Reducing Multi-Agent Systems in Discrete Time

Bachelor thesis Applied Mathematics

October 2013

Student: M. Jaspers

Supervisor: Prof. dr. H.L. Trentelman, N. Monshizadeh

Advisor: Prof. dr. H. Waalkens

Reducing multi-agent systems in discrete time

Abstract

Multi-agent systems are, in general, complicated and have a high complexity. In order to make it easier to work with these systems, the complexity of the models has to be reduced. This reduction should be done without loss of properties of the system. A method for this in continuous time is made by H. L. Trentelman and N. Monshizadeh [1]. In this report we will describe a method to reduce the complexity of multi-agent systems in discrete time, conserving its properties as much as possible.

Contents

1 Introduction

In the 80's and throughout the 90's there has been an increase of interest in multi-agent systems in computer science and systems and control. This interest has also spread in other fields, leaving even fields such as physics and biology requiring more theory on applications and properties of multi-agent systems. In mathematics this flow of interest has yielded in more research on multi-agent systems and properties of these systems.

As said before, multi-agent systems are, in general, complicated and have a high complexity. This makes working with these systems more difficult and therefore we want to reduce the complexity of these systems. This can be done in several ways. In this report we will discuss a method to do this for discrete time.

In order to do this, we first need some general background on systems, which will be discussed in the second section.
After that we will briefly discuss the method for continuous time multi-agent systems in the third section. This section will contain three subsections; some theory behind Petrov-Galerkin projections will be discussed in the first subsection, the Petrov-Galerkin projection applied to the general system will be discussed in the second subsection and input-output approximations of multi-agent systems will be discussed in the third subsection.

In the fourth section we will discuss an example on the normalized reduction error between the original and the reduced order model in continuous time. The subject of the fifth section will be the method for discrete time. This section will contain four subsections; again some theory behind Petrov-Galerkin projections will be discussed in the first subsection, the Petrov-Galerkin projection applied to the general system will be discussed in the second subsection, one of the most important properties of the multi-agent system, the consensus preservation for the reduced order system, will be discussed in the third subsection and input-output approximations of multi-agent systems will be discussed in the fourth subsection. The report will end with a conclusion.

2 General knowledge

In this section we will discuss some general material on graphs, some matrices associated with graphs and multi-agent systems.

2.1 Graph theory

Definition 1: Graph
A graph G is a pair $G = (V, E)$, where $V = \{1, 2, ..., n\}$ with n a positive integer, and $E \subseteq V \times V$. Any element of V is called a vertex and any element of E is called an edge of the graph G. In general the graph G, as defined above, is called a *directed* graph. An undirected graph is a pair $G = (V, E)$ where $V = \{1, 2, ..., n\}$ and E is a set of unordered pairs $\{i, j\}$ with $i, j \in V$.

In this report self-loops (edges $(i, i) \in E$ or $\{i, i\} \in E$ of G) and multiple edges (multiple arcs in the same direction) between one particular pair of vertices are not permitted.

An example of an undirected graph is given in Figure (1).

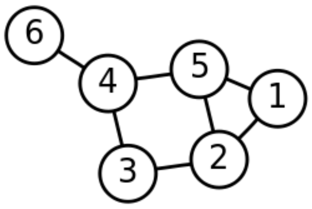


Figure 1: Example of a graph

For this example V and E are given by:

$V = \{1, 2, 3, 4, 5, 6\}$ and
$E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$.

A *path of length* k in a directed graph is a sequence of distinct vertices $i_1, \ldots, i_{k+1}$ such that $(i_m, i_{m+1}) \in E$ for $m = 1, \ldots, k$. A path of length k in an undirected graph is a sequence of distinct vertices $i_1, \ldots, i_{k+1}$ such that $\{i_m, i_{m+1}\} \in E$ for $m = 1, \ldots, k$. In a directed graph a path will be called a *directed path*, a path in an undirected graph will be called an *undirected path.*

For an undirected graph the *distance* from i to j is defined to be the length of the shortest undirected path from i to j, the distance between i and i is defined to be zero. The *degree* of a vertex i for an undirected graph is equal to the number of vertices j for which $\{i, j\} \in E$. For a directed graph the *distance* from i to j is defined to be the length of the shortest directed path from i to j, the distance between i and i is again defined to be zero. When defining the *degree* of a vertex i for a directed graph we distinguish between in-degree and out-degree. The in-degree of a vertex i is equal to the number of vertices j for which $(j, i) \in E$. The out-degree of a vertex i is equal to the number of vertices j for which $(i, j) \in E$.

Definition 2: Connected graph
A directed graph is called connected if there exists a directed path from every vertex i to every other vertex j of the graph. An undirected graph is called connected if there exists an undirected path from every vertex i to every other vertex j of the graph.

Definition 3: Directed spanning tree
A directed graph G has a directed spanning tree if there exists a vertex $r \in V$ such that all other vertices in V can be linked to r via a directed path.

Definition 4: Weighted directed graph
A weighted directed graph is a directed graph $G = (V, E)$ with $V = \{1, 2, ..., n\}$, n a positive integer, and $E \subseteq V \times V$, where with each $(j, i) \in E$ we associate a positive real number w_{ij}, called the weight of (j, i).

Definition 5: Weighted undirected graph
A weighted undirected graph is an undirected graph $G = (V, E)$ with $V = \{1, 2, ..., n\}$, n a positive integer, and E a set of unordered pairs $\{i, j\}$ with $i, j \in V$, where with each $\{j, i\} \in E$ we associate a positive real number w_{ij}, called the weight of $\{j, i\}$.

2.2 Matrices associated with graphs

Definition 6: Weighted adjacency matrix
The weighted adjacency matrix $A = [a_{ij}]$ of the weighted directed graph G with weigths w_{ij} for $(j, i) \in E$ is the square $n \times n$ matrix with elements a_{ij}, in which a_{ij} is defined as:

$$a_{ij} = \begin{cases} w_{ij} & \text{if } (j, i) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The weighted adjacency matrix of a weighted undirected graph is defined analogously as in Definition 6. Note that in this case the adjacency matrix is symmetric.

Definition 7: Stochastic matrix
A square matrix $M = [m_{ij}] \in \mathbb{R}^{n\times n}$ is called a stochastic matrix if $m_{ij} \geq 0$ for every i, j and $\displaystyle\sum_{j=1}^{n} m_{ij} = 1$ for all $i = 1, \ldots, n$.

Another matrix associated with graphs that we will use in this report is the Laplacian matrix.

Definition 8: Laplacian matrix
The Laplacian matrix of a graph G is defined by $L = D - A$. Here the matrix D is the diagonal matrix in which the in-degrees (if G is directed) or degrees (if G is undirected) of the vertices are on the diagonal and A is the weighted adjacency matrix as defined in Definition 6.

As an example we look again at the graph of Figure (1). This graph consists of 6 vertices, so our Laplacian matrix L will be a 6×6-matrix. We first construct the adjacency matrix A. Vertex 1 is only connected to vertices 2 and 5. We repeat this for all other vertices to obtain the matrix A:

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

The matrix D follows from A by putting $d_{ii} = \sum_{j=1}^{n} a_{ij}$ and $d_{ij} = 0$ for $j \neq i$.

To obtain L we subtract A from D:

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

Since $L = D - A$, we can also write $L = [l_{ij}]$ with

$$l_{ii} = \sum_{j=1}^{n} a_{ij}, l_{ij} = -a_{ij}, \text{ for } i \neq j. \tag{1}$$

For an undirected graph, the adjacency matrix is symmetric so the Laplacian matrix will also be symmetric. Furthermore all off-diagonal elements of the Laplacian matrix are non-positive and $\sum_{j=1}^{n} l_{ij} = 0$ for each i.

If the Laplacian matrix L is weighted and symmetric, it can also be written as $L = RWR^T$. Here the matrix $R = [r_{ij}]$, which is called the *incidence matrix* of the directed graph G, is defined as:

$$r_{ij} = \begin{cases} 1 & \text{if the } j\text{-th edge starts at vertex } i, \\ -1 & \text{if the } j\text{-th edge ends at vertex } i \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$, where k is the total number of edges. An incidence matrix for an undirected graph can be obtained by first assigning an arbitrary orientation to each of the edges and next take the incidence matrix of the corresponding directed graph (see [10], p.21). Furthermore let the matrix $W \in \mathbb{R}^{k \times k}$ be given by:

$$W = \text{diag}(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_k), \tag{3}$$

with $\tilde{w}_j$ a positive number, called the weight associated to the edge j for each $j = 1, 2, \ldots, k$.

Both matrices and a list of vertices and edges can be used to describe the same graph. As seen before, matrices can be associated with graphs, but also graphs can be associated to matrices. This is what we will see in the next definition.

Definition 9: Weighted directed graph associated to a matrix A weighted directed graph associated to the matrix $M = [m_{ij}] \in \mathbb{R}^{n \times n}$, denoted by $G(M) = (V, E)$, is a weighted directed graph with $V = \{1, 2, ..., n\}$ such that $(j, i) \in E$ if $i \neq j$ and $m_{ij} \neq 0$. Note that $G(M)$ does not contain self-loops so $(j, i) \notin E$ for $j = i$.

2.3 Multi-agent systems

In this subsection we will discuss some general theory behind multi-agent systems. Let $G = (V, E)$ be a weighted undirected graph with weighted adjacency matrix $A = [a_{ij}]$. The set of vertices is given by $V = \{1, 2, ..., n\}$ and E is a set of unordered pairs $\{i, j\}$ with $i, j \in V$. Choose $V_L = \{v_1, v_2, ..., v_m\}$ to be a subset of V and let $V_F = V \setminus V_L$, a vertex $i \in V_L$ is called leader and a vertex $i \in V_F$ is called a follower.

Definition 10: Leader-follower multi-agent system, continuous time
A leader-follower multi-agent system in continuous time is given by the following dynamical system:

$$\dot{x}_i(t) = \begin{cases} \sum_{j=1}^{n} a_{ij}(x_j(t) - x_i(t)) & \text{if } i \in V_F, \\ \sum_{j=1}^{n} a_{ij}(x_j(t) - x_i(t)) + u_l(t) & \text{if } i \in V_L. \end{cases}$$

Here $x_i(t) \in \mathbb{R}^n$ represents the state of agent i and $u_l(t) \in \mathbb{R}$ is the external input applied to agent $i = v_l$.

The multi-agent system associated with the graph G can also be written as (4):

$$\dot{x}(t) = -Lx(t) + Mu(t), \tag{4}$$

with L the Laplacian matrix of G, $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $u(t) = [u_1(t), u_2(t), \ldots, u_m(t)]^T$ and $M \in \mathbb{R}^{n \times m}$ given by :

$$M_{il} = \begin{cases} 1 & \text{if } i = v_l, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 11: Leader-follower multi-agent system, discrete time
In discrete time each agent updates its state to the weighted average of the states of all other agents. Therefore a leader-follower multi-agent system in discrete time is given by the following dynamical system:

$$x_i(t+1) = \begin{cases} \sum_{j=1}^{n} q_{ij} x_j(t) & \text{if } i \in V_F, \\ \sum_{j=1}^{n} q_{ij} x_j(t) + u_l(t) & \text{if } i \in V_L. \end{cases}$$

Here, again $x_i(t) \in \mathbb{R}$ represents the state of agent i, $u_l(t) \in \mathbb{R}$ is the external input applied to agent $i = v_l$ and $q_{ij} \geq 0$ is the weight agent i assigns to agent j when agent i updates its state.

Similarly as in the continuous time case, this can be rewritten as:

$$x(t+1) = Qx(t) + Mu(t), \tag{5}$$

where $Q = [q_{ij}] \in \mathbb{R}^{n \times n}$, $x(t) = [x_1(t), \dots, x_n(t)]^T$,
$u(t) = [u_1(t), u_2(t), \dots, u_m(t)]^T$ and the matrix $M \in \mathbb{R}^{n \times m}$:

$$M_{il} = \begin{cases} 1 & \text{if } i = v_l, \\ 0 & \text{otherwise.} \end{cases}$$

The weights q_{ij} satisfy the following condition: $\sum_{j=1}^{n} q_{ij} = 1$,
$q_{ij} \geq 0$ and $q_{ii} > 0$ for $i \in \{1, 2, \dots, n\}$. So Q is a stochastic matrix.

Just like we have matrices associated with graphs, we also have matrices associated with multi-agent systems. An example of such a matrix is the transfer matrix.

Definition 12: Transfer matrix, continuous time
The transfer matrix $H(s)$ of a general system in continuous time,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned}$$

is defined by $H(s) = C(sI - A)^{-1}B$.

Definition 13: Transfer matrix, discrete time
The transfer matrix $H(z)$ of a general system in discrete time

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned}$$

is defined by $H(z) = C(zI - A)^{-1}B$.

With this general background on systems we are now able to proceed with the method to reduce the complexity of multi-agent systems. In the next section we will first briefly discuss the method for continuous time multi-agent systems.

3 Method for continuous time multi-agent systems

In order to obtain a better insight in the method for discrete time multi-agent systems, we will first briefly discuss the method for the continuous time case in this section.

Remark 3.1
In this section we will only give a brief explanation behind the method for the continuous time case, for more details we refer to [1] and the explanation behind the method for discrete time, section 5.

3.1 Petrov- Galerkin projections

We start with some general theory behind Petrov-Galerkin projections.

Let the general input-state-output system be given by:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{6}$$

with $x(t) \in \mathbb{R}^n$ the state, $u(t) \in \mathbb{R}^m$ the input and $y(t) \in \mathbb{R}^p$ the output of the system.

Furthermore let $\mathbf{V}$ and $\mathbf{W} \in R^{n \times r}$ such that $\mathbf{W}^T\mathbf{V} = I$. A reduced order model can be obtained by using the projection $\Gamma = \mathbf{V}\mathbf{W}^T$ in the following way: first substitute $\dot{x}(t)$ by $\mathbf{V}\dot{\hat{x}}(t)$ in (6) and then premultiply the first line with $\mathbf{W}^T$.

Summarizing, we obtain the following system:

$$\begin{aligned} \dot{\hat{x}}(t) &= \mathbf{W}^T A\mathbf{V}\hat{x}(t) + \mathbf{W}^T Bu(t), \\ y(t) &= C\mathbf{V}\hat{x}(t), \end{aligned} \tag{7}$$

in which $\hat{x}(t) \in \mathbb{R}^r$ is the state of the reduced order system.

This projection is called a Petrov-Galerkin projection. More information on Petrov-Galerkin projections is described in section 5, subsection 5.1.

3.2 Projection by graph-partitions

In this section we will apply the Petrov-Galerkin projection, as discussed in section 3.1, to the system (4). This is a method to reduce the complexity of the system.

If we apply the Petrov-Galerkin projection to the system (4), the system becomes:

$$\dot{\hat{x}}(t) = -\mathbf{W}^T L \mathbf{V} \hat{x}(t) + \mathbf{W}^T M u(t). \tag{8}$$

A disadvantage of a Petrov-Galerkin projection is that it in general destroys the spatial structure of the network. In general the matrix $\mathbf{W}^T L \mathbf{V}$ in the representation of the reduced order system will not be structured, and so the reduced order system can not be written as (4). By using graph partitions the structure of the network will be preserved. This is what we will exploit.

We now introduce some general background on partitions.
Let $V = \{1, 2, \dots, n\}$ be the set of vertices of the graph G. Any nonempty subset of V is called a *cell* C_i of V. A collection of cells, given by $\pi = \{C_1, C_2, \dots, C_r\}$ is called a *partition* of V if $\cup_i C_i = V$ and $C_i \cap C_j = \emptyset$ whenever $i \neq j$. Vertices i and j belong to the same cell, i.e. are *cell mates* in π, if $i, j \in C_k$ for $k \in \{1, 2, \dots, r\}$. The *characteristic vector* of a cell $C_k \subseteq V$ is defined by:

$$P(C_k) := \begin{pmatrix} p_1(C_k) \\ p_2(C_k) \\ \vdots \\ p_n(C_k) \end{pmatrix}, \tag{9}$$

in which $p_i(C_k) := \begin{cases} 1 & \text{if } i \in C_k, \\ 0 & \text{otherwise,} \end{cases}$

with $i \in \{1, 2, \dots, n\}$.

The *characteristic matrix* of the partition $\pi = \{C_1, C_2, \dots, C_r\}$ is defined by: $P(\pi) = [P(C_1)\ P(C_2)\ \dots\ P(C_r)]$.

Let us now again look at the system (8), with associated graph $G = (V, E)$, $\pi = \{C_1, C_2, \dots, C_r\}$ a partition of V and $P(\pi)$ the characteristic matrix of π.

Choosing $\mathbf{W}^T$ and $\mathbf{V}$ in the following way :

$$\begin{aligned} \mathbf{W}^T &= (P^T(\pi)P(\pi))^{-1}P^T(\pi), \\ \mathbf{V} &= P(\pi), \end{aligned} \tag{10}$$

we obtain that $\mathbf{W}^T\mathbf{V} = I$. The columns of $P(\pi)$ are orthogonal, $P^T(\pi)P(\pi)$ is a diagonal matrix with strictly positive diagonal elements and is therefore invertible. If we write $P(\pi)$ as P then system (8) can be written as:

$$\dot{\hat{x}}(t) = -\hat{L}\hat{x}(t) + \hat{M}u(t), \tag{11}$$

with

$$\begin{aligned} \hat{L} &= (P^TP)^{-1}P^TLP, \\ \hat{M} &= (P^TP)^{-1}P^TM. \end{aligned} \tag{12}$$

Let the graph of the reduced order system be called $\hat{G}$. Since P contains only zeros and ones, the structures of $\hat{M}$ and $\hat{L}$ are similar to the structures of M and L respectively, but the input signals are now weighted. By this each cell of $\pi \subset V$ of the graph G is mapped to a vertex of $\hat{G}$, which means that the number of cells in π is equal to the number of vertices in $\hat{G}$. The matrix $\hat{L}$ does not have to be symmetric (the number of vertices may differ from cell to cell in π) but is similar to the symmetric matrix $(P^TP)^{-\frac{1}{2}}P^TLP(P^TP)^{\frac{1}{2}}$, thus $\hat{L}$ inherits properties of L like diagonalizability and having real eigenvalues.

Summarizing, this method (the projection) describes a way to bundle some vertices together and map them to a single vertex. By this we reduce the order of the system and the reduced order model becomes associated with a new multi-agent system based on the associated graph $\hat{G}$. Furthermore the reduced state $\hat{x}$ approximates the average of the states of the agents that are in a same cell of π. If the cell mates in π have a similar connection to the rest of the network, the approximation will become exact.

In the next section we will discuss appropriate choices of partitions such that the input-output behavior of the reduced order model remains 'close' to the input-output behavior of the original model.

3.3 Input-Output approximation of multi-agent systems

In this section we will discuss appropriate choices of partitions such that the input-output behavior of the reduced order model remains 'close' to the input-output behavior of the original model.

So far, we have only looked at multi-agent systems described by the inputs and states of the agents. We will now also look at multi-agent systems with outputs. In order to do this we first assume that the associated graph G is connected. If G is not connected, the proposed model reduction technique can be applied to the disconnected components of G individually.

We choose the output as $y(t) = W^{\frac{1}{2}} R^T x(t)$, where R is the incidence matrix of G defined by (2) and W is defined by (3). (The explanation why we exactly choose this to be the output is given in section 5, subsection 5.4.)

The following input-state-output representation is now obtained for the original multi-agent system:

$$\begin{aligned} \dot{x}(t) &= -Lx(t) + Mu(t), \\ y(t) &= W^{\frac{1}{2}} R^T x(t), \end{aligned} \tag{13}$$

where the first equation is as before, R is the incidence matrix of G, W is defined by (3) and $L = [l_{ij}]$ is the Laplacian matrix given by (1).

We now choose π again to be a partition of G. The input-state-output model for the reduced order multi-agent system is defined by:

$$\begin{aligned} \dot{\hat{x}}(t) &= -\hat{L}\hat{x}(t) + \hat{M}u(t), \\ y(t) &= W^{\frac{1}{2}} \hat{R}^T \hat{x}(t), \end{aligned} \tag{14}$$

where $\hat{L}, \hat{M}$ are given by (12), $\hat{x} \in \mathbb{R}^k$ with $k \leq n$, W is given by (3), and $\hat{R}^T = R^T P$, where P is a shorthand notation for $P(\pi)$.

In order to approximate the behavior of the original multi-agent system as efficient as possible, we have to choose an appropriate partition. There are two trivial partitions: one is taking each vertex as a singleton and the other one is taking the whole set of vertices as the projection, i.e. $\pi = \{V\}$. No order reduction occurs in the first case and therefore the corresponding reduction error will be zero. In the second case the reduced model will be a single agent with a zero transfer matrix from u to y. The finest and the coarsest approximation by graph partitions are obtained by these two trivial partitions. In general we have to find a compromise between the order of the reduced model and the accuracy of the approximation.
We first introduce some background on almost equitable partitions.

Definition 14: Almost equitable partition, unweighted undirected graph
Let $G = (V, E)$ be an unweighted undirected graph. For a given cell $C \subseteq V$, we write $N(i, C) = \{j \in C | \{i, j\} \in E\}$. Now a partition $\pi = \{C_1, C_2, \ldots, C_r\}$ is called an almost equitable partition of G if for each $p, q \in \{1, 2, \ldots, r\}$ with $p \neq q$ there exists an integer d_{pq} such that $|N(i, C_q)| = d_{pq}$ for all $i \in C_p$.

Definition 15: Almost equitable partition, weighted undirected graph
Let $G = (V, E)$ be a weighted undirected graph. Recall that a_{ij} indicates the nonzero weights associated to edge $\{i, j\}$. Now a partition $\pi = \{C_1, C_2, \ldots, C_r\}$ is called an almost equitable partition of G if for each $p, q \in \{1, 2, \ldots, r\}$ with $p \neq q$ there exists an integer d_{pq} such that $\sum_{j \in N(i, C_q)} a_{ij} = d_{pq}$ for all $i \in C_p$.

Let us now assume that $\pi = \{C_1, C_2, \ldots, C_r\}$ is an almost equitable partition of the weighted undirected graph G that contains no self-loops. Furthermore suppose that the reduced order model (14) is obtained from the original model (13) by the partition π. Recall that $V_L = \{v_1, v_2, ..., v_m\}$ and let k_i be an integer such that $v_i \in C_{k_i}$ for each $i = \{1, 2, \ldots, m\}$. Then the normalized model reduction error between the original and the reduced order model is provided in the following theorem.

Theorem 3.1. *Let G be a weighted undirected graph that is connected and contains no self-edges. Let $\pi = \{C_1, C_2, \ldots, C_r\}$ be an almost equitable partition of G. Furthermore let the reduced order model (14) be obtained from the original model (13) by the partition π. Also let $H(s)$ and $\hat{H}(s)$ be the transfer matrices from u to y in (13) and (14) respectively. Then the normalized model reduction error between the original and the reduced order model is given by:*

$$\frac{\|H(s) - \hat{H}(s)\|_2^2}{\|H(s)\|_2^2} = \frac{\sum_{i=1}^{m}(1 - \frac{1}{|C_{k_i}|})}{m(1 - \frac{1}{n})},$$

where n is the total number of vertices of G and where k_i is an integer such that $v_i \in C_{k_i}$ for each $i \in \{1, 2, \ldots, m\}$.

Proof. For the proof of this theorem we refer to [1].

□

4 Example

In this section we will discuss an example on the normalized reduction error between the original and the reduced order model in continuous time.

First we need some nomenclature.

A vertex i is called *pendant* if the degree of i is equal to one. A *path graph* $G_P = (V_P, E_P)$ is an undirected graph with $V_P = \{1, 2, \dots, n\}$ and E a set of unordered pairs $\{i, j\}$ with $i, j \in V$, for which exactly two vertices are pendant vertices and the rest of the vertices have a degree equal to two.

Consider two path graphs $G_{P_1} = (V_{P_1}, E_{P_1})$ and $G_{P_2} = (V_{P_2}, E_{P_2})$, where $V_{P_1} = \{1, 2, 3, 4, 5\}$ with vertices 1 and 5 being the pendant vertices and $V_{P_2} = \{1, 2, 3, 4, 5, 6\}$ with vertices 1 and 6 being the pendant vertices. Furthermore for both graphs, let $V_L = \{1\}$.

The graph G_{P_1} is depicted in Figure 2 and the graph G_{P_2} is depicted in Figure 3.

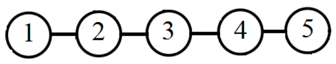


Figure 2: Graph G_{P_1}

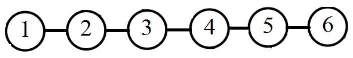


Figure 3: Graph G_{P_2}

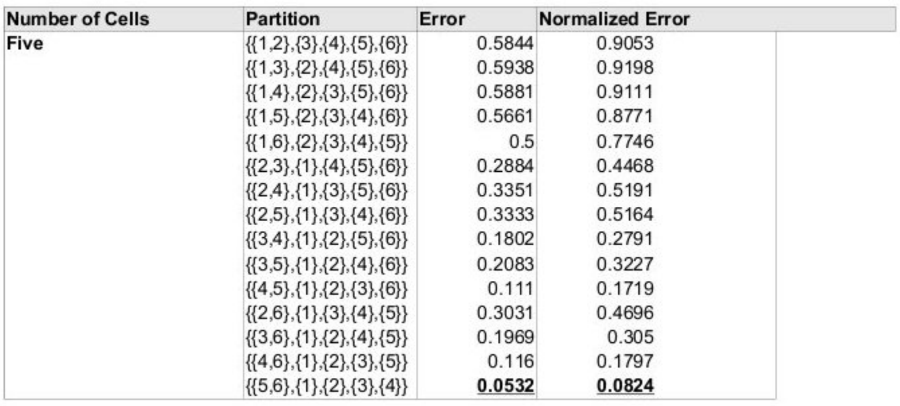
For these graphs we want to know which partition gives the least model reduction error within the set of all partitions with the same number of cells. In these cases however the partitions are not almost equitable partitions. So we can not use the theory as stated in subsection 3.3. Therefore we wrote a script in matlab to calculate the errors and normalized errors for a partition with a certain number of cells. An example of the script for a certain partition is given in appendix A.

In Table 1 and Table 2 the error and the normalized error for all possible partitions for a certain number of cells are given. The partitions resulting in the least error bounds for graphs G_{P_1} and G_{P_2} are given in Table 3.

Table 1

Number of Cells	**Partition**	**Error**	**Normalized Error**
Two	{{1,2},{3,4,5}}	0.6165	0.9748
	{{1,3},{2,4,5}}	0.5981	0.9456
	{{1,4},{2,3,5}}	0.5706	0.9022
	{{1,5},{2,3,4}}	0.5102	0.8068
	{{2,3},{1,4,5}}	0.5349	0.8458
	{{2,4},{1,3,5}}	0.5985	0.9463
	{{2,5},{1,3,4}}	0.5875	0.9289
	{{3,4},{1,2,5}}	0.6424	1.0158
	{{3,5},{1,2,4}}	0.6608	1.0449
	{{4,5},{1,2,3}}	0.6488	1.0259
	{{1},{2,3,4,5}}	**0.3345**	**0.5288**
	{{2},{1,3,4,5}}	0.579	0.9155
	{{3},{1,2,4,5}}	0.6356	1.0049
	{{4},{1,2,3,5}}	0.6481	1.0247
	{{5},{1,2,3,4}}	0.6472	1.0233
Three	{{1,2},{3,4},{5}}	0.6115	0.9669
	{{1,2},{3,5},{4}}	0.6119	0.9674
	{{1,2},{4,5},{3}}	0.5892	0.9316
	{{1,3},{2,4},{5}}	0.5985	0.9463
	{{1,3},{2,5},{4}}	0.5981	0.9456
	{{1,3},{4,5},{2}}	0.5992	0.9474
	{{1,4},{2,3},{5}}	0.5706	0.9022
	{{1,4},{2,5},{3}}	0.5706	0.9022
	{{1,4},{3,5},{2}}	0.5706	0.9022
	{{1,5},{2,3},{4}}	0.5009	0.7919
	{{1,5},{2,4},{3}}	0.5	0.7906
	{{1,5},{3,4},{2}}	0.5127	0.8107
	{{2,3},{4,5},{1}}	0.2967	0.4691
	{{2,4},{3,5},{1}}	0.3154	0.4987
	{{2,5},{3,4},{1}}	0.2907	0.4596
	{{1},{234},{5}}	0.3322	0.5252
	{{1},{235},{4}}	0.331	0.5234
	{{1},{245},{3}}	0.3168	0.5009
	{{1},{345},{2}}	**0.1802**	**0.2849**
	{{2},{134},{5}}	0.5936	0.9386
	{{2},{135},{4}}	0.5985	0.9463
	{{2},{145},{3}}	0.5349	0.8458
	{{3},{124},{5}}	0.6634	1.0489
	{{3},{125},{4}}	0.6424	1.0158
	{{4},{123},{5}}	0.6411	1.0136
Four	{{1,2},{3},{4},{5}}	0.5838	0.923
	{{1,3},{2},{4},{5}}	0.5898	0.9326
	{{1,4},{2},{3},{5}}	0.5706	0.9022
	{{1,5},{2},{3},{4}}	0.5	0.7906
	{{2,3},{1},{4},{5}}	0.2832	0.4477
	{{2,4},{1},{3},{5}}	0.3162	0.5
	{{2,5},{1},{3},{4}}	0.2907	0.4596
	{{3,4},{1},{2},{5}}	0.163	0.2577
	{{3,5},{1},{2},{4}}	0.1681	0.2658
	{{4,5},{1},{2},{3}}	**0.0763**	**0.1207**

Table 2



Number of Cells	Partition	Error	Normalized Error
Five	{{1,2},{3},{4},{5},{6}}	0.5844	0.9053
	{{1,3},{2},{4},{5},{6}}	0.5938	0.9198
	{{1,4},{2},{3},{5},{6}}	0.5881	0.9111
	{{1,5},{2},{3},{4},{6}}	0.5661	0.8771
	{{1,6},{2},{3},{4},{5}}	0.5	0.7746
	{{2,3},{1},{4},{5},{6}}	0.2884	0.4468
	{{2,4},{1},{3},{5},{6}}	0.3351	0.5191
	{{2,5},{1},{3},{4},{6}}	0.3333	0.5164
	{{3,4},{1},{2},{5},{6}}	0.1802	0.2791
	{{3,5},{1},{2},{4},{6}}	0.2083	0.3227
	{{4,5},{1},{2},{3},{6}}	0.111	0.1719
	{{2,6},{1},{3},{4},{5}}	0.3031	0.4696
	{{3,6},{1},{2},{4},{5}}	0.1969	0.305
	{{4,6},{1},{2},{3},{5}}	0.116	0.1797
	{{5,6},{1},{2},{3},{4}}	**0.0532**	**0.0824**

Table 3: Least model reduction error per number of cells.

Graph	Number of Cells	Partition	Error	Normalized Error
G_{P_1}	Two	$\{\{1\},\{2,3,4,5\}\}$	0.3345	0.5288
G_{P_1}	Three	$\{\{1\},\{3,4,5\},\{2\}\}$	0.1802	0.2849
G_{P_1}	Four	$\{\{1\},\{4,5\},\{2\},\{3\}\}$	0.0763	0.1207
G_{P_2}	Five	$\{\{1\},\{5,6\},\{2\},\{3\},\{4\}\}$	0.0532	0.0824

The results given in Table 3 give us a heuristic idea on how to choose the partition which gives the least model reduction error within the set of all partitions with the same number of cells. In particular, we recommend that clustering of the vertices is done based on the distance that each vertex has to the leader. We state this idea more formally next.

For a path graph $G_P = (V_P, E_P)$ let π_r^* denote the partition which gives the least model reduction error within the set of all partitions with r cells. Then we have the following conjecture for a path graph with n vertices.

Conjecture 4.1. *Let $G_P = (V_P, E_P)$ be a path graph where $V_P = \{1, 2, \ldots, n\}$ with vertices 1 and n being the pendant vertices. Suppose that $V_L = \{1\}$. Then, for each $2 \leqslant r \leqslant n$, π_r^* is given by*

$$\pi_r^* = \{1\}, \{2\}, \ldots, \{r-1\}, \{r, r+1, \ldots, n\},$$

and obviously $\pi_1^* = \boldsymbol{V}_P$.

5 Method for discrete time

For discrete time multi-agent systems a method similar to continuous time multi-agent systems can be developed. However we will give a more extensive explanation on the discrete time method in this section.

5.1 Petrov- Galerkin projections

As in the continuous time case, in this section we will describe the general theory behind Petrov-Galerkin projections.

Let a general input-state-output system be given by:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{15}$$

with $x(t) \in \mathbb{R}^n$ the state, $u(t) \in \mathbb{R}^m$ the input and $y(t) \in \mathbb{R}^p$ the output of the system.

Furthermore let $\mathbf{V}$ and $\mathbf{W} \in R^{n\times r}$ such that $\mathbf{W}^T\mathbf{V} = I$. A reduced order model can be obtained by using the projection $\Gamma = \mathbf{V}\mathbf{W}^T$ in the following way: first substitute $x(t+1)$ by $\mathbf{V}\hat{x}(t+1)$ in (15) and then premultiply the first line with $\mathbf{W}^T$.

Summarizing, we obtain the following system:

$$\begin{aligned} \hat{x}(t+1) &= \mathbf{W}^T A\mathbf{V}\hat{x}(t) + \mathbf{W}^T Bu(t), \\ y(t) &= C\mathbf{V}\hat{x}(t), \end{aligned} \tag{16}$$

in which $\hat{x} \in \mathbb{R}^r$ is the state of the reduced order system.

This projection is called a Petrov-Galerkin projection. If the matrix $\mathbf{W}$ is equal to the matrix $\mathbf{V}$, the projection Γ becomes orthogonal and is called a Galerkin projection. The matrices $\mathbf{W}$ and $\mathbf{V}$ can be chosen differently, depending on the application, to preserve certain properties. A disadvantage is however that a direct application of a Petrov-Galerkin projection will destroy the spatial structure of the network. We propose a method to handle this in the following section.

For more information on Petrov-Galerkin projections we refer to [4].

5.2 Projection by graph-partitions

In this section we will develop a reduction method that is analogous to the one that was developed for the continuous time case. The basic idea is again to apply the Petrov-Galerkin projection, as discussed in section 5.1, to the system (5). The general background of graph-partitions, definitions and nomenclature is the same as stated in subsection 3.2. For example the characteristic vector of a cell will again be given by (9).

If we apply the Petrov-Galerkin projection to the system (5), the system becomes:

$$\hat{x}(t+1) = \mathbf{W}^T Q \mathbf{V} \hat{x}(t) + \mathbf{W}^T M u(t). \tag{17}$$

As said before, a disadvantage of the Petrov-Galerkin projection is that it will destroy the spatial structure of the network. In general the matrix $\mathbf{W}^T Q \mathbf{V}$ in the representation of the reduced order system will not be structured, and so the reduced order system cannot be written in the form (5). By using graph partitions the structure of the network will be preserved. This is what we will exploit.

Let us now return to the system (17), with associated graph $G = (V, E)$, $\pi = \{C_1, C_2, \ldots, C_r\}$ a partition of V, and $P(\pi)$ the characteristic matrix of π. Choosing $\mathbf{W}^T$ and $\mathbf{V}$ in the following way :

$$\begin{aligned} \mathbf{W}^T &= (P^T(\pi)P(\pi))^{-1}P^T(\pi), \\ \mathbf{V} &= P(\pi), \end{aligned} \tag{18}$$

yields $\mathbf{W}^T\mathbf{V} = I$. The columns of $P(\pi)$ are orthogonal, $P^T(\pi)P(\pi)$ is a diagonal matrix with strictly positive diagonal elements and is therefore invertible. If we write $P(\pi)$ as P then (17) can be written as:

$$\hat{x}(t+1) = (P^TP)^{-1}P^TQP\hat{x}(t) + (P^TP)^{-1}P^TMu(t). \tag{19}$$

By defining new matrices $\hat{Q}$ and $\hat{M}$ as:

$$\begin{aligned} \hat{Q} &= (P^TP)^{-1}P^TQP \\ \hat{M} &= (P^TP)^{-1}P^TM \end{aligned} \tag{20}$$

this simplifies to

$$\hat{x}(t+1) = \hat{Q}\hat{x}(t) + \hat{M}u(t). \tag{21}$$

Since P contains only zeros and ones, the structures of $\hat{Q}$ and $\hat{M}$ are similar to the structures of Q and M respectively, but the input signals are now weighted. By this, each cell of the partition π of the graph G is mapped to a vertex of $\hat{G}$, which means that the number of cells in π is equal to the number of vertices in $\hat{G}$. There is an edge between two vertices in $\hat{G}$ if and only if this edge was already there between two vertices of the corresponding different cells in G. Stated mathematically: there is an edge between vertices a and b ($a \neq b$) in $\hat{G}$ if and only if there exists a vertex $i \in C_a$ and a vertex $j \in C_b$ such that $(i, j) \in E$. Therefore if we choose $(i, j) \in \hat{E}$ then also $(j, i) \in \hat{E}$, so $\hat{G}$ is a directed graph which is symmetric.

Between the adjacency matrices $\hat{A}$ of $\hat{G}$ and A of G there exists the following relationship:

$$\begin{aligned} A &= [a_{ij}] \\ \hat{A} &= [\hat{a}_{ab}], \text{with } \hat{a}_{ab} = \tfrac{1}{|C_a|} \sum_{i \in C_a,\ j \in C_b} a_{ij}, \end{aligned} \tag{22}$$

with $|C_a|$ the cardinality (the number of elements) of the set C_a.

The matrix $\hat{Q}$ is not necessarily symmetric (the number of vertices may differ from cell to cell in π) but is similar to the symmetric matrix $(P^T P)^{-\frac{1}{2}} P^T Q P (P^T P)^{\frac{1}{2}}$. Thus $\hat{Q}$ inherits properties of Q like diagonalizability and having real eigenvalues.

Overall, this method describes a way to cluster some vertices and map them to a single vertex. By this we reduce the order of the system and the reduced order model becomes associated with a new multi-agent system based on the associated reduced graph $\hat{G}$. Furthermore, the reduced state $\hat{x}$ approximates the average of the states of the agents that are in the same cell of π. The approximation becomes exact by using almost equitable partitions in which agents with similar connections to the rest of the network are placed within the same cell.

As said before, $\hat{Q}$ will inherit certain properties of Q (diagonalizability and having real eigenvalues). However, these are not the only useful properties. In the following section we will discuss what happens with another useful property, namely consensus.

5.3 Consensus preservation for the reduced order model

In this section we will see if the property that consensus has been reached in the original system is retained in the reduced system.
Consider a network of n agents given by (5). In discrete time each agent updates its state according to the weighted average of the states

of its neighbor agents. The most common consensus algorithm is defined without external input so the system becomes:

$$x(t+1) = Qx(t). \tag{23}$$

We say that the multi-agent system (23) reaches consensus if $x_i(t) - x_j(t)$ converges to 0 as t goes to infinity, for every $i, j \in \{1, 2, \ldots n\}$.

Now we want to prove that if the system (23) reaches consensus, then also the reduced order system $\hat{x}(t+1) = \hat{Q}\hat{x}(t)$ reaches consensus. We first introduce some background in order to give this proof.

Definition 16: Interlacing eigenvalues
Let A and B be real symmetric matrices in $\mathbb{R}^{n\times n}$ and $\mathbb{R}^{m\times m}$ respectively, with $m \leqslant n$. Furthermore let the eigenvalues of A be denoted by $\lambda_{A_1}, \ldots, \lambda_{A_n}$ in an increasing order and the eigenvalues of B by $\lambda_{B_1}, \ldots, \lambda_{B_m}$ in an increasing order. We say that the eigenvalues of B interlace the eigenvalues of A if:

$$\lambda_{A_i} \leqslant \lambda_{B_i} \leqslant \lambda_{A_{n-m+i}} \text{ for each } i = 1, 2, \ldots, m.$$

Theorem 5.1. *Let A be a real symmetric $n \times n$ matrix and let H be an $n \times m$ matrix such that $H^T H = I$. Set B equal to $H^T AH$ and let $v_1, \ldots, v_m$ be an orthogonal set of eigenvectors of B such that $Bv_i = \lambda_{B_i} v_i$. Then the eigenvalues of B interlace the eigenvalues of A.*

Proof. For the proof of this theorem we refer to [2], page 203, theorem 9.5.1 a. □

Theorem 5.2. *Let $Q = [q_{ij}] \in \mathbb{R}^{nxn}$, with $q_{ij} \geq 0$, be a stochastic and symmetric matrix, and let $\hat{Q}$ be given by (20) for a given partition π. Then the eigenvalues of $\hat{Q}$ interlace the eigenvalues of Q.*

Proof. As seen before $P^T P$ is a diagonal matrix with strictly positive diagonal elements. Recall that $\hat{Q}$ is similar to the symmetric matrix $(P^T P)^{-\frac{1}{2}} P^T Q P (P^T P)^{\frac{1}{2}}$ (section 5.2). Choosing $F = P(P^T P)^{-\frac{1}{2}}$ then $F^T Q F = (P^T P)^{-\frac{1}{2}} P^T Q P (P^T P)^{\frac{1}{2}}$, so $\hat{Q}$ is similar to the symmetric matrix $F^T Q F$, for $F = P(P^T P)^{-\frac{1}{2}}$. Furthermore $F^T F = I$, so by theorem 5.1 we obtain that the eigenvalues of $\hat{Q}$ interlace the eigenvalues of Q. □

Theorem 5.3. *Let Q be a stochastic matrix. Then 1 is an eigenvalue of Q. Furthermore 1 is a simple eigenvalue of Q if and only if its weighted directed associated graph $G(Q)$, has a directed spanning tree. Moreover if $G(Q)$ has a directed spanning tree and $q_{ii} > 0$ for $i = 1, 2, \dots, n$, then 1 is the unique eigenvalue of maximum modulus.*

Proof. For the proof of this theorem we refer to [9] and [8]. $\square$

Theorem 5.4. *The discrete time multi-agent system (23) achieves consensus if and only if the weighted directed associated graph $G(Q)$ has a directed spanning tree.*

Proof. For the proof of this theorem we refer to [9] and [8]. $\square$

With these theorems we are now able to prove that if the system (23) reaches consensus, then also the reduced order system $\hat{x}(t+1) = \hat{Q}\hat{x}(t)$ reaches consensus.

Theorem 5.5. *If the multi-agent system $x(t+1) = Qx(t)$ reaches consensus, then also the reduced order system $\hat{x}(t+1) = \hat{Q}\hat{x}(t)$ reaches consensus.*

Proof. Since $x(t+1) = Qx(t)$ reaches consensus, by theorem 5.4 the associated weighted directed graph $G(Q)$ has a directed spanning tree. Therefore, by theorem 5.3, 1 is a simple eigenvalue of Q. Moreover since q_{ij} is the weight agent i assigns to agent j when agent i updates its state, q_{ij} is larger than 0, so 1 is the unique eigenvalue of Q with maximum modulus. By theorem 5.2 and since $\hat{q_{ii}} > 0$ for $i = 1, 2, \dots, n$, 1 is also an unique eigenvalue of $\hat{Q}$. Theorem 5.3 then implies that the associated weighted directed graph $G(\hat{Q})$ has a directed spanning tree. Now applying theorem 5.4 implies that $\hat{x}(t+1) = \hat{Q}\hat{x}(t)$ reaches consensus, which had to be proved. $\square$

Remark 5.2
The rate of convergence of a multi-agent system depends on the eigenvalues of the associated matrix. The interlacing property holds for the reduced order system, therefore every eigenvalue of $\hat{Q}$ will be smaller or equal to an eigenvalue of Q and the biggest eigenvalue of $\hat{Q}$ will be smaller or equal to the biggest eigenvalue of Q. Consequently the rate of converge in the reduced order model is at least as fast as that of the original model.

So far we have seen that a reduced order model can be obtained by applying an appropriate projection to the original multi-agent system defined on an associated graph G. This reduced order model can be modeled as a multi-agent system defined on a new associated graph $\hat{G}$.

Furthermore $\hat{Q}$ will inherit certain properties of Q (diagonalizability and having real eigenvalues), and consensus and the convergence rate are preserved by the model reduction. In the next section we will discuss appropriate choices of partitions such that the input-output behavior of the reduced order model remains 'close' to the input-output behavior of the original model.

5.4 Input-Output approximation of multi-agent systems

In this section we will discuss appropriate choices of partitions such that the input-output behavior of the reduced order model is 'close' to the input-output behavior of the original model.

So far, in discrete time we have only looked at multi-agent systems with inputs and states. We will now also look at multi-agent systems with outputs. In order to do this we first assume that the associated graph G is connected. If G is not connected the proposed model reduction technique can be applied to the disconnected components of G individually. Furthermore, in the context of distributed control, the differences of the states of the agents play a crucial role in consensus. We therefore want the incidence matrix R (in which the differences of the states of the agents are embedded), to be present in the output variables. So let us choose the output as $y(t) = W^{\frac{1}{2}}R^T x(t)$, where R is defined by (2) and W is defined by (3).

Since G is connected, reaching consensus for the multi-agent system (5) means that $y(t)$ converges to zero as t goes to infinity for all initial states of the multi-agent system.

The following input-state-output representation is now obtained for the original multi-agent system:

$$\begin{aligned} x(t+1) &= Qx(t) + Mu(t), \\ y(t) &= W^{\frac{1}{2}}R^T x(t), \end{aligned} \tag{24}$$

where x, M, R and W are as defined before. Furthermore to satisfy that $\sum_{j=1}^{n} q_{ij} = 1, q_{ij} \geq 0$ and $q_{ii} > 0$ for $i \in \{1, 2, \ldots, n\}$ (see Definition 11), we choose $Q = I - \epsilon L$. Here I is the identity matrix, $L = [l_{ij}]$ is the Laplacian matrix given by formula (1) and ϵ is a parameter $0 < \epsilon < \frac{1}{max\{l_{ii}\}}$.

We now choose π again to be a partition of G. The input-state-output model for the reduced order multi-agent system is given by:

$$\begin{aligned} \hat{x}(t+1) &= \hat{Q}\hat{x}(t) + \hat{M}u(t), \\ y(t) &= W^{\frac{1}{2}}\hat{R}^T\hat{x}(t), \end{aligned} \tag{25}$$

where $\hat{Q}, \hat{M}$ are given by formula (20), $\hat{x} \in \mathbb{R}^k$ with $k \leq n$, W is given by formula (3), and $\hat{R}^T = R^T P$.

Just as with the continuous case we want the behavior of the original multi-agent system to be approximated as efficient as possible. So we have to choose an appropriate partition. The finest and the coarsest approximation are again given by the two trivial partitions. We want to find a compromise between the order of the reduced model and the accuracy of the approximation.

For this we again need almost equitable partitions (as explained in section 3.3, defined by Definition 14 and Definition 15). The key property of an almost equitable partition is that $\text{im}P(\pi)$ is L-invariant. This will be stated in the following theorem.

Theorem 5.6. *Let π be a partition of a weighted undirected graph G and let L denote the Laplacian matrix of G. Then π is an almost equitable partition if and only if $\text{im}P(\pi)$ is L-invariant, i.e.*

$$L\ \text{im}P(\pi) \subseteq \text{im}P(\pi).$$

Proof. For the proof of this theorem we refer to [1]. □

Remark 5.3
For the discrete time method we work with $Q = I - \epsilon L$. So we want that a subspace is Q-invariant if the subspace is L-invariant, i.e. if $L\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$ then also $Q\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$. Note that since $L\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$, also $-\epsilon L\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$ and furthermore $I\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$. Hereby $Q\ \text{im}P(\pi) \subseteq \text{im}P(\pi)$.

We now want to find the normalized model reduction error between the original and the reduced order model $\frac{\|H(s)-\hat{H}(s)\|_2^2}{\|H(s)\|_2^2}$. For this, we try to mimic the steps given in the proof of [1], theorem 6.

We first construct a matrix $T = [P\ F]$, where $P = P(\pi)$ and
$F \in \mathbb{R}^{n\times(n-k)}$ such that the columns of T are orthogonal.
Then $P^T F = 0$.

Now we apply the state space transformation $x(t) = T\tilde{x}(t)$ to (24). This yields

$$x(t+1) = Qx(t) + Mu(t) = QT\tilde{x}(t) + Mu(t) = T\tilde{x}(t+1).$$

So

$$\tilde{x}(t+1) = T^{-1}QT\tilde{x}(t) + T^{-1}Mu(t).$$

Since $T^{-1} = [(P^TP)^{-1}P^T \ (F^TF)^{-1}F^T]^T$, the following input-state-output system is obtained:

$$\begin{gathered}\tilde{x}(t+1) = \begin{bmatrix}(P^TP)^{-1}P^TQP & (P^TP)^{-1}P^TQF\\ (F^TF)^{-1}F^TQP & (F^TF)^{-1}F^TQF\end{bmatrix}\tilde{x}(t) + \begin{bmatrix}(P^TP)^{-1}P^TM\\ (F^TF)^{-1}F^TM\end{bmatrix}u(t)\\ y(t) = \begin{bmatrix}W^{\frac{1}{2}}R^TP & W^{\frac{1}{2}}R^TF\end{bmatrix}\tilde{x}(t)\end{gathered} \tag{26}$$

The transfer matrices of system (24) and system (26) are of course identical. Also, by truncating the second state component of $\tilde{x}$ the reduced order model is obtained.

Since π is an almost equitable partition of G, im$P$ is L-invariant and, as explained in Remark 4, im$P$ is also Q-invariant. Therefore there exists a matrix X such that $QP = PX$. Hence we obtain $F^TQP = F^TPX = (P^TF)^TX = 0$ and $P^TQF = (F^TQP)^T = 0$.

In this way system (26) can be written as:

$$\begin{gathered}\tilde{x}(t+1) = \begin{bmatrix}(P^TP)^{-1}P^TQP & 0\\ 0 & (F^TF)^{-1}F^TQF\end{bmatrix}\tilde{x}(t) + \begin{bmatrix}(P^TP)^{-1}P^TM\\ (F^TF)^{-1}F^TM\end{bmatrix}u(t)\\ y(t) = \begin{bmatrix}W^{\frac{1}{2}}R^TP & W^{\frac{1}{2}}R^TF\end{bmatrix}\tilde{x}(t).\end{gathered} \tag{27}$$

The transfer matrix of (27), called $H(z)$, (and therefore also the transfer matrix of the original system (24)) is obtained by the following relation:

$$H(z) = \hat{H}(z) + \Delta(z),$$

with $\hat{H}(z) = W^{\frac{1}{2}}R^TP(zI - (P^TP)^{-1}P^TQP)(P^TP)^{-1}P^TM$ the transfer matrix of the reduced order system (25),
and $\Delta(z) = W^{\frac{1}{2}}R^TF(zI - (F^TF)^{-1}F^TQF)(F^TF)^{-1}F^TM$ the error between the transfer matrices $H(z)$ and $\hat{H}(z)$.

Also the following relation holds: $\hat{H}(z)^T(\bar{z})\Delta(z) = 0$, since $P^TLF = 0$. Therefore

$$\|H(z)\|_2^2 = \|\hat{H}(z)\|_2^2 + \|\Delta(z)\|_2^2.$$

We now want to find the error

$$\|\Delta(z)\|_2^2 = \|H(z) - \hat{H}(z)\|_2^2 = \|H(z)\|_2^2 - \|\hat{H}(z)\|_2^2. \tag{28}$$

For this we first calculate $\|H(z)\|_2^2$ and $\|\hat{H}(z)\|_2^2$.

In general for a discrete time system of the form:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{29}$$

the finite two-norm of the transfer matrix $\| H \|_2^2$ is given by:

$$\| H \|_2^2 = \text{trace}(B^T S B), \tag{30}$$

where $S = \sum_{t=0}^{\infty} (A^T)^t C^T C A^t$ (S is not equal to the null-matrix) is a solution to the discrete Lyapunov equation

$$A^T S A - S + C^T C = 0. \tag{31}$$

So $\|H(z)\|_2^2 = \text{trace}(M^T \sum\limits_{t=0}^{\infty} (Q^T)^t L Q^t M)$,
and $\|\hat{H}(z)\|_2^2 = \text{trace}(\hat{M}^T \sum\limits_{t=0}^{\infty} (\hat{Q}^T)^t P^T L P \hat{Q}^t \hat{M})$.

Now let the matrices $\chi \in \mathbb{R}^{n\times n}$ and $\psi \in \mathbb{R}^{r\times r}$ be defined by:

$$\chi = \sum_{t=0}^{\infty} (Q^T)^t L Q^t$$

$$\psi = \sum_{t=0}^{\infty} (\hat{Q}^T)^t P^T L P \hat{Q}^t.$$

We want to find a relation between χ and ψ.
First of all χ can be written as $\chi = \sum\limits_{t=0}^{\infty} L Q^{2t}$ since Q is symmetric and $LQ^t = Q^t L$. Some further calculations give $\chi = L \lim\limits_{N\to\infty} \sum\limits_{t=0}^{N} Q^{2t}$.

Now we want to use the general rule:

$$\sum_{t=0}^{N} A^t = (I - A)^{-1}(I - A^{N+1}). \tag{32}$$

If we substitute $A = Q^2$ we obtain the following:

$$\sum_{t=0}^{N} Q^{2t} = (I - Q)^{-1}(I - Q^{2N+2}).$$

However $(I - Q)^{-1}$ does not exist since Q^2 has an eigenvalue equal to 1 (in order to reach consensus) so $(I - Q)$ has an eigenvalue equal to zero, which makes it not invertible. Even if $(I - Q)^{-1}$ would exist, we could not apply the same trick to ψ since $\hat{Q}$ is not symmetric and $P^T LP\hat{Q}^t \neq \hat{Q}^t P^T LP$, so the general rule (32), can not be used to rewrite and calculate χ and ψ or find a relation between them.

So what goes wrong here what does not go wrong for continuous time? In the continuous time case we also have

$$\|\Delta(z)\|_2^2 = \|H(z) - \hat{H}(z)\|_2^2 = \|H(z)\|_2^2 - \|\hat{H}(z)\|_2^2.$$

There however $\|H(z)\|_2^2 = \text{trace}(M^T \int_0^\infty e^{-Lt}Le^{-Lt}dt\ M)$,
and $\|\hat{H}(z)\|_2^2 = \text{trace}(\hat{M}^T \int_0^\infty e^{-\hat{L}^T t}P^T LPe^{-\hat{L}t}dt\ \hat{M})$.

The integral $\int_0^\infty e^{-Lt}Le^{-Lt}dt$ can be computed since Q is symmetric and furthermore

$$P^T \int_0^\infty e^{-Lt}Le^{-Lt}dtP = \int_0^\infty e^{-\hat{L}^T t}P^T LPe^{-\hat{L}t}dt.$$

This can be obtained by the properties of e^{-Lt}.
So let us try to use the same method for the discrete time case. If we set $Q = e^{-A}$, we obtain the following matrices χ and ψ:

$$\chi = \sum_{t=0}^{\infty} e^{-At}Le^{-At}$$

$$\psi = \sum_{t=0}^{\infty} (\hat{Q}^T)^t P^T LP\hat{Q}^t,$$

with $\hat{Q} = (P^T P)^{-1}P^T e^{-AP}$.

Now we get the following:

$$P^T \chi P = P^T \sum_{t=0}^{\infty} e^{-At} L e^{-At} P$$

$$= \sum_{t=0}^{\infty} P^T e^{-At} L e^{-At} P$$

$$= \sum_{t=0}^{\infty} e^{-((P^TP)^{-1}P^TAP)^T t} P^T L P e^{-(P^TP)^{-1}P^TAPt}.$$

To let $P^T \chi P = \psi$, this would mean that $\hat{Q}^t = ((P^TP)^{-1}P^Te^{-AP})^t$ should be equal to $e^{-(P^TP)^{-1}P^TAPt}$. But that is not true. So we are not able to compute χ or ψ and it is also not possible to derive $P^T \chi P = \psi$.

Another thing we could think of is choosing other input variables. So that, for example, $\|H(z)\|_2^2$ and $\|\hat{H}(z)\|_2^2$ become:

$\|H(z)\|_2^2 = \text{trace}(M^T \sum_{t=0}^{\infty} (Q^T)^t (I - Q) Q^t M)$ and
$\|\hat{H}(z)\|_2^2 = \text{trace}(\hat{M}^T \sum_{t=0}^{\infty} (\hat{Q}^T)^t P^T (I - Q) P \hat{Q}^t \hat{M})$.

The problems will however remain the same. We are still not able to compute χ or ψ and it is also not possible to derive $P^T \chi P = \psi$. This leaves this part very interesting for future research.

6 Conclusion

In this report we discussed a method to reduce the complexity of multi-agent systems in discrete time.
In order to do this, we first discussed some general background on systems in the second section. Secondly the method for continuous time multi-agent systems was discussed in the third section. This section contained three subsections; Petrov-Galerkin projections, Projection by graph-partitions and Input-Output approximation of multi-agents systems.

In the fourth section we discussed an example on the normalized reduction error between the original and the reduced order model in continuous time. The example described a path graph of n vertices with vertices 1 and n being the pendant vertices and the first vertex being the leader. For this example we noticed that the best results were reached by clustering the vertices that have the longest distance to the leader. Therefore we recommended clustering to be done on distance to the leader. Furthermore we gave a conjecture for a general path graph.

In the fifth section we described the method for discrete time, this section contained four subsections; theory behind Petrov-Galerkin projections was discussed in the first subsection and the Petrov-Galerkin projection applied to the general system was discussed in the second subsection. One of the most important properties of the multi-agent system, the consensus preservation for the reduced order system, was discussed in the third subsection and input-output approximations of multi-agent systems were discussed in the fourth subsection. In the last subsection we faced some problems which are very interesting for future research.

References

[1] *A projection based approximation of multi-agent systems by using graph partitions*, Trentelman H.L., Monshizadeh N., 2013.

[2] *Algebraic Graph Theory*, Royle G., Godsil C., Springer-Verlag, New-York, 2001.

[3] *Approximation of Large-Scale Dynamical Systems*, Antoulas A.C., SIAM Advances in Design and Control, 2005.

[4] *Computational Methods of Science (lecture notes)*, Wubs F.W., 2012.

[5] *Consensus and Cooperation in Networked Multi-Agent Systems*, Olfati-Saber R., Murray R.M., Fax J.A., Proceedings of the IEEE, IEEE Transactions on Automatic Control, Vol. 95, No. 1, 2007.

[6] *Consensus Problems in Networks of Agents With Switching Topology and Time-Delays*, Olfati-Saber R., Murray R.M., IEEE Transactions on Automatic Control, Vol. 49, No. 9, 2004.

[7] *Consensus seeking in multiagent systems under dynamically changing interaction topologies*, Ren W., Beard R.W., IEEE Transactions on Automatic Control, 50:655-661, 2005.

[8] *Distributed Algorithms for Interacting Autonomous Agents*, Xia W., thesis, 2013.

[9] *Distributed consensus in Multi-Vehicle Cooperative Control*, Ren W., Beard R.W., Springer-Verlag, London, 2008.

[10] *Graph Theoretic Methods in Multiagents Networks*, Mesbahi M., Egerstedt M., Princeton University Press, 2010.

[11] *Information Consensus in Multivehicle Cooperative Control*, Ren W., Beard R.W., Atkins E.M., 2007, IEEE control systems magazine.

[12] *Information Consensus of Asynchronous Discrete-time Multi-agent Systems*, Fang L., Antsaklis P.J., American Control Conference, 2005.

[13] *Laplacian eigenvectors and eigenvalues and almost equitable partitions*, Cardoso D.M., Delorme C., Rama P., Elsevier, 2005.

[14] *Matrix Analysis*, Horn R.A., Johnson C.R., Cambridge University Press, Cambridge, 1985.

[15] *Robust Synchronization of Uncertain Linear Multi-Agent Systems*, Trentelman H.L., Takaba K., Monshizadeh N., 2012.

A Script Example

```
% Defining all given matrices

M = [ 1; 0; 0; 0; 0 ];
L = [1 -1 0 0 0; -1 2 -1 0 0; 0 -1 2 -1 0; 0 0 -1 2 -1; 0 0 0 -1 1];
R = [ 1 0 0 0; -1 1 0 0; 0 -1 1 0; 0 0 -1 1; 0 0 0 -1];
Rtra= transpose(R);
P = [ 1 0 ; 1 0; 0 1; 0 1; 0 1];
Ptra = transpose(P);
Pinv = inv(Ptra * P);
Ppro = Pinv * Ptra;
Lhat = Ppro * L * P;
Mhat =  Ppro * M;
Rhat = Ptra * R;
Rhattra= transpose(Rhat);
D = [ 0; 0; 0; 0];

% Creating the transfermatrix for the original model

[bb,aa] = ss2tf(-L,M,Rtra,D);
sys1m=tf(bb(1,:),aa);
sys1=minreal(sys1m);
sys2m=tf(bb(2,:),aa);
sys2=minreal(sys2m);
sys3m=tf(bb(3,:),aa);
sys3=minreal(sys3m);
sys4m=tf(bb(4,:),aa);
sys4=minreal(sys4m);
Tf = [ sys1; sys2; sys3; sys4]

% Creating the transfermatrix for the reduced order model

[dd,cc] = ss2tf(-Lhat,Mhat,Rhattra,D);
sysr1m=tf(dd(1,:),cc);
sysr1=minreal(sysr1m);
sysr2m=tf(dd(2,:),cc);
sysr2=minreal(sysr2m);
sysr3m=tf(dd(3,:),cc);
sysr3=minreal(sysr3m);
sysr4m=tf(dd(4,:),cc);
sysr4=minreal(sysr4m);
Tfr = [ sysr1; sysr2; sysr3; sysr4]
```

```
% Calculating the errors

 Dif = minreal(Tf-Tfr)

 norm1 = norm(Tf,2);
 E = norm(Dif,2)
 Enor =  E/norm1
```