NTNU – Trondheim
Norwegian University of
Science and Technology

Hazard and Operability Study by Utilizing Hybrid Automata

Bjørn Tore Mathisen

Chemical Engineering and Biotechnology
Submission date: June 2014
Supervisor: Heinz A. Preisig, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Abstract

The following thesis is the end of a five year master program at the Norwegian University of Science and Technology, as part of the process eng. group at the institute of chemical engineering. Supervisor for the thesis has been professor Heinz Preisig, who also contributes directly as the thesis revolves around an algorithm developed during P. Philips Phd research period [1] in 2001, with Preisig as a supervisor. In addition his graphical approach to modelling of continuous systems has also served as a large inspiration in this thesis. His continuous support, encouragement and open door policy has been greatly appreciated.

This thesis utilizes the hybrid automaton modelling procedure. A modelling technique designed for dealing with hybrid systems. Hybrid systems are systems with mixed discrete and continuous dynamics. A description fitting a large number of systems, i.e. a digital signal affecting a system with continuous natural behavior. This makes the technique highly applicable in chemical engineering. All modelling of process systems is affected by the person responsible, in term of assumptions and simplifications. In this thesis a three tank system is modelled as a hybrid automaton by discretizing the continuous state space into a set of hypercubes. By doing so a large number of possible state trajectories can be evaluated crudely by considering the predetermined directionality of the transition between hypercubes.

By utilizing the algorithm developed by Phillips [1] a hybrid automaton with a table of possible state transitions is returned. The transition tables displays the directionality of the state in question in regard to a dependable state. By defining a hazardous boundary, i.e. highest allowed temperature, any state reaching a hazardous value can be identified by a transition direction to the hazardous region. Two hypothesis was suggested and evaluated by implementing the algorithm in a case study. The hypothesis was tested by comparing the hazards identified by searching the transition tables to a conventional structurized hazard and operability study. The two hypothesises were:

Hypothesis 1: *Any guideword/parameter combination in a conventional hazard and operability study can be swapped with an evaluation of the hybrid automaton transition table*

Hypothesis 2: *If the automaton is generated over the boundaries spanning the safe operation domain. Any operational hazards must be visible in the*

transition table as a possible transition out of the domain

The case study was a three tank hot and cold liquid mixing plant. The system has a total of 6 states, leading to 6 transition tables. A search algorithm was developed to identify possible hazards. The algorithm successfully identified hazardous transitions for the case study, and the hazards match the results form manual evaluation.

After a comparison between the conventional hazard and operability study and the hybrid automaton approach the hypothesis still stands. However, there is some uncertainty whether the discrete input used by the hybrid automaton, e.g., flow rates being set to max or min, is sufficient for more complex systems. For example, it is well-known that maximum temperature out of exothermic reactors may occur at flow-rates between max and min due to non-linear effects.

In order to apply the algorithm on industrial plants, capabilities to detect and handle completely independent states must be improved. A state variable transfer from enthalpy to temperature resulted in a software crash due to independent states. Such independency will be normal in real plants. Disregarding independent states completely is not beneficial since other states may depend on said state, and this makes it vital to include in the hybrid automaton.

Declaration of Compliance

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Bjørn Tore Mathisen, Trondheim, June 20, 2014

Contents

1 Introduction

The subject of this thesis revolves around trying to automate the hazardous operability (hazop) study frequently performed in chemical processes. This will be done by modeling the system as a hybrid automaton. A hybrid automaton is a modelling frame for systems with both discreet and dynamic behaviour, hybrid systems. The thesis will provide an in dept introduction to the hybrid automaton technique later on, as well as argue that any system may fit under the hybrid system label. The subject of hybrid automaton modelling in chemical process industry is a still undeveloped. Previous master thesis [2] have experienced large difficulties with implementing the technique to complicated systems. This thesis therefore has a large focus on the ground work. Supported with examples that hopefully will awaken interest for the subject as well as displaying key elements. The thesis emphasis a thorough explanation of the hybrid automata, with the hope that the next generation of master students will find the subject less overwhelming, and might wish to continue the development. The subject is recommended to all students who share the authors interest for modelling process systems. And hopefully this thesis will shorten the time needed for research, so that the procedure can be further developed in to handling increasingly complex systems.

The result is two overlapping objectives for the thesis

Objective 1 *Develop an thorough, from the ground up, introduction that may work as a learning tool for future students.*

Objective 2 *Explore the possibility of using the hybrid automaton modelling technique to automate the Hazop procedure*

1.1 Motivation

Personal previous experience in the process industry includes a particular event that motivated the following study. A closed of tank had a leak from it's internal steam coils, transferring energy into the remaining liquid receding in the tank. After enough time had passed the energy which accumulated in the tank, was enough to induce a chemical reaction in an explosive fashion. The plant had been running for decades without a similar experience when the incident took place. A valuable lesson was thought that day. Complex system have an infinite number of possible state trajectories. Trying to predict them all in a conventional modelling fashion is just not viable. No model used for design purposes includes "leak streams" or "hole in the tank

streams". And even if they are included the initial states have a huge impact on how the trajectory develops. If the previously mentioned tank was filled, then the energy transfer from the steam coil leak would reach a steady state with the transfer of energy from the tank to the environment. This means that the only simulation that would portray the hazard, would have to be performed with an initial level close to zero. This is all information that is available after a hazardous incident is investigated. And therefor easily overlooked in the modelling procedure. It can be hard to predict exactly which initial states and disturbances that can lead to a dangerous situation. Therefor a cruder modelling procedure, able to evaluate the net direction of the state trajectory, and therefor cover a larger scale of state trajectories in the state space, is needed.

2 Theoretic Background

2.1 Modelling of continuous systems

2.1.1 The modelling process

The task of establishing a working and sufficient mathematical model is in Preisig [3] divided in to three primary steps:

- **Primary mapping**

The first step of modelling a real-world system is to describe the system mathematically. So if solved, the mathematical model will portray the development of the states of interest over time. To establish such a model requires theoretic knowledge. Of which discipline depends on the timescale, the actual scale of the system and the nature of the system. The established model will in any case be a set of differential equations with a set of accompanying algebraic equations. The differential equations can (depending on the system and performed model simplifications) be partial differential equations or ordinary differential equations. In Preisig [3] a set of ordinary differential equations are labeled as lumps and partial differential equations as distributed. The chosen use and connection of lumps and/or distributed systems tells how the system is assumed to behave and it's important to be aware of the consequences that follows from the chosen differential equations.

- **Model simplifications**

Further simplifications are usually made when the established model is solved for a chosen timescale. A common simplification is linearizing the model in way that simplifies integrating the differential equations. While simplifications that have a direct impact on the algebraic equations, i.e.the chosen equation of state is at this point already established. The selection of such algebraic simplifications belongs under the primary mapping tab. Other model simplifications may be numerical simplifications connected to the chosen solver.

- **Model fitting**

The final primary step is model fitting. This step is dedicated to match the states predicted from the solver with the measured values from the real system. An objective function is established which compares error between measured and calculated values, such that it can be used for adjusting a model parameter. Hence the step is sometimes referred to as parameter identification

What at this point may have become apparent is that a model is subject to individual preferences and procedures. The road to a sufficient model will be largely influenced by the person responsible for the modelling.

2.1.2 Defining lumped and distributed systems

When modelling system which evaluates algebraic and/or differentials by use of intensive quantities, i.e species concentration, it may often be transparent that the intensive variable experience variation even within a given boundary. E.g. a frying pan is heated from the bottom, where a steak resting on it's surface may be burned on the bottom side, while the top of the steak remains quite cold. The intensive quantity temperature varies greatly throughout the steak, which has a large significance on the process performance as the steak may be burned on one side and therefore not meet the quality requirement for serving. Control of the intensive quantity is in this case of great importance. A casserole how ever, which main goal is to boil water, only needs a sufficient amount of energy transfer in to the liquid phase. Any difference in the intensive quantity will in the end be negated by the turbulence of the boiling liquid anyway. This is an example where to similar systems (home cooking) should be modelled different based how relevant the intensive quantities is. As mentioned in section 2.1.1the lumped system is defined as:

Lumped System: A spatial domain where the intensive quantities **don't** change with the position, i.e. boiling water.

Distributed System: A spatial domain where the intensive quantities **do** change with the position,i.e.cooking a steak in a frying pan.

When a need for a distributed description is transparent, the next task is to seek simplifications in the spatial coordinate system. Going back to the steak it can be argued that the only coordinate of interest is the vertical z coordinate, where the energy is transmitted from the frying pan and in to the steak, and then assume that the change in x and y direction is negligible. This assumption should be reasonable provided that the state close to the edges is not the main intrest. In another case maybe the steak is rather a perfectly round meatball, where spherical coordinates can provide symmetric simplifications reducing work load and computational time. Adapting the spatial coordinates to your system have a direct impact on the complexity of the differentials and is therefor a very important modelling step.

2.1.3 The impact of time scales

Selection of timescales is a modelling issue which can a times be overlooked. Where some systems, or part of them, are classified as steady state without any additional reflection is added in the modelling process. In a real system it's easily arguable that nothing is ever constant. At least not at a molecular level, where the somewhat random motion of electrons may induce a sudden dipole in the molecule and therefore work attractive or repulsive at

it's surroundings, i.e. the London dispersion force, one of the classified Van der Waals forces. These changes can happen in a timescale so small that for any system working with rates such as per second, minute, hour, the changes can not be observed. Therefore the selection of the timescales is not just a question of convenience. It's an important assessments one performs which will reflect time scale assumptions. What is the objective of the model? Is it a real time control issue in a plant, where hourly rates are of common use? Is it a model for estimating the need of delivery of raw materials to the plant, where the quantities are monthly delivers with ships? Something that appear event like in one timescale may appear dynamic when looking at a larger scale. And something that look dynamic in one scale may be simplified to steady state in an even larger timescale. Preisig [3] sums up how the timescale affect the appearance of the system with the following figure:

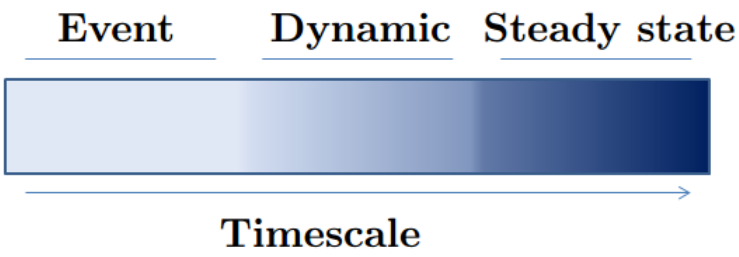

Figure 1: How observing a system under different timescales affect how the system appears

Example highly reactive intermediates

A very good example of time scale assumptions is the theory of highly reactive intermediates. A commonly used simplification for determining the reaction rate of a series of reactions. Here the presence of fast intermediates is used for simplifying the differential equations in to a set of differential equations and algebraic equations. A generic representation of such a reaction is presented in Preisig [3]:

$$A \overset{k_1}{\rightleftharpoons} B \overset{k_2}{\Rightarrow} C \overset{k_3}{\Rightarrow} D$$

Modelling the reaction with the intensive quantity concentration, in a CSTR,

yields the following equations:

$$\dot{c}_A = -k_1 c_A \tag{1}$$
$$\dot{c}_B = k_1 c_A - k_2 c_B \tag{2}$$
$$\dot{c}_C = k_2 c_B - k_3 c_C \tag{3}$$
$$\dot{c}_D = k_3 c_C \tag{4}$$

Or in matrix form by introducing the stoichiometric matrix N:

$$\begin{bmatrix} c_A \\ c_B \\ c_C \\ c_D \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k_1 c_A \\ k_2 c_B \\ k_3 c_C \\ k_4 c_D \end{bmatrix} \tag{5}$$

This is where using figure 1 timescale graph comes in handy. The concentration of interest is the final product. Substance D. And this is where the focus of the timescale lies. The dynamic development of concentration D will be the main output of the model. The intermediates have a dynamic far faster than the dynamics of the starting reactant and the end product. When observing the intermediates they will at all time appear to have a reaction rate always matching the rate of the slow reactions.

$$k_1 c_A = k_2 c_B = k_3 c_C \tag{6}$$

This is where the textbooks like H.S. Fogler [4] refers to a pseudo steady state system, setting the time differentials of species C and D to zero, i.e. saying that the concentrations don't change with time. But in fact the concentration of the intermediate evolves with the concentration of the reactant as can be seen in equation 6. Therefore this thesis suggests that it is much more consistent to refer to this behaviour as having discrete dynamics:

$$\dot{c}_A = -k_1 c_A \tag{7}$$
$$0 = k_1 c_A - k_2 c_B \tag{8}$$
$$0 = k_2 c_B - k_3 c_C \tag{9}$$
$$\dot{c}_D = k_3 c_C \tag{10}$$

Where the concept of steady state, which in effect is a description of a stable long timescale, is removed from vocabulary when dealing with extremely fast dynamics. And that equations 7 to 10 is more appropriately labeled as a hybrid system. The use of the discrete dynamics concept should be a strength, not a limitation as this thesis will work towards showing.

2.2 Graph theory

When developing a model for continuous systems, and especially when dealing with hybrid systems, a graph representation can be a highly powerful tool

when developing the model. The use of the term graph in this context describe a visualization of a network consisting of a set of nodes and displayed transitions between them, in the form of arc's and arrows. Two different definitions is used in this thesis, both have great modelling importance.

2.2.1 Continous systems

The graph used for continuous systems has a purpose of visualizing the system components important for the modelling issue. In general this means keeping control of conserved quantities. The capacities that store them, the direction they are transported and alteration they experience. When done correctly the mathematical description becomes directly linked to the graphical representation. Making it easier to do the actual write up. The chosen elements for presenting the system in a graph is taken from Preisig [3]. Below follows the elements used in this thesis.

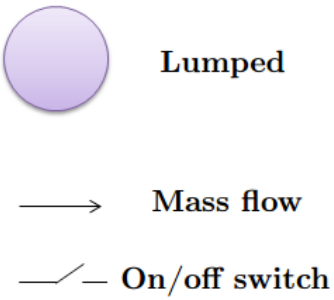

Figure 2: Graph elements for continous systems

2.2.2 Hybrid systems

Hybrid systems will be presented in a shape that visualize the hybrid automaton modelling technique. The technique is presented in 3.1 and for now it's should be sufficient to say that:

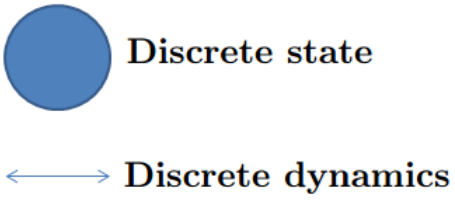

Figure 3: Graph elements for hybrid systems systems

3 Hybrid Systems

The short description of a hybrid system is a system with mixed continuous and discrete behaviour. Where the continuous and event dynamics are interacting and directly effect each other. [5] proposes three sub groups with hybrid behaviour.

- **Phased Operation**

The Bouncing ball - where the continuous trajectory of the velocity of a falling ball is changed directly each time the ball hits the ground.

- **Continuous systems controlled by discrete logic**

Chemical processes - where discrete manipulating interacts with continuous systems.

- **Coordinating Process**

Transportation systems - systems where multiple agents interact and where the automaton can be used i.e. for resolving conflict between the subsystems.

The term hybrid is highly nonrestrictive, as almost any system may be stretched to fit the description. Especially this holds true for the process industry, as a computer operated system is at some level bound to have a discrete nature because of how a computer actually operates. This wide and inconclusive definition turns defining a system as hybrid as a mere starting point. Since it is necessary to restrict the system much further to reach a model suitable for the purpose.

The subject of hybrid systems is also affected by the communities currently contributing to the field. In computer science the main focus may be placed on modelling how discrete computer signals interacts with analog systems, while people working on control theory may explore the possibility of discrete switching of control strategies. These two exampled portray how modelling is an important part of working with hybrid systems. As there is no set of rules that decides exactly what a discrete event is or what is continuous. Manipulating the model to a given purpose may unlock a new set of possibilities as will be seen in the examples in section 3.1.

3.1 Hybrid Automata

A hybrid automata is a model of a hybrid system defined in respect to what the model includes:

- **Discrete states :** Labeled as $\underline{q}$ where the single state $q \in \underline{q}$
- **Continuous states :** Labeled as $\underline{x}$ where the single state $x \in \underline{x}$ and $\underline{x} \in R^n$ where n is the number of continuous states.
- **Discrete inputs :** Labeled as $\underline{\sigma}$ where the single state $\sigma \in \underline{\sigma}$
- **Continuous inputs :** Labeled as $\underline{u}$ where the single state $u \in \underline{u}$
- **Initial states :** $q(0) = q_0$ and $x(0) \in R^n$
- **Continuous dynamics :** $\dot{\underline{x}} = f(\underline{q}, \underline{x}, \underline{u})$
- **Invariant :** E.g. $Inv \subseteq \underline{x} \in R^n$ The outer boundaries for which the model is valid.
- **Discrete dynamics :** R Where a guard (or event detector) initate a discrete change in $\underline{\mathrm{q}}$

When combining the procedure with a graphical representation, the result is a organized and well defined system that may cast a new light at issues that may occur when implementing the model in a solver. In the following sections some different approaches is presented. Where the hybrid automaton is used to better handle issues that may be somewhat concealed and lead to difficulties during implementation. The purpose of the coming examples is also to provide a good understanding of hybrid automaton modelling before entering the case study sections.

3.1.1 Example discrete dynamics: The Bouncing Ball

Modelling a bouncing ball is not a straight forward task if one is lacking experience with event detection. Even though a simplified version of the continuous states, from a momentum balance in a 1-D positional space, seems easily solveable:

$$ma = -mg \tag{11}$$

Rewritten as two first order differential equations with position as the state x, and therefore the acceleration as the second derivative $\dot{\dot{x}}$:

$$\dot{x}_1 = x_2 \tag{12}$$
$$\dot{x}_2 = -g \tag{13}$$

From the equations, starting with the constant acceleration (due to the gravitational force) it's evident that the position x_1 has second order characteristics, while the velocity x_2 has linear characteristics. Making the total system

not very complex. However the complication arises when the ball hits the ground. As the velocity vector readily change from a negative direction to a positive direction. Resulting in a jump in the velocity, in contrast to the continuous linear development experienced while the ball is falling. This event can be modelled as:

$$x_2 := -kx_2 \tag{14}$$

Where k is the coefficient of restitution of the ball. The expected behaviour of the states is a jump in x_2 each time the ball hits the ground. This behavior can be seen below in figure 4 where the system has been simulated in MATLAB (see section C.1.1):

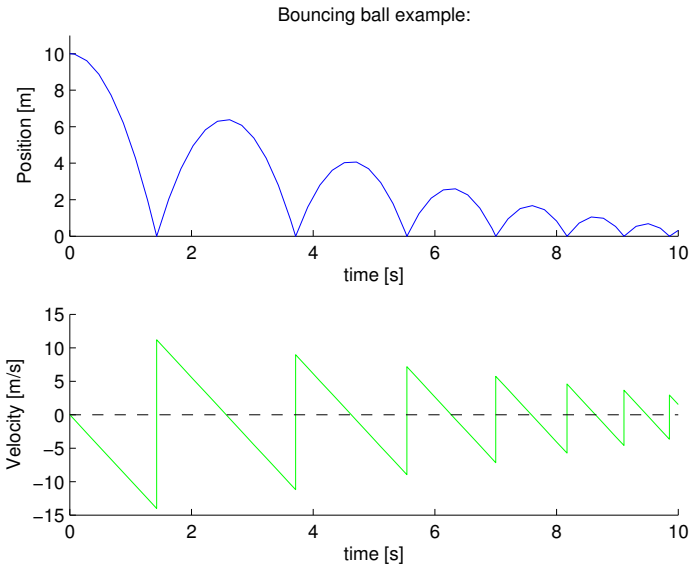

Figure 4: Bouncing ball expected behaviour

By looking at the velocity development it becomes evident that the system has mixed continuous and discrete dynamics, and it may therefor be suitable to model the system as a hybrid automata. Following the formula from top the bottom:

- **Discrete states :** q_0
  After each event the system returns in to following the continuous dynamics proposed in 13. No discrete state change is therefor observed.
- **Continuous states :** $x = [x_1 \ x_2]^T \in \Re^2$
- **Discrete inputs :** None
- **Continuous inputs :** None

- **Initial states :** q_0 and $x \in \Re^2 : x_1 \leq 0$
  All real numbers where the position of the ball is above the ground is valid as an initial state.
- **Continuous dynamics :** $\dot{x} = [x_2 \ -g]^T$
- **Invariant :** $Inv : q_0$ and $x \in \Re^2 : x_1 \geq 0$
  As previously mentioned, the discrete state stays invariant and the postion will naturally be above ground at all times.
- **Discrete dynamics :** $R(q_0, \{x : x_1 = 0 \wedge x_2 \leq 0\} = (q_0, (x_1, -kx_2))$
  When postioned in q_0 and x is defined as $x_1 = 0 \wedge x_2 \leq 0$ (notice the negative velocity requirement) the function returns the system to q_0 with x_1 unchanged and x_2 damped with a changed direction

The system is best visualized in it's corresponding graph:

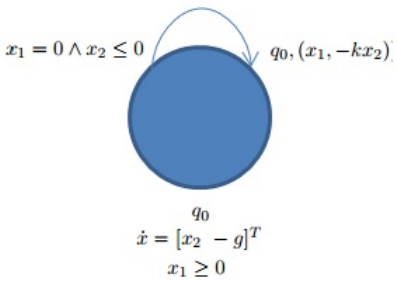

Figure 5: Bouncing ball hybrid automaton

Reading directly from the graph it may seem like it is possible to solve the system by using a logic operator in the form off:

```
1 if x(1) >= 0 && x(2) <= 0
2       x(2) = -k*x(2);
3 end
```

Most differential solvers iterates with the notation that the states are continuous at all time. With some solvers able to handle some types of discrete events. In this case, with a simple logic, the results are a very poor performance at the point where the discrete dynamics interfere with the states. Simulating a model in MATLAB (C.1.2) with the simple discrete logic results in:

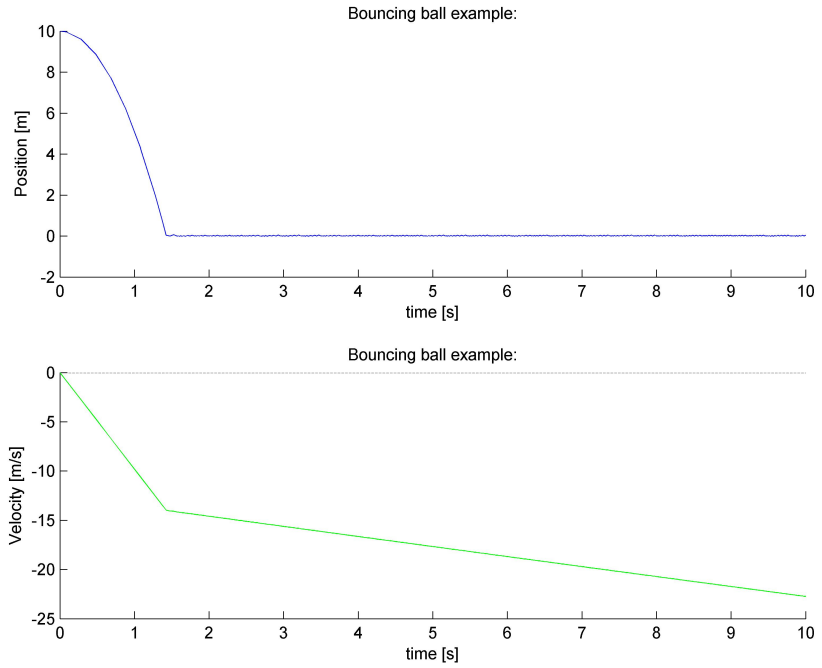

Figure 6: Bouncing ball hybrid automaton

The correct method for solving such systems is using solvers with *zero crossing detection*. These solvers obviously enough detects when a state crosses. In the bouncing ball example, when the position reaches zero, the whole differential solver is stopped. The output signal (the state at the stopping time) is manipulated according to the discrete dynamics and then used as the initial state, before restarting the differential solver. Utilizing the hybrid automaton modelling technique does not in it self solve the problem, as knowledge of zero crossing detection is required. But the hybrid automaton provides an overview of the system, letting the user know that when an event (hitting the ground) is detected, a discrete dynamic interfere with the states. Such behaviours may cause problems when utilizing a generic solver.

3.1.2 Example discrete state approach to control: Stabilizing an inverted pendulum

The following example displays how a qualitative modelling approach, together with the hybrid automaton modelling technique, unlocks new possibilities for system control. Introducing controller switching. An introduction to qualitative modelling is included in the following section as experienced based qualitative modelling also is an important aspect of the hazard and

operability procedure, elaborate at a later stage.

Qualitative modellling

In chemical engineering quantitative modelling has become the main tool for describing dynamic systems. Differential equations are established for given capacities and integrated for the desired time horizon. However, in some cases such a model may be considered overly extensive for it's purpose. If the system in question is a water tank with no drain, a single in flow will eventually fill the tank making it overflow. A quantitative model will provide detailed information about the level for every time step taken. A qualitative model can with this example be considered as common sense. A capacity with zero outflow and a non zero inflow is going to overflow. And the question is; will the detailed and computational more expensive quantitative model provide anything more than the common sense approach for it's given purpose?

Every model is incorrect, some are just suitable for it's given purpose, is a common expression when discussing qualitative models. Considering the filling tank system previously mentioned. Are the volume described correctly? Have the manufacturer actually accomplished the impossible and made a perfect cylinder, making the volume description correct? Or is the volume actually f.ex $40.034m^3$ and not $40.000m^3$? In comparison, the simplified quantitative model only establish the fact that the volume will increase until it at some point overflows. And without specifying the volume relation, this can be described as:

$$Volume = M^{+}(Level) \tag{15}$$

$M^{+}(Level)$ describes a positive function of level. When the level rises the volume rises, which unlike the quantitative function holds true for every capacity of any given size and shape. Quantitative modelling stands and falls on it's parameters and simplifications. And increasingly complicated systems is often trimmed by scaling factors and parameter fitting based on experiments. Acquiring these parameters can be time consuming, expensive and down right difficult. At this point an abstraction of such quantitative descriptions, in to a simple increasing function M^{+}, may seem a bit limited in terms of share usefulness. But when the required models purpose isn't fine real time optimization, but rather of use in a Hazop procedure (see section 4), the feedback of a qualitative model is sufficient for it's purpose, and this saves time and resources for the people involved.

Qualitative approach to hybrid automaton modelling. The inverted pendulum

The inverted pendulum is an often revisited control problem and in the research period for this thesis a specific paper by Kuipers and Ramamoorthy

[6] was found specifically interesting as it involved a qualitative approach to a hybrid system. Together with the learning learning tools published by Professor Claire Tomlin at Stanford University [5] a hybrid automaton approach to the inverted pendulum was performed. The inverted pendulum consists of a moving cart with an attached pendulum connected with a stiff axle. See figure 7.

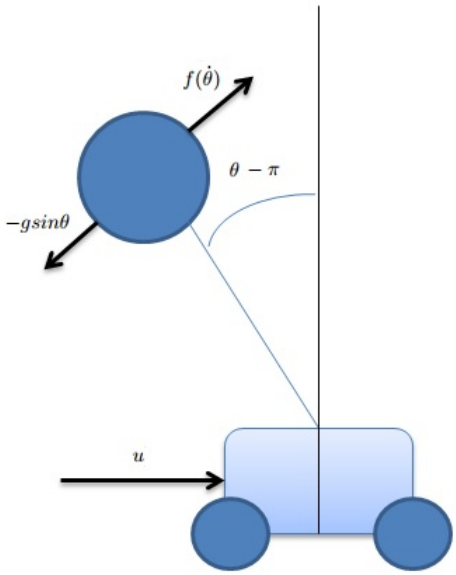

Figure 7: Sketch of the pendulum control problem

Where:

$$\gamma = \theta - \pi \tag{16}$$

The inflicting forces is that of the moving cart (the continuous input - u), $(-gsin(\theta))$ and the small amount of damping friction $f(\dot{\theta})$

Damped harmonic oscillators
The inverted pendulum is chosen as an example for it's close resemblance to the damped spring system, commonly used as an entry point to second order differential equations. The damped spring system consists of a mass connected vertically to a spring that establishes an equilibrium with the gravitational forces. Below the spring is a plate submerged in liquid, where the viscosity decides the level of damping. See figure 8.

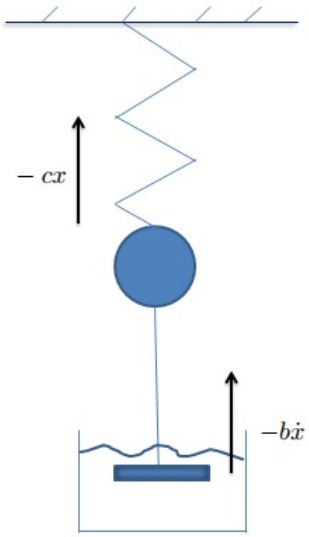

Figure 8: Sketch of a damped harmonic oscillator

The model is simplified by setting the equilibrium point to $x = 0$ and balancing the gravitational force with the spring resistance. Where the spring resistance is modelled by using Hook's law. The momentum balance then becomes:

$$m\ddot{x} = -b\dot{x} - cx \tag{17}$$

$$m\ddot{x} + b\dot{x} + cx = 0 \tag{18}$$

Now, the point of this section is to show the benefit of some times taking a qualitative modelling procedure. Equation 18 is used for entry level 2. order differential equations because of how it is easily solvable analytically. In other words, integrating the system can be done by ease. But instead of doing that the same system will instead be observed with a qualitative approach. Retaining the full complexity of the system. This is done by describing the damping and spring resistance with unspecified monotonic functions instead of the previous linear dependence:

$$\ddot{x} + f(\dot{x}) + g(x) = 0 \tag{19}$$

In the qualitative modelling section (section 3.1.2) the modelling technique was linked to reasoning. This very much still applies for the damped spring system. If the functions f and g where to be unlimited, it would not be possible to proceed and the model would remain generic and uninformative. To limit the functions reason is used . At $x = 0$ the damped harmonic

oscillator is set at equilibrium. This means that any displacement should experience a driving force towards the equilibrium, towards $x = 0$. In the model the restoring force working from the spring is defined moving upwards in the positive x direction. If the mass is displaced to $x = -1$ the spring will "restore" the system forcing acceleration in the positive direction. With this reasoning it's possible to determine that $sign(x) = sign(g(x))$ by studying equation 18. Similar reasoning can be made with f, at max velocity the damping is working against further acceleration. Since the state x and it's time derivatives contains direction information, if f' changes from positive to negative f changes from being a damping force to a amplifier. Therefore in the system in figure 8; $f' > 0$. In addition, as previously defined, the equilibrium is set at $x = 0$. Thie means that the only valuable set of f functions are the ones that includes $f(0) = 0$. In Kuipers and Ramamoorthy [6] the functions are defined in a similar fashion. Below follows a rewritten version with emphasis on what is regarded the most important for this thesis.

- **Definition 1: Only allowing the damping to be a continuous, smooth function over the valid range of x**
  Let the closed interval $[a, b]$ describe the velocity of the mass, which is a subset of $\Re$; $[a, b] \subseteq= \Re$. Then f extends the subset $f : [a, b] \to \Re$. The function is a reasonable description of the system if:

  1. f is continuous on $[a, b]$
  2. f is differentiable on $[a, b]$
  3. f has only a finite number of critical points in any bounded interval
  4. The one sided limits, approaching from inside the set, $\lim x \to a^+$ and $\lim x \to b^-$ exist

- **Definition 2: Making sure f always remain and damping force and is inactive at the equilibrium point**
  $f' > 0$ for any $f : [ab]$ and $f(0) = 0$

- **Definition 3: Making sure g always remain a restoring force and is inactive at the equilibrium point**
  $sign(g(x)) = sign(x)$. This also implies that $g'(0) > 0$.

- **Lemma 1** The qualitative DE

$$\ddot{x} + f(\dot{x}) + g(x) = 0 \tag{20}$$

With the defined f and g any trajectory within the bounded region will behave such that:

$$\lim \infty(x(t), \dot{x}(t)) = (0, 0) \tag{21}$$

- **Proof 1**

A familiar procedure to determining whether a system is stable, is by evaluating the eigenvalues. If these have negative real parts, Szidarovszky and Bahill [7] validates that the system is asymptotically stable. The question then arises, how to determine the eigenvalues with a qualitative model approach? The first step is to do a familiar rewrite of equation 34:

$$\dot{x_1} = x_2 \tag{22}$$
$$\dot{x_2} = -f(x_2) - g(x_1) \tag{23}$$

The system state space representation

$$\Delta\underline{\dot{x}} = \underline{\underline{A}}\Delta\underline{x} + \underline{\underline{B}}\Delta\underline{u} \tag{24}$$

For a non input system:

$$\Delta\underline{\dot{x}} = \underline{\underline{A}}\Delta\underline{x} \tag{25}$$

To acquire the A matrix a Taylor approximation is used. As $f(0) = 0$ and $g(0) = 0$ has been defined at $x = 0$ it seems like an appropriate reference point for the Taylor approximation

$$\underline{f}(\underline{x}) \approx \underline{f}(\underline{x}(0)) + \frac{\partial \underline{f}(\underline{x}}{\partial \underline{x}^T}\bigg|_{\underline{x}(0)} (\underline{x} - \underline{x}(0)) \tag{26}$$
$$\tag{27}$$

Calculating the jacobian:

$$\underline{\underline{J}}_A = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \frac{\partial \dot{x}_2}{\partial x_2} \end{bmatrix} \tag{28}$$

$$\underline{\underline{J}}_A = \begin{bmatrix} 0 & 1 \\ 0 - g'(0) & -f'(0) - 0 \end{bmatrix} \tag{29}$$

The A matrix can than be used for evaluating the eigenvalues:

$$det(\lambda I - A) = det(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -g'(0) & -f'(0) \end{bmatrix}) \tag{30}$$
$$= (\lambda)(\lambda + f'(0)) + g'(0) \tag{31}$$
$$= \lambda^2 + \lambda f'(0) + g'(0) \tag{32}$$

Which yields for the eigenvalues:

$$\lambda = \frac{-f'(0) \pm \sqrt{f'(0)^2 - 4g'(0)}}{2} \tag{33}$$

Since $f'(0) > 0$ the eigenvalues have negative real parts and the system is asymptotically stable towards (0.0) for any of the previously defined f's.

Connecting the damped harmonic oscillator with the inverted pendulum

Introducing negative damping, i.e. amplifing the oscillations:

- **Lemma 2** The qualitative DE with negative damping:

$$\ddot{x} + f(\dot{x}) + g(x) = 0 \tag{34}$$

Using a similar approach as in proof 1 yields:

$$\lambda = \frac{f'(0) \pm \sqrt{f'(0)^2 - 4g'(0)}}{2} \tag{35}$$

This is an unstable system, which in turn means that in time the states will leave the defined area.

Using the Lemmas to define different control strategies for the inverted pendulum

- q_2

The following procedure show how a hybrid automaton approach with the help of the two Lemmas can be used to control the inverted pendulum. This means taking the generic differential equation and tweak it so that it describes the inverted pendulum instead. First it's the positional resistance. As long as the pendulum don't hang in a position vertically straight down $\theta = 0$ the gravitational force will pull on the ball attached to the pendulum. But at the exact point of $\theta = \pi$ it rests on the pendulum axle.

$$\ddot{x} + f(\dot{x}) + g(x) = 0 \tag{36}$$

$$g(x) : ksin(\pi) \tag{37}$$

Where the k is used for differentiating between the function g and the gravitational acelleration k. Adding a controller action from the moving cart yields:

$$\ddot{\theta} + f(\dot{\theta}) + ksin(\pi) + u(\theta, \dot{\theta}) = 0 \tag{38}$$

Lemma 1 proves that this is stable at (0.0) when $u = 0$. However, the objective is to stabilize the system at $\theta = -\pi$, meaning that the current objective is to destabilize the system. Therefore using the knowledge of lemma 2 and modelling the input as a negative damping force:

$$u(\theta, \dot{\theta}) = -h(\dot{\theta}) \tag{39}$$

To instabilize the system this damping force have to override the natural damping of $f(\dot{\theta})$ so that the net damping force is negative:

$$f(\dot{\theta}) - h(\dot{\theta}) < 0 \tag{40}$$

This is where the hybrid automaton approach comes in to play. It is easy to see that it's beneficial to push the system out of the stable point by using an input force. But instability is not going to help when one want to stabilize at $\theta = \pi$. Therefore setting the approach above in to a discrete state which can be moved in and out of may be effective. Naming this region for the discrete state q_2 and setting the issue of stabilizing at $\theta = \pi$ as the discrete state q_1.

- q_1

By studying the lemmas it's quick to acknowledge that any input that leads to an unstable system is out of the question. It is therefore much more interesting to explore the possibility of mimicking lemma 1. To do this θ is first substituted with ϕ so that the stability region is the same as in lemma 1 (0.0). The damping is merely an effect of the angular velocity, e.g. air drag, regardless if the objective is to stabilize at θ or ϕ. This leads to the approach of trying to combine $ksin\phi$ and $u(\dot{\phi}, \phi)$ in to fitting the above definitions of $g(x)$:

$$sign(g(x)) = sign(x) \tag{41}$$

$$sign(-ksin\phi + u(\dot{\phi}, \phi)) = sign(\phi) \tag{42}$$

In other words; the control action has to surpass the negative force in effect from the gravitational field in addition to adding something extra. E.g $sign(\phi) > 0$, this means that $u(\dot{\phi}, \phi) > ksin(\phi)$. In addition faster convergence can be acquired by adding a control action based on the rotation velocity ($\dot{\phi}$) augmenting the natural damping of the system.

- q_3

Since the control action is likely to be limited it is a probability that the system can enter a situation where the pendulum spins faster than it's able to control. A system purely amplifying the natural damping is the last discrete state:

$$sign(g(x)) = sign(x) \tag{43}$$

$$u(\dot{\theta}, \theta)) = f_2(\dot{\theta}) \tag{44}$$

Establishing the discrete dynamics At this point three different discrete states have been established. Each of them has it's own unique control action. The last objective, following the model definition in 3.1, is to establish

the discrete dynamics bounding the region. The first boundary is set by the maximum velocity $\dot{\phi}_{max}$ the controller can handle. Kuipers and Ramamoorthy [6] propose a maximum conversion to potential energy to kinetic energy approach:

$$\frac{\phi^2}{\phi_{max}^2} + \frac{\dot{\phi}^2}{\dot{\phi}_{max}^2} \leq 1 \tag{45}$$

Equation 45 is therefore applicable as the guard leading in t th discrete state that will balance the pendulum in the upright position; q_1. For anything outside equation 45 a switch between the system that spins too fast and the system that is almost at rest in the downward position is needed. Kuipers and Ramamoorthy [6] propose a simple kinetic + potential energy comparison. If the total energy is less than the potential energy the system has when the pendulum is balanced at the upright position, there is a need for the state q_2, the pump state. The total energy of the system per mass is in the stable $\dot{\theta} = 0$ upright position is:

$$K_S + P_S = \frac{1}{2}\dot{\theta}^2 + \int_0^{\theta} ksin\theta \; d\theta = 2k \tag{46}$$

The observer to deterimine if the total energy per mass is lower than $2k$ may then be:

$$s(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^2 - k(1 + cos(\theta)) \tag{47}$$

If equation 47 is below 0 the total energy of the pendulum is not enough to reach the upright position. q_2 pump controller is needed. If the total energy is above 0 the pendulum will spin past the upright position, extra damping form state q_3 is needed. But note that this test is secondary as the balance region q_1 has priority and may overlap.

The Inverted Pendulum Hybrid Automaton
By assigning each of the different control actions to different discrete states the hybrid autamaton becomes:

- **Discrete states :** q_1, q_2, q_3
  Each state has a individual input performance
- **Continuous states :** $x = [\theta \; \dot{\theta}]^T \in [0 \; 2\pi]$
- **Discrete inputs :** None
- **Continuous inputs :**$q_1 : u = (c_{11} + k)(\theta - \pi) + c_{12}\dot{\theta}$, $q_2 : u = -(c + c_3)(\theta)$,$q_3 : u = -c_2\theta$
- **Initial states :** q_2 and $\theta \in [0 \; 2\pi]$ and $\dot{\theta} \in \Re$

- **Continuous dynamics :** $\dot{\dot{\theta}} + c\theta + ksin\theta + u(\dot{\theta}, \theta) = 0$
- **Invariant :** $Inv : Q = \{q_1, q_2, q_3\}$ and $\dot{\theta} \in \Re$ and $\theta \in [02\pi]$
  3 discrete states
- **Discrete dynamics :** $R(q_1, \{\frac{\phi^2}{\phi_m ax^2} + \frac{\dot{\phi}^2}{\dot{\phi}_m ax^2} > 1 \cap \frac{1}{2}\dot{\theta}^2 - k(1+cos(\theta)) < 0\} = (q_2, (\theta, \dot{\theta}))$ and $R(q_1, \{\frac{\phi^2}{\phi_m ax^2} + \frac{\dot{\phi}^2}{\dot{\phi}_m ax^2} > 1 \cap \frac{1}{2}\dot{\theta}^2 - k(1+cos(\theta)) > 0\} = (q_3, (\theta, \dot{\theta}))$

As usual the automaton is best visualized in a graph:

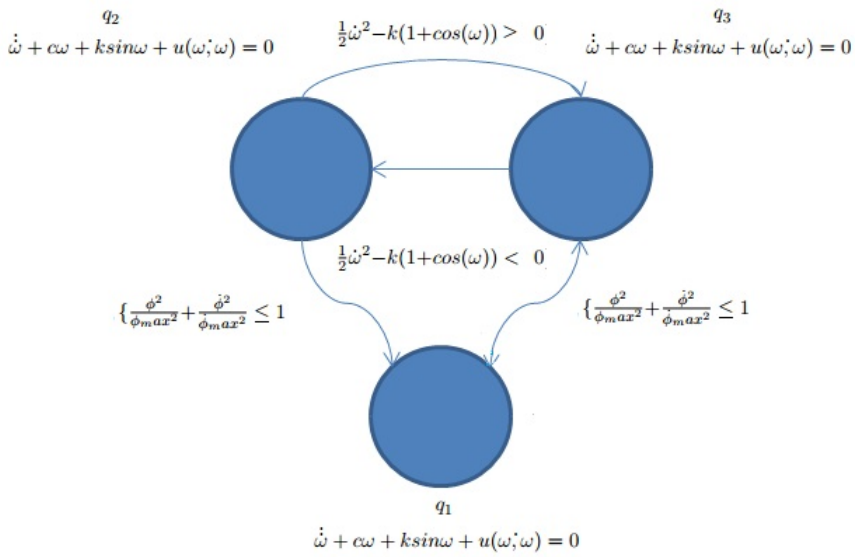

Figure 9: The inverted pendulum hybrid automaton

The automaton was then simulated using MATLAB, see appendix C.2.1 for code. Below follows the development of the continuous and discrete states:

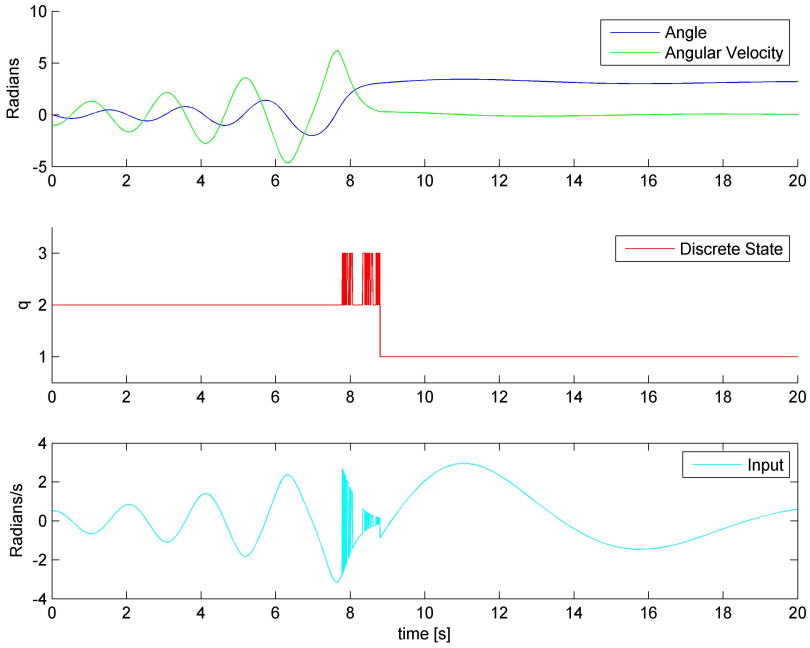

Figure 10: The inverted pendulum hybrid automaton simulation. Displaying the pendulum getting pumped up to the upright position while using different control strategies (discrete states).

3.1.3 Summary of hybrid automaton applications so far

The example of the bouncing ball and the inverted pendulum shows how a hybrid automaton approach may be used for solving modelling issues for vastly different hybrid systems. In the bouncing ball example it was clear that the continuous state development had an abrupt stop each time the ball hit the ground and a discrete dynamic had to be implemented. Leading to alteration of traditional solvers. In the inverted pendulum one control strategy was not optimal for the entire range the state variables where defined in. Establishing a discrete state for each control strategy resulted in efficient switching between the controllers. In the case study another implementation of the hybrid automaton will be presented. Here the continuous states will be transformed in to a set of discrete states, and this paves the way for what will hopefully be more optimized hazard identification.

4 Hazard and operability study - HAZOP

A Hazard and operability study (hazop) is method for systematically identifying and evaluating risk scenarios for a given system [8]. Be that risk for personnel operating on the system, equipment in the process or the surrounding environment. Even process efficiency deviations may be caught by use of hazop. Originally it was used within the chemical process industry, but has since seen application in several other systems. The method is a qualitative approach to risk assessment structured by a set of given guide words and should be carried out by a team consisting of competent personnel within several different disciplines.

In short the hazop method can be summarized as reversing the procedure where the cause of an incident is investigated, e.g. why did we experience dangerously high temperature in our reactor today? And instead ask what may happen if the cooling water flow stops? This makes sense since the whole objective of hazard study is to avoid said hazards or "effect" before they actually happen. But while a hazardous incident that already happened can leave evidence of a specific cause, like how the e.g. flow rate history can tell if the cooling water flow failed. Trying to operate in a reverse matter, where you try the imagine every hazard at a large system, a large number of scenarios may be plausible. Overlooking something with small or large consequences is certainly not impossible. To make sure a system can be operated in complete safety, even if the system is neglected for a while, a systematic and structured approach is called for to make sure that every possiblr scenario is extensively covered, so that a hazardous outcome is out of the question.

To make sure that all possible hazards are extensively covered a set of hazard causes are proposed in the hazop procedure. Or more correctly a set of guide-words is used to help the team define a cause (or more specific a process deviation). An example of a guide word may be "lower". Putting this word in an example where a team studies temperature in a reactor may set a scenario with a low reaction rate for the system. The team then assess whether such a scenario is logical, could this happen if the cooling fluid valve would become stuck in fully opened mode? If such a scenario seem logical the current existing safeguards is evaluated to determine if it is necessary to take action to improve the current situation. The use of guide-words are the most essential part of the hazop method, as the lay the foundation for systematical and structured evaluation of the system. Negating the possibilities of some hazardous scenarios being overlooked. The most common guide-words used in most analysis are:

- No (none, not)

- Higher (more)
- Lower (less)
- As well as (as well as additional activity)
- Part of
- Reverse
- Other than

In the industry a hazop is usually conducted in the design state after the proposed design has a working process simulation and flowsheets. In other words, it's performed in stage where a change in the design to eliminate risk still is a low cost issue. But hazop is also conducted at existing plants as possible hazardous scenarios may change over time with change in procedures and installation/change in existing equipment and personnel.

4.1 The procedure

Before getting started a information gathering phase is of the utmost importance. This includes, but is not limited to, gathering of: process models, design specifications, material safety data sheets, material properties, flowsheets, lists of essential personnel with additional expertise, process and instrumentation diagrams (P& ID's, both for existing equipment and planned) and operating procedures. An overview of each section of the system should also include a detailed explanation of the main purpose of the section. Including a specified operation range for the states.

Already at this point it's evident that the validity of the hazop is largely dependent on having qualified personnel and a sufficient documentation of high quality. This sets the bar for process engineers and manufacturers at the design state. Moving on to the procedure it self it can be summed up in the following steps:

- **Divide the system in to sections**
- **Select a node to study**
- **Describe the design intentions**
- **Select a process parameter**
- **Apply a HAZOP guide word**
- **Determine the cause in respect to the guide word**
- **Recommend actions**

In Crawleys et al. [8] a very handy flow diagram is supplied for ensuring a systematic approach throughout the whole process:

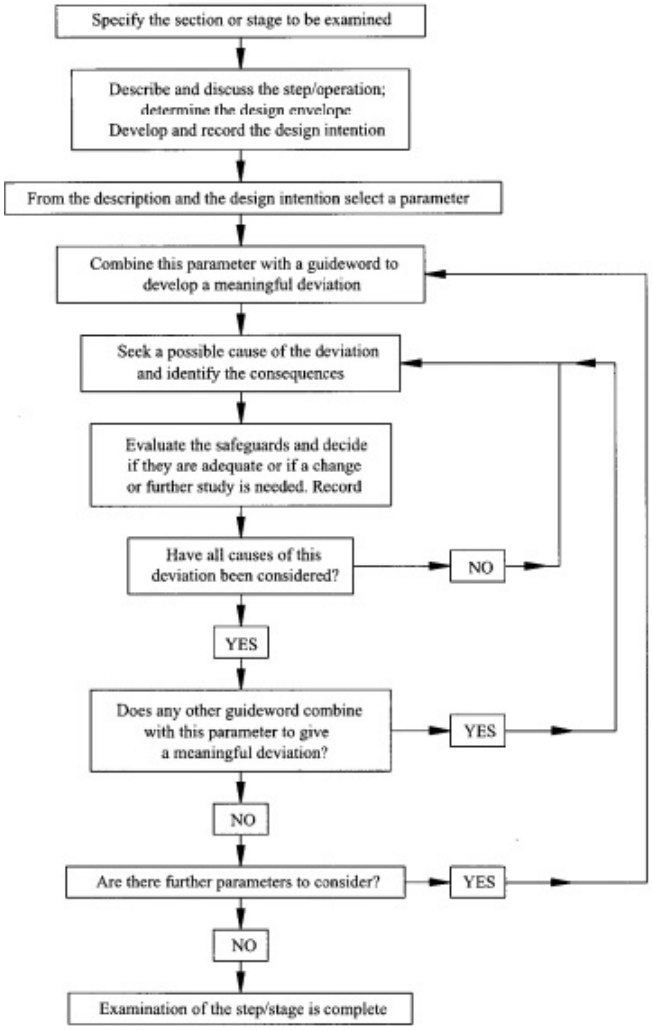

Figure 11: From Crawleys et al. [8] a Hazop procedure flow sheet

Going trough each step should be done with an emphasis on precise and comprehensive documentation. During this thesis the following categories will be listed in a table to make sure no steps are forgotten or undocumented:

- Unit number
- Parameter
- Guideword

- Deviation
- Cause
- Effect
- Existing protective system
- Action

The Hazop procedure is finished when every unit have been examined by utilizing every guideword and any hazards not covered by the existing safety system, has been evaluated and an action to improve the security has been proposed.

5 Case Study- Modelling and Automaton generation

In the case study the hybrid automaton is utilized in a more unconventional way. Until now the modelling procedure has been used to describe switching behavior, exemplified in section 3.1.1, where the direction of the falling ball is switched by use of event dynamics and also in section **??** where different control strategies is rapidly switched. In the case study a completely different approach is utilized. With reference to the procedure in section 3.1 the continuous dynamics are transferred in to discrete dynamics, i.e. the continuous states are now transferred in to being discrete states. At first glance this might seem counter intuitive. As the model will be less accurate in terms of state predictions and feedback. But later on the resulting hybrid automata will be utilized in a Hazop procedure and the usefulness of this approach will at the time hopefully be more transparent. For now an thorough explanation of the procedure and the resulting system is in order.

5.1 2 Tank Isothermic model

To clarify important concepts when it comes to establishing an hybrid automata of a continuous system, a 2D model is very useful. The system consists of two tanks which is connected to each other and an inlet pipe. See figure 12

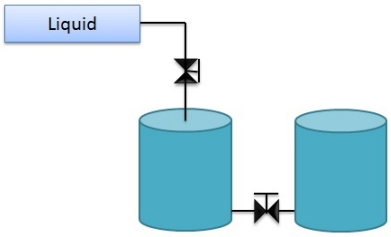

Figure 12: A simplified flowsheet displaying a simple 2 tank flow system

The system is the same system utilized in the paper by Preisig et al. [9]. The system inlet has no changing intensive quantities at any point in time. By defining the initial temperatures in the reservoir to be equal to the initial temperature in both tanks, as well as neglecting any heat loss to the environment, the system is isothermal over it's entirety. Thus any energy assessment may be disregarded and the complexity of the system is reduced to one mass balance for each capacity. The system topology is portrayed below:

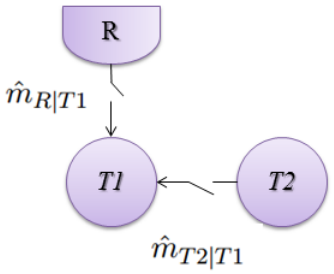

Figure 13: The topology for the 2 tank model

Where the size of the system is:

$$\dot{\underline{x}} = \underline{f}(t, \underline{x}, \underline{u}, \underline{d}) \qquad x \in \Re^2 \tag{48}$$

5.1.1 Modelling

Mass Balance
Conservation of mass is ensured when all capacities and every connecting flow are accounted for:

$$\frac{dM_{T1}}{dt} = \hat{m}_{R|T1} + \hat{m}_{T1|T2} \tag{49}$$

$$\frac{dM_{T2}}{dt} = -\hat{m}_{T1|T2} \tag{50}$$

A very general expression for the flow rates is used. Here the transfer rate is only dependent on the resistance in the pipes. In addition two disturbances are added for simulating pipe clogging or inlet valve malfunction. The conserved quantity will be set as the state:

$$\hat{x}_{R|T1} = (1 - d_2)u_1\Theta_2 \tag{51}$$

$$\hat{x}_{T1|T2} = (1 - d_1)\Theta_1(x_2 - x_1) \tag{52}$$

5.1.2 Defined system

After modelling the initial conditions and the system validity is decided the resulting simulation is based on:

- **Initial Conditions**:

$$\underline{x}(0) = \begin{bmatrix} x_{T1}(0) \\ x_{T2}(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \tag{53}$$

$$\underline{x}(0) \in \Re^2 \qquad x_n[=]kg \tag{54}$$

- **Default Input**:

$$\underline{u} = \begin{bmatrix} u_1 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{55}$$

$$u_n[=]on/off \;=\; 1/0 \quad for\ n = 1, 2, 3 \tag{56}$$

- **Parameters and constants**:

$$\underline{\Theta} = \begin{bmatrix} \Theta_1 \end{bmatrix} = \begin{bmatrix} .01 \end{bmatrix} \tag{57}$$

- **Invariant**:

$$0 \leq x_n \leq 4 \tag{58}$$

System overflow at $x_n = 4$

- **State equations**:

$$\frac{dx_1}{dt} = (1 - d_2)u_1\Theta_2 + (1 - d_1)\Theta_1(x_2 - x_1) \tag{59}$$

$$\frac{dx_2}{dt} = -(1 - d_1)\Theta_1(x_2 - x_1) \tag{60}$$

$$\tag{61}$$

5.1.3 State Space representation

Presenting the system in standard state space representation is necessary for later utilization of Heinz Preisig's 2-dimensional linear automaton script written in MATLAB(section C.3.1). The script is chosen because it's an easier to follow algorithm, while still retaining key elements needed for solving more complex systems. The script is limited to handling only a single input system. The disturbances is therefore neglected below.

$$\underline{\dot{x}} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}\underline{u} \tag{62}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\Theta_1 & \Theta_1 \\ \Theta_1 & -\Theta_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \Theta_2 \\ 0 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{63}$$

5.1.4 Continuous model simulations

To be able to compare the hybrid automaton representation with a conventional model, the system was simulated with continuous states as well. The differentials are solved by use of MATLAB and the script are portrayed in section C.3.2. Simulation of a filling process, with the default initial conditons $x(0) = [0\ 2]$ and input $u = [1\ 0\ 0]$, yields the following development of the states:

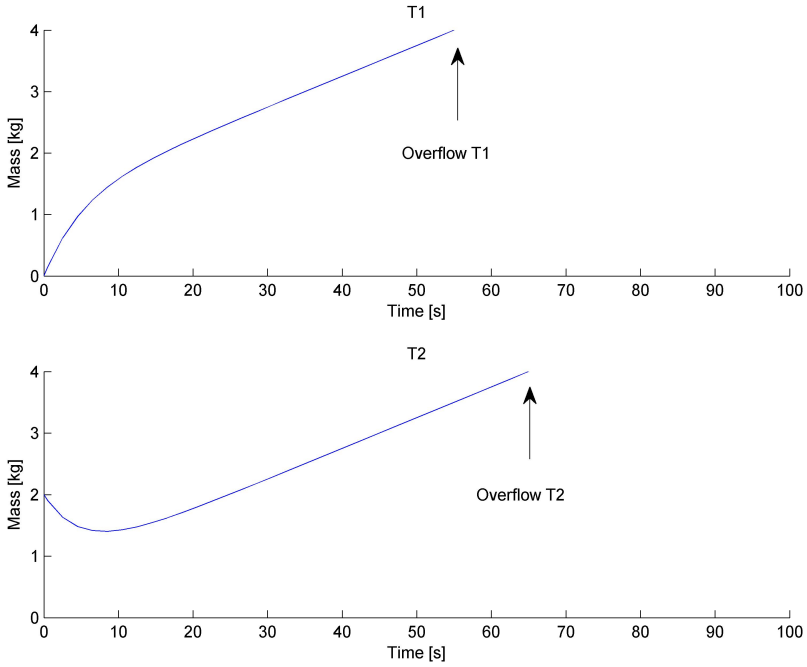

Figure 14: Simulating the mass content in the two tanks

Figure 14 is the standard way of portraying state development in a simulation. At $t = 0$ the half full tank 2 starts flowing in to tank 1 and at the same time it is getting filled. Then at approximately $t = 10$ the flow between the tanks changes direction and a equilibrium between the two tanks is established. The continued increase in level is due to the continued filling. Tank 2 "lag" behind tank 1 at the equilibrium line. The reason for this behavior is the added resistance due to the friction in the connection pipe. But the interesting part of the two tank system is not how it behaves. As it is not very complex. The reason it's used as an example for hybrid automaton modelling, is how a two state system can be easily visualized. To prepare the reader on the hybrid automaton representation the states are first presented in a state plot:

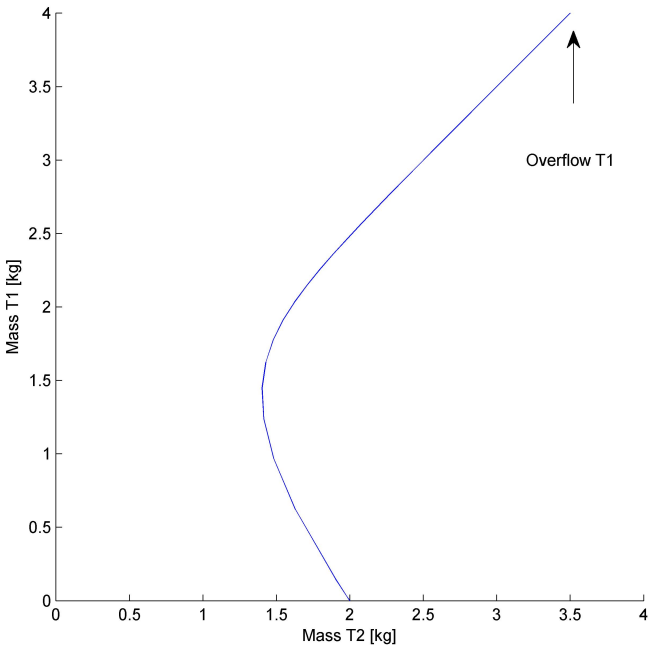

Figure 15: Simulating the mass content in the two tanks in a phase plot

In the phase plot the equilibrium between the two tanks are much more visible. Also it is easy to spot how the equilibrium is shifted towards T1 as T1 is on the y-axis. Below follows the creation of the hybrid automaton whitch will be compared to the phase plot.

5.1.5 The hybrid automaton for the two tank system

By following the procedure in section 3.1 it's easy to jump ahead and model the system straight forward. With one discrete states and two continuous states. But as previously mentioned, the objective now is to model the states discreetly. To do this a set of boundaries are introduced and are in accordance with Preisig et al. [9] defined as:

$$\mathcal{B}_i := \{\beta_i^1, \cdots, \beta_i^{a_i}, \cdots, \beta_i^{n_i}\} \tag{64}$$

Here the state in question is labeled with subscript i and the total number of boundaries for that specific state is labeled with n. Setting up a simple set used for the following automaton, visualized by utilizing MATLAB (script in C.3.3):

$$n_1 = n_2 = 4 \tag{65}$$

$$\mathcal{B}_1 = \mathcal{B}_2 = \{0, 1.333, 2.666, 4\} \tag{66}$$

In a 2-dimensional a phase plot the boundaries will span a set of quadrants:

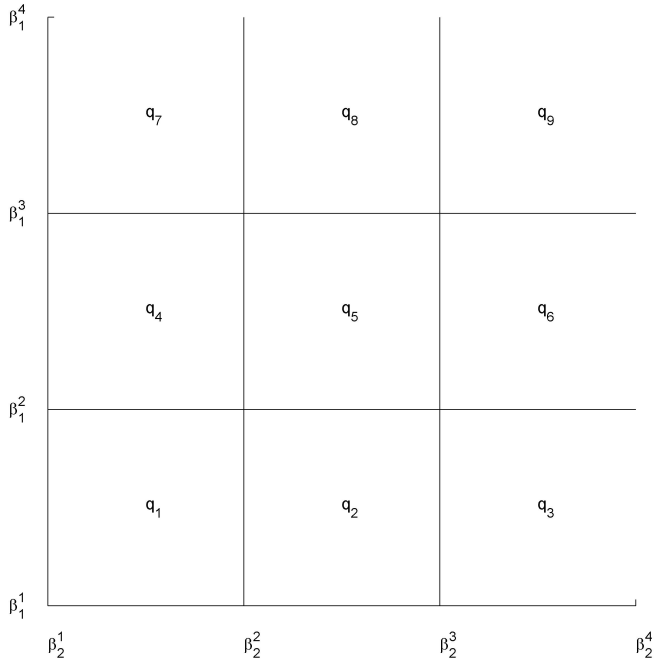

Figure 16: A two states, 4 equal boundaries hybrid automaton.

In figure 16 the discrete states have been defined as a quadrant in the automata spanned by the boundaries. For a 3-dimensional system the discrete states would form cubes and for any system above a set of hypercubes. In other words, the boundaries are mapping the continuous states in to it's respective discrete state. What is also clear is that mapping utilizing the boundaries in rising order, like figure 16, **no transfer may happen between discrete states if they are not adjacent, i.e. share at least one boundary plane.** Defining the hybrid automaton in accordance to the procedure in 3.1 yields:

- **Discrete states :** $q_k \in Q = [q_1, \cdots, q_9]$
  Each quadrant inside the set of boundaries are assigned to a discrete state
- **Continuous states :** $x = [x_1 \ x_2]^T \in \Re^2$
- **Discrete inputs :** $u = [u_1 \ d_1 \ d_2]^T$
- **Continuous inputs :** None

- **Initial states :** $q_k \in Q$ given by x and $x \in \Re^2 : 0 \leq x_n \leq 4$
- **Continuous dynamics :** $\dot{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}\underline{u}$
- **Invariant :** $Inv : q_k \in Q$ and $x \in \Re^2 : 0 \leq x_n \leq 4$ The invariant is defined by how the system will overflow at 4kg and must have non negative content.
- **Discrete dynamics :** $R(q_1, x) = (q_2, x)$ if $x_2 \geq \beta_2^2$ and $x_1 \leq \beta_1^2$. The continuous is under not influence by discrete dynamics. Only the discrete state change, with a similar rule set for the other states.

When tracking the previous continouos simulation, with the same initial conditions and inputs as in figure 14 and 15. The simulations hybrid automaton representation will start in the discrete state q_2, in accordance with the rules set by the boundaries, i.e. $x = [0\ 2]$ is mapped to the discrete state q_2. A script which continuously maps the development of the discrete states and present them visually was developed in section C.3.3.

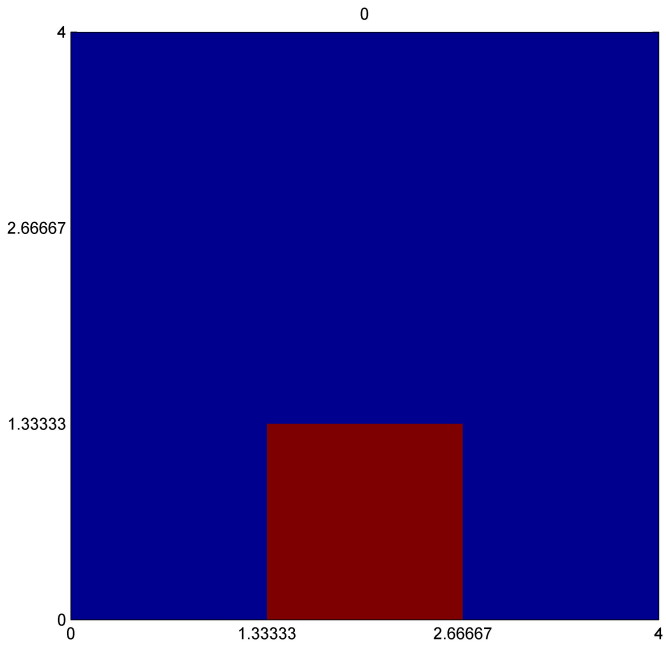

Figure 17: The hybrid automaton representation at $t = 0$

At $t = 7$ boundary β_1^2 is breached and the automaton moves from $q_2 \rightarrow q_5$, which can be verified by looking at figure 15.

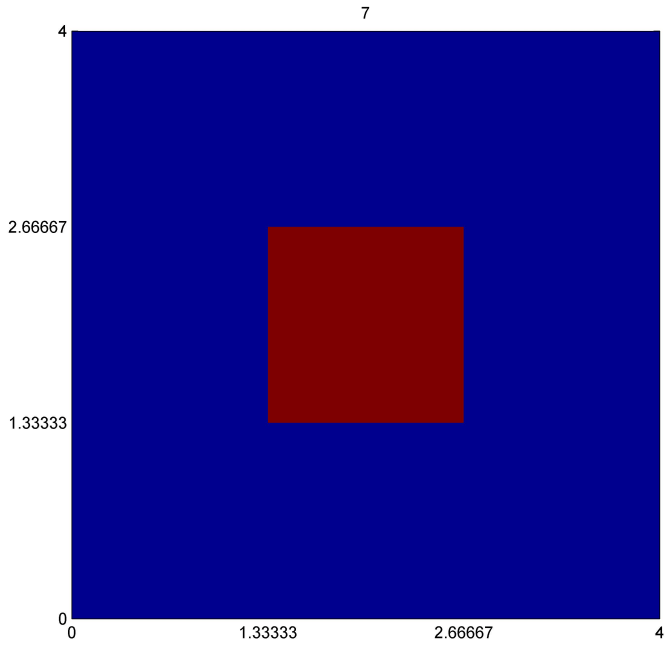

Figure 18: The hybrid automaton representation at $t = 7$

Further on the observed development until the invariant is breached is:

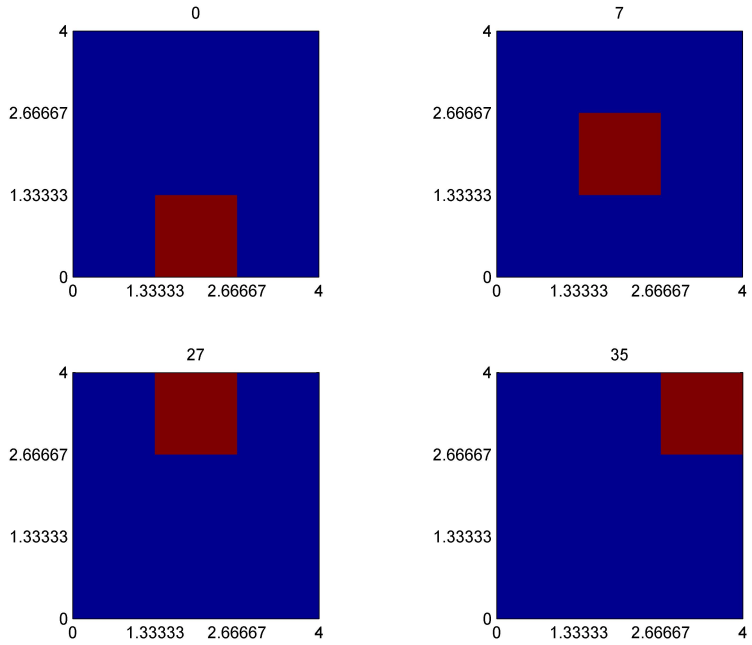

Figure 19: The hybrid automaton development before going out of bounds a $t = 51$

At this point the hybrid automaton does not provide any significant amount of information. In some cases it may be adequate, but for the current situation e.g. if one knows that the system is in q_5, it's still equally probable that tank 1 contains $1.4kg$ as $2.6kg$. The power of the tool is first visible when one consider the automaton in respect to the continuous system. As one extra look at 15 reveals that the path of the automaton was predetermined. The system would seek equilibrium and then follow the "equilibrium line". Meaning that it was only a set of discrete dynamics allowed by the continuous dynamics. The transfer from $q_5 \rightarrow q_2$ could at no circumstance happen. This leads to the some what obvious realization that based on the input and differential equations the state trajectory is predetermined.

As discussed in the section 2.1 a simulated state trajectory rarely (and at a small enough scale never) fits a real system trajectory. This is where the crudeness to the hybrid automaton actually becomes it's strength. E.g. say that tank 1 is empty and tank 2 is full and the pipe connecting them is open. The rate of transfer can easily be incorrectly modelled as maybe the pipe surface is more crude than anticipated. But one thing remains undeniable true. The water will flow from the full tank to the empty tank. This portrays

how an automaton is at the extreme points less dependent of rates. What is of interest is the direction of the discrete transfers.

In Preisig et al. [9] and by evaluating the continuous development it's possible to understand that for any quadrant completely below and out of contact of the equilibrium, i.e q_3 the time derivative of x_2 is negative as liquid flows from this tank and in to tank one to equalize the hydrostatic pressure. For the direction of transfer to change sign the trajectory of the state x_2 must go through the continuous state:

$$\dot{x}_1 = f_i(t, \underline{x}, \underline{u}) = 0 \tag{67}$$

For a system with an open pipe and inflow as previously modelled ($u = [1\ 0\ 0]$) the equilibrium surfaces for the two states are:

$$\frac{dx_1}{dt} = 0 = \Theta_2 - \Theta_1(x_2 - x_1) \tag{68}$$

$$x_1 = x_2 + \frac{\Theta_2}{\Theta_1} \tag{69}$$

$$\frac{dx_2}{dt} = 0 = \Theta_1(x_2 - x_1) \tag{70}$$

$$x_2 = x_1 \tag{71}$$

Therefore the theory is that for each single discrete state it should be possible to assign each continuous state with either $+, -$ or 0 to mark the possible transition. Where a plus would mean that in the current discrete state (or hypercube), this continuous state will look to increase it's value towards the equilibrium line and therefore cross it's discrete boundary. E.g. considering the 2 dimensional two tank system. With the same specifications as the simulated system the continuous x_1 state would be assign a *plus* in the discrete state q_2. Which allows for the system to transition over the boundary β_1^2, see figure 20

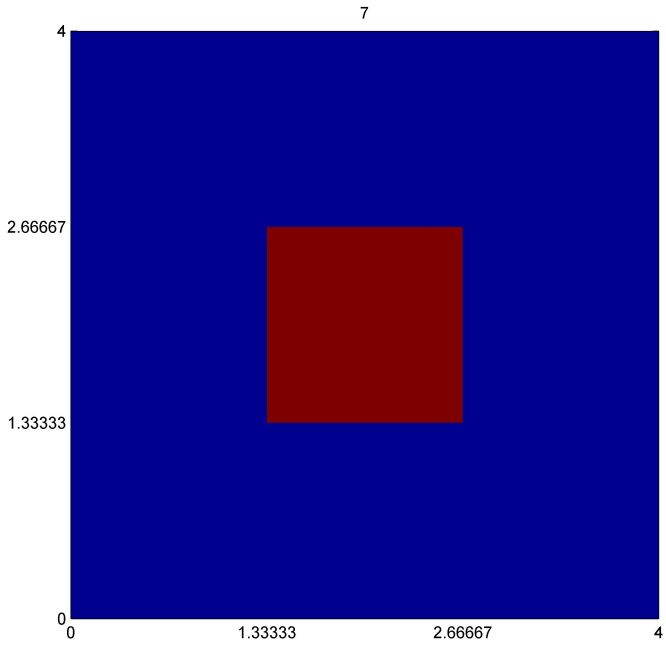

Figure 20: Marking the possible transition form q_2 for the state x_1. $+1$ represent that any transition as an effect of movement in x_1 happens in the positive direction for x_1

As mentioned before, the objective of the thesis is to streamline the hazop procedure using the technique displayed by hand above. Being able to predict every possible transition for every possible input and quickly at that. The effect of a manual cooling valve being left open in a large factory depends on the current states. If the reactor is "firing on all cylinders" then maybe all that happens is that the automated cooling system adjusts it self. But if this happened at the start up sequence then maybe the added cooling water would make it close to impossible to surpass the activation energy for the reactions. The point is, for a given input an unlimited amount of state trajectory is possible and trying to simulated a large number of them involves a great computational cost. This is where Phillips [1] the DEDS algorithm comes in to play. As a complete automaton with transition tables quickly can provide an overview of large number of state trajectories.

5.1.6 DEDS -Algorithm completing the automaton with transition tables

In this section the script *linauto2d.m* by Heinz preisig is utilized. Following this paragraph is a walkthrough of the algorithm that returns a full automaton, with transition tables for a two dimensional single input system.

Tuple representation of the hypercubes
Until this point q_k has been used as a notation for the discrete states. But for hybrid automaton models that discreetize continuous states the notation just don't provide sufficient information. When dealing with several states with several state boundaries the number of discreet states quickly rises. Keeping track of the discrete states in a multidimensional state space becomes to difficult. Therefore the boundaries are used for identifying the discrete states. In the two dimensional case the quadrant q_1 spanned by $\mathcal{B}_1^1, \mathcal{B}_2^1, \mathcal{B}_2^2, \mathcal{B}_2^2$ is named as the ordered list $(1, 1)$, or a tuple. The notation is easily expanded for larger dimensions, i.e. the first quadrant in a 3-D system is labeled as $(1, 1, 1)$ and so on. The entire two tank hybrid automaton is labeled as a tuple in 21 .

The Algorithm in pseudo code
Below follows a pseudo code of *linauto2d.m*. Different segments are labeled alphabetically and further explained below.

2 Dimensional linear automaton *linauto2d.m*

B| Establish a tuple representation of the automaton
C| FOR all continuous states
FIND next other state (x_j) impacting the current state (x_i)
Acquire the jacobi sign pattern of state x_j impacting state x_i
D| FOR all internal boundaries of the current state ($\mathcal{B}_n^i$)
EVALUATE state x_j at the cross over between the current and the equilibrium line
Pull result slightly apart in the direction of x_j to $dxmin$ and $dxmax$
FIND all boundaries less than the $dxmin$
E|EVALUATE x_i direction impact at boundary
F|IF x_i is moving to a lower boundary ($\mathcal{B}_i^{l-1}$), then:
FOR all internal boundaries from $dxmin$ to the last boundary of state j ($\mathcal{B}_j^q$)
Evaluate the transition table at the quadrant ($\mathcal{B}_j^q$), ($\mathcal{B}_i^{l-1}$) by using the jacobi direction.
FIND all boundaries larger than the $dxmax$ value
G|EVALUATE x_i direction impact at boundary
H|IF x_i remains inside it's current boundary ($\mathcal{B}_i^l$)
FOR all internal boundaries up to $dxmax$ ($\mathcal{B}_j^q$)
Evaluate the transition table at ($\mathcal{B}_m^q$), ($\mathcal{B}_n^l$) by using the jacobi direction.

Stepwise explanation of the algorithm:

- **A|Supplying the appropriate arguments**

First a system fitting the requirements, with two states and a single discrete input, is transformed in to the standard state space notation. The A and B matrix and the value for u are the first three arguments for *linauto2d.m*. The next argument is a set of boundaries defining the automaton representation, where row i contains the boundaries for state i. The total number of boundaries in each row is the last required input.

- **B|Establishing the automaton**

The amount of quadrants, or discrete states, are the number of boundaries for state $i-1$ times the number of boundaries for state $j-1$. The returned matrix is tuple representation of the automaton, where each row is one tuple/one discrete state.

- **C|Primary FOR loop (i)**

For all the states (which is two in this case) search for the next nondiagonal, nonzero element in the A matrix. Which translate in to finding the next dependable state (not it self). For example purposes the first iteration result

will be used as further reference. After iteration one two indexes is stored; $i = 1$ and $j = 2$, representing x_1 and x_2 respectively. Before moving in to the secondary for loop an importent calcualtion is made. The sign of $A(i, j)$ is evaluated and the direction of impact x_j has on x_i is identified.

- **D|Secondary FOR loop (l)**

For all the internal boundaries l of state i. Evaluate x_j at the cross-over between the boundary l and the equilibrium line. For the first iteration this translates in to evaluating:

$$\dot{x_1} = A(1,1)x_1 + A(1,2)x_2 + B(1,1)u \tag{72}$$

$$x_2 = \frac{-A(1,1)\mathcal{B}_1^2 + A(1,2) + B(1,1)u}{A(1,2)} \tag{73}$$

Pull the results slightly apart in the direction of x_j and save the results as $dxmin$ and $dxmax$.

- **E|Moving in the default negative direction**

From the boundary l apply the impact x_j has on x_i in the negative direction. E.g. for the first iteration, $i = 1, j = 2, l = 2$ and $sign(A(i, j)) = 1$. Applying the impact x_j has on x_i in the negative direction leads to a movement from 2 to 1 when the sign of $A(i, j)$ is positive. See figure 21.

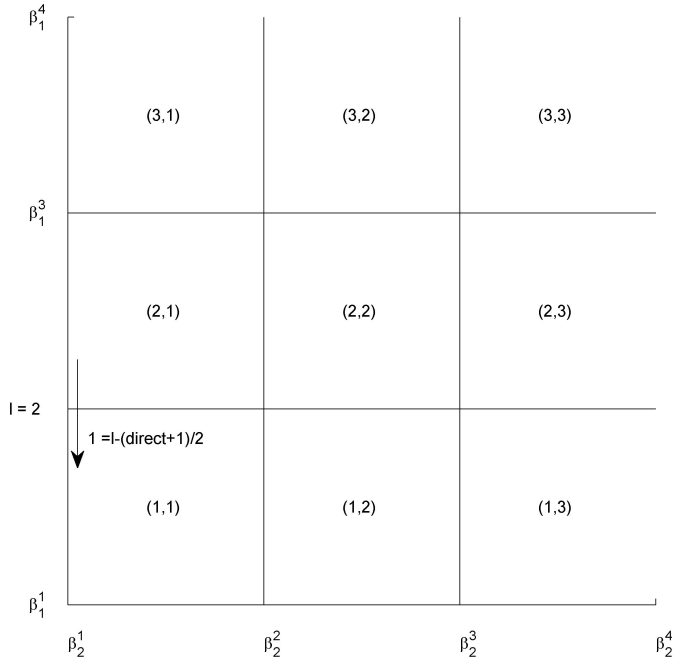

Figure 21: The result of evaluating $l-(direct+1)/2$ at the first iteration of linauto2d.m

Meaning that the current hypercube being evaluated has a tuple representation starting with 1. The second digit is undefined at this point in the algorithm.

- **F|Tertiary FOR loop (default negative direction (q))**

If the system still is inside it's defined outer boundaries or the sign of $sign(A(i,j))$ is positive, the tertiary for loop is initiated. The loop starts from $dxmin$ and takes the value of all larger internal boundaries for x_j. The positions tuple representation is identified and the transition for x_i is filled with the "adjusted" direction value. Where 0 is no change, 1 negative direction change, 2 positive direction. After the tertiary FOR loop has concluded, all hypercubes starting with 1 is assigned with the transition variable of x_i

- **G|Moving in the default positive direction**

This section performs the opposite of section E. Evaluating the expression $l--(direct+1)/2$. For the first iteration the result is 2, meaning that the hypercubes under evaluation has a tuple representation starting with 2.

- **H|Tertiary FOR loop (default positive direction (q))**

If the system remains inside it's defined outer boundaries or the sign of $sign(A(i,j))$ is negative q loops from $1 : dxmax$. For the first iteration this section only evaluates $(2,1)$.

5.1.7 Evaluating the results; the two tank automaton transition tables

The full transition table returned from linauto2d.m:

$$\left(\begin{array}{c|cc} & x_1 & x_2 \\ \hline (1,1) & 2 & 0 \\ (1,2) & 2 & 1 \\ (1,3) & 2 & 1 \\ (2,1) & 1 & 2 \\ (2,2) & 2 & 0 \\ (2,3) & 2 & 1 \\ (3,1) & 1 & 2 \\ (3,2) & 1 & 2 \\ (3,3) & 0 & 0 \end{array}\right) \tag{74}$$

To check it's viability and further improve understanding the following figure was created:

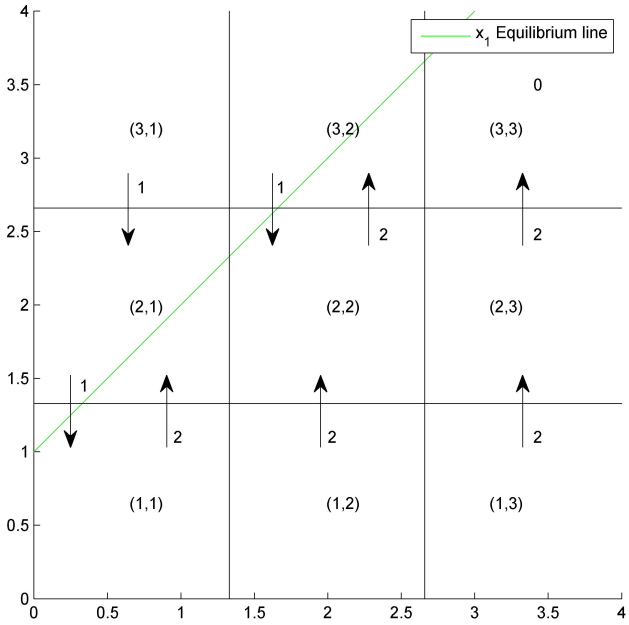

Figure 22: The transition table for x_1 for the two tank system represented by arrows

In figure 22 the transition table for x_1 is visualized, which is why every transition is vertical. As can be seen, the evaluation of the equilibrium line is very important for calculating the transition tables. As the transition values clearly show x_1 moving towards the equilibrium. Every transition of x_1 is towards the equilibrium line, validating the calculations.

5.2 3 Tank hot and cold liquid mixing

The 3 tank system has been chosen purposely for being able to clearly display the principles of using an automaton model in a Hazop analysis. As the modelling procedure is highly unprecedented, the complexity of the system is kept somewhat low. This is to ensure that that gradual improvements/bug fixes on the algorithm is possible within the given time frame. The system is also chosen so that objective 2 in the introduction may be fulfilled. Providing a solid foundation for future work expanding the utilization of hybrid automaton theory.

The system contains three tanks where one is filled with hot water, one with cold water and a third with a mixture of the previously mentioned tanks.

Meaning that the process unit is a direct heat exchanger. From this point and onwards the tanks will be specified as follows: hot water tank H, cold water tank C and mixing tank M. The initial temperatures of the tanks are:

$$T_H(0) = 80°C \qquad T_M(0) = 40°C \qquad T_C(0) = 20°C \tag{75}$$

For each tank there is an internal mixing unit keeping the temperatures uniform. Below follows a simplified flowsheet displaying the flow patterns in the system.

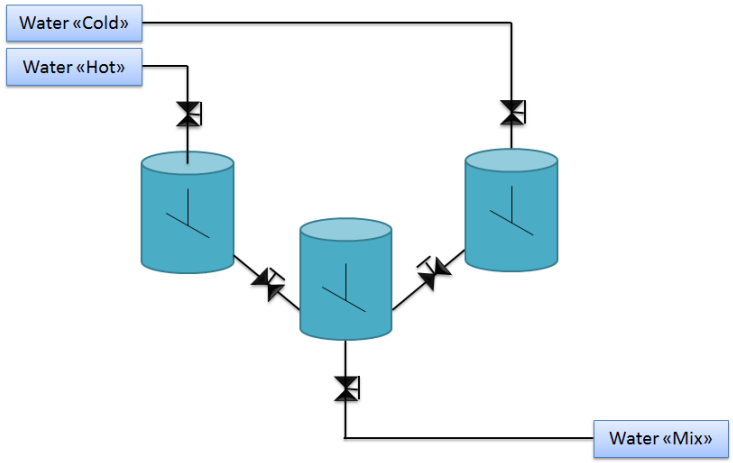

Figure 23: A simplified flowsheet displaying the 3 tank mixing system

5.2.1 Modelling

Modelling the tank is done accordingly to the technique displayed in section 2.1 as referenced in Preisig [3]. The three main tanks are described by lumps which indicates that they are assumed to have uniform properties throughout a given capacity, i.e. the CSTR simplification. Three reservoirs has been included in the model, portrayed as a half circle. From a modelling point of view they represent uniform intensive quantities, as well as endless capacity. Yielding no differentials. One reservoir supplies the system with hot water, another with cold and the last reservoir is the system environment that receives the outflow. The total system topology is therefore:

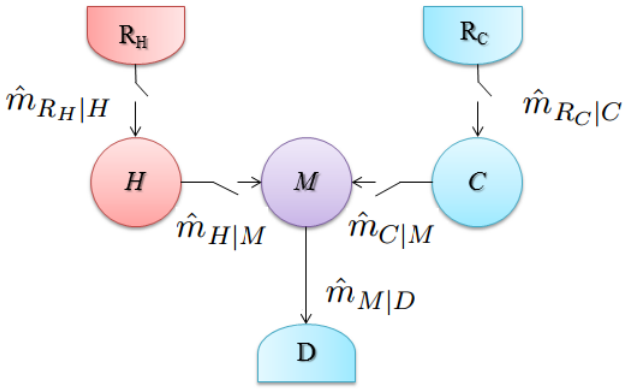

Figure 24: The topology for the 3 tank model

With a single species present and three capacities the model will include three mass balances and three energy balances

$$\dot{\underline{x}} = \underline{f}(t, \underline{x}, \underline{u}, \underline{d}) \tag{76}$$

Mass Balance
Conservation of mass is ensured when all capacities and every connecting flow are accounted for:

$$\frac{dM_H}{dt} = \hat{m}_{R_H|H} - \hat{m}_{H|M} \tag{77}$$
$$\frac{dM_C}{dt} = \hat{m}_{R_C|C} - \hat{m}_{C|M} \tag{78}$$
$$\frac{dM_M}{dt} = \hat{m}_{H|M} + \hat{m}_{C|M} - \hat{m}_{M|D} \tag{79}$$

A conventional method of describing liquid streams is by state conversion to volume flow and subsequently use of pressure difference as the driving force e.g:

$$\hat{m_k} = \rho_{H_2O}\hat{V}_{l|k} \tag{80}$$

Modelling the volume flow as dependent on the square of the pressure difference, multiplied with a factor dependent on the valve characteristic, is an equally conventional method of modelling the volume flow rate:

$$\hat{m_k} = \rho_{H_2O}(-c\sqrt{|p_k - p_l|}) \cdot sign(p_k - p_l) \tag{81}$$

Where the sign function is used for allowing change in the flow direction (negative pressure difference) without having to deal with complex numbers. The reason this procedure is included is to show how easy it is to take a misstep when modelling systems where high accuracy is required, e.g. in a temperature sensitive environment. While mass is a consistent quantity, by using simple conversation like the ones previously showed the consistency can no longer be guaranteed. Is it sufficient to allow the density to be constant in the operating area of the system? Can the safety assessment be performed with confidence? Or will maybe one single control question in an upcoming hazop analysis be answered differently by use of the model? What can such a mistake lead to?

The model is kept in mass, i.e. the mass is the chosen state, and the flow rate is modeled readily as:

$$\hat{x}_{R_H|H} = u_1\Theta_1 \tag{82}$$
$$\hat{x}_{R_C|C} = u_2\Theta_2 \tag{83}$$
$$\hat{x}_{H|M} = u_3(x_H - x_M) \tag{84}$$
$$\hat{x}_{C|M} = u_4(x_C - x_M) \tag{85}$$

For the last flow, the drain from the mixing tank, several options for controlled and uncontrolled flow is applicable . A simple proportional controller may look like

$$\hat{x}_{M|D} = p_1(x_M - x_{M_sp}) \tag{86}$$

Resulting in:

$$\frac{dx_H}{dt} = u_1\Theta_1 - u_3(x_H - x_M) \tag{87}$$
$$\frac{dx_C}{dt} = u_2\Theta_2 - u_4(x_C - x_M) \tag{88}$$
$$\frac{dx_M}{dt} = u_3(x_H - x_M) + u_4(x_C - x_M) - p_1u_5(x_M - x_{M_sp}) \tag{89}$$

Energy Balance
The energy balanced is modeled based on the topology as well. Resulting in three energy balances. To ensure consistency the model is constructed from the system total energy. Below follows some simplifications that applies to all three capacities:

$$E_S = U_S + K_S + P_S \tag{90}$$

Here the total energi (E), the conserved quantity, is separated in to internal energi (U), kinetic energy (K) and potential energy (P). The subscript S

is just as a notation for a temporary example system. Since the capacities in the 3 tank system is fixed in space over any reasonable time front, the change of potential and kinetic energy with time is zero. This simplifies the time derivatives to:

$$\frac{dE_S}{dt} = \frac{dU_S}{dt} \tag{91}$$

For illustration purposes any energy flow to the surroundings is neglected. The resulting energy balances are:

$$\frac{dU_H}{dt} = \hat{U}_{R_H|H} - \hat{U}_{H|M} + \hat{w}_{R_H|H} - \hat{w}_{H|M} \tag{92}$$

$$\frac{dU_C}{dt} = \hat{U}_{R_C|C} - \hat{U}_{C|M} + \hat{w}_{R_C|C} - \hat{w}_{C|M} \tag{93}$$

$$\frac{dU_M}{dt} = \hat{U}_{H|M} + \hat{U}_{C|M} - \hat{U}_{M|D} + \hat{w}_{H|M} + \hat{w}_{H|C} - \hat{w}_{M|D} \tag{94}$$

$$\tag{95}$$

To simplify the balances further the enthalpy is introduced:

$$H = U + pV \tag{96}$$

An open tank system can reasonably be assumed as having constant pressure. The time derivatives the develops as follows:

$$\frac{dH_S}{dt} = \frac{dU_S}{dt} + V_S\frac{dp_S}{dt} + p_S\frac{dV_S}{dt} \tag{97}$$

$$\frac{dH_S}{dt} = \frac{dU_S}{dt} + p_S\frac{dV_S}{dt} \tag{98}$$

$$\tag{99}$$

The work terms in the energy balances are defined as:

$$\hat{w}_{S|E} = p_S\frac{dV_S}{dt} \tag{100}$$

Which are work done on the environment when the volume changes. Introducing the enthalpy as the state function therefore eliminates the work segments in the energy balance:

$$\frac{dH_H}{dt} = \hat{H}_{R_H|H} - \hat{H}_{H|M} \tag{101}$$

$$\frac{dH_C}{dt} = \hat{H}_{R_C|C} - \hat{H}_{C|M} \tag{102}$$

$$\frac{dH_M}{dt} = \hat{H}_{H|M} + \hat{H}_{C|M} - \hat{H}_{M|D} \tag{103}$$

$$\tag{104}$$

The enthalpy transportation terms depends on the mass flow rate and the specific enthalpy of the substance. For a one species system this is defined as:

$$\hat{H_S} = \frac{\partial H_S}{\hat{n}_{S|E}} \hat{n}_{S|E} \tag{105}$$

$$\hat{H_S} = h_S(T_S)\hat{n}_{S|E} \tag{106}$$

The specific enthalpy is denoted h and is in turn dependent on the current temperature and the heat capacity of the substance. The heat capacity may also be dependent on the temperature in the current operating region. Thus resulting in the following definition:

$$\hat{H_S} = \int_{T_{ref}}^{T_S} \frac{\partial}{\partial T_S} \left(\frac{\partial H_S}{\partial \hat{n}_{S|E}} \right) dT \ \hat{n}_{S|E} \tag{107}$$

$$\hat{H_S} = \int_{T_{ref}}^{T_S} c_p(T) dT \ \hat{n}_{S|E} \tag{108}$$

$$\tag{109}$$

For initial testing a constant heat capacity may be sufficient. Equation 109 may then be further simplified to:

$$\hat{H_S} = c_p(T_S - T_{Ref})\hat{n}_{S|E} \tag{110}$$

To keep the system consistent the quite common state variable transformation $H \rightarrow T$ wil not be performed, i.e enthalpy is the chosen state. But the expression in equation 110 is used for the calculation of the initial enthalpy of the system. The reverse function is used for producing graphs which follows the temperature development. In the latter case the flow rates are substituted with the total mass of the capacity in question

$$T_S = \frac{H_S}{c_p N_S} + T_{Ref} \tag{111}$$

5.2.2 Defined system

The structured results after modelling in accordance with Preisig [3] displayed in section 2.1.1 is the following system:

- **Initial Conditions**:

$$\underline{x}(0) = \begin{bmatrix} x_H(0) \\ x_C(0) \\ x_M(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \underline{T}(0)[=]^\circ C = \begin{bmatrix} T_H(0) \\ T_C(0) \\ T_M(0) \end{bmatrix} = \begin{bmatrix} 20 \\ 80 \\ 25 \end{bmatrix} \tag{112}$$

$$\underline{x}(0) \in \Re^6 \qquad x_n[=]kg \ for \ n = 1,2,3 \qquad x_n[=]J \ for \ n = 4,5,6 \tag{113}$$

- **Default Input**:

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{114}$$

$$u_n[=]on/off \;=\; 1/0 \quad for\ n = 1,2,3,4 \qquad\qquad u_5[=]kg \tag{115}$$

- **Parameters and constants**:

$$\underline{\Theta} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \\ \Theta_4 \\ \Theta_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 10 \end{bmatrix} \qquad \underline{\gamma} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} h_H & = c_p(T_H - T_{Ref}) \\ h_C & = c_p(T_C - T_{Ref}) \end{bmatrix} \tag{116}$$

- **Algebraic equations**:

$$T_{Ref} = 20°C \tag{117}$$
$$H_S = c_p(T_S - T_{Ref})N_S \tag{118}$$
$$h_S = c_p(T_S - T_{Ref}) \tag{119}$$

- **State equations**:

$$\frac{dx_1}{dt} = u_1\Theta_1 - u_3\Theta_3(x_1 - x_3) \tag{120}$$
$$\frac{dx_2}{dt} = u_2\Theta_2 - u_4\Theta_4(x_2 - x_3) \tag{121}$$
$$\frac{dx_3}{dt} = u_3\Theta_3(x_1 - x_3) + u_4\Theta_4(x_2 - x_3) - \Theta_5(x_3 - u_5) \tag{122}$$
$$\frac{dx_4}{dt} = \gamma_1 u_1\Theta_1 - \frac{x_4}{x_1}u_3\Theta_3(x_1 - x_3) \tag{123}$$
$$\frac{dx_5}{dt} = \gamma_2 u_2\Theta_2 - \frac{x_5}{x_2}u_4\Theta_4(x_2 - x_3) \tag{124}$$
$$\frac{dx_6}{dt} = \frac{x_4}{x_1}u_3\Theta_3(x_1 - x_3) + \frac{x_5}{x_2}u_4\Theta_4(x_2 - x_3) - \frac{x_6}{x_3}u_5\Theta_5(x_3 - 2) \tag{125}$$

5.2.3 Aquiring the Jacobi incidence matrix

One of the necessary arguments for calling the script *nlinauto*, as well as being an essential part of calculating the transition tables, is the sign of the jacobian; $sign(\underline{\underline{J}}_A)$. Where the jacobian is the matrix:

$$\underline{\underline{J}}_A = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} & \frac{\partial \dot{x}_1}{\partial x_3} & \frac{\partial \dot{x}_1}{\partial x_4} & \frac{\partial \dot{x}_1}{\partial x_5} & \frac{\partial \dot{x}_1}{\partial x_6} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \frac{\partial \dot{x}_2}{\partial x_2} & \frac{\partial \dot{x}_2}{\partial x_3} & \frac{\partial \dot{x}_2}{\partial x_4} & \frac{\partial \dot{x}_2}{\partial x_5} & \frac{\partial \dot{x}_2}{\partial x_6} \\ \frac{\partial \dot{x}_3}{\partial x_1} & \frac{\partial \dot{x}_3}{\partial x_2} & \frac{\partial \dot{x}_3}{\partial x_3} & \frac{\partial \dot{x}_3}{\partial x_4} & \frac{\partial \dot{x}_3}{\partial x_5} & \frac{\partial \dot{x}_3}{\partial x_6} \\ \frac{\partial \dot{x}_4}{\partial x_1} & \frac{\partial \dot{x}_4}{\partial x_2} & \frac{\partial \dot{x}_4}{\partial x_3} & \frac{\partial \dot{x}_4}{\partial x_4} & \frac{\partial \dot{x}_4}{\partial x_5} & \frac{\partial \dot{x}_4}{\partial x_6} \\ \frac{\partial \dot{x}_5}{\partial x_1} & \frac{\partial \dot{x}_5}{\partial x_2} & \frac{\partial \dot{x}_5}{\partial x_3} & \frac{\partial \dot{x}_5}{\partial x_4} & \frac{\partial \dot{x}_5}{\partial x_5} & \frac{\partial \dot{x}_5}{\partial x_6} \\ \frac{\partial \dot{x}_6}{\partial x_1} & \frac{\partial \dot{x}_6}{\partial x_2} & \frac{\partial \dot{x}_6}{\partial x_3} & \frac{\partial \dot{x}_6}{\partial x_4} & \frac{\partial \dot{x}_6}{\partial x_5} & \frac{\partial \dot{x}_6}{\partial x_6} \end{bmatrix} \tag{126}$$

For the 3 tank system the jacobian with respect to the states is:

$$\underline{\underline{J}}_A = \begin{bmatrix} -u_3\Theta_3 & 0 & +u_3\Theta_3 & \cdots \\ 0 & -u_4\Theta_4 & +u_4\Theta_4 & \cdots \\ u_3\Theta_3 & u_4\Theta_4 & -u_3\Theta_3 - u_4\Theta_4 - \Theta_5 & \cdots \\ -\frac{x_4}{x_1^2}x_3u_3\Theta_3 & 0 & \frac{x_4}{x_1}u_3\Theta_3 & \cdots \\ 0 & -\frac{x_5}{x_2^2}x_3u_4\Theta_4 & \frac{x_5}{x_2}u_4\Theta_4 & \cdots \\ \frac{x_4}{x_1^2}x_3u_3\Theta_3 & \frac{x_5}{x_2^2}x_3u_4 & -\frac{x_4}{x_1}u_3\Theta_3 - \frac{x_5}{x_2}u_4\Theta_4 - \frac{x_6}{x_3^2}u_5\Theta_5 & \cdots \end{bmatrix}$$
$$\begin{bmatrix} \cdots & 0 & 0 & 0 \\ \cdots & 0 & 0 & 0 \\ \cdots & 0 & 0 & 0 \\ \cdots & -\frac{1}{x_1}u_3\Theta_3(x_1 - x_3) & 0 & 0 \\ \cdots & 0 & -\frac{1}{x_2}u_4\Theta_4(x_2 - x_3) & 0 \\ \cdots & \frac{1}{x_1}u_3\Theta_3(x_1 - x_3) & \frac{1}{x_2}u_4\Theta_4(x_2 - x_3) & -\frac{1}{x_3}\Theta_5(x_3 - u_5) \end{bmatrix} \tag{127}$$

The result of the sign function:

$$sign(\underline{\underline{J}}_A) = \begin{bmatrix} -1 & 0 & +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & +1 & -1 & 0 & 0 \\ 0 & -1 & +1 & 0 & -1 & 0 \\ 1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{128}$$

5.2.4 Dealing with negative flow rates in the enthalpy balances

The current model needs to be adjusted for negative flow rate situations that will occur in the transition table calculations. Currently the specific enthalpy of tank M and H is calculated and then multiplied to the flow rate. In the case of negative flow rate the specific enthalpy needs to be calculated

for tank M instead. Resulting in different continuous dynamics for different discrete states, i.e the discrete state $(1, 1, 3, x, x, x)$ has a different energy balance than i.e the discrete state $(3, 3, 1, x, x, x)$. Where the latter is the default model. A logic statement shifting between the continous dynamics is therefore included in the differential. See section C.4.1.

5.2.5 Utilizing Heinz Preisig Non-linear Hybrid Automaton generator

To generate the automaton transition tables for the nonlinear 6 states system, a modified version of Heinz Preisig nlinauto.m script was utilized. The modified version contains various bug fixes by the author and is presented in Appendix B section C.4.1.

Main algorithm differences from the script linauto2d.m
The script is not easily recognizable compared to the linear 2-D case. But the main concepts stays the same. Listed below are some key differences

- **Reducing the current state space**

When *nlinauto.m* intitate a search for a state that state x_i is dependent on it creates a minimized state space containing only the dependable states. This ensures that the problem of state explosion does not occur large system. If the algorithm did not do this then all the calculations would be performed for every independent state as well. And with the algorithm features three nested loops the number of calculations would increase exponentially. By searching for only the dependable states the algorithm will have a high performance even for large systems

- **Reducing the current input space**

Same strategies as reducing the state space. Calculations on inputs with no impact on the current state is wasted computational time.

Initializing nlinauto.m
The automaton generator requires 5 inputs, two which is normally supplied in the run file and three separate files supplying model information. See Appendix B section C.4.2 for the specific input files. Below follows a general explanation of the required input files:

- **Run script**

Specify automaton boundaries in cell arrays corresponding to the state, i.e. x_1 boundaries are located in a vector in $Boundaries\{1\}$. In addition specify the number of inputs and their possible values ([0 1] for discrete input). Read the system model, jacobian sign pattern and input pattern (See explanations below):

- **System model**

Insert the state equations in a single column vector. A single differential equation in each row. Alle states should be presented in vector form so that $x(1)$ calls state one and so on. All inputs should in a similar fashion be presented as $u(1)$.

- **Jacobian sign pattern**

The Jacobian matrix for the system needs to be evaluated and the sign pattern identified; $sign(J_A(n,m)) = J_{signA}(n,m)$. This is because the script works with direction impacts. A positive, i.e +1 sign at (n,m) means that increasing the state m increases state n.

- **Input sign pattern**

A sign matrix for which states are impacted by which input. The direction of impact is ,as previously mentioned, carried in the jacobian sign pattern and all sign are therefore +1 in this matrix. +1 sign at (n,m) means that input 1 has an effect on the change of state m.

Reading the results
Below follows a example of an automaton table generated by calling *nlin-auto.m*. NB! The transition table holds no value other than being an example for reading the output. The algorithm searches for all states dependable on the current state called x_s. In the example below x_1 depends only on x_3. The generated automaton with basis in x_1 is therefore two dimensional. The following output is received x_1, x_3:

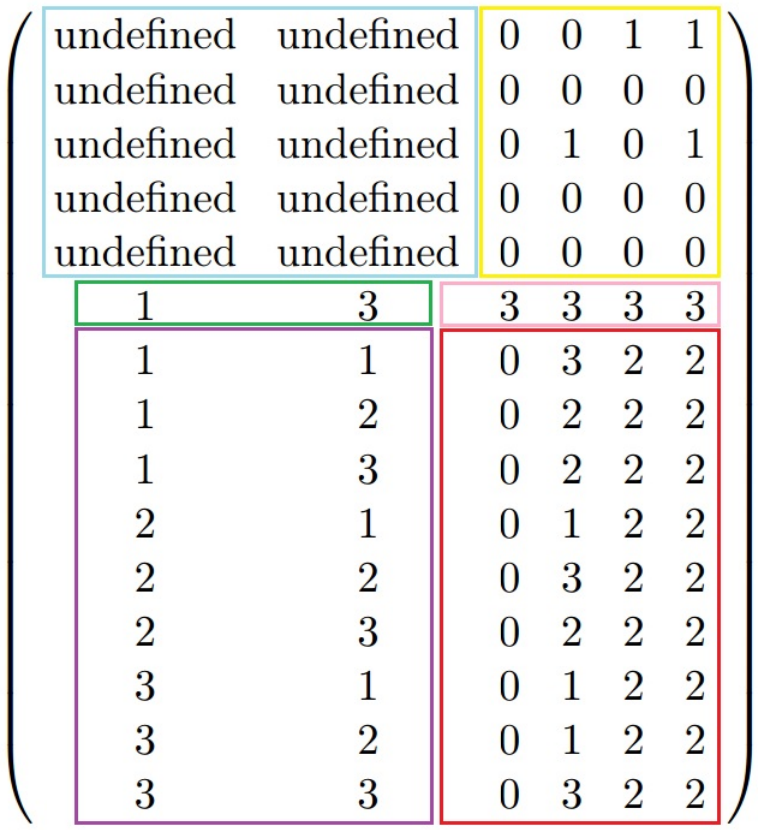

Figure 25: Automaton generated with H. Presig nlinauto.m for the 3 tank model

- **Top right corner (Teal)**

The top right $5x2$ corner is marked as undefined. This is merely a place holder to align the matrix data.

- **Top left corner (Orange)**

The top left $5x4$ is all combinations of the active set of inputs. Where the columns represents the current configuration. The inputs interfering with state x_1 is u_1 (top row, representing inlet stream to tank h) and u_3 (third row, representing the pipe connecting tank H and M).

- **Row 6 (Green)**

Row 6 is a marker, or a column labeler. E.g. $(6, 2) = 3$ means that all data below row 6 in column 3 is the tuple representation for state $3 := x_3$.

- **Row 6 (Pink)**

The dependent state and subject for the current equilibrium calculation.

- **Bottom right corner (Purple)**

The bottom right $9x2$ corner is a tuple representation of the current hypercube.

- **Bottom left corner (Red)**

The transition table for state x_1 in the x_1, x_3 two dimensional automaton. Where 1 is the negative direction, 2 positive and 3 either($\pm$).

Referring to appendix B for all 6 transition tables for the 3 tank system.

6 Case Study: Comparison between a traditional approach and utilizing the automaton model

6.1 Traditional hzop analysis of the 3 tank system

As mentioned in the introduction, the main purpose of this thesis is to take a hybrid automaton approach to the hazop procedure. This section is therefore devoted to a traditional hazop to be used as a reference for the automaton method later on.

6.1.1 Node selection and purpose identification

The primary step, as according to the flowsheet in figure 11, is to get an over view of the system and split it into appropriate nodes, to be examined by use of the guidewords (see section 4).

Node	Area description	Purpose
1	Hot water tank (N_H) including on/off inlet valve for stream $\hat{n}_{R_H\|H}$ and on/off outlet valve for stream $\hat{n}_{H\|M}$	A local, on site buffer for supplying hot water to the mixing tank. Done so that the process may be continued in the case of short supply problems. Transfer rate and direction between the hot tank and the mixing tank is in effect decided by the difference in hydrostatic pressure. Default flow direction is to the mixing tank. Intended temperature is $80^{\cdot}C$
2	Cold water tank (N_C) including on/off inlet valve for stream $\hat{n}_{R_C\|C}$ and on/off outlet valve for stream $\hat{n}_{C\|M}$	A local, on site buffer for supplying cold water to the mixing tank. Done so that the process may be continued in the case of short supply problems. Transfer rate and direction between the cold tank and the mixing tank is in effect decided by the difference in hydrostatic pressure. Default flow direction is to the mixing tank. Intended temperature is $20^{\cdot}C$
3	Water mixing tank (N_M) including level control connected to outlet stream $\hat{n}_{M\|D}$	Mixing a supply of hot and cold water to a controlled temperature decided by the "customer". Intended to work in the range of $20^{\cdot}C$ to $80^{\cdot}C$

6.1.2 Selection of guidewords and process parameters

The guidewords below are selected for fitting a liquid flow setting where temperature control is important

- Higher than
- Lower than
- Reverse

The states and the transfer rates, as well as the pressure is selected as process parameters. As these parameters are the only one with a realistic impact on the system. The system is not pressurized (open to the environment) so any pressure difference should be due to difference in hydrostatic pressure. Any hazardous incidents regarding pressure should therefore be covered by the level parameter.

- Flow rate
- Level
- Temperature
- Phase

6.1.3 Combining parameters and guidewords. Evaluating a possible deviations and a causes

Below is the Hazop form for tank H, the other Hazop forms are presented in appendix B.2

Nr	Parameter	Guideword	Deviation	Cause
1	Flow rate in	Higher than	Visable as a resulting high hot water flow in to mixer if the inlet stream is supposed to be closed.	Malfunction on/off valve at the inlet (leaking). When signal is "on" there is no visable deviation
2	Flow rate in	Lower than	Visable as a resulting low hot water flow in to mixer. Temperature fall in the mixer.	Pipe clogged at the inlet. When signal is off there is no deviation
3	Flow rate in	Reverse	Negative flow rate in inlet stream. Temperature drop in hot water tank, flow through tank H.	Mixer outlet clogged pipe. Higher pressure at the cold water side of the system.
4	Flow rate out	Higher than	High hot water flow in to mixer, low level in tank M	Malfunction level controller in mixer. Valve almost fully open
5	Flow rate out	Lower than	Low hot water flow in to mixer	Pipe clogged between mixer and hot water tank. Or clogged pipe/-malfunction level controller giving high evel in mixer.
6	Flow rate out	Reverse	Negative flow rate in the stream between mixer and hot water tank	Higher level in mixer. Level controller failing and possible clogged pipe. filling hot water tank.

Nr	Consequence	Protection	Action
1	Overflowing if system is supposed to be offline.	No protection against leaking.	Implement a high alarm on tank H.
2	Low temperature in mixer. Mixer not working according to intention	No protection against low flowrate.	Implement a low alarm on tank temperature tank M
3	Overflow cold water tank	No protection against negative flowrate.	Covered by action in number one
4	Potentially no consequence other than temperature shift in tank M.	No protection.	Deviation alarm temperature mixer M
5	Overflow hot water tank	No protection	Coverd by action in number one
6	Low temperature in hot water tank. Failed level control in mixer.	No protection against negative flowrate.	Implement a high alarm on level tank M

Nr	Parameter	Guideword	Deviation	Cause
7	Level	Higher than	High level in hot water tank	Clogged outlet/malfunction level controller in mixer leading to high level mixer
8	Level	Lower than	Low level in hot water tank	Clogged inlet or malfunction level controller in mixer leading to low level mixer
9	Temperature	Higher than	Higher temperature in hot water tank than normal operation/intention	deviation in inlet stream
10	Temperature	Lower than	Lower temperature in hot water tank than normal operation/intention	Deviation in inlet stream or reverse flow rate
11	Phase	Higher than	Level drop and flow rate drops	Fire in the plant
12	Phase	Lower than	Flow rate drop	Outside temperature significantly below freezing

Nr	Consequence	Protection	Action
7	Possibility of overflowing	No protection.	Implement a high alarm on tank H.
8	Possibility of emptying	No protection.	Implement a low level alarm
9	Operation deviation. Stream from mixer not meeting requirements.	No protection.	High temperature alarm tank H. Or temperature controller
10	Operation deviation. Stream from mixer not meeting requirements.	No protection.	Low temperature alarm tank H. Or temperature controller
11	Damage to equipment plant. Outlet stream not meeting requirements	Fire safet is assumed to be according to local law.	No action
12	Overflowing. Damage to equipment. Outlet stream blocked	No protection against negative flowrate.	If outside, isolate

Summary of recommended actions for mass based parameters tank H (flow, level):

1. Low alarm tank H
2. High alarm tank H
3. High alarm tank M
4. Low alarm tank M
5. Temperature deviation alarm tank M

6.1.4 Summary of proposed actions by utilizing the Hazop procedure

After hazop evaluation of all three tanks the following actions are recommended for the entire system:

Tank H

1. Low level alarm tank H
2. High level alarm tank H
3. Low temperature alarm tank H
4. High temperature alarm tank H

Tank C

1. High level alarm tank C
2. Low level alarm tank C
3. Low temperature alarm tank C
4. High temperature alarm tank C

Tank M

1. Low level alarm tank M
2. High level alarm tank M
3. Deviation alarm tank M
4. Low temperature alarm tank M
5. High temperature alarm tank M

Flows

1. Deviation alarm $\hat{m}_{R_H|H}$
2. Deviation alarm $\hat{m}_{R_C|C}$
3. Deviation alarm $\hat{m}_{M|D}$

6.2 Hazop analysis utilizing the 3 tank automaton

This section is in reality a test of the hypoteses:

Hypothesis 1: *Any guideword/parameter combination can be swapped with an evaluation of the hybrid automaton transition table*

Following hypothesis 1 is a hypothesis regarding how to locate which transitions in the transition tables that leads to a potential hazard.

Hypothesis 2: *If the automaton is generated over the boundaries spanning the safe operation domain. Any operational hazards must be visible in the transition table as a possible transition out of the domain*

A implication of hypothesis 2 is that all possible deviations must be presented in the inputs. I.e the vector containing all inputs u must also contain all disturbances d. The other result of hypothesis 2 is that for the 2-D $3x3$ automaton as seen in section 16, the only discrete states of interest are the one spanning the frame a long the outer boundaries $(1,1) \rightarrow (3,1) \rightarrow (3,3) \rightarrow (1,3) \rightarrow (1,1)$ i.e. the $(2,2)$ quadrant is not of interest. Where the "hazardous transitions" depends on the pathway a long the frame. In other words, a potential hazard is connected to a situation where the tank overfills or empties. For the path $(3,1) \rightarrow (3,3)$ the outer boundary that may be crossed is $\mathcal{B}_1^4$, the capacity limit of tank 1. And any crossing out of the outer constraint must therfore be in the positive direction. Meaning that if the transition table for x_1 at any $(3,k)$, where k is the sequence: $1, \cdots, n_{\mathcal{B}}$is 2 a potential hazard is identified.

6.2.1 Evaluating the hybrid automaton for tank H

As this is an explanatory thesis with an emphasis on being easy to follow the hybrid automaton transition table for tank H is displayed below. For the other tables see appendix B.

$$\begin{pmatrix} \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 1 & 0 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ 1 & 3 & 3 & 3 & 3 & 3 \\ 1 & 1 & 0 & 3 & 2 & 3 \\ 1 & 2 & 0 & 2 & 2 & 2 \\ 1 & 3 & 0 & 2 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 2 & 2 & 0 & 3 & 2 & 3 \\ 2 & 3 & 0 & 2 & 2 & 2 \\ 3 & 1 & 0 & 1 & 2 & 1 \\ 3 & 2 & 0 & 1 & 2 & 3 \\ 3 & 3 & 0 & 3 & 2 & 3 \end{pmatrix} \tag{129}$$

State x_1 outer boundaries can possibly be crossed at $(1, x)$ and $(3, x)$. In $(1, x)$ negative transition is searched for, i.e emptying tank H. For $(x, 3)$ a positive transition is searched for, or a situation where tank H overflows.

Nr	Input pattern	Tuple	Hazardous transition	Action
1	$[0\ 0\ 1\ 0\ 0]^T$	(1,1)	3: Possibility of emptying tank H. Reflect what happens if inlet flow $\hat{m}_{R_H\|H}$fails to open.	Low alarm tank H/low flow alarm $\hat{m}_{R_H\|H}$
2	$[0\ 0\ 1\ 0\ 0]^T$	(3,3)	3: Possibility of overfilling tank H, an effect of overfilling tank M. Reflect what happens if outlet flow $\hat{m}_{M\|D}$fails to open.	High alarm tank H/High alarm tank M or low flow alarm $\hat{m}_{M\|D}$
3	$[1\ 0\ 0\ 0\ 0]^T$	(3,x)	2: Trajectory in the direction of overfilling tank H. Reflect what happens if flow $\hat{m}_{H\|M}$ fails to open.	High alarm tank H or low flow alarm $\hat{m}_{H\|M}$
4	$[1\ 0\ 1\ 0\ 0]^T$	(3,2) (3,3)	3: Trajectory in the direction of overfilling tank H. Depends on level in tank M	High alarm tank M
5	$[1\ 0\ 1\ 0\ 0]^T$	(1,1)	3: Possibility of emptying tank H. Dependent on tank M Level	Low alarm tank M

From this point the automaton models grows in size. And manual evaluation is at best as time consuming as regular Hazop. A search script to automatically catch dangerous transitions have therefore been written, see section C.6. The output of the script can be verified as the matrix below contains the same information as the manual version above:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 \\ 1 & 0 & 0 & 0 & 0 & 3 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 & 3 & 2 & 2 \\ 1 & 0 & 0 & 0 & 0 & 3 & 3 & 2 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 3 \\ 1 & 0 & 1 & 0 & 0 & 3 & 2 & 3 \\ 1 & 0 & 1 & 0 & 0 & 3 & 3 & 3 \end{pmatrix} \tag{130}$$

Each row in matrix represented an identified hazardous transition. The first data represent the system input, followed by the tuple representation and the transition variable in the last column.

Summary of total recommended actions: The following list displays

the total recommended actions. This is fairly automated. If x_1 can transition out of the outer constraint then a high and low level

1. Low level alarm tank H
2. High level alarm tank H
3. Low temperature alarm tank H
4. High temperature alarm tank H

Tank C

1. High level alarm tank C
2. Low level alarm tank C
3. Low temperature alarm tank C
4. High temperature alarm tank C

Tank M

1. Low level alarm tank M
2. High level alarm tank M
3. Low temperature alarm tank M
4. High temperature alarm tank M

6.2.2 Hybrid automaton hazop performance

For the tank H hybrid automaton several 3's are identified as hazardous transition. However, those values follows a specific pattern. Every "3" is completely dependent on how the level in tank M evolves. And in effect, if tank M operates safely (level is regulated to match the set point) then a high level in tank M will be reduced. This will in effect also reduce the level in tank H.

A number of transitions are similar. In the case of the flow $\hat{m}_{H|M}$ is closed the state of tank M does not impact the transition table at all. The $(3, x)$ results are therefore a single potential hazard. For larger systems the number of hazardous transitions can become quite large, this is due to the possible number of temporary independent states, i.e. states that are disconnected due to the current input configuration. At the bottom of C.6 a contractor is written. The reduced transition matrix ignores the other states and only accounts for unique hazardous transition for the state in question. The reduced result returns the following information:

Mass balances

Tank	Nr. of unique hazardous transitions
H	5
C	5
M	10

Energy balances

Tank	Nr. of unique hazardous transitions
H	5
C	4
M	15

At first glance the results seems reasonable. The first indication lies within the number of unique hazards in the mass balances for tank H and tank C. Their respective mass balances are completely equal, something that is reflected in the number of unique hazards. At the same time the number for tank C is less than the number for tank H when it comes to the energy balances. This also makes sense since the input to tank C has a considerably lower specific enthalpy, making it impossible to cross the highest boundary without the possibility of negative flowrate from tank M to tank C. This is verified be comparing the output for unique hazardous transitions when utilizing script *autohaz.m*:

Unique hazardous transitions energy balance tank H

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 3 \\ 1 & 0 & 0 & 0 & 0 & 3 & 2 \\ 1 & 0 & 1 & 0 & 0 & 1 & 3 \\ 1 & 0 & 1 & 0 & 0 & 3 & 3 \end{pmatrix} \tag{131}$$

Unique hazardous transitions energy balance tank C

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 3 \\ 0 & 1 & 0 & 1 & 0 & 1 & 3 \\ 0 & 1 & 0 & 1 & 0 & 3 & 3 \end{pmatrix} \tag{132}$$

Separating the unique transition is the pure input configuration for tank H. Here the outer boundary for enthalpy will be crossed. The same configuration will not be considered hazardous in terms of enthalpy for the cold water tank. An overfill will be spotted by the mass balance transition table. But the temperature is not high enough to result in a violation of the enthalpy boundary as well.

6.2.3 Hybrid automaton script limitations

To further test the *nlinauto.m* script some additional models were tested. One of those was a state variable transformation from enthalpy to temperature. See section D.1 for further elaboration. If the script is to be viable for a test study on a real plant, the possibility of successfully perform a state variable transfer might be important for a successfull hazop study. Since the total enthalpy is a function of the total mass, the enthalpy transition table may not catch some temperature specific hazards. Since high enthalpy can be both a full cold tank, as well as a hot low level tank. Certain process have internal and outer hazard boundaries for temperature , yielding a need for temperature specific transition tables. Performing a state transfer to temperature in the three tank system reveals a sever script limitation:

State variable transfer from enthalpy to temperature reduces the number of dependable states

Since temperature is an intensive quantity, it is not directly dependent of any outflows. This reduces the number of dependable states, leading to a situation where the transition tables for both tank H and tank C, becomes 1-Dimensional. The current version of *nlinauto.m* have been found to not being able to handle these kind of differentials. In it self this is not a huge issue, since 1 dimensional transitions are easy to calculate after locating the equilibrium point. Manual evaluation is therefore a possibility. However, 1-D transitions is not uncommon in the process industry. The level in a buffer tank is usually only dependent on it self by utilizing a controller. And there is seldom cases like the three tank system, where tanks are situated on the same horizontal level. Allowing for reverse flow rates. If the script crashes for every 1-D transition table it comes across the complications will simply outweigh the benefits of automating the hazop procedure.

7 Discussion

7.1 Hybrid automaton modelling script *nlinauto.m*

Over the course of this thesis the hybrid automaton modelling approach has been introduced in different settings. Where the main focus lies within discretizing the state space in to a set of hypercubes. This allows for crude evaluation of net state trajectories for the entire invariant state space. Leaving the outer constraint of the state space is often synonymous with hazard. As an outer constraint may be the volume of a capacity or the temperature where side reaction may start to be significant. By searching along the outer constraints, plausible hazards have been showed to be identified fast for a wide number of configurations.

The non linear script for estimating hybrid automaton transition tables have proven to work quite fast. As a series of nested FOR loops, each additional state increases the number of calculation exponentially. And therefore a large number of states could potentially result in a very high computational cost. By continually evaluating each transition table only for the dependable state x_d the number of calculation always stays under control. Remember that the algorithm is absolutely oblivious to the state development for it's adjecent hypercubes which do not lie in the direction of the dependable state x_d, since it in fact calculates the impact the dependable state has on the state in question x_s between every boundary of the dependable state. In the three tank system this means that for tank M, the influence tank H have is calculated for a high, medium and low level in the tank.

Some bug fixes was needed to make the non linear automaton script to properly work. Such alteration always posseses a risk of resulting in new additional bugs. By evaluating the set of transition tables in B the results seems very reasonable for the mass balances. The first indicator is the purely positive transition for the single inflow, no outflow, for tank H and C. This is obviously a reasonable result, as filling a tank without removing anything from it will at some point lead to a situation where the tank overflows. The second indicator is the single outflow configuration. Here the dependence on the level of tank M is clearly visible. At low tank M levels the level of tank H/C falls and when tank M has a high level it rises. While it can go either way when the levels are the same. In addition the transition table for tank M clearly shows movement towards the setpoint which lies in tuple $(-,-,2)$. With the stable results from expanding the state space in this thesis, it would be very interesting to take the procedure to an existing plant. Where historic data would be available to compare results.

7.2 Hybrid automaton approach versus traditional hazop

The generic hazop procedure is well documented as an excellent tool for structured risk elimination. By reviewing the sources Crawleys et al. [8] it is also clear that the procedure is significantly time consuming. Not only in the actual hazop process, but also in the planning stage. Where a lot of personnel must be included to ensure a complete overview of the system. When it comes to the planning stage, it is hard to argue that the hybrid automaton approach is any less time consuming. The automaton still needs a good model, which should/may also include plausible site dependent situations. An example of such a situation was described in 1, where the usage of internal steam coils to heat a tank results in a plausible leakage situation. Another example could be potential hazards due to outer forces, e.g. a loading truck, crashing in to the system causing a leak. Such scenarios will not appear in a system design model but must in some cases be included in the hybrid automaton model, if they seem likely. Knowledge about the behaviour of the system is also very important when mapping the safe operating region. Especially in cases where a large number of internal regions posses danger, i.e. a reactor with a high number of reactions where the selectivity is important.

Using crossing of the outer boundaries as a synonym for a potential hazard makes sense in the 3-tank system. Crossing of the outer boundaries means to overfill or empty the tanks. This is obviously a situation that results in the tanks not functioning according to their purpose and in the least results in a product quality hazard. With the added possibility of defining internal hazardous regions (requires simple rewrite of the search algorithm), the transition table show great promise in catching all potential hazards. The keyword being all. A hazard identification tool that only achieve a "catch rate" of 98% is in it self a potential hazard. In the case study all potential hazards involving operating the system is detected. But only for discrete inputs. Shifting the flow rates may lead to additional hazardous transitions, but for the three tank system they will be detected if the system is secured according to the identified unique hazardous transitions. The current script evaluates "best and worst" situations, or discrete input models. A real system will have continuous inputs. Since the flow rates affects the equilibrium calculation it is very possible that evaluating max flow and no flow is insufficient for catching all hazards in a more complex system. A real plant study should emphasis whether additional simulations with change in flow rates yields different number of hazardous transitions

7.3 The developed search algorithm for hazardous transitions

The search algorithm successfully locates both unique hazardous transitions as well as the complete set. Comparison with the manual determination of hazards for tank H confirms that the algorithm works well for the three tank system. This is also further established when analysing the development of unique hazardous transitions for the energy balances for the cold tank in section 6.2.2. The cold tank has the fewest hazardous transitions, which reflects the low specific enthalpy for it's corresponding input flow. Since the search algorithm identifies all hazards in the three tank system it is deemed ready for a real plant test study.

7.4 Handling state variable transformation

An issue with the case study is how each hazard is connected to the outer constraints. For more complex systems internal hypercubes can possess potential hazards. A reactor may have a certain ph, or temperature range which shifts the reaction selectivity towards products who possess hazard to the product quality or the plant safety in itself. Being able to produce transition tables for both extensive and intensive quantities and searching for outer and inner boundaries is therefore of importance.

The state variable transformation from enthalpy to temperature reduces the number of dependable states. The outflow of the capacity have no direct impact on the temperature, in contrast to the enthalpy balance. This reduces the number of dependable states, leading to 1-dimensional transition table for tank H and C. The current version of the algorithm does not support 1-dimensional differentials, resulting in a script crash. To remove and manually evaluate such transition tables every time the script crashes can be tedious work if the system is complex with a large number of states. The script needs an addition that catches 1-dimensional problems and evaluates them seperately.

The search algorithm should be valid no matter which state transfer one performs. When it comes to temperature internal hazardous regions might be of interest. The search algorithm only needs boundaries around the hazardous temperature range, and can easily be rewritten to search for internal boundary transitions by utilizing the tuple representation. Since boundary setting is purely a user input, adding additional safety limits to account for the possibility of inaccurate results after state transformation is also just a

user input matter.

7.5 Further work before plant study

In some aspect the natural next step is to test the procedure on a real system. Doing so will provide a significant measure of "hazard catch rate" and script robustness. But before reaching that point, the inability to calculate transitions for 1-dimensional problems needs to be addressed. The 1-dimensional differentials may not be of great importance in it self. But the inability to include 1-D transitions seriously hurt the evaluation of the other states. One state might not be dependent on any other, but other states might depend on it.

8 Conclusion

The hybrid automaton transition tables (section B) can be used to correctly identify potential hazards for the mass and energy balances in the three tank system. Fulfilling both hypothesis 1;

Hypothesis 1: *Any guideword/parameter combination can be swapped with an evaluation of the hybrid automaton transition table*

and hypothesis 2;
Hypothesis 2: *If the automaton is generated over the boundaries spanning the safe operation domain. Any operational hazards must be visible in the transition table as a possible transition out of the domain*

The potential hazards detected with the hybrid automaton approach reflects the hazards detected in the conventional hazop procedure.

The developed search algorithm successfully automate the procedure of extracting hazardous transition. Corresponding with a manual approach.

State variable transformation form enthalpy to temperature is not doable as it decreases the number of dependable states. Resulting in a 1-dimensional transition table. This is due to outflows not directly affecting intensive quantities. Before a real plant test of the algorithm an addition must be must be made. So that states only dependant on it self can be evaluated without causing a software crash.

9 Suggested further development

1. Modify algorithm to handle 1-D transition tables.
2. Implement the hybrid automaton procedure on a real system
3. Test the hazard search algorithm on the real system
4. Validate the transition tables by comparing with continouos state trajectory
5. Evaluate the hazard "catch rate"

References

[1] P. Phillips. *Chapter 12: Stability Analysis.* PhD thesis, Technische Universiteit Eindhoven, 2001.

[2] M. Cicciotti. Model based safety and operability verification.

[3] H. Preisig. The abc of process modelling, 2013.

[4] L. Stryer H.S. Fogler, J.L. Tymoczko. *Elements of Chemical Engineering, 4th Edition.* Pearson Education, 2010.

[5] Claire Tomlin. Course aa278a at stanford u. lecture notes. `http://www.stanford.edu/class/aa278a/`, 2014. [Online; accessed 31-May-2014].

[6] B. Kuipers and S. Ramamoorthy. Qualitative modeling and heterogeneous control of global system behavior. Texas, 2002.

[7] F. Szidarovszky and A.T. Bahill. Chapter 12: Stability analysis. `http://www.sie.arizona.edu/sysengr/publishedPapers/SzidarStability.pdf/`, 2014. [Online; accessed 31-May-2014].

[8] Frank Crawleys, M. Preston, and B.Tyler. Hazop guide to best practice. `http://paulthorn.co.uk/healthandsafety/Risk%20Management/HAZOP-%20Guide%20to%20best%20practice-%20.pdf/`, 2014. [Online; accessed 01-June-2014].

[9] H. Preisig, Yun Xia Xi, and Khiang Wee Lim. Tailoring automata for fault diagnosability, 2003.

10 Variable List

10.1 Indexes and special nomenclature

Index	Description
α	phase index
i	Species index
j	capacity index
$\hat{x}$	Rate of state x
$\dot{x}$	Time deferential of state x
Δ	Net change in unit

10.2 Variables

Variable	Units	Description
c	mol/l	concentration
cp	$J/kg°C$	Specific heat coefficient
g	m/s^2	Gravitational acceleration
H_i^α	J/kg	Enthalpy
h_i^α	J/kg	Specific enthalpy
K	J	Kinetic energy
k	-	Reaction rate coeficient
M_j	kg	Mass
$\hat{m}_{j-1\|j}$	kg/s	Rate mass transport, from j-1 to j
P	J	Potential energy
p	Pa	Pressure
q	-	Discrete state
R	-	Guard - event detector
T_j^α	C	Temperature
t	s	time
U	J/kg	Internal energy
R	-	Reservoir
x	-	Continous statee

A Automaton Transition tables

A.1 3 Tank Mass Balances

The automaton is generated by use of nlinauto.m through MATLAB, see section NEED REF

A.1.1 $x_s = x_1$ (Tank H)

$$\begin{pmatrix} \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 1 & 0 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ 1 & 3 & 3 & 3 & 3 & 3 \\ 1 & 1 & 0 & 3 & 2 & 3 \\ 1 & 2 & 0 & 2 & 2 & 2 \\ 1 & 3 & 0 & 2 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 2 & 2 & 0 & 3 & 2 & 3 \\ 2 & 3 & 0 & 2 & 2 & 2 \\ 3 & 1 & 0 & 1 & 2 & 1 \\ 3 & 2 & 0 & 1 & 2 & 3 \\ 3 & 3 & 0 & 3 & 2 & 3 \end{pmatrix} \tag{133}$$

Displays the transition tables for x_3 in a 2-D x_1, x_3 automaton for all relevant inputs. The system is described in NEED REF

A.1.2 $x_s = x_2$ (Tank C)

$$\begin{pmatrix} \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ \text{undefined} & \text{undefined} & 0 & 1 & 0 & 1 \\ \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 \\ 2 & 3 & 3 & 3 & 3 & 3 \\ 1 & 1 & 0 & 3 & 2 & 3 \\ 1 & 2 & 0 & 2 & 2 & 2 \\ 1 & 3 & 0 & 2 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 2 & 2 & 0 & 3 & 2 & 3 \\ 2 & 3 & 0 & 2 & 2 & 2 \\ 3 & 1 & 0 & 1 & 2 & 1 \\ 3 & 2 & 0 & 1 & 2 & 3 \\ 3 & 3 & 0 & 3 & 2 & 3 \end{pmatrix} \tag{134}$$

Displays the transition tables for x_3 in a 2-D x_2, x_3 automaton for all relevant inputs. The system is described in NEED REF

A.1.3 $x_s = x_3$ (Tank M)

$$\left(\begin{array}{ccccccccccccccccc}
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 2 & 3 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\
1 & 1 & 1 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
1 & 1 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
1 & 2 & 2 & 0 & 0 & 1 & 1 & 3 & 3 & 3 & 3 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\
1 & 2 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 2 \\
1 & 3 & 2 & 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\
1 & 3 & 3 & 0 & 0 & 1 & 1 & 3 & 3 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 1 & 1 \\
2 & 1 & 1 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\
2 & 1 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 1 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
2 & 2 & 2 & 0 & 0 & 1 & 1 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 2 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
2 & 3 & 2 & 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 3 & 3 & 0 & 0 & 1 & 1 & 3 & 3 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 1 & 1 \\
3 & 1 & 1 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
3 & 1 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 \\
3 & 1 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 1 & 1 & 3 & 3 & 1 & 1 \\
3 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
3 & 2 & 2 & 0 & 0 & 1 & 1 & 3 & 3 & 3 & 3 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 \\
3 & 2 & 3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 1 & 1 & 3 & 3 & 1 & 1 \\
3 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
3 & 3 & 2 & 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 \\
3 & 3 & 3 & 0 & 0 & 1 & 1 & 3 & 3 & 1 & 1 & 3 & 3 & 1 & 1 & 3 & 3 & 1 & 1
\end{array}\right) \tag{135}$$

Displays the transition tables for x_3 in a 3-D x_1, x_2, x_3 automaton for all relevant inputs. The system is described in NEED REF

A.2 3 Tank Energy Balances

The automaton is generated by use of nlinauto.m through MATLAB, see section NEED REF

A.2.1 $x_s = x_4$ (Tank H)

$$\left(\begin{array}{cccccccccc}
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 4 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 \\
1 & 1 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 1 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 1 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 2 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 2 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 3 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
1 & 3 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 1 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
2 & 1 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 1 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
2 & 2 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 2 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
2 & 3 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
2 & 3 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
3 & 1 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 1 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 1 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
3 & 2 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 2 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 2 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3 \\
3 & 3 & 1 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 3 & 2 & 0 & 0 & 3 & 3 & 2 & 2 & 2 & 2 \\
3 & 3 & 3 & 0 & 0 & 3 & 3 & 2 & 2 & 3 & 3
\end{array}\right) \tag{136}$$

Displays the transition tables for x_4 in a 3-D x_1, x_3, x_4 automaton for all relevant inputs. The system is described in NEED REF

A.2.2 $x_s = x_5$ (Tank C)

$$
\left(\begin{array}{ccccccccccc}
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\text{undefined} & \text{undefined} & \text{undefined} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 3 & 5 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\
1 & 1 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 1 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 1 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 2 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 2 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 2 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 3 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 3 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
1 & 3 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 1 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 1 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 1 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 2 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 2 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 2 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 3 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 3 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
2 & 3 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 1 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 1 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 1 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 2 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 2 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 2 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 3 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 3 & 2 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3 \\
3 & 3 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 3
\end{array}\right) \tag{137}
$$

Displays the transition tables for x_5 in a 3-D x_2, x_3, x_5 automaton for all relevant inputs. The system is described in NEED REF

A.2.3 $x_s = x_6$ (Tank M)

The Automaton has dimensions $736x46$ which means that any presentation in paper form is meaningless because of the share size of the matrix. Since it's imperative that the columns is aligned, splitting the matrix in to several

pages does not seem like a valid solution either. An electronic copy can be attained be contacting the author at:

bjorntma@stud.ntnu.no.

B Hazop

B.1 Traditional Hazop procedure

B.1.1 Tank H

The filled hazop form for tank H:

Nr	Parameter	Guideword	Deviation	Cause
1	Flow rate in	Higher than	Visable as a resulting high hot water flow in to mixer if the inlet stream is supposed to be closed.	Malfunction on/off valve at the inlet (leaking). When signal is "on" there is no deviation visable deviation
2	Flow rate in	Lower than	Visable as a resulting low hot water flow in to mixer. Temperature fall the in mixer.	Pipe clogged at the inlet. When signal is off there is no deviation
3	Flow rate in	Reverse	Negative flow rate in inlet stream. Temperature drop in hot water tank, flow through tank H.	Mixer outlet clogged pipe. Higher pressure at the cold water side of the system.
4	Flow rate out	Higher than	High hot water flow in to mixer, low level in tank M	Malfunction level controller in mixer. Valve almost fully open
5	Flow rate out	Lower than	Low hot water flow in to mixer	Pipe clogged between mixer and hot water tank. Or clogged pipe/-malfunction level controller giving high evel in mixer.
6	Flow rate out	Reverse	Negative flow rate in the stream between mixer and hot water tank	Higher level in mixer. Level controller failing and possible clogged pipe. filling hot water tank.

Nr	Consequence	Protection	Action
1	Overflowing if system is supposed to be offline.	No protection against leaking.	Implement a high alarm on tank H.
2	Low temperature in mixer. Mixer not working according to intention	No protection against low flowrate.	Implement a low alarm on tank temperature tank M
3	Overflow cold water tank	No protection against negative flowrate.	Covered by action in number one
4	Potentially no consequence other than temperature shift in tank M.	No protection.	Deviation alarm temperature mixer M
5	Overflow hot water tank	No protection	Coverd by action in number one
6	Low temperature in hot water tank. Failed level control in mixer.	No protection against negative flowrate.	Implement a high alarm on level tank M

Nr	Parameter	Guideword	Deviation	Cause
7	Level	Higher than	High level in hot water tank	Clogged outlet/malfunction level controller in mixer leading to high level mixer
8	Level	Lower than	Low level in hot water tank	Clogged inlet or malfunction level controller in mixer leading to low level mixer
9	Temperature	Higher than	Higher temperature in hot water tank than normal operation/intention	deviation in inlet stream
10	Temperature	Lower than	Lower temperature in hot water tank than normal operation/intention	Deviation in inlet stream or reverse flow rate
11	Phase	Higher than	Level drop and flow rate drops	Fire in the plant
12	Phase	Lower than	Flow rate drop	Outside temperature significantly below freezing

Nr	Consequence	Protection	Action
7	Possibility of overflowing	No protection.	Implement a high alarm on tank H.
8	Possibility of emptying	No protection against emptying	Implement a low level alarm
9	Operation deviation. Stream from mixer not meeting requirements.	High temperature alarm tank H. Or temperature controller	
10	Operation deviation. Stream from mixer not meeting requirements.	Low temperature alarm tank H. Or temperature controller	
11	Damage to equipment plant. Outlet stream not meeting requirements	Fire safet is assumed to be according to local law.	No action
12	Overflowing. Damage to equipment. Outlet stream blocked	No protection against negative flowrate.	If outside, isolate

B.1.2 Tank C

The Hazop form for tank C:

Nr	Parameter	Guideword	Deviation	Cause
1	Flow rate in	Higher than	Visable as a resulting high cold water flow in to mixer if the inlet stream is supposed to be closed.	Malfunction on/off valve at the inlet (leaking). When signal is "on" there is no deviation visable deviation
2	Flow rate in	Lower than	Visable as a resulting low cold water flow in to mixer. Temperature rise in mixer.	Pipe clogged at the inlet. When signal is off there is no deviation
3	Flow rate in	Reverse	Negative flow rate in inlet stream	Mixer outlet clogged pipe. Higher pressure at the hot water side of the system.
4	Flow rate out	Higher than	High cold water flow in to mixer, low level in tank M or devation in inlet stream	Low level in mixer. Malfunction level controller in mixer
5	Flow rate out	Lower than	Low cold water flow in to mixer	Pipe clogged between mixer and cold water tank. Or clogged pipe/-malfunction level controller giving high evel in mixer.
6	Flow rate out	Reverse	Negative flow rate in the stream between mixer and cold water tank	Higher level in mixer. Level controller first emptying the system than the filling of cold water tank fails.

Nr	Consequence	Safeguards/Protection	Action
1	Overflowing if system is supposed to be offline.	No protection against leaking.	Implement a high alarm on tank C.
2	High temperature in mixer. Mixer not working according to intention	No protection against low flowrate.	Implement a high alarm on tank temperature tank M
3	Oveflow cold water tank	No protection against negative flowrate.	Covered by action in number one
4	Potentially no consequence. Dependent on any neighbouring system	No protection.	No action
5	Overflow cold water tank	No protection	Coverd by action in number one
6	High temperature in cold water tank. Failed level control in mixer.	No protection against negative flowrate.	Implement a high alarm on level tank M

Nr	Parameter	Guideword	Deviation	Cause
7	Level	Higher than	High level in cold water tank	Clogged outlet/-malfunction level controller in mixer. High level mixer
8	Level	Lower than	Low level in cold water tank	Malfunction level controller in mixer. Low level mixer
9	Temperature	Higher than	Higher temperature in cold water tank than normal operation/intention	Negative flow rate from mixer (see nr 6) or deviation in inlet stream
10	Temperature	Lower than	Lower temperature in cold water tank than normal operation/intention	Deviation in inlet stream
11	Phase	Higher than	Level drop and flow rate drops	Fire in the plant
12	Phase	Lower than	Flow rate drop	Outside temperature significantly below freezing

Nr	Consequence	Protection	Action
7	Possibility of overflowing	No protection.	Implement a high alarm on tank C.
8	Possibility of emptying	No protection against emptying	Implement a low level alarm on tank C
9	Operation deviation. Stream from mixer not meeting requirements.	High temperature alarm tank C. Or temperature controller	
10	Operation deviation. Stream from mixer not meeting requirements.	Low temperature alarm tank C. Or temperature controller	
11	Damage to equipment plant. Outlet stream not meeting requirements	Fire safety is assumed to be according to local law.	No action
12	Overflowing. Damage to equipment. Outlet stream blocked	No protection against negative flowrate.	If outside, isolate

B.1.3 Tank M

The Hazop form for tank M:

Nr	Parameter	Guideword	Deviation	Cause
1	Flow rate in	Higher than	If not balanced, temperature devations in mixer. Higher flow rate out from mixer	System inlet streams higher than normal
1	Flow rate in	Lower than	If not balanced, temperature devations in mixer. Lower flow rate out from mixer	System inlet streams lower than normal
3	Flow rate in	Reverse	Negative flow rate in both inlet streams at the same time not likely/possible	None
4	Flow rate out	Higher than	Possible high flow rates in to mixer as well. If not balanced, temperature deviations in mixer.	Malfunction level controller in mixer or high flow in to mixer
5	Flow rate out	Lower than	Possible low flow rates in to mixer as well. If not balanced, temperature deviations in mixer	Malfunction level controller in mixer or high flow in to mixer
6	Flow rate out	Reverse	Negative flow rate in the stream between mixer and cold water tank	Higher level in mixer. Level controller first emptying the system than the filling of cold water tank fails.

Nr	Consequence	Safeguards/Protection	Action
1	Overflowing if system is supposed to be offline.	No protection against leaking.	Implement a high alarm on tank C.
2	High temperature in mixer. Mixer not working according to intention	No protection against low flowrate.	Implement a high alarm on tank temperature tank M
3	Oveflow cold water tank	No protection against negative flowrate.	Covered by action in number one
4	Potentially no consequence. Dependent on any neighbouring system	No protection.	No action
5	Overflow cold water tank	No protection	Coverd by action in number one
6	High temperature in cold water tank. Failed level control in mixer.	No protection against negative flowrate.	Implement a high alarm on level tank M

Nr	Parameter	Guideword	Deviation	Cause
7	Level	Higher than	High level in cold water tank	Clogged outlet/-malfunction level controller in mixer. High level mixer
8	Level	Lower than	Low level in cold water tank	Malfunction level controller in mixer. Low level mixer
9	Temperature	Higher than	Higher temperature in cold water tank than normal operation/intention	Negative flow rate from mixer (see nr 6) or deviation in inlet stream
10	Temperature	Lower than	Lower temperature in cold water tank than normal operation/intention	Deviation in inlet stream
11	Phase	Higher than	Level drop and flow rate drops	Fire in the plant
12	Phase	Lower than	Flow rate drop	Outside temperature significantly below freezing

Nr	Consequence	Protection	Action
7	Possibility of overflowing	No protection.	Implement a high alarm on tank C.
8	Possibility of emptying	No protection against emptying	Implement a low level alarm on tank C
9	Operation deviation. Stream from mixer not meeting requirements.	High temperature alarm tank C. Or temperature controller	
10	Operation deviation. Stream from mixer not meeting requirements.	Low temperature alarm tank C. Or temperature controller	
11	Damage to equipment plant. Outlet stream not meeting requirements	Fire safety is assumed to be according to local law.	No action
12	Overflowing. Damage to equipment. Outlet stream blocked	No protection against negative flowrate.	If outside, isolate

B.2 Automaton Hazop procedure

See appendix A.1 for referenced transition tables.

B.2.1 Mass balance Tank H

State x_1 outer boundaries can be crossed at $(1, x)$ and $(3, x)$. In $(1, x)$ negative transition is searched for, i.e emptying tank H. For $(x, 3)$ a positive transition is searched for, or a situation where tank H overflows.

Nr	Input pattern	Tuple	Hazardous transition	Action
1	$[0\ 0\ 1\ 0\ 0]^T$	(1,1)	3: Possibility of emptying tank H. Reflect what happens if inlet flow $\hat{m}_{R_H\|H}$fails to open.	Low alarm tank H/low flow alarm $\hat{m}_{R_H\|H}$
2	$[0\ 0\ 1\ 0\ 0]^T$	(3,3)	3: Possibility of overfilling tank H, an effect of overfilling tank M. Reflect what happens if outlet flow $\hat{m}_{M\|D}$fails to open.	High alarm tank H/High alarm tank M or low flow alarm $\hat{m}_{M\|D}$
3	$[1\ 0\ 0\ 0\ 0]^T$	(3,x)	2: Trajectory in the direction of overfilling tank H. Reflect what happens if flow $\hat{m}_{H\|M}$ fails to open.	High alarm tank H or low flow alarm $\hat{m}_{H\|M}$
4	$[1\ 0\ 1\ 0\ 0]^T$	(3,2) (3,3)	3: Trajectory in the direction of overfilling tank C. Possible display of low gain level controller tank M	High alarm tank M or gain increase flow $\hat{m}_{M\|D}$
5	$[1\ 0\ 1\ 0\ 0]^T$	(1,1)	3: Possibility of emptying tank H. Dependent on tank M Level	Low alarm tank M
6	$[1\ 0\ 1\ 0\ 0]^T$	(3,3)	3: Possibility of overfilling tank H. Dependent on tank M Level	High alarm tank M

By utilizing *autohaz.m* (section C.6):

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 \\ 1 & 0 & 0 & 0 & 0 & 3 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 & 3 & 2 & 2 \\ 1 & 0 & 0 & 0 & 0 & 3 & 3 & 2 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 3 \\ 1 & 0 & 1 & 0 & 0 & 3 & 2 & 3 \\ 1 & 0 & 1 & 0 & 0 & 3 & 3 & 3 \end{pmatrix} \tag{138}$$

B.2.2 Mass balance Tank C

State x_2 outer boundaries can be crossed at $(1, x)$ and $(3, x)$. In $(1, x)$ negative transition is searched for, i.e emptying tank C. For $(x, 3)$ a positive transition is searched for, or a situation where tank C overflows.

Nr	Input pattern	Tuple	Hazardous transition	Action
1	$[0\ 0\ 0\ 1\ 0]^T$	(1,1)	3: Possibility of emptying tank C. Reflect what happens if inlet flow $\hat{m}_{R_C\|C}$fails to open.	Low alarm tank C/low flow alarm $\hat{m}_{R_C\|C}$
2	$[0\ 0\ 0\ 1\ 0]^T$	(3,3)	3: Possibility of overfilling tank C, an effect of overfilling tank M. Reflect what happens if outlet flow $\hat{m}_{M\|D}$fails to open.	High alarm tank C/High alarm tank M or low flow alarm $\hat{m}_{M\|D}$
3	$[0\ 1\ 0\ 0\ 0]^T$	(3,x)	2: Trajectory in the direction of overfilling tank C. Reflect what happens if flow $\hat{m}_{C\|M}$ fails to open.	High alarm tank C or low flow alarm $\hat{m}_{C\|M}$
4	$[0\ 1\ 0\ 1\ 0]^T$	(3,2) (3,3)	3: Trajectory in the direction of overfilling tank C. Possible display of low gain level controller tank M	High alarm tank M or gain increase flow $\hat{m}_{M\|D}$
5	$[0\ 1\ 0\ 1\ 0]^T$	(1,1) (3,3)	3: Possibility of emptying tank C. Possible display of low gain level controller tank M	High alarm tank M or gain increase flow $\hat{m}_{M\|D}$

By utilizing *autohaz.m* (section C.6):

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 3 & 3 \\ 0 & 1 & 0 & 0 & 0 & 3 & 1 & 2 \\ 0 & 1 & 0 & 0 & 0 & 3 & 2 & 2 \\ 0 & 1 & 0 & 0 & 0 & 3 & 3 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 3 \\ 0 & 1 & 0 & 1 & 0 & 3 & 2 & 3 \\ 0 & 1 & 0 & 1 & 0 & 3 & 3 & 3 \end{pmatrix} \tag{139}$$

B.2.3 Mass balance Tank M

State x_3 outer boundaries can be crossed at $(x, x, 1)$ and $(x, x, 3)$. In $(x, x, 1)$ negative transition is searched for, i.e emptying tank M. For $(x, x, 3)$ a positive transition is searched for, or a situation where tank M overflows.

By utilizing *autohaz.m* (section C.6):

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 3 & 3 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \end{pmatrix} \tag{140}$$

$$\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 1 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 1 & 1 & 0 & 3 & 3 & 3 & 3
\end{pmatrix} \tag{141}$$

B.2.4 Energy balance Tank H

State x_4 outer boundaries can be crossed at $(x, x, 1)$ and $(x, x, 3)$. In $(x, x, 1)$ a negative transition is searched for, i.e emptying tank M. For $(x, x, 3)$ a positive transition is searched for, or a situation where tank M overflows or the outer energy boundary is broken.

By utilizing *autohaz.m* (section C.6):

$$
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 2 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 3 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 2 & 1 & 3 & 2
\end{pmatrix}
\tag{142}
$$

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 2 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 2 & 3 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 1 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 2 & 1 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 2 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 2 & 3 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 1 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 2 & 3 & 2 \\
1 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 2 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 3 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 3 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 3 & 1 & 3 \\
1 & 0 & 1 & 0 & 0 & 1 & 3 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 2 & 3 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 1 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 2 & 3 & 3 \\
1 & 0 & 1 & 0 & 0 & 3 & 3 & 3 & 3
\end{pmatrix} \tag{143}$$

B.2.5 Energy balance Tank C

State x_5 outer boundaries can be crossed at $(x, x, 1)$ and $(x, x, 3)$. In $(x, x, 1)$ a negative transition is searched for, i.e emptying tank M. For $(x, x, 3)$ a positive transition is searched for, or a situation where tank M overflows or

the outer energy boundary is broken.

By utilizing *autohaz.m* (section C.6):

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 2 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 3 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 3 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 2 & 3 & 3
\end{pmatrix}
\tag{144}
$$

$$
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 2 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 2 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 2 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 2 & 3 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 1 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 1 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 2 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 2 & 3 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 3 & 1 & 3 \\
0 & 1 & 0 & 1 & 0 & 3 & 3 & 3 & 3
\end{pmatrix}
\tag{145}
$$

B.2.6 Energy balance Tank M

State x_6 outer boundaries can be crossed at $(x, x, x, x, x, 1)$ and $(x, x, x, x, x, 3)$. In $(x, x, x, x, x, 1)$ a negative transition is searched for, i.e emptying tank M. For $(x, x, x, x, x, 3)$ a positive transition is searched for, or a situation where tank M overflows or the outer energy boundary is broken.

The matrix contains a large number of rows which means that any presentation in paper form is meaningless because of the share size of the matrix. Since it's imperative that the columns is aligned, splitting the matrix in to several pages does not seem like a valid solution either. An electronic copy

can be attained be contacting the author at:

bjorntma@stud.ntnu.no.

C MATLAB Scripts

C.1 Bouncing ball

C.1.1 With zero cross detection *Runball.m*

Simulates the bouncing ball:

```
clear all
close all
clc
clf

global i

%Initial values
i.x1 = 10;
i.x2 = 0;

%Parameters
p.g = 9.81;
p.k = 0.8;

sim('Bouncing.mdl')

figure(1)
title('Bouncing ball example:')

subplot(2,1,1)
title('Bouncing ball example:')
hold on
axis([0 10 0 11])
plot(t,x(:,1),'b')
ylabel( 'Position [m]')
xlabel('time [s]')

subplot(2,1,2)
hold on
plot(t,x(:,2),'g')
axis([0 10 -15 15])
ylabel( 'Velocity [m/s]')
xlabel('time [s]')
plot([0 10],[0 0],'k--')

print(1,'-djpeg','-r450','bballfail')
```

Simulink model:

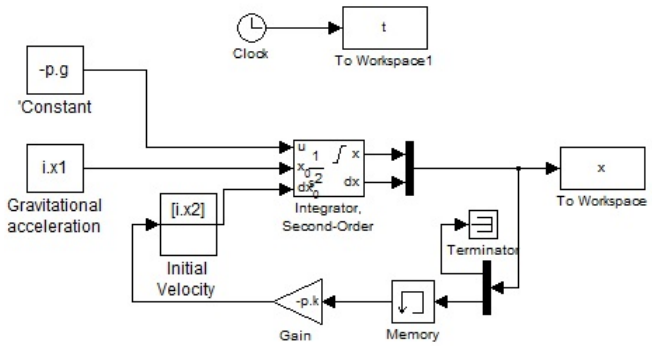

Figure 26: Simulink file for the bouncing ball model

C.1.2 Without zero cross detection *Runnonball.m*

Simulates the bouncing ball without zero cross detection:

```
clear all
close all
clc
clf

global i

%Initial values
i.x1 = 10;
i.x2 = 0;

%Parameters
p.g = 9.81;
p.k = 0.8;

sim('Bouncing.mdl')

figure(1)
title('Bouncing ball example:')

subplot(2,1,1)
title('Bouncing ball example:')
hold on
axis([0 10 0 11])
plot(t,x(:,1),'b')
ylabel( 'Position [m]')
xlabel('time [s]')

subplot(2,1,2)
hold on
plot(t,x(:,2),'g')
axis([0 10 -15 15])
```

```
33 ylabel( 'Velocity [m/s]')
34 xlabel('time [s]')
35 plot([0 10],[0 0],'k--')
36
37 print(1,'-djpeg','-r450','bballfail')
```

C.2 The inverted pendulum

C.2.1 Run script *Runpendu.m*

```
1  %Run script inverted pendulum - bjorntma@stud.ntnu.no
2  %Algorithm B.kuipers
3
4  clc
5  clf
6  clear all
7
8  global  k c umax c11 c12 c2 c3 phimax dphimax
9
10 k = 10;
11 c = 0.01;
12 umax = 5;
13 c11 = 0.4;
14 c12 = 0.3;
15 c2 = 0.5;
16 c3 = 0.5;
17 phimax = 0.4;
18 dphimax = 0.3;
19
20 % Initial conditions
21 x0 = [0 -1]; %x1 Angle, x2 angular velocity
22
23 % Time span
24 tspan = [0 20];
25
26 [t,x] = ode45(@inversependu,tspan,x0);
27
28 for i = 1:length(t)
29     [u(i),q(i)]= controller(x(i,:));
30 end
31
32 figure(1)
33 subplot(3,1,1)
34 hold on
35 plot(t,x(:,1),'b')
36 plot(t,x(:,2),'g')
37 ylabel('Radians')
38 legend('Angle','Angular Velocity')
39
40 subplot(3,1,2)
41 hold on
```

```
plot(t,q,'r')
ylabel('q')
legend('Discrete State')
axis([0 20 0.5 3.5])

subplot(3,1,3)
plot(t,u,'c')
ylabel('Radians/s')
legend('Input')
xlabel('time [s]')

print(1,'-djpeg','-r450','pendures')
```

C.2.2 Dynamics *inversependu.m*

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

global k c

u = controller(x);

dxdt(1) = x(2);
dxdt(2) = -c*x(2)-k*sin(x(1))-u;

dxdt = dxdt';
```

C.2.3 Controller *controller.m*

```
function [uend,qend] = controller(x)

global  k c umax c11 c12 c2 c3 phimax dphimax

%   x1 counter clockwise,x2 angular velocity
% angle from vertically down

alfa = (x(1)-pi)^2/phimax^2+(x(2)^2)/dphimax^2;
s = 0.5*x(2)^2-k*(1+cos(x(1)));

if alfa <= 1
    u = (c11 + k)*(x(1)-pi)+c12*(x(2));
    q = 1;
elseif s < 0
    u = -(c+c3)*x(2);
    q = 2;
else
    u = c2*x(2);
    q = 3;
end
```

```
21
22 if abs(u) > umax % saturated input
23     u = umax * sign(u);
24 end
25
26 uend=u;
27 qend=q;
```

C.3 2 tank system

C.3.1 2D linear automaton by Heinz Preisig *linauto2d.m*

```
1  % ---------------------------------------------------------------- LinAutom2 ::
2  % function [X,LinAutomat] = linauto2d(A,B,u,Boundaries,n_Boundaries)
3  %
4  % ..................................................................
5  % Computes the automaton for the continuous 2-dim plant {A,B} for the
6  % discrete input u and the domain observer definition Boundaries.
7  % ..................................................................
8  % This version cannot deal with completely decoupled states.
9  % ..................................................................
10 %
11 % arguments :
12 %   A       :: state propagation matrix, continuous plant
13 %   B       :: input gain matrix, continous plant
14 %   Boundaries  :: ragged matrix of boundaries, the domain observer
15 %   n_Boundaries    :: number of boundaries in each dimension
16 %
17 % results :
18 %   X       :: matrix of discrete states associated with
19 %   LinAutomat  :: matrix of discrete changes (binary coded)
20 %               0 :: no change
21 %               1 := -1
22 %               2 := +1
23 %               3 := {-1,+1}
24 % ------------------------------------------------------------------------
25
26 % 12:08PM  02-12-1994   H.A. Preisig
27 % 14-01-1995 11:09 AM   lots of fixes
28 % 20/01/95 16:45    derive this reduced procedure from linauto
29 % 2010-10-06           change name
30 % ------------------------------------------------------------------------
31 function [X,LinAutomat] = linauto2d(A,B,u,Boundaries,n_Boundaries)
32 % ------------------------------------------------------------------------
33
34 % error checks ========================= nothing done as yet
35    [na,ma] = size(A);
36    [nb,mb] = size(B);
37    [nu,mu] = size(u);
38    [nB,mB] = size(Boundaries);
39 % error checks ========================= nothing done as yet
```

```
40
41
42 % generate event-discrete state space
43    X     = combin(n_Boundaries-ones(size(n_Boundaries)))
44    X     = fliplr(X);
45    nw    = length(X(:,1));
46
47 % generate empty automaton table
48    LinAutomat = zeros(nw,na);
49
50    I   = ones(length(X(:,1)),1);
51
52 for i = 1:na             % loop through all states
53   j = 1;
54   while A(i,j)==0 | j == i  % find nondiagonal, nonzero element
55     j = j+1;
56     if j > na
57       disp('LinAutom :: error this state is independent')
58       return
59     end%if
60   end
61   direct    = sign(A(i,j));  % direction is given by this element
62
63
64   for l = 2:n_Boundaries(i)-1   % all internal boundaries of state i
65       disp('Boundaries(i,l)')
66       disp(Boundaries(i,l))
67       disp('-(A(i,i)')
68       disp(-(A(i,i)))
69       disp('A(i,j)')
70       disp(A(i,j))
71       disp('B(i,:)*u')
72       disp(B(i,:)*u)
73      x_int = -(A(i,i)*Boundaries(i,l) + B(i,:)*u)/A(i,j);
74      disp('CHECK')
75      disp(x_int*(1+eps*direct))
76      disp(Boundaries(j,:))
77      dxmin = do(x_int*(1+eps*direct),Boundaries(j,:));  % pull slightly
78      dxmax = do(x_int*(1-eps*direct),Boundaries(j,:));  % appart
79
80      x(i) =l-(direct+1)/2;
81      if (x(i) < n_Boundaries(i)-1) | (direct < 0)   % outer constraint
82        for q = dxmin:n_Boundaries(j)-1
83           x(j)             = q;
84           w_               = find(all((X-I*x)'==0)==1);
85           LinAutomat(w_,i) = LinAutomat(w_,i)+(direct+3)/2;
86        end%for q
87      end% if (l
88
89      x(i) = l-(-direct+1)/2;
90      if (x(i) > 1) | (-direct > 0)          % outer constraint
91        for q = 1:dxmax
92           x(j)             = q;
93           w_               = find(all((X-I*x)'==0)==1);
```

```
94            LinAutomat(w_,i) = LinAutomat(w_,i)+(-direct+3)/2;
95          end%for q
96        end%if(l
97      end%for l
98    end%for i
99
100   % ------------------------------------------------------------ LinAuto2.
```

C.3.2 Continouos model simulation

```
1   clc
2   clf
3   clear all
4
5   global p i
6   % Process parameters
7
8   p.p1 = .1 %Pipe flow resistance
9   p.p2 = .1 %Pump delivery rate
10
11  % Control and disturbancs
12
13  u = 1;
14  d = [0 0]';
15
16  % Initial conditions
17  i.x = [0 2]';
18
19  sim('Tank.mdl')
20
21  figure(1)
22  hold on
23  plot(x(:,2),x(:,1))
24  ylabel('Mass T1 [kg]')
25  xlabel('Mass T2 [kg]')
26  annotation('arrow',[0.75 0.75],[0.8 0.9])
27  text(3.2,3,'Overflow T1')
28  axis([0 4 0 4])
29  axis('square')
30
31  print(1,'-djpeg','-r450','twotanksim')
32
33  figure(2)
34
35  subplot(2,1,1)
36  hold on
37  title('T1')
38  plot(t,x(:,1))
39  ylabel('Mass [kg]')
40  xlabel('Time [s]')
41  axis([0 100 0 4 ])
```

```
annotation('arrow',[0.56 0.56],[0.8 0.9])
text(48,2,'Overflow T1')

subplot(2,1,2)
hold on
title('T2')
plot(t,x(:,2))
ylabel('Mass [kg]')
xlabel('Time [s]')
axis([0 100 0 4])
annotation('arrow',[0.635 0.635],[0.33 0.43])
text(58,2,'Overflow T2')

print(2,'-djpeg','-r450','twotankphase')

figure(3)
hold on
plot(x(:,2),x(:,1))
ylabel('Mass T1 [kg]')
xlabel('Mass T2 [kg]')
plot([3.5 3.5],[3 4],'k')
text(3.55,3,'O T1')
plot([1.333 1.333],[0 3],'k')
plot([0 3],[1.333 1.333],'k')
plot([2.666 2.666],[0 3],'k')
plot([0 3],[2.666 2.666],'k')
annotation('arrow', [0.5 0.5], [0.4 0.6])
text(2.03,1.42,'x_1 transition -> +1')
legend('Continous trajectory from x(0) = [0 2] with u=[1])')
axis([1 3 0 3])
axis('square')

print(3,'-djpeg','-r450','xonetrans')
```

```
function [sys,x0,str,ts] = sf_2Tank(t,x,u,flag)

global p i
switch flag

    case 0  %Initialize
      sys = [2,      % number of continuous states
             0,      % number of discrete states
             2,      % number of outputs
             3,      % number of inputs
             0,      % reserved must be zero
             0,      % direct feedthrough flag
             1];     % number of sample times

        x0 = [i.x];
        str = [];
        ts  = [0 0];   % sample time: [period, offset]

```

```
    case 1 %Derivatives


        % State equations:
        dxdt(1) = (1-u(2))*p.p1*(x(2)-x(1))+(1-u(3))*u(1)*p.p2;
        dxdt(2) = -(1-u(2))*p.p1*(x(2)-x(1));
        dxdt = dxdt(:);
        sys = dxdt;

    case 2 % Discrete state udate

        sys = []; %There is none

    case 3 % Outputs

        sys = [x(1) x(2)]';

    case 9 % Terminate

        sys = [];

    otherwise
        error(['unhandled flag = ',num2str(flag)]);

end
```

C.3.3 Automaton model visualization

```
% ============ Two tank automaton visualizer ================================

%
% 2014-05-30  B.T. Mathisen
% ------------------------------------------------------------------------
clear all
clf
clc

figure(1)
hold on
plot([2 2], [1 4],'k')
plot([3 3], [1 4],'k')
plot( [1 4],[2 2],'k')
plot( [1 4],[3 3],'k')
% set(Figure1,'defaulttextinterpreter','latex')
axis([1 4 1 4])
axis('square')
% 1 = text('Interpeter','latex','String','$\beta_2^1$')
set(gca,'xtickLabel',{'\beta_{2}^{1}','\beta_2^2','\beta_2^3','\beta_2^4'})
set(gca,'XTick',1:4,'xtickLabel',{[]})
set(gca,'YTick',1:4,'YTickLabel',{[]})
text(0.8,1,'\beta_1^1')
```

```
text(0.8,2,'\beta_1^2')
text(0.8,3,'\beta_1^3')
text(0.8,4,'\beta_1^4')

text(1,0.8,'\beta_2^1')
text(2,0.8,'\beta_2^2')
text(3,0.8,'\beta_2^3')
text(4,0.8,'\beta_2^4')

text(1.5,1.5,'q_1')
text(1.5,2.5,'q_4')
text(1.5,3.5,'q_7')
text(2.5,1.5,'q_2')
text(2.5,2.5,'q_5')
text(2.5,3.5,'q_8')
text(3.5,1.5,'q_3')
text(3.5,2.5,'q_6')
text(3.5,3.5,'q_9')
drawnow

print(1,'-djpeg','-r450','havisualize')
```

```
% clc
% clf
 clear all
 close all
figure(2)

dt = 1%
n = 4%
tmax = 100% Timerange

global p i
% Process parameters

p.p1 = .1 %Pipe flow resistance
p.p2 = .1 %Pump delivery rate

% Control and disturbancs

u = 1;
d = [0 0]';

% Initial conditions
x = [0 2]';

%Plant
dxdt(1) = (1-d(1))*p.p1*(x(2)-x(1))+(1-d(2))*u(1)*p.p2
dxdt(2) = -(1-d(1))*p.p1*(x(2)-x(1))

% Automata tracker
%x(1) is tracked in y direction, x(2) the x direction
```

```
%Split in a 3x3 squares. 4 boundaries

%With given initial conditions (First row, bottom row in figure)
%Ingores last column last row
 A = zeros(4,4);
 x1mark = 1;
 x2mark = 2;
A(x1mark,x2mark) = 1

flag = false
pause on
t = 0
while flag == false

        A = zeros(4,4);
        A(x1mark,x2mark) = 1;

        pcolor(A)
        axis('square')
        title(num2str(t))
        shading flat
        set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
        set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
        drawnow
        pause(0.25)

        if t==0
            figure(2)
            subplot(2,2,1)
            A = zeros(4,4);
            A(x1mark,x2mark) = 1;

            pcolor(A)
            axis('square')
            title(num2str(t))
            shading flat
            set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
            set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
            drawnow

             figure(1)
            A = zeros(4,4);
            A(x1mark,x2mark) = 1;

            pcolor(A)
            axis('square')
            title(num2str(t))
            shading flat
            set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
            set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
            drawnow
            print(1,'-djpeg','-r450','hat1')
        end
      if t==7
```

```
85              figure(2)
86              subplot(2,2,2)
87              A = zeros(4,4);
88              A(x1mark,x2mark) = 1;
89
90              pcolor(A)
91              axis('square')
92              title(num2str(t))
93              shading flat
94              set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
95              set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
96              drawnow
97
98              figure(3)
99              A = zeros(4,4);
100             A(x1mark,x2mark) = 1;
101
102             pcolor(A)
103             axis('square')
104             title(num2str(t))
105             shading flat
106             set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
107             set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
108             drawnow
109              print(3,'-djpeg','-r450','hat7')
110         end
111         if t==27
112             figure(2)
113             subplot(2,2,3)
114             A = zeros(4,4);
115             A(x1mark,x2mark) = 1;
116
117             pcolor(A)
118             axis('square')
119             title(num2str(t))
120             shading flat
121             set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
122             set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
123             drawnow
124         end
125         if t==35
126             figure(2)
127             subplot(2,2,4)
128             A = zeros(4,4);
129             A(x1mark,x2mark) = 1;
130
131             pcolor(A)
132             axis('square')
133             title(num2str(t))
134             shading flat
135             set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
136             set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
137             drawnow
138         end
```

```
139             figure(1)
140          x(1) = x(1) +(1-d(1))*p.p1*(x(2)-x(1))+(1-d(2))*u(1)*p.p2;
141          x(2) = x(2)- (1-d(1))*p.p1*(x(2)-x(1))
142
143          if x(1)>= 0
144              if x(1)> 4/3
145                   if x(1)> (4/3)*2
146                       if x(1)> 4
147                            A = zeros(4,4);
148                            pcolor(A)
149                            axis('square')
150                            title(['Out of bounds at: ',num2str(t)])
151                             shading flat
152                              set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
153                              set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
154                            drawnow
155                           flag = true
156                       else
157                           x1mark = 3
158                       end
159                   else
160                      x1mark = 2
161                   end
162              else
163                  x1mark = 1
164              end
165          else
166               A = zeroes(4,4);
167               title('Out of bounds at: ',num2str(t))
168              flag = true
169          end
170
171         if x(2)>= 0
172              if x(2)> 4/3
173                   if x(2)> (4/3)*2
174                       if x(2)> 4
175                            A = zeros(4,4);
176                            pcolor(A)
177                            axis('square')
178                            title(['Out of bounds at: ',num2str(t)])
179                             shading flat
180                              set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
181                              set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
182                            drawnow
183                           flag = true
184                       else
185                           x2mark = 3
186                       end
187                   else
188                      x2mark = 2
189                   end
190              else
191                  x2mark = 1
192              end
```

```
193          else
194               A = zeros(4,4);
195               title('Out of bounds at: ',num2str(t))
196
197              shading flat
198               set(gca,'XTick',1:4,'XTickLabel',[0:1.3333333:4])
199               set(gca,'YTick',1:4,'YTickLabel',[0:1.3333333:4])
200               drawnow
201                           flag = true
202         end
203          t=t+1
204          if t == tmax+1
205              flag = true
206              A = zeros(4,4);
207              A(x1mark,x2mark) = 1;
208          end
209 end
210
211              print(2,'-djpeg','-r450','hatall')
```

```
1  % ============ Tuple visualizer ================================
2
3  %
4  % 2014-05-30  B.T. Mathisen
5  % ------------------------------------------------------------------------
6  clear all
7  clf
8  clc
9
10 figure(1)
11 hold on
12 plot([2 2], [1 4],'k')
13 plot([3 3], [1 4],'k')
14 plot( [1 4],[2 2],'k')
15 plot( [1 4],[3 3],'k')
16 % set(Figure1,'defaulttextinterpreter','latex')
17 axis([1 4 1 4])
18 axis('square')
19 % 1 = text('Interpeter','latex','String','$\beta_2^1$')
20 set(gca,'xtickLabel',{'\beta_{2}^{1}','\beta_2^2','\beta_2^3','\beta_2^4'})
21 set(gca,'XTick',1:4,'xtickLabel',{[]})
22 set(gca,'YTick',1:4,'YTickLabel',{[]})
23 text(0.8,1,'\beta_1^1')
24 text(0.7,2,'l = 2')
25 text(0.8,3,'\beta_1^3')
26 text(0.8,4,'\beta_1^4')
27 annotation('arrow',[0.222 0.222],[0.45 0.30])
28 text(1.1,1.85,'1 =l-(direct+1)/2')
29
30 text(1,0.8,'\beta_2^1')
31 text(2,0.8,'\beta_2^2')
32 text(3,0.8,'\beta_2^3')
```

```
text(4,0.8,'\beta_2^4')

text(1.4,1.5,'(1,1)')
text(1.4,2.5,'(2,1)')
text(1.4,3.5,'(3,1)')
text(2.4,1.5,'(1,2)')
text(2.4,2.5,'(2,2)')
text(2.4,3.5,'(3,2)')
text(3.4,1.5,'(1,3)')
text(3.4,2.5,'(2,3)')
text(3.4,3.5,'(3,3)')
drawnow

print(1,'-djpeg','-r450','algvisual')
```

```
% ============ Tuple visualizer ================================

%
% 2014-05-30  B.T. Mathisen
% ------------------------------------------------------------------------
clear all
clf
clc

figure(1)
hold on
plot([2 2], [1 4],'k')
plot([3 3], [1 4],'k')
plot( [1 4],[2 2],'k')
plot( [1 4],[3 3],'k')
% set(Figure1,'defaulttextinterpreter','latex')
axis([1 4 1 4])
axis('square')
% 1 = text('Interpeter','latex','String','$\beta_2^1$')
set(gca,'xtickLabel',{'\beta_{2}^{1}','\beta_2^2','\beta_2^3','\beta_2^4'})
set(gca,'XTick',1:4,'xtickLabel',{[]})
set(gca,'YTick',1:4,'YTickLabel',{[]})
text(0.8,1,'\beta_1^1')
text(0.8,2,'\beta_1^2')
text(0.8,3,'\beta_1^3')
text(0.8,4,'\beta_1^4')

text(1,0.8,'\beta_2^1')
text(2,0.8,'\beta_2^2')
text(3,0.8,'\beta_2^3')
text(4,0.8,'\beta_2^4')

text(1.4,1.5,'(1,1)')
text(1.4,2.5,'(2,1)')
text(1.4,3.5,'(3,1)')
text(2.4,1.5,'(1,2)')
text(2.4,2.5,'(2,2)')
```

```
text(2.4,3.5,'(3,2)')
text(3.4,1.5,'(1,3)')
text(3.4,2.5,'(2,3)')
text(3.4,3.5,'(3,3)')
drawnow

print(1,'-djpeg','-r450','tuplevisual')
```

```
% ============ Two tank visualizer ================================

%
% 2014-05-30  B.T. Mathisen
% ------------------------------------------------------------------
clear all
clf
clc

figure(1)
hold on

% Equlibrium line
x2eq(1) = (0.1*0 - 0.1)/0.1;
x2eq(2) = (0.1*4 - 0.1)/0.1;
x1eq(1) = 0;
x1eq(2) = 4;

x1eq(3) = 0;
x1eq(4) = 4;
x2eq(3) = 0;
x2eq(4) = 4 ;

plot([x2eq(1) x2eq(2)], [x1eq(1) x1eq(2)],'g')
legend('x_1 Equilibrium line')
plot([1.33 1.33], [0 4],'k')
plot([2.66 2.66], [0 4],'k')
plot( [0 4],[1.33 1.33],'k')
plot( [0 4],[2.66 2.66],'k')
% set(Figure1,'defaulttextinterpreter','latex')
axis([0 4 0 4])
axis('square')

annotation('arrow',[0.35 0.35],[0.32 0.42])
text(0.95,1.1,'2')
annotation('arrow',[0.51 0.51],[0.32 0.42])
text(2,1.1,'2')
annotation('arrow',[0.72 0.72],[0.32 0.42])
text(3.4,1.1,'2')
annotation('arrow',[0.72 0.72],[0.60 0.70])
text(3.4,2.48,'2')
annotation('arrow',[0.56 0.56],[0.60 0.70])
text(2.35,2.48,'2')
annotation('arrow',[0.25 0.25],[0.42 0.32])
```

```
text(0.31,1.45,'1')
annotation('arrow',[0.31 0.31],[ 0.70 0.60])
text(0.7,2.8,'1')
annotation('arrow',[0.46 0.46],[ 0.70 0.60])
text(1.65,2.8,'1')

text(3.4,3.5,'0')

text(0.65,0.65,'(1,1)')
text(0.65,1.99,'(2,1)')
text(0.65,3.2,'(3,1)')
text(1.99,0.65,'(1,2)')
text(1.99,1.99,'(2,2)')
text(1.99,3.2,'(3,2)')
text(3.1,0.65,'(1,3)')
text(3.1,1.99,'(2,3)')
text(3.1,3.2,'(3,3)')
drawnow

print(1,'-djpeg','-r450','resvisual')
```

C.4 3 tank system

C.4.1 Modified non-linear automaton by H. Preisig and B. T. Mathisen *nlinauto.m*

```
% ========================================================================= NLinAutom ::
% function [X,SH,SU,M,Int] = nlinauto(model,jacob,u_pattern,Us,B);
% ..........................................................................
% Computes the automaton for the nonlinear continuous plant
%    dx = d(x,u)
% with the discrete input u
% and the domain observer defined by a set of boundaries.
%
% ..........................................................................
% arguments :
%   model          :: f(x,u)
%  jacob          :: name of function with
%                    sign pattern of jacobian of f(.,.) with respect to x
%  u_pattern      :: name of function with
%                    pattern of u's in state equations
%    x            :: state
%  u              :: input (discrete event)
%    B             :: set of vectors with boundaries: the domain observer
%
% results :
%    X             :: set of matrices of discrete states associated with
%  SH             :: set of coupled states
%  SU             :: set of active inputs
%    Automat       :: set of matrices of discrete changes (binary coded)
%                          0 :: no change
```

```
26 %                          1 := -1
27 %                          2 := +1
28 %                          3 := {-1,+1}
29 %  Int             :: intersection information
30 % ------------------------------------------------------------------------------
31
32 % 2000-12-17  H A Preisig
33 % ------------------------------------------------------------------------------
34 function [X,SH,U,SU,M,Int] = nlinauto(model,jacob,u_pattern,Us,B);
35 % ------------------------------------------------------------------------------
36
37 % error checks ============================================== nothing done as yet
38 % error checks ============================================== nothing done as yet
39
40    df_pattern   = feval(jacob);          % jacob returns sign pattern
41    patt_u       = feval(u_pattern);     % gives pattern of u's in the state eq's
42
43    card_S       = length(B);             % number of state variables in the plant
44    card_B       = cellfun('length',B);
45    card_Us      = cellfun('length',Us);
46    card_U       = length(card_Us);
47
48 for s = 1:card_S                         % all state variables
49    % this procedure cannot handle systems where the s state variable is not in the
50    % active set.
51    if df_pattern(s,s) == 0;
52       disp('s not active, cannot handle this case yet');
53       return
54    end
55    direct     = df_pattern(s,s);
56
57   % sets of active components with respect to system s
58    e_s        = zeros(1,card_S);
59    e_s(s)     = 1;
60    ne_s       = ~e_s;
61    A          = find(          df_pattern(s,:)   ~=0 ); % active states
62    H          = find( (ne_s .* df_pattern(s,:))  ~=0 ); % active but nut current
63    H_plus     = find( (ne_s .* df_pattern(s,:))   >0 ); % active, ~s & pos grad
64    H_min      = find( (ne_s .* df_pattern(s,:))   <0 ); % active, ~s & neg grad
65
66    card_H_plus= length(H_plus);
67    card_H_min = length(H_min);
68    card_H     = length(H);
69    card_A     = length(A);
70    ord_s_A    = find(A==s);
71
72   % some additional variables
73    range_s    = B{s}([1,card_B(s)]);                    % validity range of state s
74    rh         = 1:card_H;                               % index vector for H
75    direc      = df_pattern(s,s);                        % direction of change
76
77   % generate reduced event-discrete state space
78   % Note 1: that active co-ordinates are re-numbered
79   % Note 2: the state identifiers are ordinal numbers
```

```
80      rX       = combin(card_B(A) - ones(size(card_B(A))));
81      card_rD  = length(rX(:,1));              % number of discrete states
82      rrX      = combin(card_B(H) - ones(size(card_B(H))));
83      card_rrD = length(rrX(:,1));
84
85      SU{s}    = find(patt_u(s,:) == 1);
86      card_SU  = length(SU{s});
87      rU       = combin(card_Us(SU{s}));
88      card_rU  = length(rU(:,1));
89
90      for nu = 1:card_rU
91        % pick current input
92         u             = zeros(card_U,1);
93         u(SU{s})      = cellvec2vec(Us,SU{s},rU(nu,:));
94
95        % generate empty automaton table
96         M{s,nu}       = zeros(card_rD,card_H);
97         Int{s,nu}     = zeros(card_rrD,4);
98
99         for drr = 1:card_rrD                                  % allUs discrete states
100           l_h       = rrX(drr,:);
101           dummy     = zeros(1,card_A);
102 %               disp(dummy);
103           dummy(H)  = l_h;
104 %           disp(rX);
105 %           disp(ones(card_rD,1));
106 %           disp(ones(card_rD,1)*dummy);
107 %           disp(card_rD)
108
109           intermediate = ones(card_rD,1);
110           intermediate2 =  intermediate*dummy;
111 %           disp(length(intermediate2(1,:)))
112 %             disp(length(dummy(1,:)))
113           if length(intermediate2(1,:))>length(rX(1,:))
114 %               disp('test')
115               intermediate2(:,1) = [];
116               dummy(:,1) = [];
117           end
118           dummy2    = rX -  intermediate2;
119           dummy3    = diag(dummy ~=0);
120           dummy4    = dummy2*dummy3;
121           p_s       = find(all((dummy2)*dummy3==0 , 2));
122           x         = zeros(card_S,1);                        % start with zero vecto
123           x(s)      = (range_s(2)+range_s(1))/2;              % for x_s it's its val ra
124
125         % min crossing
126           if ~isempty(H_plus)
127              x(H_plus) = cellvec2vec(B,H_plus,l_h);
128           end
129           if ~isempty(H_min)
130              x(H_min)  = cellvec2vec(B,H_min,l_h+1);
131           end
132
133           if any(abs(x) == inf)
```

```
134             disp('not yet done');
135          else
136             [x_int_min,mdir] = get_zero(range_s,model,x,u,s);
137          end %if
138
139          % max crossing
140          if ~isempty(H_plus)
141             x(H_plus) = cellvec2vec(B,H_plus,l_h+1);
142          end
143          if ~isempty(H_min)
144             x(H_min)  = cellvec2vec(B,H_min,l_h);
145          end
146          if any(abs(x) == inf)
147             disp('not yet done');
148          else
149             [x_int_max,mdir] = get_zero(range_s,model,x,u,s);
150          end
151
152          if x_int_min > x_int_max
153             disp('linauto >> min > max ??? ')
154          end
155
156         % fill-in automaton table for state s
157          ex_x_min = ~isempty(x_int_min);
158          ex_x_max = ~isempty(x_int_max);
159
160          if ex_x_max | ex_x_min,
161             if ex_x_min
162                dx_int_min   = do(x_int_min*(1+2*eps),B{s});  % pull slightly
163             else
164                x_int_min    = NaN;
165                dx_int_min   = B{s}(1);
166             end
167             if ex_x_max
168                dx_int_max   = do(x_int_max*(1-2*eps),B{s});  % appart
169             else
170                x_int_max    = NaN;
171                dx_int_max   = B{s}(card_B(s));
172             end
173             rd_plus         = p_s(find(rX(p_s,ord_s_A) >= dx_int_min));
174             rd_min          = p_s(find(rX(p_s,ord_s_A) <= dx_int_max));
175             M{s,nu}(rd_plus,rh)= M{s,nu}(rd_plus,rh)+ ...
176                                  ones(size(M{s,nu}(rd_plus,rh)))*dir_code( direct);
177             M{s,nu}(rd_min ,rh)= M{s,nu}(rd_min ,rh)+ ...
178                                  ones(size(M{s,nu}(rd_min ,rh)))*dir_code(-direct);
179          else
180             M{s,nu}(p_s,rh)     = M{s,nu}(p_s    ,rh)+ ...
181                                  ones(size(M{s,nu}(p_s    ,rh)))*dir_code(
    mdir);
182             x_int_max       = NaN;
183             x_int_min       = NaN;
184             dx_int_max      = NaN;
185             dx_int_min      = NaN;
186          end
```

```
187           Int{s,nu}(drr,1:4)     = [x_int_min,x_int_max,dx_int_min,dx_int_max];
188        end % for rrd
189        U{s}(:,nu) = u;
190     end % nu
191        X{s}  = rX;
192        SH{s} = H;
193  end %for s
194  % ================================================================================ NLinAutom .
195  % ================================================================================ cellvex2vec::
196  % function M = cellvec2vec(Cv,r,c)
197  % ---------------------------------------------------------------------------------------------
198  % Cv   :: "matrix" rows are cells, columns are vectors
199  %         thus a set of vectors
200  % r    :: vector of row    indices
201  % c    :: vector of column indices
202  %         must both be of the same lenght
203  %
204  % 2000-12-18  H A Preisig
205  % ---------------------------------------------------------------------------------------------
206  function M = cellvec2vec(Cv,r,c)
207
208  j = 0;
209  for i = r
210     j = j+1;
211     M(j,:) = Cv{i}(c(j));
212  end
213  % ================================================================================ cellvex2vec .
214  % ================================================================================ get_zero ::
215  function [zero,f] = get_zero(range_s,model,x,u,s)
216
217  dx1 = model_eq(range_s(1),model,x,u,s)
218  dx2 = model_eq(range_s(2),model,x,u,s)
219  op = optimset('Display','off');
220  if sign(dx1)==sign(dx2)
221     zero = []
222     f    = sign(dx1)
223  elseif  isnan(dx1)
224
225     zero = []
226     f = 0
227  else
228     [zero,f] = fzero('model_eq',range_s,op,model,x,u,s)
229  end
230  % ================================================================================ get_zero
     |.                                                                                |
231  % ================================================================================ dir_code ::
232  function d = dir_code(direct)
233
234  if direct == 0
235     d = 0;
236  else
237     d = (direct+3)/2;
238  end
```

```
239 % ================================================================== get_code
    |.
```

C.4.2 Input file for *nlinauto.m*

```
1  % ============ three-tanks model script ================================ three_tanks
2  %
3  % 2000-11-30  H.A. Preisig
4  % 2010-10-06    adapting to new names
5  % ---------------------------------------------------------------------------
6
7  clear all
8
9  Boundaries{1} = [0,1.33,2.66,4];
10 Boundaries{2} = [0,1.33,2.66,4];
11 Boundaries{3} = [0,1.33,2.66,4];
12 temp=([20,40,60,80]-ones(1,4)*20).*4187.*4;
13
14 Boundaries{4} = temp;
15 Boundaries{5} = temp;
16 Boundaries{6} = temp;
17
18 for iu = 1:5
19    Us{iu} = [0 1];
20 end
21
22 model          = 'three_tanks_diff';
23 jacob          = 'three_tanks_jacob';
24 u_patt         = 'three_tanks_u_pattern';
25
26 [X,SH,U,SU,Automat,Int] = nlinauto(model,jacob,u_patt,Us,Boundaries);
27
28
29 for s = 1:6
30    headx = sort([s,SH{s}]);
31    dummy1 = NaN*ones(length(U{s}(:,1)),length(headx));
32    Aut{s} = [dummy1;headx;X{s}]
33
34    for nu = 1:length(U{s}(1,:))
35       dummy = U{s}(:,nu)*ones(size(SH{s}));
36       Aut{s} = [Aut{s},[dummy;SH{s};Automat{s,nu}]];
37       if nu == 1
38          INT{s} = Int{s,nu}
39       else
40          INT{s} = [INT{s},Int{s,nu}];
41       end
42    end
43 end
44
45 for s = 1:6
46    fn = ['Auto',num2str(s),'.txt'];
```

```
    A  = Aut{s};
    save(fn,'-ascii','-tabs','A')
    fn = ['Int',num2str(s),'.txt'];
    A  = INT{s};
    save(fn,'-ascii','-tabs','A')
end
```

```
% ========= 3 Tanks water mixing plant ====================================
%
%  function dx = three_tank_diff(x,u);
%
% Differentials for the 3 Tank system with energy mixing
% LHS calculates the stats ( 3 mass states followed by 3 energy states)
% x         :: states
% u         :: settings of valves and controllers
%              u(1) :: Inflow tank H)
%              u(2) :: Inflow tank C
%              u(3) :: Flow between tank H & M
%              u(4) :: Flow between tank C & M
%              u(5) :: Flow from tank M to (D)rain
%
% ------------------------------------------------------------------------
% uses three_tank_para
% ------------------------------------------------------------------------
% 2014-13-05  B.T. Mathisen - Adapted to fit the script by H.A. Preisig
% ------------------------------------------------------------------------

function dxdt = three_tanks_diff(x,u)

% Initialize parameter script
    three_tanks_para

% Flows
        n_RH_H = u(1)*p.p1;
        n_RC_C = u(2)*p.p2;
        n_H_M  = u(3)*p.p3*(x(1)-x(3));
        n_C_M  = u(4)*p.p4*(x(2)-x(3));
        if x(3)>2
            n_M_D  = u(5)*p.p5*(x(3)-2);
        else
            n_M_D  = 0;
        end

% State equations
        dxdt(1) = n_RH_H - n_H_M;
        dxdt(2) = n_RC_C - n_C_M;
        dxdt(3) = n_H_M  + n_C_M - n_M_D;
        if x(1)>x(3) & x(2)>x(3)
        dxdt(4) = p.h4*n_RH_H - (x(4)/x(1))*n_H_M;
        dxdt(5) = p.h5*n_RC_C - (x(5)/x(2))*n_C_M;
        dxdt(6) = (x(4)/x(1))*n_H_M  + (x(5)/x(2))*n_C_M...
            -  (x(6)/x(3))*n_M_D;
```

```
        elseif x(1)>x(3)
        dxdt(4) = p.h4*n_RH_H - (x(4)/x(1))*n_H_M;
        dxdt(5) = p.h5*n_RC_C - (x(6)/x(3))*n_C_M;
        dxdt(6) = (x(4)/x(1))*n_H_M  + (x(6)/x(3))*n_C_M...
            -  (x(6)/x(3))*n_M_D;
        elseif x(2)>x(3)
        dxdt(4) = p.h4*n_RH_H - (x(6)/x(3))*n_H_M;
        dxdt(5) = p.h5*n_RC_C - (x(5)/x(2))*n_C_M;
        dxdt(6) = (x(6)/x(3))*n_H_M  + (x(5)/x(2))*n_C_M...
            -  (x(6)/x(3))*n_M_D;
        else
        dxdt(4) = p.h4*n_RH_H - (x(6)/x(3))*n_H_M;
        dxdt(5) = p.h5*n_RC_C - (x(6)/x(3))*n_C_M;
        dxdt(6) = (x(6)/x(3))*n_H_M  + (x(6)/x(3))*n_C_M...
            -  (x(6)/x(3))*n_M_D;
        end


        dxdt = dxdt';
```

```
% ========= 3 tanks plant Jacobian sign pattern ============================
%
% function ddx = three_tanks_jacob;
% sign pattern of component equilibrium surfaces
%
% ------------------------------------------------------------------------
%
% ------------------------------------------------------------------------
% 2014-05-13  B.T Mathisen rewrite of H.A Preisig veirson
% ------------------------------------------------------------------------

function ddx = three_tanks_jacob;


% sign pattern of equilibrium surface:
   ddx(1,:) = [-1  0  1  0   0   0];
   ddx(2,:) = [0  -1  1  0   0   0];
   ddx(3,:) = [1   1  -1  0   0   0];
   ddx(4,:) = [-1  0  1  -1   0   0];
   ddx(5,:) = [0  -1  1  0   -1   0];
   ddx(6,:) = [1  1  -1  1   1   -1];
```

```
% ========= 3 Tanks water mixing plant parameters ==========================
%
%  Parameters
% ------------------------------------------------------------------------
% for use in three_tank_diff
% ------------------------------------------------------------------------
% 2014-13-05  B.T. Mathisen - Adapted to fit the script by H.A. Preisig
% ------------------------------------------------------------------------
```

```
9
10
11 p.p1 = 1; %Hot Pump delivery rate
12 p.p2 = 1; %Cold Pump delivery rate
13 p.p3 = 1; % H-M pipe resistance
14 p.p4 = 1; % C-M pipe resistance
15 p.p5 = 10; % Gain M-D control
16 p.cp = 4187; % cp H2O J/kgK
17
18 % Initial conditions for reservoir specific enthalpy
19 i.x = [20 20 10]';
20 i.T = [80 20 50]';
21 i.T_RH = 80;
22 i.T_RC = 20;
23 i.H = p.cp*(i.T-20).*i.x;
24 i.x = [i.x; i.H];
25
26 p.h4 = p.cp*(i.T_RH-20);
27 p.h5 = p.cp*(i.T_RC-20);
```

```
1  % ============ 3 Tanks_u_pattern ====================================::
2  % function patt_u = three_tanks_u_pattern;
3  % ------------------------------------------------------------------------
4  % Returns pattern of u's in the state variables
5  % ------------------------------------------------------------------------
6
7  % 2014-05-13 B.T. Mathisen
8  % ------------------------------------------------------------------------
9  function patt_u = three_tanks_u_pattern;
10 % ------------------------------------------------------------------------
11
12 %                  1  2  3  4  5
13 patt_u(1,:) = [   1  0  1  0  0  ];
14 patt_u(2,:) = [   0  1  0  1  0  ];
15 patt_u(3,:) = [   0  0  1  1  1  ];
16 patt_u(4,:) = [   1  0  1  0  0  ];
17 patt_u(5,:) = [   0  1  0  1  0  ];
18 patt_u(6,:) = [   0  0  1  1  1  ];
19 %------------------------------------------------------------------------
```

C.5 Linearization

```
1 % Initialize parameter script
2     three_tanks_para
3
4 % Linearization
5 p.p1 = 1; %Hot Pump delivery rate
6 p.p2 = 1; %Cold Pump delivery rate
7 p.p3 = 1; % H-M pipe resistance
```

```
p.p4 = 1; % C-M pipe resistance
p.p5 = 10; % Gain M-D control
p.cp = 4187; % cp H2O J/kgK

% Input
u(1) = 1; % On/off Hot pump
u(2) = 1; % On/off Cold pump
u(3) = 1; % On/off H-M valve
u(4) = 1; % On/off C-M valve
u(5) = 1;% Sp kg tank M

% Initial conditions (Tref = 20)
i.x = [0 0 0]';
i.T = [40 40 40]';
i.T_RH = 80;
i.T_RC = 20;
i.H = p.cp*(i.T-20).*i.x;
i.x = [i.x; i.H];

i.h(4) = p.cp*(i.T_RH-20);
i.h(5) = p.cp*(i.T_RC-20);

syms x_1 x_2 x_3 x_4 x_5 x_6

%Differentials
dx_1dt = u(1)*p.p1-u(3)*p.p3*(x_1-x_3);
dx_2dt = u(2)*p.p2-u(4)*p.p4*(x_2-x_3);
dx_3dt = u(3)*p.p3*(x_1-x_3)+u(4)*p.p4*(x_2-x_3)-p.p5*u(5)*(x_3-2);
dx_4dt = 1/(x_1)*((80-x_4)*u(1)*p.p1);
dx_5dt = 1/(x_2)*((20-x_5)*u(2)*p.p2);
dx_6dt = 1/(x_3)*((x_4-x_6)*u(3)*p.p3*(x_1-x_3)+...
    (x_5-x_6)*u(4)*p.p4*(x_2-x_3));

%Steady State Calculation
%Mass Balance
v = [x_1 x_2 x_3];
f = [dx_1dt;dx_2dt;dx_3dt];
df = jacobian(f,v);
s = [0; 0; 0];
k=2;
res(1) = inf;

format long
%Newton solver
while abs(res(1)) > 0.00010;
    dxdt = subs(f, [x_1 x_2 x_3],[s(1,k-1),s(2,k-1)...
        s(3,k-1)]);
    d2xdt2 = subs(df, [x_1 x_2 x_3],[s(1,k-1),s(2,k-1)...
        s(3,k-1)]);

    G =  dxdt;
    H =  d2xdt2;
    s(:,k) = s(:,k-1)-H^(-1)*G;  %Where k denotes newton iteration not time
    res = -sqrt(s(1,k)^2+s(2,k)^2+s(3,k)^2)...
```

```
62          +sqrt(s(1,k-1)^2+s(2,k-1)^2+s(3,k)^2);
63      k = k+1;
64  end
65  xmbfinal = s(:,end);
66
67  %Enthalpy balance
68
69  v = [x_4 x_5 x_6];
70   f = [dx_4dt;dx_5dt;dx_6dt];
71  df = jacobian(f,v);
72  l = i.H;
73  k=2;
74  res(1) = inf;
75
76  format long
77  while abs(res(1)) > 0.0010;
78      dxdt = subs(f, [x_1 x_2 x_3 x_4 x_5 x_6],[xmbfinal(1),xmbfinal(2),...
79          xmbfinal(3),l(1,k-1),l(2,k-1),l(3,k-1)]);
80      d2xdt2 = subs(df, [x_1 x_2 x_3 x_4 x_5 x_6],[xmbfinal(1),xmbfinal(2),...
81          xmbfinal(3),l(1,k-1),l(2,k-1),l(3,k-1)]);
82
83      G =  dxdt;
84      H =  d2xdt2;
85
86      l(:,k) = l(:,k-1)-H^(-1)*G;  %Where k denotes newton iteration not time
87      res = -sqrt(l(1,k)^2+l(2,k)^2+l(3,k)^2)...
88          +sqrt(l(1,k-1)^2+l(2,k-1)^2+l(3,k)^2);
89      k = k+1;
90  end
91  xebfinal = l(:,end);
92  x= [xmbfinal; xebfinal];
93
94  T(1) = x(4)/(x(1)*p.cp)+20;
95  T(2) = x(5)/(x(2)*p.cp)+20;
96  T(3) = x(6)/(x(3)*p.cp)+20;
97
98  %State space dotX=Adx + Bdu, calculating A and B. Evaluating the jacobians
99  %with SS calculation
100
101 syms x_1 x_2 x_3 x_4 x_5 x_6
102 v = [x_1 x_2 x_3 x_4 x_5 x_6];
103 f = [dx_1dt;dx_2dt;dx_3dt;dx_4dt;dx_5dt;dx_6dt];
104 df = jacobian(f,v);
105 A = subs(df, [x_1 x_2 x_3 x_4 x_5 x_6],[x(1),x(2),...
106         x(3),x(4),x(5),x(6)]);
107
108
109 syms u_1 u_2 u_3 u_4 u_5
110 dx_1dt = u_1*p.p1-u_3*p.p3*(x(1)-x(3));
111 dx_2dt = u_2*p.p2-u_4*p.p4*(x(2)-x(3));
112 dx_3dt = u_3*p.p3*(x(1)-x(3))+u_4*p.p4*(x(2)-x(3))-p.p5*u_5*(x(3)-2);
113 dx_4dt = 1/(x(1))*((80-x(4))*u_1*p.p1);
114 dx_5dt = 1/(x(2))*((20-x(5))*u_2*p.p2);
115 dx_6dt = 1/(x(3))*((x(4)-x(6))*u_3*p.p3*(x(1)-x(3))+...
```

```
116       (x(5)-x(6))*u_4*p.p4*(x(2)-x(3)));
117
118
119  v = [u_1 u_2 u_3 u_4 u_5];
120  f = [dx_1dt;dx_2dt;dx_3dt;dx_4dt;dx_5dt;dx_6dt];
121  df = jacobian(f,v);
122  B = subs(df, [x_1 x_2 x_3 x_4 x_5 x_6],[x(1),x(2),...
123           x(3),x(4),x(5),x(6)]);
```

C.6 Hazard finder *autohaz.m*

This function locates hazardous transitions. Can not deal with ragged boundaries

```
1  % ============ 3 Tanks Automated Hazop ====================================::
2  % function [Uhaz,Xhaz,Transhaz,Aligned] = autohaz(xs,bnr,xnr,unr,Aut)
3  % ------------------------------------------------------------------------
4  % Returns all possible hazardous transitions (transitions out of the outer
5  % boundary
6  % ------------------------------------------------------------------------
7  % 2014-05-30 B.T. Mathisen
8  % ------------------------------------------------------------------------
9  function [Uhaz,Xhaz,Transhaz,Aligned] = autohaz(xs,bnr,xnr,unr,Aut)
10 % ------------------------------------------------------------------------
11  [m,n] = size(Aut)
12
13  hazcount = 1
14 for j=(xnr+1):1:n
15       for i=(unr+2):1:m
16         if Aut(i,xs) == 1 & (Aut(i,j) == 3 | Aut(i,j) == 1)
17             Uhaz(hazcount,1:unr) = Aut(1:unr,j)';
18             Xhaz(hazcount,1:xnr) = Aut(i,1:xnr);
19             Transhaz(hazcount) = Aut(i,j);
20             hazcount = hazcount +1;
21         elseif Aut(i,xs) == 3 & (Aut(i,j) == 3 | Aut(i,j) == 2)
22             disp('FOUND')
23             Uhaz(hazcount,1:unr) = Aut(1:unr,j)';
24             Xhaz(hazcount,1:xnr) = Aut(i,1:xnr);
25             Transhaz(hazcount) = Aut(i,j);
26             hazcount = hazcount +1;
27         end
28     end
29 end
30 Transhaz = Transhaz' ;
31  [um,un] = size(Uhaz);
32  [xm,xn] = size(Xhaz);
33  [tm,tn] = size(Transhaz);
34 Aligned = Uhaz;
35 Aligned(1:xm,un+1:xn+un) = Xhaz;
36 Aligned(1:xm,xn+un+1:xn+un+tn) = Transhaz;
```

D Additional tests

D.1 State variable transformation

The product stream of the three tank system is now put under a tighter quality constraint. A temperature controller is induced with a set point of $50^{\cdot}C$. If the temperature differs with more than $10^{\cdot}C$ the product can not by utilized by the "customer". Product quality constraints appear in any production plant. The needed changes to the hybrid automaton plant is a state variable transformation. An unwritten goal of this thesis is that the hybrid automaton hazop procedure will be deemed ready for testing on a real plant. For that to be a reality the hybrid automaton also needs to be able to identify trajectories away from internal safe regions. Not only the outer constraints. This means that if the system has an initial temperature of $40^{\cdot}C$ will it move towards the safe region and stabilize? Or will it stay outside of the safe region. Not all hazards are connected to overfilling of capacities and the hybrid automaton approach needs to be able to handle these kind of systems as well.

The firs step is to set a new set of boundaries in the temperature domain that encircles the safe operating region:

$$\mathcal{B}_4 = \{0, 20, 40\} \tag{146}$$

$$\mathcal{B}_5 = \{60, 80, 100\} \tag{147}$$

$$\mathcal{B}_6 = \mathcal{B}_5 = \mathcal{B}_4 = \{0, 20, 40, 60, 80, 100\} \tag{148}$$

Where the safe region for the product stream (which have the same intensive quantities as tank M):

$$40^{\circ}C - 60^{\circ}C \tag{149}$$

State transformation:

$$\frac{dH_M}{dt} = \hat{H}_{H|M} + \hat{H}_{C|M} - \hat{H}_{M|D} \tag{150}$$

$$\frac{dH_M}{dt} = c_p(T_H - T_{Ref})\hat{n}_{H|M} + c_p(T_C - T_{Ref}) + \hat{n}_{H|M} - c_p(T_M - T_{Ref})\hat{n}_{M|D} \tag{151}$$

Enthalpy is a function of pressuere, temperature and the species composition:

$$H = (T, p, \underline{m}) \tag{152}$$

For constant pressure and an assumed constant heat cpacity this yeilds:

$$\frac{dH_M(T_M, m_M)}{dt} = m_M c_p \frac{dT_M}{dt} + h(T_M)\frac{dm_M}{dt} \tag{153}$$

$$\frac{dH_M(T_M, m_M)}{dt} = m_M c_p \frac{dT_M}{dt} + h(T_M)\frac{dm_M}{dt} \tag{154}$$

$$\tag{155}$$

Inserting in 151:

$$\frac{dT_M}{dt} = \frac{1}{m_M c_p}(c_p(T_H - T_{Ref})\hat{n}_{H|M} + c_p(T_C - T_{Ref})\hat{n}_{C|M} - c_p(T_M - T_{Ref})\hat{n}_{M|D} - c_p(T_M - T_{Ref})(\hat{n}_{H|M} + \hat{n}_{H|M} - \hat{n}_{M|D})) \quad (156)$$

$$\frac{dT_M}{dt} = \frac{1}{m_M c_p}(c_p(T_H - T_M)\hat{n}_{H|M} + c_p(T_C - T_M)\hat{n}_{C|M} \quad (157)$$

Setting temperature as the state:

$$\frac{dx_6}{dt} = \frac{1}{x_3}((x_4 - x_6)u_3\Theta_3(x_1 - x_3) + (x_5 - x_6)u_4\Theta_4(x_2 - x_3) \quad (158)$$

Implementation in *nlinauto2d.m* leads to several problems. If a energy balance is conducted for tank H, and C. The script fails to compute due to no dependent variables for normal stream direction. Testing a system where the tank H and C are omitted and used as inputs, leads to trouble in the root solver.

D.2 Utilizing linear automaton

Disregarded as it requires all states to be dependend, i.e. no zero entries in the A matrix

$$\underline{f}(\underline{x},\underline{u}) \approx \underline{f}(\underline{x}(0),\underline{u}(0)) + \left.\frac{\partial \underline{f}(\underline{x},\underline{u})}{\partial \underline{x}^T}\right|_{\underline{x}(0),\underline{u}(0)} (\underline{x} - \underline{x}(0)) + \left.\frac{\partial \underline{f}(\underline{x},\underline{u})}{\partial \underline{u}^T}\right|_{\underline{x}(0),\underline{u}(0)} (\underline{u} - \underline{u}(0)) \quad (159)$$

The Taylor expansion may than be used to transform the state equations in to standard state space representation:

$$\Delta\dot{\underline{x}} = \underline{\underline{A}}\Delta\underline{x} + \underline{\underline{B}}\Delta\underline{u} \quad (160)$$

This will partly be utilized when calculating the sign of the jacobian. As evaluating the A matrices is necessary for some of the more complex sign evaluations which is not directly readable from $\underline{\underline{J}}_A$. The A and B matrices are both retrieved calculating the Jacobian $\underline{\underline{J}}$ with respect to a given set of states and input:

$$\underline{\underline{J}}_A = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} & \frac{\partial \dot{x}_1}{\partial x_3} & \frac{\partial \dot{x}_1}{\partial x_4} & \frac{\partial \dot{x}_1}{\partial x_5} & \frac{\partial \dot{x}_1}{\partial x_6} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \frac{\partial \dot{x}_2}{\partial x_2} & \frac{\partial \dot{x}_2}{\partial x_3} & \frac{\partial \dot{x}_2}{\partial x_4} & \frac{\partial \dot{x}_2}{\partial x_5} & \frac{\partial \dot{x}_2}{\partial x_6} \\ \frac{\partial \dot{x}_3}{\partial x_1} & \frac{\partial \dot{x}_3}{\partial x_2} & \frac{\partial \dot{x}_3}{\partial x_3} & \frac{\partial \dot{x}_3}{\partial x_4} & \frac{\partial \dot{x}_3}{\partial x_5} & \frac{\partial \dot{x}_3}{\partial x_6} \\ \frac{\partial \dot{x}_4}{\partial x_1} & \frac{\partial \dot{x}_4}{\partial x_2} & \frac{\partial \dot{x}_4}{\partial x_3} & \frac{\partial \dot{x}_4}{\partial x_4} & \frac{\partial \dot{x}_4}{\partial x_5} & \frac{\partial \dot{x}_4}{\partial x_6} \\ \frac{\partial \dot{x}_5}{\partial x_1} & \frac{\partial \dot{x}_5}{\partial x_2} & \frac{\partial \dot{x}_5}{\partial x_3} & \frac{\partial \dot{x}_5}{\partial x_4} & \frac{\partial \dot{x}_5}{\partial x_5} & \frac{\partial \dot{x}_5}{\partial x_6} \\ \frac{\partial \dot{x}_6}{\partial x_1} & \frac{\partial \dot{x}_6}{\partial x_2} & \frac{\partial \dot{x}_6}{\partial x_3} & \frac{\partial \dot{x}_6}{\partial x_4} & \frac{\partial \dot{x}_6}{\partial x_5} & \frac{\partial \dot{x}_6}{\partial x_6} \end{bmatrix} \quad (161)$$

$$\underline{\underline{J}}_A = \begin{bmatrix} -u_3\Theta_3 & 0 & +u_3\Theta_3 & \cdots \\ 0 & -u_4\Theta_4 & +u_4\Theta_4 & \cdots \\ u_3\Theta_3 & u_4\Theta_4 & -u_3\Theta_3 - u_4\Theta_4 - \Theta_5 & \cdots \\ -\frac{x_4}{x_1^2}x_3u_3\Theta_3 & 0 & \frac{x_4}{x_1}u_3\Theta_3 & \cdots \\ 0 & -\frac{x_5}{x_2^2}x_3u_4\Theta_4 & \frac{x_5}{x_2}u_4\Theta_4 & \cdots \\ \frac{x_4}{x_1^2}x_3u_3\Theta_3 & \frac{x_5}{x_2^2}x_3u_4 & -\frac{x_4}{x_1}u_3\Theta_3 - \frac{x_5}{x_2}u_4\Theta_4 - \frac{x_6}{x_3^2}u_5\Theta_5 & \cdots \end{bmatrix}$$
$$\begin{bmatrix} \cdots & 0 & 0 & 0 \\ \cdots & 0 & 0 & 0 \\ \cdots & 0 & 0 & 0 \\ \cdots & -\frac{1}{x_1}u_3\Theta_3(x_1 - x_3) & 0 & 0 \\ \cdots & 0 & -\frac{1}{x_2}u_4\Theta_4(x_2 - x_3) & 0 \\ \cdots & \frac{1}{x_1}u_3\Theta_3(x_1 - x_3) & \frac{1}{x_2}u_4\Theta_4(x_2 - x_3) & -\frac{1}{x_3}\Theta_5(x_3 - u_5) \end{bmatrix} \tag{162}$$

$$\underline{\underline{J}}_B = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial u_1} & \frac{\partial \dot{x}_1}{\partial u_2} & \frac{\partial \dot{x}_1}{\partial u_3} & \frac{\partial \dot{x}_1}{\partial u_4} & \frac{\partial \dot{x}_1}{\partial u_5} \\ \frac{\partial \dot{x}_2}{\partial u_1} & \frac{\partial \dot{x}_2}{\partial u_2} & \frac{\partial \dot{x}_2}{\partial u_3} & \frac{\partial \dot{x}_2}{\partial u_4} & \frac{\partial \dot{x}_2}{\partial u_5} \\ \frac{\partial \dot{x}_3}{\partial u_1} & \frac{\partial \dot{x}_3}{\partial u_2} & \frac{\partial \dot{x}_3}{\partial u_3} & \frac{\partial \dot{x}_3}{\partial u_4} & \frac{\partial \dot{x}_3}{\partial u_5} \\ \frac{\partial \dot{x}_4}{\partial u_1} & \frac{\partial \dot{x}_4}{\partial u_2} & \frac{\partial \dot{x}_4}{\partial u_3} & \frac{\partial \dot{x}_4}{\partial u_4} & \frac{\partial \dot{x}_4}{\partial u_5} \\ \frac{\partial \dot{x}_5}{\partial u_1} & \frac{\partial \dot{x}_5}{\partial u_2} & \frac{\partial \dot{x}_5}{\partial u_3} & \frac{\partial \dot{x}_5}{\partial u_4} & \frac{\partial \dot{x}_5}{\partial u_5} \\ \frac{\partial \dot{x}_6}{\partial u_1} & \frac{\partial \dot{x}_6}{\partial u_2} & \frac{\partial \dot{x}_6}{\partial u_3} & \frac{\partial \dot{x}_6}{\partial u_4} & \frac{\partial \dot{x}_6}{\partial u_5} \end{bmatrix} \tag{163}$$

$$\underline{\underline{J}}_B = \begin{bmatrix} \Theta_1 & 0 & -\Theta_3(x_1 - x_3) & 0 & 0 \\ 0 & \Theta_2 & 0 & -\Theta_4(x_2 - x_3) & 0 \\ 0 & 0 & \Theta_3(x_1 - x_3) & \Theta_4(x_2 - x_3) & \Theta_5 \\ \gamma_1\Theta_1 & 0 & -\frac{x_4}{x_1}\Theta_3(x_1 - x_3) & 0 & 0 \\ 0 & \gamma_2\Theta_2 & 0 & -\frac{x_5}{x_2}\Theta_4(x_2 - x_3) & 0 \\ 0 & 0 & \frac{x_4}{x_1}\Theta_3(x_1 - x_3) & \frac{x_5}{x_2}\Theta_4(x_2 - x_3) & \frac{x_6}{x_3}\Theta_5 \end{bmatrix} \tag{164}$$

To avoid any typing errors while deriving the functions the Jacobians was also calculated by use of Matlab. See Appendix (NEED REF) To finalize the system in a linear state space format a steady state calculation is quite handy. When using steady state data to fill in the jacobian the state space expression simplifies:

$$\Delta\underline{\dot{x}} = \underline{\underline{A}}\Delta\underline{x} + \underline{\underline{B}}\Delta\underline{u} \tag{165}$$
$$\underline{\dot{x}} - \underline{\dot{x}}\big|_{SS} = \underline{\underline{A}}\Delta\underline{x} + \underline{\underline{B}}\Delta\underline{u} \tag{166}$$
$$\underline{\dot{x}} = \underline{\underline{A}}\Delta\underline{x} + \underline{\underline{B}}\Delta\underline{u} \tag{167}$$
$$\tag{168}$$

Moving on a steady state calculation is used to finalize the linearized state space representation.

D.2.1 Steady State Calculation

A system is at steady state when the states are constant over the current time horizon. Resulting in:

$$0 = \underline{f}(t, \underline{x}, \underline{u}, \underline{d}) \tag{169}$$

Since the Jacobian already is calculated and the problem requires a root solver a Newton solver should be sufficient. The general expression for the Newton method is seen below:

$$x_{n+1} = x_n - J^{-1}(x_n) f(x_n) \tag{170}$$

In this special case the subscript denotes the iteration sequence number. The above procedure is looped until the difference between $x_{n+1} - x_n$ is sufficiently small, i.e.$f(x_n) = 0$. By use of the Newton solver the problem was solved for the default input and initial conditions, see appendix C.5 for MATLAB script.

D.2.2 Linear state space representation

The standard state space representation for a model linearized over a steady state is:

$$\dot{\underline{x}} = \underline{\underline{A}} \Delta \underline{x} + \underline{\underline{B}} \Delta \underline{u} \tag{171}$$