common mistake (which I made) was attempting to understand and include as much as possible relating to consensus, exploding the scope. This was unproductive, although it taught me the challenging side of researching and how to improve: by focusing on a problem of manageable size.

The structure of the project could be seen as over-structured, however, I believe the choices of subsections were appropriate to categorise the topics and improve readability and clarity. The report runs across ≈ 70 pages from cover to cover, which may seem excessive: the main cause being the inclusion of all Simulink diagrams/ scopes. Including all figures was a decision made as for those unfamiliar with MATLAB/Simulink, it would be difficult to visualise, even more so because I refer to the block parameters and the agent dynamics. Although the inclusion of all diagrams increases the report length drastically, it was a decision made for the project to be easier to follow. Another reason it may appear long-running is due to the many equations and figures; (which don't add to word count) but are necessary as this report is theory and mathematics heavy. Therefore, the many figures, equations, tables and graphs are vital for better understanding, flow and consistency of the project, although adding additional length.

References

[1] F. Jiang, L. Wang and Y. Jia, "Consensus in leaderless networks of higher-order-integrator agent", *Proc. 2009 Amer. Contr. Conf.*, (2009)

[2] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rule", *IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 988–1001*, (2003)

[3] P. Rapisarda, H. Sira-Ramírez, R. Savva, "Consensus for linear agents with unknown parameter", *IFAC-PapersOnLine, Volume 50, Issue 1, Pages 2505-2510*, (2017).

[4] Solmaz S. Kia et al. "Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms", *IEEE Control Systems* (2018)

[5] E. Sontag, "Mathematical control theory: deterministic finite-dimensional systems", (1998)

[6] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", *https://bitcoin.org/bitcoin.pdf.* (2008)

[7] V. Buterin, "A next-generation smart contract and decentralized application platform", *https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.* (2014)

[8] B Bitcoin "BlockChain Technology", *https://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf* (2015)

[9] M. Patel "Consensus Algorithms in Blockchain", *https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/* (2020)

[10] T. Omitola and C. Cirstea "COMP2207: Distributed Systems and Networks-Distributed Algorithms Part V: Consensus", *https://secure.ecs.soton.ac.uk/noteswiki/images/COMP2207-Consensus-2021.pdf*

[11] S. Brakken-Thal "Gershgorin's Theorem for Estimating Eigenvalues", *http://buzzard.ups.edu/courses/2007spring/projects/brakkenthal-paper.pdf* (2007)

[12] ECE 680 Modern Automatic Control "Routh's stability criterion", *http://et.engr.iupui.edu/~skoskie/ECE680/Routh.pdf* (2007)

[13] M. Jabbari, "Gershgorin_Disk MATLAB Central File Exchange", *https://www.mathworks.com/matlabcentral/fileexchange/72190-gershgorin_disk* (2021)

[14] S. Khodaverdian, M. Schneider and J. Adamy, "Synchronizing networks of heterogeneous linear systems via input-output decoupling", *IEEE Conference on Control Applications (CCA), 2014, pp. 681-686* (2014)

[15] Gregory L. Plett and M. Scott Trimboli "ECE4510/5510: Feedback Control Systems", *http://mocha-java.uccs.edu/ECE4510/ECE4510-Notes06.pdf* (1998–2013)

[16] Bondi, André B. "Characteristics of scalability and their impact on performance", *Proceedings of the second international workshop on Software and performance – WOSP '00. p. 195.* (2000)

[17] Franklin, Gene, et al. Feedback Control of Dynamic Systems, EBook, Global Edition, Pearson Education, Limited, ProQuest Ebook Central *https://ebookcentral.proquest.com/lib/soton-ebooks/detail.action?docID=5770170.* (2019)

[18] MathWorks, MATLAB and Requirements Toolbox, "System Requirements for MATLAB R2022a" *https://uk.mathworks.com/support/requirements/matlab-system-requirements.html*

[19] Tegling, Emma & Middleton, R.H. & Seron, M.M. "Scalability and Fragility in Bounded-Degree Consensus Networks", *IFAC-PapersOnLine. 52. 85-90.* (2019)

[20] Tessinari, Rodrigo & Paiva, Marcia & Monteiro, Maxwell & Segatto, Marcelo & Garcia, Anilton & Kanellos, George & Nejabati, Reza & Simeonidou, Dimitra, "On the Impact of the Physical Topology on the Optical Network Performance", *IEEE British and Irish Conference on Optics and Photonics (BICOP 2018)*

[21] I. Saboori and K. Khorasani, "An H∞ consensus achievement for LTI multi-agent systems with directed fixed networks," *25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2012, pp. 1-5,* (2012)

[22] E. Semsar-Kazerooni and K. Khorasani, "Optimal con- sensus seeking in a network of multiagent systems: An lmi approach," *IEEE Trans. on Systems, Man, and Cybernetics, vol. 40, no. 20, pp. 540–547, 2010.* (2010)

[23] A. Mokhtari, Q. Ling and A. Ribeiro, "Network Newton Distributed Optimization Methods" *IEEE Transactions on Signal Processing, vol. 65, no. 1, pp. 146-161, 1 Jan.1, 2017* (2017)

[24] A. Nedić and A. Olshevsky, "Distributed Optimization Over Time-Varying Directed Graphs," *IEEE Transactions on Automatic Control, vol. 60, no. 3, pp. 601-615, March 2015*

[25] F. Harary and E. Palmer, Graphical Enumeration. Elsevier Science, 2014. *https://books.google.co.uk/books?id=ZrvSBQAAQBAJ* (2014)

A Project Management

A.1 Gantt Chart

A.1.1 Original Projected Gantt Chart

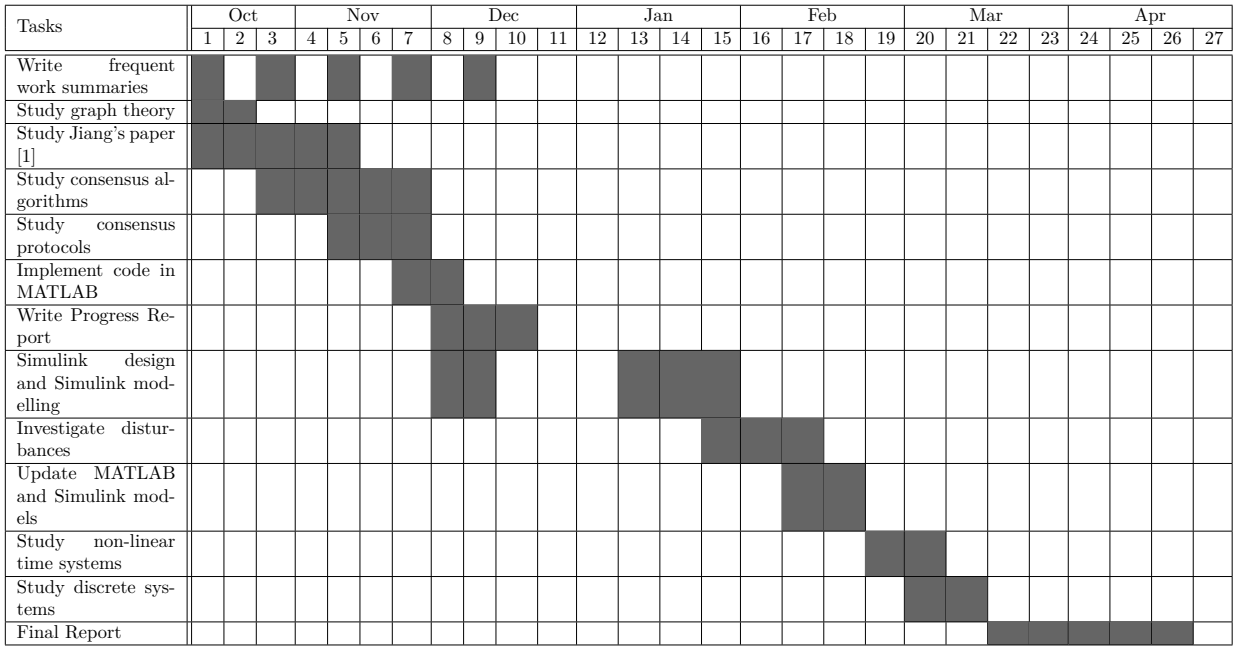


Figure 33: The projected gantt chart planning the project

A.1.2 Modified Gantt Chart

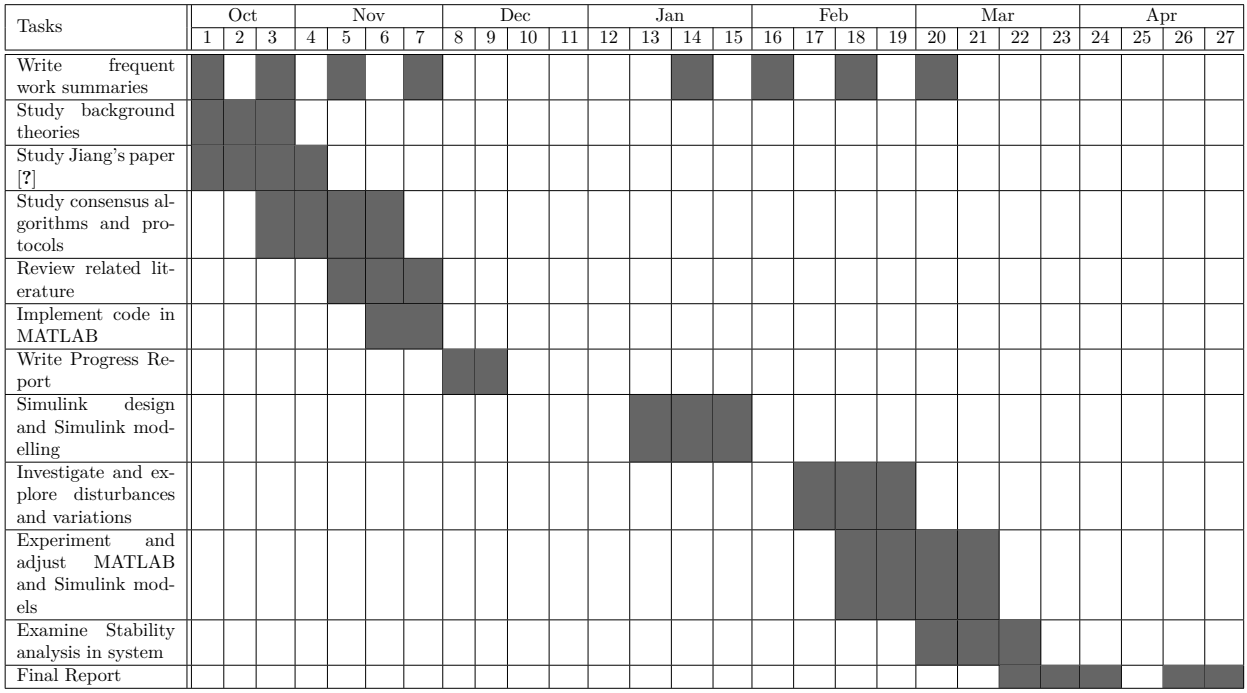


Figure 34: The modified gantt chart presenting the progression of the project

A.2 Risk Assessment

Risk	Likelihood	Impact	Solution
Poor time management due to workload from other deadlines/-modules	H	H	Set tasks, prioritise and schedule tasks and deadlines, dedicate X amount of time per day/week for the individual project
Illness/medical concerns	M	H	Special considerations, support for academic performance from university and updating supervisor on well-being if necessary
Data/work being lost	L	H	Use of multiple services: cloud, USB, etc for the report and notes, ensuring there are backups
Cloud-based LaTeX editor (Overleaf) Server being down	M	H	Saving and resuming on an offline TeX editor allows write-ups to continue and mitigates the impacts.

B MATLAB Code

Gershgorin Circle Theorem

```
%% state space system

% Adjacency matrix
Adj=[ 0 1 1 1; 1 0 0 0 ; 1 0 0 1; 1 0 1 0];
% Degree matrix
Deg= [ 3 0 0 0; 0 1 0 0 ; 0 0 2 0; 0 0 0 2];
% Laplacian matrix= Degree-Adjacency
L= Deg-Adj;

I=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
Y=I;
E=[ 0 1 0; 0 0 1; 0 -2 -3];
F=[ 0 0 0; 0 0 0; 1 0 0 ];

O=(kron(I,E))-(kron((Y*L),F));
V=ones(12,1);
W=eye(12);
X=0;
sys=ss(O,V,W,X);

%MohammadReza Jabbari (Jabari) (2021). Gershgorin Disk (
   https://www.mathworks.com/matlabcentral/fileexchange
   /72190-gershgorin_disk),
%MATLAB Central File Exchange. Retrieved December 5, 2021.

A=O;

if size(A,1) ~= size(A,2)
error ('Matrix should be square');
return
end
% Gershgorin Disk Theorem
Lambda = eig(A); %Computing true eigenvalues of desired
   Matrix 'A'
% Obtaining the centers of disks
X = real(diag (A)) ;
Y = imag(diag (A)) ;
% Obtaining the radius of disks
R = sum(abs(A) ,2)-abs(diag(A));
% Plotting Results
```

```
% Plot the location of true eigenvalues in real-imaginary
   coordinate(X axis==real part, Y axis==imaginary part)
plot(real(Lambda),imag(Lambda),'or');
hold on
% Plot the centers of disks approximated by Gershgorin
   Theorem
plot (X, Y, 'k*') ;
% To plot the disks, we use polar coordinate equation of a
   circle
N= 300;
t = 0:2*pi/N: (2*pi)- (2*pi/N);
X1 = X+R.*repmat (cos(t),size(R, 1),1);
Y1 = Y+R.*repmat(sin(t),size(R,1),1);
plot (X1' ,Y1', 'b-')

axis equal;
hold off
xlabel ('Real' );
ylabel('Imaginary')
title ('Gershgorin Disk')
legend ('Actual Eigenvalues', 'Center of Disks', 'Disks')
grid on
```

Code from [13] to calculate Gershgorin circles for the desired graph network and system dynamics on Matlab

Four Agent System Consensus Algorithm Implementation

```
close all
% Adjacency matrix
A=[ 0 1 1 1; 1 0 0 0 ; 1 0 0 1; 1 0 1 0];
% Degree matrix
D= [ 3 0 0 0; 0 1 0 0 ; 0 0 2 0; 0 0 0 2];
% Laplacian matrix= Degree-Adjacency
L= D-A;

%Agent initial position (random)
x0=rand(12,1);

%% system
I=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
Y=I;
%E=[ 0 1 0; 0 0 1; 0 -2 -3];
E=[ 0 1 0; 0 0 1; 0 -4 -6];
F=[ 0 0 0; 0 0 0; 1 0 0 ];

O=(kron(I,E))-(kron((Y*L),F));
V=ones(12,1);
W=eye(12);
X=0;
sys=ss(O,V,W,X);

t = 0:0.01:30;
%u = max(0,min(t-1,1));
u=zeros(size(t));

[y,t,x]=lsim(sys, u, t, x0);

%% plot
figure;
plot(t,y(:,1),'r');
hold on
plot(t,y(:,4),'b');
plot(t,y(:,7),'g');
plot(t,y(:,10),'m');
grid on
title('1st state component')
hold off

```

```
figure;
plot(t,y(:,2),'r');
hold on
plot(t,y(:,5),'b');
plot(t,y(:,8),'g');
plot(t,y(:,11),'m');
grid on
title('2nd state component')
hold off

figure;
plot(t,y(:,3),'r');
hold on
plot(t,y(:,6),'b');
plot(t,y(:,9),'g');
plot(t,y(:,12),'m');
grid on
title('3rd state component')
hold off
```

Matlab Code designed and implemented to compute the consensus algorithm for the graph network discussed in 8, using parameters $c_1 = 4$ and $c_2 = 6$

C MATLAB Implementation of Scalability

C.1 Two Agents, N=2

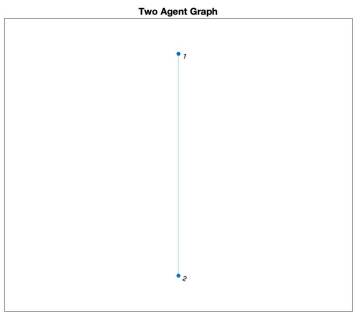
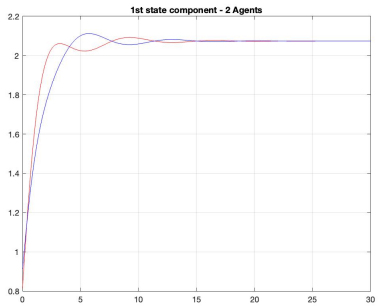
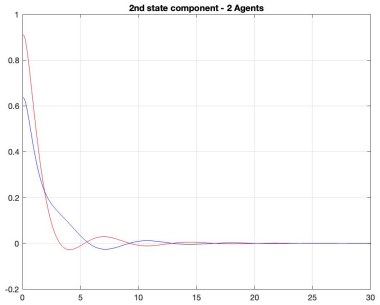
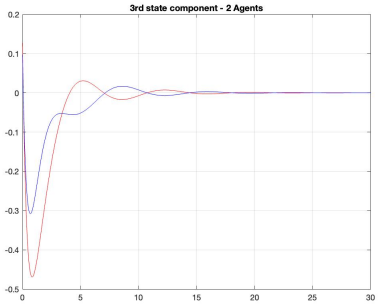
Graph	
1st State Component	
2nd State Component	
3rd State Component	

Figure 35: MATLAB Consensus Algorithm implementation of Two Agent Graph Network using `'TwoAgents.mlx'`

C.2 Ten Agents, N=10

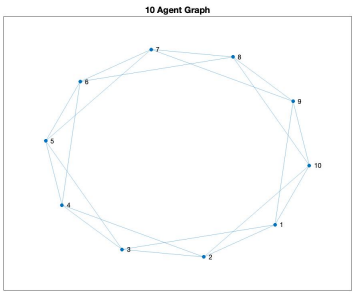
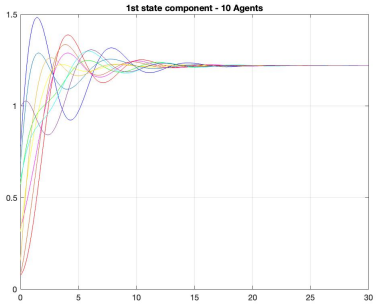
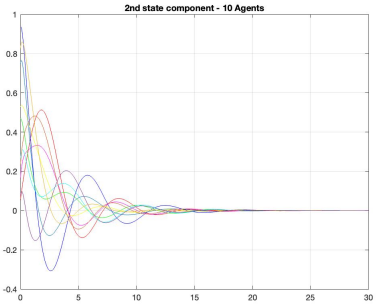
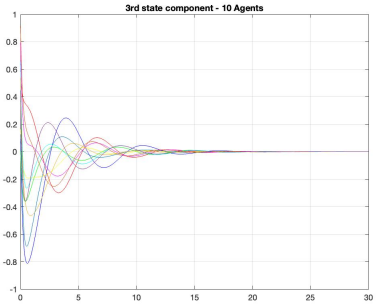
Graph	
1st State Component	
2nd State Component	
3rd State Component	

Figure 36: MATLAB Consensus Algorithm implementation of Ten Agent Graph Network using `'TenAgents.mlx'`

C.3 Twenty Agents, N=20

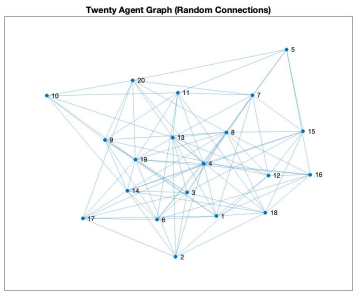
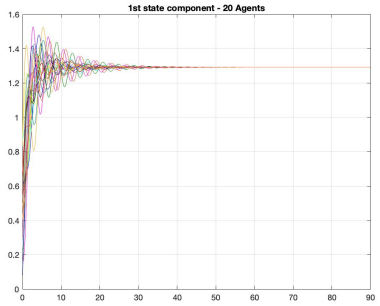
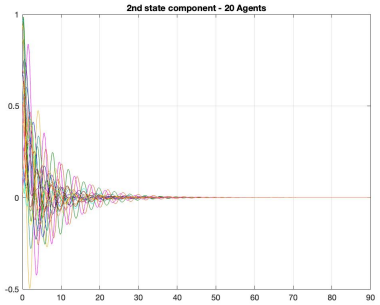
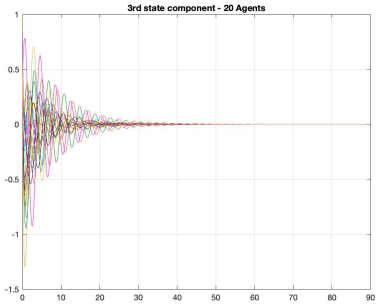
Graph	
1st State Component	
2nd State Component	
3rd State Component	

Figure 37: MATLAB Consensus Algorithm implementation of Twenty Agent Graph Network using `'TwentyAgents.mlx'`

D Design Archive

- Design Archive
  - MATLAB Agent Simulation Code
    - Plain Scripts (.m files)
      - `FourAgents.m`
    - Live Scripts for $N \neq 4$ (.mlx files)
      - `TwoAgents.mlx`
      - `TenAgents.mlx`
      - `TwentyAgents.mlx`
  - Simulink Models
    - FourAgents.slx
    - FourAgentsNoise.slx
    - FourAgentsSinusoidal.slx
    - FourAgentsTimeDelay.slx
  - Stability MATLAB Code
    - `Gershgorin.mlx`
  - Consensus State Verification MATLAB Code
    - `ConsensusState.mlx`

E Word Count

Verifiable Overleaf Word-count: 8999

F Original Project Brief

1 Problem

A distributed system is a group of network components that work together to act as a single entity. For them to act as a single entity they must agree on 'some' data value i.e. consensus. To obtain this, there are algorithms which instruct each agent, also known as consensus protocols. These protocols must be fault tolerant, resilient, and designed to deal with a limited number of faulty processes. There are two methods: distributed method and centralised method. Due to the drawbacks of centralised solutions, the distributed method will be pursued. Solving the consensus problem and working towards an ideal solution (with no failures / must tolerate process failures) is necessary due to the applications consensus has on networks such as autonomous formation flight, control of communication networks, distributed sensor fusion in sensor networks, swarm-based computing, etc. A graph is said to be simply strong or strongly connected if there is a path in each direction between pairs of nodes. In this case a node will represent an agent.

2 Goals

The objective is to implement and simulate consensus protocols for linear continuous-time multi-agent systems. The project deliverables will be MATLAB programs and Simulink models to evaluate the performance of various consensus algorithms for systems with fixed topology, under some realistic scenarios (including disturbances, malicious attacks). The first approach to be analysed is that described in the paper by Jiang et al. [1] for higher-order integrator systems consensus. The methodology used will be a foundation for this paper as similarly the dynamics are unknown, but also using [2] and [3] for context and altering the process as best to fit the project.

3 Scope

Planning with Gantt to manage tasks and assigning an approximate time frame to each major and sub task, but also being flexible and taking under consideration that things that may not go to plan. For example, background research may take longer than expected, it may be apparent later that some topics require more research, there may be data loss or software issues, and it is likely that at some point due to illness there will be low productivity. Background research for the relevant topics such as graph theory, matrix theory and control theory as well as revising the topics is necessary. To develop understanding, looking at well-known consensus algorithms may be explored. Agreeing on a geometry of agents will verify understanding and then be simulated on MATLAB to visually represent the complex system. Simulink models will put the approaches to test in more realistic situations

References

[1] F. Jiang, L. Wang and Y. Jia, "Consensus in leaderless networks of higher-order-integrator agents", *Proc. 2009 Amer. Contr. Conf.*, 2009.

[2] . Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules", *IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 988–1001*, 2003.

[3] . Rapisarda, H. Sira-Ramírez, R. Savva, "Consensus for linear agents with unknown parameters", *IFAC-PapersOnLine, Volume 50, Issue 1, Pages 2505-2510*, 2017.

Figure 38: Original project brief submission