CHALMERS

Modeling and simulation of anaerobic manure digestion into biogas

Master's Thesis in Applied Physics

OSKAR DANIELSSON

Department of Physics & Engineering Physics
Applied Physics
Chalmers University of Technology
Gothenburg, Sweden 2014
Master's Thesis 2014:5

Sammanfattning

I dagens läge är det allt dyrare att ta hand om gödsel från hästar, men också från djurparker. Detta projekt, som är en del i ett större samarbetsprojekt, syftar till att ta fram metoder för att test samrötning mellan djurgödsel och avloppsslam till biogas.

I projektet är en modell byggd baserad på *Anaerobic digestion model No. 1* (ADM1). Till denna standardmodell har sedan ytterligare ekvationer lagts och flera simuleringar byggts, med syftet att ta fram verktyg för att undersöka möjligheten till samrötning mellan gödsel och slam. Intressanta aspekter så som optimal blandning och parameterberoende är undersökt.

Arbetet är genomfört i `MATLAB`, men för att öka nyttan med projektet är också ett fristående grafiskt användarinterface byggt. Detta låter personer utan tillgång till eller kunskap om `MATLAB` använda den byggda modellen.

Abstract

Today it becomes more and more expensive to take care of produced manure, not only from horses, but also from zoos. This project, that is a small part of a large collaboration, aims to develop methods to test co digestion of manure and waste water sludge to biogas.

In the project a model based on the *Anaerobic Digestion Model No. 1* (ADM1) is built. To this standard model further equations have been added and simulations have been build, with the aim to develop tools to investigate the possibilities to co digest manure and sludge. Interesting aspects such as optimal mixture and parameter dependence are investigated.

The work is performed in `MATLAB`, but to make wider use of the project result, a stand alone graphical user interface is also created. This lets peoples without knowledge about or access to `MATLAB` use the build model.

Acknowledgements

This Master's Thesis has been performed at Chalmers University of Technology in collaboration with SP Energiteknik in Lund. The work has been both fun and interesting. I would like to send many thanks to my main supervisor Magnus Karlsteen at Chalmers that has taken care of everything from missed spelling, to supporting me in everything.

I will also thank my other supervisor at SP, Karin Willquist, Who has taken her time to explain every biological and chemical system so that also a physicist can understand them.

Oskar Danielsson, Gothenburg May 26, 2014

CHALMERS

Contents

1

Theory and Resources

1.1 Background

There are exceptionally many horses in Sweden. A lot of those horses are grouped around cities leading to problems in disposing the manure produced. In those areas of high density of horses there are relative small possibilities in spreading the manure at an own field, due to that the horses are located in or in direct vicinity to the cities and the owners don't have access to a field for spreading. So the only solution for a lot of horse owning people and organizations are to buy manure disposing services from external.

The horse manure produced during a year in Sweden represents 4.3 TWh [1]. Even with this energy potential the cost to get rid of the produced manure is approximately 1k SEK per year and horse [2]. A good solution would be to find a use for this energy potential meanwhile the cost for disposing manure in the horse business would decrease. Possibly not to the extent that one gets reimbursement to get rid of the manure, but as far as one do not need to pay for disposing the manure.

One way of disposing the manure and taking care of the energy potential it contains would be to produce biogas out of the manure. Another solution would be to burn the manure. It is hard to say which one of those methods that is the best. This master thesis focuses on producing biogas out of the horse manure and further more also on manure from zoos. The thesis is focused on modeling the digestion process in existing biogas plants that can be used for testing and verifying what happens when manure is added to the waste water sludge.

Furthermore, today the society is striving for replacing fossil based energy sources, and here the horse manure can play an important role. By digesting the manure to biogas the manure can replace part of the fossil raw material used for energy production in the society today.

1.2 Background

1.2.1 General

The horse business in Västra Götalands län close to Gothenburg started to investigate why the costs connected to disposing produced manure were so high. Would it be possible to do some changes so that they could get rid of the manure without needing to pay for disposing it? They thought about it and started a group effort between a lot of different small horse businesses in the region. From this collaboration they found out that they all had many problems and various solutions. But some of the problems were not unique, as for example the problem with disposing the manure. The solutions perhaps were different but the problems were still there.

This master thesis is not directly coupled to the conclusion drawn in the report from the small horse businesses [2]. Instead it is part of a large collaboration between *Västragötalandsregionen*, *SP*, *Borås Energi och Miljö AB*, *Borås Ridskola, Borås djurpark* and *Chalmers University of Technology* that has as the purpose to find out what to do with produced manure. Not only by horses, but also from animals at zoos that usually also are situated close to cities. The idea is to compare the pros and cons between digesting the manure in existing waste water biogas plants, and burning it to get heat instead. The idea of the thesis is to simulate the outcome of the digesting in order to be able to say whether a digesting process is preferable or not. The results are then to be compared to results from measurements. The comparison to measurements is not part of the present thesis, but it is done as a part of the collaboration in the larger project.

1.2.2 Conditions

At end of the thesis should the following conditions be fulfilled

- The mathematical anaerobic digestion model no. 1 ADM1 (with small changes) should be used for simulations.
- The robustness of the implemented model should be possible to test.
- The model should be tested on typical data.
- The model should be possible to fit to Gässlösa Avloppsreningsverk in Borås.
- The result of the simulation should be presented in a way so that it could form the basis for implementation in other biogas plants besides Gässlösa in Borås.
- A software tool that can be used of peoples not experienced with programming and simulations should be compiled on top of the mathematical model.

1.2.3 Features

Features to include in the solution should be:

- In the simulation model it should be possible to choose different kinds of in feed to the digester.
- The simulations should count not only for all standard steps and features in ADM1, but also for other commonly used and interesting features.
- Different kind of simulations should be implemented on top of the model.

1.2.4 Research questions

The following research questions should be addressed:

- How can ADM1 be changed to take care of manure as well as standard input?
- How should the communication between the user and the model be implemented?
- Which parameters in the model are the most important parameters to measure correct to get simulations and reality to correspond?
- How are those parameters changing when moving in parameter and variable space?
- Which is the optimal mixture of in-feed to a digester in order produce as much biogas as possible?

1.2.5 Limitations

The result of the thesis should not count for problems linked to digesting horse manure to biogas not direct linked to the digestion step. For example it should not count for problems with small stones and sand that can be present in the manure. Those types of problems are still present, but they are to be taken care of by other persons in the collaboration. The thesis is only linked to questions directly coupled to the digestion when combining waste water sludge and manure.

The thesis work is also only limited to cover simulation and modeling of the digester steps in the process. There is a lot of practical work done in the major project which not is covered in this thesis. The main focus is in building the tools for doing simulations and optimizations of biogas plants.

1.2.6 Outline

The outline of the thesis is given as

- **Introduction and Background** introduces the problem and specifies the purpose.
- **Theory and Resources** explain basics about the model used and all theory needed to understand the thesis.
- **Method** covers methods used.

- **Results** displays results from modeling and simulations.
- **Discussion** covers results and explains them.
- **Conclusions** conclude what is done, and further work possible to do on the model.

2

Theory and Resources

2.1 Abbreviation and things that are good to know

This thesis contains several abbreviations and to make it readable those abbreviations can be found in table 2.1. Due to the fact that the thesis also contains a lot of chemistry, a set of explanations is also included to increase readability for physicists, those can be seen in table 2.2.

Table 2.1: Abbreviations

General	
Term	**Explanation**
ADM1	Anaerobe Digestion Model no. 1
COD	Chemical Oxygen Demand
DAE	Differential and Algebraic Equation
DE	Differential Equation
ODE	Ordinary Differential Equation
GUI	Graphical User Interface
The task group	IWA Task Group for Mathematical Modelling of Anaerobic Digestion Processes - The group who developed ADM1.
Substrates	
Term	**Explanation**
LCAF	Long Chain Fatty Acids
$LCAF^-$	LCAF base equivalent
aa	Amino acid
ms	Monosaccharides
su	Sugar
fa	Fatty acid
va	Valerate
bu	Butyrate
pro	Propionate
ac	Acetate
h_2	Hydrogen
ch_4	Methane
IC	Inorganic carbon
IN	Inorganic nitrogen
I	Inerts
xc	Composites
ch	Carbohydrates
pr	Proteins
li	Lipids
c_4	Butyrate and Valerate particulate
cat	Cations
an	Anions
va^-	Valerate acid
bu^-	Butyrate acid
pro^-	Propionate acid
ac^-	Acetate acid
hco_3^-	Bicarbonate
nh_3	Ammonia
lac	Lactate
lac,f	Particulate lactate (fermentation)
lac,o	Particulate lactate (oxidation)
ca	Calcium

Table 2.2: Explanations for physicists

Term	Explanation
Acetogenesis	A process through which acetate is produced with help of anaerobic bacterias
Acidogenesis	A process where monomers are converted to shorter volatile fatty acids
COD	Chemical oxygen demand tells the amount of oxygen needed to decompose a specific amount of organic material. A high COD value corresponds to a high concentration of organic compounds in the materials, leading to a high gas exchange.
Extra cellular	Outside the cells
Hydrolysis	A chemical process where a molecule is divided after water is added.
Intra cellular	Inside the cells
Methanogenesis	A biological reaction where acetates are converted into methane and carbon dioxide
Monod kinetics	Mathematical model for growth of microorganisms
Fermentation	A metabolic process that converts sugar to acids, gas and/or alcohol
Oxidation	Chemical reaction in which a substrate gives away one or more electrons
Reaction order	Determines rate of reactions
β-oxidation	Cyclic degradation process of fatty acids in five steps
Arrhenius equation	Determines the reaction rates as a function of the temperature
Monomer	Unit (molecule) that can be linked to other monomers to form chains
Polymers	Large unit consisting of multiple monomers
Inhibition	Preventing a chemical reaction
Liquid-liquid process	Process only involving liquids
Reactor head space	Gas volume of reactor above liquid volume in reactor

2.2 Units

The units in this thesis is chosen to fit the choice of parameters used in ADM1. So for explanation of the units see section about units in ADM1 2.4.2.

2.3 Nomenclature

The nomenclature of this thesis are chosen to fit the nomenclature of ADM1 for increased readability. So for explanation see section about nomenclature in AMD1 2.4.3.

2.4 Anaerobic Digestion Model No. 1 - ADM1

The mathematical anaerobic digestion model ADM1 is the model that the simulations of this thesis are based on. The model is developed by *IWA Task Group for Mathematical Modeling of Anaerobic Digestion Processes* [3] and is widely used for modeling of biogas plants. The model consists originally of 29 processes and 24 substances and the substrate level is of Monod-type kinetics.

The model is changed a bit when implemented in this thesis to fulfill requirements set up in the project. The difference is not in the main features of the model, but instead in the amount of substrates in the model. Along with substrates such as sugar, amino acids and propionate also lactate is added to the process resulting in a more complex, but correct, picture of substrates to take care of. Besides this calcium precipitation is also added to the model due to its affect on carbon dioxide partial pressure. This section first covers the standard ADM1 model and then the differences made in this specific implementation.

In the report written by the *IWA task group*, here referred to as the task group, the model is explained [3]. The report explains the different steps and how they are coupled to each other and what affect change in one step will cause in another step. ADM1 is explained as *"a structured model with disintegration and hydrolysis, acidogenesis, acetogenesis and methanogenesis steps"*.

ADM1 is created to be used with a lot of possible types of anaerobic processes and might therefore not be as exact as other models that are developed for a specific task. The advantage of this is however that the model can be applied to a wide field of applications.

2.4.1 Basis of the model

The model models cellular kinetics as substrate uptake, growth and decay. Of which the substrate uptake (for different substrates) can be described with the formula

$$v_i = k_m \frac{S_i}{S_i + K_s} X_i \times I_i \tag{2.1}$$

Where v_i is the actual uptake, k_m is the maximum uptake rate, $\frac{S_i}{S_i+K_s}$ is the substrate concentration factor, X_i is the biomass concentration and I_i is the inhibition factor,

explained below. The reactions taking place can be seen in the ADM1 report's matrix for biochemical reactions, which can be found in table A.1 and A.2 in appendix A. This is the normal way of presenting the biochemical equations in the model, but due to the fact that this thesis belongs to the physics field, all reactions are also presented as normal sets of differential equations in the method section, see section 3.

Disintegration is mainly included into the model to describe degradation of composite materials to well known inerts, particulate carbohydrates, proteins and lipids. The second step is enzymatic reactions that convert particulate carbohydrates, lipids and proteins to monosaccharides (MS), amino acids (AA) and long chain fatty acids respectively (LCAF).

The model contains both biochemical and physico-chemical reactions, of which the physico-chemical reactions are implemented as reversible, while the biochemical reactions are implemented as irreversible reactions. The biochemical processes are normally catalyzed by intra- and extracellular enzymes that acts on the available biomass, while the physico-chemical processes not are biological mediating. Physiochemical processes include ion association/dissociation and gas-liquid transfer.

The flow of the material in the model can be seen as the flow of COD in the model. The flow rates and the different steps can therefore be visualized as a COD-flow chart, which can be seen in figure 2.1. The picture comes from [3]. In order to fit the model to a specific type of mixture or process a lot of things can be done before the first step in the model. For example the mixture between sludge and manure can be changed. The inflow to this block will define how much manure and how much standard waste water that are in the inflow to the digestion chamber.

If the main interest not is in finding a quick way to see all flows in the model, but instead in all reactions that is taking place in the modeled process, the flowchart in figure 2.2 can be seen. The figure can be divided into two parts, where the top part marked with "A" takes care of the hydrolysis of substrate and the second part, denoted "B", handle the reactions catalyzed by different groups of Monod type kinetics. In the second part different numbers are presented describing which biochemical reaction that occurs in each point.

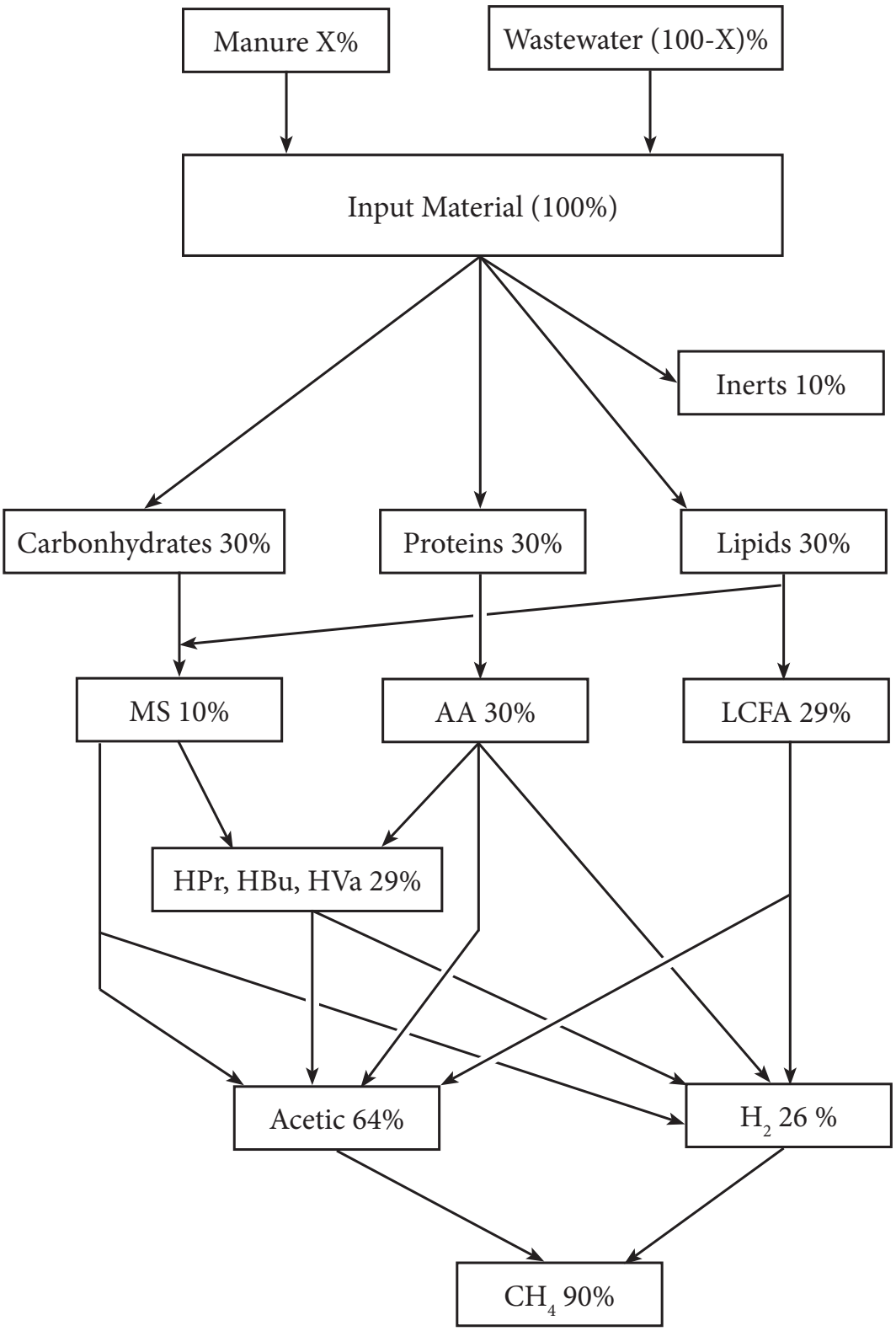

Figure 2.1: Flow chart for COD ind ADM1. The figure comes from the report about ADM1 [3]. This flow model is modified a bit to fit the implementation done in this work, see section 2.5 for further explanation.

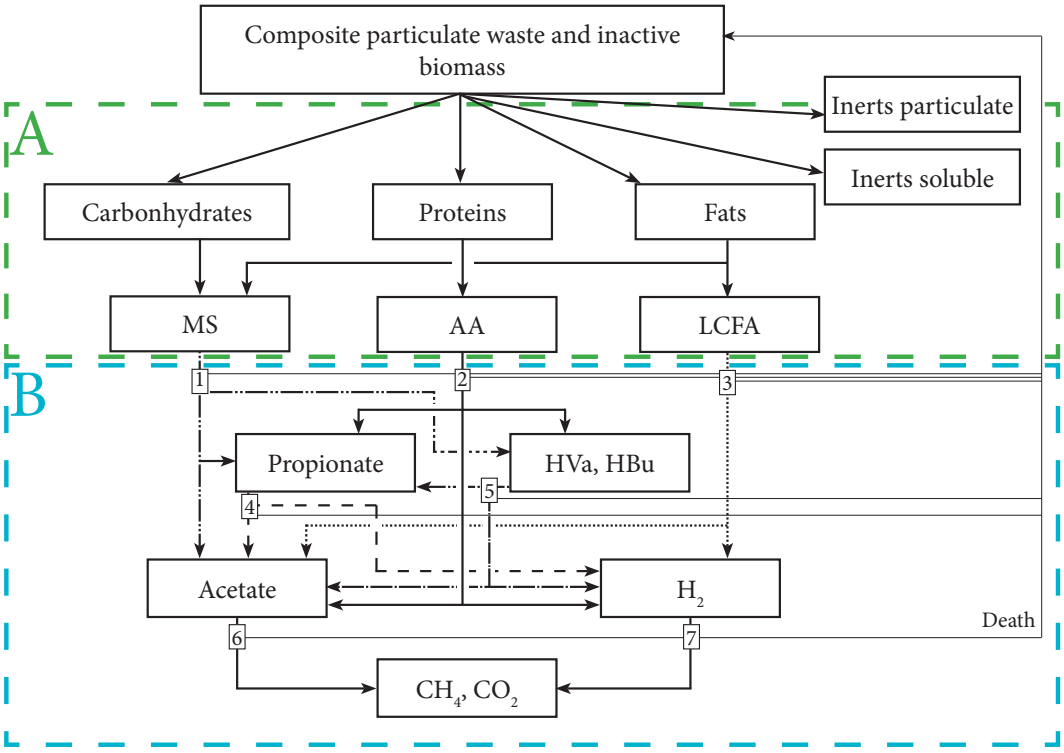

Figure 2.2: Reactions flow chart for ADM1. The numbers indicates which biochemical process that occurs at each point; (1) acidogenesis from sugar, (2) acidogenesis from amino acids, (3) acetogenesis from LCFA, (4) acetogenesis from propionate, (5) acetogenesis from butyrate and valerate, (6) aceticlastic methanogenesis and (7) hydrogenothropic methanogenesis. Dead material is circulated through the process as inerts. [3]

2.4.2 Units in ADM1

COD is used in ADM1 as the chemical base unit component and it has the unit kgCOD $m^{-3} \equiv$ gCOD l^{-1}. Molar basis is used in the model for describing things that has no COD, such as inorganic carbon and nitrogen. The unit for Molar is kmole $m^{-3} \equiv$ M $\equiv$ mole l^{-1} . The units used in the ADM1 can be seen in table 2.3.

2.4.3 Nomenclature, Parameters and variables in ADM1

The model is developed with four main types of parameters and variables; stoichiometric coefficients, equilibrium coefficients, kinetic parameters and finally dynamical state and algebraic variables. To increase usability of this project and its report the same nomenclature is used here as it is in the report about the model [3] and further the same notation as proposed by Rosén and Jeppsson [4], due to that their proposed implementation is used as far as possible. This nomenclature can be seen in tables 2.4 - 2.7. The same nomenclature that is used in the report is also used in the simulation

Table 2.3: Units to be used in the ADM1 model

Measurement	Units
Concentration	kgCOD m^{-3}
Concentration (non-COD)	kmoleC m^{-3}
Concentration (nitrogen non-COD)	kmoleN m^{-3}
Pressure	bar
Temperature	K
Distance	m
Volume	m^3
Energy	J (kJ)
Time	d (day)

Table 2.4: Stoichiometric coefficients [3]

Symbol	Description	Units
C_i	Carbon content of component i	kmoleC kgCOD $^{-1}$
N_i	Nitrogen content of component i	kmoleN kgCOD $^{-1}$
$\nu_{i,j}$	Rate coefficient for component i on process j	nominally kgCOD $^{-1}$
$f_{\text{product,substrate}}$	Yield (catabolism only) of product on substrate	kgCOD kgCOD $^{-1}$

implementation.

2.4.4 Dynamic state variables

The model can be implemented as a set of differential equations (DE) or as a set of Differential and Algebraic Equations (DAE), of which the DE method is used.

Table 2.5: Equilibrium coefficients and constants [3]

Symbol	Description	Units
H_{gas}	Gas law constant (equal to K_H^{-1})	bar M^{-1} (bar m^3 $kmole^{-1}$)
$K_{a,acid}$	Acid-base equilibrium coefficient	M (kmole m^{-3})
K_H	Henry's law coefficient	M bar^{-1} (kmole $m^{-3}bar^{-1}$)
pK_a	$\log_{10}[K_a]$	
R	Gas law constant (8.314×10^{-2})	bar $M^{-1}K^{-1}$ (bar m^3kmole $^{-1}K^{-1}$
ΔG	Free energy	J $mole^{-1}$

Table 2.6: Kinetic parameters and rates [3]

Symbol	Description	Units
$k_{A/Bi}$	Acid base kinetic parameter	$M^{-1}d^{-1}$
k_{dec}	First order decay rate	d^{-1}
$I_{inhibitor,process}$	Inhibition function (see K_l)	
$k_{process}$	First order parameter (normally for hydrolysis)	d^{-1}
k_{La}	Gas-liquid transfer coefficient	d^{-1}
$k_{l,\ inhibit,\ substrate}$	50% inhibitory concentration	kgCODm $^{-3}$
$k_{m,\ process}$	Monod maximum specific uptake rate (μ_{max}/Y)	kgCOD_S $kgCOD\_X^{-1}$ d^{-1}
$k_{S,\ process}$	Half saturation value	kgCOD_S m^{-3}
ρ_j	Kinetic rate of process j	kgCOD_S $m^{-3}d^{-1}$
$Y_{substrate}$	Yield of biomass on substrate	kgCOD_X $kgCOD_S^{-1}$
μ_{max}	Monod maximum specific growth rate	d^{-1}

Table 2.7: Dynamic state and algebraic variables [3]

Symbol	Description	Units
pH	$-\log_{10}[H^+]$	
$p_{gas,i}$	pressure of gas i	bar
P_{gas}	Total gas pressure	bar
S_i	Soluble component i	kgCOD m^{-3}
$t_{res,\ X}$	Extended retention of solids	d
T	Temperature	K
V	Volume	m^3
X_i	Particulate component i	kgCOD m^{-3}

2.4.5 Biochemical processes

The model includes three main biological cellular steps; acidogenesis, acetogenesis and methanogenesis, as well as an extracellular disintegration step.

The disintegration step is supposed to contain several steps so that the product from the step can be assumed to be homogeneous before it disintegrates to carbohydrates, proteins and lipids, reason for that is that multiple digestion ensures total digestion. Those can then in the next step be disintegrated in the next biochemical steps. [3]

In the report written by the IWA task group the key rate equation in the ADM1 is the substrate uptake, which is based on substrate level Monod-type kinetics.

To know how to implement the biochemical steps the rate equation matrices should be studied A.1 and A.2. This matrix includes booth the rates for the equations implemented in the model. The matrices can be seen in Appendix A. To get a more common way of seeing this for a physicist the equations in section 3.1.1 are presented, which presents the matrices' reactions as normal ODE system.

Disintegration and hydrolysis

Disintegration and hydrolysis are extra-cellular steps handling breakdown and solubilization of complex organic materials to monomers. Those substrates are composite particulates and particulate carbohydrates, proteins and lipids. Of which carbohydrates, proteins and lipids can be digested to monosaccharides, amino acids and long chain fatty acids with help of enzymatic reactions.

A mainly non-biological disintegration step is included as a first step in the model to make the model possible to use for a large variety of input materials. The enzymatic step is a complex combination of multiple steps taking care of carbohydrates, proteins and lipids. Due to that first order and second order kinetics could fit biogas production equally well, the task group did find out that it was enough to use a more easy to use

model with only first order kinetics. [3]

Acidogenesis
Acidogenesis, or fermentation is generally defined as an anaerobic acid-producing microbial process without an additional electron acceptor or donor. The IWA task group has decided to use glucose molecule as modeling monomer for the process. Furthermore, the task group decided to include acetate, propionate and butyrate in the model due to its importance as end-products from the digestion of monosaccharides acidogenesis.

Syntrophic hydrogen-producing acetogenesis and hydrogen-utilizing methanogenesis
Degradation of organic acids to acetate is an oxidation process without internal electron acceptors and therefore is the process depending on external acceptors such as hydrogen ions or carbon dioxide to produce hydrogen gas. The thermodynamics of syntrophic reaction described here is only possible in a narrow range of hydrogen concentration. According to the IWA task group this is important for modeling, due to its affect for hydrogen inhibition, half saturation coefficients and yields, and it is therefore included in ADM1. [3]

The electron carrier described here can not only be hydrogen, but also formate. However due to the task group has decided to only count on hydrogen as electron carrier, formate is not taken care of in this text.

The anaerobic degradation of fatty acids above propionate is done by the so called β-oxidation which is a cyclic process. The substrates produced from this degradation taken care of in the ADM1 are butyrate, valerate and LCFA.

To model hydrogen inhibition for acetogenesis is the more simple non-competitive inhibition used in the model and liquid phase hydrogen concentration is used for hydrogen inhibition.

Aceticlastic methanogenesis
The most important step in forming methane from acetate contains cleaving of acetate to methane and carbon dioxide.

$$CH_3COOH \longrightarrow CH_4 + CO_2 \tag{2.2}$$

Inhibition
Inhibition is included in ADM1 in form of *Biostatic inhibition*, which is a "Nonreactive toxicity, normally reversible" form of inhibition affecting kinetic uptake and growth [3].

The affect of included inhibition in the models rate equations is calculated on the form

$$\rho_j = \frac{k_m S}{K_s^+ S} X \cdot I_1 \cdot I_2 \cdots I_n \tag{2.3}$$

Table 2.8: Forms of inhibition used in ADM1. Forms not used but mentioned in the task groups report are skipped to mention here. [3]

Description	Equation	Used for
Non-competitive inhibition	$I = \frac{1}{1+S_1/K_1}$	Free ammonia and hydrogen inhibition
Empirical upper and lower inhibition	$I = \frac{1+2\times10^{0.5(pH_{LL}-pH_{UL})}}{1+10^{(pH-pH_{UL})}+10^{(pH_{LL}-pH)}}$	pH inhibition when both high and low pH inhibition occurs
Empirical lower inhibition only	$\begin{cases} I = \exp\left(-3\left(\frac{pH-pH_{LL}}{pH_{UL}-pH_{LL}}\right)^2\right)\Big\vert_{pH<pH_{UL}} \\ I = 1\big\vert_{pH>pH_{UL}} \end{cases}$	pH inhibition when only low pH inhibition occurs
Competitive uptake	$I = \frac{1}{1+S_1/S}$	Butyrate and valerate competition for C_4
Secondary substrate	$I = \frac{1}{1+K_1/S_1}$	All uptake, to inhibit uptake when $C_{IN} \sim 0$

Where the first part of the function is the uninhibited Monod-type uptake and $I_{1...n}$=f($S_{I,1...n}$) are the inhibited functions presented in the biochemical rate matrices A.1 and A.2. For those cases where this equation not can be used are instead the inhibition functions presented in table 2.8.

pH inhibition are used for all intracellular interactions in ADM1.

Temperature dependence

For anaerobic digesting processes there are three major temperature ranges one usually talks about. Those are *psychrophilic* (4-15°C), *mesophilic* (20-40°C) and *thermophilic* (45-70°C).

According to the task group there are three major system types that need to be modeled with respect to temperature.

1. *Temperature controlled with small changes in operating temperature (±3°C).*
2. *Uncontrolled but fluctuating within one range (either mesophilic or thermophilic).*
3. *Fluctuation between mesophilic and thermophilic temperatures.*

The temperature dependence follows Arrhenius equation to an optimal temperature and then drops rapidly to zero. Although this implements that there are a continuous function describing temperature dependence are separate values used for thermophilic and mesophilic conditions in ADM1. [3]

2.4.6 Physico-chemical processes

Physico-chemical processes are processes that not are biological mediating processes. Those include liquid-liquid processes, liquid-gas processes and liquid-solid processes. In ADM1 the last type of mentioned processes are excluded, but in this implementation calcium carbonate precipitation are added to model affects of carbon dioxide production. When modeling a system physico-chemical processes are really important due to their key role, which for example includes parameters such as pH and gas flow.

Liquid-liquid processes
The liquid-liquid processes that are modeled mainly take care about ion association and dissociation with hydrogen and hydroxide ions. Due to the rapidness of association/dissociation processes those processes are often referred to as equilibrium processes.

Liquid-gas processes
When creating the model, the task group considered the following three gas components important H_2, CH_4 and CO_2 due to their strong effect on biological processes or outputs of them and included them into the model.

The gas and liquid phases will, when they are in contact to each other reach an equilibrium state [3]. When the gas phase are enough diluted Henry's law can be used to describe the concentrations

$$K_H p_{gas,i,ss} - S_{liq,i,ss} = 0 \tag{2.4}$$

Where $S_{liq,i,ss}$ is the steady state liquid phase concentration for component i, $p_{gas,i,ss}$ is the steady state gas phase partial pressure of component i and K_H is Henry's law coefficient.

Temperature dependence
Physico-chemical systems are highly affected by change in temperature, due to changes in equilibrium coefficients.

2.4.7 Implementation and regulation of ADM1

The model can be implemented in two main different ways; firstly a set of differential equations (DE) can be used and secondly a set of algebraic differential equations (DAE) can be used, the implementation approach used is the DE method, implying that this report only covers this method.

Liquid phase equations
The reaction system is implemented as a system of mass balance reactions, those mass balance reactions can be written as

$$\frac{\mathrm{dVS_{liq,i}}}{\mathrm{d}t} = \mathrm{q_{in}S_{in,i}} - \mathrm{q_{out}S_{liq,i}} + \mathrm{V}\sum_{j=1}^{19}\rho_{\mathrm{j}}\nu_{\mathrm{i,j}} \tag{2.5}$$

Where $\sum_{j=1}^{19}\rho_{\mathrm{j}}\nu_{\mathrm{i,j}}$ is the sum of specific kinetic rates for process j multiplied by $\nu_{\mathrm{i,j}}$, which is the rate coefficient pof component i on process j. This includes shift in volume, but if a constant volume and flow is assumed, ($\mathrm{q}=\mathrm{q_{in}}=\mathrm{q_{out}}$) can be used implementing that the reaction boils down to.

$$\frac{\mathrm{dS_{liq,i}}}{\mathrm{d}t} = \frac{\mathrm{qS_{in,i}}}{\mathrm{V_{liq}}} - \frac{\mathrm{qS_{liq,i}}}{\mathrm{V_{liq}}} + \sum_{j=1}^{19}\rho_{\mathrm{j}}\nu_{\mathrm{i,j}} \tag{2.6}$$

In addition to the rate equations specified the biochemical rate matrices (A.1) and (A.2) do also the following rate equations need to be added to the simulation

$$\begin{aligned}
\rho_{\mathrm{T,H_2}} &= \mathrm{k_L a}\,(\mathrm{S_{liq,H_2}} - 16\,\mathrm{K_{H,H_2}p_{gas,H_2}}) \\
\rho_{\mathrm{T,CH_4}} &= \mathrm{k_L a}\,(\mathrm{S_{liq,CH_4}} - 64\,\mathrm{K_{H,CH_4}p_{gas,CH_4}}) \\
\rho_{\mathrm{T,IC}} &= \mathrm{k_L a}\,(\mathrm{S_{liq,CO_2}} - \mathrm{K_{H,CO_2}p_{gas,CO_2}})
\end{aligned} \tag{2.7}$$

Where $\rho_{\mathrm{T,i}}$ is the transfer rate of gas i and $\mathrm{S_{liq,CO_2}}$ is the fraction of inorganic carbon as $\mathrm{CO_2}$.

Besides this also ion equations needs to be added to the simulation, reaction rates for those looks like:

$$\rho_{A,i} = k_{A,B,i} \cdot (S_{i-} \cdot (K_{a,i} + S_{h^+}) - K_{a,i} \cdot S_i) \tag{2.8}$$

Where i stands for ion number i, $\mathrm{K_{a\,i}}$ is the acid-base equilibrium coefficient for acid i, S_{h^+} is the concentration of H^+ ions and $k_{A,B,i}$ is the acid base kinetic parameter parameter for acid i. From those rates DE equations looks like

$$\frac{\mathrm{d}S_{i^-}}{\mathrm{d}t} = -\rho_{A,i} \tag{2.9}$$

Gas phase equations

The gas phase rate equations are very similar to the liquid phase equations, except there is no advective influent flow [3]. For a system with constant gas volume the equations looks like

$$\frac{\mathrm{dS_{gas,i}}}{\mathrm{d}t} = -\frac{\mathrm{S_{gas,i}q_{gas}}}{\mathrm{V_{liq}}} + \rho_{\mathrm{T,i}}\frac{\mathrm{V_{liq}}}{\mathrm{V_{gas}}} \tag{2.10}$$

The factor $\frac{\mathrm{V_{liq}}}{\mathrm{V_{gas}}}$ is required due to that gas transfer kinetics is liquid-volume specific.

The pressure of each gas can be calculated with help of the gas law and the factors (*16, 64, 1*) introduced in equation (2.7) according to [3]. The factors comes corresponds to COD equivalents of the gases.

$$\begin{aligned} p_{gas,H_2} &= S_{gas,H_2}RT/16 \\ p_{gas,CH_4} &= S_{gas,CH_4}RT/64 \\ p_{gas,CO_2} &= S_{gas,CO_2}RT \end{aligned} \tag{2.11}$$

According to the task group it is accurate to assume that the reactor headspace is saturated with water vapor. By using the formula for gas pressure and inserting values for water vapor in it one will obtain values for the water vapor pressure. From this can the total gas flow be calculated, by setting it equal to the total gas transfer, corrected for the water vapor

$$q_{gas} = \frac{RT}{p_{gas} - p_{gas,H_2O}} V_{liq} \left(\frac{\rho_{T,H_2}}{16} + \frac{\rho_{T,CH_4}}{64} + \rho_{T,CO_2} \right) \tag{2.12}$$

Regulation
When implementing the model, a lot of different equations and variables are to be used. The coupling between those variables is large and by changing in one part of the system one other part of the system might be affected. For example by accelerating the production in one part of the system might inhibit an other part of the system then leading to for example decreased digesting or gas production in a third part of the system.

2.5 Changes made to standard ADM1

The model implemented in this thesis is not exactly the same as the standard ADM1 described by [3]. Instead new equations proposed by Willquist *et al.* [5] are added. Those equations follows the same logic as the other reactions, but ad extra possibilities to simulate the process in the biogas plant.

To the substrates are lactates added in terms of fermentation and oxidation. Reason for adding them both is that it could be interesting in some applications to see what happens with different types of lactate biomass. The equations derived from lactate can be seen in the same biochemical rate matrix as all other biochemical processes A.1 and A.2.

Besides lactate, also precipitation of calcium is added to the model. Due to that precipitation are affecting the production of carbon dioxide, which highly affects the overall production of biogas, it is interesting to have the possibility to correct for those affects during simulation of an entire system. This precipitation is a *liquid-solid* process, which was neglected by the task group.

2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a biologically inspired optimization method that mimics the advantage of forming swarms. In nature the tendency of forming swarms appears in many different spices. For example the tendency can be found in populations of fishes and birds. The advantages of forming swarms are for example the protection from predators. An individual close to the center of the swarm is unlikely to be attacked by a predator and the swarm may confuse the predator by coordinated movements, but more important is that the risk for each individual to be hit by a predator decreases rapidly in a large swarm. However if the predator has located the swarm he do know where the pray is, so the likelihood for that any of the prays will be hit is quite large. [6]

A swarm consisting of a lot of individuals has a large advantage in searching for things, for example food, due to the large amount of individuals contributing to the search. By coordinating the swarm the search can be even more effective and this is what particle swarm optimization method is about.

PSO-algorithm tries to capture the beneficial properties of a swarm. Aspects mimicked are those who in an optimization point of view are important, namely the search efficiency of a swarm. In PSO a population of possible solutions to an optimization problem is founded and evaluated. The initialization of those solutions are randomly chosen inside an allowed search space. The best known position in search space for all solutions is saved as the best know global solution and the best known (at initialization the only known) solution for each particle is saved as the local best known solution so far.

By steering all test solutions in direction of the global best known position more positions in search space are evaluated in a more efficient manner, compared to random search. There are however some drawbacks with this method, no guaranties that the best solution is ever found can be given. The advantages of using PSO are however in the amount of parameters that can be evaluated and the amount of information needed about the optimization system needed. The complexity of the ADM1 system together with the large amount of parameters makes the use of PSO a good choice.

2.7 Horse manure as a resource for biogas production

This project aims to find a good way of mixing horse manure with waste water as a resource for a biogas plant. Horse manure is not only hard to get rid of, it is also a good resource to more carbon in the biogas processes. When adding horse manure to the raw material in biogas processes do one not only get manure, one do also get a lot of bedding material included in the manure waste, leading to low biogas yield per volume unit of raw material. However horse manure has, as stated, a lot of carbon that can be used in the process to balance the carbon/nitrogen ratio. [7]

Dry weight of horse manure/bedding mixture is approximately 20-25% of which about 75% are organic materials interesting for biogas production. Mixtures will vary depending on bedding materials, feed, size of horse and activity of the horse. [7]

2.8 Waste water sludge as resource for biogas production

Sludge for biogas production is characterized by the fact that it has been digested in earlier steps of some purification process and therefore it contains a lower amount of material that can be digested to manufacturer biogas. The dry weight for sludge varies between 2-100%. [7]

2.9 Co-digestion with different substrates

Normally when one talks about co-digestion of different substrates, one mean that a main substrate is used and to this a secondary substrate is added in smaller amounts.

The co-digestion branch of digestion to biogas is interesting due to the possibilities in trimming the process in the direction wanted. Processes can be both more stable and more effective when a co-digestion method is applied, as the full effect of the digestion chamber is used. For example, a bad carbon/nitrogen ratio can be improved with different mixtures of substrates. Furthermore manure could be digested in existing plants without needs for enormous investments in new plants.

3

Method

The main part of the work done in this thesis is in implementing ADM1, testing and verifying the model and finally compile results from the simulations to something that can be used for co digesting manure with waste water sludge.

3.1 Model implementation

When implementing the model one report by Rosén and Jeppsson [4] have been used, that deals with ADM1 implementation through an ODE system as well as a DAE system, of which the ODE implementation is used in this work. In the report it is discussed that the initial carbon and nitrogen balance when digesting composites to carbohydrates, proteins and lipids not holds. In the report do Rosén and Jeppsson discuss why the balance not holds and what needed to be changed for it to hold. The affect of the original incomplete balances is that in the first step of the model it disappears about 5-6% of carbon and nitrogen, but by changing the fraction of obtained proteins, lipids and carbohydrates from the initial composites are the carbon and nitrogen balances closed.

In this thesis implementation of ADM1 is therefore the changes proposed by Rosén and Jeppsson done in order to close the carbon and nitrogen balances.

The variable names in this thesis are also consistent with the one used by Rosén and Jeppsson in order to increase readability of the implementation. The structure does also follow the one in the report due to readability and to make it easier for further implementations of the model.

However does the implementation not completely follow the implementation proposed by Rosén and Jeppsson due to that new equations are added to the model in order to take care of other, not standard ADM1, substrates. Those equations are proposed by Willquist *et al.* [5]. The new equations take care of lactate and calcium. Reason for adding those equations is that one of the main part of this project, SP Energiteknik in Lund, is interested in those two substrates and how they affects the outcome of the

simulations. However are those equations not included in the theoretical testing of this work due to that those equations not are standard equations and therefore it is not possible to verify the effect of those equations. This does not mean that the equations not are tested, they are not only included inside the testing and verifying presented here.

Lactate uptake is implemented in form of oxidation and fermentation processes. Decay of lactate biomass is also implemented separately for fermentation and oxidation. Calcium is implemented as precipitation in the model.

The model is built in a way so that if one not have some of the substrates presented in the model, this can be set to zero and Willquist therefore not affect the simulation outcome. For example, if one is interested in standard ADM1 without lactate and calcium those two substrates can be set to zero and will then not affect outcome of the model, hence the standard ADM1 implementation without lactate and calcium can be obtained, by setting lactate and calcium amounts to zero.

3.1.1 Model equations

In order to describe the model implemented it is important to know all the equations implemented in the system. The most of the equations can in a quick way be seen in the biochemical rate matrices A.1-A.2. However to get a complete overview of the system implemented are here all equations described in a standard way for a physicist.

The equations are built up by all the reactions occurring in the system. However there are a lot of parameters as well as variables. All of those parameters define the position in parameter space at which the reactions take place in. Some of those parameters can be modified to fit an implemented process, meanwhile some of them are fixed due to laws of nature. All of those parameters can be found in appendix B.

Process rates

Biochemical process rates for standard ADM1 (line 1-19) and new process rates (line 20-23).

$$
\begin{aligned}
\rho_1 &= k_{dis} \cdot X_c \\
\rho_2 &= k_{hyd,ch} \cdot X_{ch} \\
\rho_3 &= k_{hyd,pr} \cdot X_{pr} \\
\rho_4 &= k_{hyd,li} \cdot X_{li} \\
\rho_5 &= k_{m,su} \cdot \frac{S_{su}}{K_{S,su} + S_{su}} \cdot X_{su} \cdot I_5 \\
\rho_6 &= k_{m,aa} \cdot \frac{S_{aa}}{K_{S,aa} + S_{aa}} \cdot X_{aa} \cdot I_6 \\
\rho_7 &= k_{m,fa} \cdot \frac{S_{fa}}{K_{S,fa} + S_{fa}} \cdot X_{fa} \cdot I_7 \\
\rho_8 &= k_{m,c4} \cdot \frac{S_{va}}{K_{S,c4} + S_{va}} \cdot X_{c4} \cdot \frac{s_{va}}{S_{bu} + S_{va} + 1e^{-6}} \cdot I_8 \\
\rho_9 &= k_{m,c4} \cdot \frac{S_{bu}}{K_{S,c4} + S_{bu}} \cdot X_{c4} \cdot \frac{s_{bu}}{S_{va} + S_{bu} + 1e^{-6}} \cdot I_9 \\
\rho_{10} &= k_{m,pro} \cdot \frac{S_{pro}}{K_{S,pro} + S_{pro}} \cdot X_{pro} \cdot I_{10} \\
\rho_{11} &= k_{m,ac} \cdot \frac{S_{ac}}{K_{S,ac} + S_{ac}} \cdot X_{ac} \cdot I_{11} \\
\rho_{12} &= k_{m,h2} \cdot \frac{S_{h2}}{K_{S,h2} + S_{h2}} \cdot X_{h2} \cdot I_{12} \\
\rho_{13} &= k_{dec,Xsu} \cdot X_{su} \\
\rho_{14} &= k_{dec,Xaa} \cdot X_{aa} \\
\rho_{15} &= k_{dec,Xfa} \cdot X_{fa} \\
\rho_{16} &= k_{dec,Xc4} \cdot X_{c4} \\
\rho_{17} &= k_{dec,Xpro} \cdot X_{pro} \\
\rho_{18} &= k_{dec,Xac} \cdot X_{ac} \\
\rho_{19} &= k_{dec,Xh2} \cdot X_{h2} \\
\rho_{20} &= k_{m,lacf} \cdot \frac{S_{lac}}{K_{S,lacf} + S_{lac}} \cdot X_{lacf} \cdot I_{20} \\
\rho_{21} &= k_{m,laco} \cdot \frac{S_{lac}}{K_{S,laco} + S_{lac}} \cdot X_{laco} \cdot I_{21} \\
\rho_{22} &= k_{dec,Xlacf} \cdot X_{lacf} \\
\rho_{23} &= k_{dec,Xlaco} \cdot X_{laco}
\end{aligned}
\tag{3.1}
$$

To compensate for bad choices of S_{bu} and S_{va} a small constant had been added to expressions ρ_8 and ρ_9, according to [4].

Acid-base rates

$$\begin{aligned}
\rho_{A,4} &= k_{A,Bva}(S_{va^-}(K_{a,va} + S_{H^+}) - K_{a,va}S_{va}) \\
\rho_{A,5} &= k_{A,Bbu}(S_{bu^-}(K_{a,bu} + S_{H^+}) - K_{a,bu}S_{bu}) \\
\rho_{A,6} &= k_{A,Bpro}(S_{pro^-}(K_{a,pro} + S_{H^+}) - K_{a,pro}S_{pro}) \\
\rho_{A,7} &= k_{A,Bac}(S_{ac^-}(K_{a,ac} + S_{H^+}) - K_{a,ac}S_{ac}) \\
\rho_{A,10} &= k_{A,Bco2}(S_{hco3^-}(K_{a,co2} + S_{H^+}) - K_{a,co2}S_{IC}) \\
\rho_{A,11} &= k_{A,BIN}(S_{nh3}(K_{a,IN} + S_{H^+}) - K_{a,IN}S_{IN})
\end{aligned} \tag{3.2}$$

Gas transfer rates

$$\begin{aligned}
\rho_{T,8} &= k_{La}(S_{h2} - 16 \cdot K_{H_H,h2}p_{gas,h2}) \\
\rho_{T,9} &= k_{La}(S_{ch4} - 64 \cdot K_{H_H,ch4}p_{gas,ch4}) \\
\rho_{T,10} &= k_{La}(S_{co2} - K_{H_H,co2}p_{gas,co2})
\end{aligned} \tag{3.3}$$

Precipitation rates
This equation are new compared to standard ADM1.

$$\rho_{P,24} = \begin{cases} 0 & : S_{ca}, S_{hco3^-} < 0 \\ K_{r,caco3} \cdot \left(\sqrt{S_{ca} \cdot S_{hco3^-}} - K_{S,p,caco3}\right)^2 & : S_{ca}, S_{hco3^-} \geq 0 \end{cases} \tag{3.4}$$

Process inhibition

$$pH = -\log_{10}(S_{H^+}) \tag{3.5}$$

$$\begin{aligned}
I_{pH,aa} &= \begin{cases} exp\left(-3\left(\frac{pH - pH_{UL,aa}}{pH_{UL,aa} - pH_{LL,aa}}\right)^2\right) & : pH < pH_{UL,aa} \\ 1 & : pH > pH_{UL,aa} \end{cases} \\
I_{pH,ac} &= \begin{cases} exp\left(-3\left(\frac{pH - pH_{UL,ac}}{pH_{UL,ac} - pH_{LL,ac}}\right)^2\right) & : pH < pH_{UL,ac} \\ 1 & : pH > pH_{UL,ac} \end{cases} \\
I_{pH,h2} &= \begin{cases} exp\left(-3\left(\frac{pH - pH_{UL,h2}}{pH_{UL,h2} - pH_{LL,h2}}\right)^2\right) & : pH < pH_{UL,h2} \\ 1 & : pH > pH_{UL,h2} \end{cases}
\end{aligned} \tag{3.6}$$

In the report by Rosén and Jeppsson [4] it is discussed about $I_{pH,i}$ functions above, that are original from the standard ADM1 model [3]. Rosén and Jeppsson proposes some other functions that can be used if there are problems with stability of those switch functions. In our implementation were this not a problem and therefore were the standard model for pH inhibition used.

$$
\begin{aligned}
I_{IN,lim} &= \frac{1}{1 + K_{S,IN}/S_{IN}} \\
I_{h2,fa} &= \frac{1}{1 + S_{h2}/K_{I,h2,fa}} \\
I_{h2,c4} &= \frac{1}{1 + S_{h2}/K_{I,h2,c4}} \\
I_{h2,pro} &= \frac{1}{1 + S_{h2}/K_{I,h2,pro}} \\
I_{nh3} &= \frac{1}{1 + S_{nh3}/K_{I,nh3}} \\
I_{5,6} &= I_{pH,aa} \cdot I_{IN,lim} \\
I_7 &= I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,fa} \\
I_{8,9} &= I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,c4} \\
I_{10} &= I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,pro} \\
I_{11} &= I_{pH,aa} \cdot I_{IN,lim} \cdot I_{nh3} \\
I_{12} &= I_{pH,h2} \cdot I_{IN,lim} \\
I_{h2,laco} &= \frac{1}{1 + S_{h2}/K_{I,h2,laco}} \\
I_{20} &= I_{IN,lim} \\
I_{21} &= I_{h2,laco} \cdot I_{IN,lim}
\end{aligned}
\tag{3.7}
$$

Soluble phase equation

The nomenclature here follows Rosén and Jeppssons nomenclature [4]. The idea of dividing the equations into sections comes from their report. The reason for keeping the same dividing here is to make it more readable and comparable.

Differential equations 1-4 for soluble matter

$$
\begin{aligned}
& & & & \text{State} \\
\frac{\mathrm{d}S_{su}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(S_{su,in} - S_{su}\right) + \rho_2 + \left(1 - f_{fa,li}\right)\rho_4 - \rho_5 & & [1] \\
\frac{\mathrm{d}S_{aa}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(S_{aa,in} - S_{aa}\right) + \rho_3 - \rho_6 & & [2] \\
\frac{\mathrm{d}S_{fa}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(S_{fa,in} - S_{fa}\right) + f_{fa,li}\rho_4 - \rho_7 & & [3] \\
\frac{\mathrm{d}S_{va}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(S_{va,in} - S_{va}\right) + \left(a - Y_{aa}\right) f_{va,aa}\rho_6 - \rho_8 & & [4]
\end{aligned}
$$

Differential equations 5-8 for soluble matter. Equation 6-8 are modified to count for lactate.

State

$$\frac{dS_{bu}}{dt} = \frac{q_{in}}{V_{liq}}(S_{bu,in} - S_{bu}) + (1 - Y_{su})\, f_{bu,su}\rho_5 + (1 - Y_{aa})\, f_{bu,aa}\rho_6 - \rho_9 \quad [5]$$

$$\frac{dS_{pro}}{dt} = \frac{q_{in}}{V_{liq}}(S_{pro,in} - S_{pro}) + (1 - Y_{su})\, f_{pro,su}\rho_5 + (1 - Y_{aa})\, f_{pro,aa}\rho_6 + (1 - Y_{c4})0.54\rho_8 - \rho_{10}$$
$$(1 - Y_{lacf})0.785\rho_{20} \quad [6]$$

$$\frac{dS_{ac}}{dt} = \frac{q_{in}}{V_{liq}}(S_{ac,in} - S_{ac}) + (1 - Y_{su})\, f_{ac,su}\rho_5 + (1 - Y_{aa})\, f_{ac,aa}\rho_6 + (1 - Y_{fa})0.7\rho_7 +$$
$$(1 - Y_{c4})0.31\rho_8 + (1 - Y_{c4})0.8\rho_9 + (1 - Y_{pro})0.57\rho_{10} - \rho_{11}$$
$$(1 - Y_{lacf})0.215\rho_{20} + (1 - Y_{laco})\tfrac{2}{3}\rho_{21} \quad [7]$$

$$\frac{dS_{h2}}{dt} = \frac{q_{in}}{V_{liq}}(S_{h2,in} - S_{h2}) + (1 - Y_{su}) f_{h2.su}\rho_5 + (1 - Y_{aa}) f_{h2,aa}\rho_6 + (1 - Y_{fa})0.3\rho_7 +$$
$$(1 - Y_{c4})0.15\rho_8 + (1 - Y_{c4})0.2\rho_9 + (1 - Y_{pro})0.43\rho_{10} - \rho_{12} - \rho_{T,8} + (1 - Y_{laco})\tfrac{1}{3}\rho_{21} \quad [8]$$

Differential equations 9-12 for soluble matter. Equations 10 and 11 are modified to count for lactate.

State

$$\frac{dS_{ch4}}{dt} = \frac{q_{in}}{V_{liq}}(S_{ch4,in} - S_{ch4}) + (1 - Y_{ac})\rho_{11} + (1 - Y_{h2})\rho_{12} - \rho_{T,9} \quad [9]$$

$$\frac{dS_{IC}{}^{*}}{dt} = \frac{q_{in}}{V_{liq}}(S_{IC,in} - S_{IC}) - \sum_{j=1}^{23}\left(\sum_{i=1-9,11-24,36-38} C_i\nu_{i,j}\rho_j\right) \quad [10]$$

$$\frac{dS_{IN}}{dt} = \frac{q_{in}}{V_{liq}}(S_{IN,in} - S_{IN}) - Y_{su}N_{bac}\rho_5 + (N_{aa} - Y_{aa}N_{bac})\rho_6 Y_{fa}N_{bac}\rho_7 - Y_{fa}N_{bac}\rho_8 -$$
$$Y_{c4}N_{bac}\rho_9 - Y_{pro}N_{bac}\rho_{10} - Y_{ac}N_{bac}\rho_{11} - Y_{h2}N_{bac}\rho_{12} - Y_{lacf}N_{bac}\rho_{20} - Y_{laco}N_{bac}\rho_{21}$$
$$+(N_{bac} - N_{xc})\sum_{\substack{i=13-19,\\22-23}}\rho_i + (N_{xc} - f_{xI,xc}N_I - f_{sI,xc}N_I - f_{pr,xc}N_{aa})\rho_1 \quad [11]$$

$$\frac{dS_I}{dt} = \frac{q_{in}}{V_{liq}}(S_{I,in} - S_I) + f_{sI,xc}\rho_1 \quad [12]$$

* See equation (3.8) for explanation.

The sum in the equation for the tenth state is calculated as

$$\sum_{j=1}^{19}\left(\sum_{\substack{i=1-9,\\11-24,\\36-38}} C_i\nu_{i,j}\rho_j\right) = \sum_{\substack{k=1-12,\\20-21}} s_k\rho_k + s_{13}(\rho_{13} + \rho_{14} + \rho_{15} + \rho_{16} + \rho_{17} + \rho_{18} + \rho_{19} + \rho_{22} + \rho_{23}) \quad (3.8)$$

where

$$
\begin{aligned}
s_1 &= -C_{xc} + f_{sI,xc}C_{sI} + f_{ch,xc}C_{ch} + f_{pr,xc}C_{pr} + f_{li,xc}C_{li} + f_{xI,xc}C_{xI} \\
s_2 &= -C_{ch} + C_{su} \\
s_3 &= -C_{pr} + C_{aa} \\
s_4 &= -C_{li} + (1 - f_{fa,li})C_{su} + f_{fa,li}C_{fa} \\
s_5 &= -C_{su} + (1 - Y_{su})(f_{bu,su}C_{bu} + f_{pro,su}C_{pro} + f_{ac,su}C_{ac}) + Y_{su}C_{bac} \\
s_6 &= -C_{aa} + (1 - Y_{aa})(f_{va,aa}C_{va} + f_{bu,aa}C_{bu} + f_{pro,aa}C_{pro} + f_{ac,aa}C_{ac}) + Y_{aa}C_{bac} \\
s_7 &= -C_{fa} + (1 - Y_{fa})0.7C_{ac} + Y_{c4}C_{bac} \\
s_8 &= -C_{va} + (1 - Y_{c4})0.54C_{pro} + (1 - Y_{c4})0.31C_{ac} + Y_{c4}C_{bac} \\
s_9 &= -C_{bu} + (1 - Y_{c4})0.8C_{ac} + Y_{c4}C_{bac} \\
s_{10} &= -C_{pro} + (1 - Y_{pro})0.57C_{ac} + Y_{pro}C_{bac} \\
s_{11} &= -C_{ac} + (1 - Y_{ac})C_{ch4} + Y_{ac}C_{bac} \\
s_{12} &= -C_{h2} + (1 - Y_{h2})C_{ch4} + Y_{h2}C_{bac} \\
s_{13} &= -C_{bac} + C_{xc} \\
s_{20} &= -C_{lac} + (1 - Y_{lacf})0.785C_{pro} + (1 - Y_{lacf})0.215C_{ac} + Y_{lacf}C_{bac} \\
s_{21} &= -C_{lac} + (1 - Y_{laco})\frac{2}{3}C_{ac} + Y_{laco}C_{bac}
\end{aligned}
\tag{3.9}
$$

Differential equations 13-16 for particulate matter. Equation 13 is modified to count for lactate.

$$
\begin{aligned}
& & & & \text{State} \\
\frac{\mathrm{d}X_c}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{c,in} - X_c) - \rho_1 + \sum_{\substack{i=13-19,\\22-23}} \rho_i & [13] \\
\frac{\mathrm{d}X_{ch}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{ch,in} - X_{ch}) + f_{ch,xc}\rho_1 - \rho_2 & [14] \\
\frac{\mathrm{d}X_{pr}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{pr,in} - X_{pr}) + f_{pr,xc}\rho_1 - \rho_3 & [15] \\
\frac{\mathrm{d}X_{li}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{li,in} - X_{li}) + f_{li,xc}\rho_1 - \rho_4 & [16]
\end{aligned}
$$

Differential equations 17-20 for particulate matter

$$
\begin{aligned}
& & & & \text{State} \\
\frac{\mathrm{d}X_{su}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{su,in} - X_{su}) + Y_{su}\rho_5 - \rho_{13} & [17] \\
\frac{\mathrm{d}X_{aa}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{aa,in} - X_{aa}) + Y_{aa}\rho_6 - \rho_{14} & [18] \\
\frac{\mathrm{d}X_{fa}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{fa,in} - X_{fa}) + Y_{fa}\rho_7 - \rho_{15} & [19] \\
\frac{\mathrm{d}X_{c4}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{c4,in} - X_{c4}) + Y_{c4}\rho_8 + Y_{c4}\rho_9 - \rho_{16} & [20]
\end{aligned}
$$

Differential equations 21-24 for particulate matter

$$\begin{aligned} & & & & \text{State} \\ \frac{\mathrm{d}X_{pro}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{pro,in} - X_{pro}\right) + Y_{pro}\rho_{10} - \rho_{17} & [21] \\ \frac{\mathrm{d}X_{ac}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{ac,in} - X_{ac}\right) + Y_{ac}\rho_{11} - \rho_{18} & [22] \\ \frac{\mathrm{d}X_{h2}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{h2,in} - X_{h2}\right) + Y_{h2}\rho_{12} - \rho_{19} & [23] \\ \frac{\mathrm{d}X_{I}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{I,in} - X_{I}\right) + f_{xI,xc}\rho_{1} & [24] \end{aligned}$$

Differential equations 25-26 for cations and anions

$$\begin{aligned} & & & \text{State} \\ \frac{\mathrm{d}S_{cat+}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{cat^+,in} - X_{cat^+}\right) & [25] \\ \frac{\mathrm{d}S_{an-}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}\left(X_{an^-,in} - X_{an^-}\right) & [26] \end{aligned}$$

Differential equations 27-32 for ion states

$$\begin{aligned} & & & \text{State} \\ \frac{\mathrm{d}S_{va-}}{\mathrm{d}t} &= -\rho_{A,4} & [27] \\ \frac{\mathrm{d}S_{bu-}}{\mathrm{d}t} &= -\rho_{A,5} & [28] \\ \frac{\mathrm{d}S_{pro-}}{\mathrm{d}t} &= -\rho_{A,6} & [29] \\ \frac{\mathrm{d}S_{ac-}}{\mathrm{d}t} &= -\rho_{A,7} & [30] \\ \frac{\mathrm{d}S_{hco3-}}{\mathrm{d}t} &= -\rho_{A,10} & [31] \\ \frac{\mathrm{d}S_{nh3}}{\mathrm{d}t} &= -\rho_{A,11} & [32] \end{aligned}$$

Algebraic equations. Θ-equation are modified to count for adding of calcium.

$$\begin{aligned} S_{H^+} &= -\frac{\Theta}{2} + \frac{1}{2}\sqrt{\Theta^2 + 4K_W} \\ \Theta &= S_{cat^+} + S_{nh4^+} - S_{hco3^-} - \frac{S_{ac^-}}{64} - \frac{S_{pro^-}}{112} - \frac{S_{bu^-}}{160} - \frac{S_{va^-}}{208} - S_{an^-} + S_{ca^+} \\ S_{nh4^+} &= S_{IN} - S_{nh3} \\ S_{co2} &= S_{IC} - S_{hco3^-} \end{aligned} \quad (3.10)$$

Gas phase equations
Differential equations

$$\begin{aligned}
& & & & \text{State} \\
\frac{\mathrm{d}S_{gas,h2}}{\mathrm{d}t} &= -\frac{S_{gas,h2}q_{gas}}{V_{gas}} + \rho_{T,8} \cdot \frac{V_{liq}}{V_{gas}} & [33] \\
\frac{\mathrm{d}S_{gas,ch4}}{\mathrm{d}t} &= -\frac{S_{gas,ch4}q_{gas}}{V_{gas}} + \rho_{T,9} \cdot \frac{V_{liq}}{V_{gas}} & [34] \\
\frac{\mathrm{d}S_{gas,co2}}{\mathrm{d}t} &= -\frac{S_{gas,co2}q_{gas}}{V_{gas}} + \rho_{T,10} \cdot \frac{V_{liq}}{V_{gas}} & [35]
\end{aligned}$$

Algebraic equations

$$\begin{aligned}
p_{gas,h2} &= S_{gas,h2} \cdot \frac{RT_{op}}{16} \\
p_{gas,ch4} &= S_{gas,ch4} \cdot \frac{RT_{op}}{64} \\
p_{gas,co2} &= S_{gas,co2} \cdot RT_{op} \\
P_{gas} &= p_{gas,h2} + p_{gas,ch4} + p_{gas,co2} + p_{gas,h2o} \\
q_{gas} &= k_p(P_{gas} - P_{atm}) \cdot \frac{P_{gas}}{P_{atm}}
\end{aligned} \tag{3.11}$$

Not standard ADM1 equations Some of the equations above are modified too count for the added lactate and calcium. But there are also differential equations for the new substrates as well. Those equations can bee seen here

$$\begin{aligned}
& & & & \text{State} \\
\frac{\mathrm{d}S_{lac}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(S_{lac,in} - S_{lac}) - \rho_{20} - \rho_{21} & [36] \\
\frac{\mathrm{d}X_{lacf}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{lacf,in} - X_{lacf}) + Y_{lacf}\rho_{20} - \rho_{22} & [37] \\
\frac{\mathrm{d}X_{laco}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(X_{laco,in} - X_{laco}) + Y_{laco}\rho_{21} - \rho_{23} & [38] \\
\frac{\mathrm{d}S_{ca}}{\mathrm{d}t} &= \frac{q_{in}}{V_{liq}}(S_{ca,in} - S_{ca}) - \rho_{P,24} & [39]
\end{aligned} \tag{3.12}$$

3.1.2 Verifying the implementation

Verifying the model is important, but also time consuming. If not done properly the outcome of the model will not be valid and simulation results nonsense. Therefore as a first step files of course were studied carefully in order to find small slips. More significant verifications were done by comparing the outcome of the simulations with the outcome of Rosén and Jeppssons simulations [4]. Since the indata used in the simulation was consistent to the one used in the report by Rosén and Jeppsson the outcome should be equivalent.

To verify the model after adding extra equations for lactate and calcium were those first set to zero and it was controlled that the result still were consistent with the results

before adding them. When turning the contribution of the new equations on, there were no real theoretical equations to compare with, so instead the overall look of the equations were controlled and testing were left as a verification to do when the model is fitted to a real system.

3.2 Simulation

The model build is used in order to test different scenarios. Compared to do tests in a real situation the flexibility of simulations is endless. In a real life situation it is not possible to risk the process ongoing in digester tanks in order to test a lot of new substrates or other aspects of methane production. But in a simulation environment the affects can be investigated and conclusions can be drawn to see if new ideas works and what outcome of them will be.

3.2.1 Sensitivity analysis

The model is built up by a large system of parameters and variables. For a real life adoption of results from simulations to an implementation in a real plant, it is important how parameters affects reliability of results. Are there some parameters defining space, in which calculations done, affecting results more than others? If this is the case, it is more important to study those parameters in order to chose them correctly, compared to parameters not affecting results equally much.

To study this parameter dependence of the model, produced methane gas were chosen as interesting result outcome from the model. By choosing all parameters as they were supposed to be and then increasing them 1% at a time, affect of methane production were investigated. From those recorded affects in gas production the definition of the derivative (3.13) is possible use to investigate parameter dependency, in the equation is the parameter denoted by x.

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(x + \Delta h) - f(x)}{\Delta h} \tag{3.13}$$

However in order to do calculations as easy as possible the following is done instead of using the standard derivative definition

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(1.01x) - f(x)}{0.01x} \tag{3.14}$$

By creating a plot with this method for determining sensitivity (using times 1.01 instead of Δh) the plot for parameter dependency, can be considered a one dimensional Jacobian matrix, containing numerical evaluated partial derivatives of gas production, as a function of each parameter investigated. For parameter p_1,$p_2, \ldots, p_n$ this can be written as

$$J = [\frac{\partial \mathrm{q}_{gas,ch4}}{\partial p_1}, \frac{\partial \mathrm{q}_{gas,ch4}}{\partial p_2}, \ldots, \frac{\partial \mathrm{q}_{gas,ch4}}{\partial p_n}] \tag{3.15}$$

Even though this is a logical and easily motivated method used for determining variable dependence there will be problems comparing different parameter dependencies to each others. The denominator in the definition of the derivative is dimensional dependent, causing all sensitivities having different dimensions and therefore is comparison between different parameters impossible with this method. Instead a similar method using only the numerator in the definition is used. This method is normally referred to as finite difference, having an equation looking like

$$\text{change in gas production} = |f(x \cdot 1.01) - f(x)| \tag{3.16}$$

Using this method the result will be affected of methane gas production depending on parameter tested.

An important aspect with this numerical method of calculating parameter dependency is that it is only valid in one specific position of parameter space and by changing any of the parameters result is not valid any more and a new analysis is needed to be done. Which also is the case if input variables defining flow into the model are changed. Those will also affect the parameter space in which calculations are performed.

Therefore it is interesting to compare results of simulations of dependence for different parts of parameter space. For example if a parameter dependence investigation is performed before an optimization in gas production is done, this dependence evaluation needs to be redone after optimization as well to find if there are any changes in parameter dependency.

When input variables are dynamically changed it will cause changes in parameter sensitivity as well. To find out how much a shift in input variables cause in outflow functions were developed for dynamically changing input variables meanwhile sensitivity were calculated.

Sensitivity analysis implementation

In order to run the sensitivity analysis described above the following pseudo code can be applied

- `Read indata from file`
- `Select start values`
- `Calculate reference gas value for unchanged system,` $f(x)$
- `For each interesting parameter do:`
  - `increment parameter value by 1%`
  - `calculate new value of produced gas,` $f(0.01x)$
  - `calculate change (sensitivity)`
    $|f(x + \Delta h) - f(x)| = |f(0.01x) - f(x)|$
- `Plot result`

3.2.2 Variable dependence of parameters

As stated before, parameter dependence calculated, is only valid in one part of parameter space. However there are sometimes inflows that varies with time. Those flows into the digester affects the position in parameter space in which calculations are performed. To get knowledge of what affects this will have on the system sensitivity multiple analysis needs to be done, one per position investigated. The general dependence of sensitivity coupled to one variable is therefore investigated by gradually changing inflow meanwhile calculating sensitivity of interesting parameters.

Simulations of this type are therefore performed with the following method, where v is variable value, p is parameter value and f(v,p) is the amount of produced CH_4 gas (measured as flow [m^3/d]) as a function of v,p.

- `Read indata`
- `Chose start values`
- `Do a finite number of times:`
  - `select new variable value`
  - `calculate gas production,` $f(v,p)$
  - `increment parameter with 1%`
  - `calculate gas production,` $f(v,1.01p)$
  - `calculate sensitivity` $|f(v,1.01p) - f(v,p)|$
- `Visualize`

3.3 Maximizing gas production

One of the goals with the co digestion together with getting rid of the produced manure is to obtain as much gas as possible. To test and verify this the standard way is to do a trial and error search for optimal settings on the digesting process. This is not optimal and hence there need to be a better and more effective way of optimizing the problem.

In this project simulation is used and around the implemented ADM1 model a Particle Swarm Optimization (PSO) [6] framework is build, which can be used to investigate optimal settings for obtaining as much gas as possible.

Even though the model consists out of a huge number of parameters a large number of those can not be changed during an optimization method in order to find the optimal gas production. Most of the parameters are fixed and are therefore uninteresting in an optimization algorithm. However are the performed sensitivity analysis of those parameters still important in order to know which one of them is most important to obtain reliable results.

The things that can be changed is mixture in the ingoing feed to the digestion chamber, or actually temperature and retention of substrate can be changed as well, but the interesting parts in this project are the mixtures in the ingoing feed. By for example chancing the fraction between carbon and nitrogen the amount of produced gas can be affected. This then illustrates an adding of manure with a typical fraction of nitrogen and carbon to already existing fraction of carbon and nitrogen in the standard manure-free in feed to the digester. There are a lot of variables in the ingoing feed so even though there are a lot of fixed parameters there still are a large set of variables that can be changed during an optimization procedure.

3.3.1 PSO implementation

The dimensions that a particle in PSO can move in will therefore be the concentrations in the ingoing feed to the digestion chamber and also the amount of flow. The interesting property to optimize is the outflow of CH_4 and therefore is amount of produced CH_4 used as the result of an evaluation of a position in the search space.

Normally when implementing an particle swarm optimization method there are fixed intervals in which the particles can be initialized and then is only the velocity at the particles controlled by a maximum velocity [6]. However this is not valid in this specific problem. For example it is only possible to change the mixture in a specific region and therefore it is not interesting to know if the blending of the mixture is more preferable outside this region. So therefore is movement of each particle restricted to a region in which it is initialized.

An other change made to standard PSO-algorithm presented by [6] is that due to the different dimensions that the particles have is the maximum velocity chosen differently in different dimensions. Compared to if the possible propagating dimensions were dimensionless and in same magnitude would it be possible to change the different maximum velocities to one single maximum velocity. But due to the dimensional dependence of the space are different maximum velocities chosen.

Pseudo code

The algorithm used for this implementation of PSO implementation can be seen below, as mentioned above is the target function as high flow of methane gas as possible. As noticeable are the positions in search space evaluated and verified to be in the allowed regions. Without those changes the code is an implementation of the proposed PSO implementation in [6]

- `Initialize particles`
- `Set best global function`
- `Repeat a finite number of iterations`
  - `for each particle do:`
    - `evaluate current value`

```
    * test if global best (yes -> set as new global best)
    * test if particle best (yes-> set as new local best)
  - for each particle and dimension update:
    * calculate new velocity
    * verify velocity < maximum velocity (no -> velocity = maximum ve-
      locity)
    * calculate new position
    * verify min position < position < max position (no -> bring the
      particle back to the boundary)
```

PSO equations

The pseudo code above can be translated into equations. This is also some sort of pseudo code, but it contains some equations as well. In the code there are a lot of notations defined here. The notations follows [6] for maximum readability.

- α - constant in PSO algorithm, often set to 1
- c_1 - parameter in PSO algorithms, the term containing c_1 is normally referred to as the cognitive component
- c_2 - parameter in PSO algorithms, the term containing c_2 is normally referred to as the social component
- r and q - random numbers
- N - number of particles
- n - dimensions
- i - particle number (in swarm)
- j - dimension number
- p - particle
- $\mathbf{x}_i^{pb}$ - local best known position of particle
- $\mathbf{x}^{sb}$ - global best known position of the swarm
- Δt - time step length, for simplicity set to 1

Using this notation one gets:

1. Initialize positions p_i for particle i:
   (a) $x_{ij} = x_{min} + r(x_{max} - x_{min}), i = 1, \ldots, N, j = 1, \ldots, n$
   (b) $v_{ij} = \frac{\alpha}{\Delta t}\left(-\frac{-x_{max}-x_{min}}{2} + r\left(x_{max} - x_{min}\right)\right), i = 1, \ldots, N, j = 1, \ldots, n$

2. Evaluate each particle in the swarm: $f(\mathbf{x}_i), i = 1, \ldots, N$
3. Update best position for each particle and in the swarm: for all particle $p_i, i = 1, \ldots, N$:
   (a) if $f(\mathbf{x}_i) < f(\mathbf{x}_i^{pb})$ then $\mathbf{x}_i^{pb} \leftarrow \mathbf{x}_i$
   (b) if $f(\mathbf{x}_i) < f(\mathbf{x}^{sb})$ then $\mathbf{x}^{sb} \leftarrow \mathbf{x}_i$
4. Update particle velocities and positions:
   (a) $v_{ij} \leftarrow v_{ij} + c_1 q \left(\frac{x_{ij}^{pb} - x_{ij}}{\Delta t} \right) + c_2 r \left(\frac{x_j^{pb} - xij}{\Delta t} \right), i = 1, \ldots, N, j = 1, \ldots, n$
   (b) Restrict velocities so that $|v_{ij}| < v_{max,i}$
   (c) $x_{ij} \leftarrow x_{ij} + v_{ij}\Delta t, i = 1, \ldots, N, j = 1, \ldots, n$
   (d) Restrict positions so that $p_{min,j} < p_{ij} < p_{max,j}$
   (e) Return to step 2, unless maximum number of iterations is fulfilled

3.4 GUI

In order to make results of this thesis available for persons not normally working in `MATLAB` a graphical user interface (GUI) is created. This allows changes to be made to the simulations without actual chancing in source code of the project.

The GUI it self can not be used to evaluate all possible uses of the model build. The most important use of the model will still be to use it as a framework for new and interesting simulations. The model is ideally used together with some interest in simulating some parts of a system, as for example the optimization process done or the sensitivity analysis described above. Those simulations are built upon the model. They uses resources from the model and add something interesting to analysis done. The GUI on the other hand is useful when wanting to test what happens with some set of data as they are evaluated with help of the model.

3.4.1 Basins of the GUI

The GUI are based on the framework of the model. The only things it adds are ability to chose inputs and starting values to simulations and the ability to choose plots. There are no things that can be controlled from the GUI that not can be controlled from the source code it self. The GUI is meant to be used by persons unfamiliar to `MATLAB` but interested in evaluating the model. The more experienced user are recommend using the model by building its own simulation based upon the models framework.

The use of the GUI can be seen in the below pseudo code

- `Start`
- `Iterate until shut down:`

- chose start values
- chose variables in inflow
- chose parameters
- if wanted save parameters and variables
- chose plots wanted
- run simulation with selected parameters and variables
- visualize wanted plots
- save or load parameters

4

Results

The results presented in this section shows typically results when using the implemented model and also results from when the on top of the model implemented simulations are run. The indata to those simulations and tests are proposed by Rosén and Jeppsson [4] as long as not stated otherwise.

4.1 Model implementation

The implemented model can be used as a stand alone model for testing a specific system or it can be used as a base for further applications. The framework provided from this thesis is supposed to be easy to understand and highly adoptable for further applications. It follows the proposed implementation by Rosén and Jeppsson [4] as far as possible in all from nomenclature and notation to structure, however there are changes made to the proposed implementation, but those are described in the Method section, see section 3.

All substrates presented in the model can be seen in table 4.1. From those substrates it is easy to compile plots of simulation results, but also other concentrations are easily calculated, such as gas pressure, gas flow and pH-values.

For gas pressure just use equations in (2.11) and calculated concentrations of different gases, from table 4.1. For gas flow use outflow of total gas from model and multiply it with partial pressure for the investigated type of gas (CH_4, CO_2 or H_2). For pH-values, combine equations (3.5) and (3.10) and use the output from the model.

When running the model with typically values giving rise to steady state solutions one gets the following plots for the components in model, see figure 4.1 to 4.5 and when compiling the result in order to get more info out of the data, as described above one gets the plot in figure 4.6 for produced gas and the plot in figure 4.7 for pH, gas production can be seen as gas flow in figure 4.8.

Table 4.1: Substrates in model

Substrate concentration	Notation	Note
Soluble component sugar	S_{su}	
Soluble component amino acid	S_{aa}	
Soluble component fatty acid	S_{fa}	
Soluble component valerate	S_{va}	
Soluble component butyrate	S_{bu}	
Soluble component propionate	S_{pro}	
Soluble component acetate	S_{ac}	
Soluble component hydrogen	S_{h2}	
Soluble component methane	S_{ch4}	
Soluble component inorganic carbon	S_{IC}	
Soluble component inorganic nitrogen	S_{IN}	
Soluble component inerts	S_I	
Particulate component composites	X_c	
Particulate component carbohydrates	X_{ch}	
Particulate component proteins	X_{pr}	
Particulate component lipids	X_{li}	
Particulate component sugar	X_{su}	
Particulate component amino acids	X_{aa}	
Particulate component fatty acids	X_{fa}	
Particulate component c_4	X_{c4}	
Particulate component propionate	X_{pro}	
Particulate component acetate	X_{ac}	
Particulate component hydrogen	X_{h2}	
Particulate component inerts	X_I	
Soluble component cations	S_{cat}	
Soluble component anions	S_{an}	
Soluble component valerate ions	S_{va-}	
Soluble component butyrate ions	S_{bu-}	
Soluble component propionate ions	S_{pro-}	
Soluble component acetate ions	S_{ac-}	
Soluble component hydrogen carbonate	S_{hco3-}	
Soluble component ammonia ions	S_{nh3}	
Soluble component hydrogen gas	$S_{gas,h2}$	
Soluble component methane gas	$S_{gas,ch4}$	
Soluble component carbon dioxide gas	$S_{gas,co2}$	
Soluble component lactate	S_{lac}	Not included in standard ADM1
Particulate component lactate fermentation	$X_{lac,f}$	Not included in standard ADM1
Particulate component lactate oxidation	$X_{lac,o}$	Not included in standard ADM1
Soluble component calcium	S_{ca}	Not included in standard ADM1

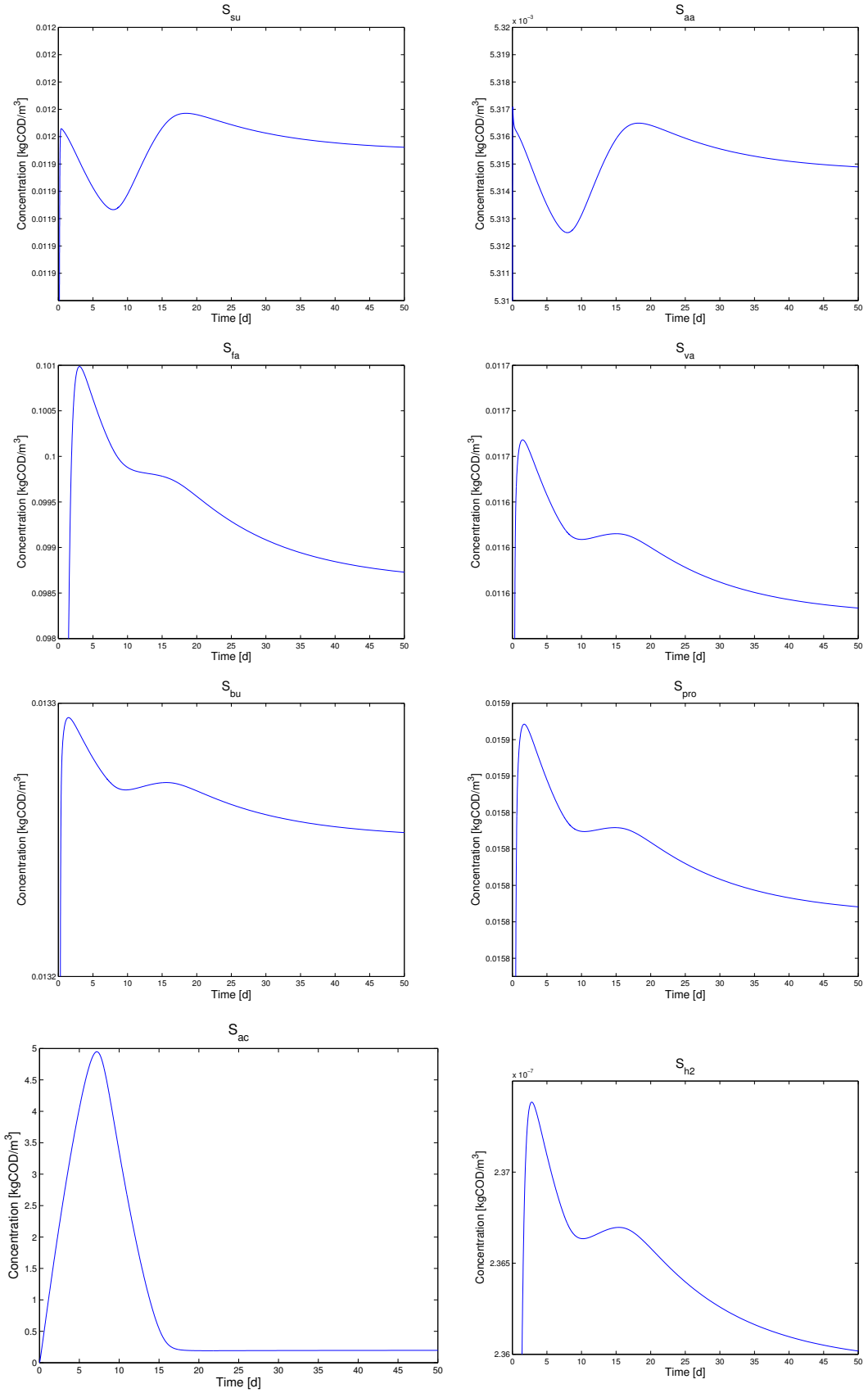

Figure 4.1: Component output from model simulation. Output of substrate number 1-8. Parameters and inflow are defined in appendix B and appendix C.

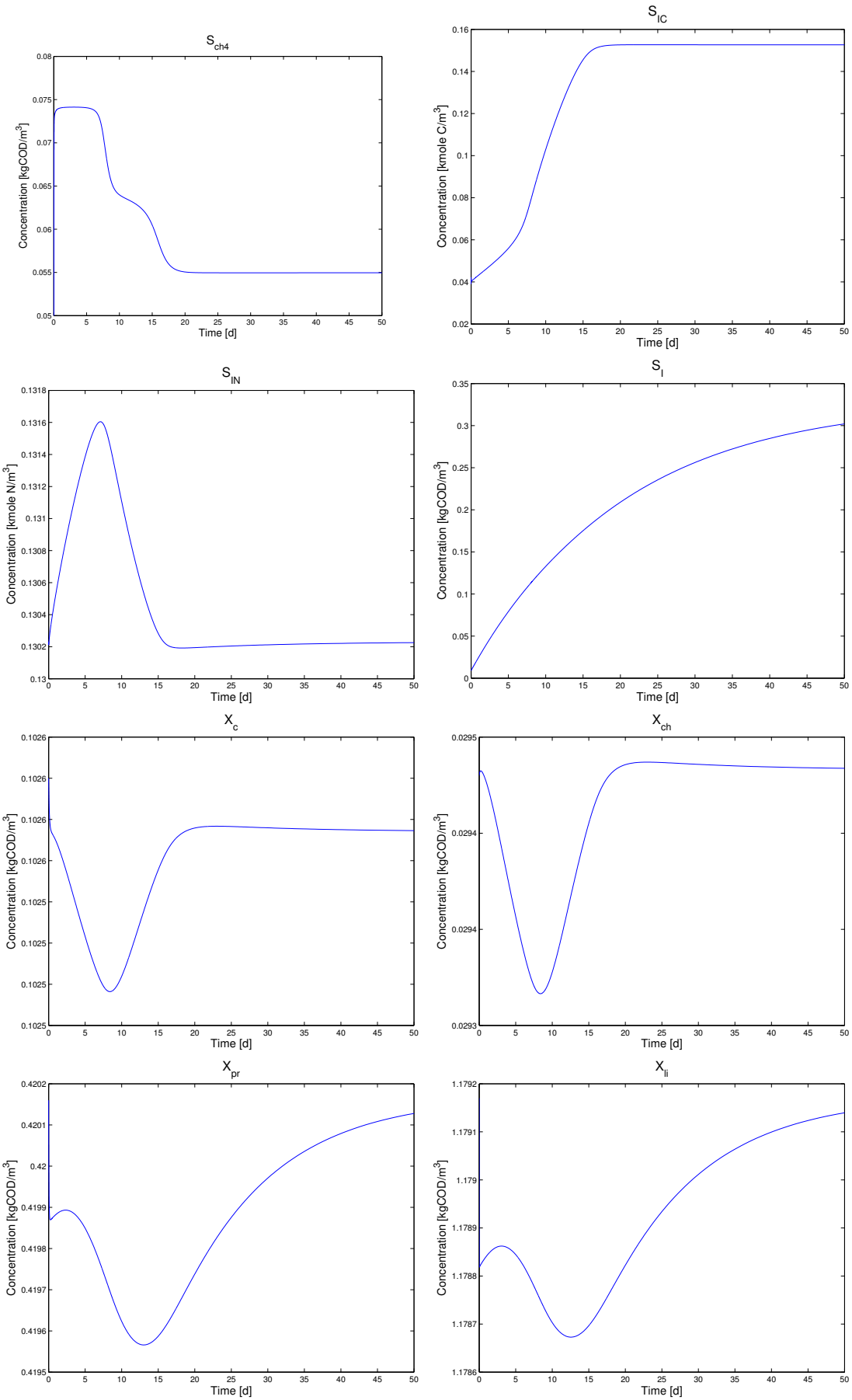

Figure 4.2: Component output from model simulation. Output of substrate number 9-16. Parameters and inflow are defined in appendix B and appendix C.

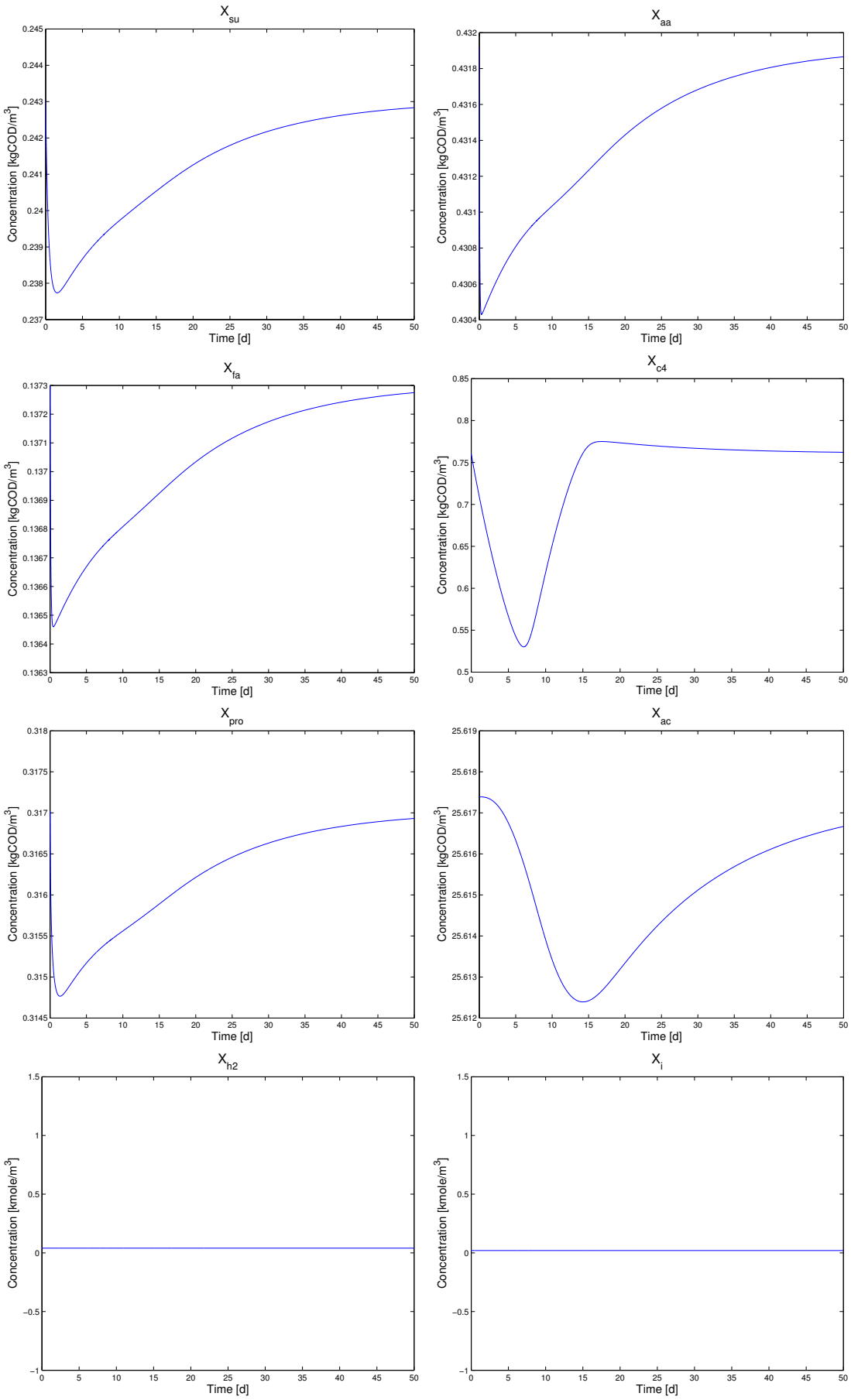

Figure 4.3: Component output from model simulation. Output of substrate number 17-24. Parameters and inflow are defined in appendix B and appendix C.

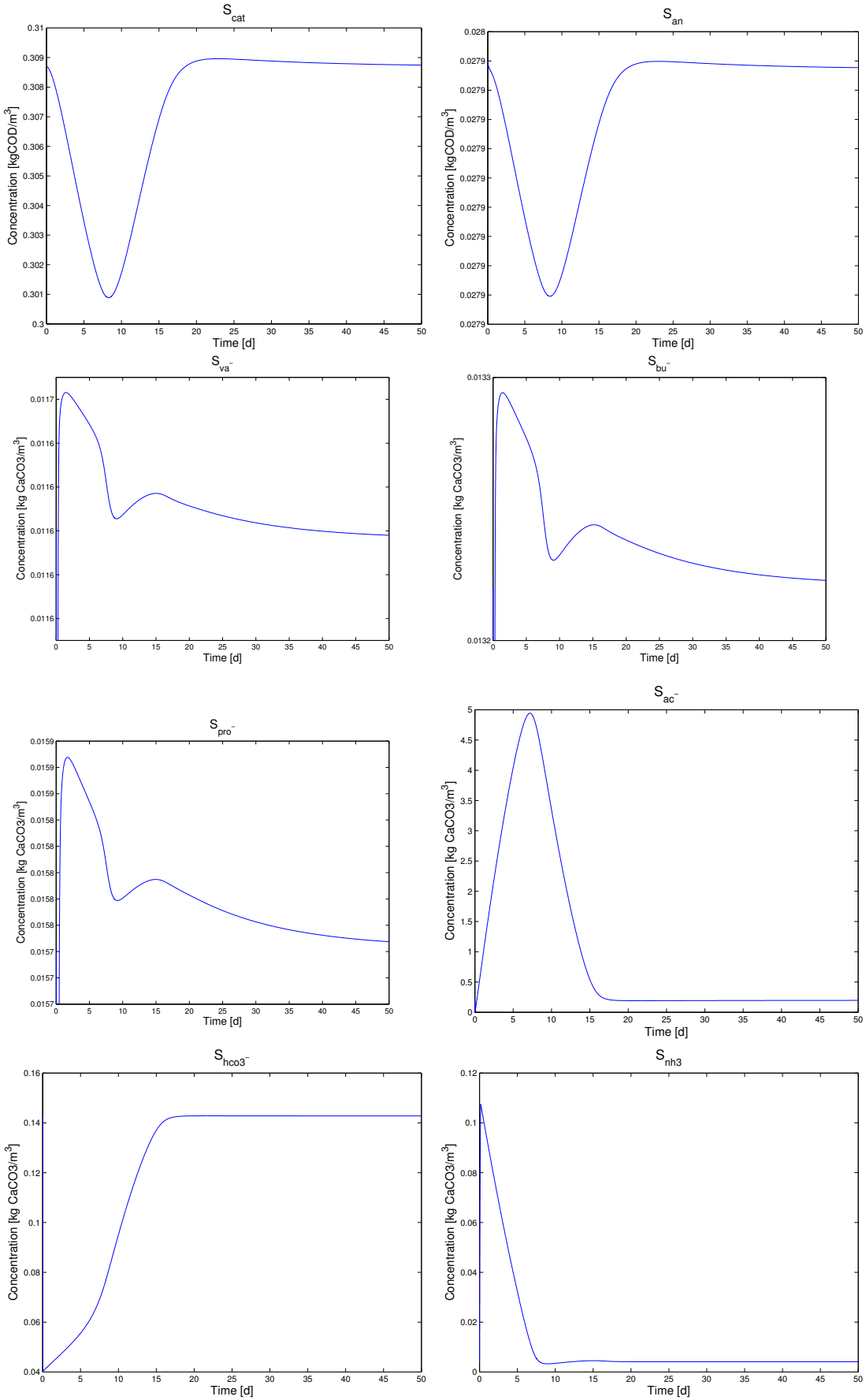

Figure 4.4: Component output from model simulation. Output of substrate number 25-32. Parameters and inflow are defined in appendix B and appendix C.

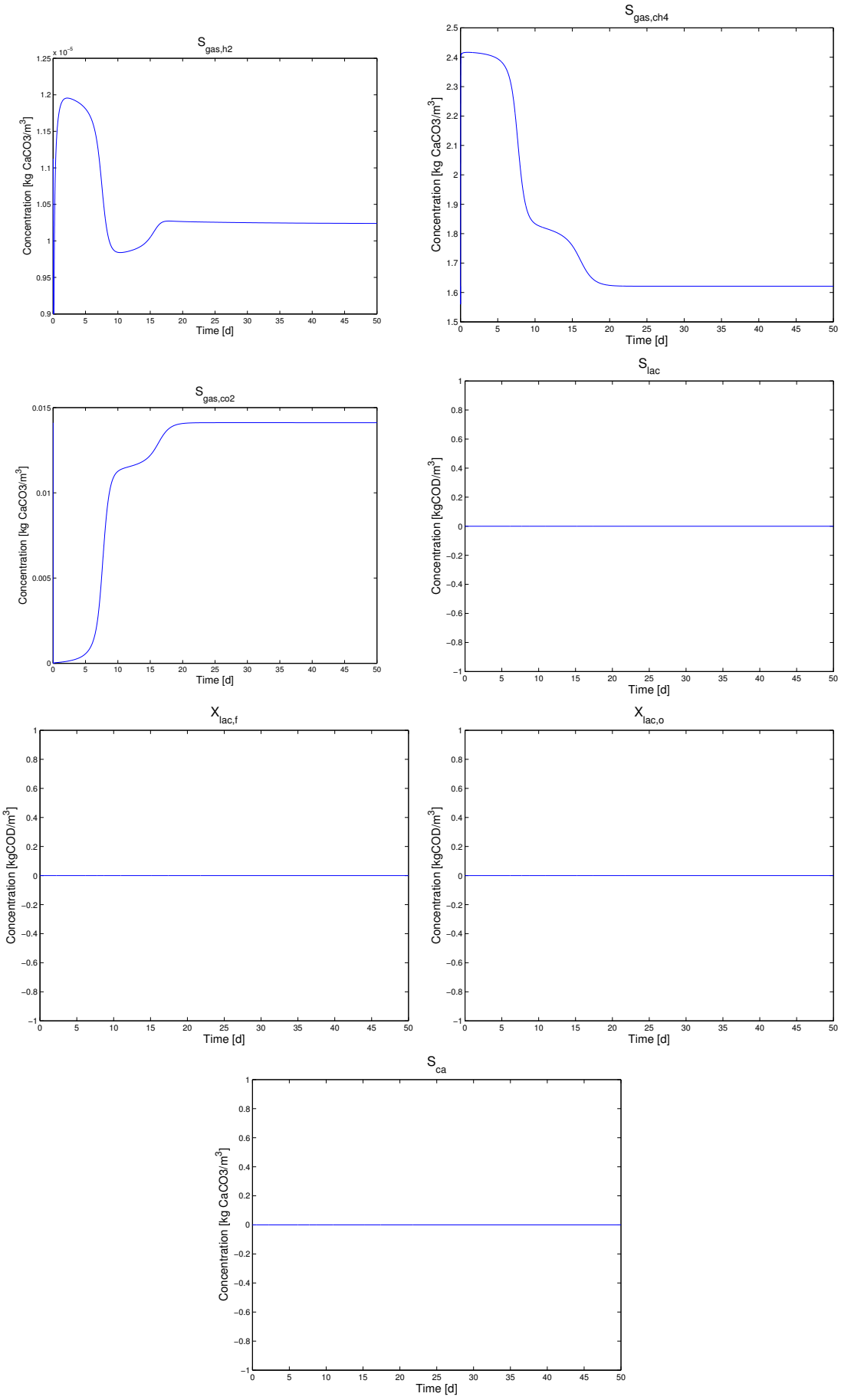

Figure 4.5: Component output from model simulation. Output of substrate number 33-39. Parameters and inflow are defined in appendix B and appendix C.

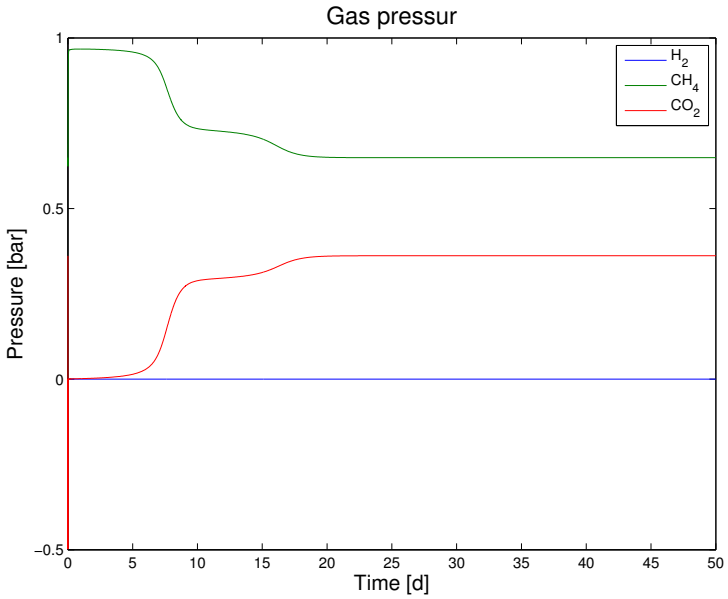

Figure 4.6: Calculated produced gas from model simulation. Parameters and inflow are defined in appendix B and appendix C.

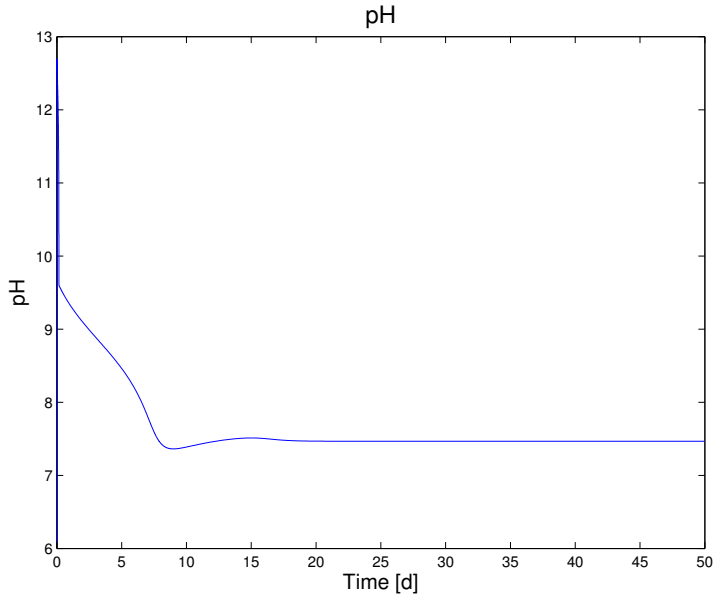

Figure 4.7: Calculated pH from model simulation. Parameters and inflow are defined in appendix B and appendix C.

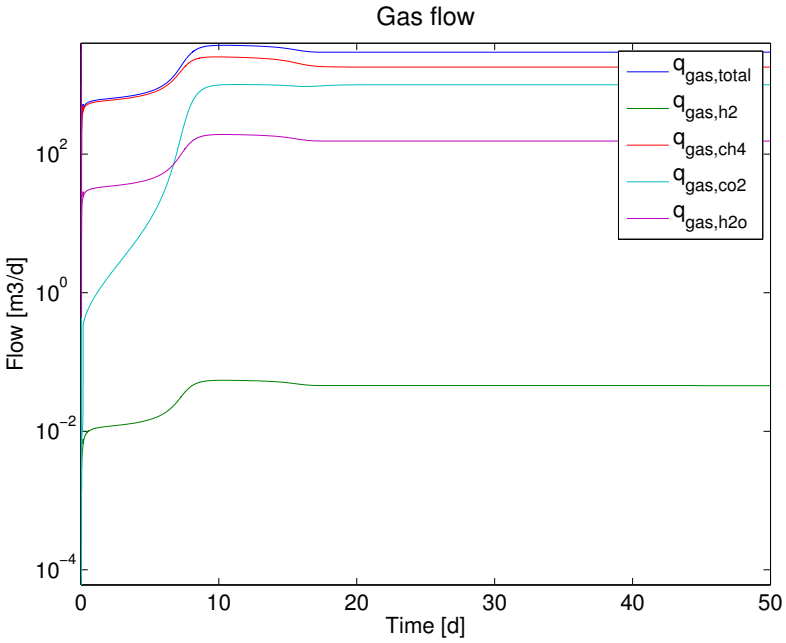

Figure 4.8: Calculated gas production (outflow from model). To see all flows is the plot done with logarithmic scale. Parameters and inflow are defined in appendix B and appendix C.

4.1.1 Verifying the implementation

Verification of the model is an important part of the thesis, due to that it verifies that model results are reliable. The verifying is performed by comparing results from model with results from Rosén and Jeppssons ADM1 implementation [4]. In the report by Rosén and Jeppsson it is stated that results will not be identically due to different choices of entire equation systems and individual equations, they should however correspond even though perhaps not identically. By running the simulation with identical input as in [4], see C.1 in appendix C and turning concentrations of newly added substrates to zero one gets exactly the same case as analyzed by Rosén and Jeppsson. The output should therefor correspond, this comparison can be seen in table 4.2.

There are large correlations between the values even though they not are identically.

Table 4.2: Steady state output comparison between [4] and actual model implementation. The input for creating those values can be seen in table C.1

State No.	Variable	Value [4]	Value simulation	Difference ratio	Unit
1	S_{su}	0.0119548297170	0.0119547850647	0.000003735098520	kgCODm $^{-3}$
2	S_{aa}	0.0053147401716	0.0053147202657	0.000003745427606	kgCODm $^{-3}$
3	S_{fa}	0.0986214009308	0.0986212138885	0.000001896572681	kgCODm $^{-3}$
4	S_{va}	0.0116250064639	0.0116249647170	0.000003591142082	kgCODm $^{-3}$
5	S_{bu}	0.0132507296663	0.0132506818888	0.000003605663497	kgCODm $^{-3}$
6	S_{pro}	0.0157836662845	0.0157836088264	0.000003640365181	kgCODm $^{-3}$
7	S_{ac}	0.1976297169375	0.1977953113077	0.000837200685422	kgCODm $^{-3}$
8	S_{h2}	0.0000002359451	0.0000002359449	0.000000847655533	kgCODm $^{-3}$
9	S_{ch4}	0.0550887764460	0.0549591519906	0.002358559961445	kgCODm $^{-3}$
10	S_{IC}	0.1526778706263	0.1526407718329	0.000243046421703	kmoleCm $^{-3}$
11	S_{IN}	0.1302298158037	0.1302298441112	0.000000217365691	kmoleNm $^{-3}$
12	S_{I}	0.3286976637215	0.3283360943788	0.001101217164028	kgCODm $^{-3}$
13	X_{xc}	0.3086976637215	0.3086976049107	0.000000190512654	kgCODm $^{-3}$
14	X_{ch}	0.0279472404350	0.0279472398610	0.000000020538700	kgCODm $^{-3}$
15	X_{pr}	0.1025741061067	0.1025741055327	0.000000005595954	kgCODm $^{-3}$
16	X_{li}	0.0294830497073	0.0294830488463	0.000000029203221	kgCODm $^{-3}$
17	X_{su}	0.4201659824546	0.4201659524103	0.000000071505793	kgCODm $^{-3}$
18	X_{aa}	1.1791717989237	1.1791717626995	0.000000030720036	kgCODm $^{-3}$
19	X_{fa}	0.2430353447194	0.2430348264437	0.000002132516181	kgCODm $^{-3}$
20	X_{c4}	0.4319211056360	0.4319209645749	0.000000326590074	kgCODm $^{-3}$
21	X_{pro}	0.1373059089340	0.1373058288317	0.000000583386012	kgCODm $^{-3}$
22	X_{ac}	0.7605626583132	0.7605615234530	0.000001492134646	kgCODm $^{-3}$
23	X_{h2}	0.3170229533613	0.3170227140545	0.000000754856953	kgCODm $^{-3}$
24	X_{I}	25.6173953274430	25.6173875632640	0.000000303082388	kgCODm $^{-3}$
25	S_{cat}	0.0400000000000	0.0400000000000	0	kmolem $^{-3}$
26	S_{an}	0.0200000000000	0.0200000000000	0	kmolem $^{-3}$
-	Q	170.00000000000	170.00000000000	0	m ^{3}d $^{-1}$
-	T_{op}	35.000000000000	35.000000000000	0	°C
-	pH	7.4655377698929	7.4665935310362	0.000141397966678	-
-	S_{H+}	0.0000000342344	0.0000000341512	0.002436224788587	°C
27	S_{va-}	0.0115962470726	0.0115962750850	0.000002415637763	kgCODm $^{-3}$
28	S_{bu-}	0.0132208262485	0.0132208510214	0.000001873774991	kgCODm $^{-3}$
29	S_{pro-}	0.0157427831916	0.0157428248907	0.000002648768584	kgCODm $^{-3}$
30	S_{ac-}	0.1972411554365	0.1974073666054	0.000841970448004	kgCODm $^{-3}$
31	S_{hco3-}	0.1427774793921	0.1427652639943	0.000085562814499	kmoleCm $^{-3}$
-	S_{co2}	0.0099003912343	0.0098755078386	0.002519707958991	kmoleCm $^{-3}$
32	S_{nh3}	0.0040909284584	0.0041005728913	0.002351972067235	kmoleNm $^{-3}$
-	S_{nh4+}	0.1261388873452	0.1261292712198	0.000076240235966	kmoleNm $^{-3}$
33	$S_{gas,h2}$	0.0000102410356	0.0000102361173	0.000480484919805	kgCODm $^{-3}$
34	$S_{gas,ch4}$	1.6256072099814	1.6212531698151	0.002685601636663	kgCODm $^{-3}$
35	$S_{gas,co2}$	0.0141505346784	0.0141147460189	0.002535551079139	kmoleCm $^{-3}$
-	$p_{gas,h2}$	0.0000163991826	0.0000163913070	0.000480474192815	bar
-	$p_{gas,ch4}$	0.6507796328232	0.6490365791240	0.002685601636740	bar
-	$p_{gas,co2}$	0.3625527133281	0.3616357673681	0.002535551078571	bar
-	p_{gas}	1.0690164904089	1.0663564828741	0.002494482452651	bar
-	$q_{gas}{}^{*}$	2955.70345419378	2964.86535854150	0.003090158654703	Nm ^{3}d $^{-1}$

* = In order to being able to compare the outflow of gas from the model to Rosén and Jepssons result, the calculation of gas flow is changed to atmospheric pressure instead of headspace pressure as normally used in the model.

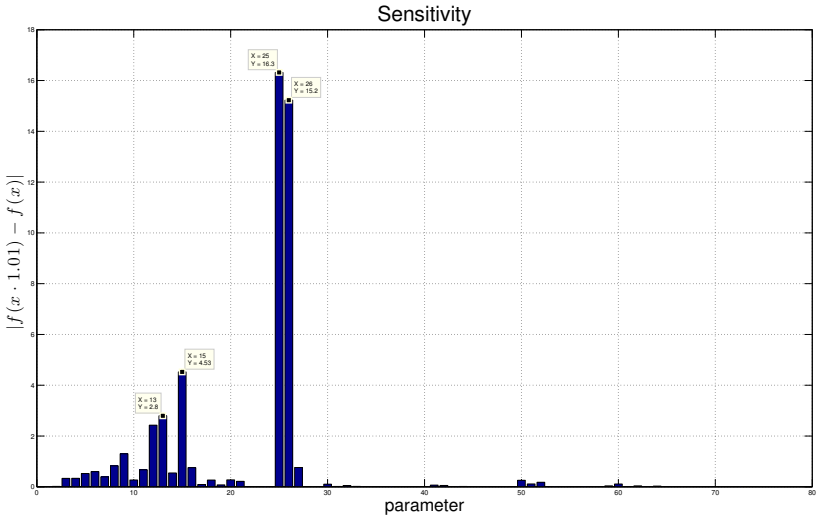

Figure 4.9: Sensitivity analysis for ADM1 model, with input to the model from [4]. Parameter values corresponds to values in table 4.3.

4.1.2 Sensitivity analysis

Sensitivity analysis can be performed for different positions in parameter space and results are depending on which position in space investigated. Therefore the sensitivity analysis presented here is just a representation of a possible result. To get adequate results for a specific implementation it is crucial to do a new sensitivity analysis for that case. Sensitivity is measured in affect of produced methane through numerical calculation of derivative or finite difference, see section 3.2.1 for further explanation.

The parameters used when doing the sensitivity analysis presented here comes from [4]. Variables and parameters used can be seen in table C.1 (variables) and in appendix B (parameters). Outgoing from those are the sensitivity analysis in figure 4.9 created. On the figures x-axis a lot of parameters are presented as numbers, those numbers can be coupled to a parameter by table 4.3.

Table 4.3: Parameter denomination in sensitivity analysis plot. The number in the table corresponds to a position on the x-axis in figure 4.9.

Number	Parameter	Number	Parameter	Number	Parameter
1	$f_{sI,xc}$	26	P_{atm}	51	$K_{S,ac}$
2	$f_{xI,xc}$	27	k_{p}	52	$K_{I,nh3}$
3	$f_{ch,xc}$	28	$k_{L,a}$	53	$pH_{UL,ac}$
4	$f_{pr,xc}$	29	V_{gas}	54	$pH_{LL,ac}$
5	$f_{li,xc}$	30	k_{dis}	55	$k_{m,h2}$
6	$f_{h2,su}$	31	$k_{hyd,ch}$	56	$K_{S,h2}$
7	$f_{bu,su}$	32	$k_{hyd,pr}$	57	$pH_{UL,h2}$
8	$f_{pro,su}$	33	$k_{hyd,li}$	58	$pH_{LL,h2}$
9	$f_{ac,su}$	34	$K_{S,IN}$	59	$k_{dec,X,su}$
10	Y_{su}	35	$k_{m,su}$	60	$k_{dec,X,aa}$
11	$f_{h2,aa}$	36	$K_{S,su}$	61	$k_{dec,X,fa}$
12	$f_{va,aa}$	37	$pH_{UL,aa}$	62	$k_{dec,X,c4}$
13	$f_{bu,aa}$	38	$pH_{LL,aa}$	63	$k_{dec,X,ac}$
14	$f_{pro,aa}$	39	$k_{m,aa}$	64	$k_{dec,X,h2}$
15	$f_{ac,aa}$	40	$K_{S,aa}$	65	$k_{m,lac,f}$
16	Y_{aa}	41	$k_{m,fa}$	66	$K_{S,lac,f}$
17	Y_{fa}	42	$K_{S,fa}$	67	$k_{m,lac,o}$
18	Y_{c4}	43	$K_{I,h2,fa}$	68	$K_{S,lac,o}$
19	Y_{pro}	44	$k_{m,c4}$	69	$K_{I,h2,lac,o}$
20	Y_{ac}	45	$K_{S,c4}$	70	$K_{I,vfa}$
21	Y_{h2}	46	$K_{I,h2,c4}$	71	$k_{dec,X,lac,f}$
22	$Y_{lac,f}$	47	$k_{m,pro}$	72	$k_{dec,X,lac,o}$
23	$Y_{lac,o}$	48	$K_{S,pro}$	73	$K_{S,p,caco3}$
24	T_{base}	49	$K_{I,h2,pro}$	74	$K_{r,caco3}$
25	T_{op}	50	$k_{m,ac}$		

4.1.3 Variable dependence

The sensitivity analysis done is only valid in one part of parameter space. To get information about what happens if the position is changed by changes in inflow is method for testing this produced. This method varies one or more variables meanwhile sensitivity is plotted for one or more parameters.

For illustrating a typically result when doing this is the plot in figure 4.10 constructed. The plot illustrates change in sensitivity for two parameters when the amount of inorganic nitrogen in inflow is changed. To do this analysis are the inflow of particulates changed from Rosén and Jeppssons choice of variables C to no inflow of particulate components except X_{xc}, which were changed to 2.0 kg COD m^{-3}.

The result plotted can be a bit hard to read out of figures, due to fluctuations. So to make it more easy to identify tendencies is also a 50 point average plotted, as a red line above the others. The averaging is performed be using the function `smooth` in `MATLAB`, which makes an average over surrounding points. The slopes of those lines indicates overall change of sensitivity for investigated parameters.

From the plot it is shown that there seems to be something interesting happening in the region between 0-1 kmoleNm^{-3}, so therefore is a new plot done in this region, see figure 4.11. This is the way in which the simulation of variable dependence is supposed to be used; Start by choosing a large interval and then look carefully at the interval in which there seams to happen something interesting. As a result of this more careful investigation on the interval 0-1, it can be seen that the sensitivity are large in a small interval around 0.05 kmoleNm^{-3}.

4.1.4 Maximizing gas production

In order to maximizing gas production it is important to notice that there are multiple variables possible to modified. Due to the objective function chosen, maximizing gas production, it is possible that different parts of the system will cancel out positive affects done by other parts of the same system.

The affect of this is that all interesting parameters needs to be optimized at same time to find an optimal choice, which is done by PSO. Results of the PSO algorithm chosen are limited to a predefined range for all variables, which easily can be modify to fir different dimensions.

As mentioned before the optimal mixture found by this process is only valid in one part of the parameter space, so by changing the parameters, the optimization process needs to be redone.

In order to build a so useful method as possible, a method allowing changes to be done in any dimension wanted were created. The method allows ranges to be individually defined for all substrates presented in the model. The ranges are actually also possible to set to a finite value if one not wants to optimize in any of the dimensions presented. The result presented here shows how the optimal mixture of three different input variables propagates during a simulation and how the resulting gas production is changed during the simulation, see figure 4.12. In the simulation there are 20 PSO particles used and

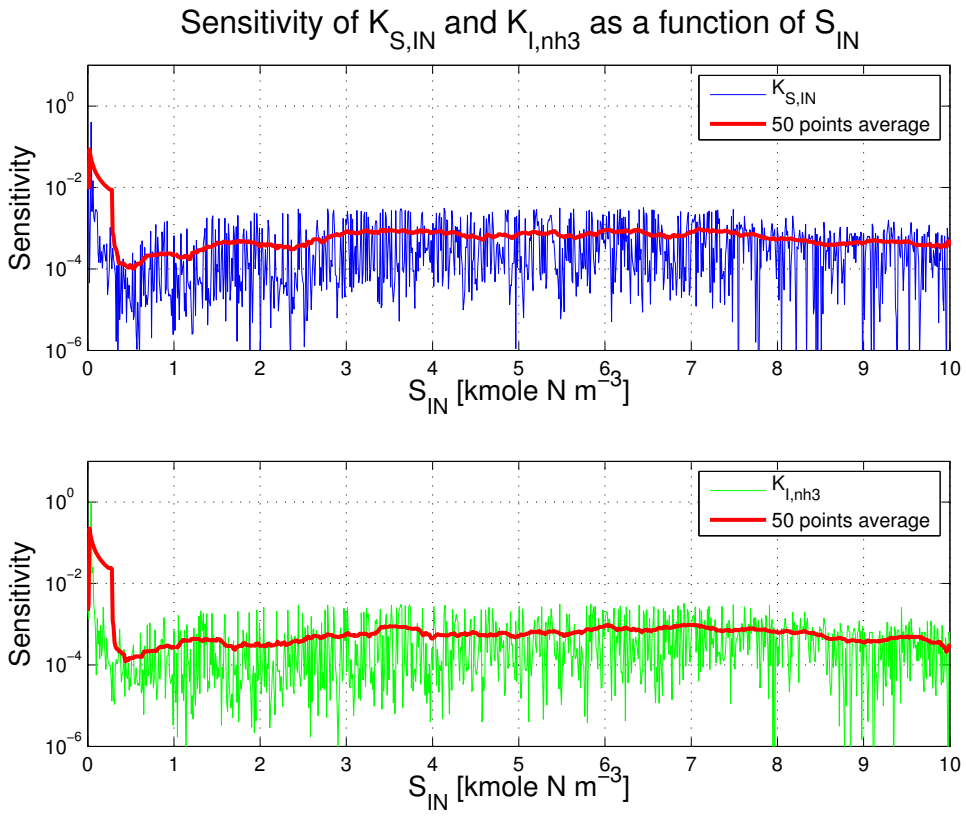

Figure 4.10: Variable dependence of sensitivity for two parameters. Sensitivity is constantly tested meanwhile a variable is changed. In this case the variable changed is S_{IN} meanwhile $K_{S,IN}$ and $K_{I,nh3}$ are investigated during this process. The fluctuating of the curve is reduced by a 50-point average plotted above the curves, showing the trend of the parameters. The values in the simulation are as explained in section 4.1.3.

the simulation was let run until it had reached the boundaries of the box, which were [0,1] kmole/m^3 for N,C and [100,200] m^3/d for the flow.

In the plot it is possible to see that the optimal position not lies in the selected interval, due to that the particles did not reached a best position inside the boundaries, instead the particles were relocated to the boundaries. In order to find the theoretical best position, the interval therefore needs to be extended so that a best position is possible to find inside the interval.

4.1.5 GUI

For people not that skilled in `MATLAB` a basic GUI is create. In this GUI it is only possible to run the model and change variables and parameters. There is no possibilities to do the more advanced simulations build upon the model, such as PSO or sensitivity analysis, but all standard plots of substrates concentrations presented in this report are

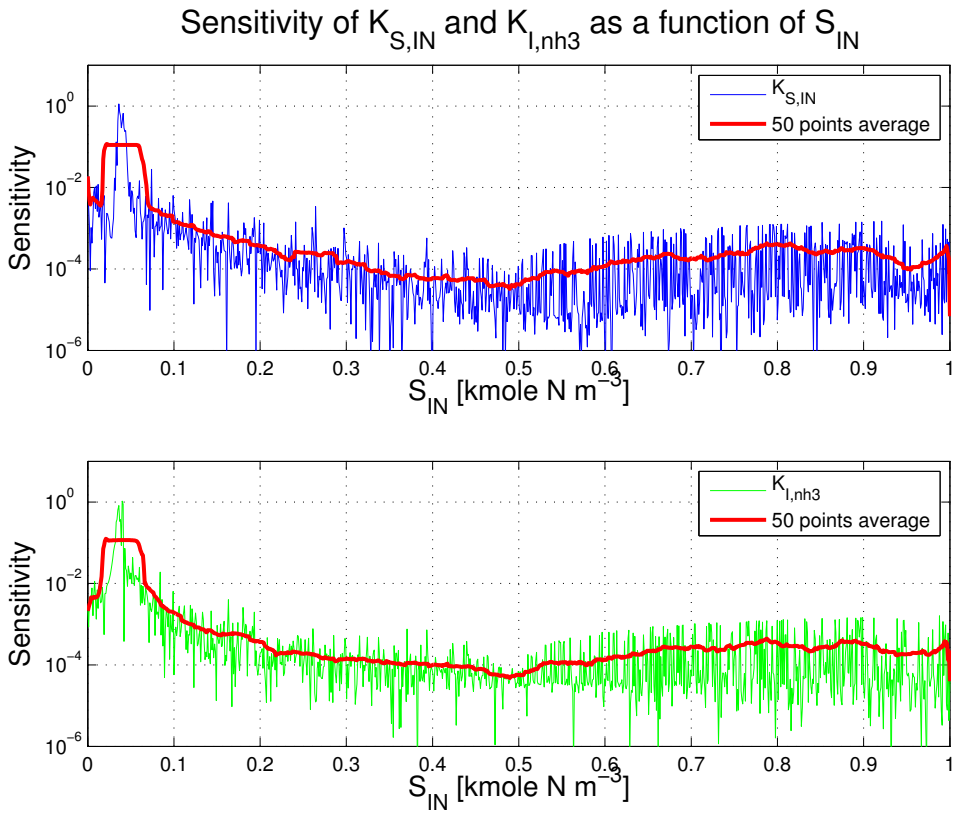

Figure 4.11: Variable dependence of sensitivity for two parameters. Sensitivity is constantly tested meanwhile a variable is changed. In this case the variable changed is S_{IN} meanwhile $K_{S,IN}$ and $K_{I,nh3}$ are investigated during this process. The fluctuating of the curve is reduced by a 50-point average plotted above the curves, showing the trend of the parameters. The values in the simulation are as explained in section 4.1.3.

possible to draw and also some additional plots are possible to draw.

It is also possible to save sets of analyzed parameters and variables for further use and of course also load old saved sets. From start the model is loaded with the default values used in this report.

The main window in the GUI looks like figure 4.13 and by pressing each of the lower left buttons different sets of parameters can be set in new windows. The tables in the main windows consists of start values (the mixture in the digestion chamber at start) and inflow values (what the flow into the chamber consists of). The plot selection part of the GUI makes it possible to see how concentrations for substrates varies during simulation. By activating any (one or more) of the boxes the run button becomes active and the model can be analyzed. Each of the plots shows up in a new own window.

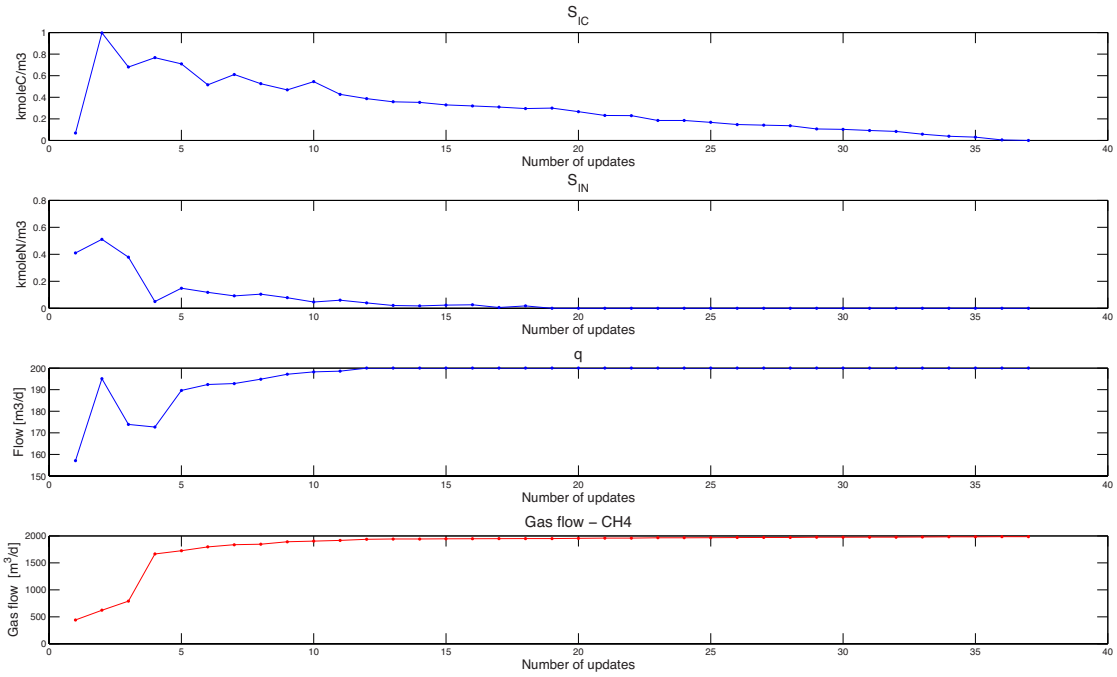

Figure 4.12: Propagation of optimal mixture in invariable to PSO optimization of gas production. The parameters are chosen according to appendix B and the inflow values to the simulation are defined in C.

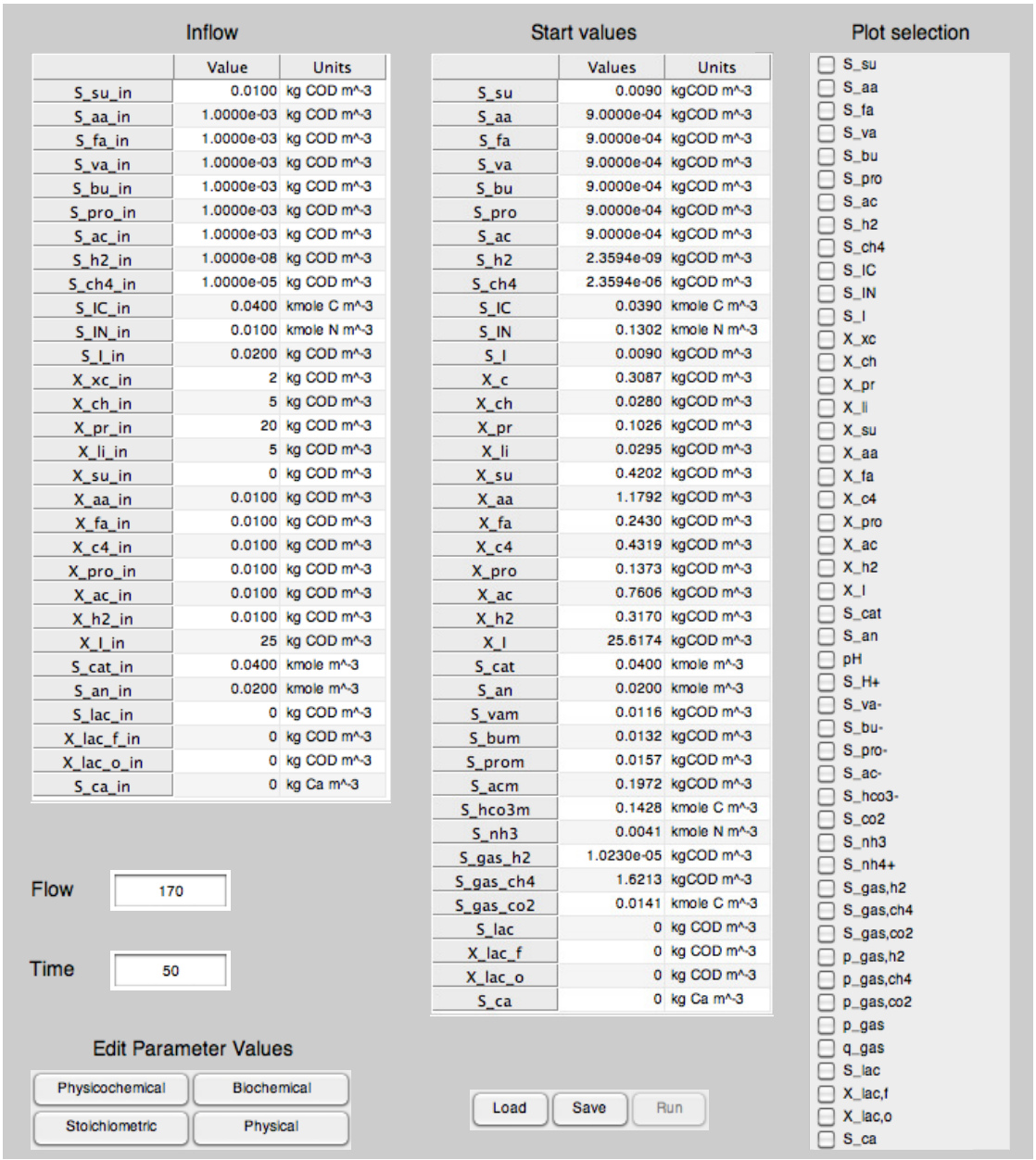

Figure 4.13: Main window in GUI.

5

Discussion

All discussion for time consumptions done here is depending on the computer used. The results presented here are done on a MacBook Pro with a 2.53 GHz Intel Core 2 Duo processor and 4 GB 1067 MHz RAM.

5.1 Model implementation

The model implemented can be used as a stand alone model to test different inputs in a digestion chamber. The typical results from this type of test are presented in the report and also compared to Rosén and Jeppsons results [4]. An interesting part of the usability of the model is that combination of start values and steady state infeeds.

During testing phase of the model a large range of start values were tested and evaluated and the results of them were compared to see if the model is robust enough to handle different kinds of inputs by itself. The results of those simulations were that the model seemed to handle different start values in a good way.

By setting unknown concentrations of ions to zero the model was self correcting for problems in the balance of anions and cations. This is a good behavior of the model due to it makes it possible to handle problems with correct start values in an easy and effective way.

The time it took to run a standard simulation was around three seconds, depending on the complexity of the choices of the model parameters and how long from steady state the start was. The model as it is implemented uses the ODE-solver `ode15s` in `MATLAB`, which is a smart solver that uses different step length depending on stiffness of problems and complexity of concentration curves.

5.2 Verifying the implementation

By using the same infeeds as Rosén and Jeppsson[4] the correctness of the present simulation were evaluated. The solutions for different substrates do not vary much compared to Rosén and Jeppssons results. The largest difference is around 3% and every other is smaller and all solutions are therefore considered as good solutions.

5.3 Sensitivity analysis

The sensitivity method produced during the thesis gives clues on which parts that are important in a real world situation when implementing the model to an existing plant. The method gives clues on which parameters that are important to choose correctly due to their effect on the final result.

The method used for performing sensitivity analysis is a finite difference method. The more mathematical logical method using the definition of the derivative requires dimensionless units and due to that the model is built up with dimensional properties it is easier to keep working with dimensions and therefore the method used is dimensional dependent.

The results are obtained by changeling each parameter 1% at a time and then calculate the effect on methane gas production with the new parameter set obtained. In order to visualize the results a bar graph is compiled, which can be seen in figure 4.9. If there are parameters equivalent to zero in the standard parameter set, those are kept zero at comparison implying no measurable change.

Large bars for measured sensitivity correlates to important parameters in the implementation. Those parameters are therefore more important to measure correctly when the model is implemented to a real situation.

The time consumption of doing a sensitivity analysis depends on how many parameters that are investigated, but normally it takes around three seconds per parameter to be evaluated and three seconds for the reference as well. So by simple math it is possible to calculate the time consumption.

5.4 Variable dependence

The sensitivity analysis done in this work is only valid in one position in parameter space. The drawback of this is clear, if one variable in infeed is changed, the entire sensitivity analysis needs to be redone. Therefor it is interesting to see how the sensitivity changes with changing variables. For this a framework is implemented. To test this framework, only one variable were changed meanwhile the sensitivity for two different parameters were evaluated and plotted.

It was supposed that the sensitivity for the two parameters ($K_{S,IN}$ and $K_{I,nh3}$) would decrease with increased values of the variable (S_{IN}). The reason to suppose this came from the project participate Karin Willquist at *SP*. To test if this was possible to see, firstly a large interval was created and evaluated, see figure 4.10. This did not show

much more than noise and a relatively constant signal (notice the logarithmic y-axis). But there seemed to be something interesting in the low region around 0.5 kmole N m^{-3}. So a second figure was created, see figure 4.11, to investigate this interval and here it was possible to see that something is happening in this region, larger values of S_{IN} will contribute to less sensitivity of the parameters, as expected.

The test done indicates that the method to investigate variable dependence of parameters can be done in a clear and understandable manner.

The time consumption for doing the analysis is relatively large. In the analyzed interval were 1000 points investigated, and by doing three analyses at each point (one per parameter and one reference) 3000 evaluations of the model were performed. The time consumptions of those investigations varied depending on position in space. Some investigations took about 20 seconds and some about 2 seconds, leading to a total time consumption of about 3 hours.

5.5 Maximizing gas production

The goal of the entire project is to minimize waste from keeping animals meanwhile the gas production in digestion chambers is increased. To achieve this it is important to investigate which is the most preferable mixture into the digester, in order to get as much gas out as possible. To know this the normal working manner is a trail and error procedure. In order to make the search more effective an optimization framework is implemented on top of the ADM1 model.

The optimization framework chosen is Particle Swarm Optimization (PSO). The reason for choosing PSO in front of other methods is that very little about the problem needs to be known. The only thing needed is a function value for each position in search space and in this case amount of produced methane was chosen as wanted quantity to optimize against.

The chosen algorithm is not the fastest possible and it might not ever find the optimal solution, due to that it can be trapped inside a local maximum, but it is however a smart algorithm that allows for optimization of a large set of variables simultaneously. Compared to optimize in each dimension, one at the time, this is a far better alternative, depending on that different quantities affect each others. So by using PSO the global best combination can be found.

The result presented here, in figure 4.12, shows how the solution propagates to the best possible solution. To compute the picture three different dimensions were chosen and the solution were plotted to get a feeling on how the algorithm works. Compared to doing a real simulation where the only output will be the now best known position this is a god way of illustrating how the simulation works. However in a real situation the actual best found mixture in numbers is more interesting and therefore the plots are only constructed to get a feeling for the algorithm and its propagation.

The amount of possible things to optimize as the method is written are all input variables along with the flow. The algorithm does not need to be rewritten to count for different sets of variables to optimize in. The only corrections needed to be done are

choosing the intervals in which the concentrations lies to be fixed values (for things that not are interesting to optimize in) or a finite interval (for thing that one are interesting to optimize).

The time consumption for the PSO method varies depending on what is optimized and how many steps needed for convergence, but also randomly chosen start positions in search space effects the simulation time. As before each simulation takes about three seconds and it seams sufficient to use about 20 particles in the optimization implying the initialization process to taking around one minute and then the actual process takes far more time. About 10-20 minutes were used when testing the method.

5.6 GUI

The GUI constructed is for people not experienced in `MATLAB`. The possibilities to do things in the GUI are limited, but there are possibilities to do all basic simulations of the model. However the more advanced features, with the constructed ADM1 framework, are for people understanding what the simulations does and therefor this is not included inside the GUI.

The output from the GUI is exactly the same as one gets when using the model in own simulations, with one major difference. In the GUI the only thing possible to obtain is the plots for different substrates and when using the model in `MATLAB` one gets substrate concentrations as well. Those substrate concentrations can be used for a wide range of things, for example plotting, but also a lot of other things that not can be done with help of the GUI. On the other hand, the GUI can be run without having access to `MATLAB`. By compiling the GUI in an `Windows` environment and packing the `MATLAB Compiler Runtime - mcr` [8] with the executable file everyone can run the file, regardless if they have a `MATLAB` license or not.

Time consumption varies with what is being done. If it is a complex system that is being solved it will take more time and if it is a simple system it will take less. But around three seconds per run is what was measured during execution.

5.7 Usability of results

The results of this thesis are mostly in the methods that can be used to test and verify different mixtures in a digesting process. The process itself is only tested in terms of typical data of which the correctness is not verified in this thesis. To get real usage of the methods the process needs to be fitted to the specific case in each process. This fitting process is performed in multiple steps of which fitting the parameters to the actual case is the first step. When this is done, verifying the implementation is another important step that needs to be performed to achieve correct and reliable results.

When fitting the model to processes, the knowledge about sensitivity parameters are important due to their affect on the final result.

The functions and methods developed during this thesis will be used in the grater project on manure in Borås. However this is not part of the thesis and therefor not

included in this report.

5.8 Further development

5.8.1 Fitting the model to the project

In order to use the methods created it is important to know about the systems that it is supposed to be used on. This requires some lab analysis to be done and also some converting to standard ADM1 units, that not always are the same as used in lab scale experiments. By doing this the model can be used to test and verify the usability of different raw materials for digesting processes.

5.8.2 Building more methods and simulations

There is an important bit of programming that needed to be done and it is implementing possibilities to choose different amount of mixtures between manure and standard waste water sludge as infeeds to the model. As the case is now it is possible to change different amounts of substrate concentrations in the infeed. But to really get a lot out of the model different manure types needs to be tested and then those results needs to be converted to substrates that can be added to the standard in feeds.

By changing the amount of added manure to the infeed the affect of adding manure can be investigated and an optimal mixture be found. Those analyses has not been performed at the moment and therefore are those parts not done in this thesis. At the moment this can be implemented by firstly chancing the concentrations that works as infeed to the model.

It is proposed that a framework that uses two set of variables, one for the manure and one for the waste water, are developed. By then defining a level of mixing between those sets of variable the actual infeed to the digester can be determined.

6

Conclusion

An anaerobic digestion system is mathematically described and implemented using ADM1 model, with some small changes done. On top of this modeling system several simulations are implemented, evaluating for example stability of process and optimal mixture. All of those systems are implemented using `MATLAB`.

To increase usability of system the way the model is implemented follows Rosén and Jeppssons [4] implementation. However also some additional equations added, those are also presented in the same way.

For unexperienced `MATLAB`-users there is a GUI developed that implements and runs the model as written.

The execution time varies between different simulations and systems but the average time consumed to evaluate one set of variables and parameters is three seconds, on a MacBook Pro with a 2.53 GHz Intel Core 2 Duo processor and 4 GB 1067 MHz RAM.

Bibliography

[1] J. I. för jordbruks-och miljöteknik, Energiproduktion med hästgödsel - från gödsel till värme på 4 dagar, http://www.bioenergiportalen.se/?p=6527&m=1723 (04 2014).
URL http://www.bioenergiportalen.se/?p=6527&m=1723

[2] P. Wennerberg, C. Dahlander, Hästgödsel som en resurs - en förstudie om olika hanteringskedjor för hästgödsel, Tech. rep. (2013-05-07).

[3] D. Batstone, J. Keller, R. Angelidaki, S. Kalyuzhnyi, S. Pavlostathis, A. Rozzi, W. Sanders, H. Siegrist, V. Vavilin, Anaerobic digestion model no1, isbn: 1900222787, Tech. rep. (2002).

[4] C. Rosén, U. Jeppsson, Aspects on adm1 implementation within the bsm2 framework, Tech. Rep. 7224, Dept. of Industrial Electrical Engineering and Automation, Lund Institute of Technology (2005).

[5] K. Willquist, V. N. Nkemka, H. Svensson, S. Pawar, M. Ljunggren, H. Karlsson, M. Murto, C. Hulteberg, E. W. van Niel, G. Liden, Design of a novel biohythane process with high {H2} and {CH4} production rates, International Journal of Hydrogen Energy 37 (23) (2012) 17749 – 17762.
URL http://www.sciencedirect.com/science/article/pii/S0360319912019428

[6] M. Wahde, Biologically inspired optimization methods, WIT Press, Southampton, UK.

[7] H. Rylander, W. Wiqvist, Substrahandbok för biogasproduktion, Report, Avfall Sverige, Malmö (Jun 2009).

[8] I. The MathWorks, Matlab compiler runtime (mcr), http://www.mathworks.se/products/compiler/mcr/ (04 2014).
URL http://www.mathworks.se/products/compiler/mcr/

[9] S. S. Johansson, Modelling of step response in modelling of step response in anaerobic digesters, Msc. thesis (May 2012).

A

Biochemical Rate Matrix

In order to fit the biochemical rate matrices to the pages in this report are the sides made blank, meanwhile they also are rotated 90 degrees. To achive this are the matrices placed at own pages, so to see them continue to next page.

Table A.1: Biochemical rate coefficient ($\nu_{i,j}$) and kinetic rate equations (ρ_j) for soluble components. The matrix comes from the task report [3] with complementing equations from Willquist *et al.*[5]

Component→	**i**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Rate (ρ_j kgCODmm$^{-3}\cdot$d^{-1})**
j	**Process↓**	**S_{su}**	**S_{aa}**	**S_{fa}**	**S_{va}**	**S_{bu}**	**S_{pro}**	**S_{ac}**	**S_{h2}**	**S_{ch4}**	**S_{IC}**	**S_{IN}**	**S_I**	
1	Disintegration												$f_{sl,xc}$	$k_{dis}X_c$
2	Hydrolysis of carbohydrates	1												$k_{hyd,ch}X_{ch}$
3	Hydrolysis of proteins		1											$k_{hyd,pr}X_{pr}$
4	Hydrolysis of lipids	1 - $f_{fa,li}$		$f_{fa,li}$										$k_{hyd,li}X_{li}$
5	Uptake of sugar	-1				$(1-Y_{su})f_{bu,su}$	$(1-Y_{su})f_{pro,su}$	$(1-Y_{su})f_{ac,su}$	$(1-Y_{su})f_{h2,su}$		$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,5}$	$-(Y_{su})N_{bac}$		$k_{m,su}\frac{S_{su}}{K_s+S_{su}}X_{su}I_1$
6	Uptake of amino acids		-1		$(1-Y_{aa})f_{va,aa}$	$(1-Y_{aa})f_{bu,aa}$	$(1-Y_{aa})f_{pro,aa}$	$(1-Y_{aa})f_{ac,aa}$	$(1-Y_{aa})f_{h2,aa}$		$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,6}$	$N_{aa}-(Y_{aa})N_{bac}$		$k_{m,aa}\frac{S_{aa}}{K_s+S_{aa}}X_{aa}I_1$
7	Uptake of LCFA			-1				$(1-Y_{fa})0.7$	$(1-Y_{fa})0.3$			$-(Y_{fa})N_{bac}$		$k_{m,fa}\frac{S_{fa}}{K_s+S_{fa}}X_{fa}I_2$
8	Uptake of valerate				-1		$(1-Y_{c4})0.54$	$(1-Y_{c4})0.31$	$(1-Y_{c4})0.15$			$-(Y_{c4})N_{bac}$		$k_{m,c4}\frac{S_{va}}{K_s+S_{va}}X_{c4}\frac{1}{1+S_{bu}/S_{va}}I$
9	Uptake of butyrate					-1		$(1-Y_{c4})0.8$	$(1-Y_{c4})0.2$			$-(Y_{c4})N_{bac}$		$k_{m,c4}\frac{S_{bu}}{K_s+S_{bu}}X_{c4}\frac{1}{1+S_{va}/S_{ba}}I$
10	Uptake of propionate						-1	$(1-Y_{pro})0.57$	$(1-Y_{pro})0.43$		$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,10}$	$-(Y_{pro})N_{bac}$		$k_{m,pr}\frac{S_{pro}}{K_s+S_{pro}}X_{pro}I_2$
11	Uptake of acetate							-1		$(1-Y_{ac})$	$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,11}$	$-(Y_{ac})N_{bac}$		$k_{m,ac}\frac{S_{ac}}{K_s+S_{ac}}X_{ac}I_3$
12	Uptake of hydrogen								-1	$(1-Y_{h2})$	$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,12}$	$-(Y_{h2})N_{bac}$		$k_{m,h2}\frac{S_{h2}}{K_s+S_{h2}}X_{h2}I_1$
13	Decay of X_{su}													$k_{dec,Xsu}X_{su}$
14	Decay of X_{aa}													$k_{dec,Xaa}X_{aa}$
15	Decay of X_{fa}													$k_{dec,Xfa}X_{fa}$
16	Decay of X_{c4}													$k_{dec,Xc4}X_{c4}$
17	Decay of X_{pro}													$k_{dec,Xpro}X_{pro}$
18	Decay of X_{ac}													$k_{dec,Xac}X_{ac}$
19	Decay of X_{h2}													$k_{dec,Xh2}X_{h2}$
20	Uptake of lactate (fer.)						$(1-Y_{lac,f})0.785$	$(1-Y_{lac,f})0.215$			$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,20}$	$-(Y_{lac,f})N_{bac}$		$k_{m,lacf}\frac{S_{lac}}{K_s+S_{lac}}X_{lacf}I$
21	Uptake of lactate (ox.)							$(1-Y_{lac,o})\frac{2}{3}$	$(1-Y_{lac,o})\frac{1}{3}$		$-\sum_{\substack{i=1-9,\\11-29}} C_i\nu_{i,21}$	$-(Y_{lac,o})N_{bac}$		$k_{m,laco}\frac{S_{lac}}{K_s+S_{lac}}X_{laco}I_2$
22	Decay of $X_{lac,f}$													$k_{dec,Xlacf}X_{lacf}$
23	Decay of $X_{lac,o}$													$k_{dec,Xlaco}X_{laco}$
24	Precipitation of Calcium													$k_{r,CaCo3}\left(\sqrt{S_{ca}\cdot S_{hco3}}-\sqrt{KspCaCo_3}\right)^2$
		Monosaccharides (kgCOD·m^{-3})	Amino acids (kgCOD·m^{-3})	Long chain fatty acids (kgCOD·m^{-3})	Total valerate (kgCOD·m^{-3})	Total butyrate (kgCOD·m^{-3})	Total propionate (kgCOD·m^{-3})	Total acetate (kgCOD·m^{-3})	Hydrogen gas (kgCOD·m^{-3})	Methane gas (kgCOD·m^{-3})	Inorganic carbon (kmoleC·m^{-3})	Inorganic nitrogen (kmoleN·m^{-3})	Soluble inerts (kgCOD·m^{-3})	Inhibition factors: $I_1=I_{pH}I_{IN,lim}$ $I_1=I_{pH}I_{IN,lim}I_{h2}$ $I_1=I_{pH}I_{IN,lim}I_{NH3,Xac}$

Table A.2: Biochemical rate coefficient ($\nu_{i,j}$) and kinetic rate equations (ρ_j) for soluble components. The matrix comes from the task report [3] with complementing equations from Willquist *et al.*[5]

Component→	**i**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**36**	**37**	**38**	**39**	**Rate (ρ_j kgCODmm$^{-3}\cdot$d^{-1})**
j	**Process↓**	$\mathbf{X_c}$	$\mathbf{X_{ch}}$	$\mathbf{X_{pr}}$	$\mathbf{X_{li}}$	$\mathbf{X_{su}}$	$\mathbf{X_{aa}}$	$\mathbf{X_{fa}}$	$\mathbf{X_{c4}}$	$\mathbf{X_{pro}}$	$\mathbf{X_{ac}}$	$\mathbf{X_{h2}}$	$\mathbf{X_I}$	$\mathbf{S_{lac}}$	$\mathbf{X_{lac,f}}$	$\mathbf{X_{lac,o}}$	$\mathbf{S_{ca}}$	
1	Disintegration	-1	$f_{ch,xc}$	$f_{pr,xc}$	$f_{li,xc}$								$f_{xI,xc}$					$k_{dis}X_c$
2	Hydrolysis of carbohydrates		-1															$k_{hyd,ch}X_{ch}$
3	Hydrolysis of proteins			-1														$k_{hyd,pr}X_{pr}$
4	Hydrolysis of lipids				-1													$k_{hyd,li}X_{li}$
5	Uptake of sugar					Y_{su}												$k_{m,su}\frac{S_{su}}{K_s+S_{su}}X_{su}I_1$
6	Uptake of amino acids						Y_{aa}											$k_{m,aa}\frac{S_{aa}}{K_s+S_{aa}}X_{aa}I_1$
7	Uptake of LCFA							Y_{fa}										$k_{m,fa}\frac{S_{fa}}{K_s+S_{fa}}X_{fa}I_2$
8	Uptake of valerate								Y_{c4}									$k_{m,c4}\frac{S_{va}}{K_s+S_{va}}X_{c4}\frac{1}{1+S_{bu}/S_{va}}I$
9	Uptake of butyrate								Y_{c4}									$k_{m,c4}\frac{S_{bu}}{K_s+S_{bu}}X_{c4}\frac{1}{1+S_{va}/S_{ba}}I$
10	Uptake of propionate									Y_{pro}								$k_{m,pr}\frac{S_{pro}}{K_s+S_{pro}}X_{pro}I_2$
11	Uptake of acetate										Y_{ac}							$k_{m,ac}\frac{S_{ac}}{K_s+S_{ac}}X_{ac}I_3$
12	Uptake of hydrogen											Y_{h2}						$k_{m,h2}\frac{S_{h2}}{K_s+S_{h2}}X_{h2}I_1$
13	Decay of X_{su}	1				-1												$k_{dec,Xsu}X_{su}$
14	Decay of X_{aa}	1					-1											$k_{dec,Xaa}X_{aa}$
15	Decay of X_{fa}	1						-1										$k_{dec,Xfa}X_{fa}$
16	Decay of X_{c4}	1							-1									$k_{dec,Xc4}X_{c4}$
17	Decay of X_{pro}	1								-1								$k_{dec,Xpro}X_{pro}$
18	Decay of X_{ac}	1									-1							$k_{dec,Xac}X_{ac}$
19	Decay of X_{h2}	1										-1						$k_{dec,Xh2}X_{h2}$
20	Uptake of lactate (fer.)													-1	Y_{lacf}			$k_{m,lacf}\frac{S_{lac}}{K_s+S_{lac}}X_{lacf}I$
21	Uptake of lactate (ox.)													-1		Y_{laco}		$k_{m,laco}\frac{S_{lac}}{K_s+S_{lac}}X_{laco}I_2$
22	Decay of $X_{lac,f}$	1													-1			$k_{dec,Xlacf}X_{lacf}$
23	Decay of $X_{lac,o}$	1														-1		$k_{dec,Xlaco}X_{laco}$
24	Precipitation of Calcium																-1	$k_{r,CaCo3}\left(\sqrt{S_{ca}\cdot S_{hco3}}-\sqrt{KspCaCo_3}\right)^2$
		Composites (kgCOD·m^{-3})	Carbohydrates (kgCOD·m^{-3})	Proteins (kgCOD·m^{-3})	Lipids (kgCOD·m^{-3})	Sugar degraders (kgCOD·m^{-3})	Amino acid degraders (kgCOD·m^{-3})	LCFA degraders (kgCOD·m^{-3})	Valerate and butyrate degraders (kgCOD·m^{-3})	Propionate degraders (kgCOD·m^{-3})	Acetate degraders (kgCOD·m^{-3})	Hydrogen degraders (kgCOD·m^{-3})	Particulate inerts (kgCOD·m^{-3})	Total lactate (kgCOD·m^{-3})	Lactate (fermentation) degraders (kgCOD·m^{-3})	Lactate (oxidation) degraders (kgCOD·m^{-3})	Total calcium (kmole·m^{-3})	Inhibition factors: $I_1=I_{pH}I_{IN,lim}$ $I_1=I_{pH}I_{IN,lim}I_{h2}$ $I_1=I_{pH}I_{IN,lim}I_{NH3,Xac}$

B

Model parameters

The model consist out of a lot of parameters. Those parameters are defined in the following tables. The values in the tables comes from Rosén and Jeppsons report [4] about how to implement the ADM1 model. If nothing else is said in the report those parameter values are used for all parameters in this work. Those tables have the numbers B.1 to B.4.

Besides the parameters for the standard ADM1 there are also parameters for the new substrates added. Those parameter values comes from the master thesis report by Samuel Scherman Johansson[9], those values can be seen in table B.5 to B.7.

B.1 Standard ADM1 parameters

Table B.1: Stoichiometric parameter values (standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
$f_{sI,xc}$	0.1	-
$f_{xI,xc}$	0.2	-
$f_{ch,xc}$	0.2	-
$f_{pr,xc}$	0.2	-
$f_{li,xc}$	0.2	-
N_{xc}	0.0376/14	kmole N(kg COD)$^{-1}$
N_I	0.06/14	kmole N(kg COD)$^{-1}$
N_{aa}	0.007	kmole N(kg COD)$^{-1}$
C_{xc}	0.02786	kmole C(kg COD)$^{-1}$
C_{sI}	0.03	kmole C(kg COD)$^{-1}$
C_{ch}	0.0313	kmole C(kg COD)$^{-1}$
C_{pr}	0.03	kmole C(kg COD)$^{-1}$
C_{li}	0.022	kmole C(kg COD)$^{-1}$
C_{xI}	0.03	kmole C(kg COD)$^{-1}$
C_{su}	0.0313	kmole C(kg COD)$^{-1}$
C_{aa}	0.03	kmole C(kg COD)$^{-1}$
$f_{fa,li}$	0.95	-
C_{fa}	0.0217	kmole C(kg COD)$^{-1}$
$f_{h2,su}$	0.19	-
$f_{bu,su}$	0.13	-
$f_{pro,su}$	0.27	-
$f_{ac,su}$	0.41	-
N_{bac}	0.08/14	kmole N(kg COD)$^{-1}$
C_{bu}	0.025	kmole C(kg COD)$^{-1}$
C_{pro}	0.0268	kmole C(kg COD)$^{-1}$
C_{ac}	0.0313	kmole C(kg COD)$^{-1}$
C_{bac}	0.0313	kmole C(kg COD)$^{-1}$
Y_{su}	0.1	-
$f_{h2,aa}$	0.06	-
$f_{va,aa}$	0.23	-
$f_{bu,aa}$	0.26	-
$f_{pro,aa}$	0.05	-
$f_{ac,aa}$	0.40	-
C_{va}	0.024	kmole C(kg COD)$^{-1}$
Y_{aa}	0.08	-
Y_{fa}	0.06	-
Y_{c4}	0.06	-
Y_{pro}	0.04	-
C_{ch4}	0.0156	kmole C(kg COD)$^{-1}$
Y_{ac}	0.05	-
Y_{h2}	0.06	-

Table B.2: Biochemical parameter values (standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
k_{dis}	0.5	d^{-1}
$k_{hyd,ch}$	10	d^{-1}
$k_{hyd,pr}$	10	d^{-1}
$k_{hyd,li}$	10	d^{-1}
$K_{S,IN}$	1e-4	M
$k_{m,su}$	30	d^{-1}
$K_{S,su}$	0.5	kg COD m^{-3}
$pH_{UL,aa}$	5.5	-
$pH_{LL,aa}$	4	-
$k_{m,aa}$	50	d^{-1}
$K_{S,aa}$	0.3	kg COD m^{-3}
$k_{m,fa}$	6	d^{-1}
$K_{S,fa}$	0.3	kg COD m^{-3}
$K_{Ih2,fa}$	5e-6	kg COD m^{-3}
$k_{m,c4}$	20	d^{-1}
$K_{S,c4}$	0.2	kg COD m^{-3}
$K_{Ih2,c4}$	1e-5	kg COD m^{-3}
$k_{m,pro}$	13	d^{-1}
$K_{S,pro}$	0.1	kg COD m^{-3}
$K_{Ih2,pro}$	3.5e-6	kg COD m^{-3}
$k_{m,ac}$	8	d^{-1}
$K_{S,ac}$	0.15	kg COD m^{-3}
$K_{I,nh3}$	0.0018	kg COD m^{-3}
$pH_{UL,ac}$	7	-
$pH_{LL,ac}$	6	-
$k_{m,h2}$	35	d^{-1}
$K_{S,h2}$	7e-6	kg COD m^{-3}
$pH_{UL,h2}$	6	-
$pH_{LL,h2}$	5	-
$k_{dec,Xsu}$	0.02	d^{-1}
$k_{dec,Xaa}$	0.02	d^{-1}
$k_{dec,Xfa}$	0.02	d^{-1}
$k_{dec,Xc4}$	0.02	d^{-1}
$k_{dec,Xpro}$	0.02	d^{-1}
$k_{dec,Xac}$	0.02	d^{-1}
$k_{dec,Xh2}$	0.02	d^{-1}

Table B.3: Physiochemical parameter values (standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
R	0.083145	bar $M^{-1}K^{-1}$
T_{base}	298.15	K
T_{op}	308.15	K
K_w	$\exp\left(\frac{55900}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M 10^{-14}
$K_{a,va}$	$10^{-4.86}$	M
$K_{a,bu}$	$10^{-4.82}$	M
$K_{a,pro}$	$10^{-4.88}$	M
$K_{a,ac}$	$10^{-4.76}$	M
$K_{a,co2}$	$10^{-6.35}\exp\left(\frac{7645}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M
$K_{a,IN}$	$10^{-9.25}\exp\left(\frac{51965}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M
$k_{A,Bva}$	1e10	M^{-1} d^{-1}
$k_{A,Bbu}$	1e10	M^{-1} d^{-1}
$k_{A,Bpro}$	1e10	M^{-1} d^{-1}
$k_{A,Bac}$	1e10	M^{-1} d^{-1}
$k_{A,Bco2}$	1e10	M^{-1} d^{-1}
$k_{A,BIN}$	1e10	M^{-1} d^{-1}
P_{atm}	1.013	bar
$p_{gas,h2o}$	$0.0313\exp\left(5290\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	bar
k_p	5e4	ms^3 d^{-1} bar^{-1}
k_{La}	200	d^{-1}
$K_{H,co2}$	$0.035\exp\left(\frac{-19410}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}
$K_{H,ch4}$	$0.0014\exp\left(\frac{-14240}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}
$K_{H,h2}$	$7.8e-4\exp\left(\frac{-19410}{R\cdot 100}\left(\frac{1}{T_{base}}-\frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}

Table B.4: Physical parameter values (standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
V_{liq}	3400	m^3
V_{gas}	300	m^3

B.2 Not standard ADM1 parameters

Table B.5: Stoichiometric parameter values (not standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
C_{lac}	0.0313	kmole C(kg COD)$^{-1}$
$Y_{lac,f}$	0.055	-
$Y_{lac,o}$	0.055	-

Table B.6: Biochemical parameter values (not standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
$k_{m,lacf}$	16	d^{-1}
$K_{S,lacf}$	3.5169	kg COD m^{-3}
$k_{m,laco}$	16	d^{-1}
$K_{S,laco}$	0.6432	kg COD m^{-3}
$K_{Ih2,laco}$	1.4e-4	kg COD m^{-3}
$K_{Ih,vfa}$	3.5	kg COD m^{-3}
$k_{dec,Xlacf}$	0.02	d^{-1}
$k_{dec,Xlaco}$	0.02	d^{-1}
$K_{Sp,caco3}$	$\frac{\exp(-0.01183 \cdot T_{op} - 8.03)}{0.665^2}$	kg $CaCO_3$ m^{-3}
$K_{r,caco3}$	1477.44	d^{-1}

Table B.7: Physiochemical parameter values (not standard ADM1). The structure follows the one in Rosén and Jeppssons report [4] for maximum readability.

Parameter	Value	Unit
$K_{a,lac}$	$10^{-3.86}$	M

C

Steady state input variables

Table C.1: Continuous inflow to the model giving rise to steady state output flow. Values from [4]. New substrates are set to be zero.

State No.	Variable	Value	Unit
1	$S_{su,in}$	0.01	$kgCODm^{-3}$
2	$S_{aa,in}$	0.001	$kgCODm^{-3}$
3	$S_{fa,in}$	0.001	$kgCODm^{-3}$
4	$S_{va,in}$	0.001	$kgCODm^{-3}$
5	$S_{bu,in}$	0.001	$kgCODm^{-3}$
6	$S_{pro,in}$	0.001	$kgCODm^{-3}$
7	$S_{ac,in}$	0.001	$kgCODm^{-3}$
8	$S_{h2,in}$	1.0e-8	$kgCODm^{-3}$
9	$S_{ch4,in}$	1.0e-5	$kgCODm^{-3}$
10	$S_{IC,in}$	0.04	$kmoleCm^{-3}$
11	$S_{IN,in}$	0.01	$kmoleNm^{-3}$
12	$S_{I,in}$	0.02	$kgCODm^{-3}$
13	$X_{xc,in}$	2.0	$kgCODm^{-3}$
14	$X_{ch,in}$	5.0	$kgCODm^{-3}$
15	$X_{pr,in}$	20.0	$kgCODm^{-3}$
16	$X_{li,in}$	5.0	$kgCODm^{-3}$
17	$X_{su,in}$	0.0	$kgCODm^{-3}$
18	$X_{aa,in}$	0.01	$kgCODm^{-3}$
19	$X_{fa,in}$	0.01	$kgCODm^{-3}$
20	$X_{c4,in}$	0.01	$kgCODm^{-3}$
21	$X_{pro,in}$	0.01	$kgCODm^{-3}$
22	$X_{ac,in}$	0.01	$kgCODm^{-3}$
23	$X_{h2,in}$	0.01	$kgCODm^{-3}$
24	$X_{I,in}$	25.0	$kgCODm^{-3}$
25	$S_{cat,in}$	0.04	$kmolem^{-3}$
26	$S_{an,in}$	0.02	$kmolem^{-3}$
-	q_{in}	170.0	m^3d^{-1}
-	T_{op}	35.0	°C

D

Code

D.1 Running ADM1 simulations

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%File for running an implementation of anaerobic digester simulation with
%the ADM1 model. Theresult from the run should preferably be used in any
%way, for example plotting.
%
%Names of variables and parameters are used from Rosen and Jeppsson
%implementation in report "Aspect of ADM1 Implementaion within the BSM2
%Framework", 2006.



clear all
%close all

%retrive input data and parameter values
l = IndataADM1_v2;

%start values from Hamse's Algerian colleague
y0=[0.009;...    % S_su
    0.0009;...   % S_aa
    0.0009;...   % S_fa
```

```
    0.0009;...    % S_va
    0.0009;...    % S_bu
    0.0009;...    % S_pro
    0.0009;...    % S_ac
    2.3594e-9;...%S_h2
    2.3594e-6;...%S_ch4
    0.039;...     %S_IC
    0.13023;...   %S_IN
    0.009;...     %S_I
    0.30870;...   %X_c
    0.02795;...   %X_ch
    0.10260;...   %X_pr
    0.02948; ...  %X_li
    0.42016;...   %X_su
    1.17917;...   %X_aa
    0.24303;...   %X_fa
    0.43192;...   %X_c4
    0.13730;...   %X_pro
    0.76056;...   %X_ac
    0.31702;...   %X_h2
    25.61739;...  %X_I
    0.04;...      %S_cat
    0.02;...      %S_an
    0.0116;...    %S_vam
    0.01322;...   %S_bum
    0.01574;...   %S_prom
    0.19724;...   %S_acm
    0.14278;...   %S_hco3m
    0.00409;...   %S_nh3
    1.023e-5;...  %h2
    1.62125;...   %ch4
    0.01411;...   %co2
    0;...         %S_lac
    0;...         %X_lac_f
    0;...         %X_lac_o
    0];           %S_ca


%Solve the ODE-system
tic
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);
[T,solution] = ode15s(@(t,y) ADM1_fun_v2_ODE(t,y,l), [0 50],y0, options);
toc %print out the time to solve the ODE system

%retrive and store solution in result struct * r *
r.S_su      = solution(:,1);
r.S_aa      = solution(:,2);
r.S_fa      = solution(:,3);
r.S_va      = solution(:,4);
r.S_bu      = solution(:,5);
r.S_pro     = solution(:,6);
r.S_ac      = solution(:,7);
r.S_h2      = solution(:,8);
```

```
r.S_ch4     = solution(:,9);
r.S_IC      = solution(:,10);
r.S_IN      = solution(:,11);
r.S_I       = solution(:,12);
r.X_c       = solution(:,13);
r.X_ch      = solution(:,14);
r.X_pr      = solution(:,15);
r.X_li      = solution(:,16);
r.X_su      = solution(:,17);
r.X_aa      = solution(:,18);
r.X_fa      = solution(:,19);
r.X_c4      = solution(:,20);
r.X_pro     = solution(:,21);
r.X_ac      = solution(:,22);
r.X_h2      = solution(:,23);
r.X_I       = solution(:,24);
r.S_cat     = solution(:,25);
r.S_an      = solution(:,26);
r.S_vam     = solution(:,27);
r.S_bum     = solution(:,28);
r.S_prom    = solution(:,29);
r.S_acm     = solution(:,30);
r.S_hco3m   = solution(:,31);
r.S_nh3     = solution(:,32);
r.S_gas_h2  = solution(:,33);
r.S_gas_ch4 = solution(:,34);
r.S_gas_co2 = solution(:,35);
r.S_lac     = solution(:,36);
r.X_lac_f   = solution(:,37);
r.X_lac_o   = solution(:,38);
r.S_ca      = solution(:,39);

r.T = T; %save simulation time

%gas pressure
r.P_gas_h2  = r.S_gas_h2*l.R*l.T_op/16;
r.P_gas_ch4 = r.S_gas_ch4*l.R*l.T_op/64;
r.P_gas_co2 = r.S_gas_co2*l.R*l.T_op;

%gas flow
r.P_gas     = r.P_gas_h2+r.P_gas_ch4+r.P_gas_co2+l.p_gas_h2o; %total gas pressure
r.q_gas     = l.k_p*(P_gas-l.P_atm).*P_gas/l.P_atm;           %total gas flow

%calculate gas component gas flow
r.q_gas_h2  = r.P_gas_h2./r.P_gas.*r.q_gas;
r.q_gas_ch4 = r.P_gas_ch4./r.P_gas.*r.q_gas;
r.q_gas_co2 = r.P_gas_co2./r.P_gas.*r.q_gas;
r.q_gas_h2o = l.p_gas_h2o./r.P_gas.*r.q_gas;
```

D.2 ADM1 function file

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%Function to be used in an ODE solver in MATLAB. The file contain all
%calculatation to be done in an ODE implementation of ADM1
%
%Names of variables and parameters are used from Rosen and Jeppsson
%implementation in report "Aspect of ADM1 Implementaion within the BSM2
%Framework", 2006.
%
%Equations do allso follows the implementation done by Rosen and Jeppsson.
%There are som small differences done to the standrad ADM1 implementation
%due to some errors in carbon and nitrogen balances in standard
%implementation. There is also some other small changes, but the main
%implementation are equal to the standard one.


function ADM1_dt = ADM1_fun_v2_ODE(t,y,l)

S_su  = y(1);
S_aa  = y(2);
S_fa  = y(3);
S_va  = y(4);
S_bu  = y(5);
S_pro = y(6);
S_ac  = y(7);
S_h2  = y(8);
S_ch4 = y(9);
```

```
S_IC  = y(10);
S_IN  = y(11);
S_I   = y(12);
X_c   = y(13);
X_ch  = y(14);
X_pr  = y(15);
X_li  = y(16);
X_su  = y(17);
X_aa  = y(18);
X_fa  = y(19);
X_c4  = y(20);
X_pro = y(21);
X_ac  = y(22);
X_h2  = y(23);
X_I   = y(24);
S_cat = y(25);
S_an  = y(26);
S_vam = y(27);
S_bum = y(28);
S_prom= y(29);
S_acm = y(30);
S_hco3m=y(31);
S_nh3 = y(32);
S_gas_h2    = y(33);
S_gas_ch4   = y(34);
S_gas_co2   = y(35);
%................................................
%Extra input to ADM1 model (added more substartes)
%................................................
S_lac       = y(36);
X_lac_f     = y(37);
X_lac_o     = y(38);
S_ca        = y(39);

%===============================================================================
%calculation of gas preassure
%===============================================================================
P_gas_h2  = S_gas_h2*l.R*l.T_op/16;
```

```
P_gas_ch4 = S_gas_ch4*l.R*l.T_op/64;
P_gas_co2 = S_gas_co2*l.R*l.T_op;
P_gas     = P_gas_h2+P_gas_ch4+P_gas_co2+l.p_gas_h2o;
q_gas     = l.k_p*(P_gas-l.P_atm)*P_gas/l.P_atm;

%=========================================================================
%calculation of potential Sh
%=========================================================================
S_nh4 = S_IN-S_nh3;
S_co2 = S_IC-S_hco3m;

Theta = S_cat + S_nh4 - S_hco3m - (S_acm/64) - (S_prom/112) - (S_bum/160)...
    - (S_vam/208) - S_an + S_ca; %added S_ca to orginal equation
Sh = -Theta/2 + sqrt(Theta^2 + 4*l.K_w)/2;

if(Sh <= 0)
    Sh = 1e-12;
end
pH = -log10(Sh);

%pH inhibition
if(pH < l.pH_UL_aa)
    I_pH_aa = exp(-3*((pH-l.pH_UL_aa)/(l.pH_UL_aa-l.pH_LL_aa))^2);
else
    I_pH_aa = 1;
end
if(pH < l.pH_UL_ac)
    I_pH_ac = exp(-3*((pH-l.pH_UL_ac)/(l.pH_UL_ac-l.pH_LL_ac))^2);
else
    I_pH_ac = 1;
end
if(pH < l.pH_UL_h2)
    I_pH_h2 = exp(-3*((pH-l.pH_UL_h2)/(l.pH_UL_h2-l.pH_LL_h2))^2);
else
    I_pH_h2 = 1;
end
```

```
%==========================================================================
%Process inhibition
%==========================================================================
I_IN_lim = 1/(1+l.K_S_IN/S_IN);
I_h2_fa  = 1/(1+S_h2/l.K_I_h2_fa);
I_h2_c4  = 1/(1+S_h2/l.K_I_h2_c4);
I_h2_pro = 1/(1+S_h2/l.K_I_h2_pro);
I_nh3    = 1/(1+S_nh3/l.K_I_nh3);
I_5      = I_pH_aa*I_IN_lim;
I_6      = I_5;
I_7      = I_pH_aa*I_IN_lim*I_h2_fa;
I_8      = I_pH_aa*I_IN_lim*I_h2_c4;
I_9      = I_8;
I_10     = I_pH_aa*I_IN_lim*I_h2_pro;
I_11     = I_pH_ac*I_IN_lim*I_nh3;
I_12     = I_pH_h2*I_IN_lim;
%.................................................
%Extra input to ADM1 model (added more substartes)
%.................................................
I_h2_lac_o = 1/(1+S_h2/l.K_I_h2_lac_o);
I_20       = I_IN_lim;
I_21       = I_h2_lac_o*I_IN_lim;

%==========================================================================
%Biochemical process rates
%==========================================================================
rho_1  = l.k_dis  * X_c;
rho_2  = l.k_hyd_ch * X_ch;
rho_3  = l.k_hyd_pr * X_pr;
rho_4  = l.k_hyd_li * X_li;
rho_5  = l.k_m_su * (S_su/(l.K_S_su+S_su)) * X_su*I_5;
rho_6  = l.k_m_aa * (S_aa/(l.K_S_aa+S_aa)) * X_aa*I_6;
rho_7  = l.k_m_fa * (S_fa/(l.K_S_fa+S_fa)) * X_fa*I_7;
rho_8  = l.k_m_c4 * (S_va/(l.K_S_c4+S_va)) * X_c4 * (S_va/(S_bu+S_va+1e-6))*I_8;
rho_9  = l.k_m_c4 * (S_bu/(l.K_S_c4+S_bu)) * X_c4 * (S_bu/(S_va+S_bu+1e-6))*I_9;
rho_10 = l.k_m_pro * (S_pro/(l.K_S_pro+S_pro)) * X_pro*I_10;
rho_11 = l.k_m_ac * (S_ac/(l.K_S_ac+S_ac)) * X_ac*I_11;
rho_12 = l.k_m_h2 * (S_h2/(l.K_S_h2+S_h2)) * X_h2*I_12;
```

```
rho_13 = l.k_dec_X_su * X_su;
rho_14 = l.k_dec_X_aa * X_aa;
rho_15 = l.k_dec_X_fa * X_fa;
rho_16 = l.k_dec_X_c4 * X_c4;
rho_17 = l.k_dec_X_pro* X_pro;
rho_18 = l.k_dec_X_ac * X_ac;
rho_19 = l.k_dec_X_h2 * X_h2;
%.................................................
%Extra input to ADM1 model (added more substartes)
%.................................................
rho_20 = l.k_m_lac_f * (S_lac/(l.K_S_lac_f+S_lac)) * X_lac_f*I_20;
rho_21 = l.k_m_lac_o * (S_lac/(l.K_S_lac_o+S_lac)) * X_lac_o*I_21;
rho_22 = l.k_dec_X_lac_f * X_lac_f;
rho_23 = l.k_dec_X_lac_o * X_lac_o;

%=========================================================================
%Acid-base rates
%=========================================================================
rho_A_4  = l.k_A_B_va*(S_vam*(l.K_a_va+Sh)-l.K_a_va*S_va);
rho_A_5  = l.k_A_B_bu*(S_bum*(l.K_a_bu+Sh)-l.K_a_bu*S_bu);
rho_A_6  = l.k_A_B_pro*(S_prom*(l.K_a_pro+Sh)-l.K_a_pro*S_pro);
rho_A_7  = l.k_A_B_ac*(S_acm*(l.K_a_ac+Sh)-l.K_a_ac*S_ac);
rho_A_10 = l.k_A_B_co2*(S_hco3m*(l.K_a_co2+Sh)-l.K_a_co2*S_IC);
rho_A_11 = l.k_A_B_IN*(S_nh3*(l.K_a_IN+Sh)-l.K_a_IN*S_IN);


%=========================================================================
%Gas transfer rates
%=========================================================================
rho_T_8 = l.k_L_a*(S_h2-16*l.K_H_h2*P_gas_h2);
rho_T_9 = l.k_L_a*(S_ch4-64*l.K_H_ch4*P_gas_ch4);
rho_T_10 = l.k_L_a*(S_co2-l.K_H_co2*P_gas_co2);


%=========================================================================
%Precipitation rates (this is not included in standard ADM1
%=========================================================================
%to avoid complex numbers prevent it from beeing negative
```

```
if(S_ca < 1e-6|| S_hco3m < 1e-6)
    rho_P_24 = 0;
else
    rho_P_24 = l.K_r_caco3*((S_ca^.5 * S_hco3m^.5 - l.K_S_p_caco3)^2);
end

%=========================================================================
%Processes (carbon)
%=========================================================================
s_1  = -l.C_xc + l.f_sI_xc*l.C_sI + l.f_ch_xc*l.C_ch + l.f_pr_xc*l.C_pr + l.f_li_xc*l.C_li + l.f_xI_xc*l.C_xI;
s_2  = -l.C_ch + l.C_su;
s_3  = -l.C_pr + l.C_aa;
s_4  = -l.C_li + (1-l.f_fa_li)*l.C_su + l.f_fa_li*l.C_fa;
s_5  = -l.C_su + (1-l.Y_su)*(l.f_bu_su*l.C_bu + l.f_pro_su*l.C_pro + l.f_ac_su*l.C_ac) + l.Y_su*l.C_bac;
s_6  = -l.C_aa + (1-l.Y_aa)*(l.f_va_aa*l.C_va + l.f_bu_aa*l.C_bu + l.f_pro_aa*l.C_pro + l.f_ac_aa*l.C_ac) + ...
    l.Y_aa*l.C_bac;
s_7  = -l.C_fa + (1-l.Y_fa)*0.7*l.C_ac + l.Y_fa*l.C_bac;
s_8  = -l.C_va + (1-l.Y_c4)*0.54*l.C_pro + (1-l.Y_c4)*0.31*l.C_ac + l.Y_c4*l.C_bac;
s_9  = -l.C_bu + (1-l.Y_c4)*0.8*l.C_ac + l.Y_c4*l.C_bac;
s_10 = -l.C_pro + (1-l.Y_pro)*0.57*l.C_ac + l.Y_pro*l.C_bac;
s_11 = -l.C_ac + (1-l.Y_ac)*l.C_ch4 + l.Y_ac*l.C_bac;
s_12 = (1-l.Y_h2)*l.C_ch4 + l.Y_h2*l.C_bac;
s_13 = -l.C_bac + l.C_xc;
%.................................................
%Extra input to ADM1 model (added more substartes)
%.................................................
s_20 = -l.C_lac + (1-l.Y_lac_f)*0.785*l.C_pro + (1-l.Y_lac_f)*0.215*l.C_ac + l.Y_lac_f*l.C_bac;
s_21 = -l.C_lac + (1-l.Y_lac_o)*2/3*l.C_ac  + l.Y_lac_o*l.C_bac;



%=========================================================================
%Differential equations
%=========================================================================
%.........................................................................
%Differential equations 1-4, soluble matter
%.........................................................................          State No.
d_S_su_dt  = l.q_in/l.V_liq*(l.S_su_in-S_su) + rho_2 + (1-l.f_fa_li)*rho_4 - rho_5;   % 1
```

```
d_S_aa_dt  = l.q_in/l.V_liq*(l.S_aa_in-S_aa) + rho_3 - rho_6;                          % 2
d_S_fa_dt  = l.q_in/l.V_liq*(l.S_fa_in-S_fa) + l.f_fa_li*rho_4 - rho_7;               % 3
d_S_va_dt  = l.q_in/l.V_liq*(l.S_va_in-S_va) + (1-l.Y_aa)*l.f_va_aa*rho_6 - rho_8;    % 4
%.....................................................................
%Differential equations 5-8, soluble matter
%.....................................................................          State No.
d_S_bu_dt  = l.q_in/l.V_liq*(l.S_bu_in-S_bu) + (1-l.Y_su)*l.f_bu_su*rho_5 + ...      % 5
             (1-l.Y_aa)*l.f_bu_aa*rho_6 - rho_9;                                   % -
d_S_pro_dt = l.q_in/l.V_liq*(l.S_pro_in-S_pro) + (1-l.Y_su)*l.f_pro_su*rho_5 + ...   % 6   (added lac)
             (1-l.Y_aa)*l.f_pro_aa*rho_6 + (1-l.Y_c4)*0.54*rho_8 - rho_10 + ...    % -
             (1-l.Y_lac_f)*0.785*rho_20;                                           % -
d_S_ac_dt  = l.q_in/l.V_liq*(l.S_ac_in-S_ac) + (1-l.Y_su)*l.f_ac_su*rho_5 + ...      % 7   (added lac)
             (1-l.Y_aa)*l.f_ac_aa*rho_6 + (1-l.Y_fa)*0.7*rho_7 + ...               % -
             (1-l.Y_c4)*0.31*rho_8 + (1-l.Y_c4)*0.8*rho_9 + ...                    % -
             (1-l.Y_pro)*0.57*rho_10 - rho_11 + (1-l.Y_lac_f)*0.215*rho_20 + ...   % -
             (1-l.Y_lac_o)*2/3*rho_21;                                             % -
d_S_h2_dt  = l.q_in/l.V_liq*(l.S_h2_in-S_h2) + (1-l.Y_su)*l.f_h2_su*rho_5 + ...      % 8   (added lac)
             (1-l.Y_aa)*l.f_h2_aa*rho_6 + (1-l.Y_fa)*0.3*rho_7 + ...               % -
             (1-l.Y_c4)*0.15*rho_8 + (1-l.Y_c4)*0.2*rho_9 + ...                    % -
             (1-l.Y_pro)*0.43*rho_10 - rho_12 - rho_T_8 + (1-l.Y_lac_o)*1/3*rho_21;% -
%.....................................................................
%Differential equations 9-12, soluble matter (added IC and IN-terms for new prep.)
%.....................................................................          State No.
d_S_ch4_dt = l.q_in/l.V_liq*(l.S_ch4_in - S_ch4) + (1-l.Y_ac)*rho_11 + ...          % 9
             (1-l.Y_h2)*rho_12 - rho_T_9;                                          % -
d_S_IC_dt  = l.q_in/l.V_liq*(l.S_IC_in - S_IC) - (s_1*rho_1 + s_2*rho_2 + ...      % 10  (added lac)
             s_3*rho_3 + s_4*rho_4 + s_5*rho_5 + s_6*rho_6 + s_7*rho_7 + ...       % -
             s_8*rho_8 + s_9*rho_9 + s_10*rho_10 + s_11*rho_11 + s_12*rho_12 + ... % -
             s_20*rho_20 + s_21*rho_21 + s_13*(rho_13+rho_14+rho_15+rho_16...       % -
             +rho_17+rho_18+rho_19+rho_22+rho_23)) - rho_T_10;                      % -
d_S_IN_dt  = l.q_in/l.V_liq*(l.S_IN_in-S_IN) - l.Y_su*l.N_bac*rho_5 + ...           % 11  (added lac)
             (l.N_aa-l.Y_aa*l.N_bac)*rho_6 - l.Y_fa*l.N_bac*rho_7 - ...            % -
             l.Y_c4*l.N_bac*rho_8 - l.Y_c4*l.N_bac*rho_9 - ...                     % -
             l.Y_pro*l.N_bac*rho_10 - l.Y_ac*l.N_bac*rho_11 - ...                  % -
             l.Y_h2*l.N_bac*rho_12 - l.Y_lac_f*l.N_bac*rho_20 - ...                % -
             l.Y_lac_o*l.N_bac*rho_21 + (l.N_bac-l.N_xc)*...                       % -
                 (rho_13+rho_14+rho_15+rho_16+rho_17+rho_18+...                    % -
                  rho_19+rho_22+rho_23) + ...                                      % -
```

```
                (l.N_xc-l.f_xI_xc*l.N_I-l.f_sI_xc*l.N_I-l.f_pr_xc*l.N_aa)*rho_1;      % -
d_S_I_dt    = l.q_in/l.V_liq*(l.S_I_in-S_I) + l.f_sI_xc*rho_1;                        % 12



%.....................................................................
%Differential equations 13-16, particulate matter
%.....................................................................          State No.
d_X_c_dt    = l.q_in/l.V_liq*(l.X_xc_in-X_c) - rho_1 + ...                         % 13  (added lac)
                rho_13+rho_14+rho_15+rho_16+rho_17+rho_18+rho_19+rho_22+rho_23;     % -
d_X_ch_dt   = l.q_in/l.V_liq*(l.X_ch_in-X_ch) + l.f_ch_xc*rho_1 - rho_2;           % 14
d_X_pr_dt   = l.q_in/l.V_liq*(l.X_pr_in-X_pr) + l.f_pr_xc*rho_1 - rho_3;           % 15
d_X_li_dt   = l.q_in/l.V_liq*(l.X_li_in-X_li) + l.f_li_xc*rho_1 - rho_4;           % 16
%.....................................................................
%Differential equations 17-20, particulate matter
%.....................................................................          State No.
d_X_su_dt   = l.q_in/l.V_liq*(l.X_su_in-X_su) + l.Y_su*rho_5 - rho_13;             % 17
d_X_aa_dt   = l.q_in/l.V_liq*(l.X_aa_in-X_aa) + l.Y_aa*rho_6 - rho_14;             % 18
d_X_fa_dt   = l.q_in/l.V_liq*(l.X_fa_in-X_fa) + l.Y_fa*rho_7 - rho_15;             % 19
d_X_c4_dt   = l.q_in/l.V_liq*(l.X_c4_in-X_c4) + l.Y_c4*rho_8 + l.Y_c4*rho_9 - rho_16;  % 20
%.....................................................................
%Differential equations 21-24, particulate matter
%.....................................................................          State No.
d_X_pro_dt = l.q_in/l.V_liq*(l.X_pro_in-X_pro) + l.Y_pro*rho_10 - rho_17;           % 21
d_X_ac_dt   = l.q_in/l.V_liq*(l.X_ac_in-X_ac) + l.Y_ac*rho_11 - rho_18;            % 22
d_X_h2_dt   = l.q_in/l.V_liq*(l.X_h2_in-X_h2) + l.Y_h2*rho_12 - rho_19;            % 23
d_X_I_dt    = l.q_in/l.V_liq*(l.X_I_in-X_I) + l.f_xI_xc*rho_1;                     % 24
%.....................................................................
%Differential equations 25-26, cations and anions
%.....................................................................          State No.
d_S_cat_dt = l.q_in/l.V_liq*(l.S_cat_in-S_cat);                                    % 25
d_S_an_dt   = l.q_in/l.V_liq*(l.S_an_in-S_an);                                     % 26
%.....................................................................
%Differential equations 27-32, ion states
%.....................................................................          State No.
d_S_vam_dt = -rho_A_4;                                                            % 27
d_S_bum_dt = -rho_A_5;                                                            % 28
d_S_prom_dt= -rho_A_6;                                                            % 29
```

```
d_S_acm_dt = -rho_A_7;                                                          % 30
d_S_hco3m_dt=-rho_A_10;                                                         % 31
d_S_nh3_dt = -rho_A_11;                                                         % 32
%.....................................................................
%Differential equations 33-25, gas phase equations
%.....................................................................          State No.
d_S_gas_h2_dt = -S_gas_h2*q_gas/l.V_gas + rho_T_8*l.V_liq/l.V_gas;               % 33
d_S_gas_ch4_dt= -S_gas_ch4*q_gas/l.V_gas + rho_T_9*l.V_liq/l.V_gas;             % 34
d_S_gas_co2_dt= -S_gas_co2*q_gas/l.V_gas + rho_T_10*l.V_liq/l.V_gas;            % 35
%.....................................................................
%Differential equations 36-38, new equations
%.....................................................................          State No.
d_S_lac_dt   = l.q_in/l.V_liq*(l.S_lac_in-S_lac) - rho_20 - rho_21;              % 36
d_X_lac_f_dt = l.q_in/l.V_liq*(l.X_lac_f_in-X_lac_f) + l.Y_lac_f*rho_20 - rho_22;% 37
d_X_lac_o_dt = l.q_in/l.V_liq*(l.X_lac_o_in-X_lac_o) + l.Y_lac_o*rho_21 - rho_23;% 38
d_S_ca_dt    = l.q_in/l.V_liq*(l.S_ca_in-S_ca) - rho_P_24;                      % 39


%output vector from function
ADM1_dt = [d_S_su_dt; d_S_aa_dt; d_S_fa_dt; d_S_va_dt; d_S_bu_dt; ...
    d_S_pro_dt; d_S_ac_dt; d_S_h2_dt; d_S_ch4_dt; d_S_IC_dt; d_S_IN_dt; ...
    d_S_I_dt; d_X_c_dt; d_X_ch_dt; d_X_pr_dt; d_X_li_dt; d_X_su_dt; ...
    d_X_aa_dt; d_X_fa_dt; d_X_c4_dt; d_X_pro_dt; d_X_ac_dt; d_X_h2_dt; ...
    d_X_I_dt; d_S_cat_dt; d_S_an_dt; d_S_vam_dt; d_S_bum_dt; d_S_prom_dt; ...
    d_S_acm_dt; d_S_hco3m_dt; d_S_nh3_dt; d_S_gas_h2_dt; d_S_gas_ch4_dt; ...
    d_S_gas_co2_dt; d_S_lac_dt; d_X_lac_f_dt; d_X_lac_o_dt; d_S_ca_dt];
end
```

D.3 ADM1 indata file

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%File for writing indata to ODE implementation of ADM1. The file saves all
%valuable information in the struct *l*, which later can be used through
%the rest of the implementation.
%
%Names of variables and parameters are used from Rosen and Jeppsson
%implementation in report "Aspect of ADM1 Implementaion within the BSM2
%Framework", 2006.
%
%Values of variables do allso comes from the report by Rosen and Jeposson,
%unless stated otherwise.


function [l]=IndataADM1_v2

%==========================================================================
%Stoichiometric parameter values
%==========================================================================
%Parameter    Value        Unit
%.........................................
l.f_sI_xc    = 0.1;       % -
l.f_xI_xc    = 0.2;       % -
l.f_ch_xc    = 0.2;       % -
l.f_pr_xc    = 0.2;       % -
l.f_li_xc    = 0.3;       % -
l.N_xc       = 0.0376/14; %kmole N (kg COD)^-1
l.N_I        = 0.06/14;   %kmole N (kg COD)^-1
l.N_aa       = 0.007;     %kmole N (kg COD)^-1
l.C_xc       = 0.02786;   %kmole C (kg COD)^-1
l.C_sI       = 0.03;      %kmole C (kg COD)^-1
l.C_ch       = 0.0313;    %kmole C (kg COD)^-1
l.C_pr       = 0.03;      %kmole C (kg COD)^-1
l.C_li       = 0.022;     %kmole C (kg COD)^-1
l.C_xI       = 0.03;      %kmole C (kg COD)^-1
%.........................................
l.C_su       = 0.0313;    %kmole C (kg COD)^-1
%.........................................
l.C_aa       = 0.03;      %kmole C (kg COD)^-1
%.........................................
l.f_fa_li    = 0.95;      % -
l.C_fa       = 0.0217;    %kmole C (kg COD)^-1
```

```
%.....................................
l.f_h2_su   = 0.19;      % -
l.f_bu_su   = 0.13;      % -
l.f_pro_su  = 0.27;      % -
l.f_ac_su   = 0.41;      % -
l.N_bac     = 0.08/14;   %kmole N (kg COD)^-1
l.C_bu      = 0.025;     %kmole C (kg COD)^-1
l.C_pro     = 0.0268;    %kmole C (kg COD)^-1
l.C_ac      = 0.0313;    %kmole C (kg COD)^-1
l.C_bac     = 0.0313;    %kmole C (kg COD)^-1
l.Y_su      = 0.1;       % -
%.....................................
l.f_h2_aa   = 0.06;      % -
l.f_va_aa   = 0.23;      % -
l.f_bu_aa   = 0.26;      % -
l.f_pro_aa  = 0.05;      % -
l.f_ac_aa   = 0.40;      % -
l.C_va      = 0.024;     %kmole C (kg COD)^-1
l.Y_aa      = 0.08;      % -
%.....................................
l.Y_fa      = 0.06;      % -
%.....................................
l.Y_c4      = 0.06;      % -
%.....................................
l.Y_pro     = 0.04;      % -
%.....................................
l.C_ch4     = 0.0156;    %kmole C (kg COD)^-1
l.Y_ac      = 0.05;      % -
%.....................................
l.Y_h2      = 0.06;      % -
%.....................................
%Extra input to ADM1 model (added more substartes)
%.....................................
l.C_lac     = 0.0313;    %kmole C (kg COD)^-1
l.Y_lac_f   = 0.055;     % -
l.Y_lac_o   = 0.055;     % -
%.....................................


%=========================================================================
%Physicochemical parameter values
%=========================================================================
%Parameter    Value                                                 Unit
%.........................................................................
l.R         = 0.083145;                                            %bar M^-1 K^-1
l.T_base    = 298.15;                                              %K
l.T_op      = 308.15;                                              %K
%.........................................................................
l.K_w       = 1e-14*exp(55900/(l.R*100)*(1/l.T_base - 1/l.T_op));  %M
l.K_a_va    = 10^-4.86;                                            %M
l.K_a_bu    = 10^-4.82;                                            %M
l.K_a_pro   = 10^-4.88;                                            %M
l.K_a_ac    = 10^-4.76;                                            %M
```

```
l.K_a_co2   = 10^-6.35*exp(7646/(l.R*100)*(1/l.T_base - 1/l.T_op)); %M
l.K_a_IN    = 10^-9.25*exp(51965/(l.R*100)*(1/l.T_base - 1/l.T_op));%M
%.....................................................................
l.k_A_B_va  = 1e10;                                                 %M^-1 d^-1
l.k_A_B_bu  = 1e10;                                                 %M^-1 d^-1
l.k_A_B_pro = 1e10;                                                 %M^-1 d^-1
l.k_A_B_ac  = 1e10;                                                 %M^-1 d^-1
l.k_A_B_co2 = 1e10;                                                 %M^-1 d^-1
l.k_A_B_IN  = 1e10;                                                 %M^-1 d^-1
%.....................................................................
l.P_atm     = 1.013;                                                %bar
l.p_gas_h2o = 0.0313*exp(5290*(1/l.T_base - 1/l.T_op));             %bar
l.k_p       = 5e4;                                                  %m^3 d^-1 bar^-1
%.....................................................................
l.k_L_a     = 200;                                                  %d^-1
%.....................................................................
l.K_H_co2   = 0.035*exp(-19410/(l.R*100)*(1/l.T_base - 1/l.T_op));  %M_liq bar^-1
l.K_H_ch4   = 0.0014*exp(-14240/(l.R*100)*(1/l.T_base - 1/l.T_op)); %M_liq bar^-1
l.K_H_h2    = 7.8e-4*exp(-4180/(l.R*100)*(1/l.T_base - 1/l.T_op));  %M_liq bar^-1
%.....................................................................
%Extra input to ADM1 model (added more substartes)
%.....................................................................
l.K_a_lac   = 10^-3.86;
%.....................................................................

%======================================================================
%Physical parameter values
%======================================================================
%Parameter    Value   Unit
%.....................................
l.V_liq     = 3400;   %m^3
l.V_gas     = 300;    %m^3
%.....................................


%======================================================================
%Biochemical parameter values          Comment
%======================================================================
%Parameter     Value    Unit
%.....................................
l.k_dis      = 0.5;     %d^-1
l.k_hyd_ch   = 10;      %d^-1
l.k_hyd_pr   = 10;      %d^-1
l.k_hyd_li   = 10;      %d^-1
%.....................................
l.K_S_IN     = 1e-4;    %M
%.....................................
l.k_m_su     = 30;      %d^-1
l.K_S_su     = 0.5;     %kg COD m^-3
l.pH_UL_aa   = 5.5;     % -
l.pH_LL_aa   = 4;       % -
%.....................................
l.k_m_aa     = 50;      %d^-1
```

```
l.K_S_aa      = 0.3;      %kg COD m^-3
%.....................................
l.k_m_fa      = 6;        %d^-1
l.K_S_fa      = 0.4;      %kg COD m^-3
l.K_I_h2_fa   = 5e-6;     %kg COD m^-3
%.....................................
l.k_m_c4      = 20;       %d^-1
l.K_S_c4      = 0.2;      %kg COD m^-3
l.K_I_h2_c4   = 1e-5;     %kg COD m^-3
%.....................................
l.k_m_pro     = 13;       %d^-1
l.K_S_pro     = 0.1;      %kg COD m^-3
l.K_I_h2_pro = 3.5e-6;    %kg COD m^-3
%.....................................
l.k_m_ac      = 8;        %d^-1
l.K_S_ac      = 0.15;     %kg COD m^-3
l.K_I_nh3     = 0.0018;   %M
l.pH_UL_ac    = 7;        % -
l.pH_LL_ac    = 6;        % -
%.....................................
l.k_m_h2      = 35;       %d^-1
l.K_S_h2      = 7e-6;     %kg COD m^-3
l.pH_UL_h2    = 6;        % -
l.pH_LL_h2    = 5;        % -
%.....................................
l.k_dec_X_su = 0.02;     %d^-1
l.k_dec_X_aa = 0.02;     %d^-1
l.k_dec_X_fa = 0.02;     %d^-1
l.k_dec_X_c4 = 0.02;     %d^-1
l.k_dec_X_pro= 0.02;     %d^-1
l.k_dec_X_ac = 0.02;     %d^-1
l.k_dec_X_h2 = 0.02;     %d^-1
%.....................................
%Extra input to ADM1 model (added more substartes)
%.....................................
l.k_m_lac_f      = 16;       %d^-1
l.K_S_lac_f      = 3.5169;   %kg COD m^-3
l.k_m_lac_o      = 16;       %d^-1
l.K_S_lac_o      = 0.6432;   %kg COD m^-3
l.K_I_h2_lac_o   = 1.4e-4;   %kg COD m^-3
l.K_I_vfa        = 3.5;      %kg COD m^-3
%.....................................
l.k_dec_X_lac_f = 0.02;     %d^-1
l.k_dec_X_lac_o = 0.02;     %d^-1
%.....................................
l.K_S_p_caco3     = exp(-0.01183*l.T_op-8.03)/0.665^2;    %kg CaCO3 m^-3  (equilibrium constan
l.K_r_caco3     = 1477.44; %d^-1 (specific rate precipitation)
%.....................................


%=========================================================================
%Steady-state input values
%=========================================================================
```

```
%Parameter    Value     Unit             State No.
%.............................................
l.S_su_in    = 0.01;    %kg COD m^-3     1
l.S_aa_in    = 0.001;   %kg COD m^-3     2
l.S_fa_in    = 0.001;   %kg COD m^-3     3
l.S_va_in    = 0.001;   %kg COD m^-3     4
l.S_bu_in    = 0.001;   %kg COD m^-3     5
l.S_pro_in   = 0.001;   %kg COD m^-3     6
l.S_ac_in    = 0.001;   %kg COD m^-3     7
l.S_h2_in    = 1.0e-8;  %kg COD m^-3     8
l.S_ch4_in   = 1.0e-5;  %kg COD m^-3     9
l.S_IC_in    = 0.04;    %kmole C m^-3    10
l.S_IN_in    = 0.01;    %kmole N m^-3    11
l.S_I_in     = 0.02;    %kg COD m^-3     12
%.............................................
l.X_xc_in    = 2.0;     %kg COD m^-3     13
l.X_ch_in    = 5.0;     %kg COD m^-3     14
l.X_pr_in    = 20.0;    %kg COD m^-3     15
l.X_li_in    = 5.0;     %kg COD m^-3     16
l.X_su_in    = 0.0;     %kg COD m^-3     17
l.X_aa_in    = 0.01;    %kg COD m^-3     18
l.X_fa_in    = 0.01;    %kg COD m^-3     19
l.X_c4_in    = 0.01;    %kg COD m^-3     20
l.X_pro_in   = 0.01;    %kg COD m^-3     21
l.X_ac_in    = 0.01;    %kg COD m^-3     22
l.X_h2_in    = 0.01;    %kg COD m^-3     23
l.X_I_in     = 25.0;    %kg COD m^-3     24
%.............................................
l.S_cat_in   = 0.04;    %kmole m^-3      25
l.S_an_in    = 0.02;    %kmole m^-3      26
%.............................................
l.q_in       = 170;     %m^3 d^-1        -
%l.T_op       = 35.0;   %degree C        - (defined above in Kelvin)
%.............................................
%Extra input to ADM1 model (added more substartes)
%.............................................
l.S_lac_in  = 0;%0.001;    %kg COD m^-3     36
%.............................................
l.X_lac_f_in= 0;%0.01;     %kg COD m^-3     37
l.X_lac_o_in= 0;%0.01;     %kg COD m^-3     38
%.............................................
l.S_ca_in   = 0;%0.01;     %kg CaCO3 m^-3   39
%.............................................
```

D.4 Sensitivity analysis

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%
%Names of variables and parameters are used from Rosen and Jeppsson
%implementation in report "Aspect of ADM1 Implementaion within the BSM2
%Framework", 2006.
%
%Function for testing the sensitivity (stability) of the model with respect
%to the parameters of the model


function stability_ADM1_v2()
addpath('../ADM1')
clear all
close all

%retrive input data and parameter values
l = IndataADM1_v2;

%start values from Hamse's Algerian colleague
y0=[0.009;...    % S_su
    0.0009;...   % S_aa
    0.0009;...   % S_fa
    0.0009;...   % S_va
    0.0009;...   % S_bu
    0.0009;...   % S_pro
    0.0009;...   % S_ac
    2.3594e-9;...%S_h2
    2.3594e-6;...%S_ch4
    0.039;...    %S_IC
    0.13023;...  %S_IN
    0.009;...    %S_I
    0.30870;...  %X_c
    0.02795;...  %X_ch
    0.10260;...  %X_pr
    0.02948; ... %X_li
    0.42016;...  %X_su
    1.17917;...  %X_aa
    0.24303;...  %X_fa
    0.43192;...  %X_c4
    0.13730;...  %X_pro
    0.76056;...  %X_ac
```

```
    0.31702;...  %X_h2
    25.61739;... %X_I
    0.04;...     %S_cat
    0.02;...     %S_an
    0.0116;...   %S_vam
    0.01322;...  %S_bum
    0.01574;...  %S_prom
    0.19724;...  %S_acm
    0.14278;...  %S_hco3m
    0.00409;...  %S_nh3
    1.023e-5;... %h2
    1.62125;...  %ch4
    0.01411;...  %co2
    0;...        %S_lac
    0;...        %X_lac_f
    0;...        %X_lac_o
    0];          %S_ca


n_param = 74;           %number of parameters studied
var = ones(1,n_param);  %vecor to store changes in

gas_ref = calculate_mean_gas(l,y0); %reference value for gas production
sens_vec = zeros(1,n_param);        %vector with sensibility stored
ref_vec = ref_value(l,n_param);     %reference values to be used in
                                    %definition of derivative

diff_vec = sens_vec;
%do actual calculation
for i = 1:n_param
    i                       %disp calculation round
    var(i) = 1.01;          %change value with 1%
    l = change_value(l,var);%do actual change
    gas_new = calculate_mean_gas(l,y0); %calculate new gas production
    sens_vec(i) = (gas_new-gas_ref)/(0.01*ref_vec(i));  % stor new sens.value
    diff_vec(i) = (gas_new-gas_ref);
    var(i) = 1;         %change back value
    l = IndataADM1_v2;  %reset l
end

%plot the actual bar graph
bfigure = figure;
bar(abs(sens_vec));
grid on
set(get(bfigure,'Child'), 'XTick', [10 20 30 40 50 60 70])
set(get(bfigure,'Child'), 'XTickLabel', {'10' '20' '30' '40' '50' '60' '70'})
set(get(bfigure,'Child'),'Yscale', 'log')
title('Sensitivity', 'fontSize', 30)
xlabel('parameter', 'fontSize', 25)
ylabel('$\left|\frac{f(x\cdot 1.01)-f(x)}{0.01x}\;\;\;\;\right|$',...
    'interp','latex', 'fontSize', 25)
figuresize(16,12,'centimeters')
```

```
figure
bar(abs(diff_vec))
set(get(bfigure,'Child'), 'XTick', [10 20 30 40 50 60 70])
set(get(bfigure,'Child'), 'XTickLabel', {'10' '20' '30' '40' '50' '60' '70'})
title('Sensitivity', 'fontSize', 30)
xlabel('parameter', 'fontSize', 25)
ylabel('$\left|f(x\cdot 1.01)-f(x)\right|$','interp','latex', 'fontSize', 25)
grid on
set(gca, 'Yscale', 'log')

end


%calculates mean gas production in steadty state with the parameters used
function mean_gas = calculate_mean_gas(l,y0)
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);
[T,solution] = ode15s(@(t,y) ADM1_fun_v2_ODE(t,y,l), [0 100],y0, options);


%gas concentrations
S_gas_h2  = solution(:,33);
S_gas_ch4 = solution(:,34);
S_gas_co2 = solution(:,35);

%gas ressures
P_gas_h2  = S_gas_h2*l.R*l.T_op/16;
P_gas_ch4 = S_gas_ch4*l.R*l.T_op/64;
P_gas_co2 = S_gas_co2*l.R*l.T_op;

P_gas = P_gas_h2+P_gas_ch4+P_gas_co2+l.p_gas_h2o; %total gas pressure
q_gas     = l.k_p*(P_gas-l.P_atm).*P_gas/l.P_atm; %total gas flow

q_gas_ch4 = P_gas_ch4./P_gas.*q_gas;        %gas flow of CH4
mean_gas = mean(q_gas_ch4(end-100:end));    %mean gas flow at steady state

end


%function for changing one paramter at a time with 1 percent
function [l] = change_value(l,var)
l.f_sI_xc = l.f_sI_xc*var(1);
l.f_xI_xc = l.f_xI_xc*var(2);
l.f_ch_xc = l.f_ch_xc*var(3);
l.f_pr_xc = l.f_pr_xc*var(4);
l.f_li_xc = l.f_li_xc*var(5);
l.f_h2_su = l.f_h2_su*var(6);
l.f_bu_su = l.f_bu_su*var(7);
l.f_pro_su = l.f_pro_su*var(8);
l.f_ac_su = l.f_ac_su*var(9);
l.Y_su = l.Y_su*var(10);
l.f_h2_aa = l.f_h2_aa*var(11);
l.f_va_aa = l.f_va_aa*var(12);
l.f_bu_aa = l.f_bu_aa*var(13);
```

```
l.f_pro_aa = l.f_pro_aa*var(14);
l.f_ac_aa = l.f_ac_aa*var(15);
l.Y_aa = l.Y_aa*var(16);
l.Y_fa = l.Y_fa*var(17);
l.Y_c4 = l.Y_c4*var(18);
l.Y_pro = l.Y_pro*var(19);
l.Y_ac = l.Y_ac*var(20);
l.Y_h2 = l.Y_h2*var(21);
l.Y_lac_f = l.Y_lac_f*var(22);
l.Y_lac_o = l.Y_lac_o*var(23);
l.T_base = l.T_base*var(24);
l.T_op = l.T_op*var(25);
l.P_atm = l.P_atm*var(26);
l.k_p = l.k_p*var(27);
l.k_L_a = l.k_L_a*var(28);
l.V_gas = l.V_gas*var(29);
l.k_dis = l.k_dis*var(30);
l.k_hyd_ch = l.k_hyd_ch*var(31);
l.k_hyd_pr = l.k_hyd_pr*var(32);
l.k_hyd_li = l.k_hyd_li*var(33);
l.K_S_IN = l.K_S_IN*var(34);
l.k_m_su = l.k_m_su*var(35);
l.K_S_su = l.K_S_su*var(36);
l.pH_UL_aa = l.pH_UL_aa*var(37);
l.pH_LL_aa = l.pH_LL_aa*var(38);
l.k_m_aa = l.k_m_aa*var(39);
l.K_S_aa = l.K_S_aa*var(40);
l.k_m_fa = l.k_m_fa*var(41);
l.K_S_fa = l.K_S_fa*var(42);
l.K_I_h2_fa = l.K_I_h2_fa*var(43);
l.k_m_c4 = l.k_m_c4*var(44);
l.K_S_c4 = l.K_S_c4*var(45);
l.K_I_h2_c4 = l.K_I_h2_c4*var(46);
l.k_m_pro = l.k_m_pro*var(47);
l.K_S_pro = l.K_S_pro*var(48);
l.K_I_h2_pro = l.K_I_h2_pro*var(49);
l.k_m_ac = l.k_m_ac*var(50);
l.K_S_ac = l.K_S_ac*var(51);
l.K_I_nh3 = l.K_I_nh3*var(52);
l.pH_UL_ac = l.pH_UL_ac*var(53);
l.pH_LL_ac = l.pH_LL_ac*var(54);
l.k_m_h2 = l.k_m_h2*var(55);
l.K_S_h2 = l.K_S_h2*var(56);
l.pH_UL_h2 = l.pH_UL_h2*var(57);
l.pH_LL_h2 = l.pH_LL_h2*var(58);
l.k_dec_X_su = l.k_dec_X_su*var(59);
l.k_dec_X_aa = l.k_dec_X_aa*var(60);
l.k_dec_X_fa = l.k_dec_X_fa*var(61);
l.k_dec_X_c4 = l.k_dec_X_c4*var(62);
l.k_dec_X_ac = l.k_dec_X_ac*var(63);
l.k_dec_X_h2 = l.k_dec_X_h2*var(64);
l.k_m_lac_f = l.k_m_lac_f*var(65);
l.K_S_lac_f = l.K_S_lac_f*var(66);
```

```
l.k_m_lac_o = l.k_m_lac_o*var(67);
l.K_S_lac_o = l.K_S_lac_o*var(68);
l.K_I_h2_lac_o = l.K_I_h2_lac_o*var(69);
l.K_I_vfa = l.K_I_vfa*var(70);
l.k_dec_X_lac_f = l.k_dec_X_lac_f*var(71);
l.k_dec_X_lac_o = l.k_dec_X_lac_o*var(72);
l.K_S_p_caco3 = l.K_S_p_caco3*var(73);
l.K_r_caco3 = l.K_r_caco3*var(74);
end

%function for storing parameter positions to be used in the definition of
%the derivative
function [position] = ref_value(l,n_param)
position = zeros(1,n_param);
position(1) = l.f_sI_xc;
position(2) = l.f_xI_xc;
position(3) = l.f_ch_xc;
position(4) = l.f_pr_xc;
position(5) = l.f_li_xc;
position(6) = l.f_h2_su;
position(7) = l.f_bu_su;
position(8) = l.f_pro_su;
position(9) = l.f_ac_su;
position(10) = l.Y_su;
position(11) = l.f_h2_aa;
position(12) = l.f_va_aa;
position(13) = l.f_bu_aa;
position(14) = l.f_pro_aa;
position(15) = l.f_ac_aa;
position(16) = l.Y_aa;
position(17) = l.Y_fa;
position(18) = l.Y_c4;
position(19) = l.Y_pro;
position(20) = l.Y_ac;
position(21) = l.Y_h2;
position(22) = l.Y_lac_f;
position(23) = l.Y_lac_o;
position(24) = l.T_base;
position(25) = l.T_op;
position(26) = l.P_atm;
position(27) = l.k_p;
position(28) = l.k_L_a;
position(29) = l.V_gas;
position(30) = l.k_dis;
position(31) = l.k_hyd_ch;
position(32) = l.k_hyd_pr;
position(33) = l.k_hyd_li;
position(34) = l.K_S_IN;
position(35) = l.k_m_su;
position(36) = l.K_S_su;
position(37) = l.pH_UL_aa;
position(38) = l.pH_LL_aa;
position(39) = l.k_m_aa;
```

```
position(40) = l.K_S_aa;
position(41) = l.k_m_fa;
position(42) = l.K_S_fa;
position(43) = l.K_I_h2_fa;
position(44) = l.k_m_c4;
position(45) = l.K_S_c4;
position(46) = l.K_I_h2_c4;
position(47) = l.k_m_pro;
position(48) = l.K_S_pro;
position(49) = l.K_I_h2_pro;
position(50) = l.k_m_ac;
position(51) = l.K_S_ac;
position(52) = l.K_I_nh3;
position(53) = l.pH_UL_ac;
position(54) = l.pH_LL_ac;
position(55) = l.k_m_h2;
position(56) = l.K_S_h2;
position(57) = l.pH_UL_h2;
position(58) = l.pH_LL_h2;
position(59) = l.k_dec_X_su;
position(60) = l.k_dec_X_aa;
position(61) = l.k_dec_X_fa;
position(62) = l.k_dec_X_c4;
position(63) = l.k_dec_X_ac;
position(64) = l.k_dec_X_h2;
position(65) = l.k_m_lac_f;
position(66) = l.K_S_lac_f;
position(67) = l.k_m_lac_o;
position(68) = l.K_S_lac_o;
position(69) = l.K_I_h2_lac_o;
position(70) = l.K_I_vfa;
position(71) = l.k_dec_X_lac_f;
position(72) = l.k_dec_X_lac_o;
position(73) = l.K_S_p_caco3;
position(74) = l.K_r_caco3;
end
```

D.5 Variable dependence

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%
%Names of variables and parameters are used from Rosen and Jeppsson
%implementation in report "Aspect of ADM1 Implementaion within the BSM2
%Framework", 2006.
%
%Function for testing the sensitivity parameters when an parameter changes.
%To get an other parameter instead of inorganic nitrogen. Just change the
%terms correlated to inorganic nitrogen and change an other variable
%instead.


function S_IN_sensitivity
addpath('../ADM1') %to be used for the sorting applied in folder hierarchy


clear all
close all

%retrive input data and parameter values
l = IndataADM1_v2;

%start values from Hamse's Algerian colleague
y0=[0.009;...    % S_su
    0.0009;...   % S_aa
    0.0009;...   % S_fa
    0.0009;...   % S_va
    0.0009;...   % S_bu
    0.0009;...   % S_pro
    0.0009;...   % S_ac
    2.3594e-9;...%S_h2
    2.3594e-6;...%S_ch4
    0.039;...    %S_IC
    0.13023;...  %S_IN
    0.009;...    %S_I
    0.30870;...  %X_c
    0.02795;...  %X_ch
    0.10260;...  %X_pr
    0.02948; ... %X_li
    0.42016;...  %X_su
    1.17917;...  %X_aa
```

```
    0.24303;...  %X_fa
    0.43192;...  %X_c4
    0.13730;...  %X_pro
    0.76056;...  %X_ac
    0.31702;...  %X_h2
    25.61739;... %X_I
    0.04;...     %S_cat
    0.02;...     %S_an
    0.0116;...   %S_vam
    0.01322;...  %S_bum
    0.01574;...  %S_prom
    0.19724;...  %S_acm
    0.14278;...  %S_hco3m
    0.00409;...  %S_nh3
    1.023e-5;... %h2
    1.62125;...  %ch4
    0.01411;...  %co2
    0;...        %S_lac
    0;...        %X_lac_f
    0;...        %X_lac_o
    0];          %S_ca



%define range for variable interval
S_IN_in_var = linspace(0,10,1000);

sens_vec_jacobi = zeros(2,length(S_IN_in_var));   %vector with sensibility stored
                                                    %(derivative method)
sens_vec_diff   = zeros(2,length(S_IN_in_var));   %vector with sensibility stored
                                                    %(absolute diff)
ref_vec = [l.K_S_IN, l.K_I_nh3];           %reference vector for parameter values

tic
for i = 1:length(S_IN_in_var)
    %display iteration number
    fprintf('Iteration number:%.0f (%i)\nS_IN_in_var:%.4f\n', i, ...
        length(S_IN_in_var), S_IN_in_var(i));

    %chose variable value and calculate reference value
    l.S_IN_in = S_IN_in_var(i);
    gas_ref = calculate_mean_gas(l,y0);     %reference value for gas production
    toc

    %calculate change for parameter 1
    l.K_S_IN = l.K_S_IN*1.01;                   %change parameter value
    gas_new_K_S_IN = calculate_mean_gas(l,y0);  %calculate new gas production
    l.K_S_IN = ref_vec(1);                      %reset parameter
    toc

    %calculate change for parameter 2
    l.K_I_nh3 = l.K_I_nh3*1.01;                 %change parameter value
    gas_new_K_I_nh3 = calculate_mean_gas(l,y0); %calculate new gas production
```

```
    l.K_I_nh3 = ref_vec(2);                    %reset parameter
    toc

    %store new value (derivative method)
    sens_vec_jacobi(1,i) = (gas_new_K_S_IN-gas_ref)/(0.01*ref_vec(1));
    sens_vec_jacobi(2,i) = (gas_new_K_I_nh3-gas_ref)/(0.01*ref_vec(2));

    %store new value (absolute difference method)
    sens_vec_diff(1,i) = gas_new_K_S_IN-gas_ref;
    sens_vec_diff(2,i) = gas_new_K_I_nh3-gas_ref;

end

%save file for future use
fileName = ['S_IN_in_var-' datestr(now,30)] %use time and date to get unique name
save(fileName);

%plot absolute derivative method
figure
subplot(2,1,1)
semilogy(S_IN_in_var, abs(sens_vec_jacobi(1,:)))
xlabel('S_{IN}')
ylabel('Sensitivity')
legend('K_{S,IN}')
title('Sensitivity of K_{S,IN} and K_{I,nh3} as a function of S_{IN}', ...
    'fontSize', 16)


subplot(2,1,2)
semilogy(S_IN_in_var, abs(sens_vec_jacobi(2,:)),'g')
xlabel('S_{IN}')
ylabel('Sensitivity')
legend('K_{I,nh3}')


%plot absolute diff method
figure
subplot(2,1,1)
semilogy(S_IN_in_var, abs(sens_vec_diff(1,:)))
xlabel('S_{IN}')
ylabel('Sensitivity')
legend('K_{S,IN}')
title('Sensitivity of K_{S,IN} and K_{I,nh3} as a function of S_{IN}', 'fontSize', 16)


subplot(2,1,2)
semilogy(S_IN_in_var, abs(sens_vec_diff(2,:)),'g')
xlabel('S_{IN}')
ylabel('Sensitivity')
legend('K_{I,nh3}')

end
```

```
%function for calculating gas production when the process has reached a
%steady state
function mean_gas = calculate_mean_gas(l,y0)
%settings for ODE-solver
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);

%solve ADM1 problem
[T,solution] = ode15s(@(t,y) ADM1_fun_v2_ODE(t,y,l), [0 200],y0, options);

%retrive gas concentration
S_gas_h2  = solution(:,33);
S_gas_ch4 = solution(:,34);
S_gas_co2 = solution(:,35);

%calculate gas pressure
P_gas_h2  = S_gas_h2*l.R*l.T_op/16;
P_gas_ch4 = S_gas_ch4*l.R*l.T_op/64;
P_gas_co2 = S_gas_co2*l.R*l.T_op;

P_gas = P_gas_h2+P_gas_ch4+P_gas_co2+l.p_gas_h2o; %total gas pressure
q_gas     = l.k_p*(P_gas-l.P_atm).*P_gas/l.P_atm; %total gas flow

q_gas_ch4 = P_gas_ch4./P_gas.*q_gas;        %CH4 gas flow
mean_gas = mean(q_gas_ch4(end-100:end));    %mean gas production at steady state
end
```

D.6 Particle Swarm optimization

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%File for running a Particle Swarm Optimization (PSO) method in order to
%find maximum gas possibil to produce out of co digesting between manure
%and waste water.
%
%The implementation follows the book "Biologically inspired optimization
%methods" by Wahde, Mattias
%
%By chancing the intervalls called limits below the possible intervalls in
%which the solution lives can be determined. If one wants to have a fixed
%value for one or more of those quantities chosing upper and lower limits
%equal will fix this.

function OneFilePSO_vector_report
addpath('../ADM1');

%define parameters for optimization
%.....................................................................
%define limits in which the solution lives
limits      = [ 0.01,  0.01;  ... S_su_min,  S_su_max
                0.001, 0.001; ... S_aa_min,  S_aa_max
                0.001, 0.001; ... S_fa_min,  S_fa_max
                0.001, 0.001; ... S_va_min,  S_va_max
                0.001, 0.001; ... S_bu_min,  S_bu_max
                0.001, 0.001; ... S_pro_min, S_pro_max
                0.001, 0.001; ... S_ac_min,  S_ac_max
```

```
                1e-8,  1e-8;  ... S_h2_min,  S_h2_max
                1e-5,  1e-5;  ... S_ch4_min, S_ch4_max
                0,     1;     ... S_IC_min,  S_IC_max
                0,     1;     ... S_IN_min,  S_IN_max
                0.02,  0.02;  ... S_I_min,   S_I_max
                %.....................................
                2,     2;     ... X_xc_min,  X_xc_max
                5,     5;     ... X_ch_min,  X_ch_max
                20,    20;    ... X_pr_min,  X_pr_max
                5,     5;     ... X_li_min,  X_li_max
                0,     0;     ... X_su_min,  X_su_max
                0.01,  0.01;  ... X_aa_min,  X_aa_max
                0.01,  0.01;  ... X_fa_min,  X_fa_max
                0.01,  0.01;  ... X_c4_min,  X_c4_max
                0.01,  0.01;  ... X_pro_min, X_pro_max
                0.01,  0.01;  ... X_ac_min,  X_ac_max
                0.01,  0.01;  ... X_h2_min,  X_h2_max
                25,    25;    ... X_I_min,   X_I_max
                %.....................................
                0.04,  0.04;  ... S_cat_min, X_cat_max
                0.02,  0.02;  ... S_an_min,  X_an_max
                %.....................................
                100,   200;   ... q_min,     q_max
                %.....................................
                0,     0;     ... S_lac_min, S_lac_max
                0,     0;     ... X_lac_f_min, X_lac_f_max
                0,     0;     ... X_lac_o_min, X_lac_o_max
                0,     0;     ... S_ca_min,  S_ca_max
                %.....................................
                ];
nParticles  = 20;  %number of particles in PSO
alpha       = 1;   %PSO constant
c1          = 2;   %PSO constant
c2          = 2;   %PSO constant
limit       = 1000; %number of iterations

%chosing maximum speed for each dimension to a tenth of its range
vmax = abs((limits(:,2)-limits(:,1))/10);
```

```
%initialize particles
particles = InitializeParticles(nParticles, alpha, limits);

iIterations = 0; %number of iterations done
bestGlobalFunctionValue = inf; %give a large start value to use real values in simulation
bestGlobalPosition = zeros(size(limits,1),1); %define variable for best global position

%do actual optimization loop
while iIterations < limit
    iIterations = iIterations +1;

    %for each particle in swarm
    for i=1:nParticles
        currentParticle = particles(i);
        currentFunctionValue = FunctionToMinimize(currentParticle);

        %test if current value is global and/or particle best
        [bestGlobalFunctionValue, bestGlobalPosition] = ...
            UpdateGlobalBest(currentParticle, currentFunctionValue, ...
            bestGlobalFunctionValue, bestGlobalPosition);
        currentParticle = UpdateParticleBest(currentParticle, currentFunctionValue);

        particles(i) = currentParticle;
    end

    %update all swarm particles states
    for i=1:nParticles
        particles(i) = UpdateParticleState(particles(i), c1, c2, vmax, ...
            bestGlobalPosition, limits);
    end
end %main while loop


end %main function

%function for initializing particles
function particles = InitializeParticles(nParticles, alpha, limits)
```

```
%initialize all particles
for i=1:nParticles
    particle = struct; %defining a struct that will be the particle
    %define fields in the struct to store data
    particle.position     = zeros(size(limits,1),1);
    particle.velocity     = zeros(size(limits,1),1);
    particle.bestLocalPosition = zeros(size(limits,1),1);

    %Positions (at start the first position is the known best)
    for j=1:size(limits,1)
        r = rand;
        particle.position(j) = limits(j,1) + r*(limits(j,2)-limits(j,1));
        particle.bestLocalPosition(j) = particle.position(j);
    end

    %function values, set only known as function best
    particle.fb = FunctionToMinimize(particle);

    %define particle velocity
    for j=1:size(limits,1)
        r = rand;
        particle.velocity(j) = alpha*(-(limits(j,2)-limits(j,1))/...
            2+r*(limits(j,2)-limits(j,1)));
    end

    particles(i) = particle; %save current particle to vector of particles
end %initialize loop

end %initialize function

%function for calculating particles function value
function [ mean_gas ] = FunctionToMinimize(particle)
l = IndataADM1_v2; %read indata from file
%change to correct values of variables (from standard input to current value)
l.S_su_in    = particle.position(1);
l.S_aa_in    = particle.position(2);
l.S_fa_in    = particle.position(3);
```

```
l.S_va_in    = particle.position(4);
l.S_bu_in    = particle.position(5);
l.S_pro_in   = particle.position(6);
l.S_ac_in    = particle.position(7);
l.S_h2_in    = particle.position(8);
l.S_ch4_in   = particle.position(9);
l.S_IC_in    = particle.position(10);
l.S_IN_in    = particle.position(11);
l.S_I_in     = particle.position(12);
l.X_xc_in    = particle.position(13);
l.X_ch_in    = particle.position(14);
l.X_pr_in    = particle.position(15);
l.X_li_in    = particle.position(16);
l.X_su_in    = particle.position(17);
l.X_aa_in    = particle.position(18);
l.X_fa_in    = particle.position(19);
l.X_c4_in    = particle.position(20);
l.X_pro_in   = particle.position(21);
l.X_ac_in    = particle.position(22);
l.X_h2_in    = particle.position(23);
l.X_I_in     = particle.position(24);
l.S_cat_in   = particle.position(25);
l.S_an_in    = particle.position(26);
l.q_in       = particle.position(27);
l.S_lac_in   = particle.position(28);
l.X_lac_f_in = particle.position(29);
l.X_lac_o_in = particle.position(30);
l.S_ca_in    = particle.position(31);

%start values for simulation
y0=[0.009;...    % S_su
    0.0009;...   % S_aa
    0.0009;...   % S_fa
    0.0009;...   % S_va
    0.0009;...   % S_bu
    0.0009;...   % S_pro
    0.0009;...   % S_ac
    2.3594e-9;...%S_h2
```

```
    2.3594e-6;...%S_ch4
    0.039;...    %S_IC
    0.13023;...  %S_IN
    0.009;...    %S_I
    0.30870;...  %X_c
    0.02795;...  %X_ch
    0.10260;...  %X_pr
    0.02948; ... %X_li
    0.42016;...  %X_su
    1.17917;...  %X_aa
    0.24303;...  %X_fa
    0.43192;...  %X_c4
    0.13730;...  %X_pro
    0.76056;...  %X_ac
    0.31702;...  %X_h2
    25.61739;... %X_I
    0.04;...     %S_cat
    0.02;...     %S_an
    0.0116;...   %S_vam
    0.01322;...  %S_bum
    0.01574;...  %S_prom
    0.19724;...  %S_acm
    0.14278;...  %S_hco3m
    0.00409;...  %S_nh3
    1.023e-5;... %h2
    1.62125;...  %ch4
    0.01411;...  %co2
    0;...        %S_lac
    0;...        %X_lac_f
    0;...        %X_lac_o
    0];          %S_ca


%do actual calculation
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);
[T,solution] = ode15s(@(t,y) ADM1_fun_v2_ODE(t,y,1), [0 50],y0, options);
```

```
S_gas_h2  = solution(:,33);
S_gas_ch4 = solution(:,34);
S_gas_co2 = solution(:,35);


P_gas_h2  = S_gas_h2*l.R*l.T_op/16;
P_gas_ch4 = S_gas_ch4*l.R*l.T_op/64;
P_gas_co2 = S_gas_co2*l.R*l.T_op;

P_gas = P_gas_h2+P_gas_ch4+P_gas_co2+l.p_gas_h2o; %total gas pressure
q_gas     = l.k_p*(P_gas-l.P_atm).*P_gas/l.P_atm; %total gas flow

q_gas_ch4 = P_gas_ch4./P_gas.*q_gas;
mean_gas = mean(q_gas_ch4(end-100:end));
mean_gas = -mean_gas; %to find maximum instead of minimum

end %function value calculator


%function for upadate global best if the particles position is global best
function [bestGlobalFunctionValue, bestGlobalPosition] = ...
    UpdateGlobalBest(currentParticle, candidateFunctionValue, ...
    bestGlobalFunctionValue, bestGlobalPosition)

%if cadidate value are better change
if(candidateFunctionValue < bestGlobalFunctionValue)
    bestGlobalFunctionValue = candidateFunctionValue;
    bestGlobalPosition = currentParticle.position;

    fprintf('New best Position found:\n---------------------- \nS_su=%.4f \nS_aa=%.4f \nS_fa=%.4f
    \nS_va=%.4f \nS_bu=%.4f \nS_pro=%.4f \nS_ac=%.4f \nS_h2=%.4f \nS_ch4=%.4f \nS_IC=%.4f \nS_IN=%.4f
    \nS_I=%.4f \nX_xc=%.4f \nX_ch=%.4f \nX_pr=%.4f \nX_li=%.4f \nX_su=%.4f \nX_aa=%.4f \nX_fa=%.4f
    \nX_c4=%.4f \nX_pro=%.4f \nX_ac=%.4f \nX_h2=%.4f \nX_I=%.4f \nS_cat=%.4f \nS_an=%.4f \nq=%.4f
    \nS_lac=%.4f \nX_lac_f=%.4f \nX_lac_o=%.4f \nS_ca=%.4f\n---------------------- \nf=%.8f\n======================\n', ...
        bestGlobalPosition(1),  bestGlobalPosition(2),  bestGlobalPosition(3), ...
        bestGlobalPosition(4),  bestGlobalPosition(5),  bestGlobalPosition(6), ...
        bestGlobalPosition(7),  bestGlobalPosition(8),  bestGlobalPosition(9), ...
        bestGlobalPosition(10), bestGlobalPosition(11), bestGlobalPosition(12), ...
```

```
        bestGlobalPosition(13), bestGlobalPosition(14), bestGlobalPosition(15), ...
        bestGlobalPosition(16), bestGlobalPosition(17), bestGlobalPosition(18), ...
        bestGlobalPosition(19), bestGlobalPosition(20), bestGlobalPosition(21), ...
        bestGlobalPosition(22), bestGlobalPosition(23), bestGlobalPosition(24), ...
        bestGlobalPosition(25), bestGlobalPosition(26), bestGlobalPosition(27), ...
        bestGlobalPosition(28), bestGlobalPosition(29), bestGlobalPosition(30), ...
        bestGlobalPosition(31), -bestGlobalFunctionValue);
    %add minus sign to compensate to display correct value. Function
    %calculation gives minus for plus and vice verca.

end %if

end %UpdateGlobalBest


%function for updating the particle best if it is so
function currentParticle = UpdateParticleBest(currentParticle, candidateFunctionValue)
%if current value is better that best known best, change local best
if(candidateFunctionValue < currentParticle.fb)
    currentParticle.fb = candidateFunctionValue;
    currentParticle.bestLocalPosition = currentParticle.position;
end %if

end %currentParticle


%function for moving the particle and update its states
%keeps track of moving the particle inside the boundaries
function particle = UpdateParticleState(particle, c1, c2, vmax, bestGlobalPosition, limits)

%update  velocities and positions (but check if boudaries are fullfilled)
for i=1:size(limits,1)
    %find to random numbers to be used in updating new velocity
    q = rand();
    r = rand();

    particle.velocity(i) = c1*q*(particle.bestLocalPosition(i) - particle.position(i)) + ...
        c2*r*(bestGlobalPosition(i)-particle.position(i));
```

```
    %set the velocity to be in allowed region
    particle.velocity(i) = min(particle.velocity(i), vmax(i));
    particle.velocity(i) = max(particle.velocity(i), -vmax(i));

    %setting new position
    particle.position(i) = particle.position(i) + particle.velocity(i);

    %take care of boundaries
    if(particle.position(i) < limits(i,1))
        particle.position(i) = limits(i,1);
    elseif(particle.position(i) > limits(i,2))
        particle.position(i) = limits(i,2);
    end
end

end %UpdateParticleState
```

D.7 GUI

D.7.1 Run GUI (mac)

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%Main function for running the GUI on mac
%


function RunGUIMac
GUI_ADM1;
end
```

D.7.2 Run GUI (Windows)

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%Main function for running the GUI on Windows. Due to problems (chnages) in
%centimeter definitions between mac and Windows are positions redefined in
%GUI to work in an Windows environment. Everything else is kept.
%

%run gui, and get access to all graphical objects to set their psitions
go = GUI_ADM1;



%meassure screen height to fit everything it the displayable area
screenHeight = get(0, 'screenSize')/get(0, 'ScreenPixelsPerInch')*2.54;
screenHeight = screenHeight(4); %height of screen


%screenHeight = 15; %use this to simulate smaller scree

%define figure (window) for perfect conditions
windowHeight = 24;          %main fig window height
inTableHeight = 14.9;       %in table height
inTableTextPos = 22.6;      %text position on top of in table
startTableHeight = 18.6;    %start table height
startTableTextPos = 22.6;   %text position on top of start table

%redefine positions and heights if not fitted in viewable area
if(screenHeight - 2 < windowHeight)
    windowHeight = screenHeight-2;
    inTableHeight = windowHeight - 7.7 - 1;
    inTableTextPos = inTableHeight + 7.7;
    startTableHeight = windowHeight - 4 - 1;
    startTableTextPos = startTableHeight + 4;
end

%settings for main window and its input components
set(go.mainFigHandle, 'position', [10, 2, 23.5, windowHeight])
set(go.handleInTable, 'position', [1 7.7 7 inTableHeight])
set(go.inTableText, 'position', [1 inTableTextPos 7 .8])
set(go.handleStartTable, 'position', [9 4 7.1 startTableHeight])
set(go.startTableText, 'position', [9 startTableTextPos 7.1 .8])

%settings for plot selections
```

```
boxGroupHeight = 18.6;

if(boxGroupHeight > windowHeight)
    boxGroupHeight = startTableHeight;
end


set(go.handleBoxGroup, 'position', [17.1 4 5.2 boxGroupHeight]);
for i = 1:length(go.handleCheckBoxes)
    if(i<25)
        set(go.handleCheckBoxes(i), 'position', [0.1, boxGroupHeight-...
            (boxGroupHeight/(length(go.handleCheckBoxes))*2)*i, 2.5, .4]);
    else
        set(go.handleCheckBoxes(i), 'position', [2.6, boxGroupHeight-...
            (boxGroupHeight/(length(go.handleCheckBoxes))*2)*(i-24), 2.2, .4]);
    end
end
set(go.plotSelectionText, 'position', [17.1 4+boxGroupHeight, 5.2 .8])

%settings for parameter figures
set(go.handleFlowField, 'position', [3 6 3 1])
set(go.flowFieldText, 'position', [.5 6 2 .8])
set(go.handleTimeField, 'position', [3 4.5 3 1])
set(go.timeFieldText, 'position', [.5 4.5 2 .8])

set(go.handleParamTable(1), 'position', [1 1 7.8 15])
set(go.handleParamTable(1), 'columnWidth', {'auto', 110})
set(go.paramFigHandle(1), 'position', [2 2 9.8 17.5])
set(go.paramFigText(1), 'position', [1 16 7.8 .8])

set(go.handleParamTable(2), 'position', [1 1 6.7 12])
set(go.paramFigText(2), 'position', [1 13 6.7 .8])
set(go.paramFigHandle(2), 'position', [2 2 8.7 14.5])

set(go.handleParamTable(3), 'position', [1 1 6.1 1.6])
set(go.paramFigText(3), 'position', [1 2.6 6.1 .8])
set(go.paramFigHandle(3), 'position', [2 2 8.1 4])

set(go.handleParamTable(4), 'position', [1 1 8.3 15])
set(go.paramFigText(4), 'position', [1 16 8.3 .8])
set(go.paramFigHandle(4), 'position', [2 2 10.3 17.3])
```

D.7.3 Main GUI

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%File for runnig the implemented ADM1 model without using the coding
%environment of MATLAB. The source code below can be compiled to an stand
%alone executable that can be used by persons lacking access to MATLAB.
%

function [graphicalObjects] = GUI_ADM1

addpath('../ADM1')


%read this to fill fields
l = IndataADM1_v2;
l.time = 50; %chose default simulation time to be 50

hf = figure; %handle to main window

%declare figures for edit parameter values
hf_Param(1) = figure;
set(hf_Param(1), ...
    'Name', 'Stoichiometric parameter values', ...
    'NumberTitle', 'off', ...
    'MenuBar', 'none', ...
    'visible', 'off', ...
    'Units', 'centimeters', ...
    'Position', [1 1 12.5 30], ...
    'CloseRequestFcn', @Close_cb);

hf_Param(2) = figure;
set(hf_Param(2), ...
    'Name', 'Physicochemical parameter values', ...
    'NumberTitle', 'off', ...
    'MenuBar', 'none', ...
    'visible', 'off', ...
    'Units', 'centimeters', ...
    'Position', [1 1 11.6 18], ...
    'CloseRequestFcn', @Close_cb);

hf_Param(3) = figure;
set(hf_Param(3), ...
    'Name', 'Physical parameter values', ...
    'NumberTitle', 'off', ...
```

```
    'MenuBar', 'none', ...
    'visible', 'on', ...
    'Units', 'centimeters', ...
    'Position', [1 1 10.8 5], ...
    'CloseRequestFcn', @Close_cb);

hf_Param(4) = figure;
set(hf_Param(4), ...
    'Name', 'Biochemcial parameter values', ...
    'NumberTitle', 'off', ...
    'MenuBar', 'none', ...
    'visible', 'off', ...
    'Units', 'centimeters', ...
    'Position', [1 1 13 30], ...
    'CloseRequestFcn', @Close_cb);

%basic settings for current created figure
set(hf, ...
    'Name', 'ADM1 – GUI', ...
    'NumberTitle', 'off', ...
    'MenuBar', 'none', ...
    'Units', 'Centimeters', ...
    'Position', [10, 1, 26.5, 29.5], ...
    'Resize', 'off', ...
    'CloseRequestFcn', {@CloseMainFig,hf_Param});

%======================================================================
%
%                  Define graphical objects
%
%======================================================================

%fill parameter tables with data
[paramStoioTableData, paramStoioTableRowNames, paramStoioTableColumnName] = ...
    FillStoichiometricTable(1);
figure(hf_Param(1))
handleParamTable(1) = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 1.5 1 9.5 27.2],...
    'Data', paramStoioTableData, ...
    'rowName', paramStoioTableRowNames, ...
    'ColumnName', paramStoioTableColumnName,...
    'ColumnWidth', {'auto', 110}, ...
    'ColumnEditable', [true, false]);
paramFigText(1) = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [1.5 28.5 8.3 .8], ...
    'String', 'Stoichiometric parameter values', ...
    'BackgroundColor', [.8 .8 .8]);
set(hf_Param(1), 'visible', 'off')

[paramPhysicochemicalTableData, paramPhysicochemicalTableRowNames, ...
```

```
    paramPhysicochemicalTableColumnName] = FillPhysicochemicalTable(1);
figure(hf_Param(2))
handleParamTable(2) = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 1.5 1 8.6 15.2],...
    'Data', paramPhysicochemicalTableData, ...
    'rowName', paramPhysicochemicalTableRowNames, ...
    'ColumnName', paramPhysicochemicalTableColumnName,...
    'ColumnEditable', [true, false]);
paramFigText(2) = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [1.5 16.5 8.6 .8], ...
    'String', 'Physicochemical parameter values', ...
    'BackgroundColor', [.8 .8 .8]);
set(hf_Param(2), 'visible', 'off')

[paramPhysicaclTableData, paramPhysicalTableRowNames, ...
    paramPhysicalTableColumnName] = FillPhysicalTable(1);
figure(hf_Param(3))
handleParamTable(3) = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 1.5 1 7.9 2],...
    'Data', paramPhysicaclTableData, ...
    'rowName', paramPhysicalTableRowNames, ...
    'ColumnName', paramPhysicalTableColumnName,...
    'ColumnEditable', [true, false]);
paramFigText(3) = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [1.5 3.5 7.9 .8], ...
    'String', 'Physicochemical parameter values', ...
    'BackgroundColor', [.8 .8 .8]);
set(hf_Param(3), 'visible', 'off')

[paramBiochemicalTableData, paramBiochemicalTableRowNames, ...
    paramBiochemicalTableColumnName] = FillBiochemicalTable(1);
figure(hf_Param(4))
handleParamTable(4) = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 1.5 0.5 10 28.5],...
    'Data', paramBiochemicalTableData, ...
    'rowName', paramBiochemicalTableRowNames, ...
    'ColumnName', paramBiochemicalTableColumnName,...
    'ColumnEditable', [true, false]);
paramFigText(4) = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [1.5 29 10 .8], ......
    'String', 'Biochemical parameter values', ...
    'BackgroundColor', [.8 .8 .8]);
set(hf_Param(4), 'visible', 'off')
```

```
%select correct figure to place all components in
figure(hf)

%declare a table with all input flows
[inTableData, inTableRowNames,  inTableColumnName] = FillInTable(l);
handleInTable = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 1 9.7 9 18.8],...
    'Data', inTableData, ...
    'rowName', inTableRowNames, ...
    'ColumnName', inTableColumnName,...
    'ColumnEditable', [true, false]);
inTableText = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [1 28.5 9 .8], ...
    'String', 'Inflow', ...
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);


%declare a table with starting values
[startTabledata, startTableRowNames, startTableColumnName] = FillStartTable;
handleStartTable = uitable(...
    'Units', 'Centimeters', ...
    'Position', [ 11 4.35 8.6 24.15],...
    'Data', startTabledata, ...
    'rowName', startTableRowNames, ...
    'ColumnName', startTableColumnName,...
    'ColumnEditable', [true, false]);
startTableText = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [10.5 28.5 8.6 .8], ...
    'String', 'Start values', ...
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);


%define checkboxes for chosing plots from model
nameString = {'S_su', 'S_aa', 'S_fa', 'S_va', ...
    'S_bu', 'S_pro', 'S_ac', 'S_h2', 'S_ch4',...
    'S_IC', 'S_IN', 'S_I', 'X_xc', 'X_ch', ...
    'X_pr', 'X_li', 'X_su', 'X_aa', 'X_fa', ...
    'X_c4', 'X_pro', 'X_ac', 'X_h2', 'X_I', ...
    'S_cat', 'S_an', 'pH', ...
    'S_H+', 'S_va-', 'S_bu-', 'S_pro-', ...
    'S_ac-', 'S_hco3-', 'S_co2', 'S_nh3', ...
    'S_nh4+', 'S_gas,h2', 'S_gas,ch4', 'S_gas,co2', ...
    'p_gas,h2', 'p_gas,ch4', 'p_gas,co2', 'p_gas', ...
    'q_gas', 'S_lac', 'X_lac,f', 'X_lac,o', ...
    'S_ca'};
```

```
boxGroupHeight = 27.5;
handleBoxGroup = uibuttongroup('visible','off', 'Units', 'Centimeters', ...
    'Position', [20.5 1 5 boxGroupHeight]);
for i = 1:length(nameString)
    h_B(i) = uicontrol(...
        'Style', 'checkbox', ...
        'String', nameString(i), ...
        'Units', 'Centimeters', ...
        'Position', [0.1, boxGroupHeight-(boxGroupHeight/length(nameString))*...
            i, 4, .5], ...
        'Parent', handleBoxGroup, ...
        'HandleVisibility','off');
    %define callback under definition for run button
end
set(handleBoxGroup,'SelectedObject',[]);  % No selection
set(handleBoxGroup,'Visible','on');
plotSelectionText = uicontrol(...
    'Style', 'text',...
    'Units', 'Centimeters', ...
    'Position',[20.5 28.5 5 .8], ...
    'String', 'Plot selection', ...
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);


%declare a flow field
handleFlowField = uicontrol(...
    'Style', 'edit', ...
    'Units', 'Centimeters',...
    'Position', [3 7 3 1],...
    'Backgroundcolor', [1 1 1], ...
    'String', num2str(l.q_in));
flowFieldText = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [.5 7 2 .8], ...
    'String', 'Flow', ...
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);


%declare a integration time field
handleTimeField = uicontrol(...
    'Style', 'edit',...
    'Units', 'Centimeters', ...
    'Position', [3 5 3 1], ...
    'Backgroundcolor', [1 1 1], ...
    'String', num2str(l.time));
timeFieldText = uicontrol(...
    'Style', 'text', ...
    'Units', 'Centimeters', ...
    'Position', [.5 5 2 .8], ...
    'String', 'Time', ...
```

```
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);


%declare a run button
leftPos = 12; %position for most left button in group
handleRunButton = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [leftPos+4 1.5 2 1],...
    'String', 'Run', ...
    'Enable', 'off', ...
    'Callback', {@Run_cb,handleInTable,handleFlowField,handleTimeField,...
    handleStartTable, h_B, handleParamTable});
set(h_B, 'Callback', {@Check_cb, handleRunButton, h_B});

handleSaveButton = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [leftPos+2 1.5 2 1],...
    'String', 'Save', ...
    'Callback', {@Save_cb,handleInTable,handleFlowField,handleTimeField,...
    handleStartTable, handleParamTable});

handleLoadButton = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [leftPos 1.5 2 1],...
    'String', 'Load', ...
    'Callback', {@Load_cb,handleInTable,handleFlowField,handleTimeField,...
    handleStartTable, handleParamTable});

%declare buttons for chanching parameter values
paramButton(1) = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [1 1 4 1],...
    'String', 'Stoichiometric', ...
    'Tag', '1', ...
    'Callback', {@EditParameters_cb, hf_Param});
paramButton(2) = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [1 2 4 1],...
    'String', 'Physicochemical', ...
    'Tag', '2', ...
    'Callback', {@EditParameters_cb, hf_Param});

paramButton(3) = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [5 1 4 1],...
    'String', 'Physical', ...
```

```
    'Tag', '3', ...
    'Callback', {@EditParameters_cb, hf_Param});

paramButton(4) = uicontrol(...
    'Style', 'PushButton',...
    'Units', 'Centimeters',...
    'Position', [5 2 4 1],...
    'String', 'Biochemical', ...
    'Tag', '4', ...
    'Callback', {@EditParameters_cb, hf_Param});

paramText = uicontrol(...
    'Style', 'text',...
    'Units', 'Centimeters', ...
    'Position',[1 3 8 .8], ...
    'String', 'Edit Parameter Values', ...
    'FontSize', 14, ...
    'BackgroundColor', [.8 .8 .8]);

%create a struct with all handles to be used when setting sizes of objects
%in a Windows environment
graphicalObjects = struct();
graphicalObjects.parmButton = paramButton;
graphicalObjects.paramText = paramText;
graphicalObjects.loadButton = handleLoadButton;
graphicalObjects.saveButton = handleSaveButton;
graphicalObjects.runButton = handleRunButton;
graphicalObjects.timeFieldText = timeFieldText;
graphicalObjects.handleTimeField = handleTimeField;
graphicalObjects.flowFieldText = flowFieldText;
graphicalObjects.handleFlowField = handleFlowField;
graphicalObjects.plotSelectionText = plotSelectionText;
graphicalObjects.handleCheckBoxes = h_B;
graphicalObjects.handleBoxGroup = handleBoxGroup;
graphicalObjects.inTableText = inTableText;
graphicalObjects.handleInTable = handleInTable;
graphicalObjects.startTableText =  startTableText;
graphicalObjects.handleStartTable = handleStartTable;
graphicalObjects.handleParamTable = handleParamTable;
graphicalObjects.paramFigText = paramFigText;
graphicalObjects.mainFigHandle = hf;
graphicalObjects.paramFigHandle = hf_Param;
end

%.....................................................................
%main execution buttons (load, save and run)
%.....................................................................

%function that loads saved values from an old run
function Load_cb(handle, event, handleInTable,handleFlowField,...
    handleTimeField,handleStartTable, handleParamTable)
[file_name, path_to_file] = uigetfile('*.mat', 'title')
```

```
%if cancle do nothing
if(file_name == 0)
    return
end

%load file
load([path_to_file file_name]);

%update all data fields to the loaded values
set(handleInTable, 'Data', inData);
set(handleFlowField, 'String', num2str(flowData))
set(handleTimeField, 'String', num2str(timeData))
set(handleStartTable, 'Data', startData);
set(handleParamTable(1), 'Data', stoichiometricData);
set(handleParamTable(2), 'Data', physiochemicalData);
set(handleParamTable(3), 'Data', physicalData);
set(handleParamTable(4), 'Data', biochemicalData);
end

%funcctiond that save states for future use
function Save_cb(handle, event, handleInTable,handleFlowField,...
    handleTimeField,handleStartTable, handleParamTable)
%read and data from all fields
inData = get(handleInTable, 'Data');
flowData = str2double(get(handleFlowField,'String'));
timeData = str2double(get(handleTimeField,'String'));
startData = get(handleStartTable, 'Data');
stoichiometricData = get(handleParamTable(1),'Data');
physiochemicalData = get(handleParamTable(2), 'Data');
physicalData = get(handleParamTable(3), 'Data');
biochemicalData = get(handleParamTable(4), 'Data');

%save the data with a specific file name (chosen at execution time)
uisave({'inData', 'flowData', 'timeData', 'startData', ...
    'stoichiometricData', 'physiochemicalData', 'physicalData', ...
    'biochemicalData'},'unitled');
end

%calback for run button
function Run_cb(handle, event, handleInTable, handleFlowField, ...
    handleTimeField, handleStartTable, handleCheckedPlot, handleParamTable)
%read updated indata from parameters from file, table and flow field
set(handle,'enable', 'off') %prevent user from starting a new simulation
                            %while one is running
drawnow
[l, y0] = Update_values(handleInTable, handleFlowField, handleTimeField, ...
    handleStartTable, handleParamTable);

%solve ADM1 system of ODEs
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);
[T,solution] = ode15s(@(t,y) ADM1_fun_v2_ODE(t,y,l), [0 l.time],y0, options);

%retrive and store solution in result struct * r *
```

```
r.S_su      = solution(:,1);
r.S_aa      = solution(:,2);
r.S_fa      = solution(:,3);
r.S_va      = solution(:,4);
r.S_bu      = solution(:,5);
r.S_pro     = solution(:,6);
r.S_ac      = solution(:,7);
r.S_h2      = solution(:,8);
r.S_ch4     = solution(:,9);
r.S_IC      = solution(:,10);
r.S_IN      = solution(:,11);
r.S_I       = solution(:,12);
r.X_c       = solution(:,13);
r.X_ch      = solution(:,14);
r.X_pr      = solution(:,15);
r.X_li      = solution(:,16);
r.X_su      = solution(:,17);
r.X_aa      = solution(:,18);
r.X_fa      = solution(:,19);
r.X_c4      = solution(:,20);
r.X_pro     = solution(:,21);
r.X_ac      = solution(:,22);
r.X_h2      = solution(:,23);
r.X_I       = solution(:,24);
r.S_cat     = solution(:,25);
r.S_an      = solution(:,26);
r.S_vam     = solution(:,27);
r.S_bum     = solution(:,28);
r.S_prom    = solution(:,29);
r.S_acm     = solution(:,30);
r.S_hco3m   = solution(:,31);
r.S_nh3     = solution(:,32);
r.S_gas_h2  = solution(:,33);
r.S_gas_ch4 = solution(:,34);
r.S_gas_co2 = solution(:,35);
r.S_lac     = solution(:,36);
r.X_lac_f   = solution(:,37);
r.X_lac_o   = solution(:,38);
r.S_ca      = solution(:,39);

r.T = T; %save simulation time

%calculate pressure
r.P_gas_h2  = r.S_gas_h2*l.R*l.T_op/16;
r.P_gas_ch4 = r.S_gas_ch4*l.R*l.T_op/64;
r.P_gas_co2 = r.S_gas_co2*l.R*l.T_op;

%call function to plot wanted result
PlotGUIResult(l, r, handleCheckedPlot, solution);

%make it possible to run simulation again
set(handle, 'enable', 'on')
end
```

```
%calback for check boxes that activates Run button when any plot is
%selected
function Check_cb(handle, event, handleRunButton, h_B)
%if box is selected enable Run and return else test if there is an other
%box checked
if(get(handle, 'Value'))
    set(handleRunButton, 'enable', 'on')
    return
else
    %see if any box is still checked if so do nothing, else disable Run
    numPlots = 0;
    for i = 1:length(h_B)
        if(get(h_B(i), 'Value'))
            numPlots = numPlots + 1;
            continue
        end
    end
    if(numPlots == 0)
        set(handleRunButton, 'enable', 'off');
    end
end
end


%.....................................................................
%fill (at start of program) and view parameter tables
%.....................................................................

%function for filling biochemical parameter table
function [data, rowNames, columnName] = FillBiochemicalTable(l)
%read data from l
data = {...
    l.k_dis,         ' d^-1';
    l.k_hyd_ch,      ' d^-1';
    l.k_hyd_pr,      ' d^-1';
    l.k_hyd_li,      ' d^-1';
    l.K_S_IN,        ' M';
    l.k_m_su,        ' d^-1';
    l.K_S_su,        ' kg COD m^-3';
    l.pH_UL_aa,      ' -';
    l.pH_LL_aa,      ' -';
    l.k_m_aa,        ' d^-1';
    l.K_S_aa,        ' kg COD m^-3';
    l.k_m_fa,        ' d^-1';
    l.K_S_fa,        ' kg COD m^-3';
    l.K_I_h2_fa,     ' kg COD m^-3';
    l.k_m_c4,        ' d^-1';
    l.K_S_c4,        ' kg COD m^-3';
    l.K_I_h2_c4,     ' kg COD m^-3';
    l.k_m_pro,       ' d^-1';
    l.K_S_pro,       ' kg COD m^-3';
    l.K_I_h2_pro,    ' kg COD m^-3';
```

```
    l.k_m_ac,        ' d^-1';
    l.K_S_ac,        ' kg COD m^-3';
    l.K_I_nh3,       ' kg COD m^-3';
    l.pH_UL_ac,      ' -';
    l.pH_LL_ac,      ' -';
    l.k_m_h2,        ' d^-1';
    l.K_S_h2,        ' kg COD m^-3';
    l.pH_UL_h2,      ' -';
    l.pH_LL_h2,      ' -';
    l.k_dec_X_su,    ' d^-1';
    l.k_dec_X_aa,    ' d^-1';
    l.k_dec_X_fa,    ' d^-1';
    l.k_dec_X_c4,    ' d^-1';
    l.k_dec_X_pro,   ' d^-1';
    l.k_dec_X_ac,    ' d^-1';
    l.k_dec_X_h2,    ' d^-1';
    l.k_m_lac_f,     ' d^-1';
    l.K_S_lac_f,     ' kg COD m^-3';
    l.k_m_lac_o,     ' d^-1';
    l.K_S_lac_o,     ' kg COD m^-3';
    l.K_I_h2_lac_o,  ' kg COD m^-3';
    l.K_I_vfa,       ' kg COD m^-3';
    l.k_dec_X_lac_f,' d^-1';
    l.k_dec_X_lac_o,' d^-1';
    l.K_S_p_caco3,   ' kg CaCO3 m^-3';
    l.K_r_caco3,     ' d^-1'};

rowNames = {'k_dis', 'k_hyd_ch', 'k_hyd_pr', 'k_hyd_li', 'K_S_IN', 'k_m_su', ...
    'K_S_su', 'pH_UL_aa', 'pH_LL_aa', 'k_m_aa', 'K_S_aa', 'k_m_fa', 'K_S_fa', ...
    'K_I_h2_fa', 'k_m_c4', 'K_S_c4', 'K_I_h2_c4', 'k_m_pro', 'K_S_pro', ...
    'K_I_h2_pro', 'k_m_ac', 'K_S_ac', 'K_I_nh3', 'pH_UL_ac', 'pH_LL_ac', ...
    'k_m_h2', 'K_S_h2', 'pH_UL_h2', 'pH_LL_h2', 'k_dec_X_su', 'k_dec_X_aa', ...
    'k_dec_X_fa', 'k_dec_X_c4', 'k_dec_X_pro', 'k_dec_X_ac', 'k_dec_X_h2', ...
    'k_m_lac_f', 'K_S_lac_f', 'k_m_lac_o', 'K_S_lac_o', 'K_I_h2_lac_o', ...
    'K_I_vfa', 'k_dec_X_lac_f', 'k_dec_X_lac_o', 'K_S_p_caco3', 'K_r_caco3'};

columnName = {'Value', 'Units'};
end

%function for filling physical parameter table
function [data, rowNames, columnName] = FillPhysicalTable(l)
%read data from l
data = {l.V_liq, 'm^3'; ...
        l.V_gas, 'm^3'};
rowNames = {' V_liq', ' V_gas'};
columnName = {'Value', 'Units'};
end

%function for filling physiochemical parameter table
function [data, rowNames, columnName] = FillPhysicochemicalTable(l)
%read data from l
data = {...
    l.R,         ' bar M^-1 K^-1'; ...
```

```
    l.T_base,    ' K'; ...
    l.T_op,      ' K'; ...
    l.K_w,       ' M 10^-14'; ...
    l.K_a_va,    ' M'; ...
    l.K_a_bu,    ' M'; ...
    l.K_a_pro,   ' M'; ...
    l.K_a_ac,    ' M'; ...
    l.K_a_co2,   ' M'; ...
    l.K_a_IN,    ' M'; ...
    l.k_A_B_va,  ' M^-1 d^-1'; ...
    l.k_A_B_bu,  ' M^-1 d^-1'; ...
    l.k_A_B_pro,' M^-1 d^-1'; ...
    l.k_A_B_ac,  ' M^-1 d^-1'; ...
    l.k_A_B_co2,' M^-1 d^-1'; ...
    l.k_A_B_IN,  ' M^-1 d^-1'; ...
    l.P_atm,     ' bar'; ...
    l.p_gas_h2o,' bar'; ...
    l.k_p,       ' bar-1'; ...
    l.k_L_a,     ' d^-1'; ...
    l.K_H_co2,   ' M_liq bar^-1'; ...
    l.K_H_ch4,   ' M_liq bar^-1'; ...
    l.K_H_h2,    ' M_liq bar^-1'; ...
    l.K_a_lac,   ' M'};

rowNames = {'R', 'T_base', 'T_op', 'K_w', 'K_a_va', 'K_a_bu', 'K_a_pro', ...
    'K_a_ac', 'K_a_co2', 'K_a_IN', 'k_A_B_va', 'k_A_B_bu', 'k_A_B_pro', ...
    'k_A_B_ac', 'k_A_B_co2', 'k_A_B_IN', 'P_atm', 'p_gas_h2o', 'k_p', ...
    'k_L_a', 'K_H_co2', 'K_H_ch4', 'K_H_h2', 'K_a_lac'};

columnName = {'Value', 'Units'};
end

%function for filling stoichiometric table
function [data, rowNames, columnName] = FillStoichiometricTable(l)
%read data from l
data = {...
l.f_sI_xc,   ' -'; ...
l.f_xI_xc,   ' -'; ...
l.f_ch_xc,   ' -'; ...
l.f_pr_xc,   ' -'; ...
l.f_li_xc,   ' -'; ...
l.N_xc,      ' kmole N(kg COD)^-1';...
l.N_I,       ' kmole N(kg COD)^-1'; ...
l.N_aa,      ' kmole N(kg COD)^-1'; ...
l.C_xc,      ' kmole C(kg COD)^-1'; ...
l.C_sI,      ' kmole C(kg COD)^-1'; ...
l.C_ch,      ' kmole C(kg COD)^-1'; ...
l.C_pr,      ' kmole C(kg COD)^-1'; ...
l.C_li,      ' kmole C(kg COD)^-1'; ...
l.C_xI,      ' kmole C(kg COD)^-1'; ...
l.C_su,      ' kmole C(kg COD)^-1'; ...
l.C_aa,      ' kmole C(kg COD)^-1'; ...
l.f_fa_li,   ' -'; ...
```

```
l.C_fa,      ' kmole C(kg COD)^-1'; ...
l.f_h2_su,   ' -'; ...
l.f_bu_su,   ' -'; ...
l.f_pro_su,  ' -'; ...
l.f_ac_su,   ' -'; ...
l.N_bac,     ' kmole N(kg COD)^-1'; ...
l.C_bu,      ' kmole C(kg COD)^-1'; ...
l.C_pro,     ' kmole C(kg COD)^-1'; ...
l.C_ac,      ' kmole C(kg COD)^-1'; ...
l.C_bac,     ' kmole C(kg COD)^-1'; ...
l.Y_su,      ' -'; ...
l.f_h2_aa,   ' -'; ...
l.f_va_aa,   ' -'; ...
l.f_bu_aa,   ' -'; ...
l.f_pro_aa,  ' -'; ...
l.f_ac_aa,   ' -'; ...
l.C_va,      ' kmole C(kg COD)^-1'; ...
l.Y_aa,      ' -'; ...
l.Y_fa,      ' -'; ...
l.Y_c4,      ' -'; ...
l.Y_pro,     ' -'; ...
l.C_ch4,     ' kmole C(kg COD)^-1'; ...
l.Y_ac,      ' -'; ...
l.Y_h2,      ' -'; ...
l.C_lac,     ' kmole C(kg COD)^-1'; ...
l.Y_lac_f,   ' -'; ...
l.Y_lac_o,   ' -'};

rowNames = {'f_sI_xc', 'f_xI_xc', 'f_ch_xc', 'f_pr_xc', 'f_li_xc', 'N_xc', ...
    'N_I', 'N_aa', 'C_xc', 'C_sI', 'C_ch', 'C_pr', 'C_li', 'C_xI', 'C_su', ...
    'C_aa', 'f_fa_li', 'C_fa', 'f_h2_su', 'f_bu_su', 'f_pro_su', 'f_ac_su', ...
    'N_bac', 'C_bu', 'C_pro', 'C_ac', 'C_bac', 'Y_su', 'f_h2_aa', 'f_va_aa', ...
    'f_bu_aa', 'f_pro_aa', 'f_ac_aa', 'C_va', 'Y_aa', 'Y_fa', 'Y_c4', 'Y_pro', ...
    'C_ch4', 'Y_ac', 'Y_h2', 'C_lac', 'Y_lac_f', 'Y_lac_o'};

columnName = {'Value', 'Units'};
end

%calback for viewing (and hiding) parameter tables
function EditParameters_cb(handle, event, figHandle)
%get number of parameter set wanted. A tag is defined for each buttons and
%each of thsoe tags connects to one set of parameters.
value = str2double(get(handle, 'tag'));

%if hidden, display otherwise hide
for i = 1:length(figHandle)
    if(value == i)
        if(strcmp(get(figHandle(i), 'visible'), 'on'))
            set(figHandle(i), 'visible', 'off')
        else
            set(figHandle(i), 'visible', 'on')
        end
    end
```

```
end
end


%.....................................................................
%fills variable tables at start of program
%.....................................................................
%function that fills indata table
function [data, rowNames,  columnName] = FillInTable(l)
%read data from l
data = {...
    l.S_su_in,      ' kg COD m^-3'; ...
    l.S_aa_in,      ' kg COD m^-3'; ...
    l.S_fa_in,      ' kg COD m^-3'; ...
    l.S_va_in,      ' kg COD m^-3'; ...
    l.S_bu_in,      ' kg COD m^-3'; ...
    l.S_pro_in,     ' kg COD m^-3'; ...
    l.S_ac_in,      ' kg COD m^-3'; ...
    l.S_h2_in,      ' kg COD m^-3'; ...
    l.S_ch4_in,     ' kg COD m^-3'; ...
    l.S_IC_in,      ' kmole C m^-3'; ...
    l.S_IN_in,      ' kmole N m^-3'; ...
    l.S_I_in,       ' kg COD m^-3'; ...
    l.X_xc_in,      ' kg COD m^-3'; ...
    l.X_ch_in,      ' kg COD m^-3'; ...
    l.X_pr_in,      ' kg COD m^-3'; ...
    l.X_li_in,      ' kg COD m^-3'; ...
    l.X_su_in,      ' kg COD m^-3'; ...
    l.X_aa_in,      ' kg COD m^-3'; ...
    l.X_fa_in,      ' kg COD m^-3'; ...
    l.X_c4_in,      ' kg COD m^-3'; ...
    l.X_pro_in,     ' kg COD m^-3'; ...
    l.X_ac_in,      ' kg COD m^-3'; ...
    l.X_h2_in,      ' kg COD m^-3'; ...
    l.X_I_in,       ' kg COD m^-3'; ...
    l.S_cat_in,     ' kmole m^-3'; ...
    l.S_an_in,      ' kmole m^-3 '; ...
    l.S_lac_in,     ' kg COD m^-3'; ...
    l.X_lac_f_in,   ' kg COD m^-3'; ...
    l.X_lac_o_in,   ' kg COD m^-3'; ...
    l.S_ca_in,      ' kg Ca m^-3'};

rowNames = {'S_su_in', 'S_aa_in', 'S_fa_in', 'S_va_in', 'S_bu_in', ...
    'S_pro_in', 'S_ac_in', 'S_h2_in', 'S_ch4_in', 'S_IC_in', 'S_IN_in', ...
    'S_I_in', 'X_xc_in', 'X_ch_in', 'X_pr_in', 'X_li_in', 'X_su_in', ...
    'X_aa_in', 'X_fa_in', 'X_c4_in', 'X_pro_in', 'X_ac_in', 'X_h2_in', ...
    'X_I_in', 'S_cat_in', 'S_an_in', 'S_lac_in', 'X_lac_f_in', ...
    'X_lac_o_in', 'S_ca_in'};

columnName = {'Value', 'Units'};
end
```

```
%function that fills start value table
function [data, rowNames,  columnName] = FillStartTable
%read data from l
data = {...
    0.009,       ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    0.0009,      ' kgCOD m^-3';
    2.3594e-9,   ' kgCOD m^-3';
    2.3594e-6,   ' kgCOD m^-3';
    0.039,       ' kmole C m^-3';
    0.13023,     ' kmole N m^-3';
    0.009,       ' kgCOD m^-3';
    0.30870,     ' kgCOD m^-3';
    0.02795,     ' kgCOD m^-3';
    0.10260,     ' kgCOD m^-3';
    0.02948,     ' kgCOD m^-3';
    0.42016,     ' kgCOD m^-3';
    1.17917,     ' kgCOD m^-3';
    0.24303,     ' kgCOD m^-3';
    0.43192,     ' kgCOD m^-3';
    0.13730,     ' kgCOD m^-3';
    0.76056,     ' kgCOD m^-3';
    0.31702,     ' kgCOD m^-3';
    25.61739,    ' kgCOD m^-3';
    0.04,        ' kmole m^-3';
    0.02,        ' kmole m^-3';
    0.0116,      ' kgCOD m^-3';
    0.01322,     ' kgCOD m^-3';
    0.01574,     ' kgCOD m^-3';
    0.19724,     ' kgCOD m^-3';
    0.14278,     ' kmole C m^-3';
    0.00409,     ' kmole N m^-3';
    1.023e-5,    ' kgCOD m^-3';
    1.62125,     ' kgCOD m^-3';
    0.01411,     ' kmole C m^-3';
    0,           ' kg COD m^-3';
    0,           ' kg COD m^-3';
    0,           ' kg COD m^-3';
    0,           ' kg Ca m^-3'};

rowNames = {'S_su', 'S_aa', 'S_fa', 'S_va', 'S_bu', 'S_pro', ...
    'S_ac', 'S_h2', 'S_ch4', 'S_IC', 'S_IN', 'S_I', 'X_c', 'X_ch', ...
    'X_pr', 'X_li', 'X_su', 'X_aa', 'X_fa', 'X_c4', 'X_pro', 'X_ac', ...
    'X_h2', 'X_I', 'S_cat', 'S_an', 'S_vam', 'S_bum', 'S_prom', ...
    'S_acm', 'S_hco3m', 'S_nh3', 'S_gas_h2', 'S_gas_ch4', 'S_gas_co2', ...
    'S_lac', 'X_lac_f', 'X_lac_o', 'S_ca'};

columnName = {'Values', 'Units'};
```

```
end


%.....................................................................
%called at execution to get up to date values
%.....................................................................

%function that are called at execution to get up to date values of
%parameters and variables
function [l, y0] = Update_values(handleInTable, handleFlowField, ...
    handleTimeField, handleStartTable, handleParamTable)
%read and update variale l with new up to date parameter values
l = UpdateL(handleParamTable);

%get data from tables
inData = get(handleInTable, 'Data');

l.S_su_in    = inData{1,1};
l.S_aa_in    = inData{2,1};
l.S_fa_in    = inData{3,1};
l.S_va_in    = inData{4,1};
l.S_bu_in    = inData{5,1};
l.S_pro_in   = inData{6,1};
l.S_ac_in    = inData{7,1};
l.S_h2_in    = inData{8,1};
l.S_ch4_in   = inData{9,1};
l.S_IC_in    = inData{10,1};
l.S_IN_in    = inData{11,1};
l.S_I_in     = inData{12,1};
l.X_xc_in    = inData{13,1};
l.X_ch_in    = inData{14,1};
l.X_pr_in    = inData{15,1};
l.X_li_in    = inData{16,1};
l.X_su_in    = inData{17,1};
l.X_aa_in    = inData{18,1};
l.X_fa_in    = inData{19,1};
l.X_c4_in    = inData{20,1};
l.X_pro_in   = inData{21,1};
l.X_ac_in    = inData{22,1};
l.X_h2_in    = inData{23,1};
l.X_I_in     = inData{24,1};
l.S_cat_in   = inData{25,1};
l.S_an_in    = inData{26,1};
l.S_lac_in   = inData{27,1};
l.X_lac_f_in = inData{28,1};
l.X_lac_o_in = inData{29,1};
l.S_ca_in    = inData{30,1};

%get data from flow field
l.q_in = str2double(get(handleFlowField,'String'));

%read time
```

```
l.time = str2double(get(handleTimeField,'String'));

%read start values
y0Temp = get(handleStartTable, 'Data');
y0 = zeros(length(y0Temp),1);
for i = 1:length(y0Temp)
    y0(i) = y0Temp{i,1};
end

end

%update the values of parameter l for use during calculation
function l = UpdateL(handleParamTable)
%get data from tables and put it into struct l

%retrive data ...
stoichiometricData = get(handleParamTable(1),'Data');
physiochemicalData = get(handleParamTable(2), 'Data');
physicalData = get(handleParamTable(3), 'Data');
biochemicalData = get(handleParamTable(4), 'Data');

%...and store it in l
l.f_sI_xc   = stoichiometricData{1,1};
l.f_xI_xc   = stoichiometricData{2,1};
l.f_ch_xc   = stoichiometricData{3,1};
l.f_pr_xc   = stoichiometricData{4,1};
l.f_li_xc   = stoichiometricData{5,1};
l.N_xc      = stoichiometricData{6,1};
l.N_I       = stoichiometricData{7,1};
l.N_aa      = stoichiometricData{8,1};
l.C_xc      = stoichiometricData{9,1};
l.C_sI      = stoichiometricData{10,1};
l.C_ch      = stoichiometricData{11,1};
l.C_pr      = stoichiometricData{12,1};
l.C_li      = stoichiometricData{13,1};
l.C_xI      = stoichiometricData{14,1};
l.C_su      = stoichiometricData{15,1};
l.C_aa      = stoichiometricData{16,1};
l.f_fa_li   = stoichiometricData{17,1};
l.C_fa      = stoichiometricData{18,1};
l.f_h2_su   = stoichiometricData{19,1};
l.f_bu_su   = stoichiometricData{20,1};
l.f_pro_su  = stoichiometricData{21,1};
l.f_ac_su   = stoichiometricData{22,1};
l.N_bac     = stoichiometricData{23,1};
l.C_bu      = stoichiometricData{24,1};
l.C_pro     = stoichiometricData{25,1};
l.C_ac      = stoichiometricData{26,1};
l.C_bac     = stoichiometricData{27,1};
l.Y_su      = stoichiometricData{28,1};
l.f_h2_aa   = stoichiometricData{29,1};
l.f_va_aa   = stoichiometricData{30,1};
l.f_bu_aa   = stoichiometricData{31,1};
```

```
l.f_pro_aa  = stoichiometricData{32,1};
l.f_ac_aa   = stoichiometricData{33,1};
l.C_va      = stoichiometricData{34,1};
l.Y_aa      = stoichiometricData{35,1};
l.Y_fa      = stoichiometricData{36,1};
l.Y_c4      = stoichiometricData{37,1};
l.Y_pro     = stoichiometricData{38,1};
l.C_ch4     = stoichiometricData{39,1};
l.Y_ac      = stoichiometricData{40,1};
l.Y_h2      = stoichiometricData{41,1};
l.C_lac     = stoichiometricData{42,1};
l.Y_lac_f   = stoichiometricData{43,1};
l.Y_lac_o   = stoichiometricData{44,1};

l.R         = physiochemicalData{1,1};
l.T_base    = physiochemicalData{2,1};
l.T_op      = physiochemicalData{3,1};
l.K_w       = physiochemicalData{4,1};
l.K_a_va    = physiochemicalData{5,1};
l.K_a_bu    = physiochemicalData{6,1};
l.K_a_pro   = physiochemicalData{7,1};
l.K_a_ac    = physiochemicalData{8,1};
l.K_a_co2   = physiochemicalData{9,1};
l.K_a_IN    = physiochemicalData{10,1};
l.k_A_B_va  = physiochemicalData{11,1};
l.k_A_B_bu  = physiochemicalData{12,1};
l.k_A_B_pro = physiochemicalData{13,1};
l.k_A_B_ac  = physiochemicalData{14,1};
l.k_A_B_co2 = physiochemicalData{15,1};
l.k_A_B_IN  = physiochemicalData{16,1};
l.P_atm     = physiochemicalData{17,1};
l.p_gas_h2o = physiochemicalData{18,1};
l.k_p       = physiochemicalData{19,1};
l.k_L_a     = physiochemicalData{20,1};
l.K_H_co2   = physiochemicalData{21,1};
l.K_H_ch4   = physiochemicalData{22,1};
l.K_H_h2    = physiochemicalData{23,1};
l.K_a_lac   = physiochemicalData{24,1};

l.V_liq = physicalData{1,1};
l.V_gas = physicalData{2,1};

l.k_dis         = biochemicalData{1,1};
l.k_hyd_ch      = biochemicalData{2,1};
l.k_hyd_pr      = biochemicalData{3,1};
l.k_hyd_li      = biochemicalData{4,1};
l.K_S_IN        = biochemicalData{5,1};
l.k_m_su        = biochemicalData{6,1};
l.K_S_su        = biochemicalData{7,1};
l.pH_UL_aa      = biochemicalData{8,1};
l.pH_LL_aa      = biochemicalData{9,1};
l.k_m_aa        = biochemicalData{10,1};
l.K_S_aa        = biochemicalData{11,1};
```

```
l.k_m_fa        = biochemicalData{12,1};
l.K_S_fa        = biochemicalData{13,1};
l.K_I_h2_fa     = biochemicalData{14,1};
l.k_m_c4        = biochemicalData{15,1};
l.K_S_c4        = biochemicalData{16,1};
l.K_I_h2_c4     = biochemicalData{17,1};
l.k_m_pro       = biochemicalData{18,1};
l.K_S_pro       = biochemicalData{19,1};
l.K_I_h2_pro    = biochemicalData{20,1};
l.k_m_ac        = biochemicalData{21,1};
l.K_S_ac        = biochemicalData{22,1};
l.K_I_nh3       = biochemicalData{23,1};
l.pH_UL_ac      = biochemicalData{24,1};
l.pH_LL_ac      = biochemicalData{25,1};
l.k_m_h2        = biochemicalData{26,1};
l.K_S_h2        = biochemicalData{27,1};
l.pH_UL_h2      = biochemicalData{28,1};
l.pH_LL_h2      = biochemicalData{29,1};
l.k_dec_X_su    = biochemicalData{30,1};
l.k_dec_X_aa    = biochemicalData{31,1};
l.k_dec_X_fa    = biochemicalData{32,1};
l.k_dec_X_c4    = biochemicalData{33,1};
l.k_dec_X_pro   = biochemicalData{34,1};
l.k_dec_X_ac    = biochemicalData{35,1};
l.k_dec_X_h2    = biochemicalData{36,1};
l.k_m_lac_f     = biochemicalData{37,1};
l.K_S_lac_f     = biochemicalData{38,1};
l.k_m_lac_o     = biochemicalData{39,1};
l.K_S_lac_o     = biochemicalData{40,1};
l.K_I_h2_lac_o  = biochemicalData{41,1};
l.K_I_vfa       = biochemicalData{42,1};
l.k_dec_X_lac_f = biochemicalData{43,1};
l.k_dec_X_lac_o = biochemicalData{44,1};
l.K_S_p_caco3   = biochemicalData{45,1};
l.K_r_caco3     = biochemicalData{46,1};

end


%.....................................................................
%function that handels closing of figures
%.....................................................................

%function to hide parameter figures instead of closing them
function Close_cb(handle, event)
set(handle, 'visible', 'off') %hide figure that were closed (to not loose
                              %the parameter tables)
end

%function to quit enitre application
function CloseMainFig(handle, event, hf_Param)
for i = 1:length(hf_Param)
    delete(hf_Param(i))
```

```
end
delete(handle)
end
```

D.7.4 Plot GUI

```
%Master Thesis
%
%Applied Physics
%Chalmers University of Technolopgy
%Spring 2014
%
%Oskar Danielsson
%oskard@student.chalmers.se
%
%Function to be used for plotting result from simulation when using the GUI
%

function PlotGUIResult(l, r, handleCheckedPlot, solution)
%names to be used with all plots
nameString = {'S_{su}', 'S_{aa}', 'S_{fa}', 'S_{va}', ...
    'S_{bu}', 'S_{pro}', 'S_{ac}', 'S_{h2}', 'S_{ch4}',...
    'S_{IC}', 'S_{IN}', 'S_{I}', 'X_{xc}', 'X_{ch}', ...
    'X_{pr}', 'X_{li}', 'X_{su}', 'X_{aa}', 'X_{fa}', ...
    'X_{c4}', 'X_{pro}', 'X_{ac}', 'X_{h2}', 'X_{I}', ...
    'S_{cat}', 'S_{an}', 'pH}', ...
    'S_{H+}', 'S_{va-}', 'S_{bu-}', 'S_{pro-}', ...
    'S_{ac-}', 'S_{hco3-}', 'S_{co2}', 'S_{nh3}', ...
    'S_{nh4+}', 'S_{gas,h2}', 'S_{gas,ch4}', 'S_{gas,co2}', ...
    'p_{gas,h2}', 'p_{gas,ch4}', 'p_{gas,co2}', 'p_{gas}', ...
    'q_{gas}', 'S_{lac}', 'X_{lac,f}', 'X_{lac,o}', ...
    'S_{ca}'};


%test which plots that needs to be drawn and draw them
for i = 1:length(nameString)
    if(i < 27)
        if(get(handleCheckedPlot(i), 'Value')) %test if checked
            figure
            plot(r.T, solution(:,i))
            title(nameString(i), 'fontSize', 16)
            xlabel('Time [d]', 'fontSize', 14)

            %test which y-label to use
            if(i == 10)
                ylabel('kmole C m^{-3}', 'fontSize', 14)
            elseif(i == 11)
                ylabel('kmole N m^{-3}', 'fontSize', 14)
            elseif(i < 25)
                ylabel('kg COD m^{-3}', 'fontSize', 14)
            else
                ylabel('kmole m^{-3}', 'fontSize', 14)
            end

        end
    elseif(i < 29) %27 and 28
```

```
        if(get(handleCheckedPlot(i), 'Value'))
            figure
            %calculate what needed to perform the plot
            S_nh4 = r.S_IN-r.S_nh3;
            Theta = r.S_cat + S_nh4 - r.S_hco3m - (r.S_acm/64) - ...
                (r.S_prom/112) - (r.S_bum/160)...
                - (r.S_vam/208) - r.S_an + r.S_ca; %added S_ca to orginal equation
            Sh = -Theta/2 + sqrt(Theta.^2 + 4*1.K_w)/2;
            if(Sh <= 0)
                Sh = 1e-12;
            end

            if(i == 27)
                pH = -log10(Sh);
                plot(r.T, pH)
                title(nameString(i), 'fontSize', 16);
                xlabel('Time [D]', 'fontSize', 14)
            else
                plot(r.T, Sh)
                title('S_{H^+}', 'fontSize', 16)
                xlabel('Time [d]', 'fontSize', 14)
                ylabel('kmole H^+ m^{-3}', 'fontSize', 14)
            end
        end
    elseif(i < 34) %29-33
        if(get(handleCheckedPlot(i), 'Value'))
            figure
            plot(r.T, solution(:,i-2))
            title(nameString(i), 'fontSize', 16);
            xlabel('Time [d]', 'fontSize', 14)
            if(i == 33)
                ylabel('kmole C m^{-3}', 'fontSize', 14)
            else
                ylabel('kg COD m^{-3}', 'fontSize', 14)
            end
        end
    elseif(i == 34)
        if(get(handleCheckedPlot(i), 'Value'))
            figure
            S_co2 = r.S_IC-r.S_hco3m;
            plot(r.T, S_co2);
            title(nameString(i), 'fontSize', 16);
            xlabel('Time [d]', 'fontSize', 14)
            ylabel('kmole C m^{-3}', 'fontSize', 14)
        end
    elseif(i == 35)
        if(get(handleCheckedPlot(i), 'Value'))
            figure
            plot(r.T, solution(:,i-3));
            title(nameString(i), 'fontSize', 16);
            xlabel('Time [d]', 'fontSize', 14)
            ylabel('kmole N m^{-3}', 'fontSize', 14)
        end
```

```
elseif(i == 36)
    if(get(handleCheckedPlot(i), 'Value'))
        figure
        S_nh4 = r.S_IN-r.S_nh3
        plot(r.T, S_nh4);
        title(nameString(i), 'fontSize', 16);
        xlabel('Time [d]', 'fontSize', 14)
        ylabel('kmole N m^{-3}', 'fontSize', 14)
    end
elseif(i < 40) %37-39
    if(get(handleCheckedPlot(i), 'Value'))
        figure
        plot(r.T, solution(:,i-4))
        title(nameString(i), 'fontSize', 16);
        xlabel('Time [d]', 'fontSize', 14)
        if(i == 39)
            ylabel('kmole C m^{-3}', 'fontSize', 14)
        else
            ylabel('kg COD m^{-3}', 'fontSize', 14)
        end
    end
elseif(i < 44) %40-43
    if(get(handleCheckedPlot(i), 'Value'))
        figure

        %total gas pressure
        P_gas = r.P_gas_h2+r.P_gas_ch4+r.P_gas_co2+l.p_gas_h2o;
        if(i == 40)
            plot(r.T, r.P_gas_h2)
        elseif(i == 41)
            plot(r.T, r.P_gas_ch4)
        elseif(i == 42)
            plot(r.T, r.P_gas_co2)
        elseif(i == 43)
            plot(r.T, r.P_gas_h2)
            hold all
            plot(r.T, r.P_gas_ch4)
            plot(r.T, r.P_gas_co2)
            plot(r.T, P_gas, 'linewidth', 2)
            legend('H_2', 'CH_4', 'CO_2', 'total');
        end
        title(nameString(i),'fontSize', 16)
        xlabel('Time [d]', 'fontSize', 14)
        ylabel('Pressure [bar]', 'fontSize', 14)
    end
elseif(i == 44)
    if(get(handleCheckedPlot(i), 'Value'))
        figure
        %total gas pressure
        P_gas     = r.P_gas_h2+r.P_gas_ch4+r.P_gas_co2+l.p_gas_h2o;
        q_gas     = l.k_p*(P_gas-l.P_atm).*P_gas/l.P_atm;   %total gas flow

        %plot total gas flow
```

```
            %calculate gas component gas flow
            q_gas_h2 = r.P_gas_h2./P_gas.*q_gas;
            q_gas_ch4 = r.P_gas_ch4./P_gas.*q_gas;
            q_gas_co2 = r.P_gas_co2./P_gas.*q_gas;
            q_gas_h2o = l.p_gas_h2o./P_gas.*q_gas;

            %plot component gas flow
            plot(r.T, q_gas_h2)
            hold all
            plot(r.T, q_gas_ch4)
            plot(r.T, q_gas_co2)
            plot(r.T, q_gas_h2o)
            plot(r.T, q_gas, 'linewidth', 2)
            legend('H_2', 'CH_4', 'CO_2', 'H_2O', 'total');
            title(nameString(i), 'fontSize', 16)
            xlabel('Time [d]', 'fontSize', 14)
            ylabel('Flow [Nm^3 d^{-1}]', 'fontSize', 14)
        end
    else %45-48
        if(get(handleCheckedPlot(i), 'Value'))
            figure
            plot(r.T, solution(:,i-9))
            title(nameString(i), 'fontSize', 16)
            xlabel('Time [d]', 'fontSize', 14)
            if(i == 48)
                ylabel('kmole Ca m^{-3}', 'fontSize', 14)
            else
                ylabel('kg COD m^{-3}', 'fontSize', 14)
            end
        end
    end


end

end
```