INSTITUTE OF CHEMICAL TECHNOLOGY, PRAGUE

Faculty of Chemical Engineering

Department of Chemical Engineering

MASTER THESIS

Development and application of computational methods for decomposition of large reaction networks and determination of their stability

Author: Boris Fačkovec

Supervisor: prof. Ing. Igor Schreiber, CSc.

Consultant: Ing. Otto Hadač, PhD.

Study program: Process Engineering and Informatics

Study subprogram: Chemical Engineering, Bioengineering and Mathematical Modeling

Year: 2012

replace this page with thesis assignement (received from your supervisor)

This thesis/dissertation was written at the Department of Chemical Engineering of the Institute of Chemical Technology in Prague between October 2010 and August 2012.

I hereby declare that this thesis is my own work. Where other sources of information have been used, they have been acknowledged and referenced in the list of used literature and other sources.
I have been informed that the rights and obligations implied by Act No. 121/2000 Coll. on Copyright, Rights Related to Copyright and on the Amendment of Certain Laws (Copyright Act) apply to my work. In particular, I am aware of the fact that the Institute of Chemical Technology in Prague has the right to sign a license agreement for use of this work as school work under §60 paragraph 1 of the Copyright Act. I have also been informed that in the case that this work will be used by myself or that a license will be granted for its usage by another entity, the Institute of Chemical Technology in Prague is entitled to require from me a reasonable contribution to cover the costs incurred in the creation of the work, according to the circumstances up to the full amount.
I agree to the publication of my work in accordance with Act No. 111/1998 Coll. on Higher Education and the amendment of related laws (Higher Education Act).

In Prague on August 17 2012

SUMMARY

Complex reaction networks (CRN) prove to be excellent models for complex chemical systems like chemical reactors or cellular compartments. Dynamics of the modeled system can be studied qualitatively using graph theoretical methods and convex analysis. In this work, methodology for qualitative analysis of CRN is developed and implemented in MATLAB/Octave. Emphasis is put on comfort of the user and developer, i.e. on straightforward usage and lucidity of the code. Program for decomposition of a network into extreme pathways and for determination of their stability is implemented based on literature. Algorithm for automatic classification of potential oscillators is invented and an efficient genetic algorithm for finding Hopf bifurcation is proposed. The developed software is used to analyze 5 representative relevant CRN models.

ACKNOWLEDGEMENT

I would like to thank to my supervisor prof. Igor Schreiber for introducing me into the topic and for devoted guidance and help. My work would be unthinkable without personal support of my beloved girlfriend.

Contents

1 INTRODUCTION

Reaction networks are familiar to all chemists. If the size or structure of a studied network gives rise to complexity beyond the limits of chemical intuition, a rigorous and systematic approach has to be followed. Size of such "too complex" systems can be surprisingly small. Simple Oregonator model comprising only 5 reactions and 3 species can exhibit interesting dynamic properties. Belousov-Zhabotinsky reaction, which can be well modeled by a network comprising 12 reactions [1] used to puzzle many chemists and physicists for a long time. For its challenging underlying theory the field has been for years attracting mathematicians and physical chemists.

For chemical engineering applications, knowledge of chemical systems dynamics is of great importance. Based on a good mechanism, operation conditions for an industrial reaction can be set in order to optimize some desirable properties, such as maximum yield or minimum production time. Investigation of steady states of isothermal reactors and their stabilities is an important step in reactor design. Reaction networks can very well describe even heterogeneous reactions and continuous flow can be modeled by pseudoreactions.

In molecular biology, the advent of high throughput technologies has caused shift from reductionist dissection to systems integration in the past ten years. Methods of qualitative analysis of metabolic networks have been becoming increasingly popular among biochemical community. Metabolic networks present one level in hierarchy of understanding life. Together with protein and RNA folding and binding, they stand for paradigm shift from machines operating according to well known rules to living creatures with "elan vital". Therefore, it is plausible to assume that in near future, the field of biochemical network analyses will flourish.

The main contribution of this work is a software pushing the limits of thorough qualitative analyses of complex reaction networks to models composed of tens of reactions by automatization of the data analysis process. Analyses of selected models are shown to illustrate capabilities of the developed software. Therefore, the most worked part of this publication is documentation to the programs written so that it should be easy to use and modify. My hope is that the outcomes of this work will prove useful to investigators searching for simple useful code.

2 THEORETICAL PART

2.1 Chemical Reaction Networks

Chemical reaction networks are simplified discretized models of complex chemical processes. They rely on 3 levels of approximations. First, Born-Oppenheimer approximation must be made to introduce the concept of potential energy surface (PES). Second, chemical species are loosely defined as regions on PES separated from other species (regions on PES) by a barrier in the magnitudes of tens to hundreds kcal/mol. Uncountable number of structures slightly differing in soft degrees of freedom [1] are represented by single species with its unique chemical formula. Transition states are represented by saddle points on PES dividing regions corresponding to species and chemical reactions are bundles of paths connecting these regions.

The third level of approximations is deciding which species and reactions to consider in our CRN model. The number of species grows exponentially with the number of comprising atoms and the number of all possible reactions grows approximately with square of the number of species. It must be decided, which intermediates to include and which chemical reactions are This level of approximation is in chemistry called constructing the reaction mechanism.

It is worth to mention here that since paths on PES are not oriented curves, all reactions are in principle reversible. However, if free energy of products is lower than that of reactants by decades of kJ/mol, backwards reaction can be negligibly slow, since energy barrier is increased by that value. At room temperature, free energy difference of about 6 kJ/mol is equivalent to about 10-fold decrease in reaction rate. Therefore, reactions can be considered as oriented paths from reactants to products, which has significant implication for their modeling.

These approximations lead to description of a real chemical systems by system of ordinary differential equations (SODE).

$$\dot{x}_i = f_i(x_1, x_2, ...);\ i = 1..m \tag{1}$$

where m in number of species, x_i is concentration of ith species. f_i is function of concentrations, which is generally non-linear and can be written as a sum of contributions of all reactions which species i participates.

$$f_i = \sum_{j=1}^{r} \nu_{ij} v_j \tag{2}$$

where ν_{ij} is stechiometric coefficient os species i in jth reaction, r is number of reactions and v_j is rate of jth reaction defined by kinetic equations. If the model

[1] is usually the term for bond angles and torsional angles. In the context of this work, all changes of internal coordinates not causing breakage of any covalent bond.

consists of elementary reactions, the kinetic equations are monomial, i.e. they have the form

$$v_j = \prod_{k=1}^{m} k_k \; x_k^{\nu_{kj}^L} \tag{3}$$

where k_i is rate constant independent of concentrations of species and ν_{ij}^L is order of reaction j with respect to species i. So that SODE has the form

$$f_i = \sum_{j=1}^{r} \nu_{ij} \prod_{k=1}^{m} k_k \; x_k^{\nu_{kj}^L} \tag{4}$$

Such kinetics is also called power law kinetics.

Evolution of system (1) can be computationally studied, so that dynamical behavior of the system can be predicted and engineered. There are multiple levels of studying dynamics of CRNs. First, one desired trajectory in phase space can be evolved by numerical integration of the equations using for example Runge-Kutta method or a suitable predictor-corrector scheme. Second, it has been shown that stochastic modeling can be sometimes considerably more efficient [2]. Third, more general approach is a qualitative analysis using bifurcation diagrams. Finally, it has been shown that CRNs possess some inherent properties following from the network topology, so CRNs can be qualitatively studied without the knowledge of rate constants. For this purpose, graph theory and complex analysis proved useful.

From mathematical point of view, CRN is an oriented weighted hypergraph[2]. Various representations of such structure have been proposed, for example the species-reaciton graph or directed bipartite graph [3]. Another representation is by stoichiometric matrix $\boldsymbol{\nu}$, which contains overall stoichiometric coefficient of species i in jth reaction in field $\boldsymbol{\nu}_{\mathrm{ij}}$, so it is integer matrix for network of elementary reactions. Construction of such matrix obviously comes with loss of kinetic information, because a catalyst species is omitted. Therefore, stoichiometric matrix must be complemented with a kinetic matrix defined as

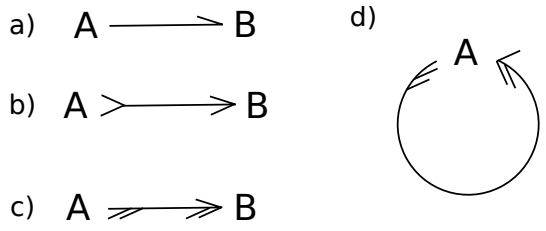
$$\kappa_{ij} = \frac{\partial \log v_j}{\partial \log x_i} \tag{5}$$

to represent the system. If non-multiplied elementary reactions are considered, kinetic matrix is identical to left stoichiometric matrix $\boldsymbol{\nu}^{\mathrm{L}}$, which is (in case of elementary reacti7ons) a positive integer matrix having absolute value of stoichiometric coefficient of reactant i in jth reaction in field $\boldsymbol{\nu}_{\mathrm{ij}}^{\mathrm{L}}$. Analogically, there is a right stoichiometric matrix $\boldsymbol{\nu}^{\mathrm{R}}$. Stoichiometric matrix is simply their difference $\boldsymbol{\nu} = \boldsymbol{\nu}^{\mathrm{R}} - \boldsymbol{\nu}^{\mathrm{L}}$.

CRN can be portrayed in a network diagram like one in the Figure 2.1. Each species is represented by its formula and each reaction by a branched arrow marking from

[2]hypergraph = generalization of a graph, edges connect more than 2 vertices

Figure 1: Diagrammatic representation of chemical reaction networks - illustration by simple examples. a) A → B b) 2 (A → B) c) 2 A → 3 B d) 2 A → 3 A



reactants to products. Barbs on the heads and feathers on the tails of arrows represent kinetics and stoichiometry. Convention fo arrow heads is straightforward; the number of barbs is equal to coefficient of the product in right stoichiometric matrix. Convention for arrow tails is a little more complicated, since left stoichiometric matrix is generally not equal to kinetic matrix This is beacuse in current diagrams, it is convenient to multiply reactions, for example 2 (A → B), which is not the same as 2 A → 2 B. The former denotes two reactions with first order kinetics, while the latter denotes one reaction with second order kinetics with respect to A. Therefore, arrows on the left side are reserved for kinetics and Examples of reactions and their diagrammatic representation are in Figure 2.1. If stoichiometric coefficient of a reactant is 1, as well as its coefficient in kinetic equation, there is an exception allowed - no feather is necessary. Such approach can cope with integer kinetics only and proves useful in diagrams of extremal pathways to indicate that some reactions are used multiple times. In most diagrams in this thesis, the convention for left and right side of arrow tails is ignored unless stated otherwise.

Because of appropriateness of the aforementioned approximations, the SODE induced by a CRN can very well describe the dynamics of the represented chemical system. Their suitability for modeling metabolic pathways has been recognized by biochemical community. Many reviews of network analyses for biochemists have been published in the last decade [4–6]

2.2 Stoichiometric Network Analysis

Stoichiometric network analysis (SNA) is a qualitative approach to studying dynamics behavior of complex reaction networks (CRNs) or other systems with stoichiometry. Stability of steady states can be examined with no knowledge of rate constants of constituent equations. The methodology was established by Bruce L. Clarke in 80's [7]. Since that time, it has been successfully applied to a variety of useful models, for example see work by our group [8]. The first step is finding the basis of manifold of stationary states in reaction rate space called current cone[3]. This set of reaction rate

[3]reaction rate vector in analogy with Kirchhoff law called current

vectors $\mathbf{e}$ has to satisfy two conditions.

1. All $\mathbf{e}$ have to lie in null space of stoichiometric matrix $\boldsymbol{\nu}$

$$\boldsymbol{\nu}\ \mathbf{e} = \mathbf{0} \tag{6}$$

2. All $\mathbf{e}$ lie in positive orthant[4] of the reaction rate space.

The second condition is the consequence of orientation of reactions from reactants to products and physical constraints on concentrations, which must be positive. These two conditions result in positive definiteness of reaction rate vectors.

Elements of convex basis (or simply edges) of current cone are called extreme currents or extreme pathways. In the following text, the latter term will be preferentially used to emphasize graph theoretical approach to the problem. An another concept providing intuitive insight into meaning of extreme pahtways are elementary flux modes. An elementary flux mode is element of basis for all subsets of the CRN hypergraph in which no species is in sum consumed or produced. Set of extreme pathways is a subset of set of elementary modes [9]. Such subsets are easy to imagine and picture. For example, each reversible reaction represents 1 extreme pathway.

Around each steady state $\mathbf{x}_0$, SODE (Equation 1) can be linearized using Taylor expansion. In vector notation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x} - \mathbf{x}_0) \tag{7}$$

where $\mathbf{f}(\mathbf{x}_0) = \mathbf{0}$ is a steady state condition and $\mathbf{J}$ is Jacobian matrix defined as

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{8}$$

It can be shown that Jacobian matrix of the system can be written as

$$\mathbf{J} = \boldsymbol{\nu}\ \mathrm{diag}(\mathbf{e})\ \boldsymbol{\kappa}|_{\mathbf{x}_0}^{\mathrm{T}}\ \left(\mathrm{diag}(\mathbf{x}_0)\right)^{-1} \tag{9}$$

where $\boldsymbol{\nu}$ is stoichiometric matrix, $\mathbf{e}$ is an element of current cone, $\mathbf{x}_0$ is concentration vector and $\boldsymbol{\kappa}|_{\mathbf{x}_0}$ is kinetic matrix evaluated in the steady state $\mathbf{x}_0$.

$$(\kappa|_{\mathbf{x}_0})_{ij} = \frac{\partial\ \log\ v_j(\mathbf{x}_0)}{\partial\ \log\ x_i} \tag{10}$$

where $v_j(\mathbf{x}_0)$ is jth equation in the setady state. If a CRN follows power law kinetics, kinetic matrix is equal to left stoichiometric matrix. Then the product can be separated into two parts, first of which does depend only on stoichiometry

$$\mathbf{B} = -\boldsymbol{\nu}\ \mathrm{diag}(\mathbf{e})\ \boldsymbol{\kappa}^{\mathrm{T}} \tag{11}$$

[4]orthant = generalization of quadrant to more dimensions

and the second one $((\mathrm{diag}(\mathbf{x}_0))^{-1})$ representing a particular steady state.

A steady state is stable if all the eigenvalues of Jacobian matrix evaluated in this steady state have negative real values. Clarke showed that this is true if none of the principal subdeterminants of $\mathbf{B}$ [5] is negative. Indices determining the subdeterminant correspond to species playing key roles in destabilization of the network $\mathbf{e}$. These species will be referred to as deteminant-indicated.

2.3 Classification of Chemical Oscillators

Theoretical approach described in the previous chapter leads to list of extreme pathways with their determinant indicated metabolites if unstable. If the system can oscillate, categorization of this oscillator leads to useful implications for its dynamics. Considerable amount of work has been done to systematize chemical oscillators [10–14] integrating various approaches, such as in situ dynamics observations, bifurcation analyses [15], network diagram analyses and stability analysis described above. Such systematization helps to experimentally determine their mechanism and to predict their dynamical properties.

Clarke [7] classified current cycles in chemical reaction networks as strong, critical or weak if the principal subdeterminant of matrix $\mathbf{B}$ defined by the species forming the cycle is negative, zero or positive respectively. In strong, critical and weak cycles, output reaction is of lower, equal and higher order than the cycle. In seminal paper by Eiswirth et al., four qualitatively different cases of unstable networks are distinguished.

1. networks that contain a critical current cycle and a suitable destabilizing reaction
2. networks that contain a strong current cycle
3. autocatalytic ring networks
4. others, yet undescribed

The first case gives rise to category 1 of chemical oscillators and the second case to category 2. Category 1 is subdivided into 3 classes, 1B and 1C, wich is subdivided into 1CX and 1CW. Category 2 can be also divided into 2B and 2C subcategories. The distinction between B and C subcategories is particularly important. Since 1C and 2C subcategory oscillators involve input feedback, they are crucially dependent on inflow. 1B and 2B subcategory involve output feedback and therefore can oscillate in batch mode. Therefore the symbols B (batch) and C (continuous).

[5] principal subdeterminant of matrix $\mathbf{B}$ of size m defined by vector $\mathbf{v}$ is a determinant of matrix $\mathbf{C}$ constructed such that $C_{ij} = B_{v_i v_j}$. Ina MATLAB notation `C = B(v,v)` . The number of such subdeterminants is combination number $\begin{pmatrix} n \\ m \end{pmatrix}$, where n is size of matrix $\mathbf{B}$.

Figure 2: Prototypes of network diagrams of models of category 1. LEFT: Category 1B. RIGHT: Category 1CX. Adapted from ref. [15].

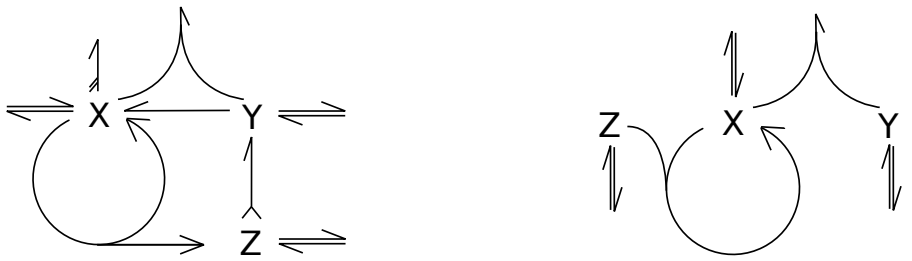
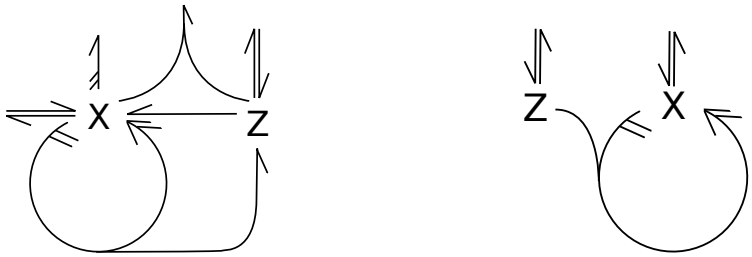


Figure 3: Prototypes of network diagrams of models of category 2. LEFT: Category 2B. RIGHT: Category 2C. 2 feathers do not stand for stoichiometric coefficient 2. Adapted from ref. [15].



In the oscillators, some species play special roles. The main classification of species distinguishes essential and non-essential species. Their definition follows from properties of Jacobian matrix and is discussed in ref [10]. Species have also symbols characterizing their roles in the oscillator. Autocatalytic species X forms the autocatalytic cycle. There might be many X species on a cycle. Exit species Y reacts with X, so it decreases its concentration. Recovery species W found in oscillators of category 1CW is formed by exit reaction and reacts with Y. Species of type X, Y and W are indicated by determinant while feedback species Z is not. Phase relations of species of type X, Y and Z have been described [14].

3 COMPUTATIONAL PART

3.1 Implementation

3.1.1 Stoichiometric Matrix Converter

`react2mat.py` , written in python, presents just a user-friendly interface for construction of stoichiometric matrices. The script was developed in order to avoid tedious work susceptible to human error. Instead of careful filling fields in a table editor (e.g. gnumeric) user writes down the equations in human readable form. Guidelines for input file are briefly described in the source code. Output of the script are

- stoichiometric matrix
- left stoichiometric matrix
- right stoichiometric matrix
- kinetic matrix if it is identical to left stoichiometric matrix

The script also creates log file, where all the metabolites and reactions are numbered.

3.1.2 Decomposition into Extreme Pathways

Decomposition of reaction network into basis is the first step of stoichiometric network analysis. Extremal pathways are here calculated by program based on algorithm proposed by Schilling and Palsson [16]. The original implementation takes into account exchange fluxes, which are not used in our analyses and therefore are not implemented in our version.

The iterative algorithm follows the principles of algorithms for finding the extremal generating vectors of convex polyhedral cones. In the first step of each iteration, the algorithm creates a temporary matrix by combining all temporary edges[6] having positive number in specified fields with those having negative numbers in those fields. This matrix is then substantially reduced in the second step by identification of redundant temporary edges by pairwise comparison. A temporary edge is considered redundant if its indices of zero fields are subset of indices of zero fields of any another temporary edge. Because of the second step, the original algorithm scales with square of the size of matrix of temporary edges with redundant ones. The total number of iterations is equal to the number of metabolites.

The program is implemented as MATLAB/Octave function and takes only stoichiometric matrix on input and returns matrix of extremal pathways sorted according to number of reactions. The algorithm was implemented with two notable enhancements.

[6] In the last iteration, edges are obtained by deletion of part of these vectors. Therefore, they are "almost" edges or in this thesis referred to as temporary edges.

First, iterative performing of the second step (check of temporary edges for redundancy) reduces formal scaling to linear with respect to the number of temporary edges with redundant ones. Vectors are randomly divided into bins and checked for redundancy inside If the number of identified redundant vectors is higher than a predefined tolerance, anothe iteration is performed. The procedure has two parameters - bin size and the number of found redundant vectors after which the iterative procedure ends. Performing the procedure until zero new redundant vectors are found is meaningless, since it does not ensure that no such vector is present in the temporary edge matrix. User does not need to enter these parameters, default values seem to perform well for systems comprising tens of reactions.

Second, memory is saved by storing the matrix as sparse matrix. This is done very easily in MATLAB language, so lucidity of the code was not negatively influenced.

3.1.3 Stability of Extreme Pathways

Implementation of network stability analysis is straightforward. Since the number of principal subdeterminants can be prohibitively large, any procedure resulting in their decrease is welcome. A simple time-saving procedure was implemented. It uses advantage of the fact that the matrix **B** defined by Equation 11 usually contains many zero columns and rows. If ith column AND ith row are zero for any index i, these are not considered when principal subdeterminants are constructed. This modification decreases time spent on calculation of stability of one extreme pathway from `m!` to `(m-z)!`, where `m` is number of metabolites and `z` is numbed of indices i satisfying the aforestated condition.

3.1.4 Classification of Potential Chemical Oscillators

Unlike the previous 2 programs, this program is original contribution of this work; to our knowledge, there is no software of that kind. Need for automatized classification of unstable steady states is suggested in the previous chapters. We are interested only in categories 1 and 2 and their subcategories B and C. We naturally cannot study "other" yet undescribed cases. Admittedly, there have been notions of new prototypes of oscillators since the classification was published [17]. However, relationship between their network topology and dynamic properties have not been studied yet, so classification would be pointless at this stage.

In order to algoritmize classification, unambiguous conditions for classification must be formulated. In our program, oscillators of categories 1B, 1C, 2B and 2C were classified based on conditions following from network topology of their prototypes and their roles in stability analysis (chapter 2.2). Skeleton prototypes of particular categories of potential oscillators are in Figures 3.1.4 and 3.1.4.

1. Species is indicated by determinant iff it is either of type X or of type Y or of

type W[7].

2. X species form a strong cycle or a critical cycle with exit reaction with Y species. Remaining determinant indicated species (if any) must be of type W.

3. The cycle can branch, but all species involved in the cycle from which category is derived are of type X.

4. In networks of category 1B, Y species cannot be produced directly by autocatalytic cycle. At least one Z species must be between. Maximum number of Z species connecting Y species (or just exit reaction in case of 2B category) with autocatalytic cycle is not defined.

5. Only one strong cycle or a critical cycle with suitable exit reaction is present.

Figure 4: Prototypes of potential oscillators of category 1 used in our classification algorithm. LEFT: category 1B. RIGHT: category 1C.

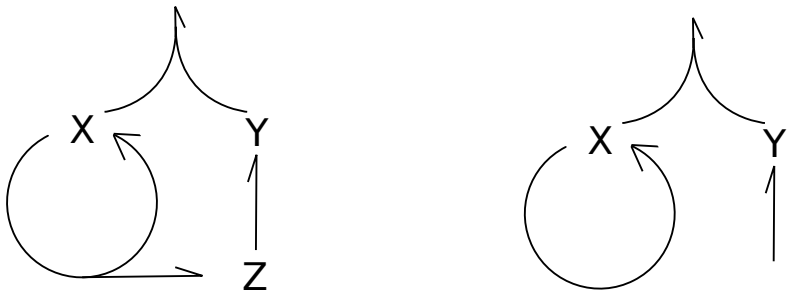
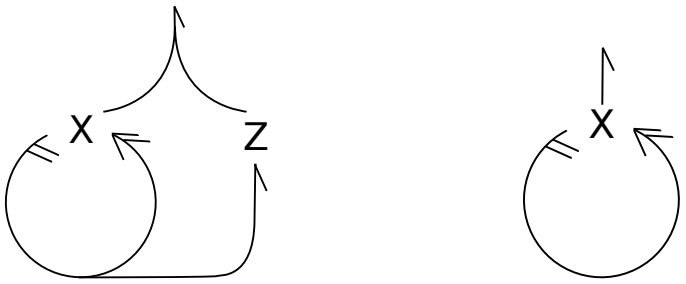


Figure 5: Prototypes of potential oscillators of category 2 used in our classification algorithm. LEFT: category 2B. RIGHT: category 2C.



The function takes an unstable steady state, set of determinant indicated species, kinetic and left and right stoichiometric matrices on input. In order to assign this input set a category, it performs the following operations.

[7]Definitions of species in prototype subnetwork is given in ref. [14].

1. Identify all the cycles composed of determinant indicated species.

2. If there is more than 1 cycle, determine (for each pair of cycles) whether there are links between them. If yes, merge the into 1 cycle.

3. Calculate strength of the cycles. If there is a strong and a critical cycle, consider strong one for following calculations. If there are more cycles of the same strength, choose any of these cycles and raise warning (according to assumption 5, only one cycle should be present).

4. If there are two or more critical cycles and no strong cycles, check for suitable exit reaction. Omit those without exit reaction with an another determinant indicated species. If more fulfiling the condition rising from assumption 5, raise warning.

5. Identify Y and/or Z species. There should be only 1 Y species in an oscillator of category 1 and no Y species in category 2.

It is entitled classification of potential oscillators because sole instability is not sufficient for oscillatory behavior. Some extreme pathways may involve the prototypes shown in Figures 3.1.4 and 3.1.4 and do not oscillate in the same time.

3.1.5 Identification of Hopf Bifurcations

Instability of an extremal pathway does not necessarily imply oscillatory behavior. Supercritical Hopf bifurcation is a strong indication of possibility of ocillations. To our knowledge there has been no method of determining whether system can under some conditions undergo Hopf bifurcation solely from network diagram yet. Therefore, concentration vector fulfiling conditions of Hopf bifurcations has to be found. Another original contribution of this work is an algorithm finding such vector in a reasonable time. The heuristic algorithm can positively answer the question, whether the system can exhibit supercritical Hopf bifurcation. Failure to converge in finite time does not ensure that the extremal pathway cannot oscillate under any conditions.

Random search in concentration manifold can be extremely time consuming because the region of Hopf bifurcations can be very small. Therefore, a fitness function guiding the search to the right direction has to be developed. The fitness function results from the following requirements for the creation of system of closed orbits in the phase space According to Hopf's theorem, there are another 2 conditions for creation of an isolated system of closed orbits in phase space. However, these seem to be usually fulfilled in most chemical systems[8].

[8]Of those asymptotical stability of the steady state is worth to mention. For more theory of bifurcations, see textbook [18].

1. Concentrations of essential species need to be smaller than those of other species.

2. Jacobian matrix of the system needs to have one pair of imaginary eigenvalues.

3. The rest of the eigenvalues of Jacobi matrix need to have negative real parts.

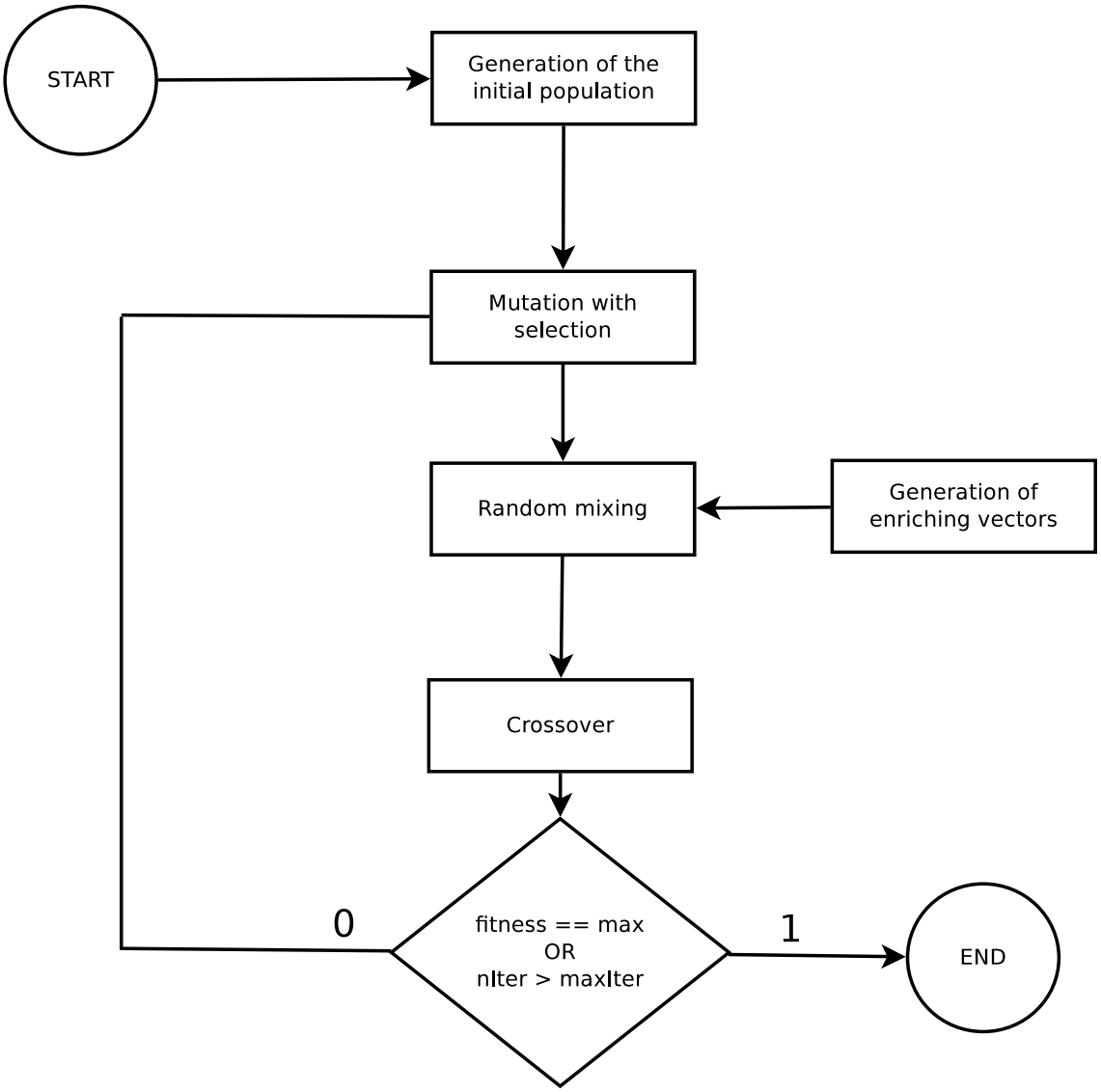
The fitness function is composed of 3 contributory functions. First, absence of complex eigenvalues is penalized. If the Jacobi matrix corresponding to evaluated concentration vector has no complex eigenvalues, a contribution to fitness function is calculated based on likelihood of the vector to acquire complex eigenvalues when changed a little. The function uses advantage of the fact that a pair of roots must become become equal before it becomes complex. Therefore, minimizing separation of the closest pair of roots leads to the desired result. This contribution smoothly (exponential function) depends on this minimum separation of 2 closest eigenvalues. Second, if there are complex eigenvalues, positive real part of any other eigenvalue $Re(q)$ is penalized by its exponential $-e^{Re(q)}$. The penalization is sum of penalizations for all positive eigenvalues smoothly increasing with the value of the corresponding eigenvalue. Third, real parts of complex eigenvalues needs to be close to 0. This last contribution depends on ratios of imaginary and real parts of complex eigenvalues.

The aforedescribed fitness function is non-linear and based on 3 non-related conditions which can cause in local minima, discontinuities and traps. Therefore, a robust algorithm converging to global minimum has to be used. We use genetic (differential evolution) algorithm. Evolution diagram of the algorithm is in Figure 3.1.5.

First, vectors are randomly generated so that concentrations of essential species is lower than those of the other species. The same subroutine is later used for generating new vectors enriching genofond of the evolving population. Then in each generation, each vector is exposed to mutations and based on fitness function, it is decided between the original vector and its mutant. The resulting set of vectors is enriched with newly generated random vectors and randomly mixed (crossover). The crossover is performed as random linear combinations of randomly chosen pairs of vectors. Offspring is sorted according the fitness function and the highest scoring individuals are selected for the next step. The algorithm terminates when it enters the region of concentration vectors fulfilling the conditions.

Program is implemented as MATLAB/Octave function. The algorithm has 18 parameters, which can be easily modified by the user via function arguments. Documentation for the program can be found in Appendix E.

Figure 6: Evolution algorithm of algorithm for finding Hopf bifurcation point.



3.2 Mitogen-Activated Protein Kinase Cascade

Mitogen Activated Protein Kinase cascade is a biochemical network involved in directing cellular responses to a wide range of biochemical and physiological stimuli. It seems to be universally present in all eukaryotic cells and plays an important role in cell cycle regulation, regulation of gene expression, apoptosis and many others. Its dynamics has been extensively studied mainly theoretically [19–21].

Table 1: Model reactions of MAPK cascade. Abbreviations are used as follows. A - MAPK kinase kinase, A* - activated MAPK kinase kinase, B - MAPK kinase, B1 - phosphorylated MAPK kinase, B2 - double phosphorylated MAPK kinase, C - MAPK, C1 - phosphorylated MAPK, C2 - doubly phosphorylated MAPK, E1 and E2 are enzymes activatgin and deactivating MAPK kinase kinase respectively, E3 - MAPK kinase phosphorylase, E4 - MAPK phosphorylase. Complex of any of the enzymes E with any of the substrates S is denoted simply ES.

1	A + E1 → E1A	11	A*B1 → B1 + A*	21	B2C → C1 + B2
2	E1A → A + E1	12	A*B1 → B2 + A*	22	C1 + B2 → B2C1
3	E1A → A* + E1	13	B2 + E3 → E3B2	23	B2C1 → C1 + B2
4	A* + E2 → E2A*	14	E3B2 → B2 + E3	24	B2C1 → C2 + B2
5	E2A* → A* + E2	15	E3B2 → B1 + E3	25	C2 + E4 → E4C2
6	E2A* → A + E2	16	B1 + E3 → E3B1	26	E4C2 → C2 + E4
7	B + A* → A*B	17	E3B1 → B1 + E3	27	E4C2 → C1 + E4
8	A*B → B + A*	18	E3B1 → B + E3	28	C1 + E4 → E4C1
9	A*B → A* + B1	19	C + B2 → B2C	29	E4C1 → C1 + E4
10	B1 + A* → A*B1	20	B2C → C + B2	30	E4C1 → C + E4

The first mechanism was proposed by Huang and Ferrell [22]. 10 biochemical transformations (Table 3.5) can be described using 30 elementary reactions. In the following text, abbreviated names for metabolites and enzymes are used. They are described in caption of Table 3.5. The reactions occur in cellular compartments which can be approximated by well mixed batch reactors. Therefore, no pseudoreactions were added and all metabolites were used in balancing.

The system can be decomposed into 15 extremal pathways. They can be clasified as of 2 types (see Figure 3.2). First, there are 10 reversible reactions in the system (type 1). Second, the cascade can be divided into 5 cycles (type 2). This is more obvious from original biochemical diagram which can be found for example in ref. [20]. In each such cycle (extremal pathway of type 2, Figure 3.2), protein is phosphorylated (or activated in case of MAPKKK) in a reaction catalyzed by one enzyme and then turned back by an another enzyme. Stability analysis showed all the 15 extremal pathways to be stable.

Figure 7: Current diagram of MAPK system. The cascade comprises 3 subnetworks being connected always through 1 metabolite, which plays role of an enzyme in the second subnetwork.

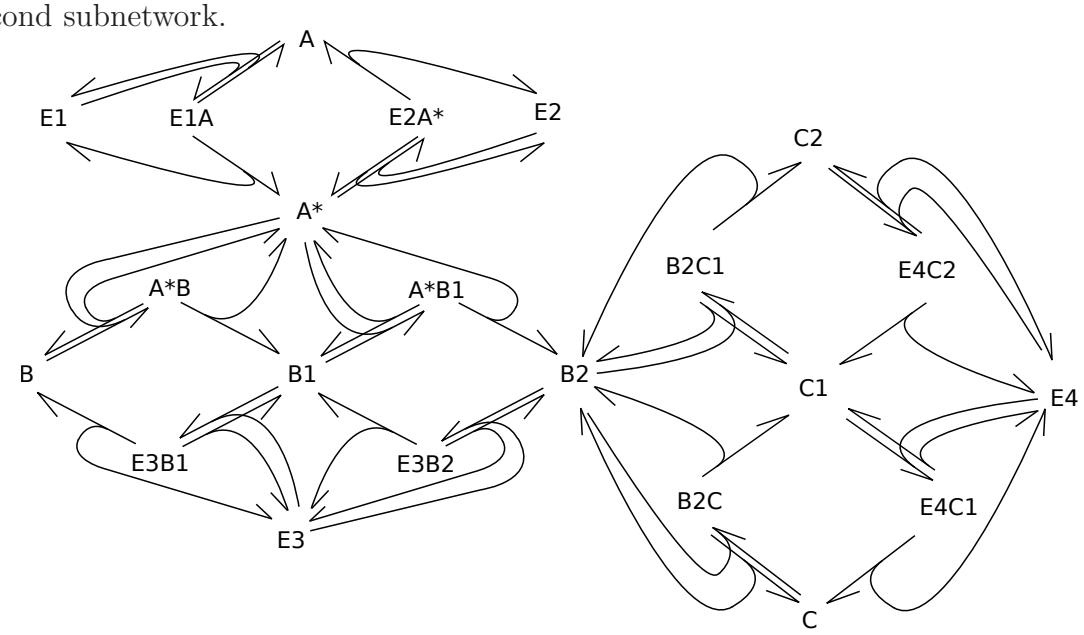
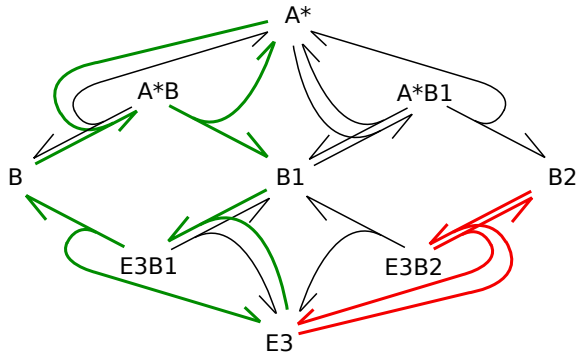


Figure 8: Two types of extremal pathways. There are 10 extremal pathaways of type 1 (red) and 5 extremal pathways of type 2 (green).



Size of the system allowed rapid stability analysis of all pairs of extremal pathways. Of all such combinations, 2 were found to be unstable, each having 2 different sources of instability, i.e. 2 different sets of determinant-indicated metabolites. None of these 4 subsystems possesses a critical or strong cycle, therefore no category of potential oscillators (section 2.3) suits to the systems.

Possibility of ocillations was examined employing the genetic algorithm described in section 3.1.5. Concentration vector satisfying conditions for supercritical Hopf bifurcation was find for one of the subsystems (determinant-indicated species C1 and C2). Considering symmetry of the system, generalization of this result to the other pairs of extremal pathways is justifiable. Conclusion can be made that the network

Figure 9: Unstable combination of 2 extremal pathways. Determinant indicated species are in black frames.

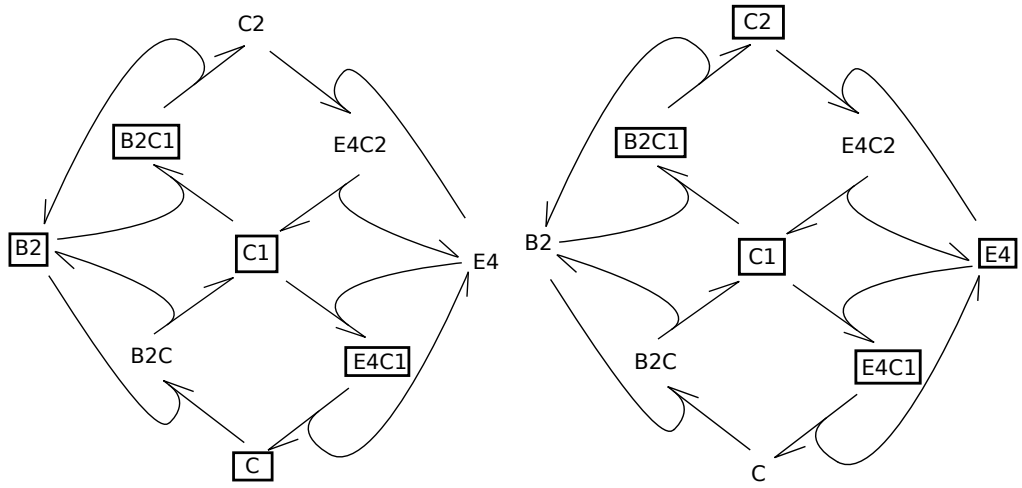
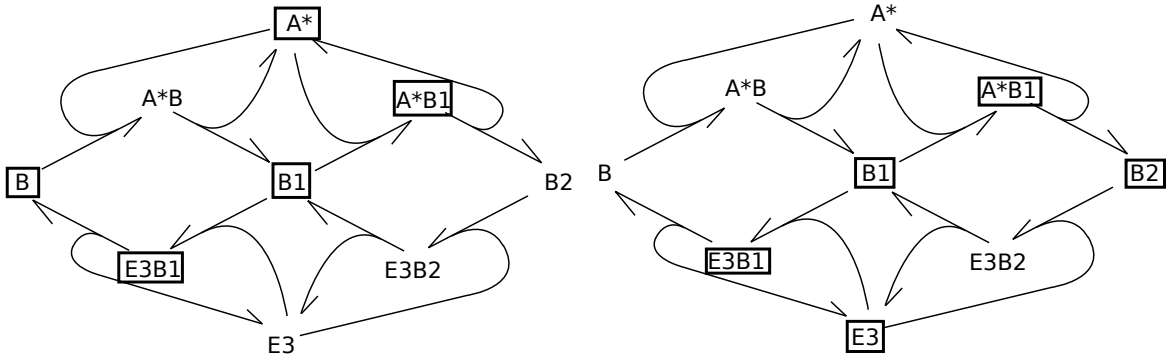


Figure 10: Unstable combination of 2 extremal pathways. Determinant indicated species are in black frames.



can oscillate for some combinations of rate constants. However, it is important to note thaty concentration of one of the determinant-indicated species (B) in the best fitting concentration vector was equal to concentration of one of unindicated species (C). Shift of a property by repeated mutations out of the preset desired region is improbable.

3.3 Continuous Flow System of H2O2 - S2O32- - SO32-

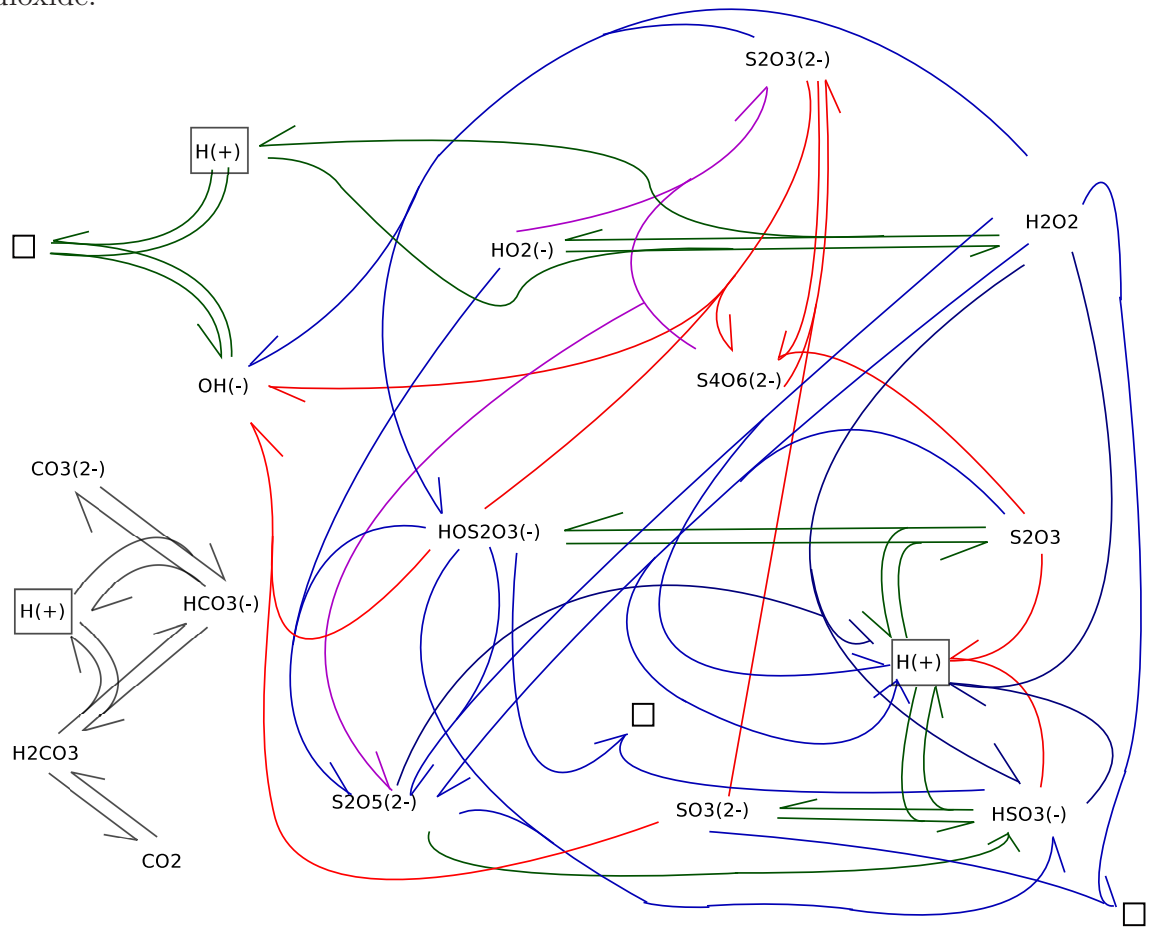
Reaction of hydrogen peroxide and tiosulphate exhibits oscillatory behavior [23] in continuous flow system (CSTR). Since that time, it has been vigorously studied theoretically [24] and experimentally [25]. In this reaction, key role of pH for oscillations was the first time observed. Here we studied an extended system involving also the effect of carbonates.

Pseudoreactions were used to model continuous flow. The sole products of reaction (SO_4^{2-} and $S_3O_6^{2-}$) were not balanced, since their concentration has no effect on dynamics of the system. Water was not balanced, since its concentration in aqueous solutions can be considered the same. The final system comprises 15 species taking part in 30 reactions and 15 outflow and 3 inflow pseudoreactions.

Table 2: Model reactions of HPTS system.

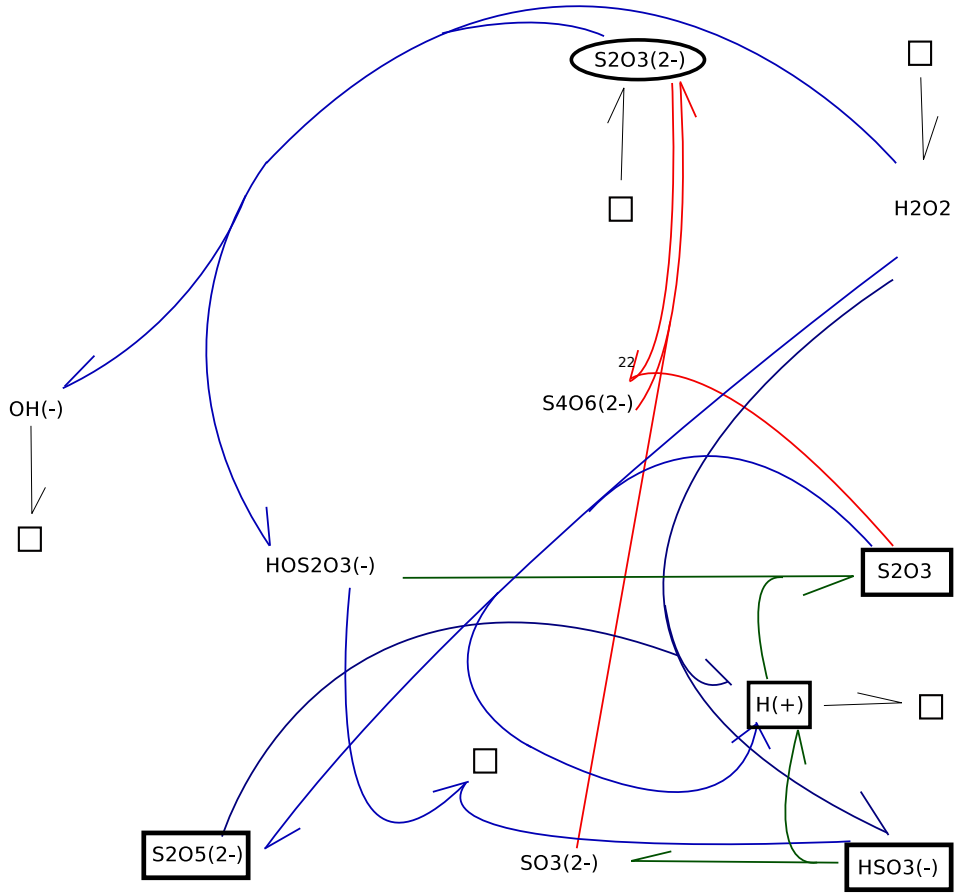
R1	S2O32- + H2O2 → HOS2O3- + OH-	R25	H+ + CO32- → HCO3-
R2	H2O2 + HOS2O3- → S2O52- + H+	R26	HCO3- → H+ + CO32-
R3	S2O32- + HOS2O3- → OH- + S4O62-	R27	H+ + HCO3- → H2CO3
R4	H2O2 + S2O52- → H+ + HSO3-	R28	H2CO3 → H+ + HCO3-
R5	H2O2 + HSO3- → H+	R29	H2CO3 → CO2
R6	→ OH- + H+	R30	CO2 → H2CO3
R7	OH- + H+ →	R31	S2O32- →
R8	HSO3- → H+ + SO32-	R32	H2O2 →
R9	H+ + SO32- → HSO3-	R33	HOS2O3- →
R10	H2O2 → H+ + HO2-	R34	OH- →
R11	H+ + HO2- → H2O2	R35	S2O52- →
R12	H2O2 + SO32- →	R36	H+ →
R13	S2O52- → 2 HSO3-	R37	CO2 →
R14	HOS2O3- + HO2- → S2O52-	R38	S4O62- →
R15	S4O62- + HO2- → S2O32- + S2O52- + H+	R39	HSO3- →
R16	S4O62- + SO32- → S2O32-	R40	SO32- →
R17	HOS2O3- + S2O52- → HSO3-	R41	HO2- →
R18	HOS2O3- + HSO3- →	R42	S2O3 →
R19	HOS2O3- + SO32- → OH-	R43	CO32- →
R20	HOS2O3- + H+ → S2O3	R44	HCO3- →
R21	S2O3 → HOS2O3- + H+	R45	H2CO3 →
R22	S2O32- + S2O3 → S4O62-	R46	→ HCO3-
R23	H2O2 + S2O3 → S2O52- + 2 H+	R47	→ S2O32-
R24	HSO3- + S2O3 → H+	R48	→ H2O2

Figure 11: Network diagram of HPTS system. Pseudoreactions R31-R48 and reaction R18 are hidden for the sake of clarity. Color of curve in diagram correspond to type of the reaction. Green curves represent acidobasic reactions, blue curves reactions in which sulphur is oxidized, red are disproportionations and magenta is used for synproportionation reaction. Grey curves represent acidobasic reactions of carbon dioxide.



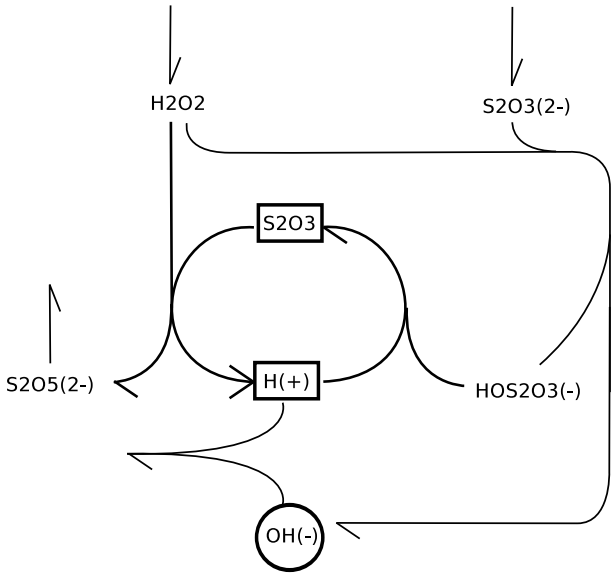
The system can be decomposed into 1177 extremal pathways. Out of them, 468 were shown to be unstable when simple kinetic matrix was used and 388 were shown to be unstable using kinetic matrix assuming acid catalysis of reaction R3. Since most of the extremal pathways had more source of instabilities (deteminant-indicated species), the final numbed of classified extremal pathways with instability sources was 1346.

Figure 12: Unstable extremal pathway in HPTS system classified as 1B. Species of type X are framed in rectangles, species of type Y in the oval. Sulphite plays role of species Z.



Far most (1239) of these instabilities did not contain any critical cycle with suitable exit reaction or strong cycle and therefore were not classified. The system has no extremal pathways exhibiting instabilities of type 2. In 13 extremal pathways, potential oscillator of type 1B was identified. All these extremal pathways possessed the same unstable cycle. A representant of this group is shown in Figure 3.3. The rest 94 unstable extremal pathways were divided into 8 groups according to species of type X, i.e. their constituents of unstable cycles. One of the groups possessed cycles comprising only H+ and S2O3. On Figure 3.3, a member of this group of elementary pathways is presented. H+ and S2O3 occur in unstable cycles of extremal pathways belonging to another 6 groups.

Figure 13: Unstable extremal pathway in HPTS system classified as 1C. Species of type X are framed in rectangles, species of type Y in the circle.



3.4 Oscillations of Hydrogen Peroxide in the Atmosphere

The model proposed by Stewart[26] seems to be in good agreement with observed annual cycle of hydrogen peroxide in the Earth's atmosphere described by Kleinman [27]. Unstable species are created in photochemical reactions where stable species are activated by photons. We assume that these reactions have zeroth order with respect to light, so that the absorption of light by atmosphere is not primarily via the studied reactions. The reactions can be considered to occur in batch mode. Inflow of carbon source had to be added in two pseudoreactions (R66 and R67). The carbon is in the model oxidized to CO_2, which is a stable terminal product of the system and therefore is not balanced in our equations. Oxygen and water vapor were not balanced since their concentration in atmosphere are much higher than concentrations of balanced species and are not influenced by the reaction system.

This large system served as a stress test for the metodology. Calculation of extreme pathways took about 2:34 hours on 1 core of AMD Phenom 2. The following stability calculation and classification took 33 minutes. Network diagram of the system is not shown here because of its size.

The system was decopmposed into 1463 extremal pathways. Out of these, 211 were found to be stable. 2165 combinations of extremal pathways with sets of deteminant-indicated sets of metabolites were analyzed. 1878 of them were not assigned any category. Out of the rest 287 extreme pathways, far most (280) were clasified as of 1C. Three were classified as of category 1B, three as of 2B and just one as of 2C.

Table 3: Model reactions of HPTS system [26]. Abbreviations are used as follows. F - formaldehyde, MP - CH3O2, EP - C2H5O2, PAN - peroxyacetyl nitrate. Metabolites present in considerable excess and final products are not shown.

R1	$O3 \rightarrow 2\ OH$	R35	$NO3 + NO \rightarrow 2\ NO2$
R2	$NO2 \rightarrow NO + O3$	R36	$2\ NO3 \rightarrow 2\ NO2$
R3	$H2O2 \rightarrow OH + OH$	R37	$OH + NO3 \rightarrow HO2 + NO2$
R4	$HNO3 \rightarrow OH + NO2$	R38	$HO2 + NO3 \rightarrow HNO3$
R5	$PAN \rightarrow CH3CO3 + NO2$	R39	$HO2 + NO3 \rightarrow OH + NO2$
R6	$OH + HO2 \rightarrow$	R40	$NO3 + F \rightarrow HNO3 + HO2 + CO$
R7	$H2O2 + OH \rightarrow HO2$	R41	$N2O5 \rightarrow HNO3 + HNO3$
R8	$OH + O3 \rightarrow HO2$	R42	$NO3 + NO2 + M \rightarrow N2O5 + M$
R9	$HO2 + O3 \rightarrow OH$	R43	$N2O5 \rightarrow NO3 + NO2$
R10	$NO + HO2 \rightarrow OH + NO2$	R44	$HNO2 \rightarrow OH + NO$
R11	$NO + O3 \rightarrow NO2$	R45	$HNO4 \rightarrow HO2 + NO2$
R12	$CO + OH \rightarrow HO2$	R46	$NO2 + HO2 + M \rightarrow HNO4 + M$
R13	$2\ OH + M \rightarrow H2O2 + M$	R47	$OH + NO + M \rightarrow HNO2 + M$
R14	$OH + NO2 + M \rightarrow HNO3 + M$	R48	$HNO4 \rightarrow HO2 + NO2$
R15	$2\ HO2 + M \rightarrow H2O2 + M$	R49	$C2H4 + OH + M \rightarrow C2H4OHO2$
R16	$F \rightarrow 2\ HO2 + CO$	R50	$C2H4OHO2 + NO \rightarrow C2H4OOH + NO2$
R17	$F \rightarrow CO$	R51	$C2H4OOH + M \rightarrow 2\ F + HO2$
R18	$CH3OOH \rightarrow MP + OH$	R52	$C2H4 + O3 \rightarrow F + CH2O2$
R19	$CH4 + OH \rightarrow MP$	R53	$C2H4 + O3 \rightarrow F + HO2 + OH$
R20	$MP + HO2 \rightarrow CH3OOH$	R54	$CH2O2 + HO2 \rightarrow F + OH$
R21	$CH3OOH + OH \rightarrow MP$	R55	$C2H6 + OH \rightarrow EP$
R22	$CH3OOH + OH \rightarrow CH2OOH$	R56	$EP + NO \rightarrow NO2 + HO2 + CH3CHO$
R23	$CH2OOH + M \rightarrow F + OH$	R57	$CH3CHO \rightarrow MP + HO2 + CO$
R24	$2\ MP \rightarrow 2\ F + 2\ HO2$	R58	$CH3CHO \rightarrow CH3CO3 + HO2$
R25	$2\ MP \rightarrow F + CH3OH$	R59	$CH3CHO + OH \rightarrow CH3CO3$
R26	$CH3OH + OH \rightarrow MP$	R60	$CH3CHO + NO3 \rightarrow CH3CO3 + HNO3$
R27	$NO + MP \rightarrow NO2 + F + HO2$	R61	$CH3CO3 + NO2 \rightarrow PAN$
R28	$F + OH \rightarrow CO + HO2$	R62	$PAN \rightarrow CH3CO3 + NO2$
R29	$NO3 \rightarrow NO2 + O3$	R63	$CH3CO3 + HO2 \rightarrow M$
R30	$NO3 \rightarrow NO$	R64	$2\ CH3CO3 \rightarrow 2\ MP$
R31	$N2O5 \rightarrow NO2 + NO3$	R65	$CH3CO3 + NO \rightarrow NO2 + MP$
R32	$HNO3 + OH \rightarrow NO3$	R66	$\rightarrow CH4$
R33	$NO2 + O3 \rightarrow NO3$	R67	$\rightarrow C2H4$
R34	$NO3 + NO2 \rightarrow NO + NO2$		

3.5 Modified Belousov-Zhabotinsky System

Belosov-Zhabotinsky (BZ) system was modified [28], so that cyclohexadione was used instead of malonic acid. Advantage over classical BZ reaction is that no bubbles are formed during reaction, which is highly suitable for experimental studies. The system was thoroughly studied by Szalai et al. [29]. This paper was source of the model equations.

The model was complemented by an outflow of the final product (HOBr). Without this output reaction, only reversible reactions were identified as extremal pathways. Benzoquinone was not balanced since it is solely product of the reaction. Bromate ion, H^+ and cyclohexadione were not balanced since their concentration is much higher than the concentrations of other involved species.

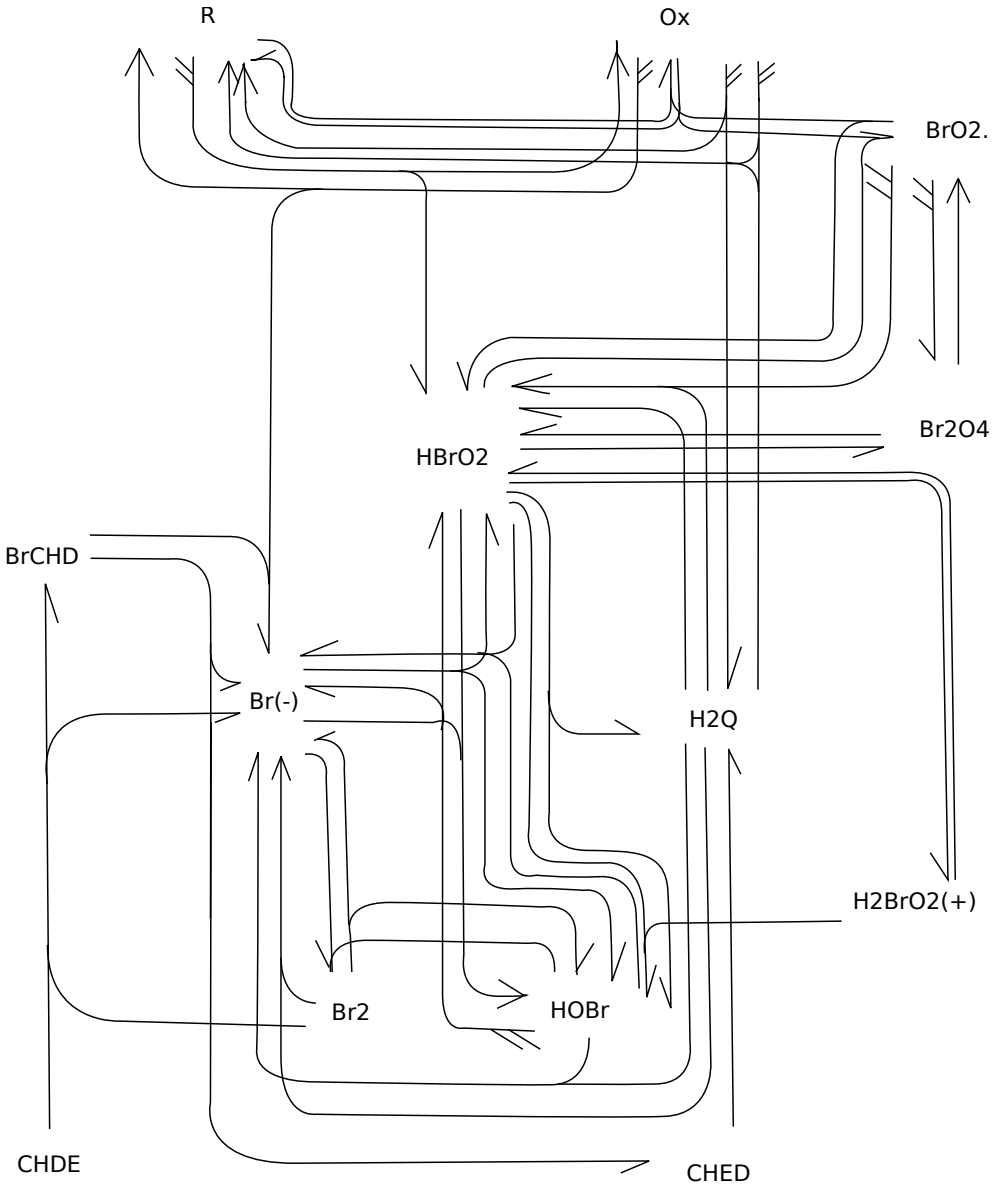


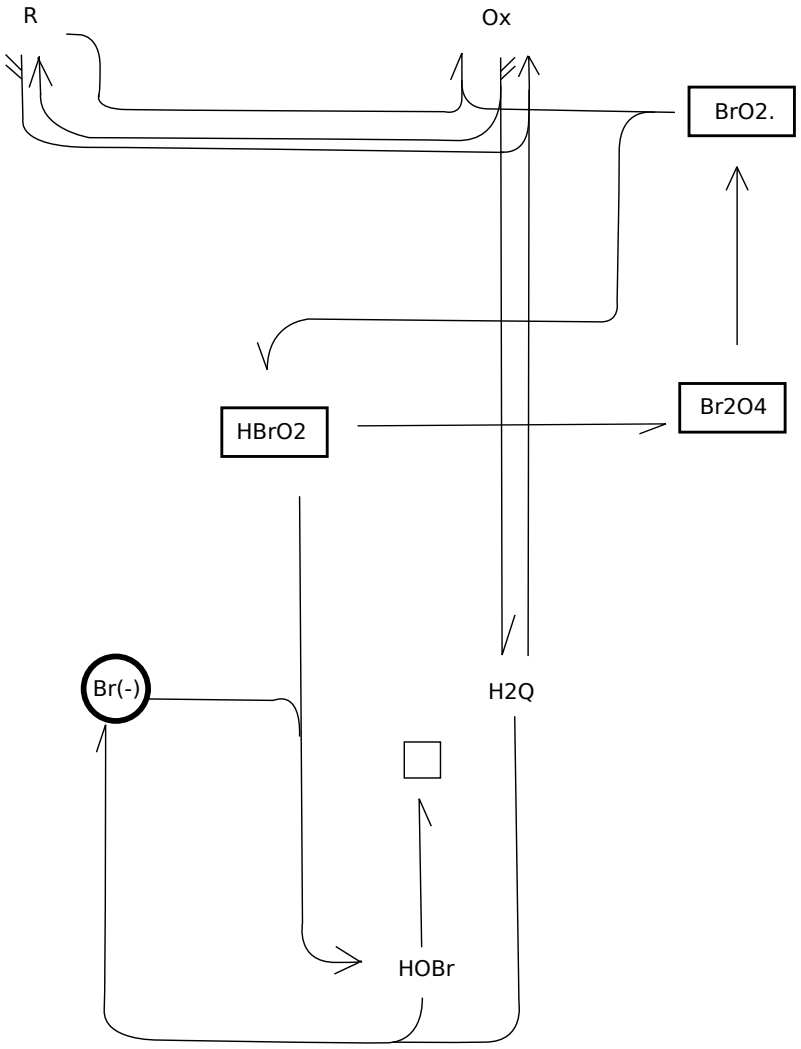
Table 4: Model reactions of modified Belousov-Zhabotinsky system [29]. Abbreviations are used as follows. CHED - cyclohex-2-en-1,4-dione, BrCHD - 2-bromocyclohexane-1,4-dione, CHDE - enol form of cyclohexa-1,4-dione (4-hydroxycyclohex-3-enone), H2Q - 1,4-hydroquinone, Ox - oxidized form of metal catalyst, R - reduced form of metal catalyst. Only balanced species are shown.

R1	Br- + HOBr → Br2	R17	H2Q + HOBr → Br-
R2	Br2 → Br- + HOBr	R18	H2Q + Br2 → 2 Br-
R3	Br- + HBrO2 → 2 HOBr	R19	Ox + HBrO2 → R + BrO2.
R4	2 HOBr → Br- + HBrO2	R20	R + BrO2. → Ox + HBrO2
R5	Br- → HOBr + HBrO2	R21	2 R → 2 Ox + HBrO2
R6	HOBr + HBrO2 → Br-	R22	2 Ox → 2 R + H2Q
R7	HBrO2 → H2BrO2+	R23	2 Ox + H2Q → 2 R
R8	H2BrO2+ → HBrO2	R24	CHED → H2Q
R9	HBrO2 + H2BrO2+ → HOBr	R25	BrCHD → CHED + Br-
R10	HBrO2 → Br2O4	R26	2 Ox + BrCHD → Br- + 2 R
R11	Br2O4 → HBrO2	R27	CHDE + Br2 → BrCHD + Br-
R12	Br2O4 → 2 BrO2.	R28	→ CHDE
R13	2 BrO2. → Br2O4	R29	CHDE →
R14	H2Q + 2 BrO2. → 2 HBrO2	R30	→ H2Q + HBrO2
R15	HBrO2 → H2Q + HOBr	R31	HOBr →
R16	H2Q → HBrO2		

The system can be decomposed into 71 extreme pathways. Out of these, 28 were found to be unstable. Many edges had more than one source of instability, therefore 46 edges with their instability sources were examined. There were 5 unclassified extreme pathways, category 1C in 4 of those and of category 1B in 12 of those. Category 2 was also observed 2B 13 times and 2C 12 times.

Species in unstable cycles were used for systematization of the unstable edges such that edges in 1 group have the same unstable cycle. The species mostly differ in reactions in which determinant-indicated species do not take part. The edges identified as of category 1B were clustered into 3 groups, 1C into 1 group, 2B into 2 and 2C into 3 groups. Representative extremal pathway similar to that found in the original Belousov-Zhabotinsky equation is shown in Figure 14. A representant of another group classified as of 1B category is shown in Figure 15. An interesting phenomenon can be found in Figure 16. Single extremal pathway has 2 different sets of metabolites indicated by determinant, whose cycles were classified as of different classes (1B and 2B). Representant of the other group classified as of category 2B is not shown here. Unstable cycles of extremal pathways classified as of category 2B include terminal product of the reaction (HOBr) whose concentration increases with time. Therefore,

Figure 14: Unstable extremal pathway in modified Belousov-Zhabotinsky system similar to one present in original Belousov-Zhabotinsky reaction. Category 1B. Species of type X are framed in rectangles, species of type Y in the circle. HOBr plays role of species Z. In an another extremal pathway in this group, type Z species is oxidized form of the catalyst.



its effect should be observed only in the initial stage of experiments. Its unstable cycle involves all the species involved in the one depicted in Figure 16 right, i.e. bromine, bromide anion, hypobromic acid and 2-bromo-1,4-cyclohexadione, and HBrO2.

Figure 15: Unstable extremal pathway in modified Belousov-Zhabotinsky system not found in original Belousov-Zhabotinsky reaction. Category 1B. Species of type X are framed in rectangles, species of type Y in the circle. HOBr and Br2 serve as type Z species.

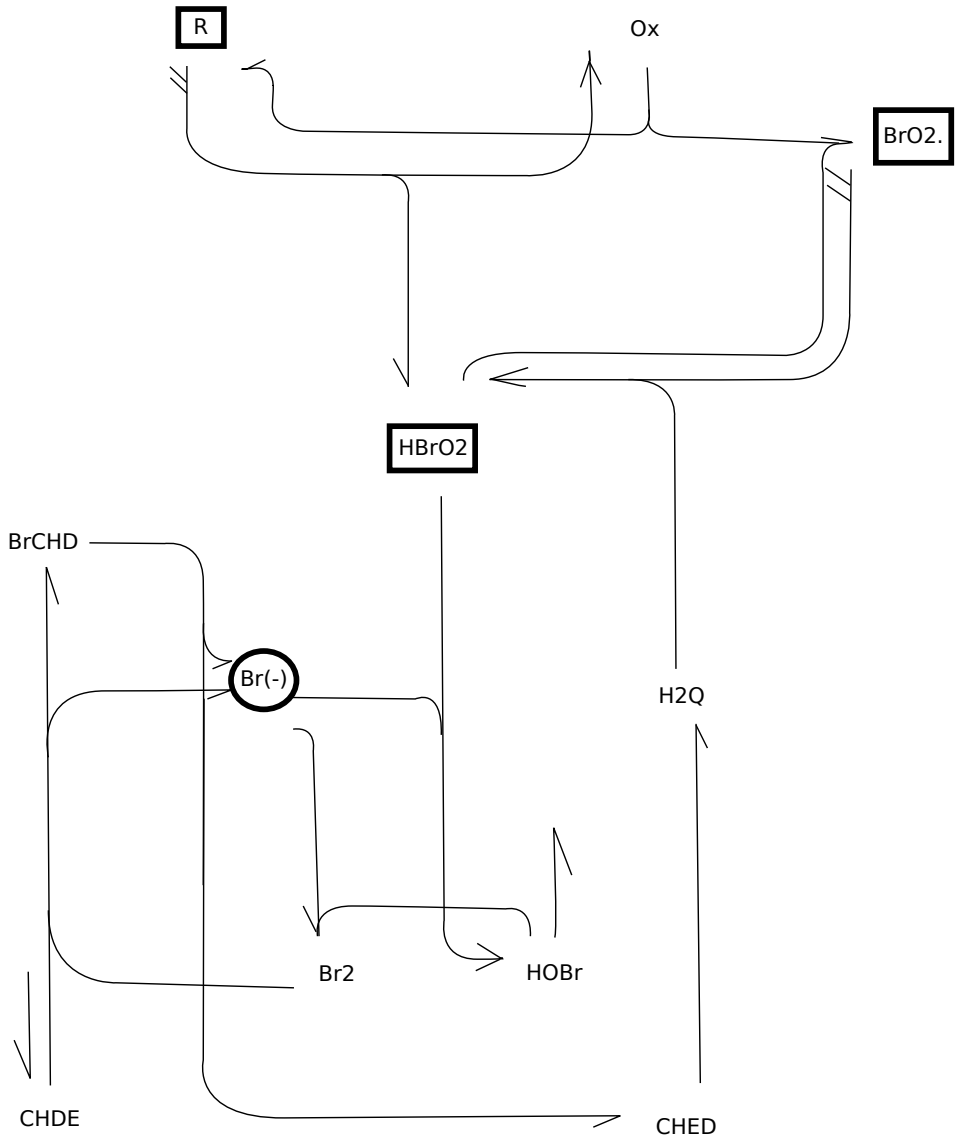
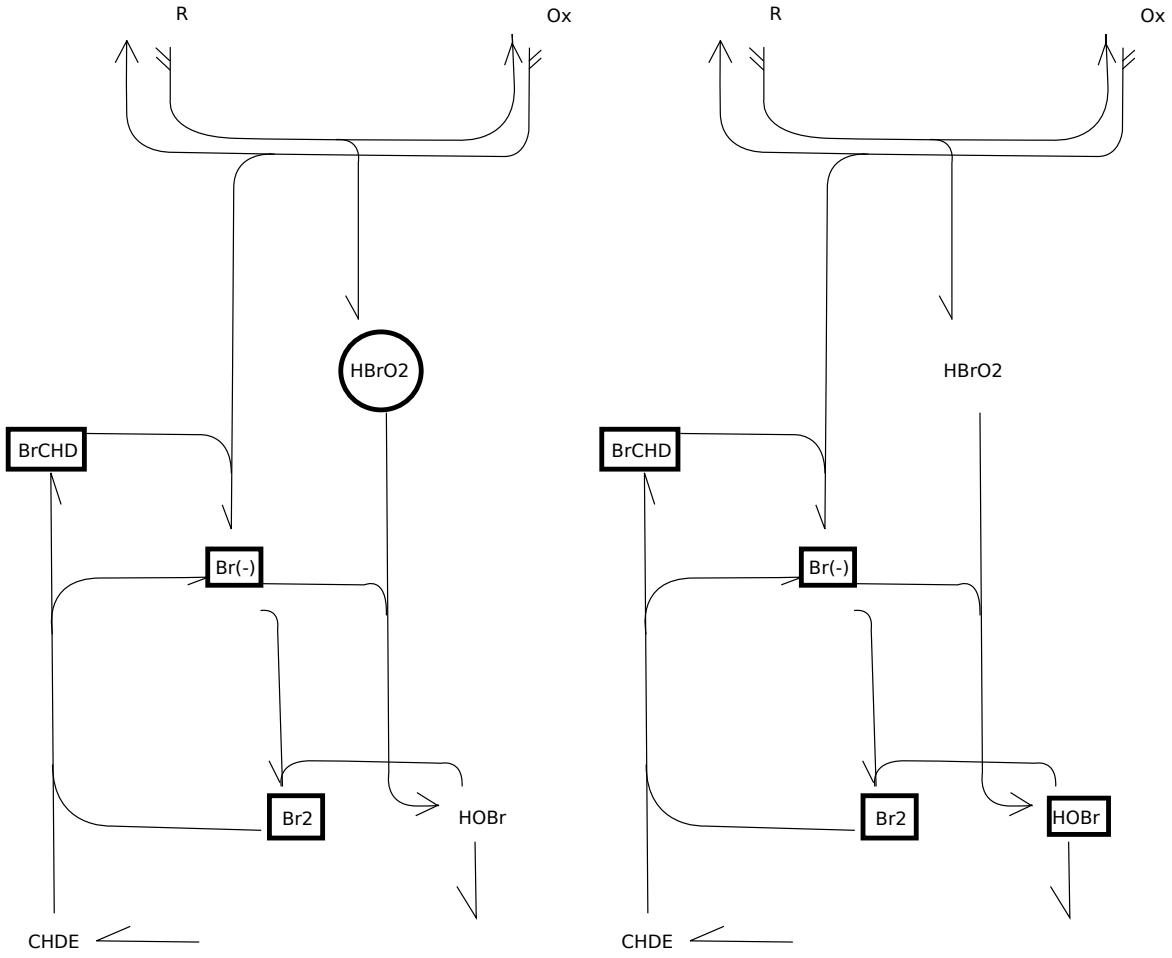


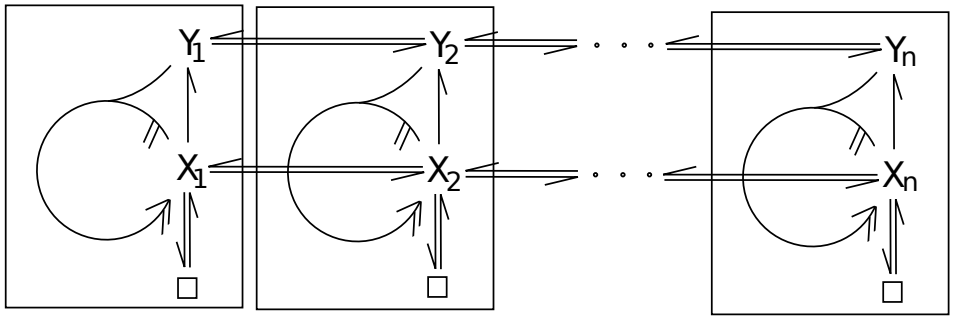
Figure 16: Unstable extremal pathways in modified Belousov-Zhabotinsky system. Species of type X are framed in rectangles, species of type Y in the circle. LEFT: Representant of one of three groups of extremal pathways classified as 1B. Here oxidized form of metal catalyst serves as type Z species. RIGHT: Representant of one of two groups of extremal pathways classified as of 2B.



3.6 Chemical Reactors with Mass Transfer

System of well-mixed identical batch reactors with mass transfer was studied. In each reactor, two species react in a reaction system known as Brusselator. Mass transfer (or diffusion) is modeled by pseudoreactions. In this model, diffusion-induced instability can be intuitively imagined as instability of extreme currents including

Figure 17: System of well-mixed reactors with mass transfer. Species X in ith reactor is represented by species X_i. Addition of a new reactor increases the system by 2 new species and 6 new reactions.



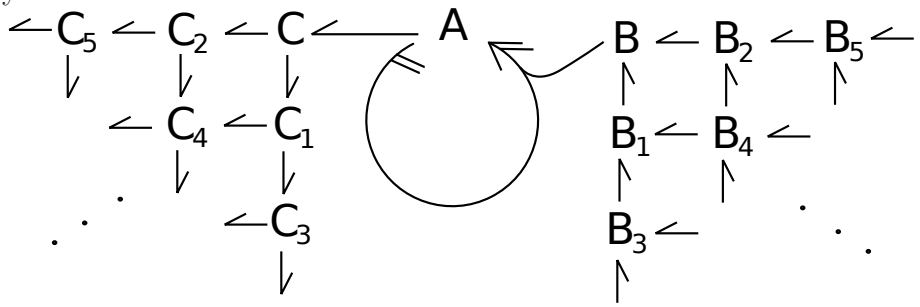
Number of extreme pathways grows fast with the system size. Simple Brusselator system has 2 extreme pathways. Brusselators in two coupled reactors were decomposed into 12 extreme pathways, in three reactors into 36 pathways, in 4 reactors into 92 pathways and in 5 reactors into 216 pathways. Most of the unstable pathways have autocatalytic reaction in one reactor and negative feedback reaction in an another one.

4 DISCUSSION AND CONCLUSIONS

Automatic classification of potential oscillators proposed and implemented in this work enables managing the output of stoichiometric network analysis. Classified unstable extreme pathways can be grouped according to their categories. Then, all extreme pathways of one category can be grouped based on set of their X species. In all the models analyzed in this work, such grouping significantly decreased the number Analogically, extreme pathways can be grouped according to their Y or Z species. We can continue to simplify the results by searching only for the most elementary unstable features. For example, if there is an extreme pathway whose set of X species is a subset of X species of an another extreme pathway, we can omit the latter.

Outcomes of this work enable comfortable qualitative analyses of large oscillators. However, their size is limited by computational complexity of stoichiometric network analysis. Unlike number of interesting features in network might increase linearly with size of the system, number of extreme pathways increases exponentially with system size [30]. Figure 4 illustrates this problem. Second, even if a single steady state subnetwork is selected, the number of principal subdeterminants to be evaluated exponentially increases with the system. The latter problem can be solved by massive paralelization, sicne calculation of all subdeterminants can be easily alloted. However, undesirable scaling sets limits for the methodology.

Figure 18: Illustration of combinatorial complexity of network decomposition problem. The number of extremal pathways increases exponentially with number of B_i's or C_i's because all pathways have to be included in basis. However, the interesting autocatalytic feature remains the same.



An another open problem is the need for combinations of extreme pathways, i.e. two- and higher-dimensional faces of current cone. The case of MAPK model illustrates that sometimes sources of instability can be suppressed in single extreme pathways and be expressed only when combined with other pathways. To our knowledge, there is no theory stating that a finite dimensionality of current cone is sufficient for identification of all instability sources.

The problem of instability sources is closely related to the question of number of potential oscillator categories. Only a marginal fraction of all unstable edges was

classified in the hereby analyzed models. It would be perhaps interesting to classify the yet unclassified unstable pathways. The categorization is based on observations and various dynamics / experimental / theoretical studies. The "others" category remains. There is a natural question, whether oscillators can be classified into a finite number of categories defined by network topology. If not, can the categories be systematically described? If yes, is there a sufficient number of extreme pathways which in combination ensure identification of such feature? And how many of network topology derived categories are relevant for dynamics?

Throughout the thesis, there is a strict distinction between oscillators, potential oscillators and unstable pathways. Not all unstable pathways satisfy conditions for oscillations. We do not know about any method of determining whether an unstable network can exhibit oscillations provided suitable reaction rate vector. By hereby proposed genetic algorithm, we can only decide whether there is a possibility of oscillation by finding a concentration vector fulfiling the conditions. Failure to converge, even in 5000 generations set by default does not ensure lack of this feature.

The case of modified BZ reaction well illustrates the fact that one system can involve more unstable networks classified as potential oscillators. Moreover, these potential oscillators may be of different categories. For example, we have identified a 2B potential oscillator in modified BZ reaction. In this potential oscillator network, HOBr plays role of X species. The oscillator might be exhibited in the system under some conditions but as the reaction proceeds and the terminal product HOBr accumulates it vanishes. Relevance of each subnetwork is given by its actual flow or by its interaction with the other networks. As mentioned in chapter 2.1, all reactions are in principle possible and determining, which reactions are present is based on their rate. Similarly, presence of a potential oscillator is not a discrete phenomenon but rather a present feature sometimes with negligible amplitude and sometimes with amplitude sufficient to dominate the dynamics.

References

[1] Field, R. J.; Koros, E.; Noyes, R. M. *Journal of the American Chemical Society* **1972**, *94*, 8649–8664.

[2] Gillespie, D. *The Journal of Physical Chemistry* **1977**, *81*, 2340–2361.

[3] Domijan, M.; Kirkilionis, M. *Journal of Mathematical Biology* **2009**, *59*, 467–501.

[4] Papin, J. A.; Stelling, J.; Price, N. D.; Klamt, S.; Schuster, S.; Palsson, B. O. *Trends in Biotechnology* **2004**, *22*, 400 – 405.

[5] Papin, J. A.; Price, N. D.; Wiback, S. J.; Fell, D. A.; Palsson, B. O. *Trends in Biochemical Sciences* **2003**, *28*, 250–258.

[6] Xi, Y.; Chen, Y.-P. P.; Cao, M.; Wang, W.; Wang, F. *BMC Bioinformatics* **2009**, *10*, year.

[7] Clarke, B. L. *Advances in Chemical Physics* **1980**, *43*, 1–215.

[8] Hadač, O.; Schreiber, I. *Physical Chemistry Chemical Physics* **2011**, *13*, 1314–1322.

[9] Klamt, S.; Stelling, J. *Trends in Biotechnology* **2003**, *21*, 64 – 69.

[10] Eiswirth, M.; Freund, A.; Ross, J. In *Mechanistic Classification of Chemical Oscillators and the Role of Species*; John Wiley and Sons, Inc., 1991; pp 127–199.

[11] Chevalier, T.; Schreiber, I.; Ross, J. *The Journal of Physical Chemistry* **1993**, *97*, 6776–6787.

[12] Schreiber, I.; Hung, Y.-F.; Ross, J. *The Journal of Physical Chemistry* **1996**, *100*, 8556–8566.

[13] Eiswirth, M.; Bürger, J.; Strasser, P.; Ertl, G. *The Journal of Physical Chemistry* **1996**, *100*, 19118–19123.

[14] Ross, J.; Schreiber, I.; MO, V. In *Oscillatory reactions*; Oxford University Press, 2005; pp 125–169.

[15] Schreiber, I.; Ross, J. *The Journal of Physical Chemistry A* **2003**, *107*, 9846–9859.

[16] Schilling, C.; Letscher, D.; Palsson, B. *Journal of Theoretical Biology* **2000**, *203*, 229 – 248.

[17] Novak, B.; Tyson, J. J. *Nature Reviews Molecular Cell Biology* **2008**, *9*, 981–991.

[18] Holodniok, M.; Klíč, A.; Kubíček, M.; Marek, M. In *Metody analýzy nelineárních dynamických modelů*; ACADEMIA, 2005.

[19] Hornberg, J.; Binder, B.; Bruggeman, F.; Schoeberl, B.; Heinrich, R.; Westerhoff, H. *Oncogene* **2005**, *24*, 5533–5542.

[20] Qiao, L.; Nachbar, R. B.; Kevrekidis, I. G.; Shvartsman, S. Y. *PLoS Comput Biol* **2007**, *3*, e184.

[21] Takahashi, K.; Tanase-Nicola, S.; ten Wolde, P. R. *Proceedings of the National Academy of Sciences of USA* **2010**, *107*, 2473–2478.

[22] Huang, C.; Ferrell, J. *Proceedings of the National Academy of Sciences of USA* **1996**, *93*, 10078–10083.

[23] Orban, M.; Epstein, I. R. *Journal of the American Chemical Society* **1987**, *109*, 101–106.

[24] Rábai, G. a. *The Journal of Physical Chemistry A* **1999**, *103*, 7268–7273.

[25] Yuan, L.; Gao, Q.; Zhao, Y.; Tang, X.; Epstein, I. *The Journal of Physical Chemistry A* **2010**, *114*, 7014–7020, PMID: 20536217.

[26] Frey, M. M.; Stewart, R. W.; McConnell, J. R.; Bales, R. C. *Journal of Geophysical Research (Atmospheres)* **2005**, *110*, 23301.

[27] Kleinman, L. *Journal of Geophysical Research (Atmospheres)* **1991**, *96*, 20721–20733.

[28] Kurin-Csörgei, K.; Zhabotinsky, A. M.; Orbán, M.; Epstein, I. R. *The Journal of Physical Chemistry* **1996**, *100*, 5393–5397.

[29] Szalai, I.; Kurin-Csorgei, K.; Epstein, I.; Orban, M. *The Journal of Physical Chemistry A* **2003**, *107*, 10074–10081.

[30] Klamt, S.; Stelling, J. *Molecular Biology Reports* **2002**, *29*, 233–236.

LIST OF ABBREVIATIONS AND SYMBOLS

Abbreviations

BZ - Belousov-Zhabotinsky
CRN - chemical reaction network
CSTR - continuous-flow stirred tank reactor
HPTS - hydrogen peroxide - tiosulphate system
MAPK - mitogen-activated protein kinase
PES - potential energy surface
SNA - stoichiometric network analysis
SODE - system of ordinary differential equtions

Symbols

$\boldsymbol{\kappa}$ - kinetic matrix
$\boldsymbol{\nu}$ - stoichiometric matrix
$\boldsymbol{\nu}^{\mathrm{L}}$ - left stoichiometric matrix
$\boldsymbol{\nu}^{\mathrm{R}}$ - right stoichiometric matrix
$\mathbf{B}$ - matrix defined in Equation **??**B), important for stability analysis
$\mathbf{e}$ - extreme pathway; vector in reaciton rate space
$\mathbf{f}$ - functions on right side of SODE
$\mathbf{J}$ - Jacobian matrix
m - number of species in the system
r - number of reactions in the system
$\mathbf{v}$ - reavtion rate vector
$\mathbf{x}$ - concentration vector of species
$\mathbf{x}_0$ - concentration vector of species in a steady state

Note: Symbols for species used specifically in only one model are not listed here. See table of reactions at the beginning of each model section in theoretical part.

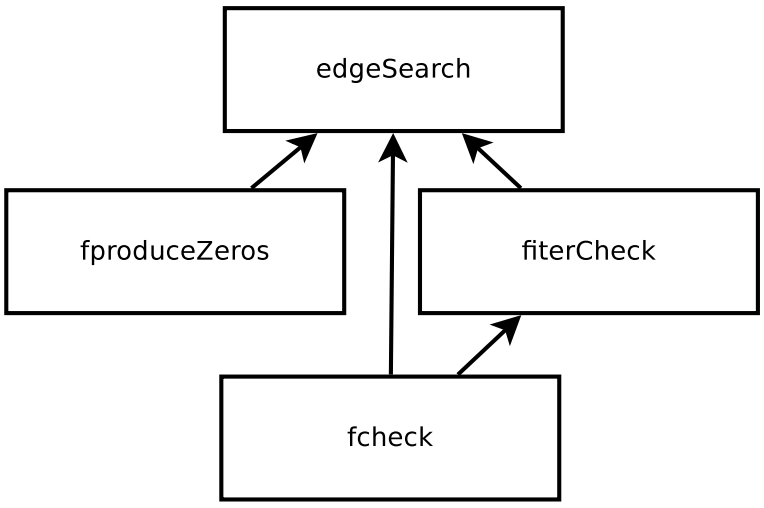
APPENDICES

I tried to maintain some standards to make the code easy to read. Identifiers of matrices always start with a capital letter, identifiers of vectors start with lower case `v` and scalars with other lower case letter. Functions (exept for the main functions) start with `f` .

In the hereby presented documentation, some additional rules can be noticed. First, all MATLAB/Octave keywords are highlighted blue and Octave/MATLAB built-in functions brown, all the identifiers (functions and variables) magenta, strings are green and comments (if any) are red. A dependency diagram is shown in the beginning of each documentation. All the variables are briefly explained with some comments of their type or unusual properties and all the used built-in Octave/MATLAB functions listed in order to help the user / developer cope with other (earlier) versions of Octave / MATLAB. Just to remember, all the presented functions were tested for `Octave 3.2.4` . MATLAB-compatible (tested on version R2010) source codes can be found in the enclosed CD.

A Documentation for the function `edgeSearch`

A.1 Dependency diagram



For brief explanation of algorithm see chapter 3.1.2. It is worth to note that the number of edges can be very high; it can grow exponentially with number of reactions. Therefore, the limiting step is in the function `fcheck`. Setting relative change of temporary edges in an iteration as the control parameter can be introduced by small changes in the code (line 71).

A.2 Main function

Variables

Input variables		
`S`	real	stoichiometric matrix each row corresponds to 1 metabolite
`binSize`	integer	size of bin in `fiterCheck` function
`maxChange`	integer	control parameter for `fiterCheck` function max number of redundant temporary edges
Output variables		
`E`	positive real	Edges. Each row represents 1 edge.
Local variables		
`E`	sparse positive real	temporary edges
`E1`	sparse positive real	temporary edges
`E2`	sparse positive real	temporary edges
`Ebool`	sparse boolean	indication of involved reactions
`vindE1`	positive integers	indices of temporary edges necessary for random rearrangement
`vnosreact`	positive integers	numbers of reactions involved in each edge
`change`	positive integer	number of redundant edges in one iteration
`i`	positive integer	index
`j`	positive integer	index
`l`	integer	number of reactions
`m`	integer	number of metabolites
`numberEdges`	positive integer	number of temporary edges in E
`numberEdges1`	positive integer	number of temporary edges in E1

The only necessary input is stoichiometric matrix S. Time spent on 1 iteration of iterCheck is proportional approximately to square of binSize, but number of iterations necessary increases with decreasing binSize. For usual applications, its most efficient value was found to be around 1000.

Code description

Setting default values if no `binSize` and/or `maxChange` is specified on input. Default values for control of `fiterCheck` seem to perform well. Then, extended matrix of edges `E` is initialized.

```
function E = edgeSearch(S,binSize,maxChange)
if nargin == 2
    maxChange = 10;
elseif nargin == 1
    binSize = 1000;
```

```
6       maxChange = 10;
7   endif
8   [m,l] = size(S);
9   E = sparse([eye(l) S']);
```

The main cycle of algorithm - for each metabolite, temporary edges are first combined using function `produceZeros`. `E1` is sparse yet very big matrix. After the combination, redundant temporary edges are eliminated. Since most of vectors produced in `produceZeros` are redundant, iterative (`while` loop, lines 13-19) approach using function `iterCheck` is employed. Temporary edges are randomly divided into bins of temporary edges of size `binSize`. Edges are scaled to remain at the same order of magnitude (lines 20-23). This procedure is a way how to cope with errors in floating point equality tests. An another way is introduction of some arbitrary tolerance. However, this approach increases final number of extreme pathways since `unique` function does not use it.

```
10  for i=1:m
11      E1 = fproduceZeros(E,i-1);
12      change = 2*maxChange;
13      while change > maxChange
14          numberEdges1 = size(E1,1);
15          vindE1 = randperm(numberEdges1);
16          E2 = E1(vindE1,:);
17          [E,change] = fiterCheck(E2, binSize);
18          E1 = E;
19      endwhile
20      numberEdges = size(E,1);
21      for j=1:numberEdges
22          E(j,:) = E(j,:) / max(abs(E(j,:)));
23      endfor
24  endfor
```

The final set of edges is non-iteratively checked for redundancies and the final set of edges is scaled. This part of program is responsible fo the formal time scaling of algorithm $O(N^2)$, where N is the final number of edges.

```
25  E=fcheck(E);
26  numberEdges = size(E,1);
27  for j=1:numberEdges
28      E(j,:) = E(j,:) / max(abs(E(j,:)));
29  endfor
```

Matrix of edges is extracted from the extended matrix of edges and the edges are sorted according to the number of reactions involved. In the end, final edge matrix must be converted from sparse to full matrix in order to be sorted; sortrows operates only on full matrices.

```
E(:,1+1:1+m)=[];
Ebool = full(E)>0;
vnosreact = Ebool*ones(1,1);
E = sortrows(full([vnosreact E]))(:,2:(1+1))
endfunction
```

A.3 Local function `fproduceZeros`

This function classifies the temporary edges as those having zero, positive or negative numbers in column corresponding to the principal metabolite. Then it calls the function combine to combine edges with negative values with those with positive values. All the output temporary edges have zeros in column corresponding to the principal metabolite.

Variables

Input variables		
E	sparse real	temporary edges
n	positive integer	index of principal metabolite
Output variables		
Enew	sparse real	without redundant temporary edges
Other variables		
K	sparse real	temporary edges with positive value of principal metabolite
Z	sparse real	temporary edges with negative value of principal metabolite
N	sparse real	temporary edges with zerovalue of principal metabolite
mi	positive integer	index of principal metabolite

Code description

This function only classifies temporary edges into those having negative and positive values in the field corresponding to principal metabolite. From these temporary edges, new matrix of temporary edges is produced using `fcombine` function.

```
function Enew = fproduceZeros(E,n)
mi = size(E,2)-n;
K = E(find(E(:,mi)>0),:);
Z = E(find(E(:,mi)<0),:);
N = E(find(E(:,mi)==0),:);
Enew = sparse([N ; fcombine(K,Z,mi)]);
endfunction
```

A.4 Local function `fcombine`

Combines all the temporary edges with negative value Double cycle is necessary. The number of lines of `NN` equals to number of lines of `Z` multiplied by number of lines of `K` .

Variables

Input variables		
`K`	sparse real	temporary edges with positive value of principal metabolite
`Z`	sparse real	temporary edges with negative value of principal metabolite
`mi`	positive integer	index of principal metabolite
Output variables		
`NN`	sparse real	large matrix of temporary edges including redundant
Other variables		
`i`	positive integer	index (of temporary edges with positive value)
`j`	positive integer	index (of temporary edges with negative value)
`k`	positive integer	number of temporary edges in `K`
`s`	positive integer	number of columnes in matrices `K, Z, NN`
`z`	positive integer	number of temporary edges in `Z`

Code description

Initialization of output matrix.

```
42 function NN = fcombine(K,Z,mi)
43 [k,s] = size(K);
44 z = size(Z,1);
45 NN = zeros(k*z,s);
```

The double cycle combines each positive value with each negative value.

```
46 for i=1:k
47     for j=1:z
48         NN((i-1)*z+j,:) = sparse(K(i,:)/K(i,mi) - Z(j,:)/Z(j,mi)
               );
49     endfor
50 endfor
51 endfunction
```

Input variables		
E	sparse real	matrix of temporary edges with redundancies
binSize	positive itneger	size of bin for pairwise redundancy check
Output variables		
Enew	sparse real	matrix of temporary edges without redundancies in bins
change	positive integer	number of found redundant temporary edges
Other variables		
A	sparse real	temporary edges from 1 bin with no redundancies
vchange	real	absolute and relative amount of all the found redundancies
a	positive integer	number of input temporary edges
b	positive integer	size of a temporary edge
memoryCheckPoint	positive integer	pointer for storage of output in memory
numberBins	positive integer	number of bins
r	positive integer	number of temporary edges in a bin after the check for redundancies

A.5 Local function `fiterCheck`

Variables

Code description

Calculates the number of bins. If the number of temporary edges in `E` is lower than `binSize` , simple check procedure is executed. Memory necessary for sparse matrix Enew has upper bound - size of `E` . The number of bins is rounded down, the remaining vectors are simply left for the next iteration.

```
52 function [Enew, change] = fiterCheck(E, binSize)
53 [a, b] = size(E);
54 numberBins = floor(a/binSize);
55 Enew = spalloc(a,b,nnz(E));
56 if numberBins==0
57     Enew = fcheck(E);
58     change = 0;
59     return;
60 endif
```

Stepwise filling of preallocated memory spares a lot of time, but is little more complicated.

```
61 memoryCheckPoint = 0;
62 for i=1:numberBins
63     A = fcheck(E((i-1)*binSize+1:i*binSize,:));
64     r = size(A,1);
```

```
65      Enew(memoryCheckPoint+1:memoryCheckPoint+r,:) = A;
66      memoryCheckPoint = memoryCheckPoint+r;
67  endfor
```

Finally, the remainig temporary edges are appended and the freed memory corresponding to erased temporary edges is deallocated. The relative change `vchange(2)` of size of `E` has only informative value.

```
68  Enew((memoryCheckPoint+1):(memoryCheckPoint+a-numberBins*binSize
        ),:) = (E(((numberBins)*binSize+1):a,:));
69  Enew((memoryCheckPoint+a-numberBins*binSize+1):a,:)=[];
70  change = a-size(Enew,1);
71  vchange = [change 100.0*(a-size(Enew,1))/size(E,1)]
72  endfunction
```

A.6 Local function `fcheck`

Variables

Input variables		
`E`	(sparse) real	temporary edges with redundancies
Output variables		
`Enew`	sparse real	temporary edges with no redundancies
Other variables		
`M1`	bool	zero fields in the selected temporary edge
`M2`	bool	zero fields in all the input temporary edges
`M2minusM1`	integer $(1\|0\|-1)$	simply `M2 - M1`
`vdecide`	bool	
`vdel`	bool	
`veq`	bool	
`visSubset`	bool	
`i`	positive integer	index of temporary edges
`r`	positive integer	number of temporary edges
`q`	positive integer	size of a temporary edge

Code description

This function presumes each vector to be non-redundant. `E` is transformed into full matrix since the function `unique` operates only on full matrices.

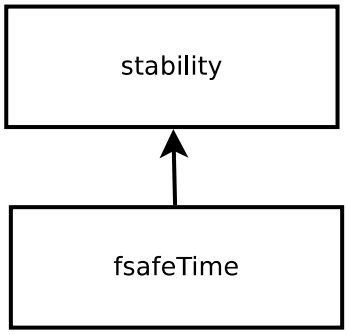
```
73  function Enew = fcheck(E)
74  E = unique(full(E), 'rows');
75  [r,q] = size(E);
76  vdel = zeros(r,1);
```

In each cycle of this main loop fcheck compares a selected vector with all other vectors. The procedure is formally only single loop because it is vectorized. M1 is simply repeated vector of the zero fields in the selected vector (temporary edge) and M2 is simply matrix of the zero fields of all vectors. In line ... boolean matrix (`M2minusM1 > -1`) is multiplied by unit vector to get the number of zeros and ones in the `vdecide` is in line ... used as vector of integers (function `sum`).

```
77 for i=1:r
78     M1 = repmat(E(i,:)==0,r,1);
79     M2 = E==0;
80     M2minusM1 = M2 - M1;
81     veq = (M2minusM1*ones(q,1)) == 0;
82     visSubset = ((M2minusM1>-1)*ones(q,1))==q;
83     vdecide = (veq == 0) & (visSubset == 1);
84     if sum(vdecide)>0
85         vdel(i)=1;
86     endif
87 endfor
88 Enew = E;
89 Enew(find(vdel),:) = [];
90 Enew = sparse(Enew);
91 endfunction
```

B Documentation for the function `stability`

B.1 Dependency diagram



B.2 Main function

Variables

Input variables		
`K`	real	kinetic matrix
`S`	real	stoichiometric matrix
`vs`	positive real	convex combination of edges
Output variables		
`vmind`	positive integer	indices of metabolites indicated by determinant can be matrix or scalar
`isStable`	bool	1 if stable, 0 if unstable
Other variables		
`B`	square real	matrix defined by equation 11
`Kombc`	positive integer	each row represents indices of one combination number of rows can be very high
`PrincSub`	square real	selected principal subdeterminant of B
`vlines`	bool	non-zero rows and columns of B
`vyber`	positive integer	one combination of `j` indices one row from `Komb`
`i`	positive integer	index of metabolites (lines of `B`)
`j`	positive integer	index of combinations (lines of `Komb`)
`isLastCycle`	bool	1 if loop has to end after current cycle
`m`	positive integer	number of rows of S
`mm`	positive integer	number of non-zero rows of B

Code description

If an unstable subdeterminant comprising `j` metabolites is found, the cycle is completed (lines 14-22) by serching for other combinations of `j` metabolites inducing an unstable

subdeterminant of `B` . Therefore, the function `stability` returns as `vmind` all the minimum index combinations corresponding to unstable subdeterminants (line 19).

```
1  function [isStable, vmind] = stability(S,vs,K)
2  m = size(S,1);
3  vmind = 0; %! scalar (0) if the edge is stable
4  B = - S * diag(vs) * K';
5  isStable = 1;
6  vlines = fsaveTime(B);
7  mm = sum(vlines);
8  isLastCycle = 0;
9  for i=1:mm
10     Kombc = nchoosek((1:m)(find(vlines)),i);
11     for j=1:size(Kombc,1)
12         vyber = Kombc(j,:);
13         Princsub = B(vyber,vyber);
14         if det(Princsub)<0
15             isStable=0;
16             if vmind == 0
17                 vmind = vyber;
18             else
19                 vmind = [vmind ; vyber];
20             endif
21             isLastCycle = 1;
22         endif
23     endfor
24     if isLastCycle == 1
25         return;
26     endif
27 endfor
28 endfunction
```

B.3 Local function `fsafeTime`

Variables

Code description

In the beginning, the algorithm presumes all rows / columns to be non-zero. In a loop, it check whether for an index `i` both row and column are zero. In that case, that row (and column) is unnecessary.

```
29 function vlines = fsaveTime(B)
30 b = size(B,1);
31 vlines = ones(1,b);
```

Input variables		
B	real square	matrix defined in 11 or see line 4 of the main function

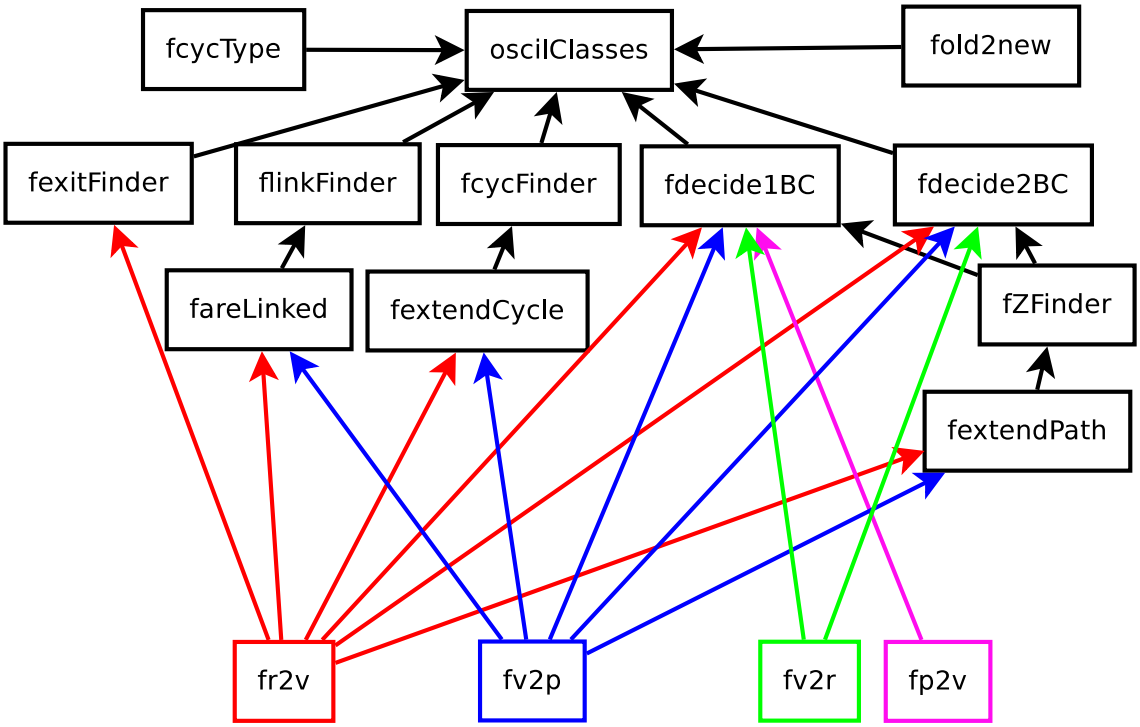
Output variables		
vlines	bool	0 for ith metabolite if ith line and row in B is zero

Other variables		
b	positive integer	size of matrix B
i	positive integer	index

```
32 for i=1:b
33     if ((B(i,:)==zeros(1,b)) && (B(:,i)==zeros(b,1)))
34         vlines(i) = 0;
35     endif
36 endfor
37 endfunction
```

C Documentation for the function `oscilClasses`

C.1 Dependency diagram



C.2 Main function

The program takes on input left and right stoichiometric matrices, extremal pathway and set of metabolites indicated by determinant. Emtol is intended to be a very low number; it is a error tolerance level in floating point equality tests. Types 12, 13, 14, 22 and 23 on output stand for 1B, 1C, unclassified, 2B and 2C respectively.

Variables

Stoichiometric matrix is calculated from left and right stoichiometric matrices. vw is usually void set.

```
function [typ, vx, vy, vz, vw, noCycl] = oscilClasses(ve, vmold,
    Slbig, Srbig, Kbig, emtol)
Sbig = Srbig - Slbig;
if nargin == 5
    emtol = 1e-6;
elseif nargin == 4
    emtol = 1e-6;
    Kbig = Slbig;
endif
noCycl = 0;
```

Input variables		
Kbig	real	kinetic matrix of the whole system
Slbig	real	left stoichipometric matrix of the whole system
Srbig	real	right stoichipometric matrix of the whole system
vmold	positive integer	indices of species indicated by determinant
ve	positive real	extremal pathway - coefficients of reactions
emtol	positive real	tolerance for floating point equality tests
Output variables		
vw	positive integer	list of indices for species of type W
vx	positive integer	list of indices for species of type X
vy	positive integer	list of indices for species of type Y
vz	positive integer	list of indices for species of type Z
noCycl	positive integer	number of found cycles
typ	(12|13|14|22|23)	the determined type of the potential oscillator
Local variables		
Cycles	positive integer	cycles
CycsSizes	positive integer	cycles with their sizes in first column
K	real	kinetic matrix of the studied subsystem
S	real	stoichiometric matrix of the studied subsystem
Sl	real	left stoichiometric matrix of the studied subsystem
SmallCycles	positive integer	cycles of determinant-indicated species indices in the studed subsystem
Sr	real	right stoichiometric matrix of the studied subsystem
vmind	positive integer	determinant-indicated species
vmnew2old	positive integer	vector for transformation of metbaolite indices
vmold2new	positive integer	vector for transformation of metabolite indices
vrnew2old	positive integer	indices for transformation of reaction indices
vrold2new	positive integer	indices for transformation of reaction indices
vs	positive real	extremal pathway - only nonzero fields
vtypesOfCycles	bool	1 if strong, 0 if critical
i	positive integer	index - cycles
typOfCycle	bool	1 if strong, 0 if critical

```
10 vx = [];
11 vy = [];
12 vz = [];
13 vw = [];
```

First, the subnetwork is extracted and transformation vectors (`new2old / old2new`) are constructed for comfortable transformations.

```
14 [S, K, Sl, Sr, vmind, vmnew2old, vmold2new, vrnew2old, vrold2new
   ] = fold2new(ve, vmold, Sbig, Slbig, Srbig, Kbig, emtol);
```

Then, all cycles of metabolites indicated by determinant are found.

```
SmallCycles = fcycFinder(S, Sl, Sr, vmind);
if size(SmallCycles,1) == 0
    typ = 14;
    return
endif
```

Each pair of cycle is checked for links. If there are reactions from one cycle to another and back, then the cycles are merged.

```
vs = ve(find(ve));
Cycles = flinkFinder(SmallCycles,Sl,Sr);
```

Strength of cycles is tested.

```
[Cycles, vtypesOfCycles] = fcycType(Cycles, vs, S, K, emtol);
if size(Cycles,1) == 0
    typ = 14;
    return
endif
```

If there are no strong cycles and at least 2 critical cycles, some of them might be composed of W species.

```
if (size(Cycles,1) > 1) && (vtypesOfCycles'*vtypesOfCycles == 0)
    Cycles = fexitFinder(Cycles, Sl, vmind);
endif

noCycl = size(Cycles,1);
if (noCycl > 1)
    warning("more than 1 strong cycle or critical cycle with
       outflow!");
    CycsSizes = [ zeros(noCycl,1) Cycles ];
    for i=1:noCycl
        CycsSizes(i,1) = size(find(Cycles(i,:)),2);
    endfor
    Cycles = sortrows(CycsSizes)(:,2:size(Cycles,2)+1);
    vx = Cycles(1,:);
    typOfCycle = vtypesOfCycles(1);
    vx = unique(vcycle(find(vcycle)));
else
    vx = Cycles;
    typOfCycle = vtypesOfCycles;
    vx = unique(vx(find(vx)));
endif
if typOfCycle == 0
    [typ, vy, vz] = fdecide1BC(vx, Sl, Sr, vmind);
else
```

```
49      [typ, vz] = fdecide2BC(vx, Sl, Sr, vmind);
50  endif
51  vx = (vmnew2old(vx))';
52  vy = vmnew2old(vy);
53  vz = (vmnew2old(vz))';
54  vw = setdiff(vmold,union(vx,vy))
55  endfunction
```

C.3 Local function `fold2new`

The purpose of this function is to extract the studied subnetwork from the whole system and to construct transformation vectors of indices.

Variables

In line ... occurence of metabolites in rows of Smedium is calculated by multiplying boolean matrix by unit vector.

```
56  function [S, K, Sl, Sr, vmind, vmnew2old, vmold2new, vrnew2old,
       vrold2new] = fold2new(ve, vmold, Sbig, Slbig, Srbig, Kbig,
       emtol)
57  [mbig,rbig] = size(Sbig);
58  vrnew2old = find(ve);
59  vrold2new = ve;
60  vrold2new(vrnew2old) = (1:size(vrnew2old,1))';
61  Smedium = Sbig(:,vrnew2old);
62  Sbool = abs(Smedium) > emtol;
63  vmbool = Sbool * ones(r = size(vrnew2old,1),1);
64  vmnew2old = find(vmbool);
65  vmold2new = zeros(size(vmnew2old));
66  vmold2new(vmnew2old) = (1:size(vmnew2old,1))';
67  S = Smedium(vmnew2old,:);
68  K = Kbig(vmnew2old,vrnew2old);
69  vmind = vmold2new(vmold);
70  Sl = Slbig(vmnew2old,vrnew2old);
71  Sr = Srbig(vmnew2old,vrnew2old);
72  endfunction
```

C.4 Local function `fcycFinder`

Variables

Memory cannot be allocated in the beginning, since theoretical maximum number of paths is `mind!`. I did not implement pseudodynamic allocation to keep the code as

Input variables		
Kbig	real	kinetic matrix of the whole system
Sbig	real	stoichiometric matrix of the whole system
Slbig	real	left stoichiometric matrix of the whole system
Srbig	real	right stoichiometric matrix of the whole system
ve	positive real	extremal pathway (coefficient of reactions)
vmold	positive integer	determinant-indicated species (indices in the whole system)
emtol	positive real	tolerance threshold for floating point equality tests

Output variables		
K	real	kinetic matrix of the studies subsystem
S	real	stoichiometric matrix of the studies subsystem
Sl	real	left stoichiometric matrix of the studies subsystem
Sr	real	rigth stoichiometric matrix of the studies subsystem
vmind	positive integer	determinant-indicated species (indices in the studied subsystem system)
vmnew2old	positive integer	indices for transformation of metabolite indices from indices in studied subsystem to indices in the whole system
vmold2new	positive integer	indices for transformation of metabolite indices from indices in the whole system to indices in studied subsystem
vrnew2old	positive integer	indices for transformation of reaction indices from indices in studied subsystem to indices in the whole system
vrold2new	positive integer	indices for transformation of reaction indices from indices in the whole system to indices in studied subsystem

Local variables		
Sbool	bool	1 if there is the metabolite is produced or consumed by the reaction
Smedium	real	stoichiometric matrix of the studied subsystem with zero vectors corresponding to species not present in the studied subsystem of the whole CRN
vmbool	positive integer	number of reactions in which metabolites participate
mbig	positive integer	number of metabolites in the whole system
rbig	positive integer	number of reactions in the whole system

simple as possible. In theory, exploration of all possible paths is the most computationally demanding procedure in the whole `oscilClasses` program and allocation of eventually large matrices in many cycles does not help at all. Perhaps Floyd's "tortoise and the hare" algorithm would satisfy a genuine computer scientist. However, in our experience this procedure is fast, since most subnetworks are connected by direct paths.

Input variables		
S	real	stoichiometric amtrix
Sl	real	left stoichiometric matrix
Sr	real	right stoichiometric matrix
vmind	positive integer	list of determinant-indicated metabolites i.e. candidates for species of type X
Output variables		
Cycles	positive integer	cycles

Local variables		
Extensions	positive integer	extended paths
TempCycles		
TerminatedCycles	positive integer	terminated paths - the last nonzero number in vector is identical to the first one
vCycle	positive integer	found cycle in raw form
vnz		set of nonzero terms in vCycle
c	positive integer	number of found cycles
i	positive integer	index - size of cycles
j	positive integer	index - found cycles
lengthOfCycles	positive integer	equal to number of indicated species + 1
m	positive integer	number of all metabolites in the system
mind	positive integer	number of determinant-indicated metabolites
numberOfCycles	positive integer	number of found cycles (with repetitions)
r	positive integer	number of reactions in the system
z1	positive integer	index - determinant-indicated metabolites initial nodes of the cycles

The main loop of this function - for each metabolite (z1) extend each of the already found paths (j). The maximum size of a paths is mind, since there are only mind determinant-indicated metabolites. Therefore, another index iis necessary

```
73 function Cycles = fcycFinder(S, Sl, Sr, vmind)
74 [m,r]=size(S);
75 mind = size(vmind,1);
76 Cycles = [];
77 for z1=vmind'
78     TempCycles = zeros(1,mind+2);
79     TempCycles(1,1) = z1;
80     for i=2:(mind+2)
81         TempCyclesNew = [];
82         c = size(TempCycles,1);
83         for j=1:c
84             [Extensions, TerminatedCycles] = fextendCycle(z1,i,
                  TempCycles(j,:),Sl,Sr,vmind);
85             TempCyclesNew = [TempCyclesNew ; Extensions];
```

```
86              Cycles = [Cycles ; TerminatedCycles];
87          endfor
88          TempCycles = TempCyclesNew;
89      endfor
90  endfor
```

The procedure finds each cycle composed of `N` metabolites `N` times since it starts from each of the metabolites. In order delete replicated cycles, they need to be saved in a consistent way by sorting the composing metabolites in each row.

```
91  [numberOfCycles, lengthOfCycles] = size(Cycles);
92  for i=1:numberOfCycles
93      vCycle = Cycles(i,:);
94      vnz = unique(vCycle(find(vCycle)));
95      Cycles(i,:) = [ zeros(1,lengthOfCycles-size(vnz,2)) vnz ];
96  endfor
97  Cycles = unique(Cycles, 'rows');
98  endfunction
```

C.5 Local function `fextendCycle`

This function returns extends a path by checking whether there is a reaction from the last metabolite of the path to any other metabolited indicatedby determinant.

Variables

For all reactions, which the last metabolite of path `vCycle` enters, for all products of this reactions, if the product is indicated by determinant, extend path. If the extending metabolite is the same as the first metabolite `z1`, the path is cyclic. Otherwise, add it to the other extended paths.

```
99  function [Extensions, TerminatedCycles] = fextendCycle(z1,m,
        vCycle,Sl,Sr,vmind)
100 Extensions = [];
101 TerminatedCycles = [];
102 vtempCycle = vCycle;
103 vreact = fr2v(vCycle(m-1),Sl);
104 for ireact = vreact'
105     vprod = fv2p(ireact,Sr);
106     for iprod=vprod' % transpose because for indexes only row
            vestors
107         vtempCycle(m) = iprod;
108         if iprod==z1
109             TerminatedCycles = [ TerminatedCycles ; vtempCycle
                    ];
```

Input variables		
Sl	real	left stoichiometric matrix (studied subsystem)
Sr	real	right stoichiometric matrix (studied subsystem)
vCycle	positive integer	indices of determinant-indicated metabolites in the path
vmind	positive integer	all determinant-indicated metabolites
m	positive integer	number of metabolites in the studied subsystem
z1	positive integer	number of reactions in the studied subsystem
Output variables		
Extensions	positive integer	extended paths
TerminatedCycles	positive integer	newly found cycles
Local variables		
vprod	positive integer	products of selected reaction react into which metabolite m enters
vreact	positive integer	reactions which the last metabolite of the cycle enters as reactant
vtempCycle	positive integer	cycle being extended in current cycle
iprod	positive integer	index - products of the reaction into which metabolite m enters
ireact	positive integer	reaction which the last metabolite of the cycle enters as reactant

```
110         elseif ismember(iprod, vmind)
111             Extensions = [ Extensions ; vtempCycle ];
112         endif
113     endfor
114 endfor
115 endfunction
```

C.6 Local function `flinkFinder`

The purpose of this function is to check whether there are links between the found cycles. If any pair of cycles is link so that there is reaction from one to the other and back, they are merged into 1 cycle.

Variables

The algorithm iteratively merges all pairs of cycles that are found to be linked, then updates set of all cycles and proceeds until no link is found (`while`loop).

```
116 function Cycles = flinkFinder(SmallCycles,Sl,Sr);
117 Cycles = SmallCycles;
118 [numberOfCycles, lengthOfCycles] = size(SmallCycles);
```

Input variables		
Sl	real	left stoichiometric matrix
SmallCycles	positive integer	cycles, some of whose might have connections to the others
Sr	real	rightstoichiometric matrix
Output variables		
Cycles	positive integer	non-connected cycles
Local variables		
UnlinkedCycles	positive integer	cycles for which no connections were found
vCycle1	positive integer	cycle 1
vCycle2	positive integer	cycle 2
vmergedCycle	positive integer	merged cycle in form suitable for being stored in Cycles matrix
vlinked	bool	if 0, cycle has no connection to any other cycle
vwhichlinked	positive integer	indices of linked cycles
j	positive integer	index - cycle 2
i	positive integer	index - cycle 1
lengthMerged	positive integer	number of metabolites in merged cycle
lengthOfCycles	positive integer	number of determinant-indicated metabolites + 2
numberOfCycles	positive integer	number of cycles in Cycles
stop	bool	1 if there have been no new links found controls whileloop

```
119 if numberOfCycles==1
120     return
121 endif
122 stop = 0;
123 while stop==0
124     [numberOfCycles, lengthOfCycles] = size(Cycles);
125     vlinked = zeros(numberOfCycles,1);
126     stop = 1;
127     SmallCycles = Cycles;
128     Cycles = [];
129     for i=1:(numberOfCycles-1)
130         for j=(i+1):numberOfCycles
131             vCycle1 = SmallCycles(i,:);
132             vCycle2 = SmallCycles(j,:);
133             if fareLinked(vCycle1,vCycle2,Sl,Sr)
134                 vmergedCycle = unique([vCycle1(find(vCycle1))
                      vCycle2(find(vCycle2))]);
135                 lengthMerged = size(vmergedCycle, 2);
```

```
136                     vmergedCycle = [zeros(1,lengthOfCycles-
                           lengthMerged) vmergedCycle];
137                     Cycles = [ Cycles ; vmergedCycle ];
138                     vlinked(i) = 1;
139                     vlinked(j) = 1;
140                     stop = 0;
141                 endif
142             endfor
143         endfor
144         vwhichlinked=find(vlinked==0);
145         UnLinkedCycles = SmallCycles(vwhichlinked,:);
146         if size(UnLinkedCycles,1)>1
147             Cycles = [ Cycles; UnLinkedCycles ];
148         endif
149         Cycles = unique(Cycles, 'rows');
150         if size(Cycles,1) == 1
151             stop = 1;
152         endif
153     endwhile
154     endfunction
```

C.7 Local function `fareLinked`

This function returns 1 if there exists

1. reaction in which any of metabolites of the first cycle enters as reactants and any of the metabolites of the second cycle is produced.

2. reaction in which any of metabolites of the second cycle enters as reactants and any of the metabolites of the first cycle is produced.

Variables

The algorithm presumes all cycle pairs to be disconnected. For each metabolite in cycle 1, for each reaction which the metabolite enters as reactant save products of this reaction. If any of the sampled products is member of cycle 2, then there is a link from cycle 1 to cycle 2.

```
155 function yesORno = fareLinked(vCycle1,vCycle2,Sl,Sr)
156 vCyc1 = vCycle1(find(vCycle1));
157 vCyc2 = vCycle2(find(vCycle2));
158 j2d = 0;
159 d2j = 0;
160 vlistOfProducts1 = [];
```

Input variables		
Sl	real	left stoichiometric matrix
Sr	real	right stoichiometric matrix
vCycle1	positive integer	cycle 1 (indices)
vCycle2	positive integer	cycle 2 (indices)
Output variables		
yesORno	bool	1 if the cycles 1 and 2 are linked
Local variables		
vCyc1	positive integer	cycle 1 without complementing zeros
vCyc2	positive integer	cycle 1 without complementing zeros
vlistOfProducts1	positive integer	products of all reactions originating in cycle 1
vlistOfProducts2	positive integer	products of all reactions originating in cycle 2
vprod	positive integer	metabolite indices
vreact	positive integer	reaction indices
d2j	bool	1 if there is a link from cycle 2 to cycle 1
imetab	positive integer	index - metabolites
ireact	positive integer	index - reactions
j2d	bool	1 if there is a link from cycle 1 to cycle 2

```
for imetab=vCyc1
    vreact = fr2v(imetab,Sl);
    for ireact=vreact'
        vprod = fv2p(ireact,Sr);
        vlistOfProducts1 = [ vlistOfProducts1 ; vprod ];
    endfor
endfor
if size(intersect(vlistOfProducts1, vCyc2),2)>0
    j2d = 1;
endif
```

Analogically, test for reactions from cycle 2 to cycle 1.

```
vlistOfProducts2 = [];
for imetab=vCyc2
    vreact = fr2v(imetab,Sl);
    for ireact=vreact'
        vprod = fv2p(ireact,Sr);
        vlistOfProducts2 = [ vlistOfProducts2 ; vprod ];
    endfor
endfor
if size(intersect(vlistOfProducts2, vCyc1),2)>0
    d2j = 1;
endif
yesORno = j2d && d2j;
```

```
endfunction
```

C.8 Local function `fcycType`

The purpose of this algorithm is to assign each cycle its "strength" - 1 if it is strong cycle and 0 if it is critical cycle. All weak cycles are omitted.

Variables

Input variables		
K	real	kinetic matrix of the whole system
S	real	stoichiometric matrix of the whole system
OriginalCycles	positive integer	
vs	positive real	extremal pathway
emtol	positive real	tolerance for floating point equality tests
Output variables		
Cycles	positive integer	strong and critical cycles
vtypesOfCycles	bool	1 if cycle is strong, 0 if critical
Local variables		
B	real	matrix defined in Equation11
Bsmall	real	principal subdeterminant of B
CycsAndCosts	positive integer	cycles with their types (first column)
vCyc	positive integer	list of cycle members, no zeros
vCycle	positive integer	list of cycle members; zeros are present in order to store the cycle in 1 matrix with the ones
detBsmall	real	determinant of Bsmall
i	positive integer	index - cycles
lengthOfCycles	positive integer	equal to number of determinant-indicated metabolites + 2
numberOfCycles	positive integer	number of all cycles

Critical cycle is defined based on determinant of metrix `Bsmall`, which has to be zero. However, floating point operations may result in inaccuracies. Therefore, test for zero equality is replaced by test for absolute value, which has to be under a specified threshold.

```
function [Cycles, vtypesOfCycles] = fcycType(OriginalCycles, vs,
    S, K, emtol);
Cycles = [];
vtypesOfCycles = [];
[numberOfCycles,lengthOfCycles]=size(OriginalCycles);
B = -S*diag(vs)*K'
for i=1:numberOfCycles
```

```
    vCycle = OriginalCycles(i,:);
    vCyc = unique(vCycle(find(vCycle)));
    Bsmall = B(vCyc,vCyc);
    detBsmall = det(Bsmall);
    if abs(detBsmall) < emtol
        Cycles = [ Cycles ; vCycle ];
        vtypesOfCycles = [ vtypesOfCycles ; 0 ];
    elseif detBsmall < -emtol
        Cycles = [ Cycles ; vCycle ];
        vtypesOfCycles = [ vtypesOfCycles ; 1 ];
    endif
endfor
```

Sorting the cycles according to their strength.

```
if size(vtypesOfCycles,1)>0
    CycsAndCosts = [vtypesOfCycles Cycles];
    CycsAndCosts = flipud(sortrows(CycsAndCosts));
    vtypesOfCycles = CycsAndCosts(:,1);
    Cycles = CycsAndCosts(:,2:lengthOfCycles+1);
endif
endfunction
```

C.9 Local function `fexitFinder`

We cannot exclude possibility occurence of critical cycles of determinant-indicated metabolites without an exit reactions. These would be composed of type W species, which are indicated by determinant. This function finds exit reactions of all the previously found critical cycles.

Variables

The algorithm presumes all cycles to lack exit reaction until it finds one. Then it literally checks for Y species by finding it. This whole operation seems to be redundant, since `fdecide1BC`must also find Y However, the check must be done in advance and outsourcing of this procedure simplifies the main function and `fdecide1BC`.

```
function Cycles = fexitFinder(Cycles, Sl, vmind)
[numberOfCycles,lengthOfCycles]=size(Cycles);
vnoExit = ones(numberOfCycles, 1);
vlistReactants = [];
for iCycle=1:numberOfCycles
        vCycle = Cycles(iCycle,:);
        vCyc = vCycle(find(vCycle));
        for imetab=vCyc
```

Input variables		
Cycles	positive integer	found critical cycles
Sl	real	left stoichiometric matrix (studied subsystem)
vmind	positive integer	determinant-indicated species (indices)
Output variables		
Cycles	positive integer	cycles possessing exit reactions
Local variables		
vCyc	positive integer	without zeros
vCycle	positive integer	line in Cycles corresponding to 1 cycle
vnoExit	bool	1 if cycle has no exit and is to be deleted
vreactants	positive integer	metabolite indices
vlistReactants	positive integer	metabolite indices
vreact	positive integer	metabolite indices
vreactants	positive integer	reaction indices
vy	positive integer	indices of species of type Y
iCycle	positive integer	index - cycles
imetab	positive integer	index - metabolites
ireaction	positive integer	index - reactions
numberOfCycles	positive integer	number of all cycles
lengthOfCycles	positive integer	equal to number of determinant-indicated metabolites + 2

```
217             vreact = fr2v(imetab,Sl);
218             if size(vreact,1) > 0
219                 for ireaction=vreact'
220                     vreactants = fv2r(ireaction,Sl);
221                     vlistReactants = [vlistReactants; vreactants
                            ];
222                 endfor
223                 vy = setdiff(intersect(vmind,vlistReactants),
                       vCyc);
224                 if size(vy,2)>0
225                     vnoExit(iCycle) = 0;
226                 endif
227             endif
228         endfor
229 endfor
230 Cycles(find(vnoExit),:) = [];
231 endfunction
```

C.10 Local function `fdecide1BC`

This function dicides between category 1B and 1C. It searches for Y species and then for the first species of type Z, which is produced as by-product by the autocatalytic cycle and there is a pathway of reactions from Z to Y. All the species on this pathaway are classified as of type Z too. The length of this pathway (number of Z species) must be at least 1 and at most equal to the number of species not indicated by determinant. Lists of Y species and Z species are generally vectors, but excluding rare pathological cases Y is scalar.

Variables

Input variables		
`Sl`	real	left stoichiometric matrix of the extremal pathway
`Sr`	real	right stoichiometric matrix of the extremal pathway
`vcycle`	positive integer	indices of X species
`vmind`	positive integer	indices of all species indicated by determinant
Output variables		
`vy`	positive integer	list of Y species indices
`vz`	positive integer	list of Z species indices
`type`	(12|13)	determined type
Local variables		
`vmNind`	positive integer	metabolites not indicated by determinant
`vlistReactants`	positive integer	candidates for Y species
`vlistReactantsy`	positive integer	reactants in reactions producing Y
`vlistReactions`	positive integer	exit reactions
`vlistReactionsy`	positive integer	reaction in which Y is produced
`vproducts`	positive integer	reaction indices
`vreact`	positive integer	metabolite indices
`vreactants`	positive integer	metabolite indices
`vzx`	positive integer	initial points for Z species search
`vzy`	positive integer	terminal points for Z species search
`imetab`	positive integer	index - metabolites
`ireaction`	positive integer	index - reactions

Metabolite of type Y is identified. First, reactions in which X species are both reactants and products are sampled.

```
232 function [typ, vy, vz] = fdecide1BC(vcycle, Sl, Sr, vmind)
233 vmNind = setdiff(1:size(Sl, 1),vmind);
234 vlistReactions = [];
235 for imetab=vcycle
236     vreact = fr2v(imetab,Sl);
```

```
    for ireaction=vreact'
        vproducts = fv2p(ireaction,Sr);
        if size(intersect(vcycle,vproducts),2)==0
            vlistReactions = [vlistReactions; ireaction];
        endif
    endfor
endfor
```

Reactants entering these reactions which are not involved in the autocatalytic cycle are of type Y. Rare pathological case of more than 1 Y is signalized.

```
vlistReactants = [];
for ireact=vlistReactions'
    vreactants = fv2r(ireact,Sl);
    vlistReactants = [vlistReactants; vreactants];
endfor
vy = setdiff(intersect(vlistReactants, vmind), vcycle);
if size(vy,2) > 1
    warning("more metabolites of type Y");
endif
```

Find all species not indicated by determinant from which Y is produced. In the first `for` loop, all reactions where Y is created are sampled. In the second `for` loop, all reactants entering these reactions are sampled. Possibility of Y production directly as a by-product of autocatalytic cycle is forbidden. Y species must be produced via a pathway of Z species. `vzy`means literally vector of (candidate) Z species determined based on species Y.

```
vlistReactionsy = [];
vlistReactantsy = [];
for imetab=vy
    vreact = fp2v(imetab,Sr);
    vlistReactionsy = [vlistReactionsy; vreact];
endfor
for ireact=vlistReactionsy'
    vreactant = fv2r(ireact,Sl);
    vlistReactantsy = [vlistReactantsy; vreactant];
endfor
vzy = setdiff(vlistReactantsy,vmind);
```

All the species produced by the critical cycle as by-products which are not indicated by determinants are sampled. Produced as a by-product means that they are products of some of the reactions involved in the critical cycle.

```
vzx = [];
for imetab=vcycle
```

```
266     vreact = fr2v(imetab,Sl);
267     for ireaction=vreact'
268         vproducts = fv2p(ireaction,Sr);
269         if size(intersect(vcycle,vproducts),2)>0
270             vzx = [vzx; setdiff(vproducts,vcycle)'];
271         endif
272     endfor
273 endfor
```

Finally, all pathways from non-indicated metabolites originating in the critical cycle are explored. If any of the pathways leads to Y, these species are of type Z.

```
274 [yesORno, vz] = fZFinder(vzx, vmind, vmNind, vzy, Sl, Sr)
275 if yesORno
276     typ = 12;
277 else
278     typ = 13;
279 endif
280 endfunction
```

C.11 Local function `fdecide2BC`

This function decides between category 2B and 2C.

Variables

Input variables		
`Sl`	real	left stoichiometric matrix of the extremal pathway
`Sr`	real	right stoichiometric matrix of the extremal pathway
`vcycle`	positive integer	indices of X species
`vmind`	positive integer	indices of all species indicated by determinant
Output variables		
`vz`	positive integer	list of Z species indices
`type`	(22|23)	determined type
Local variables		
`vlistReactants`	positive integer	reactnts in exit reactions
`vlistReactions`	positive integer	possible exit reactions
`vreact`	positive integer	metabolite indices
`vzx`	positive integer	initial points for Z species search
`vzz`	positive integer	terminal points for Z species search
`imetab`	positive integer	index - metabolites
`ireact`	positive integer	index - reactions
`numberOfMetabs`	positive integer	number of all metabolites in the studied subsystem

The algorithm is different from that in function `fdecide1BC` in that no species of type Y is present. Therefore, species taking part in exit reaction is not indicated by determinant. First, all species taking part in exit reactions are identified.

```
function [typ, vz] = fdecide2BC(vcycle, Sl, Sr, vmind)
vy = []
vlistReactions = [];
for imetab=vcycle
    vreact = fr2v(imetab,Sl);
    vlistReactions = [vlistReactions; vreact];
endfor
vlistReactants = [];
for ireact=vlistReactions'
    vreact = fv2r(ireact,Sl);
    vlistReactants = [vlistReactants; vreact];
endfor
vzz = setdiff(vlistReactants, vmind);
```

In analogy with `fdecide1BC`, all the by-products of autocatalytic cycle are sampled.

```
vzx = [];
numberOfMetabs = size(Sl, 1);
vmNind = setdiff(1:numberOfMetabs,vmind);
for react=vlistReactions'
    vprod = fv2p(react,Sr)
    if (size(intersect(vprod,vmind))>0) && (size(setdiff(vprod,
       vmind),2)>0)
        vzx = [vzx; (setdiff(vprod,vmind))];
    endif
endfor
```

Similarly to `fdecide1BC` , all paths from autocatalytic cycle need to be explored.

```
[yesORno, vz] = fZFinder(vzx, vmind, vmNind, vzz, Sl, Sr)
if yesORno
    typ = 22;
else
    typ = 23;
endif
endfunction
```

C.12 Local function `fZFinder`

`fZFinder`explores all the pathways from a specified set of metabolites. This subroutine is very similar to `fcycFinder`. There are two main differences. First, `fZFinder`explores pathways of species not indicated by determiant. Second, search is successful if one

of the defined terminal points is reached, in contrast to `fcycFinder`. Similarly to `fcycFinder`, `fZFinder`has its own subroutine for path extension.

Variables

Input variables		
`Sl`	real	left stoichiometric matrix of the extremal pathway
`Sr`	real	right stoichiometric matrix of the extremal pathway
`vmind`	positive integer	all species indicated by determinant
`vmNind`	positive integer	all species not indicated by determinant
`vzx`	positive integer	set of indices of species not indicated by determinant
`vzz`	positive integer	set of indices of species not indicated by determinant
Output variables		
`vz`	positive integer	list of indices of Z species
`yesORno`	bool	1 if there is a path from `vzx`to `vzz`
Local variables		
`ExtendedPaths`	positive integer	all paths created by `fextendPath` by extendeding 1 path
`Paths`	positive integer	all paths starting in set `vzx`
`TempPaths`	positive integer	paths changing each iteration
`vpath`	positive integer	one path beginning in one of vzx metabolites
`extPathsNumber`	positive integer	number of extended paths extended by one metabolite from one shorter path
`i`	positive integer	index - length of paths
`j`	positive integer	index - all metabolites not indicated by determinant
`k`	positive integer	index - paths
`numberNind`	positive integer	number of metabolites not indicated by determinant

```
function [yesORno, vz] = fZFinder(vzx, vmind, vmNind, vzz, Sl,
   Sr)
yesORno = 0;
vz = [];
if size(vzx,1)==0
    return;
endif
numberNind = size(vmNind,2);
Paths = zeros(size(vzx,1),numberNind);
Paths(:,1) = vzx;
for i=2:numberNind
    TempPaths = [];
```

```
    for j=1:size(Paths,1)
        ExtendedPaths = fextendPath(i,Paths(j,:),Sl,Sr,vmind,
           vmNind);
        extPathsNumber = size(ExtendedPaths,1);
        for k=1:extPathsNumber
            vpath = ExtendedPaths(k,:);
            if ismember(vpath(i),vzz)
                yesORno = 1;
                vz = unique(vpath(find(vpath)));
                return
            endif
        endfor
        TempPaths = [ TempPaths ; ExtendedPaths ];
    endfor
    Paths = TempPaths;
endfor
endfunction
```

C.13 Local function `fextendPath`

This function extends paths for `fZfinder`. It is similarl to `fetendCycle`. It gets a path of species not indicated by determinant and returns all its extensions.

Variables

Input variables		
`Sl`	real	left stoichiometric matrix (studied subsystem)
`Sr`	real	right stoichiometric matrix (studied subsystem)
`vmind`	positive integer	determinant-indicated metabolites
`vmNind`	positive integer	metabolites not indicated by determinant
`vpath`	positive integer	path to be extended
`i`	positive integer	length of input path + 1
Output variables		
`ExtendedPaths`	positive integer	extended paths
Local variables		
`vprod`	positive integer	products of a reaction (output of `fv2p`
`vreact`	positive integer	all reactions of a metabolite where the metabolite enters as a reactant
`iprod`	positive integer	index - metabolites
`ireact`	positive integer	index - reactions

```
function ExtendedPaths = fextendPath(i,vpath,Sl,Sr,vmind,vmNind)
ExtendedPaths = [];
if vpath(i-1) == 0
    return;
endif
vreact = fr2v(vpath(i-1),Sl);
for ireact = vreact'
    vprod = fv2p(ireact,Sr);
    for iprod=vprod'
        vpath(i) = iprod;
        if ismember(iprod, vmNind)
            ExtendedPaths = [ ExtendedPaths ; vpath ];
        endif
    endfor
endfor
endfunction
```

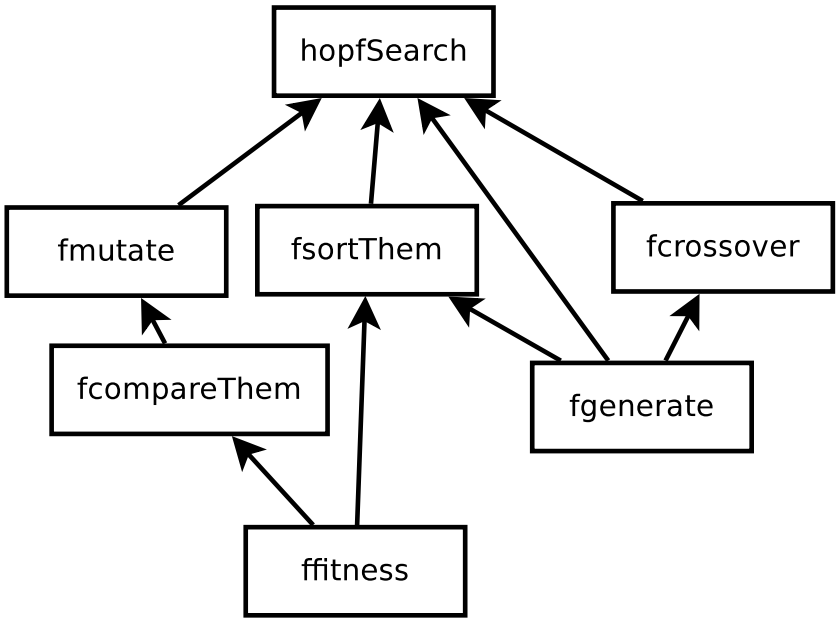
C.14 Local functions `fv2r`, `fv2p`, `fp2v`and `fr2v`

These extensively used simple functions perform the operations of network exporation. Each of them contains only 1 local variable, which has meaning of index of the input reaction or metabolite.

```
function vreactions = fr2v(metab,Sl)
irow = (Sl(metab,:))';
vreactions = find(irow > 0);
endfunction

function vreactions = fp2v(metab,Sr)
irow = (Sr(metab,:))';
vreactions = find(irow > 0);
endfunction

function vprod = fv2p(react,Sr)
icol = Sr(:,react);
vprod = find(icol > 0);
endfunction

function vreact = fv2r(react,Sl)
icol = Sl(:,react);
vreact = find(icol > 0);
endfunction
```

D Documentation for the function `hopfSearch`

D.1 Dependency diagram



D.2 Main function

Genetic programming approach to finding Hopf bifurcation. Evolution diagram is in Figure 3.1.5.

It is important to mention here, that input matrix $\mathbf{B}$ is here defined as

$$\mathbf{B} = \boldsymbol{\nu}\ \mathrm{diag}(\mathbf{e})\ \boldsymbol{\kappa}^{\mathrm{T}} \tag{12}$$

with no minus sign unlike in Equation 11.

Variables

```
function [vbestConc, yesORno] = hopfSearch(B, vmind, vpars,
   popSize, noGenerations, mutate, minNind, maxNind, minInd,
   maxInd, offspringSize, enrichSize)
if nargin == 2
    vpars = [0.1 100 10 30 10 20 10];
    mutate = 0.05;
    noGenerations = 5000;
    popSize = 200;
    minNind = 0.1;
    maxNind = 1;
    minInd = 10;
    maxInd = 100;
```

Input variables		
B	real	matrix **B**
vmind	positive integer	indices of essential species
vpars	real	parameters for fitness function
enrichSize	positive integer	number of newly generated individuals for enriching genofond
maxInd	positive real	maximum inverse concentration of essential species
maxNind	positive real	maximum inverse concentration of non-essential species
minInd	positive real	minimum inverse concentration of essential species
minNind	positive real	minimum inverse concentration of non-essential species
mutate	real	parameter for mutation
noGenerations	positive integer	maximum number of generations
offspringSize	positive integer	size of population after crossover before selection
popSize	positive integer	size of population (before mutations)
Output variables		
vbestConc	positive real	vector of inverse concentrations closest to Hopf bifurcation point
yesORno	bool	found supercritical Hopf bifurcation
Local variables		
AfterMutation	positive real	survivors of mutation test
Population	positive real	survivors of previous selections
LotsOfOffspring	positive real	offspring before selection
i	positive integer	index - generations

```
11      offspringSize = 400;
12      enrichSize = 20;
13  endif
14  yesORno = 0;
15  sizeofB = size(B,1);
16  Population = fgenerate(popSize, sizeofB, vmind, minNind, maxNind
        , minInd, maxInd);
17  for i=1:noGenerations
18      AfterMutation = fmutate(Population, mutate, vpars, B);
19      LotsOfOffspring = fcrossover(AfterMutation, offspringSize,
            enrichSize, vmind, minNind, maxNind, minInd, maxInd);
20      [yesORno, Population] = fsortThem(LotsOfOffspring, popSize,
            vpars, B);
21      if yesORno == 1
22          vbestConc = Population(1,:);
23          break;
24      endif
25  endfor
```

```
endfunction
```

D.3 Local function `fmutate`

This functions mutates each vector in population and then evaluates change in the fitness. Better individual of the pair original - mutant is selected.

Variables

Input variables		
`B`	real	matrix **B**
`Populations`	positive real	input population
`vpars`	real	parameters for fitness function
`mutate`	positive real	parameter of mutaiton fucntion
Output variables		
`AfterMutation`	positive real	output population - improved
Local variables		
`BasisPart`	positive real	contribution to mutation matrix
`Mutants`	positive real	mutated population
`MutMultiply`	real	mutation matrix
`RandomPart`	positive real	random contribution to mutation matrix

```
function AfterMutation = fmutate(Population, mutate, vpars, B)
[m,n]=size(Population);
BasisPart = (1-0.5*mutate)*ones(m,n);
RandomPart = mutate*rand(m,n);
MutMultiply = BasisPart + RandomPart;
Mutants = Population .* MutMultiply;
AfterMutation = fcompareThem(Population, Mutants, vpars, B);
endfunction
```

D.4 Local function `fcrossover`

This functions generates offspring by linear combinations of parents. Parents are selected randomly and for each pair, and new vector is calculated as

$$v_{new} = av_1 + (1-a)v_2$$

where a is a random number between 0 and 1, v_1 and v_2 are the parents.

Variables

The loop in this function is easy to vectorize, I here did not do so for clarity.

Input variables		
Survivors	positive real	input survivors population
vmind	positive integer	indices of essential species
enrichSize	positive integer	number of individuals enriching genofond
maxInd	positive real	maximum inverse concentration of essential species
maxNind	positive real	maximum inverse concentration of non-essential species
minInd	positive real	minimum inverse concentration of essential species
minNind	positive real	minimum inverse concentration of non-essential species
offspringSize	positive integer	size of offspring population
Output variables		
Offspring	positive real	output offspring population
Local variables		
Enriching	positive real	population of newly generated vectors
Parent1	positive real	parental population 1
Parent2	positive real	parental population 2 (randomly chosen)
WithNew	positive real	population of survivors + newly generated vectors
i	positive integer	index - offspring
m	positive integer	size of input population
n	positive integer	size of a inverse concentration vector
lc	positive real (0-1)	coefficient of linear combination

```
35 function Offspring = fcrossover(Survivors, offspringSize,
      enrichSize, vmind, minNind, maxNind, minInd, maxInd)
36 [m,n]=size(Survivors);
37 Enriching = fgenerate(enrichSize, n, vmind, minNind, maxNind,
      minInd, maxInd);
38 Parent1 = zeros(offspringSize,n);
39 Parent2 = zeros(offspringSize,n);
40 Offspring = zeros(offspringSize,n);
41 WithNew = [Survivors; Enriching];
42 for i=1:offspringSize
43     Parent1(i,:) = Survivors(mod(i,m)+1,:);
44     Parent2(i,:) = WithNew(ceil(rand(1)*(m+enrichSize)),:);
45     lc = rand(1);
46     Offspring(i,:) = lc*Parent1(i,:) + (1-lc)*Parent2(i,:);
47 endfor
48 endfunction
```

D.5 Local function `fSortThem`

This function sorts all the vectors in new generation according to their fitness and selectes those with best fitness.

Variables

Input variables		
`B`	real	matrix **B**
`LotsOfOffspring`	positive real	input offspring population
`vpars`	real	parameters for fitness function
`popsize`	positive integer	size of survivor population
Output variables		
`Population`	positive real	output survivor population
`yesORno`	bool	was found super critical Hopf bifurcation?
Local variables		
`PopWCosts`	real	offspring population with futness in 1st column
`SortedPop`	positive real	offspring population sorted according to fitness
`vfitness`	real	fitness functions for offspring
`i`	positive integer	index - offspring
`m`	positive integer	size of input population
`n`	positive integer	size of an individual vector
`theBestOne`	real	best fitness function value
`threshold`	real	threshold for Hopf bifurcaiton

```
49 function [yesORno, Population] = fsortThem(LotsOfOffspring,
      popSize, vpars, B)
50 threshold = vpars(2) + vpars(4) + vpars(6);
51 [m,n] = size(LotsOfOffspring);
52 vfitness = zeros(m,1);
53 for i=1:m
54     vfitness(i) = ffitness(LotsOfOffspring(i,:), vpars, B);
55 endfor
56 PopWCosts = [ vfitness LotsOfOffspring ];
57 SortedPop = flipud(sortrows(PopWCosts));
58 Population = SortedPop(1:popSize, 2:(n+1));
59 theBestOne = SortedPop(1,1)
60 if theBestOne >= threshold
61     yesORno = 1;
62 else
63     yesORno = 0;
64 endif
65 endfunction
```

D.6 Local function `fcompareThem`

This function is used to compare each individual with its mutant. Individual with better fitness is selected.

Variables

Input variables		
`B`	real	matrix **B**
`Population`	positive real	input population of survivors
`vpars`	real	parameters for fitness function
`Mutants`	positive real	input population of mutant
Output variables		
`Survivors`	positive real	better of mutant — original survivor
Local variables		
`i`	positive integer	index - population
`m`	positive integer	size of population

```
66 function Survivors = fcompareThem(Population, Mutants, vpars, B)
67 m = size(Population,1);
68 Survivors = Population;
69 for i=1:m
70     if ffitness(Population(i,:),vpars, B) < ffitness(Mutants(i
       ,:),vpars, B)
71         Survivors(i,:) = Mutants(i,:);
72     endif
73 endfor
74 endfunction
```

D.7 Local function `fgenerate`

Variables

Vectors are generated randomly with uniform distribution between maximum and minimum values (default or specified on input). Different maximum and minimum values are for essential and non-essential species.

```
75 function Population = fgenerate(popSize, sizeofB, vmind, minNind
       , maxNind, minInd, maxInd);
76 BasisPart = minNind*ones(popSize, sizeofB);
77 RandomPart = (maxNind-minNind)*rand(popSize, sizeofB);
78 Population = BasisPart + RandomPart;
79 vbasisPart = ones(popSize,1)*minInd;
```

Input variables		
maxInd	positive real	maximum inverse concentration of essential species
maxNind	positive real	maximum inverse concentration of non-essential species
minInd	positive real	minimum inverse concentration of essential species
minNind	positive real	minimum inverse concentration of non-essential species
popSize	positive integer	desired size of generated population
sizeofB	positive integer	size of matrix **B**
vmind	positive integer	indices of essential species
Output variables		
Population	positive real	output population
Local variables		
BasisPart	positive real	non-random contribution to generated population
RandomPart	positive real	random contribution to generated population
vbasisPart	positive real	analogue of row in BasisPart but for essential species
i	positive integer	index - essential species

```
80 for i=vmind
81     Population(:,i) = vbasisPart + rand(popSize, 1)*(maxInd-
          minInd);
82 endfor
83 endfunction
```

D.8 Local function `ffitness`

Variables

Fitness function is given in Equation ... It is based on evaluation of eigenvalues, which is the time determining step of the algorithm.

```
84 function fitness = ffitness(vh, vpars, B)
85 m = size(B,1);
86 thresholdRatio = vpars(1);
87 fitness = 0;
88 H = diag(vh);
89 Jacobi = B*H;
90 veig = eig(Jacobi);
91 vim = imag(veig);
92 vre = real(veig);
```

Input variables		
B	real	matrix **B**
vh	positive real	inverse concentration vector
vpars	real	vector of parameters for fitness function
Output variables		
fitness	real	fitness function fo vector vh
Local variables		
H	diagonal real	diagonal matrix of inverse concentrations
Jacobi	real	Jacobian matrix
veig	complex	vector of eigenvalues
veigDiffs	positive real	distances between real eigenvalues
vim	real	vector of real parts of eigenvalues
vimIms	positive real	real parts of complex eigvals
vimInds	positive integer	indices of eigenvalues with non-zero imaginary parts
vposRes	positive integer	indices of positive real eigenvalues
vposs	positive real	positive real eigenvalues
vratios	positive real	ratios of real and imaginary parts
vre	real	vector of imaginary parts of eigenvalues
vreIms	positive real	imaginary parts of complex eigenvaluesvals
vreRes	real	pure real eigenvalues
m	positive integer	size of matrix B
minDiff	positive real	minimum distance between real eigenvalues
rri	positive real	positive real eigenvalue
theSmallestOne	positive real	complex eigenvalue with real part closest to zero
thresholdRatio	positive real	minimum ratio of imaginary and real aprt

```
93  vimInds = find(vim);
94  vreRes = vre(find(vim==0));
95  if size(vimInds)>0
96      fitness += vpars(2);
97      vreIms = abs(vre(vimInds));
98      vimIms = abs(vim(vimInds));
99      vratios = vreIms ./ vimIms;
100     theSmallestOne = min(vratios);
101     if theSmallestOne < thresholdRatio
102         fitness += vpars(6);
103     else
104         fitness += vpars(7)*exp(-theSmallestOne);
105     endif
106 else
107     veigDiffs = ([ 0 ; vre ] - [ vre; 0 ])(2:m);
```

```
108     minDiff = min(abs(veigDiffs));
109     fitness += vpars(3)*exp(-minDiff);
110 endif
111 vposRes = find(vreRes>0);
112 if size(vposRes,1) == 0
113     fitness += vpars(4);
114 else
115     vposs = vreRes(vposRes);
116     for rr=vposs'
117         fitness += (-vpars(5))*exp(rr);
118     endfor
119 endif
120 endfunction
```