Model order reduction of CFFLs

Methods of model order reduction
for (families of) Coherent Feedforward Loops

J. Guldenaar (4274741)

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Model order reduction of CFFLs

Methods of model order reduction for (families of) Coherent Feedforward Loops

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

J. Guldenaar (4274741)

June 2, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
DCSC

The undersigned hereby certify that they have read and recommend to the Faculty of 3mE for acceptance a thesis entitled

Model order reduction of CFFLs

by

J. Guldenaar (4274741)

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: June 2, 2021

Supervisor(s): _____
dr. E. Steur

Reader(s): _____
dr. E. Steur

Abstract

Biochemical reactions play a crucial role and tell us many about the behaviour of the biological regulation processes . A lot of similar biochemical processes are discussed in several resources, but the methods and approaches that have been used mostly differ. To make a nice overview of the found methods we discuss the most important ones here. Besides we will apply several methods of order reduction to describe the overall dynamics in a more compact way.

For modelling a set of biochemical reactions is rewritten as first order differential equations. This set of first order differential equations defines the state space model and makes us able to analyse the overall system where the corresponding biochemical reactions are involved. This rewriting is based on mass-action kinetics and Michaelis-Menten (MM) theory. By looking at the left and right nullspaces of the so called Stoichiometry matrix the conservation laws and flux distribution in steady state can respectively be deduced. The systems we are looking at can be distinguished in many different biochemical regulatory networks .

Examples of these networks can be found in gene expression, protein production and/or hormone production. The system that includes all given biochemical reactions of the network can be seen as a Coherent Feedforward Loop or CFFL. In synthetic biology these CFFLs are studied to gain insight into the desired production/expression: think about medicine production, agriculture and manufacturing. In biochemistry mostly a set of biochemical reactions can be given as a family or combination of CFFLs. To realise this a so called AND-gate or toehold switch is used.

Most of the sets of biochemical reactions can be reduced in order since a few of these reactions are less relevant for the overall process or since the conservation laws define a plane within the higher dimensional space in which the solutions lie. Mostly the conservation laws can already be explained biologically with the given feedback mechanisms and cycles. Several conservation laws can be found with mathematical concepts of e.g. Ordinary Differential Equations and indeed these laws will imply the real biological conservation laws. There are several ways to reduce the order of the system or to reduce the number of reactions involved in the overall process. This depends on the structure of the biochemical regulatory network.

First it is shown that a system has conservation laws if the left null space of the Stoichiometry matrix is not empty. If it is non-empty these laws can thus be used to reduce the system in order equal to the dimension of this left nullspace.

Afterwards we have the option to reduce the system even more by applying the Quasi Steady State Approach in a given network like the CFFL. This method suggests that some species

concentrations will reach its steady states much sooner than other species concentrations (if we look at slow timescale). Therefore it is assumed that some species already have their steady state at the beginning of the experiment. This is the so called classical QSSA.

Another way to reduce the system order is by applying the Kron reduction order method. This method assumes a complexes network that reduces the complexes (rather seen as internal nodes) and thus the number of reactions for the given system. By rewriting the update equations or set of ordinary differential equations we have the option to reduce the internal dynamics in terms of complexes. The external dynamics will somewhat be influenced by doing this and we can compare the two models. Here the concept of complex balancedness will determine whether the steady states for both models will be the same.

Eventually another interesting aspect will be to have a look at the kinetics of the overall system. One can see this as an alternative or extra application for the Quasi Steady State Approach and here the cycles and feedback mechanisms will be replaced by more simple ones. Then afterwards mass-action kinetics along with classical QSSA can be applied. To get an optimal reduction order model the way in which parameters within the model are estimated can be discussed by optimization techniques. It is important to keep in mind that other kinetics like Michaelis Menten for the subsystems will work better as Alternative network.

Furthermore we will see how the system can be transformed if we also have to do with in- and outflows. It actually means that we will need to add an extra term . There are two ways to do this even though they will have a different interpretation. One will be in matrix-vector form while the other method merely uses vector-scalar notation. We will also look at the relation between these two forms.

Since the structures of the global biochemical regulatory network is rather complex it was advised to deal with the order reduction of the system first and to focus less on the control of these systems. In general, the biochemical regulatory system will find a steady state behaviour over time.

By comparing measured data with data from the simulation of the model we can get an idea of the reliability. The models itself can provide valuable information about the precise functioning of biological cells and system. Here we can consider modular biochemical circuits for creation of sensors and actuators. A future challenge would be to make an auto based system that directly converts the given system into its reduced order form. Here the best reduction order model will be selected automatically and applied in the best determined sequence.

Contents

Acknowledgements

The past three years during my master Systems and Control were challenging, but it gave me more insight in this field of Mechanical Engineering. With a rather mathematical background (I did my Bachelor in Industrial and Applied Mathematics) a few concepts were fully new to me and it took me some time to understand and master these. Nevertheless it was the exact reason why Systems and Control got my interest and motivation. Soon a connection between my Bachelor and Master was made and the topics became more familiar and inspiring to me. After all I've learned a lot during my study Systems and Control ...

Hereby I would like to thank my supervisor dr. E. Steur for his assistance during the writing of this thesis and the extra explanation of the articles he gave me during the meetings... Moreover I would like to thank my fellow students with whom Ive worked with and supported me during this master's degree ...

Index terms - Biochemical regulatory network, Model reduction method, Biochemisty, molecular noise filtering, robust biochemical circuit design.

> "I think if people are passionate about something, it could be real estate or biochemistry, and that spark gets turned on in them, everyone's beautiful in that zone."
>
> — *Cindy Crawford*

Delft, University of Technology
June 2, 2021

J. Guldenaar (4274741)

Chapter 1

Introduction

1-1 General Intro

In biology one deals with several issues involving e.g. the surplus/shortage of species in several complexes during reactions, hormone regulation and the principle of gene expression [1], [2]. Here many phenomena deal with combinations of multiple paths that need to be followed in order to produce an output protein, to turn on genes or to produce the corresponding hormones [3]. Therefore this is why this biological or biochemical process is called a combinatorial problem. To find solutions for this involved in biochemistry sometimes one wants to regulate or analyze the combinatorial and coherent process by adding the so called (DNA) triggers properly to activate the actual production/formation process as desired [3]. Synthetic biology is the field in which these phenomena and biochemical changes can solve these problems that have e.g. in turn to do with designing a medicine, manufacturing and agriculture [11].

1-2 System description

In this MSc Thesis we are particularly interested in the production of output proteins and how an indirect and direct path interact with each other for the so called Coherent Feedforward Loop (CFFL). Besides we will be interested in the model order reduction of the given system or CFFL network.

1-2-1 The Coherent Feedforward Loop (CFFL)

The CFFL within biochemical reaction networks/circuits describes how a sign-sensitive delay element (or noise-filtering element) functions during regulation of an output protein (e.g. Green Fluorescent Protein (eGFP)) [3]. It is defined to be a network motif as its appearance number is higher in the transcription networks of bacteria Escherichia Coli (E. Coli) and

Saccharomyces cerevisiae than in randomised networks. Even though there are more types of Coherent Feedforward Loops with component-wise differences a more common structural CFFL is used as found in [3]. In general one can make a distinction between a direct and indirect path from a so called sigma-factor that initiates DNA transcription to make a copy of the DNA strand into mRNA. The direct path consists of a DNA trigger that activates a toehold switch on one side. The indirect path consists of another sigma-factor which will either be necessary to activate the switch. If both the sigma-factor and DNA trigger activate the AND gate of the toehold switch the output protein eGFP will be formed. This is an RNA-based AND gate that requires the binding of the different RNA elements in order to start translation. The question that arises is how the AND gate needs to be activated in order to regulate the output protein concentration. Or more specifically written, one wants to know how much and in what time the species X should be added in order to form the desired amount of species Z (with indirect path that includes species Y). To illustrate this, we will give an example.

Let X describe the sigma-factor for the input. Let Y describe the other sigma-factor of the indirect path and let Z describe the formed output protein, then the direct and indirect path or system can schematically be depicted as in figure (1-1).

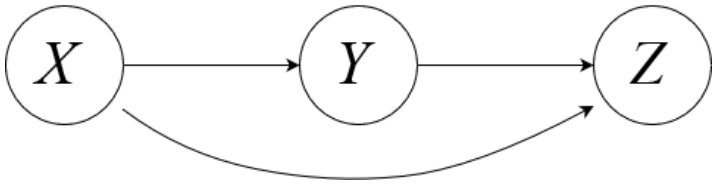


Figure 1-1: Schematic overview of the production of an output protein with the given sigma-factors

In fact, the three nodes X, Y and Z in figure (1-1) represent respectively σ-factor 70 (S70), σ-factor 28 (S28) and the eGFP as developed at Eindhoven University of Technology. These elements have been depicted in more detail in figures (1-2a) and (1-2b):

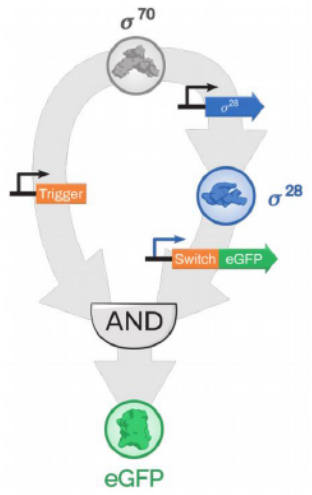
Note that the differences between the various CFFLs can mainly be found in the number of nodes and the composition of the CFFL itself. [3]

1-2-2 Family of CFFL's

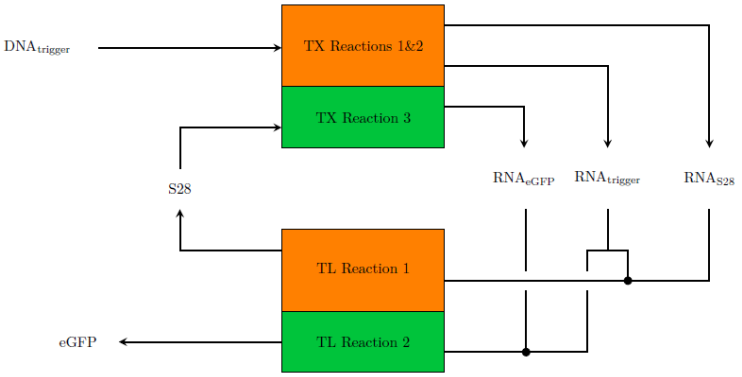
In [3] several opportunities are presented in order to extend and improve the modelling and analysis framework.

A complete modelling and analysis framework was developed and can be applied on more complex networks. There are twelve unique combined feedforward loops that are the centre of one of the research directions at Institute of Complex Molecular Systems (ICMS) from Eindhoven University of Technology:

Some of these clustering types (see figure (1-3)) for two feedforward loops haven't been considered in much detail yet. Therefore it could be useful for future work and similar methods can be applied at the corresponding specific CFFL.



(a) Used composition for CFFL by Pascal Pieters, Eindhoven University of Technology



(b) Schematic overview of the used CFFL

Figure 1-2: CFFL for the production of an output protein eGFP [3]

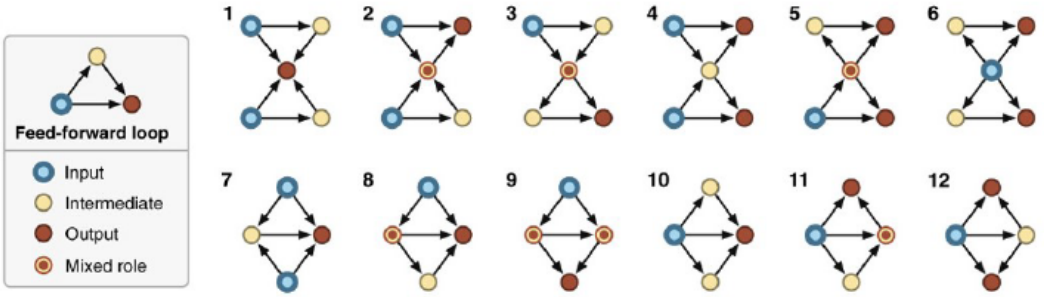


Figure 1-3: The 12 unique motif clustering types for two feedforward loops [3]

Besides, Other types of feedforward loops have paths that are not activating but inhibiting, for example type 1 incoherent feedforward loop. In addition, these network motifs are also present in the composition of the combined feedforward loops. Therefore, the modelling tools have to be extended to include kinetics that represent inhibition [3].

1-2-3 Batch (autonomous) and flow (non-autonomous) experiments

In order to get the necessary data for our Ordinary Differential Equation (ODE) models certain batch and flow experiments will be or have been made. The usage of data from these experiments will result in the desired models that fit these data. Afterwards simulations can be made with these models. By doing new batch and flow experiments and use of structural analysis tools the given model can eventually be verified and/or validated [3], [11].

Something we need to keep in mind is that adding species to a mixture will be a fast process and therefore inputs will rather be modelled as the setpoints of an initial state. For flow experiments on the other hand this adding process changes over time dependent on the flow

rate of the adding species. In this case we can describe the overall system as a state-space model in such a way that controllers/observers can be realized. How to adapt the flow rate in order to maintain a certain concentration or how to adapt this in order to find the desired concentration of the output protein over time will be of our interest.

1-3 Problem statement

Once we have set up the ODE or state-space model, we are able to apply model order reduction methods [4]. The purpose of the MSc Thesis is to apply several of these methods, to analyse what kind of differences occur between the original and reduced order model and to verify/validate the given methods for various biochemical circuits and networks. The main purpose of model order reduction is to simplify the overall model and to facilitate the overall analysis of our CFFL. Certainly in case of flow experiments model order reduction will improve the accuracy of the analysis and/or control approaches. The model order reduction method produces a less complex model where irrelevant parameters and equations have been dropped out of the system.

1-3-1 Research question / Objective

In the MSc Thesis we will be challenged to answer the following questions based on the above mentions respectively:

- **How to derive the corresponding state-space models for various CFFLs and when to apply which order reduction method to gain an optimal analysis, observer (or possibly controller) for the given CFFL in case of a batch and flow experiment ?**

1-3-2 Subquestions / Subobjectives

To answer the above research question, we will have to find the answers to the following subquestions for every included topic:

Defined model of Ordinary Differential Equations for our network of CFFLs

To get more insight in the dynamics that play a role in biochemical reactions, we first start by having a look at how the system is modelled and how the biochemical reactions are analysed. Then we will also discuss the modelling strategies. Mostly we will use mass-action kinetics that will describe the relation between the concentration of a substrate S and formation of product P (reaction rate). This gives rise to the following subquestion:

- How to apply the mass action kinetics in a larger system and network of CFFLs?
- In what kind of situation/experiment or network of biochemical reactions are we allowed to use the mass action kinetics?

A set of biochemical reactions can be rewritten in ODE models. The rewriting into a mathematical model is done for a couple of biological processes in which the law of mass action have been applied. This gives rise to the following subquestion:

- How do we model our biochemical system with Ordinary Differential Equations?

Modelling biochemical reaction networks and conservation laws

The conservation laws can be found for a system of biochemical equations. This can especially be deduced mathematically and gives rise to the following subquestions:

- How to find the number of conservation laws?
- What are the conservation laws within the system?

Once we have found the conservation laws, we are particularly interested in ways to reduce the number of Ordinary Differential Equations. This gives rise to the following subquestion:

- How to use the conservation laws as a way to reduce the number of Ordinary Differential Equations?

Quasi Steady State Approach

The Quasi-steady state approach (QSSA) is based on the assumption that the steady state of the species that evolve on the fast timescale will be settled quickly as soon as the experiment starts. Then the other species will evolve on the slow timescale. This gives rise to the following subquestion:

- How to apply the Quasi Steady State Approach?

The Quasi Steady State Approach is applied by neglecting the fast reactions over the slow reactions for the slow timescale. Therefore this gives rise to the following subquestion:

- How to make a distinction between slow and fast time scales?

Alternative modelling

For the network of CFFLs we can replace certain cycles and feedbackmechanisms by more simple structures. This will be called alternative modelling. Therefore this gives rise to the following subquestions:

- What are possible alternatives for the given system of reactions or the CFFLs?
- How does the new defined system and non-reduced system relate?

For the case where Michaelis-Menten kinetics have been used, we need to take renewed rate constants into account such that the reduced and non-reduced case will only have small deviations. Therefore this gives rise to the following subquestions:

- What kind of rate constants define the system at its best?
- How to find the optimal rate constants in the new defined system?

The kinetics will mostly be based on mass-action but if we can make more assumptions we are also allowed to use different kinetics like Michaelis-Menten. Therefore we will get the following subquestion:

- What kind of kinetics do we need in the new defined system?

Kron reduction order method

The Kron reduction order method is based on the reduction of the number of complexes and is used by rewriting the original form of the given system. Therefore this gives rise to the following subquestions:

- What are the alternatives to rewrite the system of equations?
- How to apply the Kron reduction order method?

According to literature there are theories that decide when the original system and reduced order system will have the same steady states. Therefore this gives rise to the following subquestions:

- When do species have the same concentrations in steady state?
- What are the differences and/or similarities between the steady states in the non-reduced model and the steady states in the reduced order system?

Flow experiments

For now we have just dealt with batch experiments, but one will also be able to transform this into a flow experiment. Besides one will also be able to apply the previous defined methods for this flow experiment. Therefore we will get the following subquestions:

- How to transform the system from a batch to a flow experiment?
- How to apply the earlier defined reduction order methods in case of a flow experiment?

Once we have to do with flow experiments we will also see some interesting aspects in the steady states even though this is influenced by defined inflow. Therefore this gives rise to the following subquestion:

- Why do/don't we see the same relation between the steady-states (non-reduced vs reduced system) as in the batch experiments?

1-4 Organisation report

Note we have already been started by explaining the system and problem. The content of this MSc Thesis is given with the following discussed chapters and sections:

In chapter 2 we will mainly explain the way in which we set up our model. Besides it will introduce the concept of conservation laws. Conservation laws can be seen as a way to eliminate the number of ODEs, but later on we will see that this is mainly be used to analyse what kind of kinetics we will have to use. In chapter 3 we will discuss the Quasi Steady State Approach. Even though there is one best approach, here we can make a distinction between 3 approaches. The first approach is based on relatively low rate constants to set certain $\dot{x}$ to zero, the second approach is based on an average of the various $\dot{x}$ to set these $\dot{x}$ to zero. Mainly we will use the third approach that initializes a different state which is a bit closer to the final steady state. In chapter 4 we will look at alternative ways of modelling our biochemical reaction network. For this we can replace certain feedback mechanisms and cycles by simpler structures. The part that has been replaced will get the optimized kinetics. In chapter 5 we will introduce the Kron reduction order method. This method reduces the number of complexes and therefore reduces the number of reactions indirectly. In chapter 6 we will discuss the flow experiments. This is mainly an extended model of the given biochemical reactions and batch experiment. Eventually we will draw conclusions and write down the future challenges in chapter 7. Futhermore the defined matrices and tabulars can be found in the Appendix.

Chapter 2

Modelling biochemical reaction networks and conservation laws

2-1 Modelling biochemical reaction networks (simple case)

2-1-1 Law of mass action

The rate at which chemicals, whether large macromolecules or simple ions, collide and interact from different chemical combinations can be described by the law of mass action. The number of collisions per unit time is taken to be proportional to the product of the concentrations. Here the factor of proportionality depends on the geometrical shapes and sizes of the reactant molecules and on the temperature of the substance of species. [2]

2-1-2 Modelling and conservation laws

To illustrate the principle of model order reduction we start with a simple case of a reversible reaction where three species are involved. For this consider the following chemical equation in (2-1):

$$\mathrm{A} + \mathrm{B} \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} \mathrm{C}. \tag{2-1}$$

Denote the concentrations of species A, B and C by respectively a, b and c. Then according to the law of mass action we get the following Ordinary Differential Equation (ODE) model:

$$\left.\begin{aligned} \dot{a} &= +k_{-1}c - k_1 ab \\ \dot{b} &= +k_{-1}c - k_1 ab \\ \dot{c} &= -k_{-1}c + k_1 ab \end{aligned}\right\} \Longrightarrow \begin{pmatrix} \dot{a} \\ \dot{b} \\ \dot{c} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}}_{N} \underbrace{\begin{pmatrix} k_{-1}c \\ k_1 ab \end{pmatrix}}_{v(x)} \tag{2-2}$$

By inspection, we directly see that the conservation laws read as:

$$\dot{a} + \dot{c} = 0 \Longrightarrow a + c = e_0 \in \mathbb{R} \tag{2-3}$$

$$\dot{b} + \dot{c} = 0 \Longrightarrow b + c = e_1 \in \mathbb{R} \tag{2-4}$$

Because of the conservation laws (2-3), (2-4) and since $\dot{b} = -\dot{c} = \dot{a}$ we are able to express the concentration of A and B in terms of those of C. And therefore only $\dot{c}$ will be needed if we want to know the full kinetics of the system. Now we can actually describe the full kinetics of this system by:

$$\dot{c} = -\Big(k_{-1} - (e_0 + e_1)\Big)c + k_1 c^2 + k_1 e_0 e_1 \tag{2-5}$$

Note here $x = \begin{pmatrix} a & b & c \end{pmatrix}^\top$. Then $N_{red}, v_{red}(x_{red})$ and x_{red} are the matrix or vectorform of respectively $N, v(x)$ and x in case that the original model has been transformed into its reduced order form. Note that x contains species A, B and C whereas x_{red} only contains species A and C.

To verify if the conservation laws won't change the original kinetics, we implement this part in Matlab first. Here we choose the flux rates in such a way that an equilibrium will settle over time. Eventually in equilibrium all species concentrations are constant. For this simple case it means that we need to define fluxrate k_1 different from k_{-1}. Since this is a rather simple set of ODEs, the differential equations are solved by using the Euler Forward numerical method. This means, since we have the equation $\dot{x} = Nv(x)$ we get an iterative recursion: $x_{k+1} = Nv(x_k) \cdot dt + x_k$ where dt is the chosen timestep.

Then we choose the initial state for both the original and reduced order model as follows: $x_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\top$.

Now, figure (2-1) depicts the kinetics of both models:

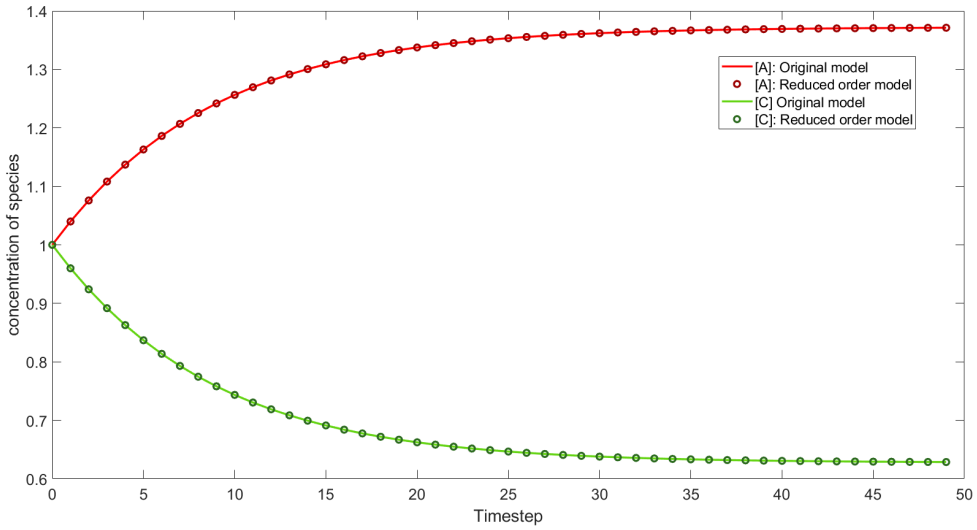


Figure 2-1: Concentrations of species A and C for the original model of reaction (2-1) and its reduced order form with used flux rates: $k_1 = \frac{1}{10}$ and $k_{-1} = \frac{3}{10}$, initial species concentrations of 1 and $dt = 1$. Note that the concentration of B will be the same as the concentration of A since the initial conditions and ODEs for these states are the same.

Clearly, the kinetics for these models are identical for the computed timesteps. As expected, dependent on the timestep, an equilibrium will settle over time .

2-2 Biochemical reaction network at ICMS

A first model is based on a predefined model as given in [3] from Institute of Complex Molecular Systems (ICMS) . For simplification lets define the concentrations of the following species to be:

$$
\begin{aligned}
x_1 &= [RNAP], & x_{12} &= [RNA_{eGFP}] \\
x_2 &= [S_{70}], & x_{13} &= [Ribo] \\
x_3 &= [DNA_t], & x_{14} &= [RNA_t : RNA_{S28}] \\
x_4 &= [DNA_{S_{28}}], & x_{15} &= [RNA_t : RNA_{S28} : Ribo] \\
x_5 &= [DNA_{e_{GFP}}], & x_{16} &= [S28] \\
x_6 &= [RNAP : S70], & x_{17} &= [RNA_t : RNA_{eGFP}] \\
x_7 &= [RNAP : S70 : DNA_t], & x_{18} &= [RNA_t : RNA_{eGFP} : Ribo] \\
x_8 &= [RNA_t], & x_{19} &= [e_{GFP_{dark}}] \\
x_9 &= [RNAP : S70 : DNA_{S28}], & x_{20} &= [eGFP] \\
x_{10} &= [RNA_{S28}], & x_{21} &= [RNAP : S28 : DNA_{eGFP}] \\
x_{11} &= [RNAP : S28],
\end{aligned}
$$

A network of biochemical reactions and CFFLs used in ICMS can schematically be drawn as in figure (2-2):

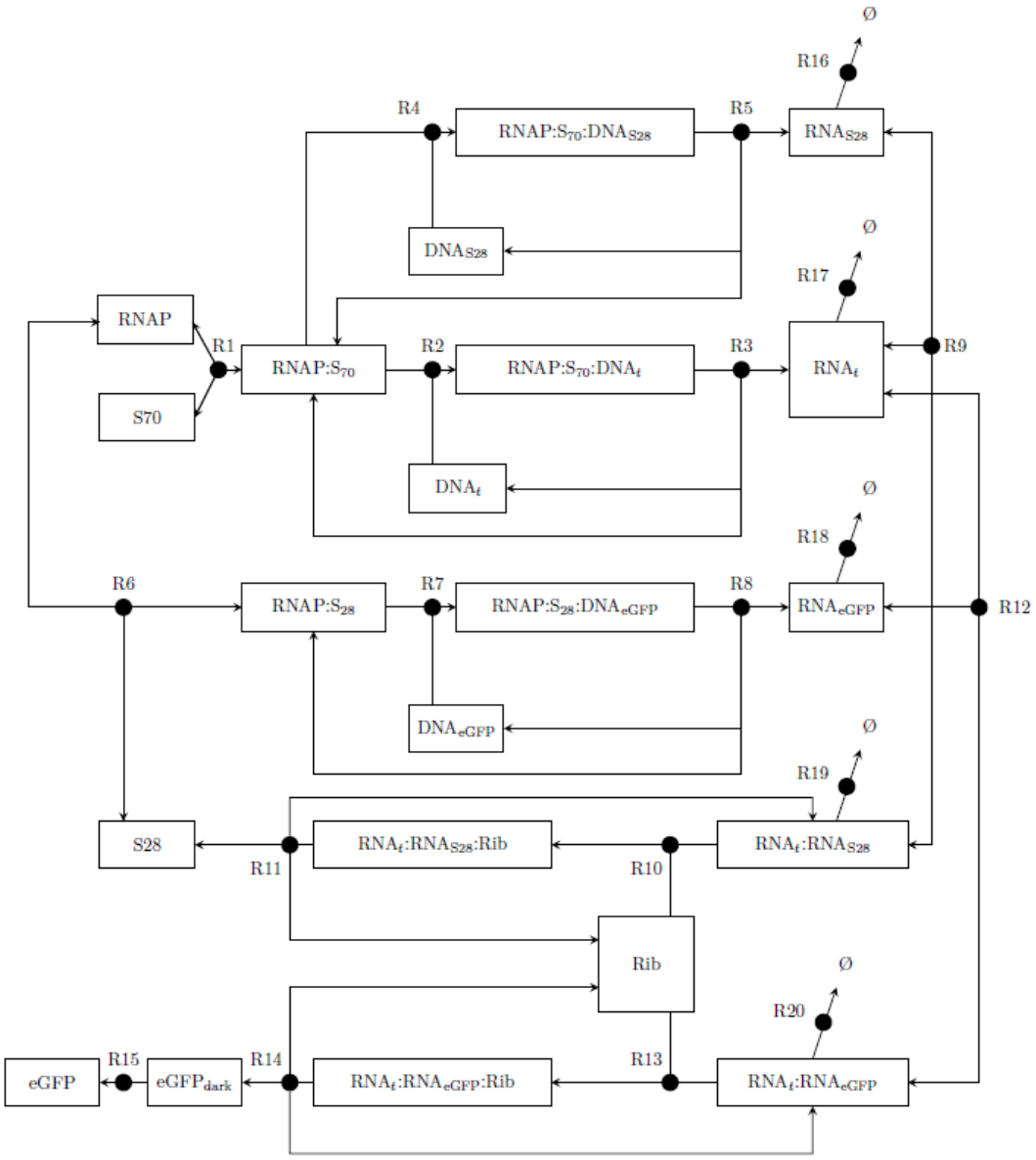


Figure 2-2: Schematic overview of biochemical reaction-network and CFFLs in ICMS

The flowscheme as given in figure (2-2) can be redrawn as depicted in figure (2-3):

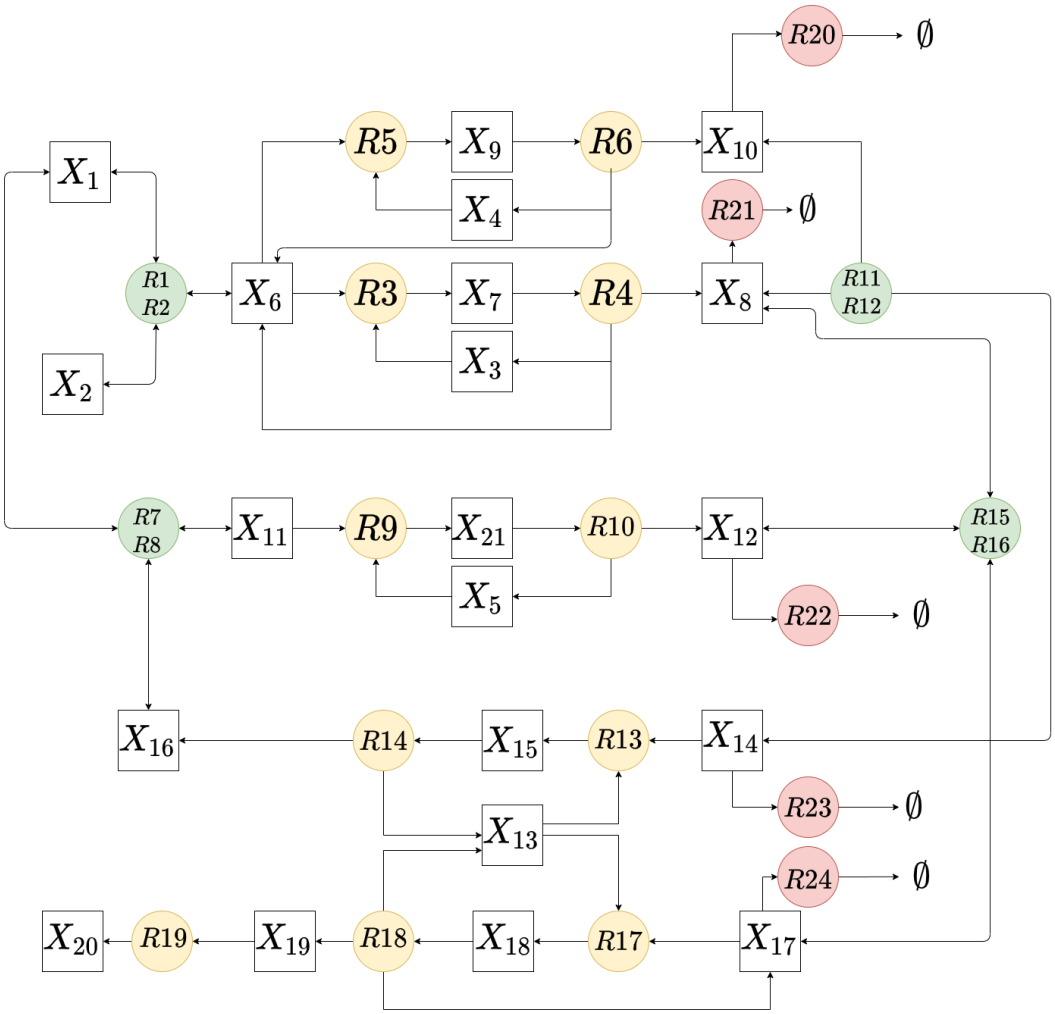


Figure 2-3: Schematic overview of biochemical reaction-network and CFFLs in ICMS: Red nodes indicate reactions of merely consumed species, green nodes indicate reversible reactions

According to this flowscheme we will get the following set of reactions $R1$ till $R24$ given by $\dot{x}$ based on the consumed/produced species:

$$
\begin{aligned}
R1\ \&\ R2: &\quad \mathrm{X_1} + \mathrm{X_2} \underset{\mathrm{k_{-2}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \mathrm{X_6} \\
R3: &\quad \mathrm{X_6} + \mathrm{X_3} \xrightarrow{\mathrm{k_3}} \mathrm{X_7} \\
R4: &\quad \mathrm{X_7} \xrightarrow{\mathrm{k_4}} \mathrm{X_8} + \mathrm{X_3} + \mathrm{X_6} \\
R5: &\quad \mathrm{X_6} + \mathrm{X_4} \xrightarrow{\mathrm{k_5}} \mathrm{X_9} \\
R6: &\quad \mathrm{X_9} \xrightarrow{\mathrm{k_6}} \mathrm{X_{10}} + \mathrm{X_4} + \mathrm{X_6} \\
R7\ \&\ R8: &\quad \mathrm{X_{11}} + \mathrm{X_{16}} \underset{\mathrm{k_{-8}}}{\overset{\mathrm{k_7}}{\rightleftharpoons}} \mathrm{X_1} \\
R9: &\quad \mathrm{X_{11}} + \mathrm{X_5} \xrightarrow{\mathrm{k_9}} \mathrm{X_{21}} \\
R10: &\quad \mathrm{X_{21}} \xrightarrow{\mathrm{k_{10}}} \mathrm{X_5} + \mathrm{X_{12}} \\
R11\ \&\ R12: &\quad \mathrm{X_{10}} + \mathrm{X_8} \underset{\mathrm{k_{-12}}}{\overset{\mathrm{k_{11}}}{\rightleftharpoons}} \mathrm{X_{14}} \\
R13: &\quad \mathrm{X_{14}} + \mathrm{X_{13}} \xrightarrow{\mathrm{k_{13}}} \mathrm{X_{15}}
\end{aligned}
\qquad
\begin{aligned}
R14: &\quad \mathrm{X_{15}} \xrightarrow{\mathrm{k_{14}}} \mathrm{X_{16}} + \mathrm{X_{13}} \\
R15\ \&\ R16: &\quad \mathrm{X_8} + \mathrm{X_{12}} \underset{\mathrm{k_{-16}}}{\overset{\mathrm{k_{15}}}{\rightleftharpoons}} \mathrm{X_{17}} \\
R17: &\quad \mathrm{X_{17}} + \mathrm{X_{13}} \xrightarrow{\mathrm{k_{17}}} \mathrm{X_{18}} \\
R18: &\quad \mathrm{X_{18}} \xrightarrow{\mathrm{k_{18}}} \mathrm{X_{13}} + \mathrm{X_{19}} + \mathrm{X_{17}} \\
R19: &\quad \mathrm{X_{19}} \xrightarrow{\mathrm{k_{19}}} \mathrm{X_{20}} \\
R20: &\quad \mathrm{X_{10}} \xrightarrow{\mathrm{k_{deg}}} \emptyset \\
R21: &\quad \mathrm{X_8} \xrightarrow{\mathrm{k_{deg}}} \emptyset \\
R22: &\quad \mathrm{X_{12}} \xrightarrow{\mathrm{k_{deg}}} \emptyset \\
R23: &\quad \mathrm{X_{14}} \xrightarrow{\mathrm{k_{deg}}} \emptyset \\
R24: &\quad \mathrm{X_{17}} \xrightarrow{\mathrm{k_{deg}}} \emptyset
\end{aligned}
$$

Then from (2-3) the full kinetics will be given by:

$$
\dot{x} = \begin{pmatrix}
k_{-2}x_6 - k_{-8}x_1 - k_1x_1x_2 + k_7x_{11}x_{16} \\
k_{-2}x_6 - k_1x_1x_2 \\
k_4x_7 - k_3x_3x_6 \\
k_6x_9 - k_5x_4x_6 \\
k_{10}x_{21} - k_9x_5x_{11} \\
k_4x_7 + k_6x_9 - k_{-2}x_6 + k_1x_1x_2 - k_3x_3x_6 - k_5x_4x_6 \\
k_3x_3x_6 - k_4x_7 \\
k_4x_7 + k_{-12}x_{14} + k_{16}x_{17} - k_{deg}x_8 - k_{11}x_8x_{10} - k_{14}x_8x_{12} \\
k_5x_4x_6 - k_6x_9 \\
k_6x_9 + k_{-12}x_{14} - k_{deg}x_{10} - k_{11}x_8x_{10} \\
k_{-8}x_1 - k_9x_5x_{11} - k_7x_{11}x_{16} \\
k_{10}x_{21} + k_{-16}x_{17} - k_{deg}x_{12} - k_{15}x_8x_{12} \\
k_{14}x_{15} + k_{18}x_{18} - k_{13}x_{13}x_{14} - k_{17}x_{13}x_{17} \\
k_{11}x_8x_{10} - k_{deg}x_{14} - k_{-12}x_{14} - k_{13}x_{13}x_{14} \\
k_{13}x_{13}x_{14} - k_{14}x_{15} \\
k_{14}x_{15} + k_{-8}x_1 - k_7x_{11}x_{16} \\
k_{18}x_{18} - k_{-16}x_{17} - k_{deg}x_{17} + k_{15}x_8x_{12} - k_{17}x_{13}x_{17} \\
k_{17}x_{13}x_{17} - k_{18}x_{18} \\
k_{18}x_{18} - k_{19}x_{19} \\
k_{19}x_{19} \\
k_9x_5x_{11} - k_{10}x_{21}
\end{pmatrix} \tag{2-6}
$$

2-3 High level model

For analysis purposes one can also have a look at the high level model of the given CFFL network. The high level model of the CFFL network is given by the equivalent schematic framework in figure 2-4 and 1-2b :

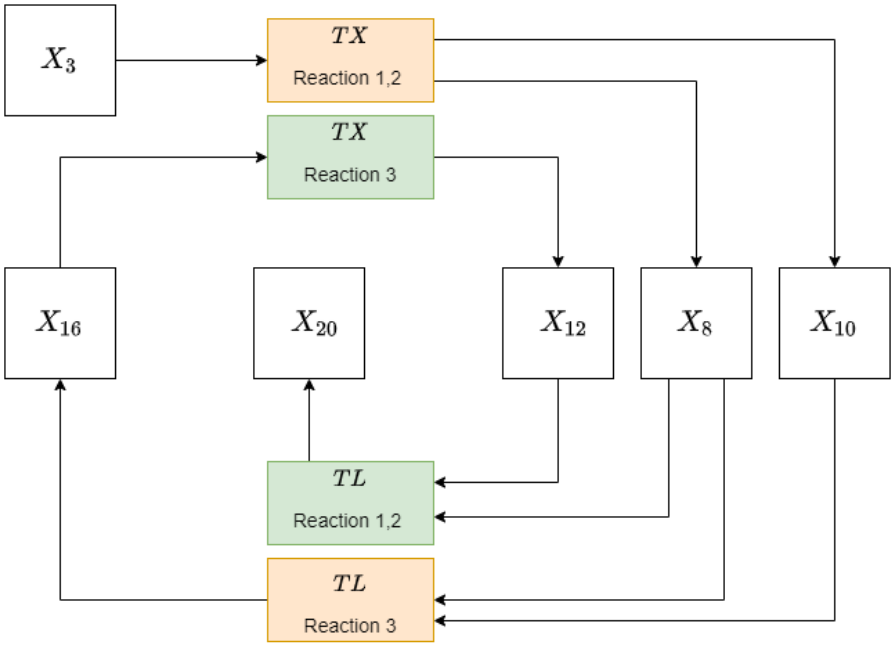


Figure 2-4: High level model of translation and transcription biochemical processes

This high level model has similar features like the one in the low level model or CFFL network. Here also e_{GFP} will be the output protein. First X_3 will be consumed which results in the production of X_{12}, X_8 and X_{10}. Then, as an internal node, X_{16} will be produced/consumed. Eventually X_{12} and X_8 are used for the production of e_{GFP}.

2-4 Characteristics for Steady State and Conservation laws

In general, the Stoichiometry matrix N is defined by a $m \times r$ matrix since the number of rows is equal to the number of species m and the number of columns is equal to the number of reactions r. Here an element $N(i,j)$ can be indicated by the species x_i and reaction R_j. More specifically,

$$N(i,j) = \begin{cases} +\delta_{ij}, & \text{if species } x_i \text{ is produced in reaction } R_j \\ -\delta_{ij}, & \text{if species } x_i \text{ is consumed in reaction } R_j \end{cases} \tag{2-7}$$

Note that, since all species in the reactions won't be consumed or produced more than once,here we have $\delta_{ij} = 1$. Thus here δ_{ij} denotes the number of produced or consumed species.

2-4-1 Fluxratios in steady-state

Another interesting aspect is the ratio of fluxes in case of the steady-state scenario. This will be found by looking at the right nullspace of the given matrix N. It can simply be proven as follows:

$$\dot{x} = Nv(x) = 0 \iff v(x) \in \ker(N) \tag{2-8}$$

2-4-2 Conservation laws

Conservation laws can be deduced by an equation of the form $a_1\dot{x}_1 + \ldots + a_n\dot{x}_n = 0$ and we are interested in finding a vector $a = \Big(a_1, \ldots, a_n\Big)^\top$ such that:

$$a^\top \dot{x} = a^\top Nv(x) = 0 \iff \tag{2-9}$$
$$a^\top N = 0 \iff \tag{2-10}$$
$$N^\top a = 0. \tag{2-11}$$

2-5 Using the Stoichiometry characteristics for elimination of equations

Thus by looking at the left nullspace of the Stoichiometry matrix N we can establish the conservation laws with which we have the option to reduce the number of equations used in the ODE model. Here the nullspace is spanned by 5 linearly independent vectors. If we arrange the entries within these vectors by $a_{i,1}, \ldots, a_{i,21}$ where $i \in \{1, \ldots, 5\}$. Then the conservation laws can be deduced as follows:

$$a_{i,1}\dot{x}_1 + \ldots + a_{i,21}\dot{x}_{21} = 0 \Longrightarrow \tag{2-12}$$
$$a_{i,1}x_1 + \ldots + a_{i,21}x_{21} = c_i \in \mathbb{R}, \text{ where } i \in \{1, \ldots, 5\} \tag{2-13}$$

Since these equations are linearly independent, we can choose five equations for the derivation of the conservation laws. For now we choose the (e.g.) first five states that will be eliminated or substituted. These can be expressed in terms of the other states and we can write:

$$x_i = \frac{c_i - \sum_{k=1,k\neq i}^{21} a_{i,k} x_k}{a_{i,i}} \tag{2-14}$$

Then these 5 equations define a plane in the higher dimensional space in $\mathbb{R}^{21}$ and reduces the space to a dimension of $\mathbb{R}^{16}$. In other words, if we have a total of m ODEs and it will be reduced to $n < m$ ODEs, we will have a total of $n - m$ algebraic equations that make this happen. Eventually the reduced order model will look like:

$$\begin{aligned} \dot{\mathbf{x}}_{\mathbf{reduced}} &= \mathbf{N}\mathbf{v}(\mathbf{x}_{\mathbf{reduced}}) \\ \mathbf{x}_{\mathbf{reduced}} &\in \mathbb{R}^{\mathbf{16}}, \\ \mathbf{N} &\in \mathbb{R}^{\mathbf{16}\times\mathbf{24}}, \\ \mathbf{v}(\mathbf{x}_{\mathbf{reduced}}) &\in \mathbb{R}^{\mathbf{24}} \end{aligned} \tag{2-15}$$

Note that $v(x)$ can be replaced by $v(x_{reduced})$ if the expressions for states x_1 till x_5 are substituted within the expressions for the original states of $v(x)$.

Eventually a plot of the numerical approach to determine the reduced order model along with the original model will give us figure (2-5)

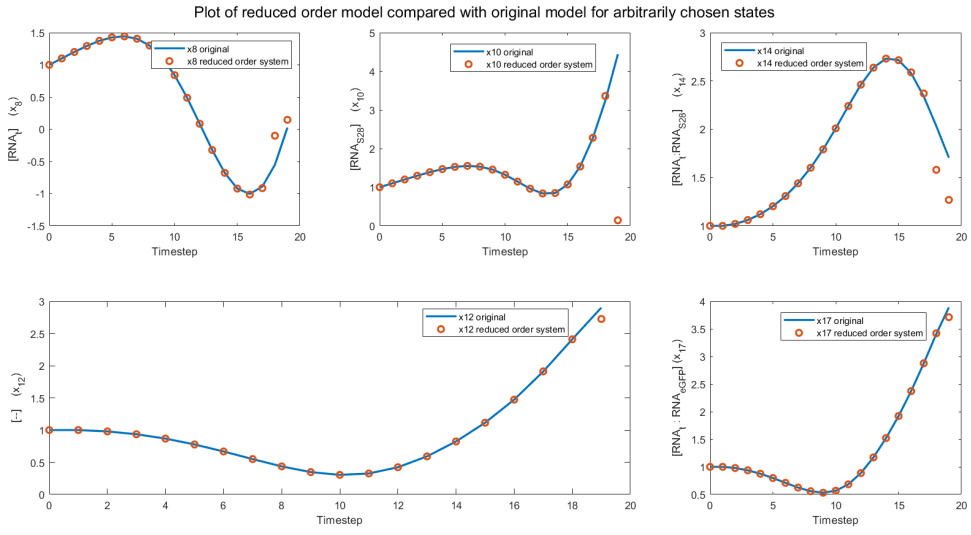


Figure 2-5: Comparison of a reduced order model for a set of biochemical reactions

Note that the reduced order model diverges (at the latter timesteps) from the original model since the values for the various states show more extreme values per timestep ($a_{i,i} \approx 0$). Nevertheless one expects that both models should be identical since this is what conservation

laws define. Luckily there are two better ways to get a model that simplifies the substitution equations for the reduced order method .

If we transform the first model of ODEs into $\dot{x} = Nv(x)$ we get:

$$v(x) = \begin{pmatrix} v_1(x) \\ v_2(x) \end{pmatrix} \text{ in which } \quad v_1(x) = \begin{pmatrix} k_1 x_1 x_2 \\ k_{-2} x_6 \\ k_3 x_3 x_6 \\ k_4 x_7 \\ k_5 x_6 x_4 \\ k_6 x_9 \\ k_7 x_{11} x_{16} \\ k_{-8} x_1 \\ k_9 x_{11} x_5 \\ k_{10} x_{21} \\ k_{11} x_{10} x_8 \\ k_{-12} x_{14} \end{pmatrix} \text{ and } v_2(x) = \begin{pmatrix} k_{13} x_{14} x_{13} \\ k_{14} x_{15} \\ k_{15} x_8 x_{12} \\ k_{-16} x_{17} \\ k_{17} x_{17} x_{13} \\ k_{18} x_{18} \\ k_{19} x_{19} \\ k_{deg} x_{10} \\ k_{deg} x_8 \\ k_{deg} x_{12} \\ k_{deg} x_{14} \\ k_{deg} x_{17} \end{pmatrix}$$

Note that the positive kinetics are given by the right-directional chemical reaction. Now the Stoichiometry matrix is defined by a 21×24 matrix since the number of rows is equal to the number of species and the number of columns is equal to the number of reactions. Once more, here an element $N(i, j)$ can be indicated by the species x_i and reaction R_j. Eventually in this way we are able to construct the matrix as can be found in the Appendix (A-3).

The first alternative is to use the equation (2-14) again, but here we avoid the lower values of $a_{i,i}$ by selecting the proper equations for $\dot{x}_i$. This means we use the suitable equation (with reasonable values) where we are allowed to make the substitution (2-14). Now we will solve the system of ODEs in Matlab. By doing so we end up in figure (2-6) where several states of the original model in comparison with the reduced order model have been depicted.

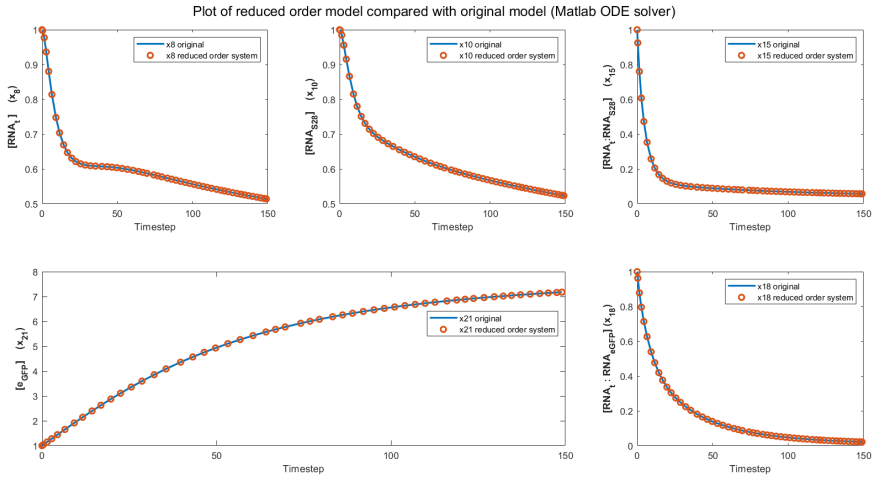


Figure 2-6: ICMS model compared with its reduced order model for several states with reactions rates set to $\frac{1}{10}$ and initial conditions set to 1.

Indeed, now both models are the same for the plotted states (and thus it is quite reasonable to conclude that this holds for the full system). So this method even visualizes the exact values of the original model. The numerical values can be considered to be realistic since they stay positive regardless of the number of timesteps taken and/or the different initial conditions or reaction rates. This also shows that the method fails as soon as the original system has to do with extreme numerical values for the given concentrations.

The second alternative would be based on inspection of the summation of the rows of the Stoichiometry matrix N. To get an idea of the explicit conservation laws we can use inspection to see which rows in N add to zero. Since the dimension of this nullspace is five, we need to look for five different additions that give zero row-vectors. In this way we are able to rewrite the conservation laws in (2-4) with constants $c'_1, \ldots, c'_5 \in \mathbb{R}$:

$$c'_1 = [DNA_t] + [RNAP : S70 : DNA_t] \tag{2-16}$$
$$c'_2 = [S70] + [RNAP : S70] + [RNAP : S70 : DNA_t] + [RNAP : S70 : DNA_{S28}] \tag{2-17}$$
$$c'_3 = [DNA_{e_{GFP}}] + [RNAP : S28 : DNA_{e_{GFP}}] \tag{2-18}$$
$$c'_4 = [DNA_{S28}] + [RNAP : S70 : DNA_{S28}] \tag{2-19}$$
$$c'_5 = [Ribo] + [RNA_t : RNA_{S28} : Ribo] + [RNA_t : RNA_{e_{GFP}} : Ribo] \tag{2-20}$$

If we are interested in the enumeration of species, the above conservation laws will be identical to the following:

$$c'_1 = x_3 + x_7 \tag{2-21}$$
$$c'_2 = x_2 + x_6 + x_7 + x_9 \tag{2-22}$$
$$c'_3 = x_5 + x_{21} \tag{2-23}$$
$$c'_4 = x_4 + x_9 \tag{2-24}$$
$$c'_5 = x_{13} + x_{15} + x_{18} \tag{2-25}$$

Indeed these equations could have been used to reduce the order of the system easier, but it ignores the generality of the method as can be used for more examples. Note that for most of the conservation laws the result follows from (multiple) feedback loops or biochemical circuits within the full system. Via these feedback loops the species will be produced or consumed either way and reformed somewhere regardless of the rate constants or the number of reactions it will take until the species have been (re)formed (in)directly. So it is a helpful way to see which rows add to zero in N directly, but unfortunately in general the helpful method cannot be used and determining vectors in $\ker(N^\top)$ will be hard to find in case of inspection.

2-6 Conclusions conservation laws and reduction order method

To get more insight in the dynamics that play a role in biochemical reactions, we first start by having a look at how the system is modelled and how the biochemical reactions are analyzed. Then we will also discuss the modelling strategies. To analyze, a set of biochemical reactions

can be rewritten in ODE models. The rewriting into a mathematical model is done for a couple of biological processes in which the law of mass action have been applied. Even though more kinetics are possible (e.g. Hill or Michaelis-Menten (MM)), mostly we will use mass-action kinetics that will describe the relation between the concentration of a substrate S and formation of product P (reaction rate). Since we know the properties in case of low-dimensional models, we can determine the stability regions relatively easy.

The number of conservation laws can be deduced by looking at the dimension of the left nullspace of the Stoichiometry matrix. Besides we can write the exact conservation laws with this left nullspace. This can be done in different ways, but all approaches lead to the same result. However, one approach might be more sensible to numberical errors than the other. By using the appropriate substitutions we can eliminate certain species updates $\dot{x}$.

Chapter 3

Quasi Steady State Approach

The Quasi-steady state approach (QSSA) is based on the assumption that the steady state of the species that evolve on the fast timescale will be settled quickly as soon as the experiment starts [18], [19], [20], [21] . Then the other species will evolve on the slow timescale. In steady state we have $\dot{x} = Nv(x) = 0$. This means that $v(x)$ is identical to one of the vectors that is included in the span of the kernel of N.By doing this the flux distribution in steady state between every element in $v(x)$ can be determined for the full experiment. In quasi steady state we will take the fastest reactions in steady state into account. This can be determined in multiple ways. Firstly, we will mainly be interested in the reactions with the relative high reaction rates with respect to the other rate constants. For simplicity, one can e.g. assume that the lowest rate constants can actually be set to zero. Furthermore this can also be determined by using the averaged $\dot{x}$. Though, in this chapter we will mainly focus on the rate at which a steady state will be settled per species. Then this species can be initialized as its steady state directly. This can be denoted as the quasi steady state.

3-1 Choosing some initial states closer to their steady states

For choosing some initial states (identical to the final steady states) we need to take the eventual steady states into account and initialize the fastest settled steady states from the beginning. Here we take the final steady states as the last data points of the state (for a large number of timesteps). For this we need to know the steady states that will have the fastest settling times with respect to the other species. We can determine these species graphically as is depicted in figure (3-1):

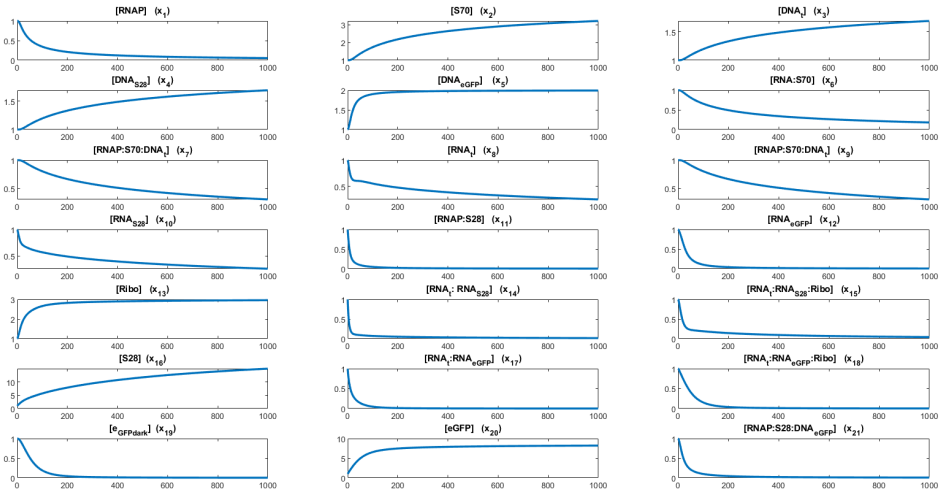


Figure 3-1: Species concentrations over time for this network of CFFLs

From this figure it will be a bit tough to read the species with the fastest settling times even though we will have an indication graphically. To determine the number of timesteps needed until the curve has reached its steady states we have two options. The first option is to look at the number of timesteps it takes until the curve for the differential state is close to zero (with a tolerance interval). The implicit relation between the fast and slow species will identify the slow manifold. The relation for the steady state x^* is given by:

$$Nv(x^*) = \dot{x}^* = 0 \iff v(x^*) \in \ker(N) \tag{3-1}$$

The other option will be a direct approach where we will have a look at the number of timesteps needed until the curve has reached the steady state within a 10% tolerance interval. Here we choose the latter option where we need to keep in mind that we will have to do with a total of 1000 timesteps.

Then we get table (3-1) from which we read the species to be initialised in that order from top to bottom (see Appendix for full table):

Species to be initialised	Timestep at which the steady states are reached within a 10% tolerance interval
X_5	51.37
X_{13}	100.81
X_{20}	182.79
X_3	468.35
X_4	468.35
$\vdots$	$\vdots$
X_{19}	948.76

Table 3-1: Species to be initialised along with the timesteps at which the steady states are reached within a 10% tolerance interval

Now, based on this settling, we can initialize the mentioned species. Then figure (3-2) shows the settling of the species $x_8, x_{10}, x_{14}, x_{20}$ and x_{17} for the non-reduced and reduced case.

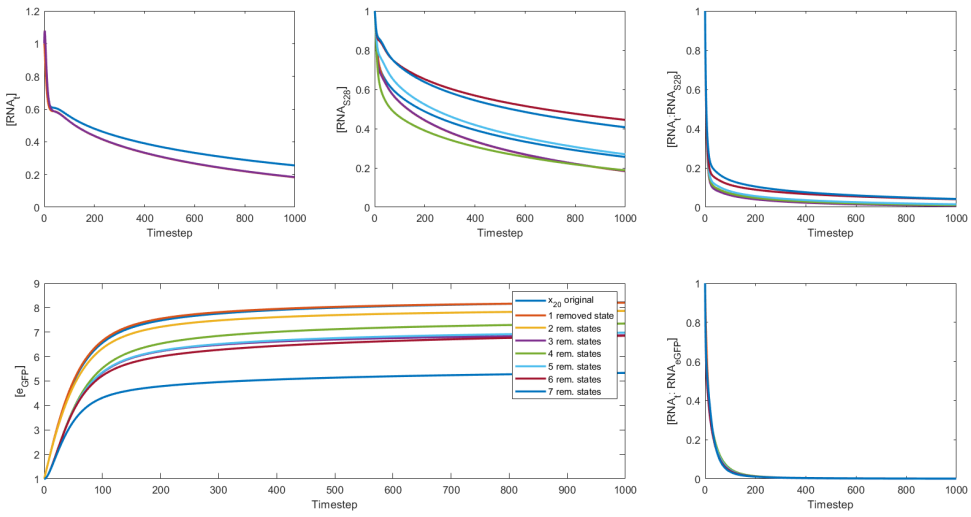


Figure 3-2: Settling of the species $x_8, x_{10}, x_{14}, x_{20}$ and x_{17} for the non-reduced and reduced case

As can be concluded from figure (3-2) the settling time reduces if a higher reduction order method is used. This is logically since we start with a close to steady state scenario. The steady states of the reduced and non-reduced case are almost the same with the exception for the output protein e_{GFP}.

For a full Quasi Steady State Approach one actually wants to initialize species as constants (the eventual steady-states) over time. Then we can finally say that the differential equations for these fast species are neglected and thus that the species are removed as the reduction order method suggests. In general, we have the following:

Let x be the fast evolving species, let y be the slow evolving species. Then assume we have $\dot{x} = f(x, y)$ and $\dot{y} = g(x, y)$. Then the dynamics for the slow timescale read as:

$$0 = f(x, y) \Longrightarrow x = h(y) \Longrightarrow \dot{y} = g(h(y), y)$$

In this model, we will have to do with the following equations for functions $f(x, y)$ and $h(y)$:

$$\dot{x}_5 = 0 \qquad \Longrightarrow x_5 = \frac{k_{10}}{k_9 x_{11}} x_{21} \tag{3-2}$$

$$\dot{x}_{13} = 0 \qquad \Longrightarrow x_{13} = \frac{k_{14}x_{15} + k_{18}x_{18}}{k_{13}x_{14} + k_{17}x_{17}} \tag{3-3}$$

$$\dot{x}_{20} = 0 \qquad \Longrightarrow x_{19} = 0 \tag{3-4}$$

$$\dot{x}_3 = 0 \qquad \Longrightarrow x_3 = \frac{k_4 x_7}{k_3 x_6} \tag{3-5}$$

$$\dot{x}_4 = 0 \qquad \Longrightarrow x_4 = \frac{k_6 x_9}{k_5 x_6} \tag{3-6}$$

Now, if we both use the initialisation of the species and $\dot{x} = 0$ for these species (i.e. the steady-states are constant over time from the beginning of the experiment), we will get the classical Quasi Steady State Approach. Here the first order reduction method means that the first species(X_5) is removed from the system and the steady-state of this species is initialized from the beginning of the experiment. Then the second order method means that the first two species(X_5 and X_{13}) are removed from the system and both X_5 and X_{13} have a constant steady-state from the beginning. In general we will have a total of n removed species for the nth order reduction method. For the sequence in which the species will be removed we refer to table (3-1).

By applying the classical Quasi Steady State Approach we get the following results as given in figure (3-3):

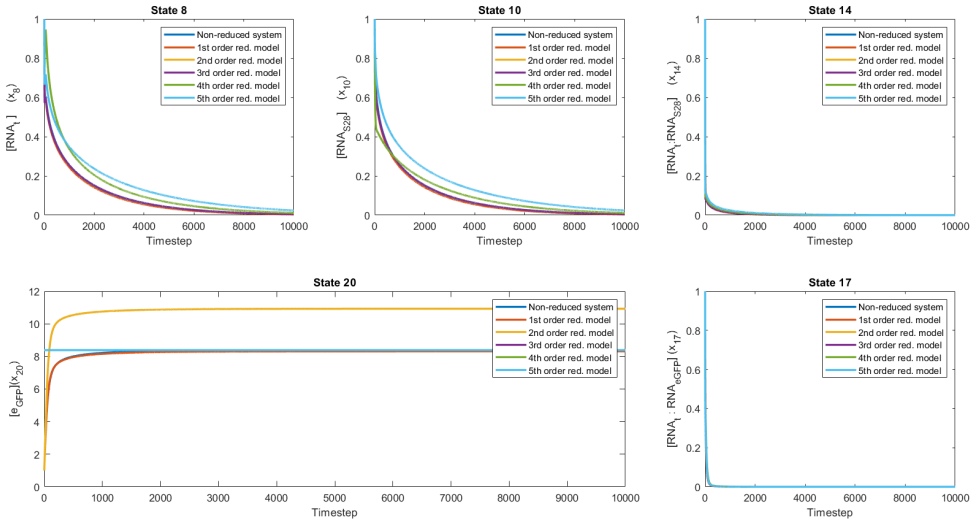


Figure 3-3: Applied Quasi Steady State Approach with reduction order 1 until 5 where initialisation of (constant) steady-states from the beginning of the experiment have been taken into account.

Based on these results, we see that applying the Quasi Steady State Approach results in a model with the same steady-states (as the original model). Though this will be different for the output protein e_{GFP}. Furthermore some models overlap somehow. For state 14 and 17 the reduction order models coincide for the largest part of their trajectories. For state 20 the blue light line also represents the third and fourth reduction order.

3-2 Conclusions Quasi Steady State Approach

The Quasi Steady State Approach is applied by neglecting the fast reactions over the slow reactions for the slow timescale. Then the differential equations for the fast reactions can be set to zero and we will, similar to the method of using conservationlaws, get substitutions or eliminations accordingly. Another useful aspect is the initialisation of the steady states where the fastest settled species are taken into account first. This is also known as the identification of the slow manifold. For this method a higher reduction order implies faster settling times since the number of initialized species (identical to the final steady states) is higher. The steady states between non-reduced and reduced case are (approximately) the same except for the output proteins.

3-3 Discussion Quasi Steady State Approach

3-3-1 Advantages Quasi Steady State Approach

The Quasi Steady State Approach will have the following advantages:

- The steady states in the reduced order system will almost be identical to the steady states in the non-reduced system.
- For smaller reduction order models there will be a small deviation between the reduced and non-reduced system.
- A combined method of slow manifold identification can show a best way to initialize the defined species concentrations

3-3-2 Disadvantages Quasi Steady State Approach

The Quasi Steady State Approach will have the following disadvantages:

- A higher order reduction model is needed before the reduced order model or non-reduced model have the same steady states.
- The initialisation of species concentrations (or the removal of certain species) makes the model completely different at the beginning of the experiment.

3-3-3 Challenges of the Quasi Steady State Approach

The identification of the slow manifold can be done in multiple ways. Basically here one wants to find the sequence of the species concentrations that need to be initialised and thus actually the species that will need to be eliminated. Here one can base this on the time it takes till it reaches its steady state or (equivalently) one can base this on the time it takes until the derivative state reaches zero. Furthermore a higher order reduction model is needed before the reduced order model or non-reduced model have the same steady states. Though a lower order reduction model will result in smaller deviations between the reduced and non-reduced system.

Chapter 4

Alternative modelling

If we look at the overall system we can remove certain structures like cycles and feedback-mechanisms and replace them by more simple reactions. By doing this the kinetics at some points will be changed. For example we can rewrite our CFFL network in smaller ones where we denote these certain subsystems. Since we did most of the kinetics with mass-action we want to keep this within the new modelling framework in the first place. Here we simply follow the reactionscheme in such a way that the schematic overview of the reduced order system is in line with the original flowscheme.

If we do this, we'll define the composed system with respect to the non-reduced system as given in figure (4-1):

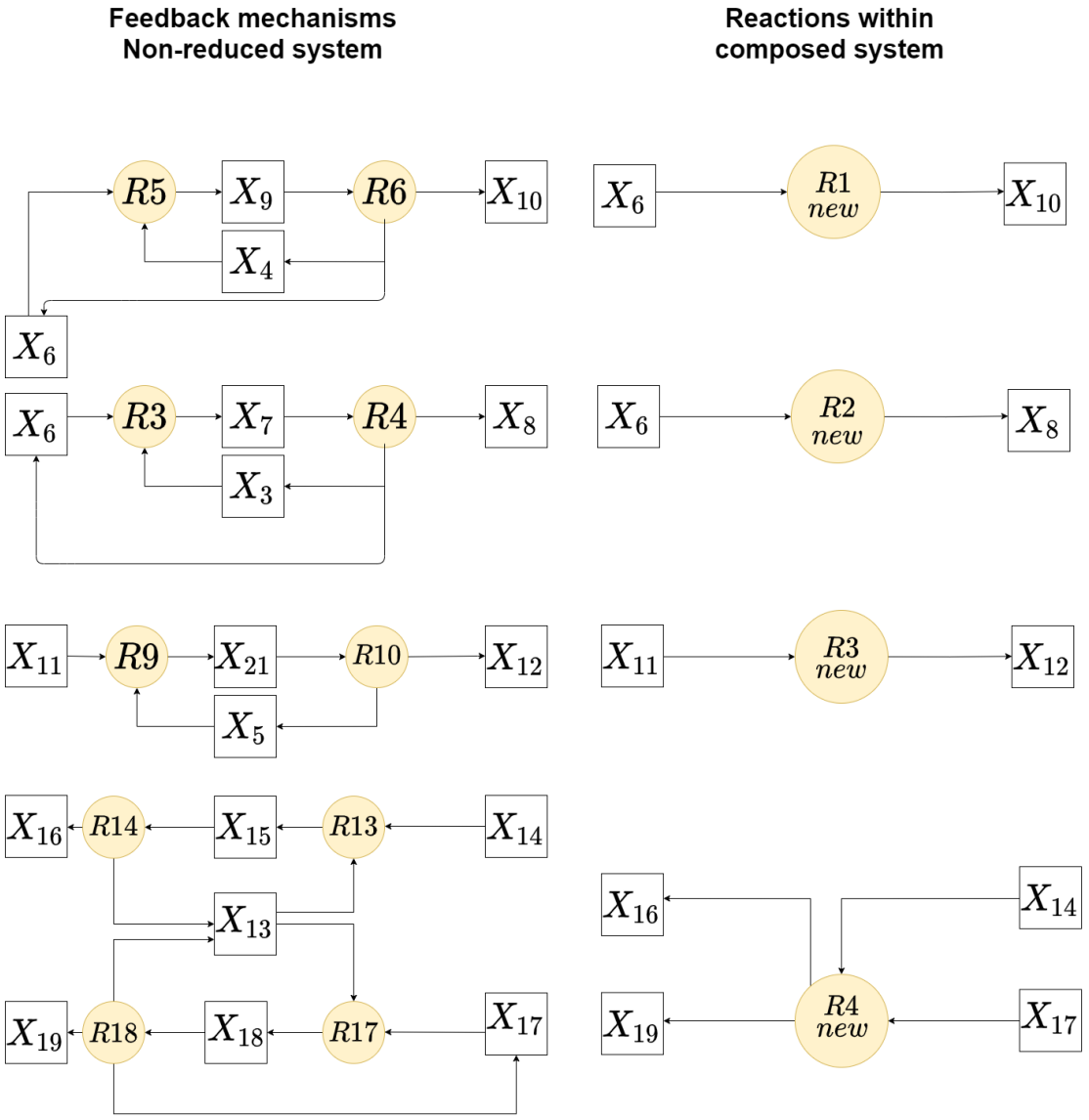


Figure 4-1: Composed subsystems with respect to original flowscheme structures

Actually, as seen here, we will remove the following species from the system of equations:

$\mathbf{X_9}$	$RNAP : S70 : DNA_{S28}$	$\mathbf{X_3}$	DNA_t
$\mathbf{X_4}$	DNA_{S28}	$\mathbf{X_{15}}$	$RNA_t : RNA_{S28} : Ribo$
$\mathbf{X_{21}}$	$RNAP : S28 : DNA_{eGFP}$	$\mathbf{X_5}$	DNA_{eGFP}
$\mathbf{X_{13}}$	$Ribo$	$\mathbf{X_{18}}$	$RNA_t : RNA_{eGFP} : Ribo$
$\mathbf{X_7}$	$RNAP : S70 : DNA_t$		

Table 4-1: Removed species in the given subsystems

Then the equivalent flowscheme can now be depicted as in figure (4-2):

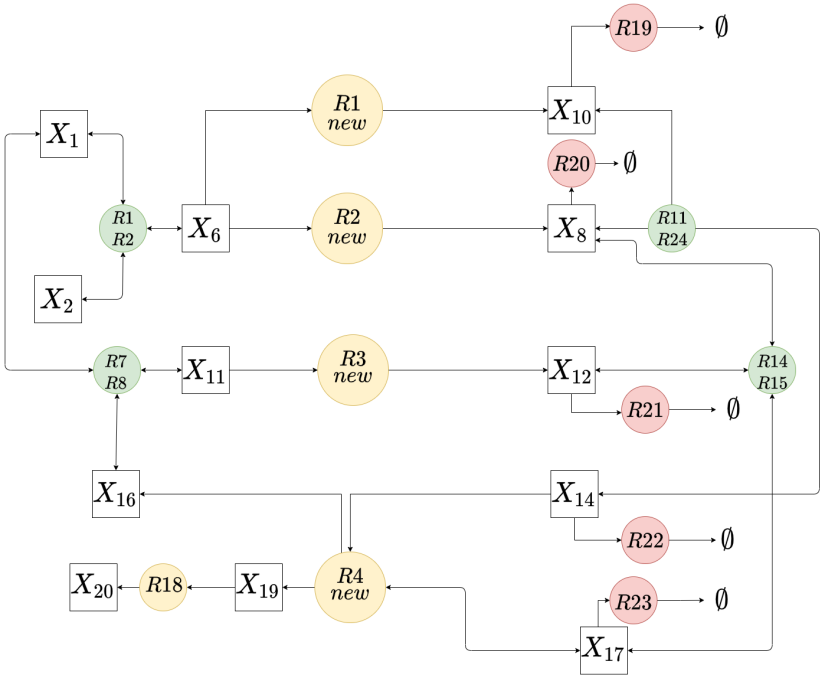


Figure 4-2: Equivalent flowscheme with the given reactions and species

Thus this in turn, will give us the following new set of reactions:

$$
\begin{aligned}
R1: &\ \mathrm{X}_1 + \mathrm{X}_2 \xrightarrow{\mathrm{k}_1} \mathrm{X}_6 & \qquad R18: &\ \mathrm{X}_{19} \xrightarrow{\mathrm{k}_{18}} \mathrm{X}_{20}\\
R2: &\ \mathrm{X}_6 \xrightarrow{\mathrm{k}_2} \mathrm{X}_1 + \mathrm{X}_2 & \qquad R11: &\ \mathrm{X}_8 + \mathrm{X}_{10} \xrightarrow{\mathrm{k}_{11}} \mathrm{X}_{14}\\
R1,new: &\ \mathrm{X}_6 \xrightarrow{\mathrm{k}_{1,\mathrm{new}}} \mathrm{X}_{10} & \qquad R19: &\ \mathrm{X}_{10} \xrightarrow{\mathrm{k}_{\mathrm{deg}}} \emptyset\\
R2,new: &\ \mathrm{X}_6 \xrightarrow{\mathrm{k}_{2,\mathrm{new}}} \mathrm{X}_8 & \qquad R20: &\ \mathrm{X}_8 \xrightarrow{\mathrm{k}_{\mathrm{deg}}} \emptyset\\
R7: &\ \mathrm{X}_{11} + \mathrm{X}_{16} \xrightarrow{\mathrm{k}_7} \mathrm{X}_1 & \qquad R21: &\ \mathrm{X}_{12} \xrightarrow{\mathrm{k}_{\mathrm{deg}}} \emptyset\\
R8: &\ \mathrm{X}_1 \xrightarrow{\mathrm{k}_8} \mathrm{X}_{11} + \mathrm{X}_{16} & \qquad R22: &\ \mathrm{X}_{14} \xrightarrow{\mathrm{k}_{\mathrm{deg}}} \emptyset\\
R3,new: &\ \mathrm{X}_{11} \xrightarrow{\mathrm{k}_{3,\mathrm{new}}} \mathrm{X}_{12} & \qquad R23: &\ \mathrm{X}_{17} \xrightarrow{\mathrm{k}_{\mathrm{deg}}} \emptyset\\
R4,new: &\ \mathrm{X}_{14} \xrightarrow{\mathrm{k}_{4,\mathrm{new}}} \mathrm{X}_{19} + \mathrm{X}_{16} & \qquad R24: &\ \mathrm{X}_{14} \xrightarrow{\mathrm{k}_{-12}} \mathrm{X}_8 + \mathrm{X}_{10}
\end{aligned}
$$

This new defined network can now be given according to the defined mass action kinetics within this network. Here we get:

$$
\hat{N}^\top = \begin{array}{r} \\ R1 \\ R2 \\ R1new \\ R2new \\ R7 \\ R8 \\ R3new \\ R4new \\ R18 \\ R11 \\ R19 \\ R20 \\ R21 \\ R22 \\ R23 \\ R24 \end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} x_1 & x_2 & x_6 & x_{10} & x_8 & x_{16} & x_{11} & x_{12} & x_{14} & x_{16} & x_{19} & x_{20} \end{array} \\
\left[\begin{array}{cccccccccccc}
-1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0
\end{array}\right]
\end{array}
\tag{4-1}
$$

$$\text{Then } \dot{\hat{x}} = \hat{N}\begin{pmatrix}\hat{v}_1\\ \hat{v}_2\end{pmatrix} \text{ where } \hat{v}_1 = \begin{pmatrix} k_1\hat{x}_1\hat{x}_2 \\ k_2\hat{x}_6 \\ k_{1,new}\hat{x}_6 \\ k_{2,new}\hat{x}_6 \\ k_7\hat{x}_1\hat{x}_{16} \\ k_8\hat{x}_{11} \\ k_{3,new}\hat{x}_{11} \\ k_{4,new}\hat{x}_{14} \end{pmatrix} \hat{v}_2 = \begin{pmatrix} k_{18}\hat{x}_{19} \\ k_{11}\hat{x}_8\hat{x}_{10} \\ k_{19}\hat{x}_{10} \\ k_{20}\hat{x}_8 \\ k_{21}\hat{x}_{12} \\ k_{22}\hat{x}_{14} \\ k_{23}\hat{x}_{17} \\ k_{24}\hat{x}_{14} \end{pmatrix} \hat{x} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_6 \\ \hat{x}_{10} \\ \hat{x}_8 \\ \hat{x}_{16} \\ \hat{x}_{11} \\ \hat{x}_{12} \\ \hat{x}_{14} \\ \hat{x}_{17} \\ \hat{x}_{19} \\ \hat{x}_{20} \end{pmatrix} \tag{4-2}$$

4-1 Changed reactions

If we remove the first subsystem reactions R5 and R6 will be removed and a new reaction will replace these two:

$$\mathrm{X_6} \xrightarrow{\mathrm{k_{1,new}}} \mathrm{X_{10}} \tag{4-3}$$

If we remove the second subsystem reactions R3 and R4 will be removed and a new reaction will replace these two:

$$\mathrm{X_6} \xrightarrow{\mathrm{k_{2,new}}} \mathrm{X_8} \tag{4-4}$$

If we remove the third subsystem reactions R9 and R10 will be removed and a new reaction will replace these two:

$$\mathrm{X_{11}} \xrightarrow{\mathrm{k_{3,new}}} \mathrm{X_{12}} \tag{4-5}$$

If we remove the fourth subsystem reactions R13, R14, R17 and R18 will be removed and a new reaction will replace these four:

$$\mathrm{X_{14}} + \mathrm{X_{17}} \xrightarrow{\mathrm{k_{4,new}}} \mathrm{X_{16}} + \mathrm{X_{19}} \tag{4-6}$$

4-2 Kinetics for the new flux term

4-2-1 Mass-action kinetics with optimal rate constants

A first choice would be to assume that all new reactions will happen according to mass-action kinetics. This sounds to be a fair choice since the original flow scheme of reactions also has to do with mass-action kinetics. Later on we will see that better kinetics can be used.

If we assume mass action kinetics for now, we can replace the reactions as discussed in section 'changed reactions'. For all systems, where subsystems have been replaced, a lot of species will have similar curves and tend to go to zero. Though we know that the conservation laws will make sure that some species must be present. Nevertheless most species will (indirectly) be used for the production of eGFP. To see this we made a plot of $[RNAP]$ and $[e_{GFP,dark}]$ as given in figure (4-3). Here 'one new subsystem' means that we change the first subsystem by its alternative form, 'two new subsystems' means that we change the first and second subsystem by its alternative forms, and so on.

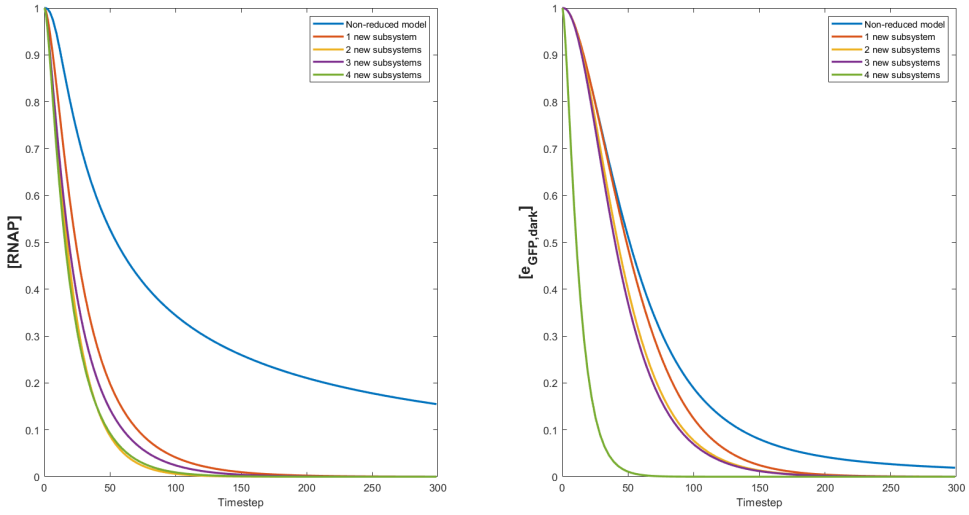


Figure 4-3: $RNAP$ and $e_{GFP,dark}$ concentrations for a change in up to four changed subsystems

Most curves follow similar trajectories like state 19 ($e_{GFP,dark}$) and end in a zero concentration somewhere. A higher number of removed subsystems will result in larger deviations from the non-reduced case. This holds both for the steady state as well as the trajectory towards this equilibrium. Though we can change this a bit by choosing the appropriate rate constants. If one wants the curves for the species concentrations more close to the non-reduced case, the following minimization problem will need to be solved:

$$\underset{k_{1,new},k_{2,new},k_{3,new},k_{4,new}}{\text{minimize}} \left\langle x - \hat{x}, x - \hat{x} \right\rangle \tag{4-7}$$

Here $x = \int Nv(x)\mathrm{d}t$ and $\hat{x} = \int \hat{N}\hat{v}(\hat{x})\mathrm{d}t$ will have to be compared in order to have an optimal reduction order model. This means we minimize the following term:

$$\left\langle \int Nv(x)\mathrm{d}t - \int \hat{N}\hat{v}(\hat{x})\mathrm{d}t, \int Nv(x)\mathrm{d}t - \int \hat{N}\hat{v}(\hat{x})\mathrm{d}t \right\rangle \tag{4-8}$$

By using the fmincon function in Matlab we can determine the optimal case with the smallest deviation from the non-reduced case. Here we'll find the rate constants to be:

$$k_{1,new} = 0.0667 \tag{4-9}$$
$$k_{2,new} = 0.0333 \tag{4-10}$$
$$k_{3,new} = 0.05 \tag{4-11}$$
$$k_{4,new} = 0.075 \tag{4-12}$$

Then if we make a plot of both systems, i.e. the reduced and the non-reduced case, we directly see more overlap in the graphs for the species concentrations over time. Once again, here 'n new subsystem(s)' means that we change the first n subsystem(s) by its alternative forms. The smaller difference between the alternative model and non-reduced case can schematically be depicted in figure 4-4

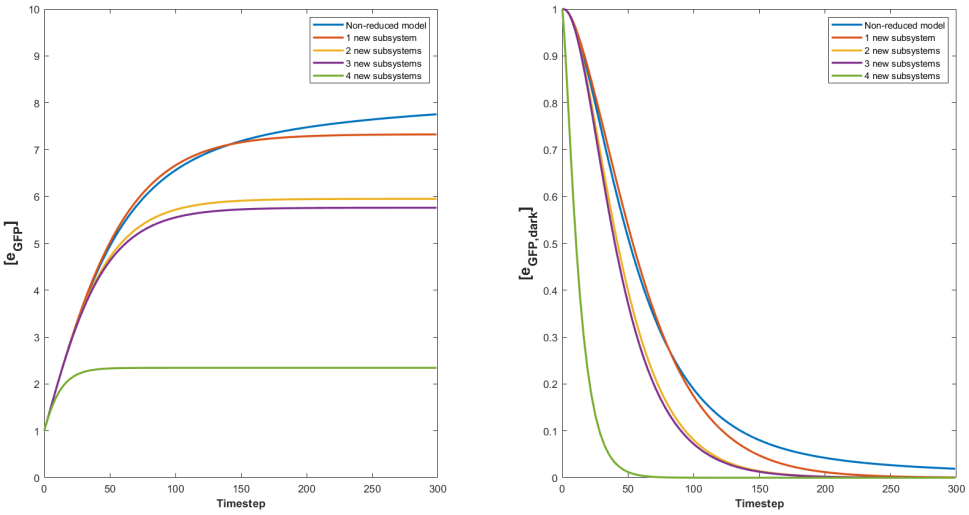


Figure 4-4: e_{GFP} and $e_{GFP,dark}$ concentrations for a change in up to four changed subsystems with optimal rate constants $k_{1,new}, k_{2,new}, k_{3,new}$ and $k_{4,new}$

Indeed, by changing one subsystem the systems are almost similar. The same holds for the system where 2 and 3 subsystems have been replaced. If there are four new subsystems it already begins to deviate a lot and the kinetics are more different. Though we see that a zero species concentration will be reached sooner if there are more changed subsystems. Since e_{GFP} is an output protein this doesn't hold for this species.

4-2-2 Introduction of Michaelis-Menten kinetics

To see if we can use Michaelis-Menten kinetics, we first want to see what kind of assumptions we need. Thereafter we will look at similar structures in our network of CFFLs. Michaelis-Menten kinetics are actually based on one particular set of reactions with product P, substrate S and enzyme E [22]:

$$\mathrm{S} \ + \ \mathrm{E} \underset{\mathrm{k_{-2}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \mathrm{SE} \xrightarrow{\mathrm{k_3}} \mathrm{E} \ + \ \mathrm{P} \qquad [17] \tag{4-13}$$

This can schematically be depicted as in figure (4-5):

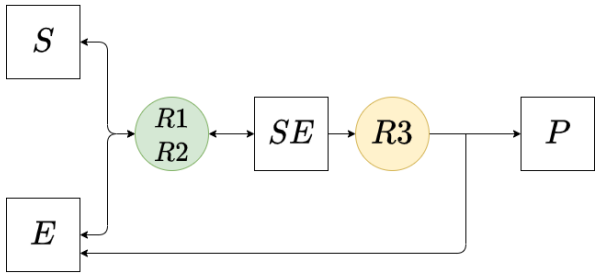


Figure 4-5: Michaelis-Menten kinetics

Then with the mass action kinetics, the conservation law and the assumption of instantaneous SE, we get:

$$\left.\begin{aligned} &[\dot{S}] = -k_1[S][E] + k_{-2}[SE] \\ &[\dot{E}] = -k_1[S][E] + k_{-2}[SE] + k_3[SE] \\ &[\dot{SE}] = k_1[S][E] - k_{-2}[SE] - k_3[SE] \approx 0 \\ &[\dot{P}] = k_3[SE] \\ &[SE] + [E] = e_0 \in \mathbb{R} \end{aligned}\right\} \Longrightarrow [\dot{P}] = \frac{V_{max}[S]}{K_M + [S]} \text{ with } K_M = \frac{k_{-2} + k_3}{k_1}, V_{max} = k_3 e_0 \tag{4-14}$$

Though the Michaelis-Menten structure of the given set of reactions isn't there in the first place [17].

Competitive Michaelis-Menten in [4] introduces a defect reaction for the enzyme E given by:

$$\mathrm{D} \ + \ \mathrm{E} \underset{\mathrm{k_{-4}}}{\overset{\mathrm{k_4}}{\rightleftharpoons}} \mathrm{DE} \tag{4-15}$$

Therefore we will get figure (4-6) with the corresponding flowdiagram:

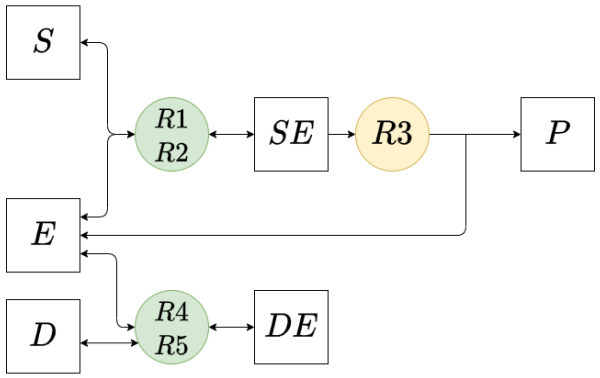


Figure 4-6: Michaelis-Menten kinetics with defectspecies D

This results in the following set of reactions and assumptions:

$$\left.\begin{aligned}
&[\dot{S}] = -k_1[S][E] + k_{-2}[SE] \\
&[\dot{E}] = -k_1[S][E] + k_{-2}[SE] + k_3[SE] + k_{-5}[DE] \\
&[\dot{SE}] = k_1[S][E] - k_{-2}[SE] - k_3[SE] \approx 0 \\
&[\dot{P}] = k_3[SE] \\
&[\dot{D}] = -k_4[D][E] + k_{-5}[DE] \\
&[\dot{DE}] = k_4[D][E] - k_{-5}[DE] \\
&[\dot{D}] + [\dot{DE}] = 0 \Longrightarrow [D] + [DE] = e_1 \in \mathbb{R}
\end{aligned}\right\} \Longrightarrow [\dot{P}] = K_M[S]\Big(\frac{e_1 - [D]}{k_4[D]}\Big) \text{ with } K_M = \frac{k_3 k_1 k_{-5}}{k_3 + k_{-2}} \tag{4-16}$$

4-2-3 More possible new fluxterms

For now consider the following example:

$$\mathrm{X}_1 \ + \ \mathrm{X}_2 \longrightarrow \mathrm{X}_3 \ + \ \mathrm{X}_4 \qquad [4] \tag{4-17}$$

For $i = 1. \ldots, 4$ let $\dot{x}_i$ denote the concentration of the species X_i and define $x := [x_1 x_2 x_3 x_4]^\top$. Here, the substrate complex $\mathcal{S}$ is $X_1 + X_2$ and the product complex is $X_3 + X_4$. Then we have:

$$B = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, Z_{\mathcal{S}} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad [4] \tag{4-18}$$

Let K_1, K_2, K_3 and K_4 denote the 'Michaelis' constants of the species X_1, X_2, X_3 and X_4 respectively. Let V_f denote the maximum rate of reaction (4-17). Then the type and mechanism of the reaction can be included by its corresponding choice of $v(x)$. [4] For inhibition we can choose the following flux:

$$v(x) = \frac{\left(\frac{V_f}{K_1 K_2}\right) x_1 x_2}{\left(1 + \frac{x_1}{K_1} + \frac{x_3}{K_3}\right)\left(1 + \frac{x_2}{K_2} + \frac{x_4}{K_4}\right)} \tag{4-19}$$

For now consider the following reaction:

$$\mathrm{X}_1 \longrightarrow \mathrm{X}_2 \tag{4-20}$$

Once more, denote the 'Michaelis' constant by K_1 and the maximum rate of reaction (4-20) by $V_{f,1}$. Let x_c and x_{nc} denote the concentrations of the competitive and non-competitive modifier. Let K_{nc} and K_c be the 'Michaelis' constants for the non-competitive and competitive modifier respectively. [4] Then in case of competitive and non-competitive reactions we can choose the following fluxes resp:

$$v_{nc}(x) = \frac{\frac{V_{f,1}}{K_1}}{\left(1 + \frac{x_1}{K_1}\right)\left(1 + \frac{x_{nc}}{K_{nc}}\right)} \tag{4-21}$$

$$v_c(x) = \frac{\frac{V_{f,1}}{K_1}}{1 + \frac{x_1}{K_1} + \frac{x_c}{K_c}} \tag{4-22}$$

4-2-4 Analysis of kinetics for new fluxterms

Now, we can check the subsystems accordingly and try to find an update-equation for the output species that relates to the so called Michaelis-Menten kinetics as discussed in the sections before. In subsystems 1 & 2 we get the following set of reactions:

$$\mathrm{X}_3 + \mathrm{X}_6 \xrightarrow{\mathrm{k}_3} \mathrm{X}_7 \xrightarrow{\mathrm{k}_4} \mathrm{X}_3 + \mathrm{X}_6 + \mathrm{X}_8 \tag{4-23}$$

$$\mathrm{X}_4 + \mathrm{X}_6 \xrightarrow{\mathrm{k}_5} \mathrm{X}_9 \xrightarrow{\mathrm{k}_6} \mathrm{X}_4 + \mathrm{X}_6 + \mathrm{X}_{10} \tag{4-24}$$

Then the kinetics of this subnetwork will read as follows (with the found conservation laws):

$$\dot{x}_3 = -k_3 x_3 x_6 + k_4 x_7 \tag{4-25}$$
$$\dot{x}_4 = -k_5 x_4 x_6 + k_6 x_9 \tag{4-26}$$
$$\dot{x}_6 = -k_3 x_3 x_6 - k_5 x_4 x_6 + k_4 x_7 + k_6 x_9 \tag{4-27}$$
$$\dot{x}_7 = k_3 x_3 x_6 - k_4 x_7 \tag{4-28}$$
$$\dot{x}_8 = k_4 x_7 \tag{4-29}$$
$$\dot{x}_9 = k_5 x_4 x_6 - k_6 x_9 \tag{4-30}$$
$$\dot{x}_{10} = k_6 x_9 \tag{4-31}$$
$$\text{Conservation laws:} \tag{4-32}$$
$$x_3 + x_7 = c_1 \in \mathbb{R} \tag{4-33}$$
$$x_4 + x_9 = c_2 \in \mathbb{R} \tag{4-34}$$
$$x_6 + x_7 + x_9 = c_3 \in \mathbb{R} \tag{4-35}$$

To show that the kinetics are Michaelis Menten for subsystems 1 & 2 we assume $\dot{x}_3 = 0$ and $\dot{x}_4 = 0$ respectively. Then we get:

$$\left.\begin{aligned} \dot{x}_3 = 0 \Longrightarrow k_3 x_3 x_6 = k_4 x_7 \Longrightarrow k_3(c_1 - x_7)x_6 = k_4 x_7 \\ \dot{x}_4 = 0 \Longrightarrow k_5 x_4 x_6 = k_6 x_9 \Longrightarrow k_5(c_2 - x_9)x_6 = k_6 x_9 \end{aligned}\right\} \Longrightarrow x_7 = \frac{k_3 c_1 x_6}{k_4 + k_3 x_6}, \quad x_9 = \frac{k_5 c_2 x_6}{k_6 + k_5 x_6}$$

Then this in turn will give us two Michaelis Menten kinetic terms:

$$\dot{x}_8 = k_4 x_7 = \frac{k_4 k_3 c_1 x_6}{k_4 + k_3 x_6} \tag{4-36}$$
$$\dot{x}_{10} = k_6 x_9 = \frac{k_6 k_5 c_2 x_6}{k_6 + k_5 x_6} \tag{4-37}$$

So we can conclude that subsystem 1 & 2 can actually be seen as one substem from x_6 to x_{10} and x_8.

For subsystem 3 we can apply a similar trick to show that we have to do with Michaelis-Menten kinetics. For this we have the following reactions within our subsystem:

$$\mathrm{x_{11} + x_5 \xrightarrow{k_9} x_{21} \xrightarrow{k_{10}} x_5 + x_{12}} \tag{4-38}$$

Then we get the following first ODEs and derivations:

$$\left.\begin{aligned} &\dot{x}_5 = -k_9 x_5 x_{11} + k_{10} x_{21} \\ &\dot{x}_{11} = -k_9 x_5 x_{11} \\ &\dot{x}_{12} = k_{10} x_{21} \\ &\dot{x}_{21} = k_9 x_5 x_{11} - k_{10} x_{21} \\ &\text{Conservation laws:} \\ &x_5 + x_{21} = c_1 \in \mathbb{R} \\ &x_{11} + x_{12} + x_{21} = c_2 \in \mathbb{R} \end{aligned}\right\} \Longrightarrow \begin{array}{c} \text{QSS Approach/ instantaneous equilibrium} \\ -k_9 x_5 x_{11} + k_{10} x_{21} = 0 \iff \\ -k_9(c_1 - x_{21})x_{11} + k_{10} x_{21} = 0 \iff \\ k_9 c_1 x_{11} = k_{10} x_{21} + k_9 x_{21} x_{11} \iff \\ x_{21} = \frac{k_9 c_1 x_{11}}{k_{10} + k_9 x_{11}} \end{array}$$

Thus $\dot{x}_{12} = k_{10}\frac{k_9 c_1 x_{11}}{k_{10}+k_9 x_{11}}$ which shows that we have to do with Michaelis Menten kinetics for reaction $R_{3,new}$.

For subsystem 4 we can apply a similar trick to show that we have to do with Michaelis-Menten kinetics. For this we have the following reactions in our subsystem:

$$\mathrm{X_{13}} + \mathrm{X_{14}} \xrightarrow{\mathrm{k_{13}}} \mathrm{X_{15}} \xrightarrow{\mathrm{k_{14}}} \mathrm{X_{16}} + \mathrm{X_{13}} \tag{4-39}$$

$$\mathrm{X_{17}} + \mathrm{X_{13}} \xrightarrow{\mathrm{k_{17}}} \mathrm{X_{18}} \xrightarrow{\mathrm{k_{18}}} \mathrm{X_{13}} + \mathrm{X_{19}} + \mathrm{X_{17}} \tag{4-40}$$

Then the mass-action kinetics read as:

$$\left.\begin{aligned}
&\dot{x}_{13} = k_{14}x_{15} - k_{13}x_{13}x_{14} + k_{18}x_{18} - k_{17}x_{17}x_{13}\\
&\dot{x}_{15} = -k_{14}x_{15} + k_{13}x_{13}x_{14}\\
&\dot{x}_{16} = k_{14}x_{15}\\
&\dot{x}_{17} = k_{18}x_{18} - k_{17}x_{13}x_{17}\\
&\dot{x}_{18} = -k_{18}x_{18} + k_{17}x_{17}x_{13}\\
&\dot{x}_{19} = k_{18}x_{18}\\
&\text{Conservation laws:}\\
&x_{13} + x_{15} + x_{18} = c_1 \in \mathbb{R}\\
&x_{17} + x_{18} = c_2 \in \mathbb{R}
\end{aligned}\right\} \begin{array}{c}\text{QSS Approach/ instantaneous equilibrium}\\ k_{18}x_{18} = k_{17}x_{17}x_{13} \iff\\ k_{18}x_{18} = k_{17}(c_2 - x_{18})x_{13} \iff\\ x_{18} = \frac{k_{17}c_2x_{13}}{k_{18}+k_{17}x_{13}}\end{array}$$

Then this in turn will give us the Michaelis-Menten kinetic term for the reaction where species x_{13} will be consumed and x_{19} will be produced:

$$\dot{x}_{19} = k_{18}x_{18} = k_{18}\frac{k_{17}c_2x_{13}}{k_{18}+k_{17}x_{13}} \tag{4-41}$$

To verify that the system indeed has to do with Michaelis Menten kinetics we made a plot of $\dot{x}_8$ vs $x_6, \dot{x}_{10}$ vs $x_6, \dot{x}_{12}$ vs x_{11} and $\dot{x}_{19}$ vs x_{13} respectively. To show a saturation for the curves representing the MM-kinetics we will have to change our model a bit. Assume that $k_3 = k_4 = k_5 = k_6 = k_9 = k_{10} = k_{17} = k_{18} = \frac{1}{1000}$. For now we make a projection where we match the corresponding timesteps with the concentrations of the desired species. If we have a look at the concentrations of the found species at the same timesteps for the corresponding $\dot{x}$ we can plot the trajectories corresponding to the Michaelis-Menten kinetics. The Michaelis-Menten relations for the subsystems have been depicted in figure (4-7).

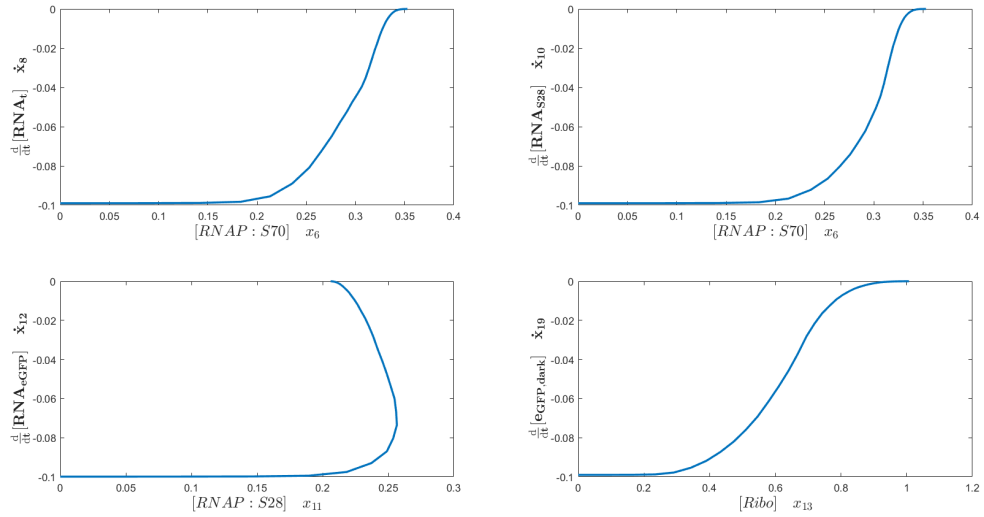


Figure 4-7: Shown Michaelis Menten kinetics for the predescribed subsystems

Here we see that the plot of $\dot{x}_{12}$ against x_{11} shows a different behavior. This is mainly due to the fact that we have to keep in mind that the full model can change the way in which the subsystems react with one another.

4-2-5 Determine V_{max} for Michaelis Menten kinetics

Summarized we have the following defined laws:

$$\dot{x}_8 = k_4 \frac{k_3 c_1 x_6}{k_4 + k_3 x_6} = \frac{2x_6}{1000 + 1000x_6} \tag{4-42}$$

$$\dot{x}_{10} = k_6 \frac{k_5 c_2 x_6}{k_6 + k_5 x_6} = \frac{2x_6}{1000 + 1000x_6} \tag{4-43}$$

$$\dot{x}_{12} = k_{10} \frac{k_9 c_1 x_{11}}{k_{10} + k_9 x_{11}} = \frac{2x_{11}}{1000 + 1000x_{11}} \tag{4-44}$$

$$\dot{x}_{19} = k_{18} \frac{k_{17} c_2 x_{13}}{k_{18} + k_{17} x_{13}} = \frac{2x_{13}}{1000 + 1000x_{13}} \tag{4-45}$$

Then for all these subsystems we will have a steady state $V_{max} = \frac{2}{1000} \approx 0$. This is equal to the following limits:

$$V_{max} = \lim_{x_6 \to \infty} \dot{x}_8 = \lim_{x_6 \to \infty} \dot{x}_{10} = \lim_{x_{11} \to \infty} \dot{x}_{12} = \lim_{x_{13} \to \infty} \dot{x}_{19} = \frac{2}{1000}. \tag{4-46}$$

4-2-6 Use of Michaelis-Menten kinetics for changed subsystems

With the analysis we did, we know that the changed subsystems within our network of CFFLs fit best with Michaelis-Menten kinetics. Therefore we extend our N matrix and add the appropriate new flux term with MM-kinetics. For now, we denote '1 changed subsystem' as the system where the first subsystem has been replaced by its alternative model, we denote '2 changed subsystems' as the system where the first and second subsystem have been replaced by its alternative models. This will be similar for '3 changed subsystems' and '4 changed subsystems'. By doing this, we will get the following results as given in figure (4-8):

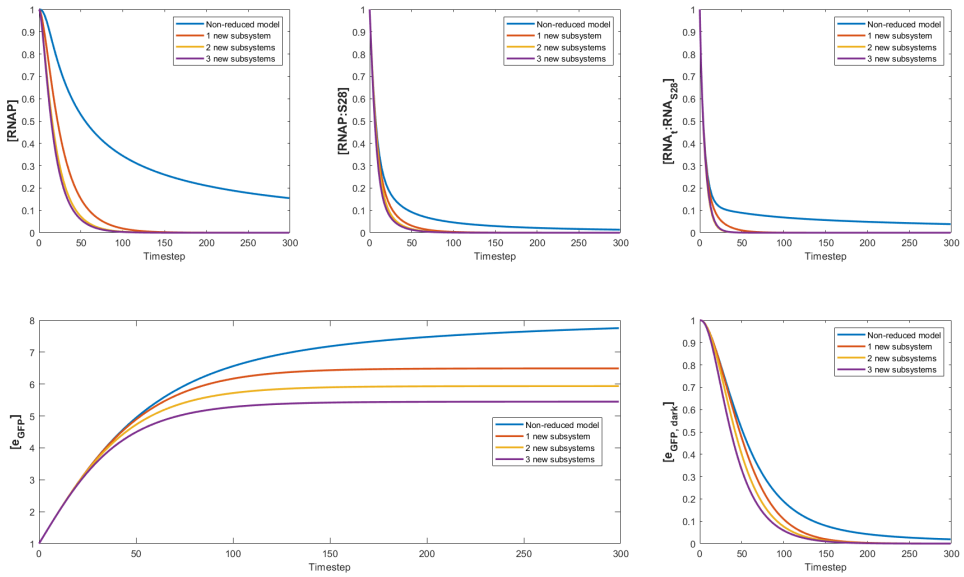


Figure 4-8: Model of CFFLs based on three types of changed subsystems with Michaelis-Menten kinetics

Indeed, now the systems will be much more similar with respect to the original case and it can be concluded that using MM-kinetics is a better way to find a fitting model for our CFFL network. Note that one wants to compare figures (4-8) with figures (4-4) and (4-3). Here it is remarkable that the alternative model, where (optimized) mass-action kinetics have been used, shows a better result for 'one changed subsystem' compared to the case where Michaelis-Menten kinetics have been used. Though if one changes more subsystems within the network, Michaelis-Menten kinetics will be a much more favourable alternative modelling strategy than (optimized) mass-action kinetics.

4-3 Conclusions of Alternative Modelling

For the network of CFFLs we can replace certain cycles and feedbackmechanisms or pathways by more simple structures. Here we need to take renewed rate constants into account such that the reduced and non-reduced case will only have small deviations. Besides one can show that certain replaced feedback mechanisms are of the Michaelis Menten form.

As written before, the kinetics will mostly be based on mass-action but if we can make more assumptions we are also allowed to use different kinetics like Michaelis-Menten. Here we will get different flux terms for the entries of the vector $v_{new}(x)$. It was shown that Michaelis-Menten kinetics fit best for the given (replaced) system of reactions since the ODEs for the output concentrations show saturations.

4-4 Discussion of Alternative Modelling

4-4-1 Advantages of Alternative Modelling

Alternative modelling will have the following advantages:

- With Alternative Modelling one can simplify the system or network of CFFLs without changing the original kinetics too much. This is realised by the comparison of the non-reduced system of CFFLs with the Alternative Model.
- One will get more insight into the most important species/nodes/complexes involved within the system of CFFLs

4-4-2 Challenges of Alternative Modelling

The challenge of Alternative Modelling will be to reduce the system in order without losing too much information of the original CFFL network. Then one could argue whether it is still realistic if more and more subsystems will be removed from the full system of CFFLs. On the other hand it will gain us more information of the importance of a species or complex within our network of CFFLs.

Chapter 5

Kron reduction order method

The Kron reduction order method will introduce a new way to reduce the system in order. This will be based on removing complexes in the first place where it will influence the internal and eventually the external kinetics. To understand this way of model order reduction we will first start with an illustration of the simple case.

5-1 Simple case of Kron reduction order methods

Consider the following reactions:

$$\mathrm{A} \ + \ \mathrm{B} \ \underset{\mathrm{k_{-1}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \ \mathrm{C} \xrightarrow{\mathrm{k_2}} \mathrm{A} \tag{5-1}$$

Here define the constants as: $k_1 = \frac{1}{2}, k_{-1} = \frac{1}{5}, k_2 = \frac{1}{10}$.

5-1-1 Model 1: Original law of mass action

According to the law of mass action we get the following kinetics for the concentrations of A, B and C (denoted by their non-capital letters) :

$$\begin{cases} \dot{a} = -k_1 ab + k_{-1} c + k_2 c \\ \dot{b} = -k_1 ab + k_{-1} c \\ \dot{c} = +k_1 ab - k_2 c - k_{-1} c \end{cases} \tag{5-2}$$

5-1-2 Model 2: Kron method

For this method define the complexes as: $C_1 = A + B, C_2 = A, C_3 = C$. Suppose we have to do with biochemical reaction (5-1). Define the speciesvector by: $x = \begin{pmatrix} a & b & c \end{pmatrix}^\top$. Then the kinetics are given by the vector

$$\dot{x} = -ZL \text{ Exp } (Z^\top \text{ Log } x) \qquad [4] \tag{5-3}$$

$$\text{with: } Z = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad L = D - A_{adj} = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_2 + k_{-1} \end{pmatrix} - \begin{pmatrix} 0 & 0 & k_{-1} \\ 0 & 0 & k_2 \\ k_1 & 0 & 0 \end{pmatrix} \tag{5-4}$$

Here Z is the $m \times c$ matrix that links the species with the complexes, the degree matrix D is the $c \times c$ matrix that contains the outer degrees for every complex (as node) on its diagonal and the adjacency matrix A_{adj} is the $c \times c$ matrix that contains the degrees from complex to complex (columns read as 'from complex' and rows read as 'to complex'). Note that the degree matrix has diagonal entries equal to the sum of the adjacency entries per column [4]

Now equations (5-2) and (5-3) are fully identical and it can be depicted as:

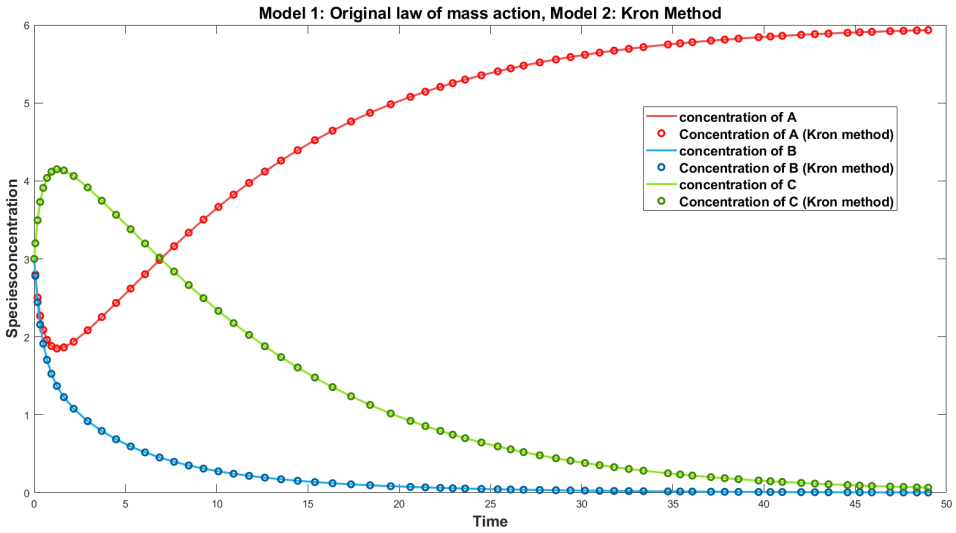


Figure 5-1: Identical model 1 & 2 with resp. the original law of mass action and the applied method of Kron

As expected the concentration of A increases and simultaneously the concentrations for B and C reduces to 0 where the steady state is reached.

5-1-3 Model 3: Kron reduction order method

Define the reduced order system by:

$$\dot{x} = -Z_{new}\hat{L}\ \text{Exp}\ (Z_{new}^{\top}\ \text{Log}\ (x)) \tag{5-5}$$

with

$$Z_{new} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and} \tag{5-6}$$

$$L = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} = \left[\begin{array}{cc|c} k_1 & 0 & -k_{-1} \\ 0 & 0 & -k_2 \\ \hline -k_1 & 0 & k_2 + k_{-1} \end{array}\right] \tag{5-7}$$

$$\hat{L} = L_{11} - L_{12}L_{22}^{-1}L_{21}\ \text{(Schur Complement)} \tag{5-8}$$

Actually here we removed complex C_3 as an internal node from the network. So actually we remove complexes and redefine the weight of the edges by $\hat{L}$. [4] As a check, the rows in the Laplacian matrix add to zero. Now the concentrations of A, B and C can be depicted as in figure (5-2):

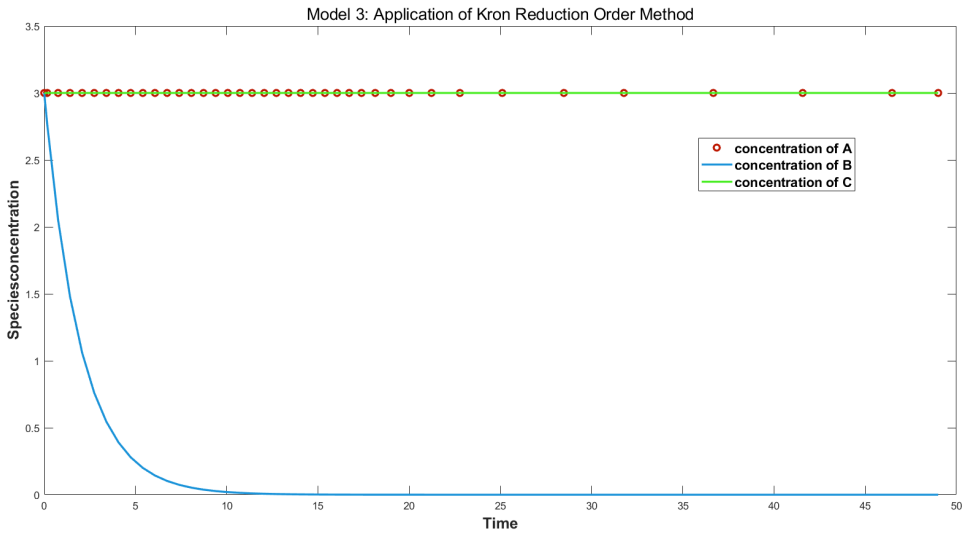


Figure 5-2: Model 3 with applied Kron reduction order method

Note: The reduced order system already deviates quite a lot from the non-reduced order system. Besides since the system is not complex-balanced, from [15] it follows that the steady states of the reduced and non-reduced systems can only be identical for some subsystems.

5-2 Complex balanced set of biochemical reactions

A set of biochemical reactions is detailed balanced if and only if the reactions within the CFFL network are reversible with positive rate constants. This is the stronger definition with respect to the so called complex balancedness systems. For complex balancedness we will need that the flux rates or rate constants counteracts each other within the system of reactions [15].

So we say shortly that complex balancedness mainly boils down to the fact that the sum of reaction rates involved in reactions producing the complex are equal to the sum of reaction rates involved in reactions where the complex will be consumed.

If we look at the previous example one can e.g. replace the latter single rightdirectional arrow by a bidirectional reaction to realise the detailed balanced reaction. This means we get the following set of biochemical reactions:

$$\mathrm{A} \ + \ \mathrm{B} \ \underset{\mathrm{k_{-1}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \ \mathrm{C} \ \underset{\mathrm{k_{-2}}}{\overset{\mathrm{k_2}}{\rightleftharpoons}} \ \mathrm{A} \tag{5-9}$$

Now the mass-action kinetics are:

$$\begin{cases} \dot{a} = -k_1 ab + k_{-1} c + k_2 c - k_{-2} a \\ \dot{b} = -k_1 ab + k_{-1} c \\ \dot{c} = +k_1 ab - k_2 c - k_{-1} c + k_{-2} a \end{cases} \tag{5-10}$$

Since we have a detailed balanced system we also know that the system is complex balanced. [15] For example here we choose $k_{-2} = \frac{1}{10}$.

Z will be the same since the definitions of the complexes hasn't been changed. D and A are defined as:

$$D = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_{-2} & 0 \\ 0 & 0 & k_2 + k_{-1} \end{pmatrix}, A = \begin{pmatrix} 0 & 0 & k_{-1} \\ 0 & 0 & k_2 \\ k_1 & k_{-2} & 0 \end{pmatrix} \tag{5-11}$$

Figure (5-3) depicts the model in which the the law of mass action has been applied and figure (5-4) depicts the states in the reduced order system (where complex C_2 has been removed): :

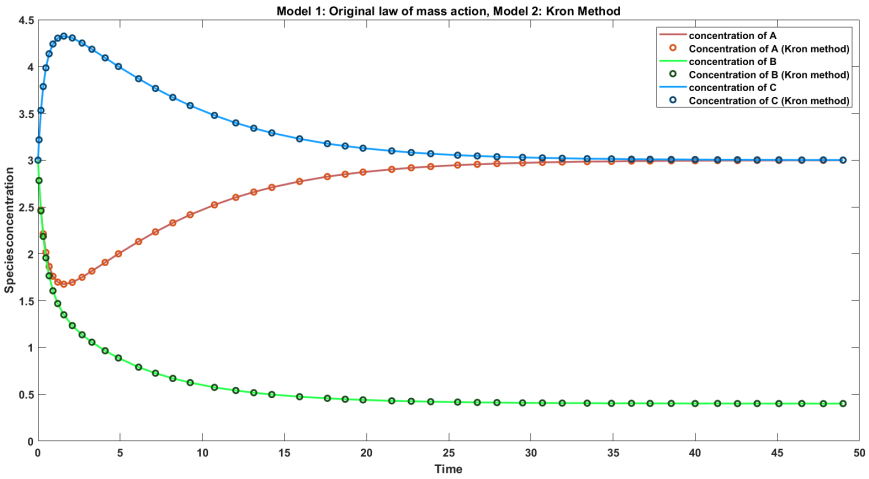


Figure 5-3: Kron method along with the original law of mass action models

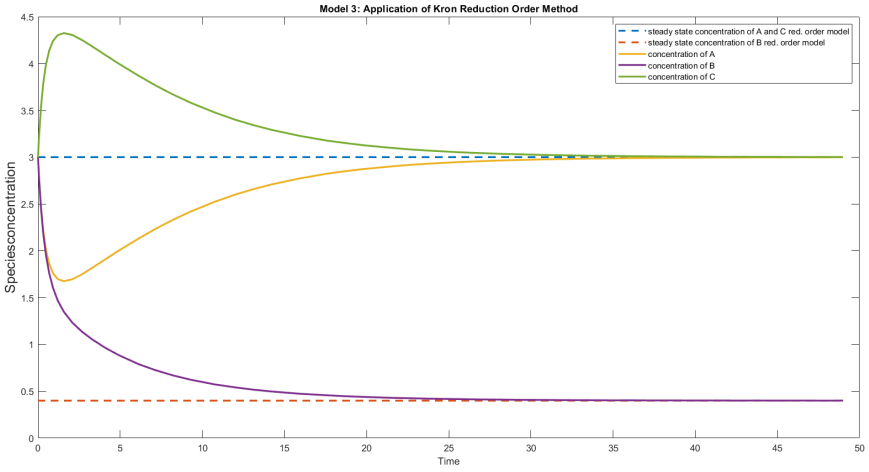


Figure 5-4: Applied Kron reduction order method with the steady-states of the reduced order system

Now if the Kron reduction order method has been applied on this set of biochemical equations, the steady-states of the non-reduced and reduced order system are exactly the same. It can be proven that these two systems are always the same in case of complex balanced biochemical reactions. [15] Figure (5-4) indeed illustrates this phenomenon.

5-3 Kron reduction order method applied in a set of chemical reactions

To apply the Kron reduction order method we start by transforming the set of reactions into the corresponding set of complexes since this method is based on reducing the number of complexes.[15] By doing so we will have the following distinct complexes within our biochemical network:

C_1	$X_1 + X_2$	C_{10}	X_1	C_{19}	X_{12}
C_2	X_6	C_{11}	$X_{11} + X_5$	C_{20}	$X_8 + X_{12}$
C_3	$X_6 + X_3$	C_{12}	X_{20}	C_{21}	$X_{10} + X_8$
C_4	$X_{13} + X_{17} + X_{19}$	C_{13}	$X_5 + X_{12}$	C_{22}	X_{18}
C_5	$X_8 + X_3 + X_6$	C_{14}	$X_{17} + X_{13}$	C_{23}	X_7
C_6	$X_6 + X_4$	C_{15}	X_{10}	C_{24}	X_9
C_7	X_{19}	C_{16}	$X_{14} + X_{13}$	C_{25}	X_{21}
C_8	$X_{10} + X_4 + X_6$	C_{17}	X_8	C_{26}	X_{14}
C_9	$X_{11} + X_{16}$	C_{18}	$X_{16} + X_{13}$	C_{27}	X_{15}
				C_{28}	X_{17}

Table 5-1: Complexes within the biochemical circuit with species $X_1, \ldots, X_{22}$

The set of biochemical reactions as before can now be written in terms of complexes as:

R1	$C_1 \xrightarrow{k_1} C_2$	R9	$C_{11} \xrightarrow{k_9} C_{25}$	R17	$C_{14} \xrightarrow{k_{17}} C_{22}$
R2	$C_2 \xrightarrow{k_{-2}} C_1$	R10	$C_{25} \xrightarrow{k_{10}} C_{13}$	R18	$C_{22} \xrightarrow{k_{18}} C_4$
R3	$C_3 \xrightarrow{k_3} C_{23}$	R11	$C_{21} \xrightarrow{k_{11}} C_{26}$	R19	$C_7 \xrightarrow{k_{19}} C_{12}$
R4	$C_{23} \xrightarrow{k_4} C_5$	R12	$C_{26} \xrightarrow{k_{-12}} C_{21}$	R20	$C_{26} \xrightarrow{k_{deg}} \emptyset$
R5	$C_6 \xrightarrow{k_5} C_{24}$	R13	$C_{16} \xrightarrow{k_{13}} C_{27}$	R21	$C_{17} \xrightarrow{k_{deg}} \emptyset$
R6	$C_{24} \xrightarrow{k_6} C_8$	R14	$C_{27} \xrightarrow{k_{14}} C_{18}$	R22	$C_{28} \xrightarrow{k_{deg}} \emptyset$
R7	$C_9 \xrightarrow{k_7} C_{10}$	R15	$C_{20} \xrightarrow{k_{15}} C_{28}$	R23	$C_{15} \xrightarrow{k_{deg}} \emptyset$
R8	$C_{10} \xrightarrow{k_{-8}} C_9$	R16	$C_{28} \xrightarrow{k_{-16}} C_{20}$	R24	$C_{19} \xrightarrow{k_{deg}} \emptyset$

Table 5-2: Biochemical reactions in terms of complexes $C_1, \ldots, C_{28}$

Within this biochemical network of complexes we have to do with the set of coherent complex loops. If we consider the rate constants to be similar, we can reduce this network of complexes directly as given in table (5-3) below:

Network of complexes	Network in reduced order form
$C_1 \underset{k_{-2}}{\overset{k_1}{\rightleftharpoons}} C_2$	$C_1 \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C_2$
$C_3 \xrightarrow{k_3} C_{23} \xrightarrow{k_4} C_5$	$C_3 \xrightarrow{k_{n,1}} C_5$
$C_6 \xrightarrow{k_5} C_{24} \xrightarrow{k_6} C_8$	$C_6 \xrightarrow{k_{n,2}} C_8$
$C_9 \underset{k_{-8}}{\overset{k_7}{\rightleftharpoons}} C_{10}$	$C_9 \underset{k_{-8}}{\overset{k_7}{\rightleftharpoons}} C_{10}$
$C_{11} \xrightarrow{k_9} C_{25} \xrightarrow{k_{10}} C_{13}$	$C_{11} \xrightarrow{k_{n,3}} C_{13}$
$C_{21} \underset{k_{-12}}{\overset{k_{11}}{\rightleftharpoons}} C_{26} \xrightarrow{k_{deg}} \emptyset$	C_{21} / $C_{26} \xrightarrow{k_{deg,1}} \emptyset$
$C_{16} \xrightarrow{k_{13}} C_{27} \xrightarrow{k_{14}} C_{18}$	$C_{16} \xrightarrow{k_{n,4}} C_{18}$
$C_{20} \underset{k_{-16}}{\overset{k_{15}}{\rightleftharpoons}} C_{28} \xrightarrow{k_{deg}} \emptyset$	C_{20} / $C_{28} \xrightarrow{k_{deg,2}} \emptyset$
$C_{14} \xrightarrow{k_{17}} C_{22} \xrightarrow{k_{18}} C_4$	$C_{14} \xrightarrow{k_{n,5}} C_4$
$C_7 \xrightarrow{k_{19}} C_{12}$	$C_7 \xrightarrow{k_{19}} C_{12}$
C_{15} , C_{17}, $C_{19} \xrightarrow{k_{deg}} \emptyset$	C_{15} , C_{17}, $C_{19} \xrightarrow{k_{deg,3}} \emptyset$

Table 5-3: Network of complexes with the corresponding network according to linkage classes

Note: Here C_{20} (or C_{28}) degrades with flux rate $k_{deg,2}$ since it is assumed that reactions R14 and R15 are balanced (thus $k_{14} = k_{-15}$). Besides C_{21} (or C_{26}) degrades with flux rate $k_{deg,1}$ since it is assumed that reactions R11 and R12 are balanced (thus $k_{11} = k_{-12}$). Now we can transform the model of complexes into the model of species concentrations and vice versa by using the given relation between them. To make the appropriate decisions for removing the given complexes or to determine the correct (new defined) rate constants more analysis will be needed in general.

5-4 Link between complexes- and species network

To find the relation between the complexes- and species network we will define some new constants. Here the number of species m, the number of complexes c and the number of reactions r are now respectively equal to 22, 28 and 24. By linking the reactions with the complexes we get the complex Stoichiometry matrix $Z \in \mathbb{R}^{m \times c}$ and by linking the complexes with the species we get the incidence matrix $B \in \mathbb{R}^{c \times r}$ (see Appendix). According to literature it holds that the Stoichiometry matrix $N = ZB$. [16] Now we are particular interested in the substrate complexes: i.e. the complexes in front of every reaction. [4] The substrate complexes are thus given by complexes $C_1, C_3, C_{23}, C_6, \ldots, C_{20}$ as in table (5-2).

Then we are allowed to partition $Z_{\mathcal{S}}$ as

$$Z_{\mathcal{S}} = \begin{pmatrix} \vdots & & \vdots \\ Z_{\mathcal{S}_1} & \cdots & Z_{\mathcal{S}_c} \\ \vdots & & \vdots \end{pmatrix} \tag{5-12}$$

The corresponding flux rate vector $v(x)$ reads as:

$$v(x) = \begin{pmatrix} k_1 x_1^{Z_{\mathcal{S}_{1,1}}} & \cdots & x_m^{Z_{\mathcal{S}_{c,1}}} \\ & \vdots & \\ k_c x_1^{Z_{\mathcal{S}_{1,r}}} & \cdots & x_m^{Z_{\mathcal{S}_{c,m}}} \end{pmatrix} \qquad [4] \tag{5-13}$$

$$\begin{aligned} \text{Note: } x_1^{Z_{\mathcal{S}_{1,1}}} \cdots x_m^{Z_{\mathcal{S}_{c,1}}} &= \text{Exp}\,(Z_{\mathcal{S}_{1,1}} Ln(x_1)) \cdots \text{Exp}\,(Z_{\mathcal{S}_{c,1}} Ln(x_m)) \\ &= \text{Exp}\,(\sum_{k=1}^{m} Z_{\mathcal{S}_{k,1}} Ln(x_k)) \\ &= \text{Exp}\,(Z_{\mathcal{S}_1}^{\top} Ln(x)) \end{aligned} \tag{5-14}$$

Since equation (5-14) holds, it follows that (5-13) is equivalent to:

$$v(x) = \begin{pmatrix} k_1\ \text{Exp}\,(Z_{\mathcal{S}_1} \text{Ln}(x)) \\ \vdots \\ k_c\ \text{Exp}\,(Z_{\mathcal{S}_c} \text{Ln}(x)) \end{pmatrix} \tag{5-15}$$

$$= K\ \text{Exp}\,(Z_s^{\top} Ln(x)) \qquad [4] \tag{5-16}$$

Then we get:

$$v(x) = K\ \text{Exp}\,(Z_{\mathcal{S}}^{\top} Ln(x)) \tag{5-17}$$

$$\implies \dot{x} = -ZL(x)\ \text{Exp}\,(Z^{\top} Ln(x)) \qquad [4] \tag{5-18}$$

where the matrix K has been defined as:

$$K := \begin{pmatrix} k_1 & 0 & \cdots & \cdots & 0 \\ 0 & k_2 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & k_{c-1} & 0 \\ 0 & \cdots & \cdots & 0 & k_c \end{pmatrix} \tag{5-19}$$

Here the matrices will have the following dimensions:

$$\begin{array}{ccc} Z_{\mathcal{S}} : r \times m & K : c \times c & Z : r \times c \\ x : m \times 1 & L = -DK : c \times c & D : c \times c \end{array}$$

It was chosen to compute the given Laplacian with an analogous method. Therefore we can define the Laplacian matrix $L = D - A$ where D the degree matrix and A the adjacency matrix given by respectively :

$$A(i,j) := \text{Defined rate constant } k_{ij} \text{ for reaction : } \mathrm{C}_i \xrightarrow{\mathrm{k_{ij}}} \mathrm{C}_j.$$

$$D(i,j) := \begin{cases} 0, \text{ if complex } C_i \text{ won't be formed with other complexes and } i = j \\ 0, \text{ if } i \neq j \\ \sum_l k_l, \text{if complex } C_i \text{ has an outer degree given by flux rate/rates } k_l \end{cases}$$

Note that D actually is a $c \times c$ diagonal matrix where the corresponding entry is the sum of the corresponding column elements of A.

(*For the full matrices L and B see appendix* (A-4) *and* (A-1))

Note that the dimensions for x won't change and the m species still remain if the number of complexes c will be reduced. Most of all it is important to keep in mind that certain complexes are linked to each other since some complexes are additions of more species even though this is invisible within the network of complexes.

5-5 Removing complexes according to linkage classes

Since we have already arranged the complexes in such a way that the removable complexes are ordered from C_{20} to C_{28}, we can remove the following complexes respectively: $C_{28}, C_{27}, \ldots, C_{20}$. This meets the chosen method of the Kron reduction to select the appropriate submatrix L_{11}. For this , note that the matrix L is partitioned as:

$$L = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \qquad [4] \tag{5-20}$$

$$L_{11} \in \mathbb{R}^{(28-c)} \times \mathbb{R}^{(28-c)} \qquad L_{21} \in \mathbb{R}^{c} \times \mathbb{R}^{(28-c)} \tag{5-21}$$

$$L_{12} \in \mathbb{R}^{(28-c)} \times \mathbb{R}^{c} \qquad L_{22} \in \mathbb{R}^{c} \times \mathbb{R}^{c} \tag{5-22}$$

Here the number of removed complexes can be varied from $c = 0$ till $c = 9$ and is contained in L_{12}, L_{21} and L_{22}.

Just like the simple case, the Kron method and the model of law of mass action are fully identical. The reduction order method is based on (5-5) again. In figure 5-5 the reduction order models are given with the species concentration of the original model:

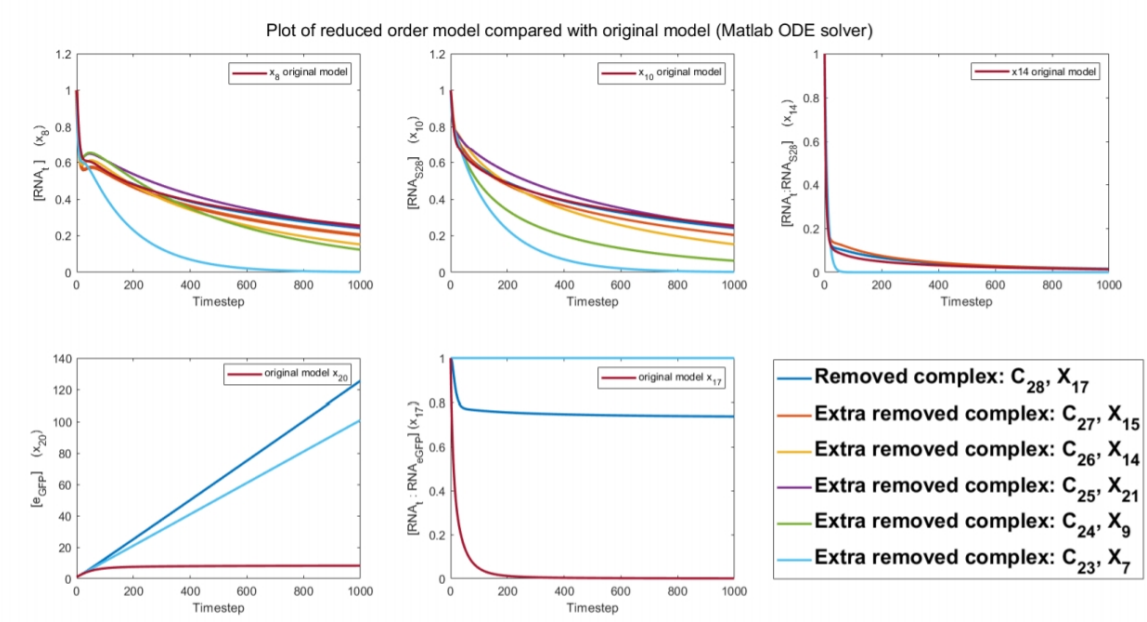


Figure 5-5: Original model compared with corresponding reduction order models that exclude the networkcomplexes according to the linkage of networkclasses

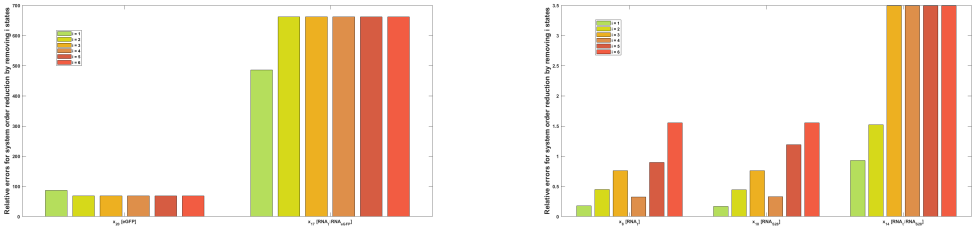
As can be concluded from figure (5-5) a higher order reduction method doesn't have to imply a worse reduction order model w.r.t. the lower order reduction models since the complexes are interconnected somehow. Note that some trajectories will overlap somehow, but we can better visualise this in the graph of RMS errors since two overlapping trajectories will give the same error. One can figure out that the following holds (where i is the reduction order):

- For state x_{20} the blue line is the plot for $i = 1$. The blue line (light) is the plot for $i = 2, \ldots, 6$.
- For state x_{17} the blue line (light) is the plot for $i = 1$. The blue line is the plot for $i = 2, \ldots, 6$.
- For state x_{14} the blue line is the plot for $i = 1$. The orange line is the plot for $i = 2$. The blue line (light) will be the plot for $i = 3, \ldots, 6$.

Furthermore note that the optimal case of order reduction will show more trajectories close to the non-reduced case as we will see later on.

In fact, here the following states for this non-optimal case will have the same steady-states: $X_1, X_6, X_8, X_{10}, X_{11}, X_{12}, X_{14}$. So if we define the subsystem as these states, it is a matter of time until it reaches the same species concentration. If we look at the flowscheme mainly the species with the given complexes that haven't been removed and aren't contained within a feedback loop will be part of this subsystem. This can also be seen in table (B-3). Here the absolute errors between the steady states in the reduced order model have been compared with the steady states in the non-reduced order model. $x_{red,ss,i}$ is the steady state of the ith reduction order model. If this is zero for all i it can be concluded that the steady states are the same (at least) until applying the sixth order method of Kron.

As can be verified, figure (5-6a) and (5-6b) show similar results. Here the number of removed complexes is independent of the relative RMS error for states $x_8, x_{10}, x_{14}, x_{20}$ and x_{17}.



(a) Relative RMS errors for states X_{20}, X_{17} **(b)** Relative RMS errors for states X_8, X_{10}, X_{14}

Figure 5-6: Relative RMS errors (within an interval of 10^5 timesteps) for the given states for models that have been reduced in order by $i = 1, \ldots, 6$

Mostly if the incoherent complexes will be removed it counteracts the first removal effect and lowers the error between the reduced and non-reduced model. The error seems to be much higher if one will look at the output species like eGFP. Moreover for some smaller order models the models coincide.

5-6 Sequence of complex removals

The full system (in terms of complexes) can thus be divided in two parts. One part will contain the reactions where we won't remove the complex whereas the other part will consist of the other reactions. The reactions without removable and removable complexes will read as follows:

Reactions without removable complexes	Reactions with removable complexes
$C_1 \underset{k_{-2}}{\overset{k_1}{\rightleftharpoons}} C_2$	$C_{14} \xrightarrow{k_{17}} C_{i,1} \xrightarrow{k_{18}} C_4$
$C_9 \underset{k_{-8}}{\overset{k_7}{\rightleftharpoons}} C_{10}$	$C_3 \xrightarrow{k_3} C_{i,2} \xrightarrow{k_4} C_5$
$C_7 \xrightarrow{k_{19}} C_{12}$	$C_6 \xrightarrow{k_5} C_{i,3} \xrightarrow{k_6} C_8$
$C_{15}\ ,\ C_{17},\ C_{19} \xrightarrow{k_{deg}} \emptyset$	$C_{11} \xrightarrow{k_9} C_{i,4} \xrightarrow{k_{10}} C_{13}$
	$C_{i,9} \underset{k_{-12}}{\overset{k_{11}}{\rightleftharpoons}} C_{i,5} \xrightarrow{k_{deg}} \emptyset$
	$C_{16} \xrightarrow{k_{13}} C_{i,6} \xrightarrow{k_{14}} C_{18}$
	$C_{i,8} \underset{k_{-16}}{\overset{k_{15}}{\rightleftharpoons}} C_{i,7} \xrightarrow{k_{deg}} \emptyset$

Table 5-4: Reactions without and with removable complexes $C_{i,1}, \ldots, C_{i,9}$

5-6-1 Rearrangement of complexes

For now, let c be the vector complexes and let c' be the relabeled vector complexes. Then there exists a permutation matrix π such that $c' = \pi Z$. This means that $x = Zc = Z(\pi^\top c') = Z'c'$. For this matrix it holds that $\pi^\top = \pi^{-1}$. One can also use this permutation matrix to find the degree matrix D, adjacency matrix A and complex Stoichiometry matrix Z in case of this coordinatetransformation.This change in matrices Z, D and A will then be identical to the following permutations:

- The complex Stoichiometry matrix Z will interchange the corresponding columns.
- The degree matrix D will interchange the corresponding elements on the diagonal of its matrix.
- The adjacency matrix A will interchange both the corresponding columns as well as the corresponding rows.

Now the sequence in which we will remove the complexes (for the reactions with changeable complexes) can be rearranged in such a way that we have an optimal removal. In other words we will need to find the complexes $C_{i,1}$, ..., $C_{i,9}$. Here we use the initial combination given in table (5-3).

As discussed, for the rearrangement of complexes we will use the second method. In this case there will be a total of 9! = 362,880 possibilities for rearranging the complexes. Nevertheless, to select a bit faster we will apply this method step by step. This means we will first assume that we have to do with the reactions and complexes as given in table (5-2). Then we will swap the complex C_{28} with complexes C_{27} till C_{20} and look at the complex that will result in the lowest error. By doing this the number of possibilities will decrease to a total of 9 + 8 + ... + 2 = 44. Here the error E, based on the L_1 norm, has been defined by:

$$E_1 = \Big(\sum_{t=0}^{n} |x_{1,t} - x_{1,red,t}| \quad \cdots \quad \sum_{t=0}^{n} |x_{m,t} - x_{m,red,t}| \Big) \tag{5-23}$$

$$E = \frac{1}{m} \sum_{i=1}^{m} E_{1,i} \tag{5-24}$$

Note that the sequence of removals will be changed, but the number of removed complexes will always be 9 per step in the optimization process. For the error we will both take the mean of the states as well as the mean over all timesteps. For computing the errors E we will have to take the number of time steps and simulation time into account since this will influence the error a lot. Therefore it is better to have a lower number of time steps since otherwise we have states that will get a mean close to the steady-state value. Here we will take $n = 1000$. In this way we rearrange the complexes and find the complex rearrangement that gives the lowest error. Once, we have found the lowest error for a certain complex, we will fix that complex in the complexes network and continue this procedure. This will continue until we have found all complexes $C_{i,1}, \ldots, C_{i,9}$.

So after all, we actually relabel the complexes in such a way that we will remove the complexes in a different order. Now we are looking for the best way to remove these complexes. Then the first nine options will get the following errors as given in table (5-5):

Rearrange the following complexes	**Errors E**
Without rearrangements ($C_{i,7} = C_{28}$)	7.2
C_{27} with C_{28} ($C_{i,7} = C_{27}$)	0.3754
C_{26} with C_{28} ($C_{i,7} = C_{26}$)	0.4390
C_{25} with C_{28} ($C_{i,7} = C_{25}$)	0.4469
C_{24} with C_{28} ($C_{i,7} = C_{24}$)	0.4334
C_{23} with C_{28} ($C_{i,7} = C_{23}$)	0.4328
C_{22} with C_{28} ($C_{i,7} = C_{22}$)	0.6399
C_{21} with C_{28} ($C_{i,7} = C_{21}$)	0.5215
C_{20} with C_{28} ($C_{i,7} = C_{20}$)	1.1209

Table 5-5: Rearrangement of complexes along with the errors (mean of the difference between states over all timesteps)

This actually states that the best option for $C_{i,7}$ is C_{27}. Thus now we fix $C_{i,7} = C_{27}$ in the complexes network and go on by finding the other complexes where we have a maximum of 8 options.

So we can continue this procedure and finally get the best removal of complexes as given in table (5-6):

Reactions without removable complexes	Reactions with optimized removable complexes
$C_1 \underset{k_{-2}}{\overset{k_1}{\rightleftharpoons}} C_2$	$C_{14} \xrightarrow{k_{17}} C_{20} \xrightarrow{k_{18}} C_4$
$C_9 \underset{k_{-8}}{\overset{k_7}{\rightleftharpoons}} C_{10}$	$C_3 \xrightarrow{k_3} C_{24} \xrightarrow{k_4} C_5$
$C_7 \xrightarrow{k_{19}} C_{12}$	$C_6 \xrightarrow{k_5} C_{26} \xrightarrow{k_6} C_8$
$C_{15}\ ,\ C_{17},\ C_{19} \xrightarrow{k_{deg}} \emptyset$	$C_{11} \xrightarrow{k_9} C_{21} \xrightarrow{k_{10}} C_{13}$
	$C_{22} \underset{k_{-12}}{\overset{k_{11}}{\rightleftharpoons}} C_{25} \xrightarrow{k_{deg}} \emptyset$
	$C_{16} \xrightarrow{k_{13}} C_{23} \xrightarrow{k_{14}} C_{18}$
	$C_{28} \underset{k_{-16}}{\overset{k_{15}}{\rightleftharpoons}} C_{27} \xrightarrow{k_{deg}} \emptyset$

Table 5-6: Reactions with optimized removable complexes $C_{20}, \ldots, C_{28}$

5-7 Removing complexes according to optimized linkage classes

Now, if one looks at the optimal case where the order of removing complexes is optimized, the error function will show us some really interesting results. For the optimal case the reduction order models are given in figure (5-7) with the species concentration of the original model:

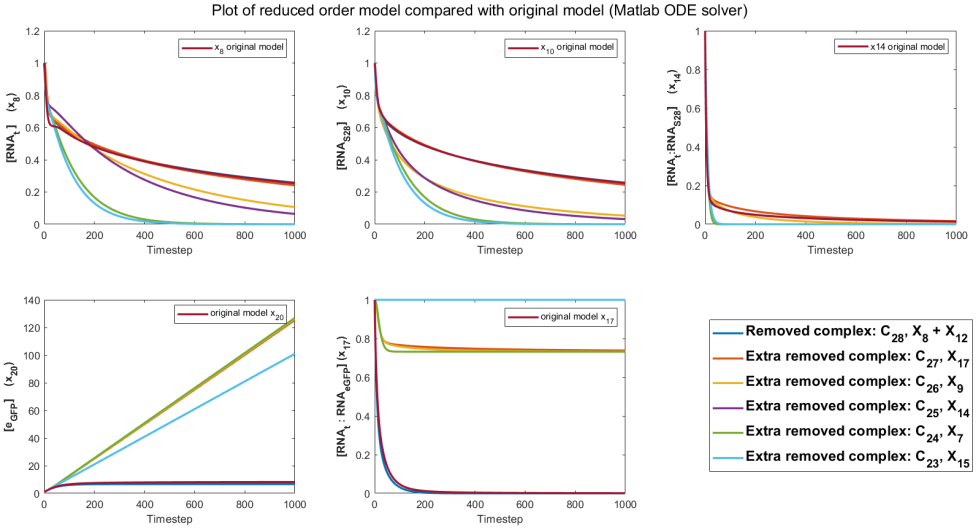
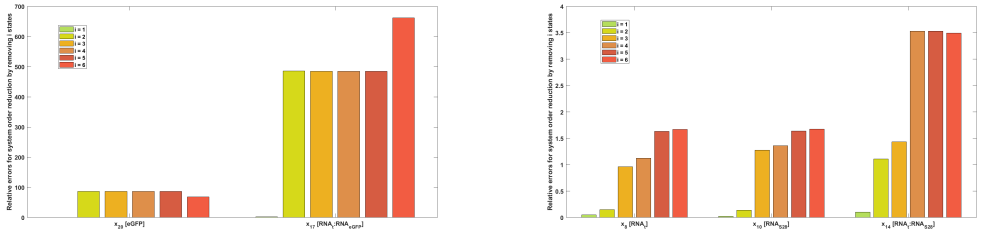


Figure 5-7: Original model compared with corresponding reduction order models that exclude the networkcomplexes according to the linkage of networkclasses in optimal case

As can be verified, figure (5-8a) and (5-8b) show the results. Mostly, a higher number of removed complexes results in a higher RMS error for states $x_8, x_{10}, x_{14}, x_{20}$ and x_{17}.



(a) Relative RMS errors for states X_{20}, X_{17} **(b)** Relative RMS errors for states X_8, X_{10}, X_{14}

Figure 5-8: Relative RMS errors (within an interval of 10^5 timesteps) for the given states for models that have been reduced in order by $i = 1, \ldots, 6$

For some reason the error signals between the optimized and non-optimized case won't differ that much or are even higher, but in general we can say that the RMS error will obviously be lower in the optimized case.

5-8 Conclusions Kron reduction order method

In fact, if we continue reasoning, complexes influence other complexes and these complexes contain different species involved in different species reactions. Therefore we actually end up in biochemical reactions as in the original flowscheme and we conclude that some information will be ignored if we look at the complexes separately. Data is lost since some species are contained in more complexes and this is not visible if one looks at the complexesnetwork only. Nevertheless for larger structures it is expected to give good results and it is more valid if we want to reduce the order of the system rapidly. Then considerably a more accurate reduction order model can be applied afterwards. It mostly depends on the species concentration we are looking at with respect to the complexes that have been removed so far. If we remove the complex that contained the species we are looking at, the corresponding species concentration will show some fully divergent characteristics wrt to the non-reduced system (see e.g. species concentration x_{17}).

The Kron reduction order method is based on the reduction of the number of complexes. By rewriting the update equations or set of ordinary differential equations we have the option to reduce the internal dynamics in terms of complexes. The external dynamics will somewhat be influenced by doing this and we can compare the two models. Here the concept of complex balancedness will guarantee whether the steady states for both models will be the same. [15]

The order at which we will remove the complexes is important. This is because of the fact that the deviation between the reduced order model and the non-reduced case becomes smaller in case of optimal removal.

5-9 Discussion Kron reduction order method

5-9-1 Advantages of the Kron reduction order method

The Kron reduction order method will have the following advantages:

- The procedure is easily applied and can be automated, e.g. error driven (in an iterative scheme)
- The method guarantees that the steady states of the reduced system are equal to the steady states of the non-reduced system if the system is complex balanced.

5-9-2 Challenges of the Kron reduction order method

The challenge for the model of CFFLs is that we want to work with a complex balanced system even though the full system isn't complex balanced. Therefore we will have to select certain complexes (subsystems) in order to have the complex balancedness property. Then the selection of complexes to be removed is done in such a way that the steady states will not be destroyed. A better application of the Kron reduction order method still has to be worked out. Since there are a lot of reduction order schemes for the Kron reduction order method we were mainly able to find a more optimal case, but the best optimized case can still be worked out.

Chapter 6

Flow experiments

6-1 Degradation reactions

In a flowexperiment every species will be subject to flow. Furthermore there are species that have, in addition, natural degradation, which in this case effectively changes the kinetic rate constants. Assume that we have to do with a species X_{deg} that experiences natural degradation. Then the degradation (which is similar to the reactions in the batch experiment) will be of the following form:

$$R_{deg} : \mathrm{X_{deg}} \xrightarrow{\mathrm{k_{deg,i}}} \emptyset \tag{6-1}$$

6-2 From batch to flow experiment

Let N be the Stoichiometry matrix that contains reactions of the internal system as defined in chapter 2. For the inflow we will define a flowrate γ related to an inflowvector x_{in} that contains the concentrations of all added species. Then the model in flow experiment reads as

$$\dot{x} = Nv(x) + \gamma(x_{in} - x), \text{ with } \gamma : \text{the flow rate of the system} \tag{6-2}$$

This model of equations is equivalent with:

$$\dot{x} = N_f v_f(x, x_{in}) \tag{6-3}$$

$$N_f = [N_{new} \quad N_{out} \quad N_{in}] = [N \quad -I \quad I] \tag{6-4}$$

$$v_f(x, x_{in}) = \begin{pmatrix} v(x) & \gamma x & \gamma x_{in} \end{pmatrix}^\top \tag{6-5}$$

Since $N_f^\top$ is of full rank the left nullspace is empty and the method of conservationlaws won't help to reduce the order of the system. Besides according to physics it is logically because every species concentration now depends on its changing inflow/outflow. Therefore we will need to use different methods if we would like to do so.

6-3 Proper choice of inflow x_{in}

Assume we will have to do with the same type of flow experiment. Then the flow experiment, where one preferably wants to assume that Z is invertible, is given by the extended model that reads as:

$$\dot{x} = -ZL \text{ Exp } (Z^\top \text{ Log } x) + \gamma(x_{in} - x) \tag{6-6}$$

In contrast to what has been done in the QSSA approach we will now choose a better inflow x_{in}. For constant inflows x_{in} the goal is to reach the steady state sooner than in the batch experiment. To do this we first start by having a look at the current steady state x^* for which holds:

$$\dot{x} = -ZL \text{ Exp } (Z^\top \text{ Log } x^*) = 0. \tag{6-7}$$

Therefore, since the internal kinetics are such that the same steady state x^* will be reached, the following must hold for the flow experiment:

$$\gamma(x_{in} - x^*) = 0 \Longrightarrow x_{in} = x^*, \tag{6-8}$$

In other words, until the equilibrium is settled we want to add a species concentration equal to the current concentration such that every species concentration stays constant from that timestep. Here we will only take the real inflows as positive (i.e. no interconnected species and output species). Theoretically an outflow is given if the species concenrtation $x_{in} < x$ and $\gamma > 0$, but this won't be visible since the initial condition is below this value in steady state.

Then the non-zero species concentrations of x_{in} are given by:

$$x_{2,in} = x_2^* = [S70]^*, \qquad x_{8,in} = x_8^* = [RNA_t]^* \tag{6-9}$$
$$x_{3,in} = x_3^* = [DNA_t]^*, \qquad x_{10,in} = x_{10}^* = [RNA_{S28}]^* \tag{6-10}$$
$$x_{4,in} = x_4^* = [DNA_{S28}]^* \qquad x_{13,in} = x_{13}^* = [Ribo]^* \tag{6-11}$$
$$x_{5,in} = x_5^* = [DNA_{eGFP}]^* \qquad x_{16,in} = x_{16}^* = [S28]^* \tag{6-12}$$

In particular it contains the species involved in the conservation laws as defined in (2-16) till (2-21) (i.e. the species that need to be present in order to reach a certain steady state).

Note that a higher flowrate γ will result in a faster settling time as can be seen in figure (6-1).

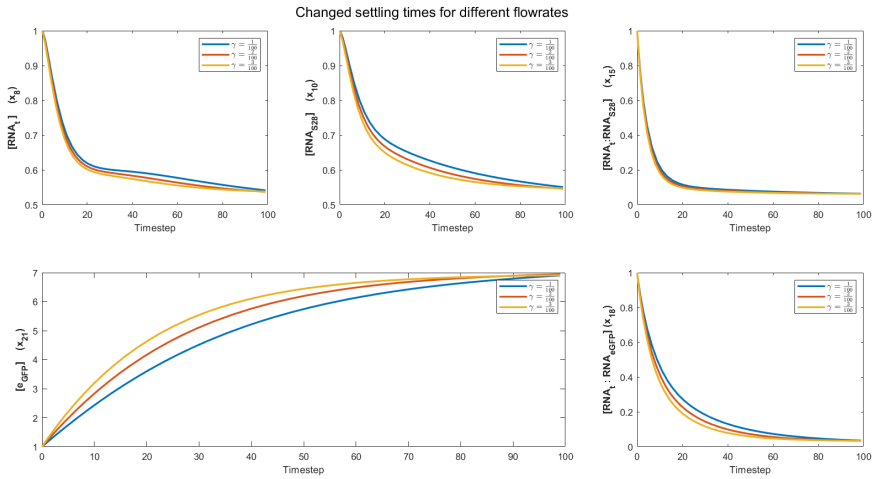


Figure 6-1: Flowexperiment for flowrates $\gamma = \frac{1}{100}, \gamma = \frac{2}{100}, \gamma = \frac{3}{100}$ and $x_{in} = x^*$

Indeed, figure (6-1) shows a faster settling time for respectively $\gamma = \frac{1}{100}, \gamma = \frac{2}{100}$ and $\gamma = \frac{3}{100}$.

6-4 QSSA method for inflowexperiments

For the QSSA method for flowexperiments we will also deal with the following form of the system: $\dot{x} = Nv(x) + \gamma(x_{in} - x)$. Now one wants to initialize the system by looking at the steady states (which is x_{in}). Here we will use the similar initialisation as in the batch experiment since the time it takes to reach the steady state will be scaled somehow and the flowterms will have less influence on this (based on a relative low flowrate γ). Here 'first order reduction method' means that we have removed or initialised one species concentration that reaches its steady state at its fastest. In general we get: A nth order reduction method means that we have removed or initialised n species concentration(s) that reaches its steady state at its fastest with respect to the other species concentrations.

If we do this flowexperiment we get the following results as given in figure (6-2):

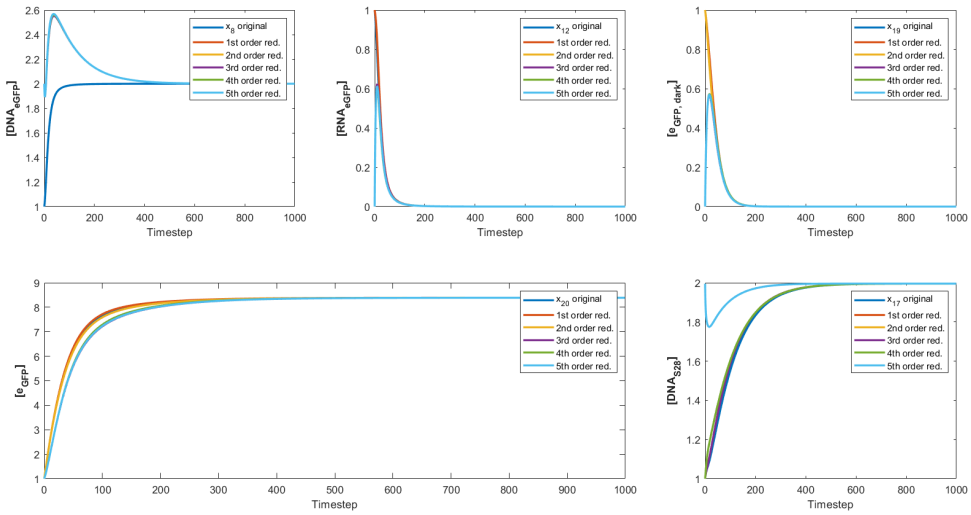


Figure 6-2: Applied QSSA method for flow experiments, note that we haven't applied the full/classical Quasi Steady State Approach for the flow experiment yet.

It becomes clear that the settling times are much more close to each other with respect to the normal experiment. Mostly the trajectory of the reduction order model will follow the same curve as the non-reduced system after a few timesteps. Though if we increase the number of initialized states we will see that the settling times become faster anyway.

6-5 Kron reduction order method for flowexperiments

To apply the Kron reduction order method for flowexperiments we first need to rewrite our CFFL model in a special form. For this rewriting we need to make sure that Z is invertable since this will give a unique vector v_b as we will see later on. Actually we want to rewrite (6-6) in the following form:

$$\dot{x} = -ZL \text{ Exp } (Z^\top \text{ Log } x) + Zv_b \Longrightarrow Zv_b = \gamma(x_{in} - x) \Longrightarrow v_b = Z^{-1}\gamma(x_{in} - x). \tag{6-13}$$

If we assume the more simplistic reaction as before:

$$\text{A} \ + \ \text{B} \ \underset{\text{k}_{-1}}{\overset{\text{k}_1}{\rightleftharpoons}} \ \text{C} \ \underset{\text{k}_{-2}}{\overset{\text{k}_2}{\rightleftharpoons}} \ \text{A} \tag{6-14}$$

Then by interchanging complex C_1 and complex C_2, we get the invertible matrix Z given by:

$$Z = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ where } C_1 := A, \quad C_2 := A + B, \quad C_3 := C \tag{6-15}$$

Then vector v_b and matrices D, A are now given by:

$$v_b = Z^{-1}\gamma(x_{in} - x) = \gamma \begin{pmatrix} (a_{in} - a) - (b_{in} - b) \\ b_{in} - b \\ c_{in} - c \end{pmatrix} \tag{6-16}$$

$$D = \begin{pmatrix} k_{-2} & 0 & 0 \\ 0 & k_1 & 0 \\ 0 & 0 & k_{-1} + k_2 \end{pmatrix}, A = \begin{pmatrix} 0 & 0 & k_2 \\ 0 & 0 & k_{-1} \\ k_{-2} & k_1 & 0 \end{pmatrix} \tag{6-17}$$

Now we actually have two models of the following forms:

$$\dot{x} = -ZL \text{ Exp } (Z^\top \text{ Log } x) + Zv_b \tag{6-18}$$

$$\dot{x} = Nv(x) + \gamma(x_{in} - x) \tag{6-19}$$

To see that these two are fully identical with the given terms we make a plot as in figure (6-3)

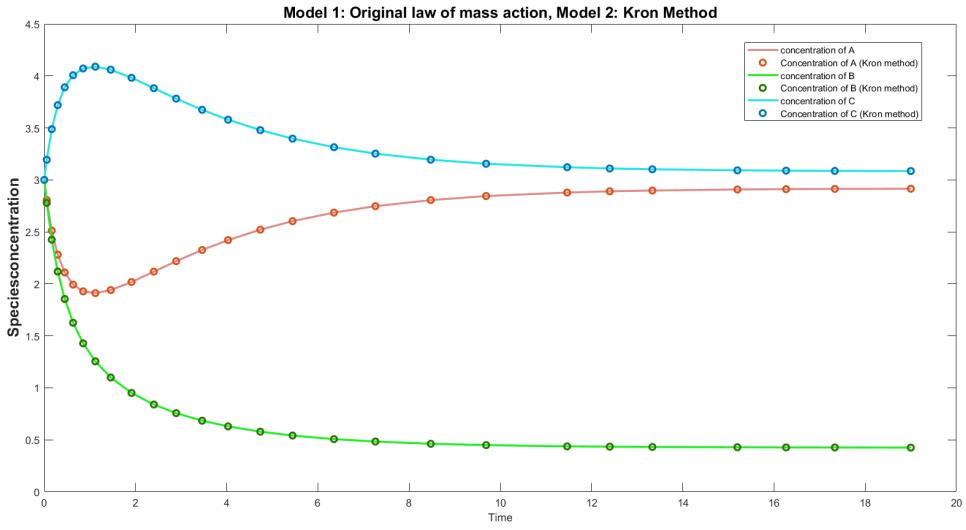


Figure 6-3: Speciesconcentrations of Kron method along with the speciesconcentrations of the model where the law of mass action has been applied

The difference between the inflowmodel and the model without inflows is directly visible by noticing the different reached concentrations of species A and C. If there is some inflow the steady states for concentration A and C differ in contrast to the case without inflows.

The inflowmodels can now be given according to equation (6-16). In this case we will end up in figure (6-4), which displays the concentrations of the species along with the speciesconcentrations as given in the reduction order models.

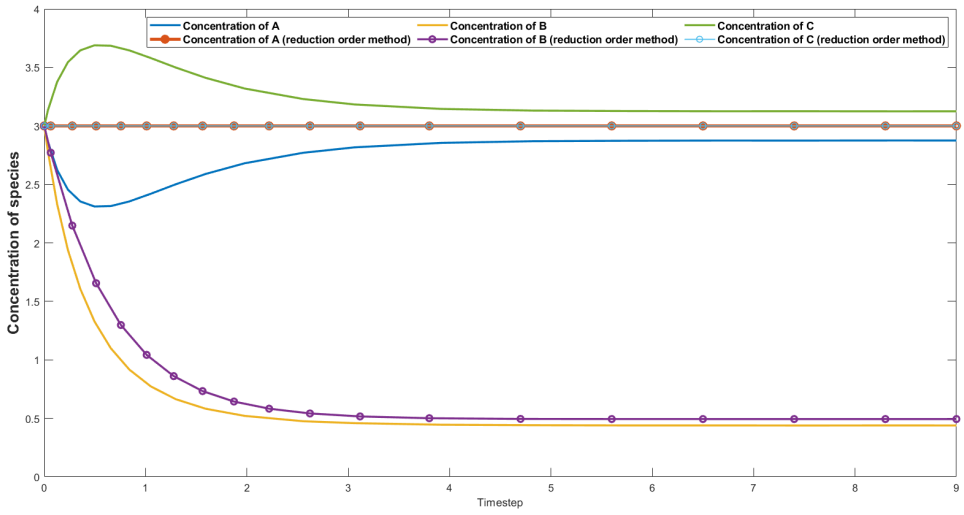


Figure 6-4: Speciesconcentrations of Kron reduction order model and non-reduced model with flowrate $\gamma = \frac{3}{15}$

Even though we have to do with a complex balanced system the inflows cause the reduction order models to have different steady states. Therefore the complex balancedness condition alone is not enough to conclude that we have to do with equal steady states for these kind of systems. [15]

If we look at the plots in which we use $\gamma = \frac{1}{15}, \gamma = \frac{2}{15}$ and $\gamma = \frac{3}{15}$, then we get figure (6-5):

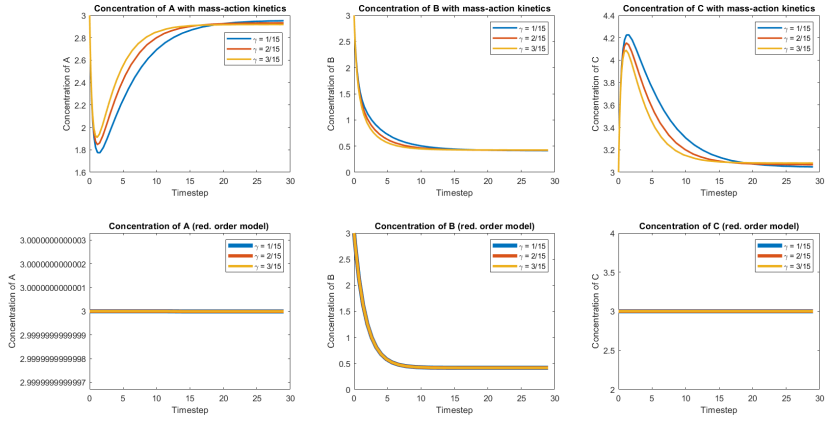


Figure 6-5: Speciesconcentration of A, B and C with $\gamma = \frac{1}{15}, \gamma = \frac{2}{15}$ and $\gamma = \frac{3}{15}$

As can be seen in figure (6-5) a higher γ results in a faster settling time and a certain change in the steady states. Though the reduction order models seem to get more constant species concentrations and a change of γ hardly influences the curves for the species concentrations.

6-6 Conclusions Flow Experiments

Here we will see how to transform the system if we also have to do with in- and outflows. It actually means that we will need to add an extra term . There are two ways to do this even though they will have a different interpretation. One will be in matrix-vector form while the other method merely uses vector-scalar notation. The relation between these two is: $Zv_b = \gamma(x_{in} - x)$ where we want to find v_b and a (pseudo-)inverse of Z. For small systems with flow we were able to analyse the reduction order models. For QSSA it gave similar results as the batch experiment. For the Kron reduction order model the order doesn't seem to have that much effect. A higher flowrate γ resulted in faster settling times for both the non-reduced and the reduced case.

Chapter 7

Conclusions and future challenges

7-1 Conclusions

To get more insight in the dynamics that play a role in biochemical reactions, we first start by having a look at how the system is modelled and how the biochemical reactions are analyzed. Thereafter we have also discussed the modelling strategies. To analyze, a set of biochemical reactions can be rewritten in ODE models. The rewriting into a mathematical model was done for a couple of biological processes in which the law of mass action has been applied. Even though more kinetics are possible (e.g. Hill or Michaelis-Menten (MM)), mostly we will use mass-action kinetics that will describe the relation between the concentration of a substrate S and formation of product P (reaction rate) [1], [2], [17].

The number of conservation laws can be deduced by looking at the dimension of the left nullspace of the Stoichiometry matrix. Besides we can write the exact conservation laws with this left nullspace [5] . By using the appropriate substitutions we can eliminate certain species updates $\dot{x}$.

The Quasi Steady State Approach is applied by neglecting the fast reactions over the slow reactions for the slow timescale [6], [15]. Then the differential equations for the fast reactions can be set to zero and we will, similar to the method of using conservationlaws, get substitutions or eliminations accordingly. Another useful aspect is the initialisation of the steady states where the fastest settled species are taken into account first. This is also known as the identification of the slow manifold. For this method a higher reduction order implies faster settling times since the number of initialized species (identical to the final steady states) is higher. The steady states between non-reduced and reduced case are (approximately) the same except for the output proteins.

For the network of CFFLs we can replace certain pathways in the chemical reaction network by more simple structures. Here we need to take renewed rate constants into account such that the reduced and non-reduced case will only have small deviations. As written before, the kinetics will mostly be based on mass-action. Though depending on the values of the rate

constants one can also reduce a chain of reactions to a single reaction of Michaelis-Menten type.

The Kron reduction order method is based on the reduction of the number of complexes instead of the number of reactions. By rewriting the update equations or set of ordinary differential equations we have the option to reduce the internal dynamics in terms of complexes. The external dynamics will somewhat be influenced by doing this and we can compare the two models. Mostly the concept of complex balancedness will determine whether the steady states for both models will be the same [15].

One can always apply the elimination of equations by using conservation laws if the left nullspace of the Stoichiometry matrix is non-empty. In this case the results will be exactly the same after applying the elimination. Therefore we shouldn't really say that this is part of a reduction order model. If we compare QSSA and the Kron reduction order method, then we would choose the Kron reduction order method if the system is complex balanced according to literature [15]. Though here, based on the results, it would be a better choice to go for QSSA. Especially for states x_{17} and x_{20} the (RMS-)errors will be quite high in case of the Kron reduction order model. First of all this is because of the fact that the network of CFFLs is not complex balanced. Secondly this is probably due to the fact that we haven't really optimized the Kron reduction order method at its best since we simply used a step by step procedure and this could be improved somehow. After all, for QSSA the steady states in the reduced order system will almost be identical to the steady states in the non-reduced system, but the Kron reduction order method guarantees that the steady states of the reduced system are equal to the steady states of the non-reduced system if the system is complex balanced. For QSSA for small reduction order models there will be a small deviation between the reduced and non-reduced system and a combined method of slow manifold identification can show a best way to initialise the defined species concentrations. Though for QSSA a higher reduction model is needed before the reduced order model or non-reduced model have the same steady states. Besides the initialisation of species concentrations (or the removal of certain species) makes the model completely different at the beginning of the experiment. Besides the procedure of the Kron reduction order method is easily applied and can be automated, e.g. error-driven (in an iterative scheme).

Once we've been discussing several reduction order methods, we are able to take further steps. Another interesting feature was the transformation of the system if we also have to do with in- and outflows. It actually means that we will need to add an extra term . There are two ways to do this even though they will have a different interpretation. One will be in matrix-vector form while the other method merely uses vector-scalar notation. The relation between these two is: $Zv_b = \gamma(x_{in} - x)$ where we want to find v_b and a (pseudo-)inverse of Z. For small systems with flow we were able to analyse the reduction order models. For QSSA it gave similar results as the batch experiment. For the Kron reduction order model the order doesn't seem to have that much effect. A higher flowrate γ resulted in faster settling times for both the non-reduced and the reduced case.

7-2 Future challenges

During this graduation project we have seen two ways to reduce our system in order. The first one was the Quasi Steady State Approach. There we demonstrated how the identification of the slow manifold could be done in multiple ways. Basically there one wants to find the sequence of the species concentrations that need to be initialised and thus actually the species that will need to be eliminated. This can be based on the time it takes till it reaches its steady state or (equivalently) this can be based on the time it takes until the derivative state reaches zero. Furthermore a higher order reduction model is needed before the reduced order model or non-reduced model have the same steady states. Though a lower order reduction model will result in smaller deviations between the reduced and non-reduced system.

The second way to reduce our system in order was the Kron reduction order method. The challenge for the model of CFFLs is that we want to work with a complex balanced system even though the full system isn't complex balanced. Therefore we will have to select certain complexes (subsystems) in order to have the complex balancedness property. Then the selection of complexes to be removed is done in such a way that the steady states will not be destroyed. A better application of the Kron reduction order method still has to be worked out. Since there are a lot of reduction order schemes for the Kron reduction order method we were mainly able to find a more optimal case, but the best optimized case can still be worked out.

The challenge of Alternative Modelling will be to reduce the system in order without losing too much information of the original CFFL network. Then one could argue whether it is still realistic if more and more subsystems will be removed from the full system of CFFLs. On the other hand it will gain us more information of the importance of a species or complex within our network of CFFLs.

In general, the challenge we would focus on is to have an automated procedure that directly transforms the biochemical system equations into its reduced order form. With computed errorfunctions the best method (or combination of methods) could be selected first automatically. Besides with experiments one could make a match of the model and see what kind of kinetics match best for the given situation/experiment.

Bibliography

[1] Del Vecchio, D. and Murray, R. (2015). *Biomolecular feedback systems*. 1st ed. Princeton, New Jersey: Princeton University Press, p. 1 - 287.

[2] Keener, J. and Sneyd, J. (2009). *Mathematical Physiology, I: Cellular Physiology* . 2nd ed. Maryland: Springer Science + Business media, p.1 - 1067.

[3] Smeu, J. (2019). *Modelling, Analysis and Verification of Biological Coherent Feedforward Loop Network*. Delft, p. 1 - 100.

[4] Shodhan, R. Schaft, A. van der. Eunen, K. van. M Bakker, B. and Jayawardhana, B. (2014). *A model reduction method for biochemical reaction networks*. BMC Systems Biology, Biomedcentral

[5] Hassan, K. (2014) *Modelling and Nonlinear Systems*. Hassan, K. , Pearson New International Edition.

[6] Snowden, T. *Model reduction for Biochemical Systems*, *Computational Methods*, University of Reading

[7] Olsman, N. Paulsson, J. Lewis, K. Strandwitz, P. (2019) *Universal control in biochemical circuits & Metabolic mischief as microbes target drugs*. Springer Nature Limited.

[8] Ellis, S. J. Titus, H. K. Lathrop, J.I. (2019) *Robust chemical circuits* Elsevier B.V.

[9] Chen, B. Chen, P. (2008) *Robust Engineered Circuit Design Principles for Stochastic Biochemical Networks With Parameter Uncertainties and Disturbances* , IEEE Transactions on Biomedical Circuits and Systems

[10] Snowden, T.J. Graaf, P.H. van der, Tindall, M.J. (2017) *Methods of Model Reduction for Large-Scale Biological Systems: A Survey of Current Methods and Trends*, Springer, Society for Mathematical Biology

[11] Adleman, L.M. (1994) *Molecular Computation of Solutions to Combinatorial Problems*, Science

[12] Feinberg, M. (2019) *Foundations of Chemical Reaction Network Theory*, Springer

[13] M. Yelleswarapu, A. J. van der Linden, B. van Sluijs, P. A. Pieters, E. Dubuc, T. F. De Greef, and W. T. Huck, *Sigma Factor-Mediated Tuning of Bacterial Cell-Free Synthetic Genetic Oscillators*, *ACS Synthetic Biology* vol. 7, no. 12, pp. 2879–2887, 2018

[14] Schaft, A.J. van der, Rao, S., Jayawardhana, B. *A network dynamics approach to chemical reaction networks* , 2015

[15] Horn, F. Jackson, R. Aris, R. *General Mass Action Kinetics*, 1972

[16] Constales, D. Marin, Guy B. , *Advanced Data Analysis & Modelling in Chemical Engineering*

[17] https://depts.washington.edu/wmatkins/kinetics/michaelis-menten.html, *Michaelis-Menten Kinetics and Briggs-Haldane Kinetics*, Accessed: 30-12-20

[18] Richard A. B. Bond, Bice S. Martincigh, Janusz R. Mika, and Reuben H. Simoyi, *The Quasi-Steady-State Approximation: Numerical Validation*, 1998

[19] Borghans, J.A.M, Segel, L.A., *Extending the Quasi Steady State Approximation by changing variables*, 1996

[20] Eilertsen, J. Schnell,S. *The quasi-steady-state approximations revisited: Timescales, small parameters, singularities, and normal forms in enzyme kinetics*, 2020

[21] Flach, E.H. Schnell, S. *Use and abuse of the quasi-steady-state approximation*, 2006

[22] https://chem.libretexts.org/Bookshelves/Biological_Chemistry/Supplemental_Modules_(Biological_Chemistry)/Enzymes/Enzymatic_Kinetics/Michaelis-Menten_Kinetics, *Michaelis-Menten kinetics* , Accessed: 06-05-21

[23] Braun, M. *Differential Equations and Their Applications* 4th ed. , Springer, 1992

[24] Keviczky, T. *Control Theory SC42015* , Delft Center for Systems and Control (DCSC).

[25] Beauregard, R.A. Fraleigh, J.B. (1995) *Linear Algebra* , Addison Wesley Longman

Appendix A

Defined matrices

$$\mathbf{B} = \begin{bmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0
\end{bmatrix} \quad \text{(A-1)}$$

$$\mathbf{Z} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 \\
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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 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 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \tag{A-2}$$

$$
\mathbf{N} = \begin{bmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 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 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{A-3}
$$

Define $k'_1 = k_{deg} + k_{-12}, k'_2 = k_{deg} + k_{-16}$ Then

$$L = \begin{bmatrix}
k_1 & -k_{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-k_1 & k_{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & k_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & k_{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & k_7 & -k_{-8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -k_7 & k_{-8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{deg} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{deg} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{deg} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{17} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{17} & k_{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{18} & k_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_4 & k_6 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_6 & k_{10} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{10} & k'_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{12} & k_{14} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{14} & k'_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_{16} & k'_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k'_2 & 0
\end{bmatrix} \tag{A-4}$$

Appendix B

Tabulars

Number of removed states	**1**	**2**	**3**	**4**	**5**
RMS error with x_8	0.27	0.39	0.89	1.29	3.46
RMS error with x_{10}	0.05	0.29	0.38	1.02	3.85
RMS error with x_{14}	0.05	0.03	0.12	0.11	0.33
RMS error with x_{20}	0.36	0.57	1.06	1.50	4.20
RMS error with x_{17}	0.05	0.07	0.14	0.20	0.58

Table B-1: RMS errors between original model x_i and reduced order model $x_{red,i}$ where i equals the number of the removed states from the non-reduced model for the Quasi-steady state approach (QSSA)

Initialised species	**Timestep at which the steady state is reached within a 10 % tolerance interval**
X_5	51.37
X_{13}	100.81
X_{20}	182.79
X_3	468.35
X_4	468.35
X_2	589.32
X_{16}	725.85
X_8	845.04
X_{10}	846.01
X_7	867.63
X_9	867.63
X_6	887.58
X_1	906.38
X_{15}	920.91
X_{14}	921.77
X_{12}	923.61
X_{21}	926.55
X_{11}	926.89
X_{18}	948.36
X_{17}	948.45
X_{19}	948.76

Table B-2: Initialised species along with the timestep at which the steady state is reached within a 10 % tolerance interval

	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,1}}\rvert$	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,2}}\rvert$	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,3}}\rvert$	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,4}}\rvert$	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,5}}\rvert$	$\lvert\mathbf{x_{ss}} - \mathbf{x_{redss,6}}\rvert$
X_1	0	0	0	0	0	0
X_2	0	0	0	0	0.0010	0.0020
X_3	0	0	0	0	0	0.0010
X_4	0	0	0	0	0.0010	0.0010
X_5	0	0	0	0.0010	0.0010	0.0010
X_6	0	0	0	0	0	0
X_7	0	0	0	0	0	0.0010
X_8	0	0	0	0	0	0
X_9	0	0	0	0	0.0010	0.0010
X_{10}	0	0	0	0	0	0
X_{11}	0	0	0	0	0	0
X_{12}	0	0	0	0	0	0
X_{13}	0.0013	0.0020	0.0020	0.0020	0.0020	0.0020
X_{14}	0	0	0	0	0	0
X_{15}	0	0.0010	0.0010	0.0010	0.0010	0.0010
X_{16}	0.0038	0.0086	0.0137	0.0137	0.0147	0.0157
X_{17}	0.0007	0.0010	0.0010	0.0010	0.0010	0.0010
X_{18}	0.0013	0.0010	0.0010	0.0010	0.0010	0.0010
X_{19}	0.0013	0.0010	0.0010	0.0010	0.0010	0.0010
X_{20}	1.2577	0.9925	0.9925	0.9925	0.9925	0.9925
X_{21}	0	0	0	0.0010	0.0010	0.0010

Table B-3: Absolute errors between steady states reduced order and non-reduced order model, $x_{red,ss,i}$ is the steady state of the ith order reduction for the Kron method

Appendix C

Matlab code

C-1 Conservation Laws

C-1-1 Simple model

```
%%  Clean up
clc; clear all; close all;
%% Initialisation of constants, vectors, matrices

n = 50;                          % Nubmer of timesteps
k1 = 1/10; k_1 = 3/10;           % Flux rates
dt = 1/10;                       % Timestep per iteration

% Orignal model
x1 = [1; 1; 1];                  % Initial vector x: [a b c]'
x = x1;
xplot = ones(length(x), n);      % Add plotvectors
N = [1 -1; 1 -1 ; -1 1];         % Stochiometry matrix original model
e0 = x(1) + x(3);                % First conservation law: a + c = constant
e1 = x(2) + x(3);                % Second conservation law: b + c = constant

% Reduced order model
x1red = [1; 1];                      % Initial vector x: [a c]'
xred = x1red;
xredplot = ones(length(xred), n); % Add plotvectors
Nred = [1 -1 -1 ; -1 1 1];           % Stochiometry matrix reduced order
    model

for t = 1:n
  % Orignal model
   xplot(:,t) = x;
   v = [k_1*x(3); k1 *x(1)*x(2)]; % Flux rates
   xnew = N*v*dt + x; % Euler Forward based iteration
```

```
28    x = xnew;
29    %  Reduced order model
30    xredplot(:,t) = xred;
31    vred = [(k_1 + k1*(e0 + e1))*xred(2); k1*xred(2)^2; k1*e0*e1]; % Flux
          rates
32    xnewred = Nred*vred*dt + xred; % Euler Forward based iteration
33    xred = xnewred;
34  end
35
36  %% Comparison of reduced order model with original model
37
38  figure(1);
39  plot(0:n-1,xplot(1,:), 'linewidth', 2  ) % Plot of concentration of A
        for original model
40  hold on
41  plot(0:n-1,xredplot(1,:), 'o', 'linewidth', 2  ) % Plot of concentration
        of A for red. order model
42  plot(0:n-1,xplot(3,:), 'linewidth', 2  ) % Plot of concentration of C
        for original model
43  plot(0:n-1,xredplot(2,:), 'o', 'linewidth', 2  ) % Plot of concentration
        of C for red. order model
44  hold off
45  xlabel('Timestep');
46  ylabel('concentration of species');
47  legend('[A]: Original model',  '[A]: Reduced order model', '[C] Original
        model', '[C]: Reduced order model' );
```

C-1-2 CFFL network

```
1  %% Clean up
2  clear all; close all; clc;
3  %%  Ode23 solver based solution of xdot = Nvx
4  n = 100; % number of timesteps
5  tspan = [0 n-1];
6  x0 = ones(22,1);
7  [t_ode,x_ode] = ode23(@odefun, tspan, x0);
8  [t_red_ode, x_red_ode] = ode23(@odefun_red, tspan, x0);
9
10 %% Plotting (for matlab ODE solvers)
11
12 figure(2);
13
14 subplot(2,3,1);
15 plot(t_ode,x_ode(:,8), 'linewidth', 2  ) % Plot of state x8 for n
       timesteps
16 hold on
17 plot(t_red_ode, x_red_ode(:,8),'o', 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
18 hold off
19 xlabel('Timestep');
20 ylabel('[RNA_t ]     (x_8)');
21 legend('x8 original', 'x8 reduced order system');
22
```

```
23 subplot(2,3,2);
24 plot(t_ode,x_ode(:,10), 'linewidth', 2 ) % Plot of state x10 for n
       timesteps
25 hold on
26 plot(t_red_ode, x_red_ode(:,10), 'o' , 'linewidth', 2 ) % Plot of state
       x10 for n timesteps (red. order model)
27 hold off
28 xlabel('Timestep');
29 ylabel('[RNA_{S28}]    (x_{10})');
30 legend('x10 original', 'x10 reduced order system');
31
32 subplot(2,3,3);
33 plot(t_ode,x_ode(:,15), 'linewidth', 2 ) % Plot of state x15 for n
       timesteps
34 hold on
35 plot(t_red_ode, x_red_ode(:,15),'o', 'linewidth', 2 ) % Plot of state
       x15 for n timesteps (red. order model)
36 hold off
37 xlabel('Timestep');
38 ylabel('[RNA_t:RNA_{S28}]    (x_{14})');
39 legend('x14 original', 'x14 reduced order system');
40
41 subplot(2,3,4:5);
42 plot(t_ode,x_ode(:,21), 'linewidth', 2 ) % Plot of state x21 for n
       timesteps
43 hold on
44 plot(t_red_ode, x_red_ode(:,21), 'o', 'linewidth', 2 ) % Plot of state
       x21 for n timesteps (red. order model)
45 hold off
46 xlabel('Timestep');
47 ylabel('[e_{GFP}]    (x_{20})');
48 legend('x20 original', 'x20 reduced order system');
49
50 subplot(2,3,6);
51 plot(t_ode,x_ode(:,18), 'linewidth', 2 ) % Plot of state x18 for n
       timesteps
52 hold on
53 plot(t_red_ode, x_red_ode(:,18), 'o','linewidth', 2 ) % Plot of state
       x18 for n timesteps (red. order model)
54 hold off
55 xlabel('Timestep');
56 ylabel('[RNA_t : RNA_{eGFP}] (x_{17})');
57 legend('x17 original', 'x17 reduced order system');
58 sgtitle('Plot of reduced order model compared with original model (Matlab
       ODE solver)');
59 %% N matrix
60
61 N = zeros(22,23);
62
63 N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
64 N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
65 N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
66 N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
```

```
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1;
N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
N(15,12) = -1;N(14,12) = -1;N(16,12) = 1;
N(16,13) = -1;N(17,13) = 1;N(14,13) = 1;
N(8,14) = -1;N(13,14) = -1;N(18,14) = 1;
N(18,15) = -1;N(8,15) = 1;N(13,15) = 1;
N(18,16) = -1;N(14,16) = -1;N(19,16) = 1;
N(19,17) = -1;N(14,17) = 1;N(20,17) = 1;N(18,17) = 1;
N(20,18) = -1;N(21,18) = 1;
N(10,19) = -1;
N(8,20) = -1;
N(13,21) = -1;
N(15,22) = -1;
N(18,23) = -1;
```

```
function dxdt = odefun(t, x)
k1 = 1/10;
k_2 = 1/10;
k3 = 1/10;
k4 = 1/10;
k5 = 1/10;
k6 = 1/10;
k7 = 1/10;
k_8 = 1/10;
k9 = 1/10;
k10 = 1/10;
k11 = 1/10;
k_12 = 1/10;
k13 = 1/10;
k14 = 1/10;
k15 = 1/10;
k_16 = 1/10;
k17 = 1/10;
k18 = 1/10;
k19 = 1/10;
kdeg = 1/10;
gamma = 1/100;

% Flux rates ( [3] )

% k1 = 249.995;
% k_1 = 1.88;
% k2 = 51.264;
% k3 = 3.1441;
% k4 = 1/10;
% k5 = 3.4445;
% k6 = 0.0390;
```

```
% k_6 = 28.837;
% k7 = 1/10;
% k8 = 3.499;
% k9 = 72.3162;
% k10 = 0.361;
% k11 = 49.944;
% k12 = 99.951;
% k_12 = 25.414;
% k13 = 3.4906;
% k14 = 494.256;
% k15 = 0.999;
% kdeg = 0.997;

N = zeros(22,24);
N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;

N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
N(20,19) = -1; N(21,19) = 1;
N(10,20) = -1;
N(8,21) = -1;
N(13,22) = -1;
N(15,23) = -1;
N(18,24) = -1;

vx = zeros(24,1);
vx(1) = k1*x(1)*x(2);
vx(2) = k_2*x(6);
vx(3) = k3*x(6)*x(3);
vx(4) = k4*x(7);
vx(5) = k5*x(6)*x(4);
vx(6) = k6*x(9);
vx(7) = k7*x(11)*x(17);
vx(8) = k_8*x(1);
vx(9) = k9*x(11)*x(5);
vx(10) = k10*x(22);
vx(11) = k11*x(10)*x(8);
vx(12) = k_12 * x(15);
```

```
86 vx(13) = k13*x(15)*x(14);
87 vx(14) = k14*x(16);
88 vx(15) = k15*x(8)*x(13);
89 vx(16) = k_16*x(18);
90 vx(17) = k17*x(18)*x(14);
91 vx(18) = k18*x(19);
92 vx(19) = k19*x(20);
93 vx(20) = kdeg*x(10);
94 vx(21) = kdeg*x(8);
95 vx(22) = kdeg*x(13);
96 vx(23) = kdeg*x(15);
97 vx(24) = kdeg*x(18);
98
99 dxdt = N*vx;
```

```
1  function dxdt = odefun_red(t, xred)
2
3  k1 = 1/10;
4  k_2 = 1/10;
5  k3 = 1/10;
6  k4 = 1/10;
7  k5 = 1/10;
8  k6 = 1/10;
9  k7 = 1/10;
10 k_8 = 1/10;
11 k9 = 1/10;
12 k10 = 1/10;
13 k11 = 1/10;
14 k_12 = 1/10;
15 k13 = 1/10;
16 k14 = 1/10;
17 k15 = 1/10;
18 k_16 = 1/10;
19 k17 = 1/10;
20 k18 = 1/10;
21 k19 = 1/10;
22 kdeg = 1/10;
23 gamma = 1/100;
24
25 % Flux rates ( [3] )
26
27 % k1 = 249.995;
28 % k_1 = 1.88;
29 % k2 = 51.264;
30 % k3 = 3.1441;
31 % k4 = 1/10;
32 % k5 = 3.4445;
33 % k6 = 0.0390;
34 % k_6 = 28.837;
35 % k7 = 1/10;
36 % k8 = 3.499;
37 % k9 = 72.3162;
38 % k10 = 0.361;
```

```
% k11 = 49.944;
% k12 = 99.951;
% k_12 = 25.414;
% k13 = 3.4906;
% k14 = 494.256;
% k15 = 0.999;
% kdeg = 0.997;

N = zeros(22,24);
N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;

N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
N(20,19) = -1; N(21,19) = 1;
N(10,20) = -1;
N(8,21) = -1;
N(13,22) = -1;
N(15,23) = -1;
N(18,24) = -1;


a_left = null(N');

% Determine constants red. order model

c1 = 0; c2 = 0; c3 = 0; c4 = 0; c5 = 0; c6 =0;

for k = 1 :22
    c1 = c1 + a_left(k,1)*xred(k);
    c2 = c2 + a_left(k,2)*xred(k);
    c3 = c3 + a_left(k,3)*xred(k);
    c4 = c4 + a_left(k,4)*xred(k);
    c5 = c5 + a_left(k,5)*xred(k);
    c6 = c6 + a_left(k,6)*xred(k);
end

Temp1 = 0; Temp2 = 0; Temp3 = 0; Temp4 = 0; Temp5 = 0; Temp6 = 0;

```

```
92  for k = 1:22
93      if k ~=7
94          Temp1 = Temp1 + a_left(k,1)*xred(k);
95      end
96      if k~=2
97          Temp2 = Temp2 + a_left(k,2)*xred(k);
98      end
99      if k~=3
100         Temp3 = Temp3 + a_left(k,3)*xred(k);
101     end
102     if k~=4
103         Temp4 = Temp4 + a_left(k,4)*xred(k);
104     end
105     if k~=5
106         Temp5 = Temp5 + a_left(k,5)*xred(k);
107     end
108     if k~=6
109         Temp6 = Temp6 + a_left(k,6)*xred(k);
110     end
111 end
112
113 % xred(1) = (c1 - Temp1 ) / a_left(1,1);
114 xred(2) = (c2 - Temp2 ) / a_left(2,2);
115 xred(3) = (c3 - Temp3 ) / a_left(3,3);
116 xred(4) = (c4 - Temp4 ) / a_left(4,4);
117 xred(5) = (c5 - Temp5 ) / a_left(5,5);
118 xred(6) = (c6 - Temp6 ) / a_left(6,6);
119 xred(7) = (c1 - Temp1) / a_left(7,1);
120
121
122 vxred = zeros(24,1);
123 vxred(1) = k1*xred(1)*xred(2);
124 vxred(2) = k_2*xred(6);
125 vxred(3) = k3*xred(6)*xred(3);
126 vxred(4) = k4*xred(7);
127 vxred(5) = k5*xred(6)*xred(4);
128 vxred(6) = k6*xred(9);
129 vxred(7) = k7*xred(11)*xred(17);
130 vxred(8) = k_8*xred(1);
131 vxred(9) = k9*xred(11)*xred(5);
132 vxred(10) = k10*xred(22);
133 vxred(11) = k11*xred(10)*xred(8);
134 vxred(12) = k_12 * xred(15);
135 vxred(13) = k13*xred(15)*xred(14);
136 vxred(14) = k14*xred(16);
137 vxred(15) = k15*xred(8)*xred(13);
138 vxred(16) = k_16*xred(18);
139 vxred(17) = k17*xred(18)*xred(14);
140 vxred(18) = k18*xred(19);
141 vxred(19) = k19*xred(20);
142 vxred(20) = kdeg*xred(10);
143 vxred(21) = kdeg*xred(8);
144 vxred(22) = kdeg*xred(13);
```

```
vxred(23) = kdeg*xred(15);
vxred(24) = kdeg*xred(18);

dxdt = N*vxred;
```

C-2 QSSA

```
%% Clean up

clear all; close all; clc;
%% Bar Graph for computed RMS values
RMS = load ('RMS_values2.mat') ;
RMS = cell2mat(struct2cell(RMS));
RMS = RMS(:, 1:5) ;

x_bar = categorical( { '1 - x_{8, red,i }/x_8 [RNA_t] ', ...
    '1 - x_{10, red,i}/x_{10} [RNA_{S28}] ', ...
    '1 - x_{15, red, i}/x_{15} [RNA_t : RNA_{S28}] ' , ...
    '1 - x_{21, red, i}/x_{21} [e_{GFP}] ', ...
    '1 - x_{18, red, i}/x_{18}  [RNA_t: RNA_{eGFP}] '  }) ;

 x_bar = reordercats(x_bar,{ '1 - x_{8, red,i }/x_8 [RNA_t] ', ...
    '1 - x_{10, red,i}/x_{10} [RNA_{S28}] ', ...
    '1 - x_{15, red, i}/x_{15} [RNA_t : RNA_{S28}] ' , ...
    '1 - x_{21, red, i}/x_{21} [e_{GFP}] ', ...
    '1 - x_{18, red, i}/x_{18}  [RNA_t: RNA_{eGFP}] '  });

% set(x_bar,'TickLabelInterpreter','latex');
figure(2);
b = bar(x_bar, RMS) ;
ylabel('Relative errors for system order reduction by removing i states')
      ; %  , 'Interpreter','latex'


set(b(1,1), 'facecolor', [0.71 0.87 0.36] );
set(b(1,2), 'facecolor', [0.84 0.85 0.10] );
set(b(1,3), 'facecolor', [0.93 0.69 0.13] );
set(b(1,4), 'facecolor', [0.87 0.56 0.29] );
set(b(1,5), 'facecolor', [0.84 0.36 0.26] );

set(b, {'DisplayName'},...
    {'i = 1', ...
     'i = 2', ...
     'i = 3' , ...
     'i = 4' ,...
     'i = 5'}');

legend()


%% Unused

```

```
% set(b(7:11), 'facecolor', 'y');
%
% b.FaceColor = 'flat';
% b.CData(2,:) =  [.5 0 .5];

% RMS = [ RMS ;  zeros(1,5) ];
% RMS_vector = RMS(:);

% x_bar = categorical({'x_{8,1}', 'x_{8,2}', 'x_{8,3}', 'x_{8,4}', 'x_{8,5}', '1' ...
%     'x_{10,1}', 'x_{10,2}', 'x_{10,3}', 'x_{10,4}', 'x_{10,5}', '2', ...
% 'x_{15,1}', 'x_{15,2}', 'x_{15,3}', 'x_{15,4}', 'x_{15,5}', '3',...
% 'x_{21,1}', 'x_{21,2}', 'x_{21,3}', 'x_{21,4}', 'x_{21,5}', '4', ...
% 'x_{18,1}', 'x_{18,2}', 'x_{18,3}', 'x_{18,4}', 'x_{18,5}', '5'
% } );
% x_bar = reordercats(x_bar,{'x_{8,1}', 'x_{8,2}', 'x_{8,3}', 'x_{8,4}', 'x_{8,5}', '1' ...
%     'x_{10,1}', 'x_{10,2}', 'x_{10,3}', 'x_{10,4}', 'x_{10,5}', '2', ...
% 'x_{15,1}', 'x_{15,2}', 'x_{15,3}', 'x_{15,4}', 'x_{15,5}', '3',...
% 'x_{21,1}', 'x_{21,2}', 'x_{21,3}', 'x_{21,4}', 'x_{21,5}', '4', ...
% 'x_{18,1}', 'x_{18,2}', 'x_{18,3}', 'x_{18,4}', 'x_{18,5}', '5'
% });


%
%
% % will produce a different color for each bar. If you want to set your own colors then simply do:

%
```

```
function dxdt = odefun2(t, x)

k1 = 1/10;
k_2 = 1/10;
k3 = 1/10;
k4 = 1/10;
k5 = 1/10;
k6 = 1/10;
k7 = 1/10;
k_8 = 1/10;
k9 = 1/10;
k10 = 1/10;
k11 = 1/10;
k_12 = 1/10;
k13 = 1/10;
k14 = 1/10;
k15 = 1/10;
k_16 = 1/10;
k17 = 1/10;
k18 = 1/10;
```

```
k19 = 1/10;
kdeg = 1/10;
gamma = 0;
xin =
    [0.000559909681997316;3.98855518598468;1.99542030372759;1.99542030372759;1.99994815
    e-05;1;5.20169112799691e-05;2.99998690320702;4.20699689366722e
    -06;1.27298311366668e-05;19.6987451094137;1.21270192347726e
    -07;3.66961843422663e-07;3.70139734764438e
    -07;8.38789844363369;5.18481459814413e-05];

N = zeros(22,24);
N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;

N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
N(20,19) = -1; N(21,19) = 1;
N(10,20) = -1;
N(8,21) = -1;
N(13,22) = -1;
N(15,23) = -1;
N(18,24) = -1;

vx = zeros(24,1);
vx(1) = k1*x(1)*x(2);
vx(2) = k_2*x(6);
vx(3) = k3*x(6)*x(3);
vx(4) = k4*x(7);
vx(5) = k5*x(6)*x(4);
vx(6) = k6*x(9);
vx(7) = k7*x(11)*x(17);
vx(8) = k_8*x(1);
vx(9) = k9*x(11)*x(5);
vx(10) = k10*x(22);
vx(11) = k11*x(10)*x(8);
vx(12) = k_12 * x(15);
vx(13) = k13*x(15)*x(14);
vx(14) = k14*x(16);
vx(15) = k15*x(8)*x(13);
```

```
69 vx(16) = k_16*x(18);
70 vx(17) = k17*x(18)*x(14);
71 vx(18) = k18*x(19);
72 vx(19) = k19*x(20);
73 vx(20) = kdeg*x(10);
74 vx(21) = kdeg*x(8);
75 vx(22) = kdeg*x(13);
76 vx(23) = kdeg*x(15);
77 vx(24) = kdeg*x(18);
78
79 xin = ones(22,1);
80 dxdt = N*vx + gamma*(xin - x) ;
```

```
1 %% Clean up
2 clear all; close all; clc;
3 %%  Ode23 solver based solution of xdot = Nvx
4
5 p = 10; % p-percent confidence interval
6 n = 500; % number of timesteps
7 tspan = [0 n-1];
8 x0 = 1*ones(22,1);
9 [t_ode,x_ode] = ode23(@odefun2, tspan, x0);
10 [t_red_ode1, x_red_ode1] = ode23(@QSSA_fun1, tspan, x0);
11 [t_red_ode2, x_red_ode2] = ode23(@QSSA_fun2, tspan, x0);
12 [t_red_ode3, x_red_ode3] = ode23(@QSSA_fun3, tspan, x0);
13 [t_red_ode4, x_red_ode4] = ode23(@QSSA_fun4, tspan, x0);
14 % [t_red_ode5, x_red_ode5] = ode23(@QSSA_fun5, tspan, x0);
15 % [t_red_ode6, x_red_ode6] = ode23(@QSSA_fun6, tspan, x0);
16 % [t_red_ode7, x_red_ode7] = ode23(@QSSA_fun7, tspan, x0);
17 % [t_red_ode8, x_red_ode8] = ode23(@QSSA_fun8, tspan, x0);
18
19
20
21 %% Plotting (for matlab ODE solvers)
22
23 figure(1);
24
25 subplot(2,3,1);
26
27 v11 = (1-p/100)*x_ode(:,8) ;
28 v12= (1+p/100)*x_ode(:,8) ;
29 xCoords = [t_ode'  (fliplr(t_ode'))  ];
30 yCoords = [v11' fliplr(v12')  ];
31 % hPatch = fill(xCoords,yCoords , [1, 1, 0.51],'EdgeColor','y');
32 hold on
33 p1 = plot(t_ode,x_ode(:,8),  'linewidth', 2 ); % Plot of state x8 for n
    timesteps
34 plot(t_red_ode1, x_red_ode1(:,8), 'linewidth', 2  ) % Plot of state x8
    for n timesteps (red. order model)
35 plot(t_red_ode2, x_red_ode2(:,8), 'linewidth', 2  ) % Plot of state x8
    for n timesteps (red. order model)
36 plot(t_red_ode3, x_red_ode3(:,8), 'linewidth', 2  ) % Plot of state x8
    for n timesteps (red. order model)
```

```
37 plot(t_red_ode4, x_red_ode4(:,8), 'linewidth', 2 ) % Plot of state x8 for
       n timesteps (red. order model)
38 % plot(t_red_ode5, x_red_ode5(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
39 % plot(t_red_ode6, x_red_ode6(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
40 % plot(t_red_ode7, x_red_ode7(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
41 % plot(t_red_ode8, x_red_ode8(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
42 hold off
43 xlabel('Timestep');
44 ylabel('[RNA_t ]    (x_8)');
45 legend('Non-reduced system: $x_8$', '1 removed state', '2 removed states'
       , '3 removed states', '4 removed states', 'interpreter', 'latex'); %,
       '2 removed states', '3 removed states', '4 removed states', '5 removed
       states', 'interpreter', 'latex');
46
47 subplot(2,3,2);
48 v101 = (1-p/100)*x_ode(:,10) ;
49 v102= (1+p/100)*x_ode(:,10) ;
50 xCoords = [t_ode'  (fliplr(t_ode'))  ];
51 yCoords = [v101' fliplr(v102')  ];
52 % hPatch = fill(xCoords,yCoords , [1, 1, 0.51],'EdgeColor','y');
53 hold on
54 p2 = plot(t_ode,x_ode(:,10), 'linewidth', 2 ); % Plot of state x10 for n
       timesteps
55 plot(t_red_ode1, x_red_ode1(:,10), 'linewidth', 2  ) % Plot of state x10
       for n timesteps (red. order model)
56 plot(t_red_ode2, x_red_ode2(:,10), 'linewidth', 2  ) % Plot of state x10
       for n timesteps (red. order model)
57 plot(t_red_ode3, x_red_ode3(:,10), 'linewidth', 2  ) % Plot of state x10
       for n timesteps (red. order model)
58 plot(t_red_ode4, x_red_ode4(:,10), 'linewidth', 2  ) % Plot of state x10
       for n timesteps (red. order model)
59 % plot(t_red_ode5, x_red_ode5(:,10), 'linewidth', 2  ) % Plot of state
       x10 for n timesteps (red. order model)
60 % plot(t_red_ode6, x_red_ode6(:,10), 'linewidth', 2  ) % Plot of state
       x10 for n timesteps (red. order model)
61 % plot(t_red_ode7, x_red_ode7(:,10), 'linewidth', 2  ) % Plot of state
       x10 for n timesteps (red. order model)
62 % plot(t_red_ode8, x_red_ode8(:,10), 'linewidth', 2  ) % Plot of state
       x10 for n timesteps (red. order model)
63 hold off
64 xlabel('Timestep');
65 ylabel('[RNA_{S28}]    (x_{10})');
66 legend('Non-reduced system: $x_{10}$', '1 removed state', '2 removed
       states', '3 removed states', '4 removed states', 'interpreter', 'latex
       '); %, '2 removed states', '3 removed states', '4 removed states', '5
       removed states', 'interpreter', 'latex');
67
68
69 subplot(2,3,3);
```

```
70 v151 = (1-p/100)*x_ode(:,15) ;
71 v152= (1+p/100)*x_ode(:,15) ;
72 xCoords = [t_ode'  (fliplr(t_ode'))  ];
73 yCoords = [v151' fliplr(v152')  ];
74 % hPatch = fill(xCoords,yCoords , [1, 1, 0.51],'EdgeColor','y');
75 hold on
76 p5 = plot(t_ode,x_ode(:,15), 'linewidth', 2  ); % Plot of state x15 for n
       timesteps
77 plot(t_red_ode1, x_red_ode1(:,15), 'linewidth', 2  ) % Plot of state x15
      for n timesteps (red. order model)
78 plot(t_red_ode2, x_red_ode2(:,15), 'linewidth', 2  ) % Plot of state x15
      for n timesteps (red. order model)
79 plot(t_red_ode3, x_red_ode3(:,15), 'linewidth', 2  ) % Plot of state x15
      for n timesteps (red. order model)
80 plot(t_red_ode4, x_red_ode4(:,15), 'linewidth', 2  ) % Plot of state x15
      for n timesteps (red. order model)
81 % plot(t_red_ode5, x_red_ode5(:,15), 'linewidth', 2  ) % Plot of state
      x15 for n timesteps (red. order model)
82 % plot(t_red_ode6, x_red_ode6(:,15), 'linewidth', 2  ) % Plot of state
      x15 for n timesteps (red. order model)
83 % plot(t_red_ode7, x_red_ode7(:,15), 'linewidth', 2  ) % Plot of state
      x15 for n timesteps (red. order model)
84 % plot(t_red_ode8, x_red_ode8(:,15), 'linewidth', 2  ) % Plot of state
      x15 for n timesteps (red. order model)
85 hold off
86 xlabel('Timestep');
87 ylabel('[RNA_t:RNA_{S28}]     (x_{14})');
88 legend('Non-reduced system: $x_{14}$', '1 removed state', '2 removed
      states', '3 removed states', '4 removed states', 'interpreter', 'latex
      '); %, '2 removed states', '3 removed states', '4 removed states', '5
      removed states', 'interpreter', 'latex');
89
90
91 subplot(2,3,4:5);
92 v201 = (1-p/100)*x_ode(:,20) ;
93 v202= (1+p/100)*x_ode(:,20) ;
94 xCoords = [t_ode'  (fliplr(t_ode'))  ];
95 yCoords = [v201' fliplr(v202')  ];
96 % hPatch = fill(xCoords,yCoords , [1, 1, 0.51] ,'EdgeColor','y');
97 hold on
98 p3 = plot(t_ode,x_ode(:,21), 'linewidth', 2 ); % Plot of state x15 for n
      timesteps
99 plot(t_red_ode1, x_red_ode1(:,21), 'linewidth', 2  ) % Plot of state x20
      for n timesteps (red. order model)
100 plot(t_red_ode2, x_red_ode2(:,21), 'linewidth', 2  ) % Plot of state x20
      for n timesteps (red. order model)
101 plot(t_red_ode3, x_red_ode3(:,21), 'linewidth', 2  ) % Plot of state x20
      for n timesteps (red. order model)
102 plot(t_red_ode4, x_red_ode4(:,21), 'linewidth', 2  ) % Plot of state x20
      for n timesteps (red. order model)
103 % plot(t_red_ode5, x_red_ode5(:,21), 'linewidth', 2  ) % Plot of state
      x20 for n timesteps (red. order model)
```

```
104 % plot(t_red_ode6, x_red_ode6(:,21), 'linewidth', 2 ) % Plot of state
        x20 for n timesteps (red. order model)
105 % plot(t_red_ode7, x_red_ode7(:,21), 'linewidth', 2 ) % Plot of state
        x20 for n timesteps (red. order model)
106 % plot(t_red_ode8, x_red_ode8(:,21), 'linewidth', 2 ) % Plot of state
        x20 for n timesteps (red. order model)
107 hold off
108 xlabel('Timestep');
109 ylabel('[e_{GFP}]    (x_{20})');
110 legend('Non-reduced system: $x_{20}$', '1 removed state', '2 removed
        states', '3 removed states', '4 removed states', 'interpreter', 'latex
        '); %, '2 removed states', '3 removed states', '4 removed states', '5
        removed states', 'interpreter', 'latex');
111
112 subplot(2,3,6);
113 v181 = (1-p/100)*x_ode(:,18) ;
114 v182= (1+p/100)*x_ode(:,18) ;
115 xCoords = [t_ode'  (fliplr(t_ode'))  ];
116 yCoords = [v181' fliplr(v182')  ];
117 % hPatch = fill(xCoords,yCoords , [1, 1, 0.51],'EdgeColor','y');
118 hold on
119 p4 = plot(t_ode,x_ode(:,18), 'linewidth', 2 ); % Plot of state x18 for n
        timesteps
120 plot(t_red_ode1, x_red_ode1(:,18), 'linewidth', 2 ) % Plot of state x18
        for n timesteps (red. order model)
121 plot(t_red_ode2, x_red_ode2(:,18), 'linewidth', 2 ) % Plot of state x18
        for n timesteps (red. order model)
122 plot(t_red_ode3, x_red_ode3(:,18), 'linewidth', 2 ) % Plot of state x18
        for n timesteps (red. order model)
123 plot(t_red_ode4, x_red_ode4(:,18), 'linewidth', 2 ) % Plot of state x18
        for n timesteps (red. order model)
124 % plot(t_red_ode5, x_red_ode5(:,18), 'linewidth', 2 ) % Plot of state
        x18 for n timesteps (red. order model)
125 % plot(t_red_ode6, x_red_ode6(:,18), 'linewidth', 2 ) % Plot of state
        x18 for n timesteps (red. order model)
126 % plot(t_red_ode7, x_red_ode7(:,18), 'linewidth', 2 ) % Plot of state
        x18 for n timesteps (red. order model)
127 % plot(t_red_ode8, x_red_ode8(:,18), 'linewidth', 2 ) % Plot of state
        x18 for n timesteps (red. order model)
128 hold off
129 xlabel('Timestep');
130 ylabel('[RNA_t : RNA_{eGFP}] (x_{17})');
131 legend('Non-reduced system: $x_{17}$', '1 removed state', '2 removed
        states', '3 removed states', '4 removed states', 'interpreter', '
        latex'); %, '2 removed states', '3 removed states', '4 removed states
        ', '5 removed states', 'interpreter', 'latex');
132 sgtitle('Plot of reduction order model compared with non-reduced system',
        'interpreter', 'latex');
133
134
135 %% Reduction order method QSSA
136
137 xdot = []; %zeros( length(x_ode), 22) ;
```

```
138
139 for k =1:length(x_ode)
140 xdot = [ xdot QSSA_fun(0, x_ode(k,:)') ];
141 end
142
143 xdot_mean =[];
144
145 for k = 1:22
146 xdot_mean = [ xdot_mean ; mean( xdot(k,:) ) ] ;
147 end
148
149 [min, pos] = min( abs( xdot_mean) );
150 [B, I] = sort( abs( xdot_mean) );
151 X = ['Order reduction by resp. removing the following states from the
    system of equations.'];
152 Y = string(I);
153 disp(X)
154 disp(Y)
155 %% Define interpolations for all modelled x
156
157 xq = 1:n ; % x coords interpolated system
158
159 x = t_ode; % x coords original system
160 x1 = t_red_ode1;
161 x2 = t_red_ode2;
162 x3 = t_red_ode3;
163 x4 = t_red_ode4;
164 % x5 = t_red_ode5;
165 % x6 = t_red_ode6;
166 % x7 = t_red_ode7;
167 % x8 = t_red_ode8;
168
169 % state 8
170
171 y08 = x_ode(:,8) ; % y Coords original system
172 y18 = x_red_ode1(:,8);
173 y28 = x_red_ode2(:,8);
174 y38 = x_red_ode3(:,8);
175 y48 = x_red_ode4(:,8);
176 % y58 = x_red_ode5(:,8);
177 % y68 = x_red_ode6(:,8);
178 % y78 = x_red_ode7(:,8);
179 % y88 = x_red_ode8(:,8);
180
181 yq08 = interp1(x,y08,xq); % y Coords interpolated system
182 yq18 = interp1(x1,y18,xq) ;
183 yq28 = interp1(x2, y28, xq);
184 yq38 = interp1(x3, y38, xq);
185 yq48 = interp1(x4, y48, xq);
186 %  yq58 = interp1(x5, y58, xq);
187 % yq68 = interp1(x6, y68, xq);
188 % yq78 = interp1(x7, y78, xq);
189 % yq88 = interp1(x8, y88, xq);
```

```
190
191 % state 10
192
193 y010 = x_ode(:,10) ; % y Coords original system
194 y110 = x_red_ode1(:,10);
195 y210 = x_red_ode2(:,10);
196 y310 = x_red_ode3(:,10);
197 y410 = x_red_ode4(:,10);
198 %  y510 = x_red_ode5(:,10);
199 % y610 = x_red_ode6(:,10);
200 % y710 = x_red_ode7(:,10);
201 % y810 = x_red_ode8(:,10);
202
203 yq010 = interp1(x,y010,xq); % y Coords interpolated system
204 yq110 = interp1(x1,y110,xq) ;
205 yq210 = interp1(x2, y210, xq);
206 yq310 = interp1(x3, y310, xq);
207 yq410 = interp1(x4, y410, xq);
208 %  yq510 = interp1(x5, y510, xq);
209 % yq610 = interp1(x6, y610, xq);
210 % yq710 = interp1(x7, y710, xq);
211 % yq810 = interp1(x8, y810, xq);
212
213 % state 15
214
215 y015 = x_ode(:,15) ; % y Coords original system
216 y115 = x_red_ode1(:,15);
217 y215 = x_red_ode2(:,15);
218 y315 = x_red_ode3(:,15);
219 y415 = x_red_ode4(:,15);
220 %  y515 = x_red_ode5(:,15);
221 % y615 = x_red_ode6(:,15);
222 % y715 = x_red_ode7(:,15);
223 % y815 = x_red_ode8(:,15);
224
225 yq015 = interp1(x,y015,xq); % y Coords interpolated system
226 yq115 = interp1(x1,y115,xq) ;
227 yq215 = interp1(x2, y215, xq);
228 yq315 = interp1(x3, y315, xq);
229 yq415 = interp1(x4, y415, xq);
230 %  yq515 = interp1(x5, y515, xq);
231 % yq615 = interp1(x6, y615, xq);
232 % yq715 = interp1(x7, y715, xq);
233 % yq815 = interp1(x8, y815, xq);
234
235 % state 21
236
237 y021 = x_ode(:,21) ; % y Coords original system
238 y121 = x_red_ode1(:,21);
239 y221 = x_red_ode2(:,21);
240 y321 = x_red_ode3(:,21);
241 y421 = x_red_ode4(:,21);
242 %  y521 = x_red_ode5(:,21);
```

```
% y621 = x_red_ode6(:,21);
% y721 = x_red_ode7(:,21);
% y821 = x_red_ode8(:,21);

yq021 = interp1(x,y021,xq); % y Coords interpolated system
yq121 = interp1(x1,y121,xq) ;
yq221 = interp1(x2, y221, xq);
yq321 = interp1(x3, y321, xq);
yq421 = interp1(x4, y421, xq);
%  yq521 = interp1(x5, y521, xq);
% yq621 = interp1(x6, y621, xq);
% yq721 = interp1(x7, y721, xq);
% yq821 = interp1(x8, y821, xq);

% state 18

y018 = x_ode(:,18) ; % y Coords original system
y118 = x_red_ode1(:,18);
y218 = x_red_ode2(:,18);
y318 = x_red_ode3(:,18);
y418 = x_red_ode4(:,18);
%  y518 = x_red_ode5(:,18);
% y618 = x_red_ode6(:,18);
% y718 = x_red_ode7(:,18);
% y818 = x_red_ode8(:,18);

yq018 = interp1(x,y018,xq); % y Coords interpolated system
yq118 = interp1(x1,y118,xq) ;
yq218 = interp1(x2, y218, xq);
yq318 = interp1(x3, y318, xq);
yq418 = interp1(x4, y418, xq);
%  yq518 = interp1(x5, y518, xq);
% yq618 = interp1(x6, y618, xq);
% yq718 = interp1(x7, y718, xq);
% yq818 = interp1(x8, y818, xq);

%% Define errors wrt number of removed states


yq08 = yq08(1:end-1) ; % y Coords interpolated system
yq18 = yq18(1:end-1) ;
yq28 = yq28(1:end-1);
yq38 = yq38(1:end-1);
yq48 = yq48(1:end-1);
%  yq58 = yq58(1:end-1);
% yq68 = yq68(1:end-1);
% yq78 = yq78(1:end-1);
% yq88 = yq88(1:end-1);

yq010 = yq010(1:end-1) ; % y Coords interpolated system
yq110 = yq110(1:end-1) ;
yq210 = yq210(1:end-1);
yq310 = yq310(1:end-1);
```

```
296 yq410 = yq410(1:end-1);
297 %  yq510 = yq510(1:end-1);
298 % yq610 = yq610(1:end-1);
299 % yq710 = yq710(1:end-1);
300 % yq810 = yq810(1:end-1);
301
302 yq015 = yq015(1:end-1) ; % y Coords interpolated system
303 yq115 = yq115(1:end-1) ;
304 yq215 = yq215(1:end-1);
305 yq315 = yq315(1:end-1);
306 yq415 = yq415(1:end-1);
307 %  yq515 = yq515(1:end-1);
308 % yq615 = yq615(1:end-1);
309 % yq715 = yq715(1:end-1);
310 % yq815 = yq815(1:end-1);
311
312 yq021 = yq021(1:end-1) ; % y Coords interpolated system
313 yq121 = yq121(1:end-1) ;
314 yq221 = yq221(1:end-1);
315 yq321 = yq321(1:end-1);
316 yq421 = yq421(1:end-1);
317 %  yq521 = yq521(1:end-1);
318 % yq621 = yq621(1:end-1);
319 % yq721 = yq721(1:end-1);
320 % yq821 = yq821(1:end-1);
321
322 yq018 = yq018(1:end-1) ; % y Coords interpolated system
323 yq118 = yq118(1:end-1) ;
324 yq218 = yq218(1:end-1);
325 yq318 = yq318(1:end-1);
326 yq418 = yq418(1:end-1);
327 %  yq518 = yq518(1:end-1);
328 % yq618 = yq618(1:end-1);
329 % yq718 = yq718(1:end-1);
330 % yq818 = yq818(1:end-1);
331
332 %%
333
334 rms_18 = abs( (   (yq08 - yq18) ./ yq08)*( (yq08 - yq18) ./ yq08)' )/n; %
      sqrt(  ( yq08 - yq18)*(yq08-yq18)' )/yq08  ;
335 rms_28 = abs( (   (yq08 - yq28) ./ yq08)*( (yq08 - yq28) ./ yq08)' )/n; %
      sqrt(  ( yq08 - yq28)*(yq08-yq28)' )/yq08  ;
336 rms_38 = abs( (   (yq08 - yq38) ./ yq08)*( (yq08 - yq38) ./ yq08)' )/n ;
    % sqrt(  ( yq08 - yq38)*(yq08-yq38)' )/yq08  ;
337 rms_48 = abs( (   (yq08 - yq48) ./ yq08)*( (yq08 - yq48) ./ yq08)' )/n ;
    % sqrt(  ( yq08 - yq48)*(yq08-yq48)' )/yq08  ;
338 % rms_58 = abs( (   (yq08 - yq58) ./ yq08)*( (yq08 - yq58) ./ yq08)' )/n
      ; % sqrt(  ( yq08 - yq68)*(yq08-yq68)' )/yq08  ;
339 % rms_68 = abs( (   (yq08 - yq68) ./ yq08)*( (yq08 - yq68) ./ yq08)' )/n
      ; % sqrt(  ( yq08 - yq68)*(yq08-yq68)' )/yq08  ;
340 % rms_78 = abs( (   (yq08 - yq78) ./ yq08)*( (yq08 - yq78) ./ yq08)' )/n
      ; % sqrt(  ( yq08 - yq68)*(yq08-yq68)' )/yq08  ;
```

```
341 % rms_88 = abs( (    (yq08 - yq88) ./ yq08)*( (yq08 - yq88) ./ yq08)' )/n
        ; % sqrt(  ( yq08 - yq68)*(yq08-yq68)' )/yq08  ;
342
343
344 rms_110 = abs( (    (yq010 - yq110) ./ yq010)*( (yq010 - yq110) ./ yq010)'
        )/n ; % sqrt(  ( yq010 - yq110)*(yq010-yq110)' )/yq010  ;
345 rms_210 = abs( (    (yq010 - yq210) ./ yq010)*( (yq010 - yq210) ./ yq010)'
        )/n ; % sqrt(  ( yq010 - yq200)*(yq010-yq200)' )/yq010  ;
346 rms_310 = abs( (    (yq010 - yq310) ./ yq010)*( (yq010 - yq310) ./ yq010)'
        )/n ; % sqrt(  ( yq010 - yq310)*(yq010-yq310)' )/yq010  ;
347 rms_410 = abs( (    (yq010 - yq410) ./ yq010)*( (yq010 - yq410) ./ yq010)'
        )/n ; % sqrt(  ( yq010 - yq410)*(yq010-yq410)' )/yq010  ;
348 % rms_510 = abs( (    (yq010 - yq510) ./ yq010)*( (yq010 - yq510) ./ yq010
        )' ) /n; % sqrt(  ( yq010 - yq610)*(yq010-yq610)' )/yq010  ;
349 % rms_610 = abs( (    (yq010 - yq610) ./ yq010)*( (yq010 - yq610) ./ yq010
        )' ) /n; % sqrt(  ( yq010 - yq610)*(yq010-yq610)' )/yq010  ;
350 % rms_710 = abs( (    (yq010 - yq710) ./ yq010)*( (yq010 - yq710) ./ yq010
        )' ) /n; % sqrt(  ( yq010 - yq610)*(yq010-yq610)' )/yq010  ;
351 % rms_810 = abs( (    (yq010 - yq810) ./ yq010)*( (yq010 - yq810) ./ yq010
        )' ) /n; % sqrt(  ( yq010 - yq610)*(yq010-yq610)' )/yq010  ;
352
353 rms_115 = abs( (    (yq015 - yq115) ./ yq015)*( (yq015 - yq115) ./ yq015)'
        )/n ; % sqrt(  ( yq015 - yq115)*(yq015-yq115)' )/yq015  ;
354 rms_215 = abs( (    (yq015 - yq215) ./ yq015)*( (yq015 - yq215) ./ yq015)'
        )/n ; % sqrt(  ( yq015 - yq200)*(yq015-yq200)' )/yq015  ;
355 rms_315 = abs( (    (yq015 - yq315) ./ yq015)*( (yq015 - yq315) ./ yq015)'
        )/n ; % sqrt(  ( yq015 - yq315)*(yq015-yq315)' )/yq015  ;
356 rms_415 = abs( (    (yq015 - yq415) ./ yq015)*( (yq015 - yq415) ./ yq015)'
        )/n ; % sqrt(  ( yq015 - yq415)*(yq015-yq415)' )/yq015  ;
357 % rms_515 = abs( (    (yq015 - yq515) ./ yq015)*( (yq015 - yq515) ./ yq015
        )' ) /n; % sqrt(  ( yq015 - yq615)*(yq015-yq615)' )/yq015  ;
358 % rms_615 = abs( (    (yq015 - yq615) ./ yq015)*( (yq015 - yq615) ./ yq015
        )' ) /n; % sqrt(  ( yq015 - yq615)*(yq015-yq615)' )/yq015  ;
359 % rms_715 = abs( (    (yq015 - yq715) ./ yq015)*( (yq015 - yq715) ./ yq015
        )' ) /n; % sqrt(  ( yq015 - yq615)*(yq015-yq615)' )/yq015  ;
360 % rms_815 = abs( (    (yq015 - yq815) ./ yq015)*( (yq015 - yq815) ./ yq015
        )' ) /n; % sqrt(  ( yq015 - yq615)*(yq015-yq615)' )/yq015  ;
361
362 rms_121 = abs( (    (yq021 - yq121) ./ yq021)*( (yq021 - yq121) ./ yq021)'
        )/n ; % sqrt(  ( yq021 - yq121)*(yq021-yq121)' )/yq021  ;
363 rms_221 = abs( (    (yq021 - yq221) ./ yq021)*( (yq021 - yq221) ./ yq021)'
        )/n ; % sqrt(  ( yq021 - yq200)*(yq021-yq200)' )/yq021  ;
364 rms_321 = abs( (    (yq021 - yq321) ./ yq021)*( (yq021 - yq321) ./ yq021)'
        )/n ; % sqrt(  ( yq021 - yq321)*(yq021-yq321)' )/yq021  ;
365 rms_421 = abs( (    (yq021 - yq421) ./ yq021)*( (yq021 - yq421) ./ yq021)'
        )/n ; % sqrt(  ( yq021 - yq421)*(yq021-yq421)' )/yq021  ;
366 % rms_521 = abs( (    (yq021 - yq521) ./ yq021)*( (yq021 - yq521) ./ yq021
        )' ) /n; % sqrt(  ( yq021 - yq621)*(yq021-yq621)' )/yq021  ;
367 % rms_621 = abs( (    (yq021 - yq621) ./ yq021)*( (yq021 - yq621) ./ yq021
        )' ) /n; % sqrt(  ( yq021 - yq621)*(yq021-yq621)' )/yq021  ;
368 % rms_721 = abs( (    (yq021 - yq721) ./ yq021)*( (yq021 - yq721) ./ yq021
        )' ) /n; % sqrt(  ( yq021 - yq621)*(yq021-yq621)' )/yq021  ;
```

```
% rms_821 = abs( (   (yq021 - yq821) ./ yq021)*( (yq021 - yq821) ./ yq021 )' ) /n; % sqrt(  ( yq021 - yq621)*(yq021-yq621)' )/yq021  ;

rms_118 = abs( (   (yq018 - yq118) ./ yq018)*( (yq018 - yq118) ./ yq018)' )/n ; % sqrt(  ( yq018 - yq118)*(yq018-yq118)' )/yq018  ;
rms_218 = abs( (   (yq018 - yq218) ./ yq018)*( (yq018 - yq218) ./ yq018)' )/n ; % sqrt(  ( yq018 - yq200)*(yq018-yq200)' )/yq018  ;
rms_318 = abs( (   (yq018 - yq318) ./ yq018)*( (yq018 - yq318) ./ yq018)' )/n ; % sqrt(  ( yq018 - yq318)*(yq018-yq318)' )/yq018  ;
rms_418 = abs( (   (yq018 - yq418) ./ yq018)*( (yq018 - yq418) ./ yq018)' )/n ; % sqrt(  ( yq018 - yq418)*(yq018-yq418)' )/yq018  ;
% rms_518 = abs( (   (yq018 - yq518) ./ yq018)*( (yq018 - yq518) ./ yq018 )' ) /n; % sqrt(  ( yq018 - yq618)*(yq018-yq618)' )/yq018  ;
% rms_618 = abs( (   (yq018 - yq618) ./ yq018)*( (yq018 - yq618) ./ yq018 )' ) /n; % sqrt(  ( yq018 - yq618)*(yq018-yq618)' )/yq018  ;
% rms_718 = abs( (   (yq018 - yq718) ./ yq018)*( (yq018 - yq718) ./ yq018 )' ) /n; % sqrt(  ( yq018 - yq618)*(yq018-yq618)' )/yq018  ;
% rms_818 = abs( (   (yq018 - yq818) ./ yq018)*( (yq018 - yq818) ./ yq018 )' ) /n; % sqrt(  ( yq018 - yq618)*(yq018-yq618)' )/yq018  ;

RMS = [ rms_18 rms_28 rms_38 rms_48; %  rms_58;  % rms_68 rms_78 rms_88;
    rms_110 rms_210 rms_310 rms_410; % rms_510;  % rms_610 rms_710 rms_810;
    rms_115 rms_215 rms_315 rms_415; %  rms_515; % rms_615 rms_715 rms_815;
    rms_121 rms_221 rms_321 rms_421; %  rms_521; % rms_621 rms_721 rms_821;
    rms_118 rms_218 rms_318 rms_418; %  rms_518; % rms_618 rms_718 rms_818
];

figure(2);
bar(RMS);
```

```
function dxdt = odefun2(t, x)

k1 = 1/10;
k_2 = 1/10;
k3 = 1/10;
k4 = 1/10;
k5 = 1/10;
k6 = 1/10;
k7 = 1/10;
k_8 = 1/10;
k9 = 1/10;
k10 = 1/10;
k11 = 1/10;
k_12 = 1/10;
k13 = 1/10;
k14 = 1/10;
k15 = 1/10;
k_16 = 1/10;
k17 = 1/10;
```

```
20 k18 = 1/10;
21 k19 = 1/10;
22 kdeg = 1/10;
23 gamma = 0;
24 xin =
       [0.000559909681997316;3.98855518598468;1.99542030372759;1.99542030372759;1.99994815
       e-05;1;5.20169112799691e-05;2.99998690320702;4.20699689366722e
       -06;1.27298311366668e-05;19.6987451094137;1.21270192347726e
       -07;3.66961843422663e-07;3.70139734764438e
       -07;8.38789844363369;5.18481459814413e-05];
25
26 N = zeros(22,24);
27 N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
28 N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
29 N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
30 N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
31 N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
32 N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
33 N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
34 N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
35 N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
36 N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
37 N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
38
39 N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
40 N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
41 N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
42 N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
43 N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
44 N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
45 N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
46 N(20,19) = -1; N(21,19) = 1;
47 N(10,20) = -1;
48 N(8,21) = -1;
49 N(13,22) = -1;
50 N(15,23) = -1;
51 N(18,24) = -1;
52
53 vx = zeros(24,1);
54
55 % x(3) = (k4*x(7) + gamma*xin(3))/(gamma + k3*x(6));
56 % x(4) = (gamma*xin(4) + k6*x(9))/(gamma + k5*x(9));
57 % x(7) = (k3*x(3)*x(6) + gamma*xin(7))/(k4 + gamma);
58 % x(9) = (k5*x(4)*x(6) + gamma*xin(9))/(gamma + k5*x(6));
59 % x(6) = (k4x(7) + k6*x(9) + k1*x(1)*x(2) + gamma*xin(6))/(gamma + k_2 +
       k3*x(3) + k5*x(4));
60
61
62
63 vx(1) = k1*x(1)*x(2);
64 vx(2) = k_2*x(6);
65 vx(3) = k3*x(6)*x(3);
66 vx(4) = k4*x(7);
```

```
vx(5) = k5*x(6)*x(4);
vx(6) = k6*x(9);
vx(7) = k7*x(11)*x(17);
vx(8) = k_8*x(1);
vx(9) = k9*x(11)*x(5);
vx(10) = k10*x(22);
vx(11) = k11*x(10)*x(8);
vx(12) = k_12 * x(15);
vx(13) = k13*x(15)*x(14);
vx(14) = k14*x(16);
vx(15) = k15*x(8)*x(13);
vx(16) = k_16*x(18);
vx(17) = k17*x(18)*x(14);
vx(18) = k18*x(19);
vx(19) = k19*x(20);
vx(20) = kdeg*x(10);
vx(21) = kdeg*x(8);
vx(22) = kdeg*x(13);
vx(23) = kdeg*x(15);
vx(24) = kdeg*x(18);

xin = ones(22,1);
dxdt = N*vx + gamma*(xin - x) ;
```

```
%% Clean up
clear all; close all; clc;
%%
x0 = ones(22,1);
n = 1000;
tspan =[0 n-1];
[t_ode,x_ode] = ode23(@odefun2, tspan, x0);
steady_states = x_ode(end,:)';
x1 = ones(22,1); x1(5) = steady_states(5);
x2 = ones(22,1); x2(5) = steady_states(5); x2(11) = steady_states(11);
x3 = ones(22,1); x3(5) = steady_states(5); x3(11) = steady_states(11); x3
    (13) = steady_states(13);
x4 = ones(22,1); x4(5) = steady_states(5); x4(11) = steady_states(11); x4
    (13) = steady_states(13); x4(15) = steady_states(15);
x5 = ones(22,1); x5(5) = steady_states(5); x5(11) = steady_states(11); x5
    (13) = steady_states(13); x5(15) = steady_states(15); x5(18) =
    steady_states(18);
x6 = ones(22,1); x6(5) = steady_states(5); x6(11) = steady_states(11); x6
    (13) = steady_states(13); x6(15) = steady_states(15); x6(18) =
    steady_states(18); x6(19) = steady_states(19);
x7 = ones(22,1); x7(5) = steady_states(5); x7(11) = steady_states(11); x7
    (13) = steady_states(13); x7(15) = steady_states(15); x7(18) =
    steady_states(18); x7(19) = steady_states(19); x7(22) = steady_states
    (22);
[t_ode1,x_ode1] = ode23(@odefun2, tspan, x1);
[t_ode2,x_ode2] = ode23(@odefun2, tspan, x2);
[t_ode3,x_ode3] = ode23(@odefun2, tspan, x3);
[t_ode4,x_ode4] = ode23(@odefun2, tspan, x4);
[t_ode5,x_ode5] = ode23(@odefun2, tspan, x5);
```

```
21 [t_ode6,x_ode6] = ode23(@odefun2, tspan, x6);
22 [t_ode7,x_ode7] = ode23(@odefun2, tspan, x7);
23
24 %lowerbound = 0.9*abs(steady_states);
25 %upperbound = 1.1*abs(steady_states);
26 % sequence of removals: 5,11,12,14,17,18,21
27
28 %%
29 figure(2);
30 subplot(2,3,1);
31 plot(t_ode, x_ode(:,8), 'linewidth', 2);
32 hold on
33 plot(t_ode1, x_ode1(:,8), 'linewidth', 2);
34 plot(t_ode2, x_ode2(:,8), 'linewidth', 2);
35 plot(t_ode3, x_ode3(:,8), 'linewidth', 2);
36 % plot(t_ode4, x_ode4(:,8), 'linewidth', 2);
37 % plot(t_ode5, x_ode5(:,8), 'linewidth', 2);
38 % plot(t_ode6, x_ode6(:,8), 'linewidth', 2);
39 % plot(t_ode7, x_ode7(:,8), 'linewidth', 2);
40 hold off
41 %legend('x_8 original', '1 removed state', '2 rem. states', '3 rem.
       states', '4 rem. states', '5 rem. states', '6 rem. states', '7 rem.
       states');
42 xlabel('Timestep');
43 ylabel('[RNA_t]');
44
45 subplot(2,3,2);
46 plot(t_ode, x_ode(:,10), 'linewidth', 2);
47 hold on
48 plot(t_ode1, x_ode1(:,10), 'linewidth', 2);
49 plot(t_ode2, x_ode2(:,10), 'linewidth', 2);
50 plot(t_ode3, x_ode3(:,10), 'linewidth', 2);
51 plot(t_ode4, x_ode4(:,10), 'linewidth', 2);
52 plot(t_ode5, x_ode5(:,10), 'linewidth', 2);
53 plot(t_ode6, x_ode6(:,10), 'linewidth', 2);
54 plot(t_ode7, x_ode7(:,10), 'linewidth', 2);
55 hold off
56 %legend('x_{10} original', '1 removed state', '2 rem. states', '3 rem.
       states', '4 rem. states', '5 rem. states', '6 rem. states', '7 rem.
       states');
57 xlabel('Timestep');
58 ylabel('[RNA_{S28}]');
59
60 subplot(2,3,3);
61 plot(t_ode, x_ode(:,15), 'linewidth', 2);
62 hold on
63 plot(t_ode1, x_ode1(:,15), 'linewidth', 2);
64 plot(t_ode2, x_ode2(:,15), 'linewidth', 2);
65 plot(t_ode3, x_ode3(:,15), 'linewidth', 2);
66 plot(t_ode4, x_ode4(:,15), 'linewidth', 2);
67 plot(t_ode5, x_ode5(:,15), 'linewidth', 2);
68 plot(t_ode6, x_ode6(:,15), 'linewidth', 2);
69 plot(t_ode7, x_ode7(:,15), 'linewidth', 2);
```

```
70 hold off
71 %legend('x_{15} original', '1 removed state', '2 rem. states', '3 rem.
       states', '4 rem. states', '5 rem. states', '6 rem. states', '7 rem.
       states');
72 xlabel('Timestep');
73 ylabel('[RNA_{t}:RNA_{S28}]');
74
75 subplot(2,3,4:5);
76 plot(t_ode, x_ode(:,21), 'linewidth', 2);
77 hold on
78 plot(t_ode1, x_ode1(:,21), 'linewidth', 2);
79 plot(t_ode2, x_ode2(:,21), 'linewidth', 2);
80 plot(t_ode3, x_ode3(:,21), 'linewidth', 2);
81 plot(t_ode4, x_ode4(:,21), 'linewidth', 2);
82 plot(t_ode5, x_ode5(:,21), 'linewidth', 2);
83 plot(t_ode6, x_ode6(:,21), 'linewidth', 2);
84 plot(t_ode7, x_ode7(:,21), 'linewidth', 2);
85 hold off
86 legend('x_{20} original', '1 removed state', '2 rem. states', '3 rem.
       states', '4 rem. states', '5 rem. states', '6 rem. states', '7 rem.
       states');
87 xlabel('Timestep');
88 ylabel('[e_{GFP}]');
89
90 subplot(2,3,6);
91 plot(t_ode, x_ode(:,18), 'linewidth', 2);
92 hold on
93 plot(t_ode1, x_ode1(:,18), 'linewidth', 2);
94 plot(t_ode2, x_ode2(:,18), 'linewidth', 2);
95 plot(t_ode3, x_ode3(:,18), 'linewidth', 2);
96 plot(t_ode4, x_ode4(:,18), 'linewidth', 2);
97 plot(t_ode5, x_ode5(:,18), 'linewidth', 2);
98 plot(t_ode6, x_ode6(:,18), 'linewidth', 2);
99 plot(t_ode7, x_ode7(:,18), 'linewidth', 2);
100 hold off
101 % legend('x_{21} original', '1 removed state', '2 rem. states', '3 rem.
       states', '4 rem. states', '5 rem. states', '6 rem. states', '7 rem.
       states');
102 xlabel('Timestep');
103 ylabel('[RNA_t: RNA_{eGFP}]');
104
105
106
107 %% Find the fast reached steady states
108 figure(1);
109 subplot(7,3,1);
110 plot(t_ode, x_ode(:,1), 'linewidth', 2)
111 subplot(7,3,2);
112 plot(t_ode, x_ode(:,2), 'linewidth', 2)
113 subplot(7,3,3);
114 plot(t_ode, x_ode(:,3), 'linewidth', 2)
115 subplot(7,3,4);
116 plot(t_ode, x_ode(:,4), 'linewidth', 2)
```

```
117 subplot(7,3,5);
118 plot(t_ode, x_ode(:,5), 'linewidth', 2)
119 subplot(7,3,6);
120 plot(t_ode, x_ode(:,6), 'linewidth', 2)
121 subplot(7,3,7);
122 plot(t_ode, x_ode(:,7), 'linewidth', 2)
123 subplot(7,3,8);
124 plot(t_ode, x_ode(:,8), 'linewidth', 2)
125 subplot(7,3,9);
126 plot(t_ode, x_ode(:,9), 'linewidth', 2)
127 subplot(7,3,10);
128 plot(t_ode, x_ode(:,10), 'linewidth', 2)
129 subplot(7,3,11);
130 plot(t_ode, x_ode(:,11), 'linewidth', 2)
131 subplot(7,3,12);
132 plot(t_ode, x_ode(:,13), 'linewidth', 2)
133 subplot(7,3,13);
134 plot(t_ode, x_ode(:,14), 'linewidth', 2)
135 subplot(7,3,14);
136 plot(t_ode, x_ode(:,15), 'linewidth', 2)
137 subplot(7,3,15);
138 plot(t_ode, x_ode(:,16), 'linewidth', 2)
139 subplot(7,3,16);
140 plot(t_ode, x_ode(:,17), 'linewidth', 2)
141 subplot(7,3,17);
142 plot(t_ode, x_ode(:,18), 'linewidth', 2)
143 subplot(7,3,18);
144 plot(t_ode, x_ode(:,19), 'linewidth', 2)
145 subplot(7,3,19);
146 plot(t_ode, x_ode(:,20), 'linewidth', 2)
147 subplot(7,3,20);
148 plot(t_ode, x_ode(:,21), 'linewidth', 2)
149 subplot(7,3,21);
150 plot(t_ode, x_ode(:,22), 'linewidth', 2)
```

C-3 Alternative Modelling

```
1  function dxdt = ode_fun2(t, x)
2
3  k1 = 1/10;
4  k_2 = 1/10;
5  k3 = 1/10;
6  k4 = 1/10;
7  k5 = 0;
8  k6 = 0;
9  k7 = 1/10;
10 k_8 = 1/10;
11 k9 = 1/10;
12 k10 = 1/10;
13 k11 = 1/10;
14 k_12 = 1/10;
15 k13 = 1/10;
```

```
k14 = 1/10;
k15 = 1/10;
k_16 = 1/10;
k17 = 1/10;
k18 = 1/10;
k19 = 1/10;
kdeg = 1/10;
gamma = 1/100;

N = zeros(22,24);
N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
N(20,19) = -1; N(21,19) = 1;
N(10,20) = -1;
N(8,21) = -1;
N(13,22) = -1;
N(15,23) = -1;
N(18,24) = -1;
N(6,25) = -1; N(10,25) = 1;

vx = zeros(24,1);
vx(1) = k1*x(1)*x(2);
vx(2) = k_2*x(6);
vx(3) = k3*x(6)*x(3);
vx(4) = k4*x(7);
vx(5) = k5*x(6)*x(4);
vx(6) = k6*x(9);
vx(7) = k7*x(11)*x(17);
vx(8) = k_8*x(1);
vx(9) = k9*x(11)*x(5);
vx(10) = k10*x(22);
vx(11) = k11*x(10)*x(8);
vx(12) = k_12 * x(15);
vx(13) = k13*x(15)*x(14);
vx(14) = k14*x(16);
vx(15) = k15*x(8)*x(13);
vx(16) = k_16*x(18);
```

```
69 vx(17) = k17*x(18)*x(14);
70 vx(18) = k18*x(19);
71 vx(19) = k19*x(20);
72 vx(20) = kdeg*x(10);
73 vx(21) = kdeg*x(8);
74 vx(22) = kdeg*x(13);
75 vx(23) = kdeg*x(15);
76 vx(24) = kdeg*x(18);
77 vx(25) = 2*x(6)/(10+10*x(6));
78 dxdt = N*vx;
```

```
1  function dxdt = ode_fun2(t, x)
2
3  k1 = 1/10;
4  k_2 = 1/10;
5  k3 = 0;
6  k4 = 0;
7  k5 = 0;
8  k6 = 0;
9  k7 = 1/10;
10 k_8 = 1/10;
11 k9 = 1/10;
12 k10 = 1/10;
13 k11 = 1/10;
14 k_12 = 1/10;
15 k13 = 1/10;
16 k14 = 1/10;
17 k15 = 1/10;
18 k_16 = 1/10;
19 k17 = 1/10;
20 k18 = 1/10;
21 k19 = 1/10;
22 kdeg = 1/10;
23 gamma = 1/100;
24
25 N = zeros(22,24);
26 N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
27 N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
28 N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
29 N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
30 N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
31 N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
32 N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
33 N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
34 N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
35 N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
36 N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
37 N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
38 N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
39 N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
40 N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
41 N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
42 N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
```

```
N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
N(20,19) = -1; N(21,19) = 1;
N(10,20) = -1;
N(8,21) = -1;
N(13,22) = -1;
N(15,23) = -1;
N(18,24) = -1;
N(6,25) = -1; N(10,25) = 1;
N(6,26) = -1; N(8,26) = 1;

vx = zeros(24,1);
vx(1) = k1*x(1)*x(2);
vx(2) = k_2*x(6);
vx(3) = k3*x(6)*x(3);
vx(4) = k4*x(7);
vx(5) = k5*x(6)*x(4);
vx(6) = k6*x(9);
vx(7) = k7*x(11)*x(17);
vx(8) = k_8*x(1);
vx(9) = k9*x(11)*x(5);
vx(10) = k10*x(22);
vx(11) = k11*x(10)*x(8);
vx(12) = k_12 * x(15);
vx(13) = k13*x(15)*x(14);
vx(14) = k14*x(16);
vx(15) = k15*x(8)*x(13);
vx(16) = k_16*x(18);
vx(17) = k17*x(18)*x(14);
vx(18) = k18*x(19);
vx(19) = k19*x(20);
vx(20) = kdeg*x(10);
vx(21) = kdeg*x(8);
vx(22) = kdeg*x(13);
vx(23) = kdeg*x(15);
vx(24) = kdeg*x(18);
vx(25) = 2*x(6)/(10+10*x(6));
vx(26) = 2*x(6)/(10+10*x(6));
dxdt = N*vx;
```

```
function dxdt = ode_fun2(t, x)

k1 = 1/10;
k_2 = 1/10;
k3 = 0;
k4 = 0;
k5 = 0;
k6 = 0;
k7 = 1/10;
k_8 = 1/10;
k9 = 0;
k10 = 0;
k11 = 1/10;
k_12 = 1/10;
```

```
15 k13 = 1/10;
16 k14 = 1/10;
17 k15 = 1/10;
18 k_16 = 1/10;
19 k17 = 1/10;
20 k18 = 1/10;
21 k19 = 1/10;
22 kdeg = 1/10;
23 gamma = 1/100;
24
25 N = zeros(22,24);
26 N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
27 N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
28 N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
29 N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
30 N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
31 N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
32 N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
33 N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
34 N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
35 N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
36 N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
37 N(8,12) = 1; N(10,12) = 1; N(15,12) = -1;
38 N(15,13) = -1; N(14,13) = -1; N(16,13) = 1;
39 N(16,14) = -1; N(17,14) = 1; N(14,14) = 1;
40 N(8,15) = -1; N(13,15) = -1; N(18,15) = 1;
41 N(8,16) = 1; N(13,16) = 1; N(18,16) = -1;
42 N(18,17) = -1; N(14,17) = -1; N(19,17)= 1;
43 N(19,18) = -1; N(14,18) = 1; N(20,18) = 1; N(18,18)=1;
44 N(20,19) = -1; N(21,19) = 1;
45 N(10,20) = -1;
46 N(8,21) = -1;
47 N(13,22) = -1;
48 N(15,23) = -1;
49 N(18,24) = -1;
50 N(6,25) = -1; N(10,25) = 1;
51 N(6,26) = -1; N(8,26) = 1;
52 N(11,27) = -1; N(12,27) = 1;
53
54 vx = zeros(24,1);
55 vx(1) = k1*x(1)*x(2);
56 vx(2) = k_2*x(6);
57 vx(3) = k3*x(6)*x(3);
58 vx(4) = k4*x(7);
59 vx(5) = k5*x(6)*x(4);
60 vx(6) = k6*x(9);
61 vx(7) = k7*x(11)*x(17);
62 vx(8) = k_8*x(1);
63 vx(9) = k9*x(11)*x(5);
64 vx(10) = k10*x(22);
65 vx(11) = k11*x(10)*x(8);
66 vx(12) = k_12 * x(15);
67 vx(13) = k13*x(15)*x(14);
```

```
vx(14) = k14*x(16);
vx(15) = k15*x(8)*x(13);
vx(16) = k_16*x(18);
vx(17) = k17*x(18)*x(14);
vx(18) = k18*x(19);
vx(19) = k19*x(20);
vx(20) = kdeg*x(10);
vx(21) = kdeg*x(8);
vx(22) = kdeg*x(13);
vx(23) = kdeg*x(15);
vx(24) = kdeg*x(18);
vx(25) = 2*x(6)/(10+10*x(6));
vx(26) = 2*x(6)/(10+10*x(6));
vx(27) = 2*x(11)/(10+10*x(11));
dxdt = N*vx;
```

C-4 Kron reduction order method

```
%% Clean up
clear all; close all; clc;
%%  Ode23 solver based solution of xdot = Nvx

p = 10; % p-percent confidence interval
n = 60; % number of timesteps
tspan = [0 n-1];
x0 = ones(22,1);
[t_ode1, x_ode1] = ode23(@odefun2, tspan, x0) ;
[t_ode,x_ode] = ode23(@Kron_fun, tspan, x0);
% [t_red_ode, x_red_ode] = ode23(@odefun2, tspan, x0);
[t_red_ode1, x_red_ode1] = ode23(@Kron_red_fun1, tspan, x0);
[t_red_ode2, x_red_ode2] = ode23(@Kron_red_fun2, tspan, x0);
[t_red_ode3, x_red_ode3] = ode23(@Kron_red_fun3, tspan, x0);
[t_red_ode4, x_red_ode4] = ode23(@Kron_red_fun4, tspan, x0);
[t_red_ode5, x_red_ode5] = ode23(@Kron_red_fun5, tspan, x0);
[t_red_ode6, x_red_ode6] = ode23(@Kron_red_fun6, tspan, x0);


Alinearization = cell2mat(struct2cell(load('Alinearization.mat')));

[t_linearization, x_linearization] = ode23(@Linearization_func, tspan, x0
    ) ;
ss_original = round (x_ode(end, 1:22), 2)';  % Compute steady - state
ss_1complex = round(x_red_ode1(end, 1:22),2)';
ss_2complex = round(x_red_ode2(end, 1:22),2)';
ss_3complex = round(x_red_ode3(end, 1:22),2)';
ss_4complex = round(x_red_ode4(end, 1:22),2)';
ss_5complex = round(x_red_ode5(end, 1:22),2)';
ss_6complex = round(x_red_ode6(end, 1:22),2)';

Errors = zeros(22,6);
Errors(1:22, 1) = abs(ss_1complex - ss_original);
Errors(1:22, 2) = abs(ss_2complex - ss_original);
```

```
34 Errors(1:22, 3) = abs(ss_3complex - ss_original);
35 Errors(1:22, 4) = abs(ss_4complex - ss_original);
36 Errors(1:22, 5) = abs(ss_5complex - ss_original);
37 Errors(1:22, 6) = abs(ss_6complex - ss_original);
38 Errors = round(Errors, 2);
39
40 %%
41
42
43 %% Plotting (for matlab ODE solvers)
44
45 figure(1);
46
47 subplot(2,3,1);
48 plot(t_ode1, x_ode1(:,8), 'linewidth', 2 ) % Plot of state x8 for n
       timesteps
49 hold on
50 plot(t_ode,x_ode(:,8),'o', 'linewidth', 2 ) % Plot of state x8 for n
       timesteps (red. order model)
51 hold off
52 xlabel('Timestep');
53 ylabel('[RNA_t ]    (x_8)');
54 legend('x8 original', 'x8 reduced order system');
55
56 subplot(2,3,2);
57 plot(t_ode1,x_ode1(:,10), 'linewidth', 2 ) % Plot of state x10 for n
       timesteps
58 hold on
59 plot(t_ode, x_ode(:,10), 'o' , 'linewidth', 2 ) % Plot of state x10 for
       n timesteps (red. order model)
60 hold off
61 xlabel('Timestep');
62 ylabel('[RNA_{S28}]    (x_{10})');
63 legend('x10 original', 'x10 reduced order system');
64
65 subplot(2,3,3);
66 plot(t_ode1,x_ode1(:,15), 'linewidth', 2 ) % Plot of state x15 for n
       timesteps
67 hold on
68 plot(t_ode, x_ode(:,15) ,'o', 'linewidth', 2 ) % Plot of state x15 for n
        timesteps (red. order model)
69 hold off
70 xlabel('Timestep');
71 ylabel('[RNA_t:RNA_{S28}]    (x_{14})');
72 legend('x14 original', 'x14 reduced order system');
73
74 subplot(2,3,4:5);
75 plot(t_ode1,x_ode1(:,21), 'linewidth', 2 ) % Plot of state x21 for n
       timesteps
76 hold on
77 plot(t_ode, x_ode(:,21), 'o', 'linewidth', 2 ) % Plot of state x21 for n
        timesteps (red. order model)
78 hold off
```

```
79 xlabel('Timestep');
80 ylabel('[e_{GFP}]    (x_{20})');
81 legend('x20 original', 'x20 reduced order system');
82
83 subplot(2,3,6);
84 plot(t_ode1,x_ode1(:,18), 'linewidth', 2  ) % Plot of state x18 for n
       timesteps
85 hold on
86 plot(t_ode, x_ode(:,18), 'o','linewidth', 2  ) % Plot of state x18 for n
       timesteps (red. order model)
87 hold off
88 xlabel('Timestep');
89 ylabel('[RNA_t : RNA_{eGFP}] (x_{17})');
90 legend('x17 original', 'x17 reduced order system');
91 sgtitle('Comparison between original law of mass action and linearization
       ');
92
93 %%
94 close all;
95 figure(2);
96
97 subplot(2,3,1);
98 plot(t_red_ode1, x_red_ode1(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
99 hold on
100 % plot(t_red_ode, x_red_ode(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
101 plot(t_red_ode2, x_red_ode2(:,8), 'linewidth', 3  ) % Plot of state x8
       for n timesteps (red. order model)
102 plot(t_red_ode3, x_red_ode3(:,8), 'linewidth', 2) % Plot of state x8 for
       n timesteps (red. order model)
103 plot(t_red_ode4, x_red_ode4(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
104 plot(t_red_ode5, x_red_ode5(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
105 plot(t_red_ode6, x_red_ode6(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
106 % plot(t_red_ode7, x_red_ode7(:,8), 'linewidth', 2  ) % Plot of state x8
       for n timesteps (red. order model)
107 p1 = plot(t_ode1,x_ode1(:,8), 'linewidth', 2  ) % Plot of state x8 for n
       timesteps
108 hold off
109 xlabel('Timestep');
110 ylabel('[RNA_t ]    (x_8)');
111 % legend(  'Removed complex: C_{28}, X_{18}', 'Extra removed complex: C_
       {27}, X_{16} ', 'Extra removed complex: C_{26}, X_{15}' , 'Extra
       removed complex: C_{25}, X_{22}' , 'Extra removed complex: C_{24}, X_
       {9}', 'Extra removed complex: C_{23}, X_{7}', 'x8 original');
112 legend(p1, 'x_8 original model') ;
113 ylim([0  1.2]);
114
115 subplot(2,3,2);
```

```
116 plot(t_red_ode1, x_red_ode1(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
117 hold on
118 plot(t_red_ode2, x_red_ode2(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
119 plot(t_red_ode3, x_red_ode3(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
120 plot(t_red_ode4, x_red_ode4(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
121 plot(t_red_ode5, x_red_ode5(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
122 plot(t_red_ode6, x_red_ode6(:,10), 'linewidth', 2 ) % Plot of state x10
    for n timesteps (red. order model)
123 % plot(t_red_ode7, x_red_ode7(:,10), 'linewidth', 2 ) % Plot of state
    x10 for n timesteps (red. order model)
124 p2 = plot(t_ode,x_ode(:,10), 'linewidth', 2 ) % Plot of state x10 for n
    timesteps
125 hold off
126 xlabel('Timestep');
127 ylabel('[RNA_{S28}]    (x_{10})');
128 ylim([0 1.2]);
129 % legend( '1 removed complex', '2 removed complexes', '3 removed
    complexes' , '4 removed complexes' , '5 removed complexes', '6 removed
     complexes', 'x10 original');
130 % legend(p2, 'x_{10} original model' );
131 legend( 'Removed complex: C_{28}, X_{17}', 'Extra removed complex: C_
    {27}, X_{15} ', 'Extra removed complex: C_{26}, X_{14}' , 'Extra
    removed complex: C_{25}, X_{21}' , 'Extra removed complex: C_{24}, X_
    {9}', 'Extra removed complex: C_{23}, X_{7}', 'x10 original');
132
133 subplot(2,3,3);
134 plot(t_red_ode1, x_red_ode1(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
135 hold on
136 plot(t_red_ode2, x_red_ode2(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
137 plot(t_red_ode3, x_red_ode3(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
138 plot(t_red_ode4, x_red_ode4(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
139 plot(t_red_ode5, x_red_ode5(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
140 plot(t_red_ode6, x_red_ode6(:,15), 'linewidth', 2 ) % Plot of state x15
    for n timesteps (red. order model)
141 % plot(t_red_ode7, x_red_ode7(:,15), 'linewidth', 2 ) % Plot of state
    x15 for n timesteps (red. order model)
142 p3 = plot(t_ode,x_ode(:,15), 'linewidth', 2 ) % Plot of state x15 for n
    timesteps
143 hold off
144 xlabel('Timestep');
145 ylabel('[RNA_t:RNA_{S28}]    (x_{14})');
146 legend( p3, 'x14 original model');
147
```

```
148
149 subplot(2,3, 4);
150 plot(t_red_ode1, x_red_ode1(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
151 hold on
152 plot(t_red_ode2, x_red_ode2(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
153 plot(t_red_ode3, x_red_ode3(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
154 plot(t_red_ode4, x_red_ode4(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
155 plot(t_red_ode5, x_red_ode5(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
156 plot(t_red_ode6, x_red_ode6(:,21), 'linewidth', 2 ) % Plot of state x21
    for n timesteps (red. order model)
157 % plot(t_red_ode7, x_red_ode7(:,21), 'linewidth', 2 ) % Plot of state
    x21 for n timesteps (red. order model)
158 p6 = plot(t_ode,x_ode(:,21), 'linewidth', 2 ) % Plot of state x21 for n
    timesteps
159 hold off
160 xlabel('Timestep');
161 ylabel('[e_{GFP}]    (x_{20})');
162 % legend( '1 removed complex', '2 removed complexes', '3 removed
    complexes' , '4 removed complexes' , '5 removed complexes', '6 removed
     complexes', 'x21 original');
163 legend( 'Removed complex: C_{28}, X_{17}', 'Extra removed complex: C_
    {27}, X_{15} ', 'Extra removed complex: C_{26}, X_{14}' , 'Extra
    removed complex: C_{25}, X_{21}' , 'Extra removed complex: C_{24}, X_
    {9}', 'Extra removed complex: C_{23}, X_{7}');
164 a=axes('position',get(gca,'position'),'visible','off');
165 legend(a,[ p6 ],'original model x_{21}');
166
167  subplot(2,3,5);
168 plot(t_red_ode1, x_red_ode1(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
169 hold on
170 plot(t_red_ode2, x_red_ode2(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
171 plot(t_red_ode3, x_red_ode3(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
172 plot(t_red_ode4, x_red_ode4(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
173 plot(t_red_ode5, x_red_ode5(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
174 plot(t_red_ode6, x_red_ode6(:,18), 'linewidth', 2 ) % Plot of state x18
    for n timesteps (red. order model)
175 % plot(t_red_ode7, x_red_ode7(:,18), 'linewidth', 2 ) % Plot of state
    x18 for n timesteps (red. order model)
176 p5 = plot(t_ode,x_ode(:,18), 'linewidth', 2 ) % Plot of state x18 for n
    timesteps
177 hold off
178 xlabel('Timestep');
179 ylabel('[RNA_t : RNA_{eGFP}] (x_{17})');
```

```
% legend(  '1 removed complex', '2 removed complexes', '3 removed complexes' , '4 removed complexes' , '5 removed complexes', '6 removed complexes', 'x18 original');
legend(p5, 'original model x_{17}' );
sgtitle('Plot of reduced order model compared with original model (Matlab ODE solver)');


%% Using linear interpolation functions

xq = [0:n-1]';

vqlin1 = interp1(t_linearization, x_linearization(:,1), xq);
vqlin2 = interp1(t_linearization, x_linearization(:,2), xq);
vqlin3 = interp1(t_linearization, x_linearization(:,3), xq);
vqlin4 = interp1(t_linearization, x_linearization(:,4), xq);
vqlin5 = interp1(t_linearization, x_linearization(:,5), xq);
vqlin6 = interp1(t_linearization, x_linearization(:,6), xq);
vqlin7 = interp1(t_linearization, x_linearization(:,7), xq);
vqlin8 = interp1(t_linearization, x_linearization(:,8), xq);
vqlin9 = interp1(t_linearization, x_linearization(:,9), xq);
vqlin10 = interp1(t_linearization, x_linearization(:,10), xq);
vqlin11 = interp1(t_linearization, x_linearization(:,11), xq);
vqlin12 = interp1(t_linearization, x_linearization(:,12), xq);
vqlin13 = interp1(t_linearization, x_linearization(:,13), xq);
vqlin14 = interp1(t_linearization, x_linearization(:,14), xq);
vqlin15 = interp1(t_linearization, x_linearization(:,15), xq);
vqlin16 = interp1(t_linearization, x_linearization(:,16), xq);
vqlin17 = interp1(t_linearization, x_linearization(:,17), xq);
vqlin18 = interp1(t_linearization, x_linearization(:,18), xq);
vqlin19 = interp1(t_linearization, x_linearization(:,19), xq);
vqlin20 = interp1(t_linearization, x_linearization(:,20), xq);
vqlin21 = interp1(t_linearization, x_linearization(:,21), xq);
vqlin22 = interp1(t_linearization, x_linearization(:,22), xq);

vq01 = interp1(t_ode, x_ode(:,1) ,xq);
vq02 = interp1(t_ode, x_ode(:,2) ,xq);
vq03 = interp1(t_ode, x_ode(:,3) ,xq);
vq04 = interp1(t_ode, x_ode(:,4) ,xq);
vq05 = interp1(t_ode, x_ode(:,5) ,xq);
vq06 = interp1(t_ode, x_ode(:,6) ,xq);
vq07 = interp1(t_ode, x_ode(:,7) ,xq);
vq08 = interp1(t_ode, x_ode(:,8) ,xq);
vq09 = interp1(t_ode, x_ode(:,9) ,xq);
vq010 = interp1(t_ode, x_ode(:,10) ,xq);
vq011 = interp1(t_ode, x_ode(:,11) ,xq);
vq012 = interp1(t_ode, x_ode(:,12) ,xq);
vq013 = interp1(t_ode, x_ode(:,13) ,xq);
vq014 = interp1(t_ode, x_ode(:,14) ,xq);
vq015 = interp1(t_ode, x_ode(:,15) ,xq);
vq016 = interp1(t_ode, x_ode(:,16) ,xq);
vq017 = interp1(t_ode, x_ode(:,17) ,xq);
vq018 = interp1(t_ode, x_ode(:,18) ,xq);
```

```
230 vq019 = interp1(t_ode, x_ode(:,19) ,xq);
231 vq020 = interp1(t_ode, x_ode(:,20) ,xq);
232 vq021 = interp1(t_ode, x_ode(:,21) ,xq);
233 vq022 = interp1(t_ode, x_ode(:,22) ,xq);
234
235 % state 8
236
237 vq08 = interp1(t_ode, x_ode(:,8) ,xq);
238 vq18 = interp1(t_red_ode1, x_red_ode1(:,8), xq);
239 vq28 = interp1(t_red_ode2, x_red_ode2(:,8), xq);
240 vq38 = interp1(t_red_ode3, x_red_ode3(:,8), xq);
241 vq48 = interp1(t_red_ode4, x_red_ode4(:,8), xq);
242 vq58 = interp1(t_red_ode5, x_red_ode5(:,8), xq);
243 vq68 = interp1(t_red_ode6, x_red_ode6(:,8), xq);
244
245 % state 10
246
247 vq010 = interp1(t_ode, x_ode(:,10) ,xq);
248 vq110 = interp1(t_red_ode1, x_red_ode1(:,10), xq);
249 vq210 = interp1(t_red_ode2, x_red_ode2(:,10), xq);
250 vq310 = interp1(t_red_ode3, x_red_ode3(:,10), xq);
251 vq410 = interp1(t_red_ode4, x_red_ode4(:,10), xq);
252 vq510 = interp1(t_red_ode5, x_red_ode5(:,10), xq);
253 vq610 = interp1(t_red_ode6, x_red_ode6(:,10), xq);
254 % vq710 = interp1(t_red_ode7, x_red_ode7(:,10), xq);
255
256 % state 15
257
258 vq015 = interp1(t_ode, x_ode(:,15) ,xq);
259 vq115 = interp1(t_red_ode1, x_red_ode1(:,15), xq);
260 vq215 = interp1(t_red_ode2, x_red_ode2(:,15), xq);
261 vq315 = interp1(t_red_ode3, x_red_ode3(:,15), xq);
262 vq415 = interp1(t_red_ode4, x_red_ode4(:,15), xq);
263 vq515 = interp1(t_red_ode5, x_red_ode5(:,15), xq);
264 vq615 = interp1(t_red_ode6, x_red_ode6(:,15), xq);
265 % vq715 = interp1(t_red_ode7, x_red_ode7(:,15), xq);
266
267 % state 21
268
269 vq021 = interp1(t_ode, x_ode(:,21) ,xq);
270 vq121 = interp1(t_red_ode1, x_red_ode1(:,21), xq);
271 vq221 = interp1(t_red_ode2, x_red_ode2(:,21), xq);
272 vq321 = interp1(t_red_ode3, x_red_ode3(:,21), xq);
273 vq421 = interp1(t_red_ode4, x_red_ode4(:,21), xq);
274 vq521 = interp1(t_red_ode5, x_red_ode5(:,21), xq);
275 vq621 = interp1(t_red_ode6, x_red_ode6(:,21), xq);
276 % vq721 = interp1(t_red_ode7, x_red_ode7(:,21), xq);
277
278 % state 18
279
280 vq018 = interp1(t_ode, x_ode(:,18) ,xq);
281 vq118 = interp1(t_red_ode1, x_red_ode1(:,18), xq);
282 vq218 = interp1(t_red_ode2, x_red_ode2(:,18), xq);
```

```
vq318 = interp1(t_red_ode3, x_red_ode3(:,18), xq);
vq418 = interp1(t_red_ode4, x_red_ode4(:,18), xq);
vq518 = interp1(t_red_ode5, x_red_ode5(:,18), xq);
vq618 = interp1(t_red_ode6, x_red_ode6(:,18), xq);
% vq718 = interp1(t_red_ode7, x_red_ode7(:,18), xq);

%% Compute RMS values between nonlinear and linear system

% errors in all states for nonlienar vs linear system

RMSlin1 = sqrt( (1/n)*(vq01-vqlin1)'*(vq01 -vqlin1))/ mean(vqlin1);
RMSlin2 = sqrt( (1/n)*(vq02-vqlin2)'*(vq02 -vqlin2))/ mean(vqlin2);
RMSlin3 = sqrt( (1/n)*(vq03-vqlin3)'*(vq03 -vqlin3))/ mean(vqlin3);
RMSlin4 = sqrt( (1/n)*(vq04-vqlin4)'*(vq04 -vqlin4))/ mean(vqlin4);
RMSlin5 = sqrt( (1/n)*(vq05-vqlin5)'*(vq05 -vqlin5))/ mean(vqlin5);
RMSlin6 = sqrt( (1/n)*(vq06-vqlin6)'*(vq06 -vqlin6))/ mean(vqlin6);
RMSlin7 = sqrt( (1/n)*(vq07-vqlin7)'*(vq07 -vqlin7))/ mean(vqlin7);
RMSlin8 = sqrt( (1/n)*(vq08-vqlin8)'*(vq08-vqlin8))/ mean(vqlin8);
RMSlin9 = sqrt( (1/n)*(vq09-vqlin9)'*(vq09-vqlin9))/ mean(vqlin9);
RMSlin10 = sqrt( (1/n)*(vq010-vqlin10)'*(vq010-vqlin10))/ mean(vqlin10);
RMSlin11 = sqrt( (1/n)*(vq011-vqlin11)'*(vq011 -vqlin11))/ mean(vqlin11);
RMSlin12 = sqrt( (1/n)*(vq012-vqlin12)'*(vq012 -vqlin12))/ mean(vqlin12);
RMSlin13 = sqrt( (1/n)*(vq013-vqlin13)'*(vq013 -vqlin13))/ mean(vqlin13);
RMSlin14 = sqrt( (1/n)*(vq014-vqlin14)'*(vq014 -vqlin14))/ mean(vqlin14);
RMSlin15 = sqrt( (1/n)*(vq015-vqlin15)'*(vq015 -vqlin15))/ mean(vqlin15);
RMSlin16 = sqrt( (1/n)*(vq016-vqlin16)'*(vq016 -vqlin16))/ mean(vqlin16);
RMSlin17 = sqrt( (1/n)*(vq017-vqlin17)'*(vq017 -vqlin17))/ mean(vqlin17);
RMSlin18 = sqrt( (1/n)*(vq018-vqlin18)'*(vq018 -vqlin18))/ mean(vqlin18);
RMSlin19 = sqrt( (1/n)*(vq019-vqlin19)'*(vq019 -vqlin19))/ mean(vqlin19);
RMSlin20 = sqrt( (1/n)*(vq020-vqlin20)'*(vq020 -vqlin20))/ mean(vqlin20);
RMSlin21 = sqrt( (1/n)*(vq021-vqlin21)'*(vq021 -vqlin21))/ mean(vqlin21);
RMSlin22 = sqrt( (1/n)*(vq022-vqlin22)'*(vq022 -vqlin22))/ mean(vqlin22);

RMS_nonlinear = [RMSlin1; RMSlin2; RMSlin3; RMSlin4; RMSlin5; RMSlin6;
    RMSlin7; RMSlin8; RMSlin9; RMSlin10; RMSlin11; RMSlin12; RMSlin13;
    RMSlin14; RMSlin15; RMSlin16; RMSlin17;RMSlin18;RMSlin19; RMSlin20;
    RMSlin21; RMSlin22];

x_bar = categorical( {'RNAP','S70','DNA_t', 'DNA_{S28}','DNA_{eGFP}', '
    RNAP:S70', 'RNAP:S70:DNA_t','RNA_t', 'RNAP:S70:DNA_{S28}', 'RNA_{S28}'
    , 'RNAP:S28', 'RNAP:S28:DNA_t', 'RNA_{eGFP}', 'Ribo', 'RNA_t:RNA_{S28}
    ', 'RNA_t:RNA_{S28}:Ribo', 'S28', 'RNA_t:RNA_{eGFP}', 'RNA_t:RNA_{eGFP
    }:Ribo', 'eGFPdark', 'eGFP', 'RNAP:S28:DNA_{eGFP}' }) ;

x_bar = reordercats(x_bar,{'RNAP','S70','DNA_t', 'DNA_{S28}','DNA_{eGFP}'
    , 'RNAP:S70', 'RNAP:S70:DNA_t','RNA_t', 'RNAP:S70:DNA_{S28}', 'RNA_{
    S28}', 'RNAP:S28', 'RNAP:S28:DNA_t', 'RNA_{eGFP}', 'Ribo', 'RNA_t:RNA_
    {S28}', 'RNA_t:RNA_{S28}:Ribo', 'S28', 'RNA_t:RNA_{eGFP}', 'RNA_t:RNA_
    {eGFP}:Ribo', 'eGFPdark', 'eGFP', 'RNAP:S28:DNA_{eGFP}' });

figure(3);
barh(x_bar, RMS_nonlinear);
```

```
324 title(' RMS values between non-reduced nonlinear system and linearized
        system');
325 ylabel('Species $X_i$', 'interpreter', 'latex');
326 xlabel('RMS value for given species between linearized and nonlinear
        system');
327
328 %% Compute RMS values between non-reduced and reduced order system
329 % errors in state 8
330
331 RMS18 = sqrt( (1/n) * (vq08 - vq18)'*(vq08 - vq18)) / mean(x_ode(:, 8) )
        ;
332 RMS28 = sqrt( (1/n) * (vq08 - vq28)'*(vq08 - vq28)) / mean(x_ode(:, 8) )
        ;
333 RMS38 = sqrt( (1/n) * (vq08 - vq38)'*(vq08 - vq38)) / mean(x_ode(:, 8) )
        ;
334 RMS48 = sqrt( (1/n) * (vq08 - vq48)'*(vq08 - vq48)) / mean(x_ode(:, 8) )
        ;
335 RMS58 = sqrt( (1/n) * (vq08 - vq58)'*(vq08 - vq58)) / mean(x_ode(:, 8) )
        ;
336 RMS68 = sqrt( (1/n) * (vq08 - vq68)'*(vq08 - vq68)) / mean(x_ode(:, 8) )
        ;
337 % RMS78 = sqrt( (1/n) * (vq08 - vq78)'*(vq08 - vq78)) / mean(x_red_ode7
        (:, 8) ) ;
338
339 % errors in state 10
340
341 RMS110 = sqrt( (1/n) * (vq010 - vq110)'*(vq010 - vq110)) / mean(x_ode(:,
        10) ) ;
342 RMS210 = sqrt( (1/n) * (vq010 - vq210)'*(vq010 - vq210)) / mean(x_ode(:,
        10) ) ;
343 RMS310 = sqrt( (1/n) * (vq010 - vq310)'*(vq010 - vq310)) / mean(x_ode(:,
        10) ) ;
344 RMS410 = sqrt( (1/n) * (vq010 - vq410)'*(vq010 - vq410)) / mean(x_ode(:,
        10) ) ;
345 RMS510 = sqrt( (1/n) * (vq010 - vq510)'*(vq010 - vq510)) / mean(x_ode(:,
        10) ) ;
346 RMS610 = sqrt( (1/n) * (vq010 - vq610)'*(vq010 - vq610)) / mean(x_ode(:,
        10) ) ;
347 % RMS710 = sqrt( (1/n) * (vq010 - vq710)'*(vq010 - vq710)) / mean(
        x_red_ode7(:, 10) ) ;
348
349 % errors in state 15
350
351 RMS115 = sqrt( (1/n) * (vq015 - vq115)'*(vq015 - vq115)) / mean(x_ode(:,
        15) ) ;
352 RMS215 = sqrt( (1/n) * (vq015 - vq215)'*(vq015 - vq215)) / mean(x_ode(:,
        15) ) ;
353 RMS315 = sqrt( (1/n) * (vq015 - vq315)'*(vq015 - vq315)) / mean(x_ode(:,
        15) ) ;
354 RMS415 = sqrt( (1/n) * (vq015 - vq415)'*(vq015 - vq415)) / mean(x_ode(:,
        15) ) ;
355 RMS515 = sqrt( (1/n) * (vq015 - vq515)'*(vq015 - vq515)) / mean(x_ode(:,
        15) ) ;
```

```
356 RMS615 = sqrt( (1/n) * (vq015 - vq615)'*(vq015 - vq615)) / mean(x_ode(:,
        15) ) ;
357 % RMS715 = sqrt( (1/n) * (vq015 - vq715)'*(vq015 - vq715)) / mean(
        x_red_ode7(:, 15) ) ;
358
359 % errors in state 21
360
361 RMS121 = sqrt( (1/n) * (vq021 - vq121)'*(vq021 - vq121)) / mean(x_ode(:,
        21) ) ;
362 RMS221 = sqrt( (1/n) * (vq021 - vq221)'*(vq021 - vq221)) / mean(x_ode(:,
        21) ) ;
363 RMS321 = sqrt( (1/n) * (vq021 - vq321)'*(vq021 - vq321)) / mean(x_ode(:,
        21) ) ;
364 RMS421 = sqrt( (1/n) * (vq021 - vq421)'*(vq021 - vq421)) / mean(x_ode(:,
        21) ) ;
365 RMS521 = sqrt( (1/n) * (vq021 - vq521)'*(vq021 - vq521)) / mean(x_ode(:,
        21) ) ;
366 RMS621 = sqrt( (1/n) * (vq021 - vq621)'*(vq021 - vq621)) / mean(x_ode(:,
        21) ) ;
367 % RMS721 = sqrt( (1/n) * (vq021 - vq721)'*(vq021 - vq721)) / mean(
        x_red_ode7(:, 21) ) ;
368
369 % errors in state 18
370
371 RMS118 = sqrt( (1/n) * (vq018 - vq118)'*(vq018 - vq118)) / mean(x_ode(:,
        18) ) ;
372 RMS218 = sqrt( (1/n) * (vq018 - vq218)'*(vq018 - vq218)) / mean(x_ode(:,
        18) ) ;
373 RMS318 = sqrt( (1/n) * (vq018 - vq318)'*(vq018 - vq318)) / mean(x_ode(:,
        18) ) ;
374 RMS418 = sqrt( (1/n) * (vq018 - vq418)'*(vq018 - vq418)) / mean(x_ode(:,
        18) ) ;
375 RMS518 = sqrt( (1/n) * (vq018 - vq518)'*(vq018 - vq518)) / mean(x_ode(:,
        18) ) ;
376 RMS618 = sqrt( (1/n) * (vq018 - vq618)'*(vq018 - vq618)) / mean(x_ode(:,
        18) ) ;
377 % RMS718 = sqrt( (1/n) * (vq018 - vq718)'*(vq018 - vq718)) / mean(
        x_red_ode7(:, 18) ) ;
378
379 %% Collect data
380
381 RMS1 = zeros(3,6) ;
382
383 RMS1(1, :) =  [RMS18 RMS28 RMS38 RMS48 RMS58 RMS68];          % state 8
384 RMS1(2, :) =  [RMS110 RMS210 RMS310 RMS410 RMS510 RMS610] ; % state 10
385 RMS1(3, :) =  [RMS115 RMS215 RMS315 RMS415 RMS515 RMS615] ; % state 15
386
387 RMS2 = zeros(2,6);
388 RMS2(1, :) =  [RMS121 RMS221 RMS321 RMS421 RMS521 RMS621] ; % state 21
389 RMS2(2, :) =  [RMS118 RMS218 RMS318 RMS418 RMS518 RMS618] ; % state 18
390
391
```

```
392 x_bar1 = categorical( {'x_8 [RNA_t]', 'x_{10} [RNA_{S28}]', 'x_{14} [
        RNA_t:RNA_{S28}]'});
393 x_bar2 = categorical( {'x_{20} [eGFP]', 'x_{17} [RNA_t:RNA_{eGFP}]'} );
394 x_bar1 = reordercats( x_bar1, {'x_8 [RNA_t]', 'x_{10} [RNA_{S28}]', 'x_
        {14} [RNA_t:RNA_{S28}]'});
395 x_bar2 = reordercats( x_bar2, {'x_{20} [eGFP]', 'x_{17} [RNA_t:RNA_{eGFP
        }]'} );
396
397 % x_bar = categorical( { '1 - x_{8, red,i }/x_8 [RNA_t] ', ...
398 %     '1 - x_{10, red,i}/x_{10} [RNA_{S28}] ', ...
399 %     '1 - x_{15, red, i}/x_{15} [RNA_t : RNA_{S28}] ' , ...
400 %     '1 - x_{21, red, i}/x_{21} [e_{GFP}] ', ...
401 %     '1 - x_{18, red, i}/x_{18}  [RNA_t: RNA_{eGFP}] '  }) ;
402 %
403 %  x_bar = reordercats(x_bar,{ '1 - x_{8, red,i }/x_8 [RNA_t] ', ...
404 %     '1 - x_{10, red,i}/x_{10} [RNA_{S28}] ', ...
405 %     '1 - x_{15, red, i}/x_{15} [RNA_t : RNA_{S28}] ' , ...
406 %     '1 - x_{21, red, i}/x_{21} [e_{GFP}] ', ...
407 %     '1 - x_{18, red, i}/x_{18}  [RNA_t: RNA_{eGFP}] '  });
408
409 figure(4);
410 b1 = bar(x_bar1, RMS1);
411 ylabel('Relative errors for system order reduction by removing i states')
        ; %  , 'Interpreter','latex'
412 set(b1, {'DisplayName'},...
413     {'i = 1', ...
414      'i = 2', ...
415      'i = 3' , ...
416      'i = 4', ...
417      'i = 5', ...
418      'i = 6'}');
419 set(b1(1,1), 'facecolor', [0.71 0.87 0.36] );
420 set(b1(1,2), 'facecolor', [0.84 0.85 0.10] );
421 set(b1(1,3), 'facecolor', [0.93 0.69 0.13] );
422 set(b1(1,4), 'facecolor', [0.87 0.56 0.29] );
423 set(b1(1,5), 'facecolor', [0.84 0.36 0.26] );
424 set(b1(1,6), 'facecolor', [0.95 0.36 0.26] );
425 legend();
426
427 figure(5);
428 b2 = bar(x_bar2, RMS2);
429 ylabel('Relative errors for system order reduction by removing i states')
        ; %  , 'Interpreter','latex'
430 set(b2, {'DisplayName'},...
431     {'i = 1', ...
432      'i = 2', ...
433      'i = 3' , ...
434      'i = 4', ...
435      'i = 5', ...
436      'i = 6'}');
437
438 set(b2(1,1), 'facecolor', [0.71 0.87 0.36] );
439 set(b2(1,2), 'facecolor', [0.84 0.85 0.10] );
```

```
440 set(b2(1,3), 'facecolor', [0.93 0.69 0.13] );
441 set(b2(1,4), 'facecolor', [0.87 0.56 0.29] );
442 set(b2(1,5), 'facecolor', [0.84 0.36 0.26] );
443 set(b2(1,6), 'facecolor', [0.95 0.36 0.26] );
444
445
446
447  legend();
448  % legend('position', 'northwest');
```

```
1  function dxdt = Kron_fun(t, x)
2  k1 = 1/10;
3  k_2 = 1/10;
4  k3 = 1/10;
5  k4 = 1/10;
6  k5 = 1/10;
7  k6 = 1/10;
8  k7 = 1/10;
9  k_8 = 1/10;
10 k9 = 1/10;
11 k10 = 1/10;
12 k11 = 1/10;
13 k_12 = 1/10;
14 k13 = 1/10;
15 k14 = 1/10;
16 k15 = 1/10;
17 k_16 = 1/10;
18 k17 = 1/10;
19 k18 = 1/10;
20 k19 = 1/10;
21 kdeg = 1/10;
22 gamma = 1/100;
23
24 A = zeros(28,28) ;
25
26 A(2,1) = k1;
27 A(1,2) = k_2;
28 A(23,3) = k3;
29 A(5,23) = k4;
30 A(24,6) = k5;
31 A(8,24) =k6;
32 A(10,9) = k7;
33 A(9,10) = k_8;
34 A(25,11) = k9;
35 A(13,25) = k10;
36 A(26,14) = k11;
37 A(14,26) = k_12;
38 A(27,16) = k13;
39 A(18,27) = k14;
40 A(28,19) = k15;
41 A(19,28) = k_16;
42 A(22,21) = k17;
43 A(4,22) = k18;
```

```
44 A(12,7) = k19;
45
46 Z = zeros(22,28);
47
48 Z(1,1)=1; Z(2,1)=1;
49 Z(6,2)=1;
50 Z(6,3)=1; Z(3,3)=1;
51 Z(14,4)=1; Z(18,4) = 1; Z(20,4) = 1;
52 Z(8,5)=1;Z(3,5)=1;Z(6,5)=1;
53 Z(6,6)=1;Z(4,6)=1;
54 Z(20,7)=1;
55 Z(10,8)=1;Z(4,8)=1;Z(6,8)=1;
56 Z(11,9)=1;Z(17,9)=1;
57 Z(1,10)=1;
58 Z(11,11)=1;Z(5,11)=1;
59 Z(21,12)=1;
60 Z(5,13)=1;Z(13,13)=1;
61 Z(10,14)=1;Z(8,14)=1;
62 Z(10,15)=1;
63 Z(15,16)=1;Z(14,16)=1;
64 Z(8,17)=1;
65 Z(17,18)=1; Z(14,18)=1;
66 Z(8,19)=1; Z(13,19)=1;
67 Z(13,20) = 1;
68 Z(18,21)=1;Z(14,21)=1;
69 Z(19,22)=1;
70 Z(7,23)=1;
71 Z(9,24)=1;
72 Z(22,25)=1;
73 Z(15,26)=1;
74 Z(16,27)=1;
75 Z(18,28)=1;
76
77
78 D = diag( [k1, k_2, k3, 0,0,k5,k19,0,k7,k_8,k9,0,0,k11,kdeg,k13,kdeg,0,
       k15,kdeg,k17,k18,k4,k6,k10,kdeg+k_12,k14,kdeg+k_16]) ;
79 L = D - A ;
80
81 dxdt = -Z*L*exp(Z'*log(x)) ; % + Z*ones(28,1) ;
82
83 % Zs = Z(1:end, [1:19 21:28]);
84 % K = [K zeros(23,4)];
85 % vxred = D * exp(Z' *log(xred) ) ;
86
87 % Construct incidence matrix
88
89 % B = zeros(28,23);
90 %
91 % B(1,1) = -1;
92 % B(2,1) = 1;
93 % B(1,2) = 1;
94 % B(2,2) = -1;
95 % B(3,3) = -1;
```

```
% B(23,3) = 1;
% B(5,4) = 1;
% B(6,5) = -1;
% B(4,5) = 1;
% B(4,6) = -1;
% B(8,6) = 1;
% B(9,7) = -1;
% B(10,7) = 1;
% B(10,8) = -1;
% B(9,8) = 1;
% B(11,9) = -1;
% B(25,9) = 1;
% B(25,10) = -1;
% B(13,10) = 1;
% B(14,11) = -1;
% B(26,11) = 1;
% B(16,12) = -1;
% B(27,12) = 1;
% B(27,13) = -1;
% B(18,13) = 1;
% B(19,14) = -1;
% B(28,14) = 1;
% B(28,15) = -1;
% B(19,15) = 1;
% B(21,16) = -1;
% B(22,16) = 1;
% B(22,17) = -1;
% B(4,17) = 1;
% B(7,18) = -1;
% B(12,18) = 1;
% B(26,19) = -1;
% B(27,20) = -1;
% B(28,21) = -1;
% B(15,22) = -1;
% B(20,23) = -1;
%
```

```
function dxdt = Kron_red_fun(t, xred)

k1 = 1/10;
k_2 = 1/10;
k3 = 1/10;
k4 = 1/10;
k5 = 1/10;
k6 = 1/10;
k7 = 1/10;
k_8 = 1/10;
k9 = 1/10;
k10 = 1/10;
k11 = 1/10;
k_12 = 1/10;
k13 = 1/10;
k14 = 1/10;
```

```
17 k15 = 1/10;
18 k_16 = 1/10;
19 k17 = 1/10;
20 k18 = 1/10;
21 k19 = 1/10;
22 kdeg = 1/10;
23 gamma = 1/100;
24
25 A = zeros(28,28) ;
26
27 A(2,1) = k1;
28 A(1,2) = k_2;
29 A(23,3) = k3;
30 A(5,23) = k4;
31 A(24,6) = k5;
32 A(8,24) =k6;
33 A(10,9) = k7;
34 A(9,10) = k_8;
35 A(25,11) = k9;
36 A(13,25) = k10;
37 A(26,14) = k11;
38 A(14,26) = k_12;
39 A(27,16) = k13;
40 A(18,27) = k14;
41 A(28,19) = k15;
42 A(19,28) = k_16;
43 A(22,21) = k17;
44 A(4,22) = k18;
45 A(12,7) = k19;
46
47 Z = zeros(22,28);
48
49 Z(1,1)=1; Z(2,1)=1;
50 Z(6,2)=1;
51 Z(6,3)=1; Z(3,3)=1;
52 Z(14,4)=1; Z(18,4) = 1; Z(20,4) = 1;
53 Z(8,5)=1;Z(3,5)=1;Z(6,5)=1;
54 Z(6,6)=1;Z(4,6)=1;
55 Z(20,7)=1;
56 Z(10,8)=1;Z(4,8)=1;Z(6,8)=1;
57 Z(11,9)=1;Z(17,9)=1;
58 Z(1,10)=1;
59 Z(11,11)=1;Z(5,11)=1;
60 Z(21,12)=1;
61 Z(5,13)=1;Z(13,13)=1;
62 Z(10,14)=1;Z(8,14)=1;
63 Z(10,15)=1;
64 Z(15,16)=1;Z(14,16)=1;
65 Z(8,17)=1;
66 Z(17,18)=1; Z(14,18)=1;
67 Z(8,19)=1; Z(13,19)=1;
68 Z(13,20) = 1;
69 Z(18,21)=1;Z(14,21)=1;
```

```
Z(19,22)=1;
Z(7,23)=1;
Z(9,24)=1;
Z(22,25)=1;
Z(15,26)=1;
Z(16,27)=1;
Z(18,28)=1;


D  = diag( [k1, k_2, k3, 0,0,k5,k19,0,k7,k_8,k9,0,0,k11,kdeg,k13,kdeg,0,
    k15,kdeg,k17,k18,k4,k6,k10,kdeg+k_12,k14,kdeg+k_16]) ;
L = D - A ;



red_ord = 6; % reduction order for the Kron reduction order method

L11 = L(1:end-red_ord, 1:end-red_ord);
L12 = L(1:end-red_ord, end-red_ord+1:end);
L21 = L(end-red_ord+1:end, 1:end- red_ord);
L22 = L(end - red_ord +1: end , end - red_ord +1: end);

Znew = Z(:, 1:end-red_ord) ;

Lhat =  L11 - L12*inv(L22)*L21; % Schur complement for auxiliary dynamics

dxdt = - ( Znew*Lhat )*exp(Znew'*log(xred) );  % + Z1*ones(21,1)  ;


%dxdt = -Z*L*exp(Z'*log(xred)) + gamma*(xin_red - xred);
```

C-5 Linearization

```
function y = Linearization_func(t, x)


Alinearization =
    [-0.498000000000000,0,0,0,0,0.100000000000000,0,0,0,0,2.34000000000000,0,0,0,0,0,0


y = Alinearization*x;


clear all; close all; clc;
syms k1 k_1 k2 k3 k4 k5 k6 k_6 k7 k8 k9 k_9 k10 k11 k12 k_12 k13 k14 k15
    kdeg real
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19
    x20 x21 x22 x real


syms vx

```

```
N = zeros(22,23);

N(1,1) = -1; N(2,1) = -1; N(6,1) = 1;
N(1,2) = 1; N(2,2) = 1;N(6,2) = -1;
N(6,3) = -1; N(3,3) = -1;N(7,3) = 1;
N(7,4) = -1;N(8,4) = 1;N(3,4) = 1;N(6,4) = 1;
N(6,5) = -1;N(4,5) = -1;N(9,5) = 1;
N(9,6) = -1;N(10,6) = 1;N(4,6) = 1; N(6,6) = 1;
N(11,7) = -1;N(17,7) = -1;N(1,7) = 1;
N(11,8) = 1;N(17,8) = 1;N(1,8) = -1;
N(11,9) = -1;N(5,9) = -1;N(22,9) = 1;
N(22,10) = -1;N(5,10) = 1;N(13,10) = 1;
N(10,11) = -1;N(8,11) = -1;N(15,11) = 1;
N(15,12) = -1;N(14,12) = -1;N(16,12) = 1;
N(16,13) = -1;N(17,13) = 1;N(14,13) = 1;
N(8,14) = -1;N(13,14) = -1;N(18,14) = 1;
N(18,15) = -1;N(8,15) = 1;N(13,15) = 1;
N(18,16) = -1;N(14,16) = -1;N(19,16) = 1;
N(19,17) = -1;N(14,17) = 1;N(20,17) = 1;N(18,17) = 1;
N(20,18) = -1;N(21,18) = 1;
N(10,19) = -1;
N(8,20) = -1;
N(13,21) = -1;
N(15,22) = -1;
N(18,23) = -1;

vx(1) = k1*x1*x2;
vx(2) = k_1*x6;
vx(3) = k2*x3*x6;
vx(4) = k3*x7;
vx(5) = k4*x6*x4;
vx(6) = k5*x9;
vx(7) = k6*x11*x17;
vx(8) = k_6*x1;
vx(9) = k7*x11*x5;
vx(10) = k8*x22 ;
vx(11) = k9*x10*x8;
vx(12) = k10*x15*x14;
vx(13) = k11*x16;
vx(14) = k12*x8*x13;
vx(15) = k_12*x18;
vx(16) = k13*x18*x14;
vx(17) = k14*x19;
vx(18) = k15*x20;
vx(19) = kdeg*x10;
vx(20) = kdeg*x8;
vx(21) = kdeg*x13;
vx(22) = kdeg*x15;
vx(23) = kdeg*x18;
vx = vx';

xdot = N*vx;

```

```
62 syms T
63
64 T(1,2) = k_1;
65 T(1,10) = -k_6;
66 T(1,1) = -k1;
67 T(1,9) = k6;
68 T(2,2) = k_1;
69 T(2,1) = -k1;
70 T(3,23) = k3;
71 T(3,3) = -k2;
72 T(4,24) = k5;
73 T(4,6) = -k4;
74 T(5,25) = k8;
75 T(5,11) = -k7;
76 T(6,23) = k3;
77 T(6,24) = k5;
78 T(6,2) = -k_1;
79 T(6,1) = k1;
80 T(6,3) = -k2;
81 T(6,6) = -k4;
82 T(7,3) = k2;
83 T(7,23) = -k3;
84 T(8,23) = k3;
85 T(8,28) = k_12;
86 T(8,17) = -kdeg;
87 T(8,14) = -k9;
88 T(8,19) = -k12;
89 T(9,6) = k4;
90 T(9,24) = -k5;
91 T(10,24) = k5;
92 T(10,15) = -kdeg;
93 T(10,14) = -k9;
94 T(11,10) = k_6;
95 T(11,11) = -k7;
96 T(11,9) = -k6;
97 T(13,25) = k8;
98 T(13,28) = k_12;
99 T(13,20) = -kdeg;
100 T(13,19) = -k12;
101 T(14,27) = k11;
102 T(14,22) = k14;
103 T(14,16) = -k10;
104 T(14,21) = -k13;
105 T(15,14) = k9;
106 T(15,26) = -kdeg;
107 T(15,16) = -k10;
108 T(16,16) = k10;
109 T(16,27) = -k11;
110 T(17,27) = k11;
111 T(17,10) = k_6;
112 T(17,9) = -k6;
113 T(18,22) = k14;
114 T(18,28) = -k_12-kdeg;
```

```
115 T(18,19) = k12;
116 T(18,21) = -k13;
117 T(19,21) = k13;
118 T(19,22) = -k14;
119 T(20,22) = k14;
120 T(20,7) = -k15;
121 T(21,7) = k15;
122 T(22,11) = k7;
123 T(22,25) = -k8;
124
125 syms L Z
126
127 Z = zeros(22,28) ;
128 Z(1,1)=1; Z(2,1)=1;
129 Z(6,2)=1;
130 Z(6,3)=1; Z(3,3)=1;
131 Z(14,4)=1; Z(18,4) = 1; Z(20,4) = 1;
132 Z(8,5)=1;Z(3,5)=1;Z(6,5)=1;
133 Z(6,6)=1;Z(4,6)=1;
134 Z(20,7)=1;
135 Z(10,8)=1;Z(4,8)=1;Z(6,8)=1;
136 Z(11,9)=1;Z(17,9)=1;
137 Z(1,10)=1;
138 Z(11,11)=1;Z(5,11)=1;
139 Z(21,12)=1;
140 Z(5,13)=1;Z(13,13)=1;
141 Z(10,14)=1;Z(8,14)=1;
142 Z(10,15)=1;
143 Z(15,16)=1;Z(14,16)=1;
144 Z(8,17)=1;
145 Z(17,18)=1; Z(14,18)=1;
146 Z(8,19)=1; Z(13,19)=1;
147 Z(13,20) = 1;
148 Z(18,21)=1;Z(14,21)=1;
149 Z(19,22)=1;
150 Z(7,23)=1;
151 Z(9,24)=1;
152 Z(22,25)=1;
153 Z(15,26)=1;
154 Z(16,27)=1;
155 Z(18,28)=1;
156
157 x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19
        x20 x21 x22]';
158
159
160 syms L D A
161 D = diag( [k1, k_1, k2, 0,0,k4,k15,0,k6,k_6,k7,0,0,k9,kdeg,k10,kdeg,0,
        k12,kdeg,k13,k14,k3,k5,k8,kdeg,k11,k_12+kdeg]) ;
162
163 % A = zeros(28,28) ;
164
165 A(1,1) = 0;
```

```
166 A(2,1) = k1;
167 A(1,2) = k_1;
168 A(23,3) = k2;
169 A(5,23) = k3;
170 A(24,6) = k4;
171 A(8,24) =k5;
172 A(10,9) = k6;
173 A(9,10) = k_6;
174 A(25,11) = k7;
175 A(13,25) = k8;
176 A(26,14) = k9;
177 A(27,16) = k10;
178 A(18,27) = k11;
179 A(28,19) = k12;
180 A(19,28) = k_12;
181 A(22,21) = k13;
182 A(4,22) = k14;
183 A(12,7) = k15;
184
185 L = D - A ;
186 -Z*L - T ;
187
188 f = N*vx;
189 syms gamma real
190 % f2 = N*vx + gamma*(xin - x);
191 % f3 = N*vx + gamma*(xin - x) - zeta*D_zeta;
192 syms jac
193 jac = jacobian(f, x);
194 xin =
        [0;3.98000000000000;1.99000000000000;1.99000000000000;2;0;0.0100000000000000;0.010
195
196
197
198 jac = subs(jac, x1, xin(1));
199 jac =  subs(jac, x2, xin(2));
200 jac =  subs(jac, x3, xin(3));
201 jac =  subs(jac, x4, xin(4));
202 jac =  subs(jac, x5, xin(5));
203 jac =  subs(jac, x6, xin(6));
204 jac =  subs(jac, x7, xin(7));
205 jac =  subs(jac, x8, xin(8));
206 jac =  subs(jac, x9, xin(9));
207 jac = subs(jac, x10, xin(10));
208 jac = subs(jac, x11, xin(11));
209 jac = subs(jac, x12, xin(12));
210 jac = subs(jac, x13, xin(13));
211 jac = subs(jac, x14, xin(14));
212 jac = subs(jac, x15, xin(15));
213 jac = subs(jac, x16, xin(16));
214 jac = subs(jac, x17, xin(17));
215 jac = subs(jac, x18, xin(18));
216 jac = subs(jac, x19, xin(19));
```

```
217 jac = subs(jac, x20, xin(20));
218 jac = subs(jac, x21, xin(21));
219 jac = subs(jac, x22, xin(22));
220
221 jac = subs(jac, k1, 1/10);
222 jac = subs(jac, k_1, 1/10);
223 jac = subs(jac, k2, 1/10);
224 jac = subs(jac, k3, 1/10);
225 jac = subs(jac, k4, 1/10);
226 jac = subs(jac, k5, 1/10);
227 jac = subs(jac, k6, 1/10);
228 jac = subs(jac, k_6, 1/10);
229 jac = subs(jac, k7, 1/10);
230 jac = subs(jac, k8, 1/10);
231 jac = subs(jac, k9, 1/10);
232 jac = subs(jac, k_9, 1/10);
233 jac = subs(jac, k10, 1/10);
234 jac = subs(jac, k11, 1/10);
235 jac = subs(jac, k12, 1/10);
236 jac = subs(jac, k_12, 1/10);
237 jac = subs(jac, k13, 1/10);
238 jac = subs(jac, k14, 1/10);
239 jac = subs(jac, k15, 1/10);
240 jac = subs(jac, kdeg, 1/10);
241
242 jac = double(jac);
243 %%
244
245 Alinear = zeros(22,22);
246 for k = 1:22
247     for m = 1:22
248         Alinear(m,k) =  jac(m,k) ;
249     end
250
251 end
```

Appendix D

Glossary

D-1 Acronyms

Delft University of Technology (TU Delft)

3mE	Mechanical, Maritime and Materials Engineering
CFFL	Coherent Feedforward Loop
DCSC	Delft Center for Systems and Control
E. Coli	Escherichia Coli
eGFP	Green Fluorescent Protein
ICMS	Institute of Complex Molecular Systems
MM	Michaelis-Menten
ODE	Ordinary Differential Equation
QSSA	Quasi-steady state approach
S28	σ-factor 28
S70	σ-factor 70
TU Delft	Delft University of Technology