T.M.J. van Zalen

Modular Response Analysis: A method for reconstructing chemical networks

Bachelor Thesis, 15th June 2011

Thesis Supervisor: dr. S.C. Hille

Mathematisch Instituut, Universiteit Leiden

Abstract

This Bachelor thesis is about the Modular Response Analysis, a method to reconstruct a biochemal network from experimental data. It mathematically describes two methods, one based on the results from the steady state of a chemical network, and one based on the results of different time series in a chemical network.
For these methods, also the Singular Value Decomposition is described. This decomposition is helpful in solving overdetermined systems of equations, which is a common situation when applying these methods in an experimental environment.

Contents

1 Introduction **3**

2 Network reconstruction with Modular Response Analysis **5**

2.1 The steady state method 5

2.2 The time series data method 7

3 Solving the MRA equations **10**

3.1 Singular Value Decomposition 10

3.2 The inhomogeneous equation 12

3.3 The homogeneous equation 12

4 The TCA cycle **14**

4.1 Description of the reaction network 14

4.2 A mathematical model for network dynamics 15

4.3 The computed Jacobian matrix 16

4.4 Network reconstruction from the Jacobian 18

4.5 Admissible perturbations 20

4.6 Approximation of the Jacobian by MRA 22

5 Application of time series data method **24**

5.1 The computed Jacobian matrix 24

5.2 Approximation of the Jacobian by MRA 24

6 Discussion and suggestions for further research **26**

A Index of notation **28**

B MATLAB codes for steady state method **29**
B.1 Jacobian matrix 30
B.2 First approximation of Jacobian matrix 30
B.3 Second approximation 32

C Codes for timeseries method **33**

References **36**

1 Introduction

When the behaviour of a chemical network is observed, a central concern is the discovery of the underlying architecture of the network. This reverse engineering problem can be solved with the unravelling of the web of interactions among the components of the network. It is hard to determine the direct interactions, since in intact (living) cells, any perturbation to a component will extend through the rest of the network, causing global changes, such as changes in the eventual steady state of the system. These global effects cannot be easily distinguished from the local effects.
This leads to the question how to use these global effects and still derive the direct interactions between individual nodes. Modular Response Analysis (MRA), the subject of this bachelor thesis, provides a means to obtain (approximate) information on local interactions from global responses of the system. It has been developed by Sontag and coworkers in a series of papers. Our exposition is based on [1] and [2].

The objectives of this bachelor project were to understand why the described MRA methods were able - theoretically - to reconstruct network structure and to investigate their functioning in an artificial experimental environment. That is, we planned to simulate data for a known reaction network and sought to reconstruct the network from this artificial data.

Let us introduce the ideas of the MRA method in some more detail. Normally, a network consists of n known nodes which are assigned by the state variables $x_i(t)$, $i = 1, \ldots, n$, and l unkown nodes which are assigned by the state variables $y_j(t)$, $j = 1, \ldots, l$, which represent the levels of compounds in a network at time t. A system of oridnary differential equations is assumed to model the dynamics:

$$\begin{cases} \dot{x} = g(x, y, p) \\ \dot{y} = h(x, y, p), \end{cases}$$

where the functions g_i and h_j describe how the rate of x_i and y_j depends on other network elements. The parameter vector $p = (p_1, \ldots, p_m)$, $m \geq n$, represent any external or internal condition. This parameter is assumed to remain constant during an experiment.

Since the problem becomes very complex if we were to consider also the unkown state variables, and since the function $h(x, y, p)$ is not known and hard to compute, we will look at a more simple network, where we omit the unknown state variables y_i. We now get a network which consists of n nodes, or the state variables $x_i(t)$, $i = 1, \ldots, n$. These state variables are collected into a vector $x = (x_1, \ldots, x_n)$.
The corresponding dynamical system is described by a system of differential equations

$$\begin{cases} \dot{x} = f(x, p) \\ x(0) = x^0, \end{cases} \tag{1.1}$$

with $x^0 \in \mathbb{R}^n$. The corresponding functions f_i describe how the rate of x_i depends on the other elements of the network.
The values which are measured are samples of the solution to the system (1.1) at various time points $t = t_1, \ldots, t_k$.

The problem is to find an expression for the unknown function $f(x,p)$ that suits the measurements. This gives a large space of possibilities, since the function space is infinite dimensional. An idea is to limit these possibilities by assuming that f has local Taylor series expansions and try to approximate the first order expansion from the data.
The Taylor series expansion of f around $x = x^0$ gives

$$f(x,p) = f(x^0,p) + Df(x^0,p)(x - x^0) + R(x,p),$$

where $R(x,p)$ is the remainder of the Taylor series expansion, which contains higher order terms.

We will consider two methods which can be used to reconstruct the local interactions in a network, based on the global respons. These methods are the so called Modular Response Analysis (MRA) methods.
The first method is based on steady state data. It compares the steady states which result after performing independent perturbations in a component of the network.
The second method measures the original and perturbed time series, which means that the network does not need to be in steady state.

Both methods can be used to obtain the Jacobian matrix $F = Df(x)$, which describes the influence of each variable x_j upon the rate f_i of change of every other variable x_i. The idea is, that the Jacobian provides sufficient information to reconstruct the network. Let us consider these methods now in further detail.

2 Network reconstruction with Modular Response Analysis

The first method of MRA we will analyse in order to reconstruct a chemical network is based on steady state data. The second method we will analyse is the MRA method based on time series data.

2.1 The steady state method

Below we mathematically explain the Modular Response Analysis (MRA) method for network reconstruction using steady state data, as developed by Sontag and coworkers [1].

Definition 1 (Steady State)**.** A system $\dot{x} = f(x,p)$ is in *steady state* if $\dot{x} = 0$. This is when $x = x^{ss}$ and $f(x^{ss}, p) = 0$.

As mentioned in the introduction, we may assume that f has a local Taylor series expansion around $x = x^{ss}$:

$$f(x,p) = f(x^{ss},p) + Df(x^{ss},p)(x - x^{ss}) + R(x,p),$$

where $R(x,p)$ represents the remainder.
Following Definition 1, $f(x^{ss}, p) = 0$, which results in the expansion

$$f(x,p) = Df(x^{ss},p)(x - x^{ss}) + R(x,p). \tag{2.1}$$

In order to avoid triviality, we make the following assumption.

Assumption 1. For $x = x^{ss}$,

$$\frac{\partial f}{\partial x}(x,p) \neq 0.$$

Now we can introduce the MRA method. This method consists of two steps:

1. First experimentally measure the steady state x^{ss} which corresponds to the unperturbed parameter vector $p = p_0$, which is the initial parameter setting.
2. Then a perturbation to a well-chosen selection of the elements of the parameter vector are made, say p_0 is changed into $p_1, \dots, p_m$, which gives for all i a new steady state $x^{ss}(p_i)$.

With these steps we are able to estimate the Jacobian matrix $Df(x^{ss}, p)$ as follows for $p = p_0$. First fix i and let us consider the reconstruction of $F_i = \frac{\partial f_i}{\partial x} = \nabla f_i$.
We need to make the following assumptions

Assumption 2. (a) The vectors $\frac{\partial x^{ss}}{\partial p_k}(p)$ are linearly independent.

(b) For every $i \in \{1, \ldots, n\}$ there exists a collection of indices $\mathcal{P}_i$ such that f_i does not depend upon p_k, and

$$\frac{\partial f_i}{\partial p_k} \equiv 0,\ k \in \mathcal{P}_i.$$

These are called *admissible* for node i.

With these assumptions, we can take the partial derivatives of $f_i(x^{ss}(p), p) = 0$ with respect to p_k. This gives the following equation

$$\begin{aligned} 0 &= \frac{\partial}{\partial p_k}\left[f_i(x^{ss}(p)), p)\right] \\ &= \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^{ss}(p), p) \cdot \frac{\partial x_j^{ss}}{\partial p_k}(p) + \frac{\partial f_i}{\partial p_k}(x^{ss}(p), p) \\ &= \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^{ss}(p), p) \cdot \frac{\partial x_j^{ss}}{\partial p_k}(p). \end{aligned} \tag{2.2}$$

For simplicity we define the $n \times m$ matrix

$$\Sigma := \left(\frac{\partial x_j^{ss}}{\partial p_k}(p)\right)_{j,k}.$$

System (2.2) can now be written simply as

$$F_i \Sigma = 0. \tag{2.3}$$

A few remarks are in place here. Firstly, note that F_i is not uniquely determined by (2.3): if $F_i = F$ satisfies (2.3), then also λF for any $\lambda \in \mathbb{R}$. So we will need a normalisation of some type to obtain a unique solution.
Secondly, we do not know $\frac{\partial x_j^{ss}}{\partial p_k}(p)$ from experiments, because it is hard or even experimentally imposible to know the complete parameter setting p_0 and the precise change made therein resulting in the new setting $p_1, \ldots, p_k$.
However, the matrix Σ may be replaced with one that can be estimated, as follows. An approximation of $\frac{\partial x_j^{ss}}{\partial p_k}(p)$ is for $p = p_0$

$$\frac{\partial x_j^{ss}}{\partial p_k}(p) \approx \frac{x_j^{ss}(p_k) - x_j^{ss}(p_0)}{||p_k - p_0||}, \tag{2.4}$$

which gives an approximate equation to (2.2)

$$0 \approx \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^{ss}(p), p) \cdot \frac{x_j^{ss}(p_k) - x_j^{ss}(p_0)}{||p_k - p_0||}. \tag{2.5}$$

If the perturbation p_k is not too large compared to the unperturbed parameter p_0, we can multiply both sides of equation (2.5) with $||p_k - p_0||$, which results in the equation

$$0 \approx \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^{ss}(p), p) \cdot (x_j^{ss}(p_k) - x_j^{ss}(p_0)). \tag{2.6}$$

We then define

$$\hat{\Sigma} := \left(x_j^{ss}(p_k) - x_j^{ss}(p_0)\right)_{j,k}.$$

Equation (2.6) can now be written simply as

$$F_i\hat{\Sigma} \approx 0. \tag{2.7}$$

There are now two possibilities in order to solve this equation.
If there are exactly $n-1$ perturbations performed, i.e. when the matrix $\hat{\Sigma}$ has full rank $n-1$, it is possible to determine F_i uniquely up to scalar multiplication. This can be done using Gaussian elimination.
When there are more then $n-1$ perturbations performed, we have an overdetermined system. This system may be solved for the 'best' solution, i.e. with the least mean square error, with the aid of the singular value decomposition.

2.2 The time series data method

The MRA method can also be employed in situations where steady state data cannot be obtained, or is irrelevant, such as in understanding signaling networks. There, the transient dynamics may be more important than the steady state of the system. Sontag and coworkers have elaborated their MRA approach in [2]. With this method the original and perturbed time series, describing the time dependence of each network variable, are measured for each perturbation. These time series approximate the solution of

$$\dot{x} = f(x,p). \tag{2.8}$$

As for the steady state method, we assume that there exists a collection $\mathcal{P}_i$ of parameters p_j, known to have the property that

$$\frac{\partial f_i}{\partial p_k} \equiv 0,\ p_k \in \mathcal{P}_i. \tag{2.9}$$

The time series method then works as follows. For each perturbation p_k, we measure for each time t_j, $j = 1,2,\ldots$, the value $x_i(t_j,p)$. These values are samples of the solutions to equation (2.8), corresponding to the time values t_j.
The time-dependent response of x_i, $i = 1,\ldots,n$, to a perturbation p_k is defined as

$$\frac{\partial x_i(t,p)}{\partial p_k} = \frac{x_i(t,p_k) - x_i(t,p_0)}{||p_k - p_0||}. \tag{2.10}$$

The response of x_i at succesive time points $0 < t_j < t_{j+1} < \ldots$, is defined as

$$\begin{aligned}\frac{\partial x_i}{\partial t}(t_j,p) &\approx d_{ij}(p)\\ &:= \frac{1}{2}\left[\frac{x_i(t_{j+1},p) - x_i(t_j,p)}{t_{j+1}-t_j} + \frac{x_i(t_j,p) - x_i(t_{j-1},p)}{t_j - t_{j-1}}\right].\end{aligned} \tag{2.11}$$

We want to estimate the Jacobian matrix $F(t) = Df(x(t,p))$ from the experimental time series. To that end we also need the second-order sensitivities, which can be estimated from the measurements of the responses $d_{ij}(p)$ at a perturbation p_k. This gives

$$\begin{aligned}\frac{\partial^2 x_i}{\partial t \partial p_k}(t_j, p) &\approx \frac{\partial d_{ij}(p)}{\partial p_k} \\ &\approx \frac{d_{ij}(p_k) - d_{ij}(p_0)}{||p_k - p_0||}. \end{aligned} \tag{2.12}$$

We are now able to find the Jacobian matrix $F(t)$. This can be done as follows. The dynamic behavior of the node i is described by $x_i(t, p)$, which is the solution to

$$\dot{x}_i = f_i(x, p).$$

Taking the derivatives on both sides with respect to p_k, and using (2.10)- (2.11), we find the unknown elements of the i-th row of the Jacobian matrix, which is $(F_{i1}, \ldots, F_{in})$. In fact the i-th row of Jacobian matrix F then satisfies

$$\begin{aligned}\frac{\partial^2 x_i}{\partial t \partial p_k}(t_j, p) &= \sum_{m=1}^{n} \frac{\partial f_i(x(t_j, p), p)}{\partial x_m} \frac{\partial x_m}{\partial p_k}(t_j) + \frac{\partial f_i(x(t_j, p), p)}{\partial p_k} \\ &= \sum_{m=1}^{n} \frac{\partial f_i(x(t_j, p), p)}{\partial x_m} \frac{\partial x_m}{\partial p_k}(t_j), \end{aligned} \tag{2.13}$$

where we used the admissibility of perturbation p_k, i.e. $\frac{\partial f_i}{\partial p_k} \equiv 0$.
If we define

$$\begin{aligned} b_i(t_j) &:= \left(\frac{\partial^2 x_i}{\partial t \partial p_k}(t_j, p) \right)_k \\ \Sigma(t_j) &:= \left(\frac{\partial x_m}{\partial p_k}(t_j) \right)_{m,k}, \end{aligned}$$

where b_i is an m-row vector and Σ an $n \times m$ matrix, we can write equation (2.13) in the condensed form

$$F_i(t_j)\Sigma(t_j) = b_i(t_j).$$

Since we cannot know the perturbations exactly, we approximate the exact equation (2.13) as follows:

$$\frac{d_{ij}(p_k) - d_{ij}(p_0)}{||p_k - p_0||} \approx \sum_{m=1}^{n} \frac{\partial f_i(x(t_j, p), p)}{\partial x_m} \frac{x_m(t_j, p_k) - x_m(t_j, p_0)}{||p_k - p_0||}. \tag{2.14}$$

If the perturbation p_k is not too large compared to the original parameter setting p_0, we can multiply both sides of equation (2.14) with $||p_k - p_0||$, which results in

$$d_{ij}(p_k) - d_{ij}(p_0) \approx \sum_{m=1}^{n} \frac{\partial f_i(x(t_j, p), p)}{\partial x_m} \left(x_m(t_j, p_k) - x_m(t_j, p_0) \right). \tag{2.15}$$

When we define

$$\begin{aligned} \hat{b}_i(t_j) &:= \left(d_{ij}(p_k) - d_{ij}(p_0) \right)_k \\ \hat{\Sigma}(t_j) &:= \left(x_m(t_j, p_k) - x_m(t_j, p_0) \right)_{m,k}, \end{aligned}$$

we can simplify equation (2.15) to

$$F_i(t_j)\hat{\Sigma}(t_j) \approx \hat{b}_i(t_j), \tag{2.16}$$

which we have to solve for $F_i(t_j)$.

We shall now discuss methods to find a (approximate or best) solution to the ‘master equations’ (2.7) and (2.16) for both MRA methods.

3 Solving the MRA equations

Since we have two equations to solve, one homogeneous and one non homogeneous equation, we will solve these solutions separately. Both approaches use the Singular Value Decomposition (SVD) for matrices. We shall discuss this first.

Remark 1. The method described in this section is based on Dym [3], p. 207–215.

We now have two equations to solve in order to find the approximate solution F. The equation for i-th row of the Jacobian matrix F for the steady state method is

$$F_i\hat{\Sigma} \approx 0, \tag{3.1}$$

and the equation for the time series method is

$$F_i\hat{\Sigma} \approx \hat{b}_i, \tag{3.2}$$

with F_i an n-row vector, Σ an $(n \times m)$-matrix and b_i an m-row vector.
We can rewrite the equations (3.1) and (3.2) in the following form

$$Ax = b, \tag{3.3}$$

by defining $A := \Sigma^T$, $x := F_i^T$ and $b := b_i^T$.
Normally, when $\text{rank}(A) = n$ and $b \in \text{Col}(A)$, there is a unique solution to equation (3.3), which can be found using Gaussian elimination.
We also have the possibility that $b \notin \text{Col}(A)$. Then there is no solution to (3.3). We then have to find an $\hat{x}$, such that $A\hat{x} = \hat{b}$, with $\hat{b} \in \text{Col}(A)$ as close as possible to b, i.e. $||\hat{b} - b||_2$ is minimal.

Remark 2. Note that $|| \cdot ||_2$ is the Euclidean norm on $\mathbb{R}^m$, with

$$||x||_2 = \left(\sum_i x_i^2\right)^{\frac{1}{2}}.$$

With the Singular Value Decomposition, we are able to find an approximate solution to equation (3.3), where we want to

$$\text{minimize}\{||Ax - b||_2 \text{ over } x \in \mathbb{R}^n\},$$

with respect to the norm $|| \cdot ||_2$.

3.1 Singular Value Decomposition

To state the Singular Value Decomposition theorem, we will first look at the non homogeneous equation, which is the equation which follows from the time series data method.

First we have to make the following definition.
Let $A \in \mathbb{R}^{m\times n}$. The matrices A^TA and AA^T are symmetric matrices of size

$n \times n$ and $m \times m$ respectively. The eigenvalues of A^TA and AA^T are non-negative, which can be seen as follows. Suppose λ is an eigenvalue of A^TA with eigenvector x. Then

$$||Ax||_2^2 = \langle Ax, Ax\rangle = \left\langle A^TAx, x\right\rangle = \langle \lambda x, x\rangle = \lambda ||x||_2^2,$$

so

$$\lambda = \frac{||Ax||_2^2}{||x||_2^2} \geq 0.$$

A similar argument shows that the eigenvalues of AA^T are non-negative.
Then there exists an orthogonal matrix $U \in \mathbb{R}^{n\times n}$ consisting of the orthonormal eigenvectors of A^TA, such that A^TA can be diagonalized as

$$A^TA = U \begin{bmatrix} s_1^2 & & \\ & \ddots & \\ & & s_n^2 \end{bmatrix} U^T, \tag{3.4}$$

with $s_1^2, \ldots, s_n^2$ the eigenvalues of A^TA.
It is assumed that these eigenvalues are indexed such that

$$s_1 \geq s_2 \geq \ldots \geq s_n \geq 0. \tag{3.5}$$

Definition 2 (Singular value)**.** The numbers s_j, $j = 1, \ldots, n$, are the *singular values* of the matrix A.

Remark 3. If rank$(A) = r$, with $r < n$, the maximal rank of A, then $s_r > 0$ and $s_{r+1}, \ldots, s_n = 0$.

Remark 4. One may show, that the singular values of A^T are the same as these of A.

Assumption 3. For the following we assume that A has full rank n.

Next we have to introduce the definition of the Singular Value Decomposition, in order to give the theorem to find an approximate solution to equation (3.3).

Theorem 1 (Singular Value Decomposition, [3], Thm. 10.1)**.** *Let $m \geq n$, and $A \in \mathbb{R}^{m\times n}$ be a matrix of rank n with singular values $s_1 \geq \ldots \geq s_n \geq 0$, and let*

$$D = diag\{s_1, \ldots, s_n\} \in \mathbb{R}^{n\times n}. \tag{3.6}$$

Then there exists an orthogonal matrix $V \in \mathbb{R}^{m\times m}$ consisting of the orthonormal eigenvectors of AA^T, and an orthogonal matrix $U \in \mathbb{R}^{n\times n}$, with $m \geq n$, such that

$$A = V \begin{bmatrix} D \\ 0_{(m-n)\times n} \end{bmatrix} U^T. \tag{3.7}$$

Equation (3.7) is called the *Singular Value Decomposition* of A. We shall write $A = VDU^T$ instead of (3.7) for simplicity.

Particular properties of the singular value decomposition enable us to find the best approximate solutions to (3.3).

3.2 The inhomogeneous equation

The following property, which we state for the maximal rank case only, will provide the 'best' solution to the inhomogeneous equation $Ax = b$:

Proposition 1 ([3], Lemma 10.6)**.** *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$ and $P_{\mathcal{R}_A}$ the orthogonal projection of $\mathbb{R}^m$ onto $\mathcal{R}_A$, i.e. the range of the matrix A. If $Ax = P_{\mathcal{R}_A}b$, then*

$$||Ax - b||_2^2 = ||(I_m - P_{\mathcal{R}_A})b||_2^2. \tag{3.8}$$

If rank$(A) = n$ and $V_1 = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_n]$ is built from the n columns in the matrix V in (3.7) *and $s_1, \ldots, s_n$ are the positive singular values of A, then*

$$P_{\mathcal{R}_A}b = V_1V_1^Tb = \sum_{j=1}^{n} \langle b, \boldsymbol{v}_j\rangle\, \boldsymbol{v}_j \tag{3.9}$$

and

$$||(I_m - P_{\mathcal{R}_A})b||_2^2 = ||b||_2^2 - \sum_{j=1}^{n} |\,\langle b, \boldsymbol{v}_j\rangle\,|^2 = \sum_{j=n+1}^{m} |\,\langle b, \boldsymbol{v}_j\rangle\,|^2. \tag{3.10}$$

Moreover, if $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ denote the columns in the matrix U in (3.7)*, then the vector*

$$x = UD^{-1}V_1^Tb = \sum_{j=1}^{n} \frac{\langle b, \boldsymbol{v}_j\rangle}{s_j}\boldsymbol{u}_j \tag{3.11}$$

is a solution of the equation

$$Ax = P_{\mathcal{R}_A}b. \tag{3.12}$$

This solution $\hat{x} = A^+b$ is also called the Moore-Penrose pseudoinverse, which is the best least square error approximation of the solution to (3.3). In MATLAB A^+ can be computed using the function `pinv(A)`.

3.3 The homogeneous equation

For the steady state method, the equation which has to be solved is

$$Ax = 0, \tag{3.13}$$

which is an homogeneous equation.
When rank$(A) = n$ is maximal, then $Ax = 0$ only has the trivial solution $x = 0$. In the steady state method we are looking for non-zero row vectors F_i for which $F_i\Sigma_j = 0$ for any column vector Σ_j of Σ in the exact settings. Since the MRA method in practise uses an approximation $\hat{\Sigma}$ to the exact matrix Σ, caused both by measurement errors and approximation of derivatives by differential quotients, one cannot expect to find a single non-zero vector F_i such that $F_i\hat{\Sigma}_j = 0$ for all columns $\hat{\Sigma}_j$ of $\hat{\Sigma}$. Instead one would look for a solution F_i for which all $F_i\hat{\Sigma}_j$ are as close to 0 as possible. This problem stated as such

does not have a solution: if F_i is such that all $F_i\hat{\Sigma}_j$ are small, then $\frac{1}{2}F_i$ will have all product smaller.
One therefore needs to normalize the vectors F_i in some fashion. We choose $||F_i||_2 = 1$.

With this choice, we get a constrained optimization problem

$$\min_{||x||_2=1} ||Ax||_2. \tag{3.14}$$

Again, we can write $A = VDU^T$.

Remark 5. A unitary matrix V preserves length:

$$||Vy||_2 \quad = \quad ||y||_2. \tag{3.15}$$

We then have to minimize

$$||Ax||_2 = ||VDU^T x||_2 = ||DU^T x||_2, \tag{3.16}$$

with $||x||_2 = ||U^T x||_2 = 1$. If we now subsitute

$$z = U^T x \in \mathbb{R}^n,$$

it means we have to minimize $||Dz||$ subject to $||z|| = 1$ since U is invertible. Since D is (essentially) a diagonal matrix, with the singular values on its diagonal, the minimal solution is $z = \pm e_n$, with e_n the n-th unit rowvector. Since $x = Uz$, the solution to equation (3.14) is $x = \pm U e_n$, that is, the vector u_n from Proposition 1.

4 The TCA cycle

In this section we will describe how the steady state method can be applied to a simplified tricarboxylic acid (TCA) cycle.

4.1 Description of the reaction network

The TCA cycle, also known as the Krebs cycle or cytric acid cycle, consists of enzyme catalysed reactions which are important for metabolic processes which take place in the mitochondrion in plants. This cycle is part of the reaction scheme which breaks down glucose into water and oxygen, and produces energy stored in the form of ATP and/or citric acid.

Figure 4.1 represents a simplified version of the TCA cycle as described in Steuer [4]. From the cytosol, pyruvate (Pyr) and malate (Mal) are transported into the mitochondrion. Mal will then be converted into Pyr or into oxaloacetate (OAA), which is a bidirectional reaction. Pyr and OAA will form citrate (Cit), which can be converted into Mal or will be transported to the cytosol. If Cit will be transported into the cytosol, there is an exchange reaction where cytosolic OAA and Mal will be transported into the mitochondrion.
All internal reactions use NAD$^+$ and convert it into NADH. In the mitochondrion there is an additional set of reactions, not shown in Figure 4.1, that converts NADH and ADP into NAD$^+$, thus producing energy for further use in the cell.

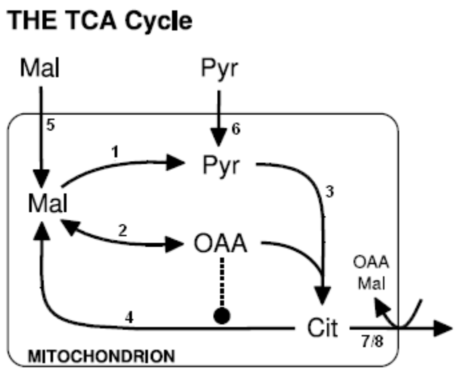

Figure 4.1: The simplified TCA cycle.

This model is adapted from a more detailed model described in Steuer *et al.* [5], where also the main reactions are described. The reactions that occur in Figure 4.1 are given in further detail in Table 4.1. There, the subscript ‘cyt’ indicates that the particular compound is located in the cytosol.

					Mitochondrion						
1	Mal	$+$	NAD^+			$\rightarrow$	Pyr	$+$	NADH	$+$	CO_2
2	Mal	$+$	NAD^+			$\rightleftharpoons$	OAA	$+$	NADH		
3	Pyr	$+$	NAD^+	$+$	OAA	$\rightarrow$	Cit	$+$	NADH	$+$	CO_2
4	Cit	$+$	2 NAD^+			$\rightarrow$	Mal	$+$	2 NADH		
					Energy						
5	2 NADH	$+$	3 ADP			$\rightarrow$	2 NAD^+	$+$	3 ATP		
					Transport						
6	Cit	$+$	OAA_{cyt}			$\rightarrow$	Cit_{cyt}	$+$	OAA		
7	Cit	$+$	Mal_{cyt}			$\rightarrow$	Cit_{cyt}	$+$	Mal		
8	Mal_{cyt}					$\rightarrow$	Mal				
9	Pyr_{cyt}					$\rightarrow$	Pyr				

Table 4.1: Main reactions of a simplified model of the TCA cylce.

4.2 A mathematical model for network dynamics

These reactions can be transformed into a system of differential equationa. Since we intend to investigate the applicability of the MRA methods using 'artificial data' that is generated by simulations, we may for this purpose use mass action kinetics for the chemical reactions shown in Table 4.1, instead of the appropriate functional forms for katalysed ractions (such as Michaelis-Menten kinetics).
Mass action kinetics can be used to give a model of the behavior of the unidirectional, i.e. reactions 1, 2, 4 and 5, and bidirectional reactions, i.e. reaction 3. For this system the mass action kinetics of the transport rate j is given by ef_j, and the mass action kinetics of the reaction rates i is denoted by ν_i. Here ef_1 and ef_2 correspond to the efflux of reaction 6 and 7 respectively, and ef_3 and ef_4 correspond to the transport reactions 8 and 9 respectively. The reaction rates, referring to the numbering in Table 4.1, are given by

ν_1	$=$	$k_1 \cdot [Mal] \cdot [NAD^+]$
ν_2	$=$	$k_2 \cdot [Mal] \cdot [NAD^+]$ - $k_3[OAA]\cdot$ $([NAD]_t - [NAD^+])$
ν_3	$=$	$k_4 \cdot [Pyr] \cdot [OAA] \cdot [NAD^+]$
ν_4	$=$	$k_5 \cdot [Cit] \cdot [NAD^+]^2$
ν_5	$=$	$k_6 \cdot ([NAD]_t\text{-}[NAD^+])^2$
ef_1	$=$	$f_1 \cdot [Cit]$
ef_2	$=$	$f_2 \cdot [Cit]$
ef_3	$=$	f_3
ef_4	$=$	f_4

Table 4.2: Reaction rates with mass action kinetics.

Here we have

$$[NAD]_t = [NAD^+] + [NADH].$$

These reaction rates can be used to make a system of first order ordinary differential equations.

$$
\begin{aligned}
\frac{d[\mathrm{Mal}]}{dt} &= \nu_4 - \nu_1 - \nu_2 + \mathrm{ef}_2 + \mathrm{ef}_3 \\
\frac{d[\mathrm{Pyr}]}{dt} &= \nu_1 - \nu_3 + \mathrm{ef}_4 \\
\frac{d[\mathrm{OAA}]}{dt} &= \nu_2 - \nu_3 + \mathrm{ef}_1 \\
\frac{d[\mathrm{Cit}]}{dt} &= \nu_3 - \nu_4 - \mathrm{ef}_1 - \mathrm{ef}_2 \\
\frac{d[\mathrm{NAD}^+]}{dt} &= -\nu_1 - \nu_2 - \nu_3 - 2\nu_4 + 2\nu_5. \qquad (4.1)
\end{aligned}
$$

Note that the volume of the mitochondrion has been implicitly incorporated into the constants $\mathrm{f}_1,\ldots,\mathrm{f}_4$ and $\mathrm{k}_1,\ldots,\mathrm{k}_6$.

We implemented this system of ordinary differential equations in MATLAB, version 7.7.0 (R2008b), to obtain a numerical solution to system (4.1), from which we can estimate the steady state. The code can be found in the appendix.
From this simulation we try to find the interactions of the nodes in the network. The MRA method is claimed to work without specific knowledge of the parameter values. Therefore we need not put effort into 'parameter mining' in the literature. The values used in the simulations can be found in Appendix B.

4.3 The computed Jacobian matrix

The Jacobian matrix can be found from the system of differential equations and is of the form

$$
Df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \qquad (4.2)
$$

The system of differential equations corresponding to the TCA cycle is known, which normally is not the case, so it is possible to find the Jacobian matrix belonging to this system. The system for the TCA cycle is given in (4.1). If we write, according to the numbering used in MATLAB,

$$
\begin{aligned}
x_1 &= [\mathrm{Mal}] \\
x_2 &= [\mathrm{Pyr}] \\
x_3 &= [\mathrm{OAA}] \\
x_4 &= [\mathrm{Cit}] \\
x_5 &= [\mathrm{NAD}^+] \\
x_{16} &= [\mathrm{NAD}]_t, \qquad (4.3)
\end{aligned}
$$

we have the following Jacobian matrix $Df(x)$. We seperate the matrix in column vectors, or F_i^T:

$$
\begin{aligned}
F_1 &= \begin{pmatrix} -k_1x_5 - k_2x_5 \\ 0 \\ k_3(x_{16} - x_5) \\ k_5(x_5)^2 + f_2 \\ 2k_5x_4 - k_1x_1 - k_2x_1 - k_3x_3 \end{pmatrix}^T, \\
F_2 &= \begin{pmatrix} k_1x_5 \\ -k_4x_3x_5 \\ -k_4x_2x_5 \\ 0 \\ k_1x_1 - k_4x_2x_3 \end{pmatrix}^T, \\
F_3 &= \begin{pmatrix} k_2x_5 \\ -k_4x_3x_5 \\ -k_3(x_{16} - x_5) - k_4x_2x_5 \\ f_1 \\ k_2x_1 + k_3x_3 - k_4x_2x_3 \end{pmatrix}^T, \\
F_4 &= \begin{pmatrix} 0 \\ k_4x_3x_5 \\ k_4x_2x_5 \\ -k_5(x_5)^2 - k_7 - f_1 - f_2 \\ k_4x_2x_3 - 2k_5x_4x_5 \end{pmatrix}^T, \\
F_5 &= \begin{pmatrix} -k_1x_5 - k_2x_5 \\ -k_4x_3x_5 \\ k_3(x_{16} - x_5) - k_4x_2x_5 \\ -2k_5(x_5)^2 \\ -k_1x_1 - k_2x_2 - k_3x_3 - 4k_5x_4x_5 - 4k_6(x_{16} - x_5) \end{pmatrix}^T, \qquad (4.4)
\end{aligned}
$$

and thus

$$
Df(x) = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix}. \qquad (4.5)
$$

The computed Jacobian matrix for the steady state method is then as follows. We use the initial and parameter values as mentioned in Appendix B.1. The steady state values found from the MATLAB simulations used to find this matrix are as follows

i	1	2	3	4	5
x_i^{ss}	3.2189	4.5720	0.3521	0.7426	0.6215

Table 4.3: Steady state values.

This results in the matrix

$$Df(x^{ss}) = \begin{pmatrix} -0.1243 & 0 & 0.2757 & 0.1386 & -0.5657 \\ 0.0622 & -0.0875 & -1.1366 & 0 & -0.3220 \\ 0.0622 & -0.0875 & -1.4123 & 0.4000 & -0.2516 \\ 0 & 0.0875 & 1.1366 & -0.5386 & 0.5516 \\ -0.1243 & -0.0875 & -0.8609 & -0.0773 & -2.7808 \end{pmatrix}. \qquad (4.6)$$

4.4 Network reconstruction from the Jacobian

From this Jacobian matrix we can derive a graph representation from which it should be possible to find the reaction network corresponding to this Jacobian. A method for network reconstruction is not described in [1], [2], or further literature. Since it should be possible, this subsection will describe an intuitive method to reconstruct the network from the Jacobian. A mathematical proof of this method is necessary, but won't be given here. This was beyond the scope of the Bachelor project.

First of all, we can find the graph representation of the Jacobian. For this graph we only use the sign of the values in the Jacobian. The method is as follows.
Each metabolite is represented by a node in the graph. Recall that around a steady state, $\dot{x} \approx Df(x^{ss})(x - x^{ss})$.
If $Df_{ij} < 0$, this means that if the concentration of metabolite j increases, while the others are kept constant, the concentration of metabolite i decreases. We then place a blunt arrow from the node representing metabolite j to node i.
If $Df_{ij} = 0$ it means that metabolite j does not influence metabolite i directly. We then place no arrow.
If $Df_{ij} > 0$ it means that metabolite j positively influences metabolite i, i.e. if the concentration of j increases, when the concentration of i increases. In this case we put in a pointed arrow from j to i. Due to auto-degradation, $Df_{ii} < 0$, which means that an increasing amount of metabolite i will decrease the amount of i. This will not be represented in the graph.
If we apply this method to the Jacobian matrix (4.6), we find the following model, see Figure 4.4. Since NAD$^+$ has influence to all other metabolites, we won't show NAD$^+$ in the model.

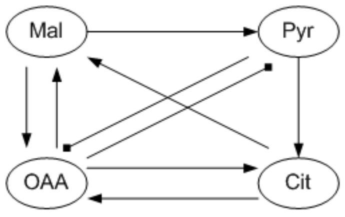

Figure 4.2: Graph representation for Jacobian matrix (4.6)

The next step is to convert this model to the reaction network. This means we have to interpret arrows or combinations of arrows. One possible interpretation is as follows.

If there is one arrow with point from metabolite A to B, there is a net reaction which convert A into B, or

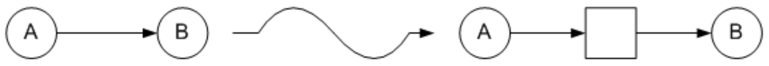

Figure 4.3: Conversion of A into B

Here we adopt a bipartite graph representation of a chemical reaction network. The box-shaped node indicates a (net) reaction.

If there are two blunt arrows, this could indicate that A and B are substrates in one reaction, which makes a new (unknown) product, or

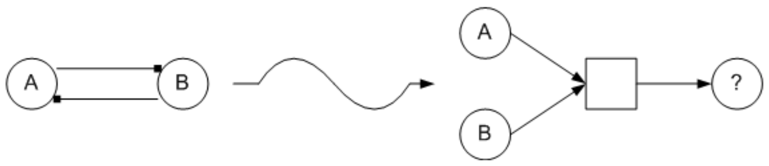

Figure 4.4: A and B togheter form an unidentified product, marked by '?'

The reasoning underlying this replacement is, that a slight increase in one of the substrates will increase the reaction rate. This effectively reduces the amount present of the other substrate.

With these interpretations, we can find a possible network associated to the graph in Figure 4.4 or the Jacobian (4.6). This results in the following network.

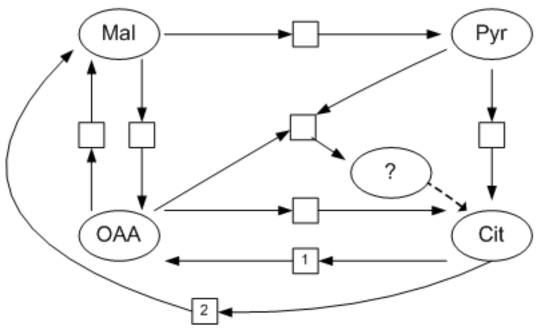

Figure 4.5: A possible reaction network corresponding to the Jacobian (4.6)

Since we see that Pyr and OAA each form Cit, we were able to draw a line from '?' to Cit. This can indicate that Pyr and OAA are the two substrates together forming Cit.

Boxes 1 and 2 indicate that these two reactions are exchange reactions with the cytoplasm. This information cannot be deduced from the experiments.

If we compare this reconstructed network with the original network from Figure 4.1, we see that the Jacobian yields a good approximation.
Now we can try to approximate the Jacobian from measured data. For this purpose, we first need to choose the perturbations.

4.5 Admissible perturbations

Recall that we need a collection $\mathcal{P}_i$ such that f_i does not depend on p_k, with $p_k \in \mathcal{P}_i$. These are called *admissible* for node i. We can eliminate some possibilities for the perturbations.
Each metabolite needs at least $n-1$ perturbations, so in this case each metabolite need at least 4 perturbations.

Normally we do not know the network in advance, so we will try to find the right perturbations based on biochemical information we know in advance. These perturbations can be chosen as follows:

(P1) We know in advance that there is an influx of Mal into the mitochondrion, so one possible perturbation is to change the influx of Mal. This perturbation only has influence on Mal.

(P2) We also know in advance that there is an influx of Pyr, so another possible perturbation is to change the influx of Pyr. This perturbation only involves Pyr.

(P3) A third possibility is to introduce a new reaction, where Cit will be broken down, by adding some enzyme which does not use NAD$^+$ or NADH. This reaction will be of the form

10	Cit $\rightarrow$ $\emptyset$

and the reaction rate then is

$$\boxed{\nu_6 \quad = \quad \mathrm{k}_7 \cdot [\mathrm{Cit}]}$$

This gives a new differential equation for $\frac{d[\mathrm{Cit}]}{dt}$:

$$\frac{d[\mathrm{Cit}]}{dt} = \nu_3 - \nu_4 - \nu_6. \tag{4.7}$$

(P4) If we know in advance that there is an efflux of mitochondrial Cit into the cytosol, where cytosolic Mal will be transported into the mitochondrion, a fourth possible perturbation is to change the export of Cit

(P5) Since there is an equilibrium reaction between OAA and Mal, another possibility is to influence this reaction, for example, by adding RNA inteference (RNAi), such that the associated enzyme level drops, or a metabolite.

(P6) Another possible perturbation is to influence the reaction where Pyr and OAA together form Cit, by adding RNAi.

(P7) A last perturbation can be to change the initial concentration of OAA.

(P8) For the time series data method, we need one more perturbation, involving only Pyr. Therefore, this perturbation can be to change the initial concentration of Pyr.

These perturbations have the following influence to the metabolites.
Here 'x' means the perturbation does not have direct influence to the metabolite and '-' means the perturbation has direct influence to the metabolite. Thus the perturbations in a row marked by 'x' together constitute the set of admissible perturbations for the metabolite associated to that row.

	P1	P2	P3	P4	P5	P6	P7	P8
Mal	-	x	x	-	-	x	x	x
Pyr	x	-	x	x	x	-	x	-
OAA	x	x	x	x	-	-	-	x
Cit	x	x	-	-	x	-	x	x
NAD^+	x	x	x	-	x	-	x	x

Table 4.4: Influence of the perturbations to the metabolites

From this we can find the Jacobian matrix F, which will be an approximation of the solution $\dot{x} = f(x,p)$. This solution represents the interactions between the nodes in the network.
After determining the approximated Jacobian matrix constructed from the results from the MRA method for steady states, we can compare this Jacobian matrix with the computed Jacobian to check the results.

4.6 Approximation of the Jacobian by MRA

The matrix A consisting of $(x_j^{ss}(p_k)-x_j^{ss}(p_0))_{k,j}$ is an (7×5)-matrix. To compute Fi, $i = 1,\ldots,5$, we only look at the smaller matrix consisting of the rows of matrix where a corresponding 'x' is in Table 4.5. For example, the matrix A1 consists of the second, third, sixth and seventh row of A.
The MATLAB code for the perturbations, and the corresponding matrices A1–A5 is as follows. In Appendix B.2 the parameter values and initial values can be found, which are used to calculate $x_j^{ss}(0), x_j^{ss}(1), \ldots, x_j^{ss}(7)$.

From the matrix MA constructed in MATLAB, we can find 5 smaller matrices, each consisting of the rows of MA where the perturbation does not have direct influence to the corresponding metabolite. These matrices then are

```
A1 = [MA(2,:);MA(3,:);MA(6,:);MA(7,:)];
A2 = [MA(1,:);MA(3,:);MA(4,:);MA(5,:);MA(7,:)];
A3 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:)];
A4 = [MA(1,:);MA(2,:);MA(5,:);MA(7,:)];
A5 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:);MA(7,:)];
```

The singular value decompositions of these matrices can be found using the MATLAB function [V,D,U]=SVD(X). This function produces a diagonal matrix D with the singular values on the diagonal, where D is of the same dimension as X, and unitary matrices V and U so that $X = VDU^T$.
The row vector F_i should satisfy $F_i\hat{\Sigma} \approx 0$, thus $\hat{\Sigma}^T F_i^T \approx 0$. If we compute the SVD of $\hat{\Sigma}^T$ we then have $\hat{\Sigma}^T = VDU^T$. For the minimal solution we have $F_i^T = \pm U e_n$, with e_n the n-th unit rowvector, such that e_n corresponds to the row of D with the smallest singular value.
From this we can construct the Jacobian matrix F^T, and thus F:

```
[V1,D1,U1] = svd(A1);
[V2,D2,U2] = svd(A2);
[V3,D3,U3] = svd(A3);
[V4,D4,U4] = svd(A4);
[V5,D5,U5] = svd(A5);

FT = [-U1(:,4) U2(:,5) -U3(:,4) -U4(:,4) U5(:,5)];
F = FT'
```

The values on the diagonal should all be negative, due to auto-degradation of the metabolites. This means we have to choose $\pm$ according to the sign on the diagonal in F.
With this normalisation we obtain for the approximate Jacobian for the first experiment the following

$$\hat{F}^{(1)} = \begin{pmatrix} -0.0584 & 0.0254 & 0.9198 & 0.3859 & -0.0314 \\ 0.0011 & -0.0392 & -0.1241 & -0.3117 & -0.9412 \\ 0.0368 & -0.0352 & -0.7498 & 0.2711 & 0.6014 \\ 0.0171 & 0.0630 & 0.2157 & -0.8427 & 0.4889 \\ 0.0117 & -0.0488 & -0.6768 & 0.1397 & -0.7210 \end{pmatrix}. \tag{4.8}$$

Since the values in this matrix do not perfectly correspond with the computed Jacobian matrix, we have to make a few changes to the simulation.

First, the relative changes in `xssp1` and `xssp5` are too large, which indicates one needs smaller perturbations, hopefully resulting in a smaller relative change in x^{ss}. The same argument holds for `xssp2` and `xssp4`, but these relative changes are smaller.

As mentioned, we first need to make the perturbations smaller. The perturbations are changed as follows

	Experiment 1	Experiment 2
p_1	x(14)=f_3: $0.2 \to 0.3$	x(14)=f_3: $0.2 \to 0.21$
p_2	x(15)=f_4: $0.2 \to 0.3$	x(15)=f_4: $0.2 \to 0.21$
p_3	x(17)=k_7: $0 \to 0.01$	x(17)=k_7: $0 \to 0.01$
p_4	x(13)=f_2: $0.1 \to 0.2$	x(13)=f_2: $0.1 \to 0.11$
p_5	x(7)=k_2: $0.1 \to 0.2$	x(7)=k_2: $0.1 \to 0.11$
p_6	x(9)=k_4: $0.4 \to 0.5$	x(9)=k_4: $0.4 \to 0.41$
p_7	x(3)=[OAA]: $10 \to 15$	x(3)=[OAA]: $10 \to 15$

Table 4.5: Changes in perturbations for the steady state method.

For the steady state method, we also should change the duration of the simulated solution using the `ode45` function, in this case from 100 to 1000.

The new MATLAB code can be found in Appendix B.3. The new Jacobian matrix is then

$$\hat{F}^{(2)} = \begin{pmatrix} -0.1587 & 0 & 0.3516 & 0.1199 & -0.9148 \\ 0.1007 & -0.0207 & -0.2213 & -0.2458 & 0.9381 \\ 0.0088 & -0.0513 & -0.7847 & 0.1258 & -0.6048 \\ 0.0372 & 0.0450 & 0.4871 & -0.1554 & 0.8574 \\ -0.0815 & 0.0215 & 0.5601 & -0.2646 & -0.7805 \end{pmatrix}. \tag{4.9}$$

In this approximated Jacobian are a few values for which the sign does not correspond with the 'exact' Jacobian 4.6. Since the sign of most of the values are correct, we can conclude that this steady state method is a good approximation for the Jacobian.

5 Application of time series data method

In this section we will describe how the time series data method can be applied to a simplied cycle. Here we use again the TCA cycle described in section 4. We will now look at different times, in stead of the final steady state.

Since the MATLAB code for this method is the same as the code for the function `TCA1` from the steady state method, we can also use the same perturbations as in section 4.

5.1 The computed Jacobian matrix

We first determine the exact Jacobian matrix at $t = 200$, with the code from Section C. This time is chosen, because the concentration of Mal then has a maximum. The values of $x(200, p_0)$ are determined using the MATLAB code yielding

i	1	2	3	4	5
$x_i(200, p_0)$	21.5949	2.0628	2.7268	1.6711	0.3105

Table 5.1: Values of $x(200, p_0)$

This gives as Jacobian matrix for the timeseries method

$$Df(x(200,p)) = \begin{pmatrix} -0.0621 & 0 & 0.3379 & 0.1096 & -4.5301 \\ 0.0311 & -0.3387 & -0.2562 & 0 & -0.0904 \\ 0.0311 & -0.3387 & -0.5941 & 0.4000 & 0.4550 \\ 0 & 0.3387 & 0.2562 & -0.5096 & 2.1461 \\ -0.0621 & -0.3387 & 0.0817 & -0.0193 & -6.7201 \end{pmatrix}. \quad (5.1)$$

The reaction network for this Jacobian is the same as the network in Figure 4.5. One should note that the sign of two values in Jacobian (5.1) are different from the values in Jacobian (4.6). These values correspond with the concentration NAD$^+$, hence it does not influence the reaction network.

5.2 Approximation of the Jacobian by MRA

We now have to determine the values of the metabolites at different times. As for the steady state method, we compute the numerical values of the metabolites corresponding to the admissible perturbations, with the function `ode45`. Here we use the same initial values and paramater values as for the steady state method. We also define vectors which exist of the metabolites values corresponding to time $t_j = 200$, $t_{j-1} = 175$ and $t_{j+1} = 225$.
According to Section 2.2, we need to calculate $\hat{\Sigma}$ and $\hat{b}_i$. These calculations can be found in Appendix C. We then have an equation of the form

$$F_i \hat{\Sigma} = \hat{b}_i, \quad (5.2)$$

with $i = 1, \ldots, 5$ corresponding to the metabolites. Since we need an equation of the form $Ax = b$, we write

$$\hat{\Sigma}^T F_i^T = \hat{b}_i^T. \tag{5.3}$$

In MATLAB code, this equation is `SiT*FiT=biT`, and if we calculate F_i^T, this can be found using the MATLAB code `FiT=pinv(SiT)*biT`. The calculations of `SiT` and `biT` can be found in Appendix C. Since we do not use the function `svd` here, we do not need to normalize the matrix such that the values on the diagonal are all negative.
If the F_i^T are calculated, we are able to find F:

```
FT = [F1T F2T F3T F4T F5T];
F= FT'
```

This yields the Jacobian matrix

$$\hat{F}^{(1)} = \begin{pmatrix} -0.0008 & -0.0098 & 0.0142 & 0.0139 & -0.4396 \\ 0.0008 & -0.0014 & 0.0004 & -0.0121 & 0.0417 \\ 0.0066 & 0.0028 & -0.0436 & 0.0277 & 0.6032 \\ 0.0001 & 0.0103 & 0.0064 & -0.0255 & 0.0245 \\ 0.0004 & 0.0011 & -0.0022 & 0.0015 & 0.0541 \end{pmatrix}. \tag{5.4}$$

As mentioned in Section 4.6, the perturbations are too large. This also means we need a new code to determine the new approximate Jacobian matrix.
The changes in the perturbations are as follows:

	Experiment 1	Experiment 2
p_1	x(14)=f_3: 0.2 $\to$ 0.3	x(14)=f_3: 0.2 $\to$ 0.21
p_2	x(15)=f_4: 0.2 $\to$ 0.3	x(15)=f_4: 0.2 $\to$ 0.21
p_3	x(17)=k_7: 0 $\to$ 0.01	x(17)=k_7: 0 $\to$ 0.01
p_4	x(13)=f_2: 0.1 $\to$ 0.2	x(13)=f_2: 0.1 $\to$ 0.11
p_5	x(7)=k_2: 0.1 $\to$ 0.2	x(7)=k_2: 0.1 $\to$ 0.11
p_6	x(9)=k_4: 0.4 $\to$ 0.5	x(9)=k_4: 0.4 $\to$ 0.41
p_7	x(3)=[OAA]: 10 $\to$ 15	x(3)=[OAA]: 10 $\to$ 11
p_8	x(2)=[Pyr]: 10 $\to$ 15	x(2)=[Pyr]: 10 $\to$ 11

Table 5.2: Changes in perturbations for the time series data method.

The new code can be found in Appendix C. With this code we can find a new approximated Jacobian matrix, which is

$$\hat{F}^{(2)} = \begin{pmatrix} -0.0034 & -0.0023 & 0.0212 & 0.0158 & -0.3491 \\ 0.0015 & -0.0372 & -0.0188 & -0.0088 & -0.1387 \\ 0.0050 & -0.0256 & -0.0608 & 0.0436 & 0.2144 \\ 0.0001 & 0.0293 & 0.0229 & -0.0458 & 0.1379 \\ -0.0075 & -0.0323 & 0.0183 & -0.0103 & -0.9951 \end{pmatrix}. \tag{5.5}$$

This Jacobian is a good approximation for the computed Jacobian (5.1). Although the values are not exactly the same as in (5.1), the sign of the values do correspond.
The time series data method therefore is a good method to approximate reaction network 4.5.

6 Discussion and suggestions for further research

In this project we succeeded in implementing in MATLAB an artificial 'experimental environment' for the simulation of global responses of a chemical reaction network to a selection of parameter perturbations. We analysed the artificial data using the MRA approaches as described in [1], [2], also supported by MATLAB.

The goal of the methods described in this Bachelor thesis, was to find a way to reconstruct a network. The MRA methods described in [1] and [2] only gave information how to find the Jacobian matrix corresponding to the network, but they did not describe how to reconstruct the network from these matrices. A mathematically rigourous reconstruction method needs to be developed in order to increase the applicability of the MRA methods in experimental settings.
The perturbations have to be chosen in the right way, such that the values for the metabolites corresponding to the perturbations do not change too much. This means one has to choose the perturbations 'smartly'.
In our set-up we tried to tune parameters such that the relative changes in the steady state values did not exceed circa 40% (see Appendix B.2 and B.3 for the simulated values for the steady state). These changes did improve the reconstructed sign structure of the Jacobian, as can be seen in (4.8) and (4.9). In Jacobian (4.9) are a few values for which the sign does not correspond with the sign in the computed Jacobian (4.6), but we were not yet able to identify the cause(s) of these deviations.

Further lines of research can be identified. We already mentioned the topic of network reconstruction from the Jacobian above. As mentioned in the introduction, we only consider simplified networks instead of the normal, complex network in which there may be 'hidden modules' connecting the observed nodes $x_1, \dots, x_n$. The methods described in Section 1 can be extended to the complex network. One therefore assumes the entire network to consist of n subsystems, each described by a set of differential equations,

$$\begin{cases} \dot{x}_j = g(x_1, \dots, x_n, \mathbf{y}_j, p_1, \dots, p_m) \\ \dot{\mathbf{y}}_j = h(x_1, \dots, x_n, \mathbf{y}_j, p_1, \dots, p_m), \end{cases} \tag{6.1}$$

where $j = 1, \dots, n$. Here the variables x_j represent the connecting nodes in subsystem j, and the vector variables $\mathbf{y}_j$ represent unknown nodes within subsystem j, see [1].
For this extended network, it can be researched how the MRA methods can be applied and how these methods determine the connection network between modules.

Since the methods described in Section 2.1 and Section 2.2 both give approximate systems of equations to the initial system of differential equation

$$\dot{x} = f(x, p), \tag{6.2}$$

it could be investigated how accurate these approximate equations are.
Also the accuracy of the SVD can be researched, since this decomposition returns an approximate solution to the systems derived in Sections 2.1 and 2.2.

In order to solve the homogeneous equation with the singular value decomposition, we used in section 3.3 a normalisation for F_i. This normalisation was

$||F_i||_2 = 1$, such that we have a constrained optimization problem.
This choice also has effect in the approximation of the Jacobian matrix in Section 4.6. This could be a reason why the approximated Jacobian matrix does not perfectly correspond with the computed Jacobian matrix.
This restriction and its influence on the determination of the Jacobian matrix can be researched.

The Jacobian matrices which follow from both MRA methods should be converted to a reaction network. Since an intuitive method for this reconstruction is not described in [1], [2] or further literature, this should be researched. Although we tried to give a method for reconstructing the graph representation and reaction network in Section 4.4, a mathematical proof was beyond the scope of this Bachelor thesis but is necessary in order to complete the MRA methods.

A Index of notation

Pyr	Pyruvate
Cit	Citrate
Mal	Malate
OAA	Oxaloacetate
NAD^+	Nicotinamide adenine dinucleotide
NADH	Reduced form of NAD^+
ADP	Adenosine diphosphate
ATP	Adenosine triphosphate
CO_2	Carbon dioxide
OAA_{cyt}	OAA in cytosol
Mal_{cyt}	Mal in cytosol
Cit_{cyt}	Cit in cytosol
$[NAD]_t$	Total concentration of NAD^+, $[NAD]_t = [NAD^+] + [NADH]$
f_j	Constant for exchange of Cit into Cit_{cyt}
k_i	Rate constant for mass action kinetics

B MATLAB codes for steady state method

The code used in section 4.6 to make the system of ODE's is as follows:

```
function dx=TCA1(t,x)

%x(1)=[Mal]
%x(2)=[Pyr]
%x(3)=[OAA]
%x(4)=[Cit]
%x(5)=[NAD^+]
%x(6)= reaction rate 1
%x(7)= reaction rate 2 (=k2+)
%x(8)= reaction rate 3 (=k2-)
%x(9)= reaction rate 4
%x(10)= reaction rate 5
%x(11)= c*[ADP]^3, c = reaction rate 6
%x(12)= reaction rate of ef(1)
%x(13)= reaction rate of ef(2)
%x(14)= reaction rate of ef(3)
%x(15)= reaction rate of ef(4)
%x(16)= [NAD]t (=[NAD^+]+[NADH])
%x(17)= reaction rate 7

dx = zeros(17,1);

ef1=x(12)*x(4); %ef(1)= exchange Cit + OAA_cyt -> OAA + Cit_cyt
ef2=x(13)*x(4); %ef(2)= exchange Cit + Mal_cyt -> Mal + Cit_cyt
ef3=x(14); %ef(3)= Mal_cyt -> Mal
ef4=x(15); %ef(4)= Pyr_cyt -> Pyr

v1= x(6)*x(1)*x(5); %v1= Mal+NAD^+ -> Pyr+NADH
v2= x(7)*x(1)*x(5)-x(8)*x(3)*(x(16)-x(5)); %v2= Mal+NAD^+ = OAA+NADH
v3= x(9)*x(2)*x(3)*x(5); %v3= Pyr+OAA+NAD^+ -> Cit+NADH
v4= x(10)*x(4)*(x(5))^2; %v4= Cit+ 2 NAD^+ -> Mal+ 2 NADH
v5= x(11)*(x(16) - x(5))^2; %v5= 2 NADH + 3 ADP -> 2 NAD^+ + 3 ATP
v6= x(17)*x(4); %v6= Cit -> 0

dx(1) = v4 - v1 - v2 + ef2 + ef3;
dx(2) = v1 - v3 + ef4;
dx(3) = v2 - v3 + ef1;
dx(4) = v3 - v4 - v6 - ef1 - ef2;
dx(5) = - v1 - v2 - v3 - 2*v4 + 2*v5;

end
```

B.1 Jacobian matrix

First we define the Jacobian matrix:

```
[T,A]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

x = [A(3600,1),A(3600,2),A(3600,3),A(3600,4),
     A(3600,5),0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0];

DF = [-x(6)*x(5)-x(7)*x(5) 0 x(8)*(x(16)-x(5)) x(10)*(x(5))^2+x(13)
        2*x(10)*x(4)-x(6)*x(1)-x(7)*x(1)-x(8)*x(3);
      x(6)*x(5) -x(9)*x(3)*x(5) -x(9)*x(2)*x(5) 0
        x(6)*x(1)-x(9)*x(2)*x(3);
      x(7)*x(5) -x(9)*x(3)*x(5) -x(8)*(x(16)-x(5))-x(9)*x(2)*x(5)
        x(12) x(7)*x(1)+x(8)*x(3)-x(9)*x(2)*x(3);
      0 x(9)*x(3)*x(5) x(9)*x(2)*x(5) -x(10)*(x(5))^2-x(17)-x(12)-x(13)
        x(9)*x(2)*x(3)-2*x(10)*x(4)*x(5);
      -x(6)*x(5)-x(7)*x(5) -x(9)*x(3)*x(5)
        x(8)*(x(16)-x(5))-x(9)*x(2)*x(5) -2*x(10)*(x(5))^2
        -x(6)*x(1)-x(7)*x(2)-x(8)*x(3)-x(9)*x(2)*x(3)-4*x(10)*x(4)*x(5)
            -4*x(11)*(x(16)-x(5))]
```

The function `ode45` is based on an explicit Runge-Kutta formula, so the time returned in the collumn vector T is not divided in steps of $t = 1$. This means we do not have to take $A(900)$ for $t = 900$, but the last value in the row vector A, in this case $A(3600)$.

B.2 First approximation of Jacobian matrix

For the MATLAB function `ode45` applied on the function `TCA1` with the initial values and admissible perturbations is as follows

```
[T,A]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

[T,B]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.3,0.2,2,0]);

[T,C]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.3,2,0]);

[T,D]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0.01]);

[T,E]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.2,0.2,0.2,2,0]);

[T,F]=ode45(@TCA1,[0 100],[10,10,10,10,1,
          0.1,0.2,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

[T,G]=ode45(@TCA1,[0 100],[10,10,10,10,1,
          0.1,0.1,0.2,0.5,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

[T,H]=ode45(@TCA1,[0 100],[10,10,15,10,1,
          0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);
```

From the new tables `A-H` I used the last values of the first five collumns which correspond to the steady state vlues of the metabolites. These values can be collected in 7 vectors, `xssp0-xssp7`, such that

```
xssp0 = [A(830,1) A(830,2) A(830,3) A(830,4) A(830,5)]
     %=[6.5413 4.8898 0.5866 0.9984 0.4485]
xssp1 = [B(830,1) B(830,2) B(830,3) B(830,4) B(830,5)]
     %=[13.1431 6.3707 0.7927 1.1983 0.3013]
xssp2 = [C(830,1) C(830,2) C(830,3) C(830,4) C(830,5)]
     %=[8.7639 8.0179 0.5455 1.2085 0.3522]
xssp3 = [D(830,1) D(830,2) D(830,3) D(830,4) D(830,5)]
     %=[5.7299 4.9441 0.5205 0.9319 0.4778]
xssp4 = [E(830,1) E(830,2) E(830,3) E(830,4) E(830,5)]
     %=[7.5497 7.8828 0.4045 0.8374 0.4022]
xssp5 = [F(830,1) F(830,2) F(830,3) F(830,4) F(830,5)]
     %=[10.2840 2.2421 1.8402 1.1402 0.3511]
xssp6 = [G(830,1) G(830,2) G(830,3) G(830,4) G(830,5)]
     %=[7.1240 3.9835 0.6156 1.0253 0.4295]
xssp7 = [H(830,1) H(830,2) H(830,3) H(830,4) H(830,5)]
     %=[9.5907 5.7073 0.6961 1.1285 0.3618]
```

From these vectors we can compute $(x_j^{ss}(p_k) - x_j^{ss}(p_0))_{k,j}$, which is collected in the (7×5) matrix

```
MA = [xssp1-xssp0;xssp2-xssp0;xssp3-xssp0;
      xssp4-xssp0;xssp5-xssp0;xssp6-xssp0;xssp7-xssp0];
```

The smaller matrices, i.e. the matrices consisting of the vectors corresponding to the admissible perturbations, are constructed as

```
A1 = [MA(2,:);MA(3,:);MA(6,:);MA(7,:)];
A2 = [MA(1,:);MA(3,:);MA(4,:);MA(5,:);MA(7,:)];
A3 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:)];
A4 = [MA(1,:);MA(2,:);MA(5,:);MA(7,:)];
A5 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:);MA(7,:)];
```

The SVD of these matrices, and from this the Jacobian matrix is

```
[V1,D1,U1] = svd(A1);
[V2,D2,U2] = svd(A2);
[V3,D3,U3] = svd(A3);
[V4,D4,U4] = svd(A4);
```

```
[V5,D5,U5] = svd(A5);

FT = [-U1(:,4) -U2(:,5) -U3(:,4) -U4(:,4) U5(:,5)];
F = FT'
```

B.3 Second approximation

For the second approximation of the Jacobian matrix, we used the following code

```
[T,A]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);
[T,B]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.21,0.2,2,0]);
[T,C]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.21,2,0]);
[T,D]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0.01]);
[T,E]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.11,0.2,0.2,2,0]);
[T,F]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.11,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);
[T,G]=ode45(@TCA1,[0 1000],[10,10,10,10,1,
            0.1,0.1,0.2,0.41,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);
[T,H]=ode45(@TCA1,[0 1000],[10,10,15,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

xssp0 = [A(3600,1) A(3600,2) A(3600,3) A(3600,4) A(3600,5)]
     %=[3.2189 4.5720 0.3521 0.7426 0.6215]
xssp1 = [B(3600,1) B(3600,2) B(3600,3) B(3600,4) B(3600,5)]
     %=[3.4791 4.4858 0.3786 0.7643 0.6036]
xssp2 = [C(3600,1) C(3600,2) C(3600,3) C(3600,4) C(3600,5)]
     %=[3.2735 4.8949 0.3428 0.7630 0.6111]
xssp3 = [D(3600,1) D(3600,2) D(3600,3) D(3600,4) D(3600,5)]
     %=[3.0563 4.9665 0.3135 0.7144 0.6312]
xssp4 = [E(3600,1) E(3600,2) E(3600,3) E(3600,4) E(3600,5)]
     %=[3.2359 4.8678 0.3324 0.7296 0.6182]
xssp5 = [F(3600,1) F(3600,2) F(3600,3) F(3600,4) F(3600,5)]
     %=[3.2189 3.7909 0.4246 0.7426 0.6214]
xssp6 = [G(3600,1) G(3600,2) G(3600,3) G(3600,4) G(3600,5)]
     %=[3.2189 4.4605 0.3521 0.7426 0.6215]
xssp7 = [H(3600,1) H(3600,2) H(3600,3) H(3600,4) H(3600,5)]
     %=[3.2189 4.5720 0.3520 0.7426 0.6214]

MA = [xssp1-xssp0;xssp2-xssp0;xssp3-xssp0;xssp4-xssp0;
      xssp5-xssp0;xssp6-xssp0;xssp7-xssp0];

A1 = [MA(2,:);MA(3,:);MA(6,:);MA(7,:)];
A2 = [MA(1,:);MA(3,:);MA(4,:);MA(5,:);MA(7,:)];
```

```
A3 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:)];
A4 = [MA(1,:);MA(2,:);MA(5,:);MA(7,:)];
A5 = [MA(1,:);MA(2,:);MA(3,:);MA(4,:);MA(7,:)];

[V1,D1,U1] = svd(A1);
[V2,D2,U2] = svd(A2);
[V3,D3,U3] = svd(A3);
[V4,D4,U4] = svd(A4);
[V5,D5,U5] = svd(A5);

FT = [U1(:,4) -U2(:,5) U3(:,4) -U4(:,4) U5(:,5)];
F = FT'
```

C Codes for timeseries method

The computed Jacobian matrix corresponding to the timeseries method is

```
[T,A]=ode45(@TCA1,[0 100],[10,10,10,10,1,
            0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0]);

x = [A(200,1),A(200,2),A(200,3),A(200,4),A(200,5),
     0.1,0.1,0.2,0.4,0.1,0.2,0.4,0.1,0.2,0.2,2,0];

DF = [-x(6)*x(5)-x(7)*x(5) 0 x(8)*(x(16)-x(5)) x(10)*(x(5))^2+x(13)
        2*x(10)*x(4)-x(6)*x(1)-x(7)*x(1)-x(8)*x(3);
      x(6)*x(5) -x(9)*x(3)*x(5) -x(9)*x(2)*x(5) 0
        x(6)*x(1)-x(9)*x(2)*x(3);
      x(7)*x(5) -x(9)*x(3)*x(5) -x(8)*(x(16)-x(5))-x(9)*x(2)*x(5)
        x(12) x(7)*x(1)+x(8)*x(3)-x(9)*x(2)*x(3);
      0 x(9)*x(3)*x(5) x(9)*x(2)*x(5) -x(10)*(x(5))^2-x(17)-x(12)-x(13)
        x(9)*x(2)*x(3)-2*x(10)*x(4)*x(5);
      -x(6)*x(5)-x(7)*x(5) -x(9)*x(3)*x(5)
        x(8)*(x(16)-x(5))-x(9)*x(2)*x(5) -2*x(10)*(x(5))^2
        -x(6)*x(1)-x(7)*x(2)-x(8)*x(3)-x(9)*x(2)*x(3)-4*x(10)*x(4)*x(5)
            -4*x(11)*(x(16)-x(5))]
```

As mentioned in Appendix B.1, we do not have to take `A(100)` for $t = 100$, because `ode45` divides the time in steps smaller then 1. This means $t = 100$ is not the same as `A(100)`.

First we define `tjmin`, `tj` and `tjplus` as follows

```
tjmin = 175;
tj = 200;
tjplus = 225;
```

The initial codes for this method are the same as de codes in section B.2, i.e. the functions `TCA1` and `ode45`. With these tables it is possible to make the matrix `S`:

```
x0 = [A(tj,1), A(tj,2), A(tj,3), A(tj,4), A(tj,5)];
   %=[21.5949 2.0628 2.7268 1.6711 0.3105]
x1 = [B(tj,1), B(tj,2), B(tj,3), B(tj,4), B(tj,5)];
   %=[22.5259 2.0954 2.7738 1.6969 0.3033]
x2 = [C(tj,1), C(tj,2), C(tj,3), C(tj,4), C(tj,5)];
   %=[21.8631 2.3653 2.6592 1.8015 0.2967]
x3 = [D(tj,1), D(tj,2), D(tj,3), D(tj,4), D(tj,5)];
   %=[21.3385 2.1227 2.6055 1.5907 0.3091]
x4 = [E(tj,1), E(tj,2), E(tj,3), E(tj,4), E(tj,5)];
   %=[22.9511 2.8167 1.9750 1.1575 0.2733]
x5 = [F(tj,1), F(tj,2), F(tj,3), F(tj,4), F(tj,5)];
   %=[20.7508 1.3792 4.6798 2.0722 0.3066]
x6 = [G(tj,1), G(tj,2), G(tj,3), G(tj,4), G(tj,5)];
   %=[21.6777 1.5217 3.1106 1.9190 0.3202]
x7 = [H(tj,1), H(tj,2), H(tj,3), H(tj,4), H(tj,5)];
   %=[24.2858 1.4916 4.1781 2.1568 0.3359]
x8 = [I(tj,1), I(tj,2), I(tj,3), I(tj,4), I(tj,5)];
   %=[22.1133 3.4327 3.2694 3.6700 0.2575]

S = [x1-x0; x2-x0; x3-x0; x4-x0; x5-x0; x6-x0; x7-x0; x8-x0];
```

We also need to compute `biT`. We fix `tj`, so we drop j from notation. First we have to compute `dki` for $i = 1, \ldots, 5$ and $k = 0, \ldots, 7$. These are (only shown for `di` for $i = 1$):

```
d01=0.5*((A(tjplus,1)-A(tj,1))/(tjplus-tj)+(A(tj,1)-A(tjmin,1))/(tj-tjmin));
d11=0.5*((B(tjplus,1)-B(tj,1))/(tjplus-tj)+(B(tj,1)-B(tjmin,1))/(tj-tjmin));
d21=0.5*((C(tjplus,1)-C(tj,1))/(tjplus-tj)+(C(tj,1)-C(tjmin,1))/(tj-tjmin));
d31=0.5*((D(tjplus,1)-D(tj,1))/(tjplus-tj)+(D(tj,1)-D(tjmin,1))/(tj-tjmin));
d41=0.5*((E(tjplus,1)-E(tj,1))/(tjplus-tj)+(E(tj,1)-E(tjmin,1))/(tj-tjmin));
d51=0.5*((F(tjplus,1)-F(tj,1))/(tjplus-tj)+(F(tj,1)-F(tjmin,1))/(tj-tjmin));
d61=0.5*((G(tjplus,1)-G(tj,1))/(tjplus-tj)+(G(tj,1)-G(tjmin,1))/(tj-tjmin));
d71=0.5*((H(tjplus,1)-H(tj,1))/(tjplus-tj)+(H(tj,1)-H(tjmin,1))/(tj-tjmin));
d81=0.5*((I(tjplus,1)-I(tj,1))/(tjplus-tj)+(I(tj,1)-I(tjmin,1))/(tj-tjmin));
```

Since we are only allowed to use the perturbations $p_k \in \mathcal{P}_i$, as in Table 4.5, we can define `bi`, and thus `biT` as follows:

```
b1 = [d21-d01 d31-d01 d61-d01 d71-d01 d81-d01];
b1T = b1';
```

We need these admissible perturbations also for the matrices `SiT`, which is:

```
S1 = [S(2,:)' S(3,:)' S(6,:)' S(7,:)' S(8,:)'];
S1T = S1';
```

From this we are able to find `FiT` as:

```
FiT=pinv(SiT)*biT
```

If we combine the vectors `FiT`, we are able to make `FT`, and thus `F`:

```
FT = [F1T F2T F3T F4T F5T];
F= FT'
```

For the second approximation, we only change the perturbations, which is the same code as mentioned in Appendix B.3. We also changed the differences between `tj`, `tjmin` and `tjplus`, since we need a value closer to the maximum value of Mal. This gives

```
tjmin = 185;
tj = 200;
tjplus = 215;
```

For `x0` – `x8` we have the following code:

```
x0 = [A(tj,1), A(tj,2), A(tj,3), A(tj,4), A(tj,5)];
   %=[21.5949 2.0628 2.7268 1.6711 0.3105]
x1 = [B(tj,1), B(tj,2), B(tj,3), B(tj,4), B(tj,5)];
   %=[21.6889 2.0661 2.7316 1.6737 0.3098]
x2 = [C(tj,1), C(tj,2), C(tj,3), C(tj,4), C(tj,5)];
   %=[21.6231 2.0921 2.7197 1.6841 0.3091]
x3 = [D(tj,1), D(tj,2), D(tj,3), D(tj,4), D(tj,5)];
   %=[21.3385 2.1227 2.6055 1.5907 0.3091]
x4 = [E(tj,1), E(tj,2), E(tj,3), E(tj,4), E(tj,5)];
   %=[21.7662 2.1389 2.6311 1.6064 0.3057]
x5 = [F(tj,1), F(tj,2), F(tj,3), F(tj,4), F(tj,5)];
   %=[21.5205 1.9474 2.9507 1.7163 0.3103]
x6 = [G(tj,1), G(tj,2), G(tj,3), G(tj,4), G(tj,5)];
   %=[21.6116 1.9977 2.7630 1.6938 0.3113]
x7 = [H(tj,1), H(tj,2), H(tj,3), H(tj,4), H(tj,5)];
   %=[22.1809 1.9267 2.9800 1.7557 0.3145]
x8 = [I(tj,1), I(tj,2), I(tj,3), I(tj,4), I(tj,5)];
   %=[21.9114 2.0519 2.8980 1.8661 0.3099]
```

The rest of the code is the same as for the first approximation for the time series data method.

References

[1] E.D. Sontag (2008), Network reconstruction based on steady-state data, *Essays in Biochemistry* **45**, 161–176

[2] E.D. Sontag, A. Kiyatkin, and B.N. Kholodenko (2004), Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data, *Bioinformatics* **20** (12), 1877–1886

[3] H. Dym (2007), *Linear Algebra in Action*, Graduate Studies in Mathematics, Volume 78, AMS

[4] R. Steuer (2007), Computational approaches to the topology, stability and dynamics of metabolic networks, *Phytochemistry* **68**, 2139–2151

[5] R. Steuer, A. Nunes Nesi, A.R. Fernie, Th. Gross, B. Blasius and J. Selbig (2007), From structure to dynamics of metabolic pathways: application to the plant mitochondrial TCA cycle, *Bioinformatics* **23** (11), 1378–1385. doi: 10.1093/bioinformatics/btm065