Simultaneous Fault Detection of Heat Exhangers

Gábor Szederkényi

12 March 1998

Model-Based Fault Detection of Heat Exchangers

written by

Gábor Szederkényi

supervisor:

Katalin Hangos

Department of Applied Computer Science

University of Veszprém

1998.

Tartalmi összefoglaló

Diplomamunkám témája ipari hőcserélők modell-alapú hibadiagnosztikája. Két hibajelenséget vizsgáltam: a hőcserélő külső tartályának kilyukadását és a hőcsere hatékonyságának változását a hőcserélő belső felületén lerakódó illetve onnan leváló anyagok (vízkő) hatására.

A feladat megoldásához szükséges a vizsgált rendszer (az ellenáramú hőcserélő) valamint a detektálandó hibajelenségek modellezése, melynek során jelentős szerepet játszik a folyamatról előzetesen rendelkezésre álló információ.

Valamennyi kifejlesztett diagnosztikai eljárás a hőcserélő modell paramétereinek becslésén és a paraméterekben bekövetkező változások detektálásán alapul. Az algoritmusok tervezése során a hőcserélő folyadékhőmérsékleteinek mérhetősége szerint három esetet különböztettem meg. Az első, és legkevésbé valósághű esetben a feltételezés az, hogy a hideg- és meleg oldali hőmérsékletek nemcsak a be- és kimeneteknél, hanem a hőcserélő több pontján mérhetők. A kifejlesztett hibadetektáló algoritmusok ebben az esetben az ellenáramú hőcserélő kaszkád-modelljének diszkrét idejű változatát használják fel a diagnosztikai információt hordozó paraméterek becslésére. A hibák külön-külön történő detektálásán kívül ebben az esetben olyan algoritmust is terveztem, amely párhuzamosan képes jelezni a két hibajelenséget még akkor is, ha azok egyszerre fordulnak elő.

A második esetben azt feltételezzük, hogy a hőcserélő ki- és bemeneti folyadékhőmérsékletein kívül a külső (hideg) fázis hőmérsékleteloszlása több ponton mérhető. Itt a hőcserélő egy-egy cellája lineáris idő-invariáns rendszerrel modellezhető a mintavételi időpontok között, így az egyes cellákat leíró átviteli függvények segítségével becslés adható a belső (meleg) fázis hőmérsékleteloszlására. E feltételezések mellett működik jó eredményekkel az egyik lyukadást detektáló algoritmus.

A legvalósághűbb eset az, amikor csupán a hőcserélő be- és kilépő folyadékáramainak hőmérsékletei mérhetők. Ekkor a rendszer dinamikája egyetlen cella modelljével, azaz három közönséges differenciálegyenlettel közelíthető. Amint azt a szimulációs eredmények mutatják, ez a közelítés sikerrel alkalmazható hibadiagnosztikai célokra. További előnye az egyszerűsített modellt alkalmazó eljárásoknak a rendkívül kicsi számítási igény.

A hőcserélő működését szimuláló modellt Simulink [15] rendszerben építettem, a hibadetektáló algoritmusok különböző változatait Matlab [14] programozási nyelven implementáltam.

Nyilatkozat

Ezt a munkát, mint diplomadolgozatot a Veszprémi Egyetem Számítástudomány Alkalmazása Tanszékén készítettem, hogy mérnök informatikus diplomát szerezzek. Ezt a dolgozatot diplomamunkaként nem adom be más felsőoktatási intézménybe.

Veszprém, 1998. május 12.

Szederkényi Gábor

Feladatkiírás

A diplomamunka célja paraméterbecslésen alapuló diagnosztikai eljárások kifejlesztése ipari hőcserélők két leggyakrabban előforduló hibajelenségének, a vízkövesedésnek és a hőcserélő külső tartálya kilyukadásának detektálására.

A feladat megvalósításához a jelölt a rendelkezésre álló irodalom alapján válasszon ki olyan hőcserélő-modellt, mely a mérnöki gyakorlatban elfogadott, és paramétereinek nyomonkövetéséből a fent leírt hibajelenségekre következtetni lehet. Szintén a szakirodalomból valamint az ipari gyakorlatból hozzáférhető folyamatmérnöki és fizikai ismeretek segítségével modellezze az egyes hibajelenségeket. Az így nyert modellekhez fejlesszen ki olyan algoritmusokat, melyek becslést adnak a diagnosztikai információt hordozó paraméterek értékére, és ennek alapján döntenek a hibaállapotok fennállásáról, illetve hiba esetén riasztást generálnak.

Az eljárásokat több változatban készítse el a folyadékhőmérsékletek mérhetőségére vonatkozó különbözö feltételezések szerint. Az egyes esetekben vizsgálja a két hibajelenség egymástól független és párhuzamosan történő detektálásának lehetőségeit.

Készítsen Matlab/Simulink környezetben a hőcserélő működését szimuláló modellt, és Matlab programozási nyelven implementálja a kifejlesztett paraméterbecslő és hibadetektáló algoritmusokat. A szimulációból származó adatokon vizsgálja az eljárások működését, adjon meg egyszerű összefüggéseket az algoritmusok hangolóparamétereinek megválasztására.

Köszönetnyilvánítás

Legelőször témavezetőmnek, Dr. Hangos Katalinnak mondok hálás köszönetet a gondos irányításért, és azért a sok-sok támogatásért, amellyel az elmúlt két év során munkámat, fejlődésemet segítette. Dolgozatom megírásához értékes szakmai segítséget kaptam Dr. Erik Weyertől, a Melbourne-i Egyetem munkatársától, amit ezúton köszönök meg. Köszönettel tartozom szüleimnek, hogy tanulmányaimat sok esztendőn át minden erejükkel segítették. Hálás vagyok mindazoknak, akik észrevételeikkel, kérdéseikkel, javaslataikkal vagy bármilyen más módon hozzájárultak e diplomadolgozat megszületéséhez.

Contents

List of Figures

List of Tables

1 Introduction

Everything should be made as simple as possible,
but not simpler.
/Albert Einstein (1879-1955)/

Fault detection and diagnosis are playing an increasingly important role in the process industries. Dynamic modelling of the equipment, called operating units is a well developed area, and model based diagnostic methods can therefore be efficiently and relatively easily applied.

Heat exchagers are widely used in the process industries and other areas. They are usually arranged in units containing several (5-200) of them. From a modelling point of view, heat exchangers are among the simplest and most investigated operating units for which we have both reliable dynamic models with known properties and accumulated operational experience. This experience can be used in formulating suitable dynamic models of the typical faults which can be encountered in heat exchangers.

Model-based fault detection methods are proposed in this diploma work. The methods are based on a first principle model of the operating unit: a countercurrent heat exchanger, and on the grey- and white box models of the faults: the deterioration of the the heat transfer by ageing and the leaking of the outer container. We propose to use recursive parameter estimation methods with a forgetting factor to track the heat transfer coefficients (HTCs) and the liquid volume in the outer container respectively. The settled material breakage fault is detected via detection of abrupt positive jumps in the estimated heat transfer coefficient using a detector based on a cumulative sum (CUSUM) test. The leakage fault is detected by estimating the liquid volume in the outer container and then applying the CUSUM test again.

The possibility to detect faults in any industrial equipment heavily depends on the availability of suitable measurements. For heat exchangers the variables related to the in- and outflows of the equipment (flow rates and temperatures) are usually measured but measurements along the equipment are rarely available. Therefore the possibilities of fault location in space are rather limited. The proposed fault detection methods have variants corresponding to various measurement settings:

the all temperature measurements available, only cold side temperature available and only the in- and outflow temperature measurements available cases.

The outline of the diploma work is as follows. The next section (Section 2) is a short literature review that summarizes those scientific results that are closely connected to the topic of this diploma work. In Section 3 the mathematical model of the countercurrent heat exchanger and the models of the faults we want to detect are discussed. Section 4 contains the derivation and description of the parameter estimation and fault detection algorithms and their tuning knobs. The purpose of Section 5 is to present the methods and results of the performed simulations.

2 Literature Review

The purpose of this section is to briefly discuss the basic concepts of fault detection, diagnosis and compensation and to review the most important scientific results in this field that are closely connected to the topic of this diploma work. It is also a task here to place the models and methods used in this paper among the different approaches that are surveyed.

2.1 Basic concepts

Dynamic process plants are becoming more and more complex, therefore there is a growing demand for fault detection and diagnosis in order to provide safe and continuous operation. *Fault detection* is the indication that something is going wrong in the system via various residual generation methods. A residual is such a function of time that is nominally zero or close to zero when no fault is present, and that is distinguishably different from zero when a certain component of the system fails [7]. The task of *fault diagnosis* is to determine the type, source and extent of the fault as well as the time of its occurence based on the observed analytical and heuristic knowledge of the symptoms . In other words, fault detection and diagnosis is the early indication of incipient faults that can help us to avoid major plant breakdowns and catastrophies and take appropriate actions in order to maintain the operation. The main task of fault compensation is to modify the normal mode configuration in order to compensate for the faults (if possible) by activating various back-up systems [12].

We can say that the detection and compensation of faults is one of the critical issues in the operation of high-performance systems: production equipment such as power stations, chemical processes, transportation vehicles like airplanes, space vehicles etc.

Fault detection, diagnosis and compensation schemes in plants detect and try to compensate faults in one or more of the following three subsystems: the actuator, the process (or plant) and sensor subsystems. Actuator faults are discrepancies between the intended control and its realization by the actuators. Sensor faults are discrepancies between the measured and true values of the plant's output or input variables. Process faults are disturbances acting on the plant causing a shift in the

plant outputs independently of the measured inputs, and may describe plant leaks, overloads, broken down components etc. With respect to the different sectors where the faults may occur, one can distinguish between actuator fault detection, sensor fault detection and process fault detection.

2.2 Approaches to fault detection and diagnosis

Approaches to fault detection and diagnosis are divided into three main streams: model-free methods, model-based methods and knowledge-based methods.

Model-free methods do not make use of a plant model. Limit value checking, which is one of them, most often works well if the process operates approximately in a steady state. The big advantage of these methods is their simplicity and reliability. But their application becomes more involved if the process changes frequently and rapidly its operating point. Since the plant variables may vary widely due to input variations, the setting of the check threshold is often a nontrivial task. Further problems may arise from the installation of special sensors and repeated hardware elements that are usually distributed spatially around the system to provide protection against localized damage. The major causes of the problems in this case are the extra cost and software requirement and the additional space needed to accommodate the equipment.

Model-based methods rely on the idea of analytical redundancy based upon theoretical ideas, signal analysis and process analysis. Signal analysis is performed through directly measurable signals by using signal models like correlation functions, frequency spectra, autoregressive moving average, statistical decision theory etc. Also, these methods use process analysis by using mathematical process and fault models together with parity space, state estimation, parameter estimation, detection filtering, variable threshold logic etc. [4].

Knowledge-based redundancy complements the analytical model-based method. In addition to the symptom generation with model-based method, heuristic symptoms can be produced by using qualitative knowledge provided by human operators. Such a method usually consist of a combination of logical rules, where each conclusion can, in turn, serve as a symptom in the next rule until the final conclusion is reached.

2.3 Important design issues of fault detection systems

The design of fault detection systems demands the thorough consideration of several issues and tradeoffs. Based on [12] the most important properties are as follows. *Rapid response* is a usual requirement since the quick detection of faults is of crucial importance in most cases. *Noise sensitivity* gives how sensitive the detection system is to certain high frequency effects. It is easy to notice that there is a tradeoff between rapid response and noise sensitivity. Increased noise sensitivity usually results in a greater number of false alarms (fault is detected in spite of the fact that the process operates normally). Another item that is also often in conflict with rapid response is the *degradation of system performance* using a given detection method. The *robustness* of a fault detection scheme in the presence of modelling errors is also of major importance in most practical cases. In addition to these issues the designer of fault detection systems should consider further questions in connection with software. These are *computational complexity*, *storage requirements* and *time requirements*. Finally, it is an essential consideration, *how the system takes advantage of new computer capabilities and structures* (e.g. designs that can be implemented in a modular or parallel way).

2.4 Process and fault models

Since the procedures that are going to be discussed in further sections are based on mathematical process and fault models, a short classification and description of such models is presented here. Generally speaking, a model is a tool for describing and understanding the world, it is a formal expression of our knowledge corresponding to the investigated phenomena [9]. Different classes of models are for example functional, physical and mathematical models. In mathematical models the relations between the variables of the system to be modelled are mapped into certain mathematical structures such as algebraic-, differential-, integral equations, logical functions etc. The knowledge represented by a mathematical model can be classified into the following four cathegories: laws, structure, parameters and states [9]. Laws are fundamental physical laws that determine the form of the equations in mathematical models. Structure refers to the inner structure of the phenomena and to the relations between its parts. The parameters of a mathematical model are

the particular values of the coefficients standing in the equations. A state is such a variable that describes the process of the investigated phenomena together with outside effects acting on the system.

According to the a priori information we have concerning the process, models can be classified into the following three groups. If we do not have any information about the structure and the parameters of the process, we use *black box models*. All we are able to do in this case is to make inferences based on the knowledge of the inputs and outputs of the system. When we have partial knowledge about the structure (and maybe about some parameters) of the system we speak about *grey box models*. Here we have a better understanding of the system than in the case of a black box model, but it still remains an incomplete view. When we know the complete structure of the system and some (or even all) of its parameters, we have a *white box model*. Here our knowledge of the system and its internal workings is virtually complete, though some parts or subsystems may remain grey boxes.

2.5 System identification and system parameter estimation

System identification gives solutions to two main problems: model parameter estimation and model structure estimation [3]. The problem statement of model parameter estimation is as follows. We are given a measurement record (input and output values), a certain model form (structure) and a loss function measuring the difference between the measured and estimated data. The goal is to estimate the model parameters from the data available that minimize the loss function. The problem statement of model structure estimation is the following. We are given a measurement record (input and output values), a set of candidate models (model structures) and a loss function. The aim in this case is to select the optimal structure that minimizes the loss function.

The approach to system identification, most of the notations and the basic methods for parameter estimation are from [6] and [5]. Both books have served as essentially important literature in this field for more than a decade.

2.6 Detecting changes in signals

The field of fault detection and diagnosis is closely connected to detecting changes in certain characteristic signals [12]. These changes can be abrupt or slow and they may refer to the presence of different kinds of faults in the system. Thus, detecting these changes reliably and quickly plays an essential role in making the binary decision whether the system is in faulty or normal operation mode. The core of the change detection algorithms used in this work were taken from [2] which became a fundamental reference for engineers and researchers involved in fault detection, diagnosis and many other fields in recent years.

2.7 Evaluation of fault detection and diagnosis methods

For the evaluation of fault detection and diagnosis methods we need measurable characteristic features of the methods according to which the comparison of the solutions becomes possible. On the basis of [12] and [16] the definitions of these "indicator" properties are as follows.

Fault detection false alarms (FD false alarms) is the number of discrete time points in which the method detected some fault but no fault occured in reality.

Fault detection missed alarms (FD missed alarms) is the number of time instants in which the method didn't detect any faults, but some fault (that is detectable by the method) occured in reality.

Detection delays (DDs) is a vector containing the time delays (the number of sampling instants) passed between the actual occurence and the detection of the faults.

Mean of detection delays (Mean of DDs) is the mean of the numbers in the Detection delays vector.

Mflops is the number of million floating point operations required by a given method as it is measured by Matlab's FLOPS function [14].

2.8 Models and methods - steps of the solution

With the summary of the models and methods used in this work, the outline of the solution steps is also summarized here.

As it has been mentioned in the Introduction the objective of this work is to develop fault detection and diagnosis methods for countercurrent heat exchangers. The aim is to detect two types of faults: the deterioration of the heat transfer surface (i.e. how $CaCO_3$ settles on and leaves the surface) and the leaking of the outer container. From this it is clear that both faults to be detected occur in the process (or plant) subsystem.

Since reliable dynamic models as well as enough operational experience concerning heat exchangers are available from the literature and engineering practice, the application of model-based methods to solve the task is a reasonable choice.

The design issues of the task can be examined now on the basis of Section 2.3. Of course, rapid response and low noise sensitivity are important goals in this case. The tradeoff between them can be handled quite successfully by using the known properties of the process and setting the appropriate tuning knobs of the methods. Since the process that takes place in heat exchangers is rather slow compared to certain rapidly changing mechanical or electrical systems, the implementation of the developed fault detection methods to microprocessor-based digital systems practically does not cause any degradation in system performance. Because of their simple implementation, rapidness, small computational complexity and low storage requirements, the application of recursive identification algorithms was chosen.

In order to complete our task we need mathematical models that describe the operation of the heat exchanger and the two faults to be detected. The continuous time heat exchanger model (as it is described in section 3) is a white box model that is in the form of ordinary differential equations. In this model, we know the physical meaning of each parameter. The fault model of leaking is also a white box model. The fault model of the settling $CaCO_3$ is based on prior heuristic knowledge of the phenomena where we know which parameter carries the diagnostic information, therefore this is a grey box model.

From the models we know which parameters are to be estimated in order to detect the faults, meaning that we encounter a model parameter estimation problem at this step, which is solved by employing various recursive algorithms.

After designing the identification methods, the next step is to choose and design the procedures that detect changes in the estimated parameters. The actual fault detection and diagnosis decisions are made on the basis of the results of change detection.

The next step is the implementation and, finally, the evaluation of the fault detection methods according to the properties listed in Section 2.7.

3 Models

The purpose of this section is to present the mathematical model of the countercurrent heat exchanger that is needed to simulate its operation and develop the parameter estimation and fault detection algorithms. Moreover, the semi-empirical models of the faults will also be discussed here.

3.1 Heat exchanger model

3.1.1 The heat exchanger from a modelling point of view

Heat exchangers are widely used in the process industries and other areas. They are usually arranged in units containing several (5-200) of them. From a modelling point of view, heat exchangers are rather simple and well investigated operating units for which we have both reliable dynamic models with known properties and accumulated operational experience. This experience can be used in formulating suitable dynamic models of the typical faults that can be encountered in heat exchangers.

The so-called 'tube-in-tube' type heat exchangers are classified into two groups based on the flowing directions of the fluids flowing in their inner and outer tubes (see Fig. 1):

- In *countercurrent heat exchangers* the fluids in the inner and outer tube are flowing in the opposite directions.
- In the case of *co-current heat exchangers* the two fluids are flowing in the same direction.

Usually the outer tube contains the fluid used for cooling and will be referred as 'the cold side' while the fluid for heating flows in the inner tube which will be referred as 'the hot side'. A countercurrent heat exchanger is modelled as a sequence of cascaded cells [10]. Let us investigate first what assumptions are taken as starting points and how we derive the mathematical model of a single heat exchanger cell using certain physical laws.

3.1.2 The model of a single heat exchanger cell

We assume that each heat exchanger cell consists of two perfectly stirred tanks with in- and outflows. The two tanks are connected by a heat transfer area between

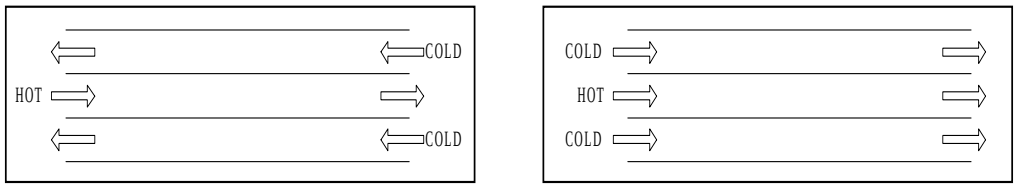

Figure 1: Schematic structure of counter- and co-current heat exchangers

them. Furthermore, it is assumed that the liquid level both on the cold and hot sides is controlled in such a way that the inlet and outlet flow rates are equal on the hot and cold side respectively. The schematic structure of one heat exchanger cell is shown in Fig. 2.a. The model equations for a single cell consist of the total mass and energy balances for the hot and cold sides respectively.

Cold side The change of the cold side volume is given by the following equation

$$\frac{dV_c(t)}{dt} = v_{ci}(t) - v_{co}(t) - v_{cl}(t) \tag{1}$$

where V_c denotes the volume of the liquid flowing on the cold side, v_{ci} and v_{co} are the cold side inlet and outlet flow rates and v_{cl} is the flow rate of the liquid flowing out from the cold side of the heat exchanger through a leak. Since, according to our assumptions, the inlet and outlet flow rates are equal (i.e. $v_{ci}(t) = v_{co}(t) = v_c(t)$) in order to keep the liquid level constant, we can write (1) as

$$\frac{dV_c(t)}{dt} = -v_{cl}(t) \tag{2}$$

The energy balance of the cold side of a single heat exchanger cell can be written as

$$\begin{aligned}\frac{d(c_{pc}\rho_c V_c(t)T_{co}(t))}{dt} &= v_c(t)\rho_c c_{pc}T_{ci}(t) - \\ v_c(t)\rho_c c_{pc}T_{co}(t) &- v_l(t)\rho_c c_{pc}T_{co}(t) + U(t)A(T_{ho}(t) - T_{co}(t))\end{aligned} \tag{3}$$

Where T_{ci}, T_{co} and T_{ho} denote the cold side inlet, outlet and hot side outlet temperatures of the heat exchanger cell respectively, ρ_c and c_{pc} are the density and the specific heat of the liquid flowing on the cold side respectively, A is the heat transfer area of the cell and U is the HTC.

After simplifying by c_{pc} and ρ_c we get

$$\frac{d(V_c(t)T_{co}(t))}{dt} = v_c(t)T_{ci}(t) - v_c(t)T_{co}(t) - v_l(t)T_{co}(t) + \frac{U(t)A}{c_{pc}\rho_c}(T_{ho}(t) - T_{co}(t)) \tag{4}$$

from which we obtain

$$T_{co}(t)\frac{dV_c(t)}{dt} + V_c(t)\frac{dT_{co}(t)}{dt} = \\ = v_c(t)T_{ci}(t) - v_c(t)T_{co}(t) - v_l(t)T_{co}(t) + \frac{U(t)A}{c_{pc}\rho_c}(T_{ho}(t) - T_{co}(t)) \quad (5)$$

Writing (2) into (5) gives the final form of the energy balance equation of the cold side, namely

$$\frac{dT_{co}(t)}{dt} = \frac{v_c(t)}{V_c(t)}(T_{ci}(t) - T_{co}(t)) + \frac{U(t)A}{c_{pc}\rho_c V_c(t)}(T_{ho}(t) - T_{co}(t)) \quad (6)$$

Hot side In our model we assume that no leaking occurs on the hot side i.e. the volume on the hot side is constant. Therefore we can write the energy balance of the hot side easily

$$\frac{d(c_{ph}\rho_h V_h T_{ho}(t))}{dt} = \\ = v_h(t)\rho_h c_{ph}T_{hi}(t) - v_h(t)\rho_h c_{ph}T_{ho}(t) + U(t)A(T_{co}(t) - T_{ho}(t)) \quad (7)$$

Where T_{hi} is the hot inlet temperature of the cell, V_h, v_h, ρ_h and c_{ph} are the volume, flow rate, density and specific heat of the liquid flowing on the hot side respectively. From which it follows that

$$\frac{dT_{ho}(t)}{dt} = \frac{v_h(t)}{V_h}(T_{hi}(t) - T_{ho}(t)) + \frac{U(t)A}{c_{ph}\rho_h V_h}(T_{co}(t) - T_{ho}(t)) \quad (8)$$

3.1.3 The cascade model

As it has been mentioned in Section 3.1 we model the countercurrent heat exchanger as a number of cascaded cells. The dynamics of such a cell have been described in section 3.1.2. Obviously, the more cells we use to describe the heat exchanger the better we approximate its real behaviour. It was shown that a model consisting of 3-5 cells is enough to describe the dynamics of a heat exchanger for fault diagnosis purposes in the case of the majority of the industrial equipments [13]. As we have seen, the model equations for a single cell consist of the total mass and energy balances for the hot and cold sides. In the cascade model constant volume (and hence mass) is also assumed on the hot side. If the whole heat exchanger is divided into n cells, the energy balance equations of jth cell of the hot and cold side respectively are as follows:

$$\frac{dT_{jc}(t)}{dt} = \frac{v_c(t)}{V_{jc}(t)}(T_{(j+1)c}(t) - T_{jc}(t)) + \frac{U_j(t)A_j}{c_{pc}\rho_c V_{jc}(t)}(T_{jh}(t) - T_{jc}(t)) \quad (9)$$

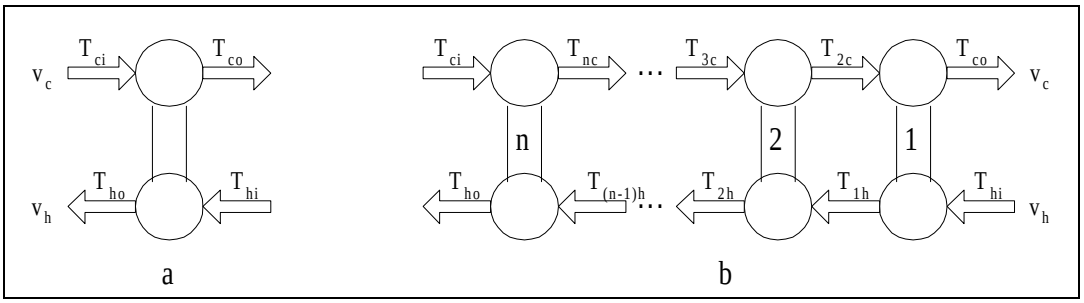

Figure 2: a. Single heat exchanger cell; b. Cascade model of a countercurrent heat exchanger

$$\frac{dT_{jh}(t)}{dt} = \frac{v_h(t)}{V_{jh}}(T_{(j-1)h}(t) - T_{jh}(t)) + \frac{U_j(t)A_j}{c_{ph}\rho_h V_{jh}}(T_{jc}(t) - T_{jh}(t)) \tag{10}$$

The mass balance equation of the jth cell is written as

$$\frac{dV_{jc}(t)}{dt} = -v_{jl}(t) \tag{11}$$

$$v_{jl} = \frac{v_{cl}}{n} \tag{12}$$

$$j = 1, 2, \ldots, n$$

where Eq.(12) does not mean that we have n leaks of equal size along the heat exchager in reality. It represents that the liquid level decreases equally in every cell if leakage occurs.

Furthermore, the following euqalities hold for the hot and cold side input and output temperatures

$$T_{(n+1)c} = T_{ci},\ T_{1c} = T_{co},\ T_{0h} = T_{hi},\ T_{nh} = T_{ho} \tag{13}$$

where the subscripts h and c denote the hot and cold sides respectively. The model above can be regarded as a lumped version of a distributed parameter model of the heat exchanger. The cells are therefore associated with a particular spatial location along the equipment length. The structure of the cascade model consisting of n cells with the numbered cells and temperatures is shown in Fig. 2.b. The state variables of the above model are the temperatures of the cold and hot sides and the cold side volume in the cells, i.e. $\{T_{1h}, \ldots, T_{nh}; T_{1c}, \ldots, T_{nc}; V_{1c}, \ldots, V_{nc}\}$. If an equidistant space lumping is performed, i.e. the parameters A_j and V_{jh} are the same for all cells (i.e. $j = 1, 2, \ldots, n$), then the heat transfer coefficients $U_1, U_2, \ldots U_n$ carry spatial information. This means that the more cells we divide the heat exchanger

into, the more accurately we can determine the physical location of the fault in connection with the HTC. On the other hand, the exact localization of faults requires temperature measurements of the hot and cold side from all along the heat exchanger that are rarely available in practice. In our case this tradeoff somewhat reduces the practical importance of fault detection methods that require all temperature measurements for the model.

3.1.4 The simplified model

A simplified dynamic model containing only a single cell can be safely used for the fault detection of the equipment. Note that on the price of less measurement points and data required we lose all spatial information about the HTC. It has to be added that in most practical cases it is not a very serious loss of information because the safe detection and identification of the fault is more important and this can be granted even in this case.

With only a single cell as in Fig. 2.a describing the heat exchanger, its outlet temperatures T_{ho} and T_{co} are the same as the outlet temperatures of one cell. In this case we only have three state variables $\{T_{co}, T_{ho}, V_c\}$ and a single triplet of Eqns.(9)-(13).

3.2 The discrete time model

Let us now convert the model described in (1), (6) and (8) into a discrete time model which is suitable for tracking the HTC and the cold side volume. Let t_s denote the sampling interval and introduce the variables

$$v_c(k) = v_c(kt_s), V_c(k) = V_c(kt_s), T_{ci}(k) = T_{ci}(kt_s), T_{co}(k) = T_{co}(kt_s) \qquad (14)$$

and similarly for the hot side. Furthermore, let $U(k) = U(kt_s)$. Using a simple Euler approximation for the derivatives the discrete time model can be written as

$$\frac{T_{co}(k+1) - T_{co}(k)}{t_s} = \frac{v_c(k)}{V_c(k)}(T_{ci}(k) - T_{co}(k)) + \frac{U(k)A}{c_{pc}\rho_c V_c(k)}(T_{ho}(k) - T_{co}(k)) \qquad (15)$$

$$\frac{T_{ho}(k+1) - T_{ho}(k)}{t_s} = \frac{v_h(k)}{V_h}(T_{hi}(k) - T_{ho}(k)) + \frac{U(k)A}{c_{ph}\rho_h V_h}(T_{co}(k) - T_{ho}(k)) \qquad (16)$$

$$\frac{V_c(k+1) - V_c(k)}{t_s} = -v_{cl}(k) \qquad (17)$$

Using the δ operator [8] which is defined as

$$\delta T(k) = \frac{(T(k+1) - T(k))}{t_s} \tag{18}$$

we can write the discrete time model as

$$\delta T_{co}(k) = \frac{v_c(k)}{V_c(k)}(T_{ci}(k) - T_{co}(k)) + \frac{U(k)A}{c_{pc}\rho_c V_c(k)}(T_{ho}(k) - T_{co}(k)) \tag{19}$$

$$\delta T_{ho}(k) = \frac{v_h(k)}{V_h}(T_{hi}(k) - T_{ho}(k)) + \frac{U(k)A}{c_{ph}\rho_h V_h}(T_{co}(k) - T_{ho}(k)) \tag{20}$$

$$\delta V_c(k) = -v_{cl}(k) \tag{21}$$

Let us introduce the variables

$$\bar{v}_c(k) = \frac{v_c(k)}{V_c(k)}, \qquad \tau_c(k) = \frac{U(k)A}{c_{pc}\rho_c V_c} \tag{22}$$

for the cold side and

$$\bar{v}_h(k) = \frac{v_h(k)}{V_h}, \qquad \tau_h(k) = \frac{U(k)A}{c_{ph}\rho_h V_h} \tag{23}$$

for the hot side.

3.2.1 Discrete time filters

In order to avoid numerical differentiation of the noisy temperature measurements we filter them through a first or second order linear Infinite Impulse Response (IIR) filter[1]. The choice of the order of the filter will depend on the particular fault detection algorithm in which the measurement data are used. Let us denote the filtered temperature measurements from now on by T^f_{xy} where xy can be any of the subscripts *ci*, *co*, *hi* or *ho*. Using the filtered temperature measurements, the equations above take the form

$$\delta T^f_{co}(k) = \bar{v}_c(k)(T^f_{ci}(k) - T^f_{co}(k)) + \tau_c(k)(T^f_{ho}(k) - T^f_{co}(k)) \tag{24}$$

$$\delta T^f_{ho}(k) = \bar{v}_h(k)(T^f_{hi}(k) - T^f_{ho}(k)) + \tau_h(k)(T^f_{co}(k) - T^f_{ho}(k)) \tag{25}$$

The above equations are also good approximations when $\bar{v}_c, \bar{v}_h, \tau_c$ and τ_h are slowly varying.

[1] The input (x) and output (y) of such a system satisfy the following difference equation
$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$
where a_i, $i = 1, \ldots, N$ and b_j, $j = 0, \ldots, M$ are the filter coefficients and N is the order of the filter.

3.3 Fault modelling

3.3.1 Deterioration of the heat exchanger surface

Under normal operating conditions the HTC is constant or slowly decreasing due to a layer of settled material building up on the heat transfer surface (see Fig. 3.a). In old heat exchangers large pieces of the settled material can break away from the surface, causing damage in the equipment. When the settled material breaks off, the HTCs will undergo abrupt positive jumps. These jumps can be grouped into two qualitatively different types.

1. In the beginning when the jumps start to occur, they are small and infrequent because of the small pieces of $CaCO_3$ that leave the heat transfer surface (see Fig. 3.b).

2. Then in the later stages the jumps become large and frequent when bigger 'stones' of $CaCO_3$ come off (see Fig. 3.c).

The usual industrial practice is to wash heat exchangers with some kind of acid between certain time intervals. However, this washing procedure requires the heat exchanger and possibly other operating units to be stopped and thus it may become a rather expensive operation. On the other hand, if some information is available about the process of the changing of the HTC, then we are able to avoid unnecessary stoppings and damage caused by the settled material coming off the heat transfer surface, too. Thus, one of the main ideas in this work is to track the HTC in the heat exchanger and detect abrupt positive jumps in it.

3.3.2 Leakage

When the outer container is leaking, there is an unmeasurable outflow v_{cl} on the cold side from the heat exchanger (see Eq.2 and Fig. 4) which may result in a slow decrease of the mass in the outer phase, or - if the levels are controlled, in the decrease of the outflow from the equipment. Here it is assumed that the volume flow rates v_h and v_c are known and constant or measured, leaking can therefore be detected via detection of a slow decrease in the cold side volume V_c. Note that we want to detect small changes in the cold side volume compared to the initial value of the volume in the heat exchanger. This is a reasonable aim because the detection

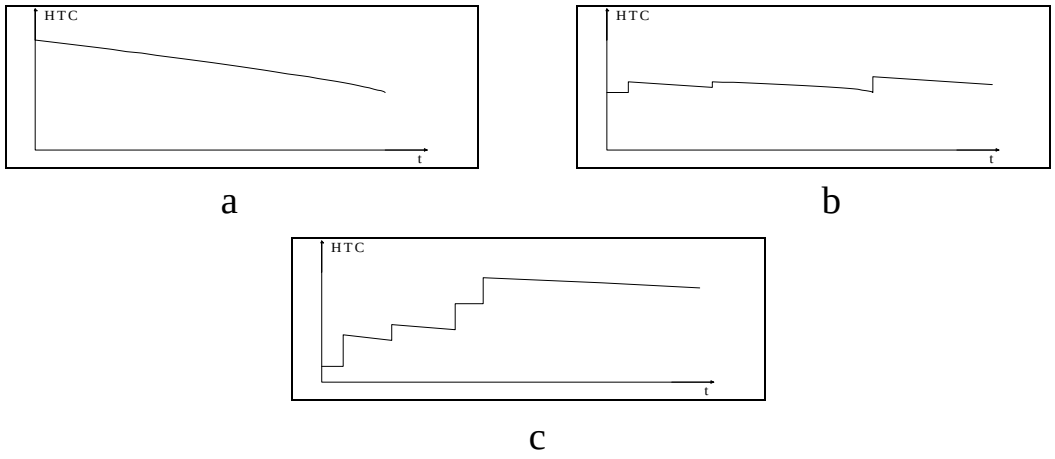

Figure 3: Three consecutive stages of the change of the heat transfer coefficient: a. Constant or slowly decreasing HTC; b. Smaller and infrequent jumps; c. Large and frequent jumps

of large and quick changes (i.e. big leaks) can be a simpler task with other methods. Moreover, there is no possibility to get spatial information about the leakage even with our cascade model, since the liquid level on the cold side decreases uniformly in the whole heat exchanger. That is why it is important to develop such a leakage detection algorithm that requires as few temperature measurement data as possible.

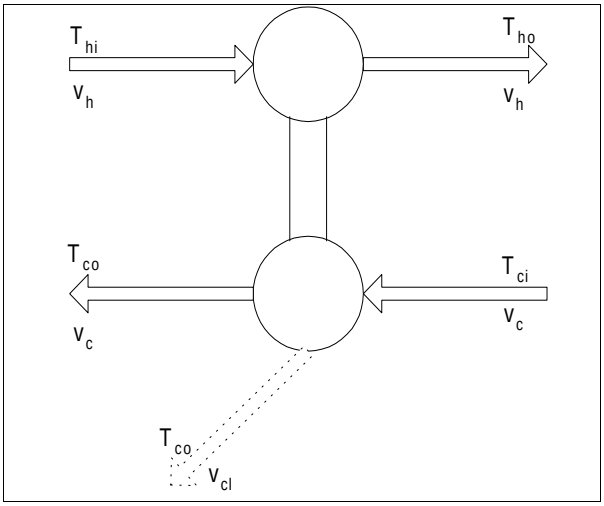

Figure 4: Leaking modelled as an unmeasurable outflow from the cold side

4 Algorithms

The purpose of this section is to present the developed fault detection algorithms. Firstly, the question of filtering the measured temperature data will be discussed. Then different variants of the tracking algorithms will be presented for both parameters (i.e. the HTC and the cold side volume). These solutions will be grouped according to the availability of temperature measurement data. Within each group, algorithms will be shown for the independent estimation of the parameters and, where it is possible, for the simultaneous tracking. Finally, the fault detection and diagnosis methods based on the estimated parameters are going to be presented.

4.1 Filtering the measurement data

As it was mentioned in Section 3.2 we filter the temperature measurements through a linear IIR filter. Filtering is needed to avoid numerical differentiation of the noisy temperature measurements, because the forthcoming parameter estimation methods require the approximation of the first and second derivatives of certain measured signals.

Numerical differentiation is sensitive to noise and round-off errors (particularly when the sampling interval is short), and large errors in the differentiation can occur, which may lead to non-robust identification methods. That is, the methods may fail

or have a very slow convergence in the presence of noise. The question is how can we avoid taking derivatives. Assume we have an nth order discrete time system in the delta-operator (obtained from a continuous time system), i.e.

$$A(\delta)y(k) = B(\delta)u(k) \tag{26}$$

where the highest order occuring in the A and B polynomials is n. This means that the nth order derivative of the signals is involved. One obvious way of avoiding taking this derivative is to integrate both sides of the equation n times, that is we filter the signals through the filter $\frac{1}{\delta^n}$. The system equation then reads

$$\frac{1}{\delta^n}A(\delta)y(k) = \frac{1}{\delta^n}B(\delta)u(k) \tag{27}$$

and we see that $\frac{1}{\delta^n}A(\delta)$ and $\frac{1}{\delta^n}B(\delta)$ are now polynomials in $\frac{1}{\delta}$, i.e. we are now integrating the signals instead of taking derivatives. The filter $\frac{1}{\delta^n}$ is just one example of a possible filter, but we have to integrate at least n times in order to avoid the derivatives which means that the filter must be at least of order n (see [8]). This means that if a particular parameter estimation algorithm requires the first or the second derivative of certain signals then a first or second order filter is applied on the measurement data respectively.

The idea of the IIR filter design in our case is that a continuous time filter is designed first using the continuous time model of the heat exchanger and then the obtained filter is transformed into discrete time using the δ operator. The general method for constructing such a filter is as follows [8]. Assume that a continuous time system is given by the following input-output model

$$\frac{Y(s)}{U(s)} = \frac{B(s)}{A(s)} \tag{28}$$

and assume that the system is strictly proper (i.e. the order of $A(s)$ is greater than the order of $B(s)$). Then the filter that should be used is $\frac{1}{A(s)}$. After applying this filter on the system we get

$$Y(s) = \frac{B(s)}{A(s)}U(s) \tag{29}$$

To obtain the discrete time filter, we substitute s in (29) for the δ operator.

On the basis of the guidelines above, the filters that will be used in the fault detection methods are the following.

First order filter. In this case we want to use a first order filter with cutoff frequency $\omega_c = e_0$. The transfer function describing such a filter is given by

$$\frac{F^f(s)}{F(s)} = \frac{e_0}{s + e_0} \tag{30}$$

where $F(s)$ and $F^f(s)$ are the Laplace transforms of the signal to be filtered and the filtered signal itself respectively. The discrete time version of this filter is written as

$$\frac{f^f(k)}{f(k)} = \frac{e_0}{\delta + e_0} \tag{31}$$

from which we get

$$\delta f^f(k) = -e_0 f^f(k) + e_0 f(k) \tag{32}$$

To calculate the appropriate filter coefficient in this case, let us select from Eqs. (9) and (10) the one that determines the dominant time constant of one heat exchanger cell. Let us assume that this equation is Eq. (9), and that the cold side liquid volume and the HTC in the heat exchanger are constant (i.e. we assume normal operation). Let us write the Laplace transform of Eq. (9) in the following form.

$$T_{jc}(s)(s + \frac{v_c}{V_{jc}} + \frac{U_j A_j}{c_{pc}\rho_c V_{jc}}) = \frac{v_c}{V_{jc}} T_{(j+1)c}(s) + \frac{U_j A_j}{c_{pc}\rho_c V_{jc}} T_{jh}(s), j = 1, \ldots, n \tag{33}$$

According to the previously discussed filter design method, the first order filter coefficient is as follows.

$$e_0 = \frac{v_c}{V_{jc}} + \frac{U_j A_j}{c_{pc}\rho_c V_{jc}} \tag{34}$$

where V_{jc} and U_j denote the nominal values of the cold side liquid volume and the HTC in the jth cell of the heat exchanger respectively.

Second order filter. Again we start from the transfer function of a continuos time second order linear filter which reads

$$\frac{F^f(s)}{F(s)} = \frac{e_0}{s^2 + e_1 s + e_0} \tag{35}$$

Converting it into discrete time gives

$$\frac{f^f(k)}{f(k)} = \frac{e_0}{\delta^2 + e_1 \delta + e_0} \tag{36}$$

from which we get

$$\delta^2 f^f(k) + e_1 \delta^2 f^f(k) + e_0 f^f(k) = e_0 f(k). \tag{37}$$

The calculation of the second order filter's parameters from the heat exchanger's parameters will be shown in Section 4.2.2.

4.2 Estimating the parameters

4.2.1 All measurements available

In this ideal and not too realistic case we assume that we have access to both the hot and cold side temperatures in each cell of the heat exchanger in addition to the inlet temperatures. In other words, we know the temperature distribution along the heat exchanger on both the cold and the hot sides. Furthermore, it is assumed that A, c_{pc}, c_{ph}, ρ_c, ρ_h, V_h are known and that we can measure the flow rates v_c and v_h. It must be admitted that it is not often possible to measure the cold and especially the hot side temperature on several points along the heat exchanger. On the other hand, we have a good reason to expect that we obtain the most easily implementable algorithms in this case that produce the most reliable estimates for the parameters.

Estimating the HTC. An additional assumption in this case is that the cold side volume is constant and known in the heat exchanger since the purpose of the algorithm discussed here is only to give an estimate for the HTC. An algorithm based on the recursive least squares method is described in a very detailed way in [11] for this case. Therefore, a different method that makes use of the gradient method will be presented in this section as another possible alternative. In [11] τ_{jc} and τ_{jh} are estimated by using the recursive least squares method and then the estimates for U_{jc} and U_{jh} are computed from $\hat{\tau}_{jc}$ and $\hat{\tau}_{jh}$ using the known proportionality constants.

Using the gradient method with a forgetting factor $0 < \lambda < 1$, it is possible to obtain estimates directly for both U_{jc} and U_{jh} using the discrete time model described in Section 3.2. The forgetting factor means that old measurements are weighted at an exponentially decreasing rate. (see [6]). On the basis of Eqns.(24) and (25) we can easily write the discrete time model of the jth cell in the heat exchanger.

$$\delta T^f_{jc}(k) = \frac{v_c(k)}{V_{jc}}(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j(k)A_j}{c_{pc}\rho_c V_{jc}}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{38}$$

$$\delta T^f_{jh}(k) = \frac{v_h(k)}{V_{jh}}(T^f_{(j-1)h}(k) - T^f_{jh}(k)) + \frac{U_j(k)A_j}{c_{ph}\rho_h V_{jh}}(T^f_{jc}(k) - T^f_{jh}(k)) \tag{39}$$

$$j = 1, 2, \ldots, n$$

One can see from (38) and (39) that the only unknown parameter in this model is the HTC, i.e. U_j that carries the diagnostic information about the deterioration of

the heat transfer surface in the jth cell. Recall that in this case it is assumed that all the hot and cold side input and output temperatures, the hot and cold side flow rates (i.e. v_c and v_h) are measurable and all the parameters of the heat exchanger are known and constant except U_j.

The prediction errors for (38) and (39) in the jth cell are given by

$$\epsilon_{jc}(k) = \delta T^f_{jc}(k) - \frac{v_c(k)}{V_{jc}}(T^f_{(j+1)c}(k) - T^f_{jc}(k) - \frac{U_j(k)A_j}{c_{pc}\rho_c V_{jc}}(T^f_{jh}(k) - T^f_{jc}(k)) \quad (40)$$

$$\epsilon_{jh}(k) = \delta T^f_{jh}(k) - \frac{v_h(k)}{V_{jh}}(T^f_{(j-1)h}(k) - T^f_{jh}(k) - \frac{U_j(k)A_j}{c_{ph}\rho_h V_{jh}}(T^f_{jc}(k) - T^f_{jh}(k)) \quad (41)$$

In order to estimate $U_j(k)$ the negative gradient of (40) and (41) with respect to $U_j(k)$ has to be calculated

$$\psi_{jc}(k) = -\frac{\partial \epsilon_{jc}(k)}{\partial U_j(k)} = \frac{A_j}{c_{pc}\rho_c V_{jc}}(T^f_{jh}(k) - T^f_{jc}(k)) \quad (42)$$

$$\psi_{jh}(k) = -\frac{\partial \epsilon_{jh}(k)}{\partial U_j(k)} = \frac{A_j}{c_{ph}\rho_h V_{jh}}(T^f_{jc}(k) - T^f_{jh}(k) \quad (43)$$

Since we don't know the real value of U_j we can approximate ϵ_{jc} and ϵ_{jh} in the following way (see [6]).

$$\hat{\epsilon}_{jc}(k) = \delta T^f_{jc}(k) - \frac{v_c(k)}{V_{jc}}(T^f_{(j+1)c}(k) - T^f_{jc}(k) - \frac{\hat{U}_{jc}(k-1)A_j}{c_{pc}\rho_c V_{jc}}(T^f_{jh}(k) - T^f_{jc}(k)) \quad (44)$$

$$\hat{\epsilon}_{jh}(k) = \delta T^f_{jh}(k) - \frac{v_h(k)}{V_{jh}}(T^f_{(j-1)h}(k) - T^f_{jh}(k) - \frac{\hat{U}_{jh}(k-1)A_j}{c_{ph}\rho_h V_{jh}}(T^f_{jc}(k) - T^f_{jh}(k)) \quad (45)$$

The standard recursive scheme for estimating U_j from the cold and hot side respectively is as follows

$$\hat{U}_{jc}(k) = \hat{U}_{jc}(k-1) + p_{jc}(k)\psi_{jc}(k)\hat{\epsilon}_{jc}(k) \quad (46)$$

$$p_{jc}(k) = \frac{p_{jc}(k-1)}{\lambda_{jc} + \psi^2_{jc}(k)p_{jc}(k-1)} \quad (47)$$

$$\hat{U}_{jh}(k) = \hat{U}_{jh}(k-1) + p_{jh}(k)\psi_{jh}(k)\hat{\epsilon}_{jh}(k) \quad (48)$$

$$p_{jh}(k) = \frac{p_{jh}(k-1)}{\lambda_{jh} + \psi^2_{jh}(k)p_{jh}(k-1)} \quad (49)$$

The choice of λ in (47) and (49) is a tradeoff between good tracking capabilities of the HTCs and sensitivity to noise and unmodelled dynamics. A small value of λ means that the estimate will track variations in the HTC quicker, but on the other hand, the it will also become more sensitive to noise and unmodelled dynamics.

The actual estimates of the HTCs in the jth cell can be computed by using a convex combination:

$$\hat{U}_j(k) = \alpha\hat{U}_{jc}(k) + (1-\alpha)\hat{U}_{jh}(k), \quad 0 \leq \alpha \leq 1 \tag{50}$$

where α should reflect the relative confidence we have in the two estimates.

Estimating the cold side volume. The only difference between the assumptions in this and the previous case is that now the HTC in each cell is assumed to be known and constant and the cold side volume becomes a time-varying parameter. In most cases this is a reasonable assumption, since the change of the HTC is usually much slower than that of the cold side volume during leakage. According to these assumptions the discrete time form of the cold side energy balance equation of the jth cell is given by

$$\delta T^f_{jc}(k) = \frac{v_c(k)}{V_{jc}(k)}(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j A_j}{c_{pc}\rho_c V_{jc}(k)}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{51}$$

In addition, the following equality holds if the cells within the heat exchanger are of the same size

$$V_{jc}(k) = \frac{V_c(k)}{n} \tag{52}$$

where $V_c(k)$ denotes the overall cold side fluid volume in the heat exchanger and n is the number of cells the heat exchanger is divided into.

For the purpose of tracking the cold side fluid volume let us introduce the following variable

$$V_{jcr}(k) = \frac{1}{V_{jc}(k)} \tag{53}$$

Thus we can rewrite (51) as

$$\delta T^f_{jc}(k) = V_{jcr}(k)v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j A_j V_{jcr}(k)}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{54}$$

Starting from (54) let us now present two possible identification schemes for the tracking of the cold side fluid volume.

Tracking with the gradient algorithm. Using (54) the prediction error is written as

$$\epsilon_j(k) = \delta T^f_{jc}(k) - V_{jcr}(k)v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) - \frac{U_j A_j V_{jcr}(k)}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{55}$$

Its negative gradient with respect to $V_{jcr}(k)$ is given by

$$\psi_j(k) = v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j A_j}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{56}$$

The approximation of $\epsilon_j(k)$ is written as

$$\hat{\epsilon}_j(k) = \delta T^f_{jc}(k) - V_{jcr}(k-1)v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) - \frac{U_j A_j V_{jcr}(k-1)}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{57}$$

Using (57) and (56) the recursive scheme for the estimation of the cold side fluid volume can be written as

$$\hat{V}_{jcr}(k) = \hat{V}_{jcr}(k-1) + p_j(k)\psi_j(k)\epsilon_j(k) \tag{58}$$

$$\hat{V}_{jc}(k) = \frac{1}{\hat{V}_{jcr}(k)} \tag{59}$$

$$p_j(k) = \frac{p_j(k-1)}{\lambda_j + \psi^2_j(k)p_j(k-1)} \tag{60}$$

where λ_j is again a tradeoff between quick tracking capabilities of the parameter and sensitivity to noise.

Tracking with the recursive least squares algorithm. To apply the recursive least squares method for estimating the cold side volume, (54) should be rewritten as

$$\delta T^f_{jc}(k) = V_{jcr}(k)\left[v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j A_j}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k))\right]. \tag{61}$$

Let us introduce the following notations

$$y_{jc}(k+1) = \delta T^f_{jc}(k) \tag{62}$$

$$u_{jc}(k) = v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j A_j}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{63}$$

Then (61) can be written as

$$y_{jc}(k+1) = V_{jcr}(k)u_{jc}(k) \tag{64}$$

Using the model (64) the estimates for V_{jc} can be computed by the following algorithm

$$\hat{V}_{jcr}(k) = \hat{V}_{jcr}(k-1) + p_{jc}(k)u_{jc}(k-1)(y_{jc}(k) - \hat{V}_{jcr}(k-1)u_{jc}(k-1)) \tag{65}$$

$$\hat{V}_{jc}(k) = \frac{1}{\hat{V}_{jcr}(k)} \tag{66}$$

$$p_{jc}(k) = \frac{p_{jc}(k-1)}{\lambda_{jc} + u^2_{jc}(k-1)p_{jc}(k-1)} \tag{67}$$

where the role of λ_{jc} is exactly the same as in Eqns.(58)-(60).

Simultaneous estimation. In this case both the HTC and the cold side volume are unknown parameters and we would like to estimate them simultaneously. An obvious strategy to achieve this goal is to combine the two algorithms described in Eqns.(42)-(50) and (56)-(60) respectively. Within the jth heat exchanger cell we have two parameters to be estimated in this case, namely V_{jc} and U_j. The idea of the solution is to let the two gradient algorithms use each other's estimations. This approach works very well if both V_{jc} and U_j vary slowly in time, but when abrupt jumps occur in the HTC, problems arise with the estimation of the cold side liquid volume. These problems can be handled successfully if we utilize all the information we have from the complete availability of measurement data.

The suitable form of the discrete time energy balance equation of the cold side in this case is written as

$$\delta T^f_{jc}(k) = \frac{v_c(k)}{V_{jc}(k)}(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{U_j(k)A_j}{c_{pc}\rho_c V_{jc}(k)}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{68}$$

while Eqn.(39) describing the hot side can be used here without change. After combining the algorithms described in Eqns.(42)-(50) and (56)-(60) we get the following procedure for the simultaneous estimation of the cold side fluid volume and the HTC in the jth cell

$$\psi_{jUcc}(k) = \frac{A_j}{c_{pc}\rho_c \hat{V}_{jc}(k-1)}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{69}$$

$$\psi_{jUch}(k) = \frac{A_j}{c_{ph}\rho_h V_{jh}}(T^f_{jc}(k) - T^f_{jh}(k)) \tag{70}$$

$$\hat{\epsilon}_{jUcc}(k) = \delta T^f_{jc}(k) - \frac{v_c(k)}{\hat{V}_{jc}(k-1)}(T^f_{(j+1)c}(k) - T^f_{jc}(k)) - \frac{\hat{U}_{jc}(k-1)A_j}{c_{pc}\rho_c \hat{V}_{jc}(k-1)} \tag{71}$$

$$\hat{\epsilon}_{jUch}(k) = \delta T^f_{jh}(k) - \frac{v_h(k)}{V_{jh}}(T^f_{(j-1)h}(k) - T^f_{jh}(k)) - \frac{\hat{U}_{jh}(k-1)A_j}{c_{pc}\rho_c V_{jh}} \tag{72}$$

$$\hat{U}_{jc}(k) = \hat{U}_{jc}(k-1) + p_{jUcc}(k)\psi_{jUcc}(k)\hat{\epsilon}_{jUcc}(k) \tag{73}$$

$$p_{jUcc}(k) = \frac{p_{jUcc}(k-1)}{\lambda_{jUcc} + \psi^2_{jUcc}(k)p_{jUcc}(k-1)} \tag{74}$$

$$\hat{U}_{jh}(k) = \hat{U}_{jh}(k-1) + p_{jUch}(k)\psi_{jUch}(k)\hat{\epsilon}_{jUch}(k) \tag{75}$$

$$p_{jUch}(k) = \frac{p_{jUch}(k-1)}{\lambda_{jUch} + \psi^2_{jUch}(k)p_{jUch}(k-1)} \tag{76}$$

$$\hat{U}_j(k) = \alpha U_{jc}(k) + (1-\alpha)U_{jh}(k), \alpha \in [0,1] \tag{77}$$

$$\psi_{jVc}(k) = v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) + \frac{\hat{U}_j(k-1)A_j}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{78}$$

$$\hat{\epsilon}_{jVc}(k) = \delta T^f_{jc}(k) - \hat{V}_{jcr}(k-1)v_c(k)(T^f_{(j+1)c}(k) - T^f_{jc}(k)) - \frac{\hat{U}_j(k-1)A_j\hat{V}_{jcr}(k-1)}{c_{pc}\rho_c}(T^f_{jh}(k) - T^f_{jc}(k)) \tag{79}$$

$$\hat{V}_{jcr}(k) = \hat{V}_{jcr}(k-1) + p_{jVc}(k)\psi_{jVc}(k)\hat{\epsilon}_{jVc}(k) \tag{80}$$

$$\hat{V}_{jc}(k) = \frac{1}{\hat{V}_{jcr}(k)} \tag{81}$$

$$p_{jVc}(k) = \frac{p_{jVc}(k-1)}{\lambda_{jVc} + \psi^2_{jVc}(k)p_{jVc}(k-1)} \tag{82}$$

The estimate for the cold side liquid volume in the whole heat exchanger is given by the sum of the estimates of the cells, i.e.

$$\hat{V}_c(k) = \sum_{j=1}^{n} \hat{V}_{jc}(k) \tag{83}$$

where n denotes the number of cells in the heat exchanger. The tuning knobs of the algorithm are λ_{jUcc}, λ_{jUch} and λ_{jVc} respectively.

4.2.2 Only cold side measurements available

Estimating the HTC. The algorithm used in this case is described in [11]. The main idea of it is to eliminate the need of the hot side temperature measurements by describing the heat exchanger cells with a linear time invariant model between the sampling instants. Using this approach we can approximately calculate the hot side temperatures along the heat exchanger and estimate the HTC by processing the measured and calculated temperature data.

Estimating the cold side volume. In this section the identification algorithm will be extended to the case where only measurements of the cold side temperatures and the hot inlet temperature are available. The approach is very similar to the previous case. It is assumed that the HTC is constant. Furthermore, we assume that we can measure the hot and cold side flow rates (i.e. v_c and v_h) and we know the constant parameters of the heat exchanger (U, A, V_h, c_{pc}, ρ_c, c_{ph}, ρ_h). Thus the

only time-varying parameter which is to be estimated in order to detect leaking is the cold side fluid volume, V_c.

We can consider a cascade model consisting of three cells for the sake of simplicity, without the loss of generality. Let T_{nc} denote the cold side output temperature in cell no. n. Let τ_{nc}, τ_{nh} be the variables τ_c and τ_h in (24) and (25) for cell no. n. Thus the energy balance equations take the form

$$\frac{dT_{1h}(t)}{dt} = \bar{v}_h(t)(T_{hi}(t) - T_{1h}(t)) + \tau_{1h}(t)(T_{1c}(t) - T_{1h}(t)) \tag{84}$$

$$\frac{dT_{1c}(t)}{dt} = \bar{v}_c(t)(T_{2c}(t) - T_{1c}(t)) + \tau_{1c}(t)(T_{1h}(t) - T_{1c}(t)) \tag{85}$$

$$\frac{dT_{2h}(t)}{dt} = \bar{v}_h(t)(T_{1h}(t) - T_{2h}(t)) + \tau_{2h}(t)(T_{2c}(t) - T_{2h}(t)) \tag{86}$$

$$\frac{dT_{2c}(t)}{dt} = \bar{v}_c(t)(T_{3c}(t) - T_{2c}(t)) + \tau_{2c}(t)(T_{2h}(t) - T_{2c}(t)) \tag{87}$$

$$\frac{dT_{3h}(t)}{dt} = \bar{v}_h(t)(T_{2h}(t) - T_{3h}(t)) + \tau_{3h}(t)(T_{3c}(t) - T_{3h}(t)) \tag{88}$$

$$\frac{dT_{3c}(t)}{dt} = \bar{v}_c(t)(T_{ci}(t) - T_{3c}(t)) + \tau_{3c}(t)(T_{3h}(t) - T_{3c}(t)) \tag{89}$$

$$T_{co} = T_{1c} \tag{90}$$

$$T_{ho} = T_{3h} \tag{91}$$

The system can approximately be described by a linear time invariant model by assuming that the flow rates and the HTC are approximately constant. After Laplace transforming (84) we get (for the sake of simplicity, variable s in the argument of the Laplace transforms of $\bar{v}_c$, $\bar{v}_h$, τ_{jc} and τ_{jh} will be suppressed in the following equations).

$$sT_{1h}(s) = \bar{v}_h T_{hi}(s) - \bar{v}_h T_{1h}(s) + \tau_{1h} T_{1c}(s) - \tau_{1h} T_{1h}(s) \tag{92}$$

Since according to our assumptions, we do not have access to the hot side temperature measurements along the heat exchanger (we can only measure the hot input and output), we need to eliminate the hot side temperatures T_{1h} and T_{2h} from Eqns.(84)-(89). In order to do it we express $T_{1h}(s)$ from (92)

$$T_{1h}(s) = \frac{\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s)}{s + \bar{v}_h + \tau_{1h}}. \tag{93}$$

The Laplace transform of (85) is given by

$$sT_{1c}(s) = \bar{v}_c T_{2c}(s) - \bar{v}_c T_{1c}(s) + \tau_{1c} T_{1h}(s) - \tau_{1c} T_{1c}(s) \tag{94}$$

Writing (93) into (94) gives

$$sT_{1c}(s) = \bar{v}_c T_{2c}(s) - \bar{v}_c T_{1c}(s) + \tau_{1c} \frac{\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s)}{s + \bar{v}_h + \tau_{1h}} - \tau_{1c} T_{1c}(s) \tag{95}$$

from which it follows that

$$\left(s + \bar{v}_c - \frac{\tau_{1c}\tau_{1h}}{s + \bar{v}_h + \tau_{1h}} + \tau_{1c} \right) T_{1c}(s) = \bar{v}_c T_{2c}(s) + \frac{\tau_{1c}\bar{v}_h T_{hi}(s)}{s + \bar{v}_h + \tau_{1h}} \tag{96}$$

Similarly to (93) we can express T_{2h} from the Laplace transform of (86) in the following way

$$T_{2h}(s) = \frac{\bar{v}_h}{s + \bar{v}_h + \tau_{2h}} \frac{\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s)}{s + \bar{v}_h + \tau_{1h}} + \frac{\tau_{2h} T_{2c}(s)}{s + \bar{v}_h + \tau_{2h}} \tag{97}$$

The Laplace transform of (87) can be written as

$$(s + \bar{v}_c + \tau_{2c}) T_{2c}(s) = \bar{v}_c T_{3c}(s) + \tau_{2c} T_{2h}(s) \tag{98}$$

Substituting (97) into (98) gives the transfer function for T_{2c} which reads

$$\left(s + \bar{v}_c + \tau_{2c} - \frac{\tau_{2c}\tau_{2h}}{s + \bar{v}_h + \tau_{2h}} \right) T_{2c}(s) = \bar{v}_c T_{3c}(s) + \frac{\tau_{2c}\bar{v}_h(\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s))}{(s + \bar{v}_h + \tau_{2h})(s + \bar{v}_h + \tau_{1h})} \tag{99}$$

After taking the Laplace transform of (88) and (89), and expressing T_{3h} from the Laplace transform of (88) and substituting it into the Laplace transform of (89) we obtain the following transfer function for T_{3c}

$$\left(s + \bar{v}_c - \frac{\tau_{3c}\tau_{3h}}{s + \bar{v}_h + \tau_{3h}} + \tau_{3c} \right) T_{3c}(s) = \bar{v}_c T_{ci}(s) + \frac{\tau_{3c}\bar{v}_h \tau_{2h} T_{2c}(s)}{(s + \bar{v}_h + \tau_{3h})(s + \bar{v}_h + \tau_{2h})} + \frac{\tau_{3c}\bar{v}_h^2(\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s))}{(s + \bar{v}_h + \tau_{3h})(s + \bar{v}_h + \tau_{2h})(s + \bar{v}_h + \tau_{1h})} \tag{100}$$

Notice the pattern in the transfer functions (96),(99) and (100) as the number of cells grows. Next, it will be shown that (99) and (100) can be brought into the form (96) by introducing filtered variables.

Let us introduce the signal

$$T_{hi}^{2f}(s) = \frac{\bar{v}_h T_{hi}(s) + \tau_{1h} T_{1c}(s)}{s + \bar{v}_h + \tau_{1h}} \tag{101}$$

Then (99) can be written as

$$\left(s+\bar{v}_c+\tau_{2c}-\frac{\tau_{2c}\tau_{2h}}{s+\bar{v}_h+\tau_{2h}}\right)T_{2c}(s)=\bar{v}_cT_{3c}(s)+\frac{\tau_{2c}\bar{v}_hT_{hi}^{2f}(s)}{(s+\bar{v}_h+\tau_{2h})} \tag{102}$$

which has exactly the same form as (96). The following filtered variable is introduced

$$T_{hi}^{3f}(s)=\frac{\bar{v}_hT_{hi}^{2f}(s)+\tau_{2h}T_{2c}(s)}{s+\bar{v}_h+\tau_{2h}} \tag{103}$$

Now Eqn.(100) reads

$$\left(s+\bar{v}_c-\frac{\tau_{3c}\tau_{3h}}{s+\bar{v}_h+\tau_{3h}}+\tau_{3c}\right)T_{3c}(s)=\bar{v}_cT_{ci}(s)+\frac{\tau_{3c}\bar{v}_hT_{hi}^{3f}(s)}{s+\bar{v}_h+\tau_{3h}} \tag{104}$$

which again has the same form as (96), and the pattern continues if we consider a model with more than three cells.

Our identification strategy is now to use a discrete time version of (96) to obtain estimates for V_{1c} and thus for τ_{1c} and $\bar{v}_c$. We then filter T_{hi} and T_{1c} according to (101) using the estimates of $\bar{v}_c$ and τ_{1c} in the implementation of the filter. Then we use a discrete time version of (102) with the newly obtained filtered signal to get an estimate of τ_{2c}. We repeat the filtering and identification procedure on (103) and (104) to obtain estimate of τ_{3c}. Because of the pattern in the transfer functions, this approach of repeated identification and filtering can be extended to models with an arbitrary number of cells.

In order to obtain a discrete time version of (96) we substitute s with the delta operator. We then obtain (for the sake of simplicity we suppress the time argument of $\bar{v}_h(k)$, $\bar{v}_c(k)$, $\tau_{1c}(k)$ and $\tau_{1h}(k)$)

$$\begin{aligned}&(\delta^2+\delta(\bar{v}_h+\tau_{1h}+\bar{v}_c+\tau_{1c})+\bar{v}_c\bar{v}_h+\bar{v}_c\tau_{1h}+\tau_{1c}\bar{v}_h)\,T_{1c}(k)=\\&(\bar{v}_c\delta+\bar{v}_c(\bar{v}_h+\tau_{1h}))\,T_{2c}(k)+\tau_{1c}\bar{v}_hT_{hi}(k)\end{aligned} \tag{105}$$

To avoid numerical differentiation we filter all the temperature measurements through a second order linear filter as it was described in Section 4.1. Recall that the filter is given by the following model

$$E(\delta)=\frac{e_0}{\delta^2+e_1\delta+e_0} \tag{106}$$

In order to calculate the filter coefficients, let us rewrite Eqn.(96) in the following form

$$(s^2+(\bar{v}_h+\bar{v}_c+\tau_{1c})s+\bar{v}_c\bar{v}_h+\bar{v}_h\tau_{1c}+\bar{v}_c\tau_{1h})T_{1c}(s)= \tag{107}$$

$$(\bar{v}_c+\bar{v}_c(\bar{v}_h+\tau_{1h}))T_{2c}(s)+\tau_{1c}\bar{v}_hT_{hi}(s) \tag{108}$$

According to the filter design method discussed in Section 4.1 the second order filter coefficients are the following

$$e_0 = \bar{v}_c\bar{v}_h + \bar{v}_h\tau_{1c} + \bar{v}_c\tau_{1h} \tag{109}$$

$$e_1 = \bar{v}_h + \bar{v}_c + \tau_{1c} \tag{110}$$

Since we don't know the exact values of $\bar{v}_c$ and τ_{1c} (because they contain such a parameter (V_{1c}) that may change in time we count with the initial nominal value of V_{1c} when calculating the filter coefficients.

Let us introduce the following notations for the filtered signals

$$\begin{gathered} T^f_{hi}(k) = E(\delta)T_{hi}(k),\ T^f_{ci}(k) = E(\delta)T_{ci}(k) \\ T^f_{jh} = E(\delta)T_{jh},\ T^f_{jc} = E(\delta)T_{jc} \\ j = 1,2,3 \end{gathered} \tag{111}$$

Using the filtered measurements (105) can be written as

$$\begin{aligned} \delta^2 T^f_{1c}(k) = -(\bar{v}_h + \bar{v}_c + \tau_{1c} + \tau_{1h})\delta T^f_{1c}(k) - (\bar{v}_c\bar{v}_h + \bar{v}_c\tau_{1h} + \bar{v}_h\tau_{1c})T^f_{1c}(k) + \\ +\bar{v}_c\delta T^f_{2c}(k) + \bar{v}_c(\bar{v}_h + \tau_{1h})T^f_{2c}(k) + \tau_{1c}\bar{v}_h T^f_{hi}(k) \end{aligned} \tag{112}$$

that can be rewritten in linear regression form as

$$\delta^2 T^f_{1c}(k) = \Phi^T(k)\theta \tag{113}$$

where

$$\Phi^T(k) = \left[\delta T^f_{1c}(k), T^f_{1c}(k), \delta T^f_{2c}(k), T^f_{2c}(k), T^f_{hi}(k)\right] \tag{114}$$

and

$$\theta = \left[-(\bar{v}_h + \bar{v}_c + \tau_{1c} + \tau_{1h}), -(\bar{v}_c\bar{v}_h + \bar{v}_c\tau_{1h} + \bar{v}_h\tau_{1c}), \bar{v}_c, \bar{v}_c(\bar{v}_h + \tau_{1h}), \tau_{1c}\bar{v}_h\right]^T \tag{115}$$

The vector θ contains both known and unknown parameters. For the purpose of estimating the cold side fluid volume in the first cell τ_{1c} and $\bar{v}_c$ are written in the following way

$$\tau_{1c} = V_{1cr}\alpha_{1c} \quad \text{and} \quad \bar{v}_c = V_{1cr}v_c \tag{116}$$

where

$$V_{1cr} = \frac{1}{V_{1c}} \quad \text{and} \quad \alpha_{1c} = \frac{U_1 A_1}{c_{pc}\rho_c} \tag{117}$$

In order to estimate V_{1c} we can either estimate θ using a standard recursive least squares identification method and then calculate V_{1c} from θ, or we can as we will do here directly estimate V_{1c} from (112) using a gradient method.

The prediction error is given by

$$\begin{gathered}\epsilon(k,V_{1c}) = \delta^2 T_{1c}^f(k) - \Phi^T(k)\theta = \\ = \delta^2 T_{1c}^f(k) + (\bar{v}_h(k) + V_{1cr}(k)v_c(k) + V_{1cr}(k)\alpha_c + \tau_{1h})\delta T_{1c}^f(k) + \\ (V_{1cr}(k)v_c(k)\bar{v}_h(k) + V_{1cr}(k)v_c(k)\tau_{1h} + \bar{v}_h(k)V_{1cr}(k)\alpha_c)T_{1c}^f(k) - \\ V_{1cr}(k)v_c(k)\delta T_{2c}^f(k) - V_{1cr}(k)v_c(k)(\bar{v}_h(k) + \tau_{1h})T_{2c}^f(k) - \\ V_{1cr}(k)\alpha_c\bar{v}_h(k)T_{hi}^f(k) \end{gathered} \tag{118}$$

and its negative gradient with respect to V_{1cr} is as follows

$$\begin{aligned}\psi(k) = -\frac{\partial\epsilon(k,V_{1c})}{\partial V_{1cr}} = -(v_c + \alpha_c)\delta T_{1c}^f(k) - (v_c\bar{v}_h + v_c\tau_{1h} + \bar{v}_h\alpha_c)T_{1c}^f(k) + \\ v_c\delta T_{2c}^f(k) + v_c(\bar{v}_h + \tau_{1h})T_{2c}^f(k) + \alpha_c\bar{v}_h T_{hi}^f(k)\end{aligned} \tag{119}$$

We approximate $\epsilon(k,V_{1c})$ in the following way

$$\begin{gathered}\hat{\epsilon}(k) = \\ = \delta^2 T_{1c}^f(k) + (\bar{v}_h(k) + \hat{V}_{1cr}(k-1)v_c(k) + \hat{V}_{1cr}(k-1)\alpha_c + \tau_{1h})\delta T_{1c}^f(k) + \\ (\hat{V}_{1cr}(k-1)v_c(k)\bar{v}_h(k) + \hat{V}_{1cr}(k-1)v_c(k)\tau_{1h} + \bar{v}_h(k)\hat{V}_{1cr}(k-1)\alpha_c)T_{1c}^f(k) - \\ \hat{V}_{1cr}(k-1)v_c(k)\delta T_{2c}^f(k) - \hat{V}_{1cr}(k-1)v_c(k)(\bar{v}_h(k) + \tau_{1h})T_{2c}^f(k) - \\ \hat{V}_{1cr}(k-1)\alpha_c\bar{v}_h(k)T_{hi}^f(k) \end{gathered} \tag{120}$$

$$\tag{121}$$

A standard recursive identification scheme for V_{1cr} is now given by

$$\hat{V}_{1cr}(k) = \hat{V}_{1cr}(k-1) + p(k)\psi(k)\hat{\epsilon}(k) \tag{122}$$

$$\hat{V}_{1c}(k) = \frac{1}{\hat{V}_{1cr}(k)} \tag{123}$$

$$p(k) = \frac{p(k-1)}{\lambda + \psi^2(k)p(k-1)} \tag{124}$$

A discrete time version of (101) using filtered temperature measurements is given by

$$(\delta + \bar{v}_h + \tau_{1h}(k))T_{hi}^{2ff}(k) = \bar{v}_h T_{hi}^f(k) + \tau_{1h}(k)T_{1c}^f(k) \tag{125}$$

We can now use T_{hi}^{2ff} in a discrete time version of (102) and obtain an estimate of V_{2c} in the same way as we calculated the estimate for V_{1c}. The procedure is repeated on Eqns.(103) and (104) to obtain an estimate of V_{3c}. For calculating the cold side liquid volume estimate corresponding to the whole heat exchanger we simply sum the estimate values in the heat exchanger cells, i.e.

$$\hat{V}_c(k) = \hat{V}_{1c}(k) + \hat{V}_{2c}(k) + \hat{V}_{3c}(k) \tag{126}$$

4.2.3 Simplified model

In this more realistic case it is assumed that only the hot and cold side input and output temperatures and the flow rates of the fluids in the heat exchanger are measurable. Besides, we assume that we know all the constant parameters of the heat exchanger except the one that is to be estimated (i.e. the heat transfer coefficient or the cold side volume).

The availability of so few temperature measurement data naturally brings the basic idea of parameter tracking, namely that the dynamics of the whole heat exchanger should be approximated with the dynamics of a single heat exchanger cell. This approach may seemingly lead to an oversimplification of the heat exchanger model, but - as we will see on the simulation results - it is completely sufficient for fault detection purposes. Let us now turn our attention to the identification algorithms in this case.

Estimating the heat transfer coefficient. For the purpose of tracking the heat transfer coefficient we start from the discrete time model of one heat exchanger cell using filtered temperature measurements that are described in Eqns.(24) and (25). First, we will give an estimate for τ_c and τ_h using the recursive least squares method and then unravel the estimated value of the heat transfer coefficient from $\hat{\tau}_c$ and $\hat{\tau}_h$ using the known relations between the parameters. For the sake of simpler notation the following variables are introduced

$$y_c(k+1) = \delta T_{co}^f(k) - \bar{v}_c(k)(T_{ci}^f(k) - T_{co}^f(k)) \tag{127}$$

$$y_h(k+1) = \delta T_{ho}^f(k) - \bar{v}_h(k)(T_{hi}^f(k) - T_{ho}^f(k)) \tag{128}$$

$$u_c(k) = T_{ho}^f(k) - T_{co}^f(k) \tag{129}$$

$$u_h(k) = T_{co}^f(k) - T_{ho}^f(k). \tag{130}$$

Then the discrete time model can be written in the following form

$$y_c(k+1) = \tau_c(k)u_c(k) \tag{131}$$

$$y_h(k+1) = \tau_h(k)u_h(k) \tag{132}$$

Using the model (131)-(132) we obtain the following estimates for τ_c and τ_h

$$\hat{\tau}_c(k+1) = \hat{\tau}_c(k) + p_c(k+1)u_c(k)(y_c(k+1) - \hat{\tau}_c(k)u_c(k)) \tag{133}$$

$$p_c(k+1) = \frac{p_c(k)}{\lambda_c + u_c^2(k)p_c(k)} \tag{134}$$

$$\hat{\tau}_h(k+1) = \hat{\tau}_h(k) + p_h(k+1)u_h(k)(y_h(k+1) - \hat{\tau}_h(k)u_h(k)) \tag{135}$$

$$p_h(k+1) = \frac{p_h(k)}{\lambda_h + u_h^2(k)p_h(k)} \tag{136}$$

We can compute the cold and hot side estimates of the heat transfer coefficients as follows

$$\hat{U}_c(k) = \frac{\hat{\tau}_c(k)c_{pc}\rho_c V_c}{A} \quad \text{and} \quad \hat{U}_h(k) = \frac{\hat{\tau}_h(k)c_{ph}\rho_h V_h}{A} \tag{137}$$

The actual estimates can again be computed using a convex combination

$$\hat{U}(k) = \alpha\hat{U}_c(k) + (1-\alpha)\hat{U}_h(k), \quad 0 \leq \alpha \leq 1 \tag{138}$$

where the parameters V_c, V_h and A denoting the cold and hot side fluid volume and the heat transfer area respectively correspond to the whole heat exchanger.

Note that in this case we can only estimate an average value for the heat transfer coefficient along the heat exchanger and we lose every spatial information about the faults, but jumps in the heat transfer coefficient, should they occur in any cell of the heat exchanger, can still be safely detected.

Estimating the cold side volume. Here we assume that the heat transfer coefficient is constant and known and the unknown parameter to be tracked in order to detect leaking is the cold side volume. For this purpose we use the cold side energy balance equation (24) which again can be written in the following input/output form

$$y(k+1) = V_{cr}(k)u(k). \tag{139}$$

where

$$y(k+1) = \delta T_{co}^f(k) \tag{140}$$

$$V_{cr}(k) = \frac{1}{V_c(k)} \tag{141}$$

$$u(k) = v_c(k)(T_{ci}^f(k) - T_{co}^f(k)) + \frac{UA}{c_{pc}\rho_c}(T_{ho}^f(k) - T_{co}^f(k)) \tag{142}$$

Considering the model (139) the recursive least squares method for estimating the cold side volume is as follows

$$\hat{V}_{cr}(k+1) = \hat{V}_{cr}(k) + p(k+1)u(k)(y(k+1) - \hat{V}_{cr}(k)u(k)) \tag{143}$$

$$\hat{V}_c(k+1) = \frac{1}{V_{cr}(k+1)} \tag{144}$$

$$p(k+1) = \frac{p(k)}{\lambda + u^2(k)p(k)} \tag{145}$$

When tracking the cold side fluid volume using the simplified model we can't expect very accurate estimates, since the single cell model this method uses gives only a rough description of the heat exchanger dynamics. The trends in the estimated cold side fluid volume are still informative and reliable as it will be visible on the simulation examples.

As we have seen, using the simplified model the estimation of neither the heat transfer coefficient nor the cold side fluid volume provides very exact results for us, issuing from the ignorance of some important dynamic properties of the real heat exchanger. Due to this fact the simultaneous application of the two methods above does not give us usable estimates, they can only be applied independently from each other and, of course, with different assumptions. *Therefore the simultaneous detection of the surface ageing and leaking is not possible in this case.*

4.3 Fault detection and diagnosis

The purpose of fault detection and diagnosis in our case is to detect changes in the estimates of the heat transfer coefficient and the cold side liquid volume respectively. These changes to be detected are of different nature: abrupt positive jumps in the heat transfer coefficient that indicate a $CaCO_3$ stones coming off the heat transfer surface and slow decrease in the cold side liquid volume that refers to leakage according to our assumptions. Still, the same method is applied successfully for both.

4.3.1 Detecting jumps in the heat transfer coefficient

Here we turn our attention to the jump detector. Under normal operating conditions the heat transfer coefficient will either be constant or slowly decreasing (see section 3.3.1 and Fig. 3). This means that after the estimates have reached a "steady state",

considering the 'all measurements available' case the prediction error estimate $\hat{\epsilon}_{jc}$ will fluctuate around zero (constant heat transfer coefficients) or be negative (decreasing heat transfer coefficients). The prediction error estimate $\hat{\epsilon}_{jh}$ will similarly either fluctuate around zero or be positive. However, after an abrupt positive jump in the heat transfer coefficient $\hat{\epsilon}_{jc}$ will become positive and $\hat{\epsilon}_{jh}$ will become negative. This makes the prediction errors or the scaled versions of them well suited signals to be monitored in order to detect a jump since *it leads to the standard situation of detecting a change in the mean value of the signal*, and in this application *we have prior knowledge of the sign of the change*. An examples for the scaled version of the prediction errors which can be used for jump detection is the difference between two consecutive estimates, namely

$$X(k) = \hat{U}_j(k+1) - \hat{U}_j(k) \tag{146}$$

in the 'all measurements available' and 'only cold side measurements available' cases and

$$X(k) = \hat{U}(k+1) - \hat{U}(k) \tag{147}$$

when using the simplified model.

Because of the above properties of $X(k)$ we propose to use a cumulative sum (CUSUM) detector for jump detection (see e.g. [2]). The detector is based on a cumulative sum test. Let

$$S_u(k) = \sum_{t=1}^{k} X(t), \qquad S_u(0) = 0 \tag{148}$$

and

$$m_u(k) = \min_{0 \le t \le k} S(t) \tag{149}$$

Since $X(t)$ is either fluctuating around zero or negative during normal operation, we expect $m(k)$ and $S(k)$ to be close. However, after a jump $S(k)$ will be considerably larger than $m(k)$. A jump is therefore detected if

$$S_u(k) - m_u(k) > \gamma \tag{150}$$

where γ is the threshold level. γ can be adjusted to the signals monitored, so the choice between the various scaled versions of the prediction error is of relatively minor importance. Again the value of γ is a tradeoff between quick detection and

sensitivity to noise and unmodelled dynamics. A small value of γ will give quick detection, but also more false alarms (detects jumps when no jump occured).

When a jump is detected the forgetting factor in the tracking algorithm is reduced to allow the estimate in question to reach its new value faster. The forgetting factor is gradually increased back to its normal value according to the formula

$$\lambda(t) = \lambda_0 \lambda(t-1) + \lambda_n(1-\lambda_0) \tag{151}$$

where λ_n is the normal value of the forgetting factor and λ_0 is the rate at which $\lambda(t)$ approaches λ_n. When a jump is detected the detection mechanism is switched off for a while in order to avoid multiple detection of the same jump. Also, should several jumps occur within a short time interval, it is only necessary to detect the first one since the reduced forgetting factor will ensure quick adaptation when the other jumps occur.

4.3.2 Detecting decrease in the cold side liquid volume

Our purpose in this case is to detect slow decrease in the mean of the estimates of the cold side volume. This problem is very similar to the previous case, with the difference that we do not have to detect abrupt changes in the monitored signal. Therefore, two possible solutions are proposed for setting an alarm of leaking.

The first solution is a simple limit checking that is safely applicable in the 'all measurements available' and 'only cold side measurements available' cases. The upper and lower bounds of the normal range of the cold side fluid volume are defined, and if the estimates drop below the lower bound for a certain time then the alarm of leaking is set.

The second solution is again the usage of the CUSUM method. This algorithm is even applicable when the 'simplified model' is used, since the trends in the estimates allow us to detect leaking in that case, too. The signal that is monitored is written as

$$Y(k) = \hat{V}_c(k+1) - \hat{V}_c(k) \tag{152}$$

The change detector is given by the following equations

$$S_{V_c}(k) = \sum_{t=1}^{k} Y(t), \qquad S_{V_c}(0) = 0 \tag{153}$$

$$m_{V_c}(k) = \min_{0 \le t \le k} S_{V_c}(t) \tag{154}$$

Leaking is detected if

$$S_{V_c}(k) - m_{V_c}(k) > \gamma_{V_c} \tag{155}$$

where γ_{V_c} is a certain threshold level.

In this case we do not have to adjust the forgetting factor in the identification algorithms, since according to the assumptions the change in the cold side fluid volume is slow. Due to this fact it is recommended to set the forgetting factors in the tracking algorithms close to 1 because the effects of noise and unmodelled dynamics can be reduced in this way.

4.3.3 Employing the change detectors simultaneously

When both time-varying parameters (i.e. the heat transfer coefficient and the cold side fluid volume) are estimated simultaneously the change detection of the signals and alarm setting have to be modified a bit. The parameter estimation algorithm in this case is described in Eqns.(69)-(82). As we can see in Eqns.(79) and (78) the previous estimate of the heat transfer coefficient is used when estimating the cold side volume. This solution gives perfect simultaneous tracking of the parameters provided that the heat transfer coefficient is constant or slowly decreasing. However, when an abrupt positive jump in the jth cell in the heat transfer coefficient occurs, the approximation of the heat transfer coefficient with its previous estimated value is no longer accurate enough causing a positive jump in the estimate of the cold side fluid volume. This behaviour makes it impossible to use the cold side volume estimates in the jth cell for fault detection until the heat transfer coefficient estimate reaches its true value.

In order to make it faster for the heat transfer coefficient estimate to reach the true value after a jump, the forgetting factors λ_{jUcc} and λ_{jUch} are reduced and then gradually increased in the same way as it is written in Eqn.(151). Moreover, the convergence of the heat transfer coefficient to the true value can be further accelerated by applying the following method. We store a certain number of the cold side fluid volume estimates in a FIFO queue which is updated at each sampling instant (i.e. the recent estimate is appended to the queue and the oldest estimate is deleted from it). When a jump in the heat transfer coefficient is detected the average value of the elements in the queue is computed and this value is substituted into Eqn.(71) instead of $\hat{V}_{jc}(k-1)$ for a given time interval. During this interval V_{jc}

is not used in Eqn.(126). If the jumps occur in all of the cells at the same time or very close in time to each other, then leakage detection is switched off until at least one HTC estimate reaches the true value. *This solution enables us to detect the two faults simultaneously.*

The detection of jumps in the heat transfer coefficient is the same as it was described in section 4.3.1, since the tracking of the heat transfer coefficient is not affected by a slow change in the cold side fluid volume.

4.4 Tuning knobs of the algorithms

The tuning knobs and the corresponding tradeoffs of the described fault detection and diagnosis methods are as follows.

Sampling interval. The selection of sampling interval has an important effect on the operation of the algorithms. A small sampling interval results in a bit quicker fault detection, but requires more numerical operations thus causes an increased load on the computer. Generally, the smallest allowable sampling rate that is computed on the basis of the rise times of the system (see [1]) is sufficient.

Filter coefficients. First and second order filter coefficients should be selected carefully not to loose information about the system's dynamic response. Filters with lower cutoff frequency remove noise more efficiently but cause increased detection delays. In practical cases it should be considered that thermometers have inertia, therefore they provide some kind of filtering before processing the temperature measurement data in any way.

Forgetting factors. All the developed recursive algorithms use exponential forgetting and their main tuning knob is the forgetting factor that was denoted by λ ($0 < \lambda < 1$). As it has been mentioned, a small value of λ enables the quick tracking of the estimated parameter but causes sensitivity to noise and unmodelled dynamics. According to this the easier problem is the estimation of the cold side volume since we assumed slow changes in that case. In the case of HTC estimation λ should be kept on its normal value during fault-free operation and decreased by 30-80% when a jump is detected to reach the true

value of the HTC quickly. When it happens, λ is gradually increased back to its normal value.

Threshold values. Fault detection itself – both in the case of leakage and the ageing of the heat transfer surface – is based on detecting changes in the estimated parameters by applying a CUSUM-test. Therefore, the setting of the threshold values in the CUSUM algorithm plays an essential role in early detection. This means that too large thresholds result in increased detection delays but cause low noise sensitivity, while too small threshold values lead to a greater number of false alarms.

5 Simulation results

The purpose of this section is to discuss the methods, tools and results of the simulations that were performed to test the developed algorithms. The section is organized as follows. First, the parameters and then the Simulink model of the heat exchanger used for producing temperature data are going to be presented. After that the implementation issues of the algorithms will be discussed which will be followed by the actual simulation results and the evaluation of the fault detection and diagnosis methods.

5.1 Description of the simulated heat exchangers

The implemented identification and fault detection algorithms were tested on two heat exchanger models with different dynamical properties, both consisting of three cells. The parameters of the two simulated operating units can be seen in Table 1 where the columns labeled 'HE 1' and 'HE 2' contain the parameter values corresponding to the first and second heat exchanger respectively. As we can see

Table 1: Parameter values of the simulated heat exchangers

Parameter	HE 1	HE 2	Unit of measure
v_c	$2.0 \cdot 10^{-3}$	$0.315 \cdot 10^{-3}$	$\frac{m^3}{s}$
v_h	$0.7 \cdot 10^{-3}$	$0.19 \cdot 10^{-3}$	$\frac{m^3}{s}$
V_c	1	$18.15 \cdot 10^{-3}$	m^3
V_h	0.5	$9.6 \cdot 10^{-3}$	m^3
c_{pc}	1910	1910	$\frac{J}{kg \cdot K}$
c_{ph}	1590	1590	$\frac{J}{kg \cdot K}$
ρ_c	1000	1000	$\frac{kg}{m^3}$
ρ_h	1000	1000	$\frac{kg}{m^3}$
A	4	1.9	m^2

on the table's data, the first simulated heat exchanger has much slower dynamics (due to its larger physical dimensions) than the second one. Therefore the first and

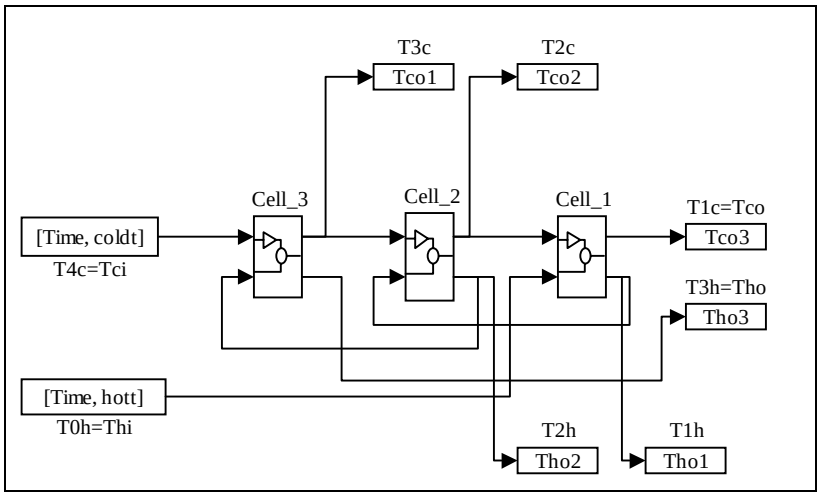

Figure 5: Simulink model of the simulated heat exchangers

second heat exchanger from now on will be referred as the 'slow' and the 'fast' one respectively.

5.2 The Simulink model of the heat exchanger

The model that simulates the operation of a 3-cell countercurrent heat exchanger was built in Matlab/Simulink. Matlab is an interactive computing environment that enables numerical computation and data visualization. Matlab has proven to be extraordinarily versatile and capable in its ability to help solve problems in applied mathematics, physics, chemistry, engineering, finance – almost any application area that deals with complex numerical calculations. Matlab has its own programming language that comprises hundreds of built-in functions to solve problems ranging from the very simple to the sophisticated and complex [14]. Simulink (which is a special optional part of Matlab) serves for the easy construction of different kinds of system models and for simulating their operation [15]. The passing of data to a model built in Simulink and the processing of the simulation results are carried out through the so-called workspace that is a large memory area for storing Matlab's variables.

The Simulink model of the countercurrent heat exchanger is able to simulate the deterioration of the heat transfer surface, the leakage of the outer container and to handle time-varying flow rates. The structure of the model is visible in Fig. 5. As we can see, the cells are numbered in such a way that the hot inlet temperature is near

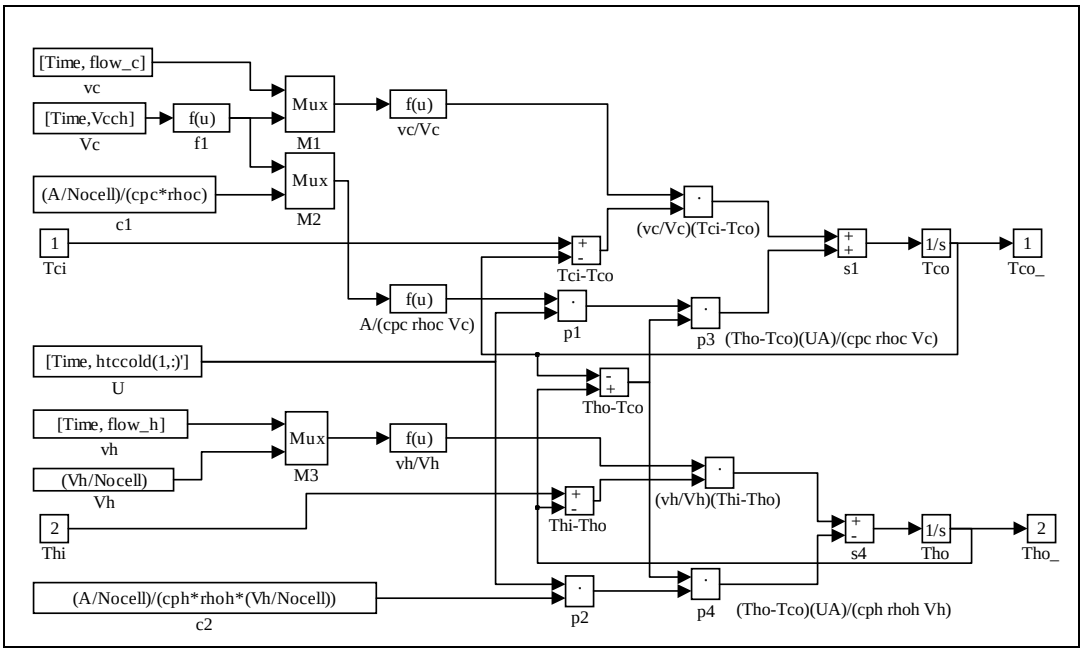

Figure 6: Simulink model of a single heat exchanger cell

cell no. 1 and the cold inlet temperature is next to cell no. 3. The inlet temperatures are taken from Matlab's workspace (see blocks 'T0h=Thi' and 'T4c=Tci') and the outlet temperatures of the cells are saved to the workspace into Matlab variables (see blocks 'T3c', 'T2c', 'T1c=Tco', 'T1h', 'T2h' and 'T3h=Tho'). The identificaiton and fault detection algorithms use these data as temperature measurements. The meaning of each block can be seen from the labels of the blocks in Fig. 5.

As we have already seen, the heat exchanger model consists of three cells. The Simulink model of one such heat exchanger cell is shown in Fig. 6. This model works on the basis of Eqns.(2), (6) and (8). The meanings of the system blocks can also be seen in the figure. The simulation of the two faults (i.e. the deterioration of the heat transfer surface and leakage) are handled by blocks 'U' and 'Vc' so that the previously generated values of U_j $(j = 1, 2, 3)$ and V_c are taken from Matlab's workspace by the Simulink model. The handling of time-varying hot and cold side flow-rates is carried out in the same way through blocks 'vc' and 'vh'.

5.3 Implementation of the parameter tracking and fault detection algorithms

The variants of the identification and fault detection algorithms were implemented in Matlab's programming language. The complete source code of the algorithms can be found in the Appendix (Section 7.1–7.6).

It is visible from the program codes that the algorithms need the complete temperature measurement records as inputs. This is because the data generation and the running of fault detection and diagnosis are performed separately in Simulink and the Matlab command prompt respectively. However, the algorithms (i.e. filtering, parameter estimation and fault detection) are written so that they need temperature measurement data from the recent and previous time step, which means that they are actually on-line algorithms. Storing the filtered temperature measurements, the estimated HTC or cold side liquid volume, the detected jumps and leakage in arrays is not absolutely necessary in a true implementation. These values are stored only for the purpose of plotting the results later.

A brief description and explanation of the six implemented methods is presented in the following where the names of Matlab variables are denoted by `typed` letters.

5.3.1 All measurements available

Estimating the heat transfer coefficient with jump detection. The commented source code of the method can be found in Appendix 7.1.

File name: AM_HTC.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco1, Tco2, Tco3`: cold side measured output temperatures of cells no. 3, 2 and 1 respectively

`Tho1, Tho2, Tho3`: hot side measured output temperatures of cells no. 1, 2 and 3 respectively

Outputs

`Uc_store, Uh_store`: HTC estimates computed on the basis of the cold and hot side energy balance equations respectively

`U_store`: convex combination of `Uc_store` and `Uh_store` obatined by applying Eq. (50)

`jump`: matrix for indicating jump detection in the HTC. If a jump is detected at sampling instant k in the jth cell of the heat exchanger then `jump(k,j)` is set to 1, otherwise it is 0.

Tuning knobs

`lambdac, lambdah`: forgetting factors of the gradient methods

`next_d`: variable showing when jump detection is to be started (no. of sample)

`delay`: variable that gives for how long the jump detection is switched off after detecting a jump in the HTC (no. of samples)

`gU`: threshold value for jump detection in the HTC

`lambdamin, lambdamax`: variables giving lower and upper bounds for the forgetting factors

`lambda0`: gives the rate at which `lambdac` and `lambdah` approach their normal value after jump detection

Operation

The procedure is the implementation of the algorithm described in Eqns. (42)-(49). Measurements are filtered through a first order IIR filter. Jump detection uses the signal `U_store` and it is based on Eqns. (146)-(151). To achieve safe numerical operations the gains (variables `pc` and `ph`) in the method are limited by variable `pmax`.

Estimation of the cold side liquid volume with leakage detection. The commented Matlab source code of the algorithm is in Appendix 7.2.

File name: AM_VC.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco1, Tco2, Tco3`: cold side measured output temperatures of cells no. 3, 2 and 1 respectively

`Tho1, Tho2, Tho3`: hot side measured output temperatures of cells no. 1, 2 and 3 respectively

Outputs

`Vc_store`: its colums contain the cold side liquid volume estimates corresponding to the heat exchanger's cells

`Vc_est`: cold side liquid volume estimates for the whole heat exchanger

`leak`: vector for indicating leakage: if leakage is detected at sampling instant k then `leak(k)` is set to -1, otherwise it is 0

Tuning knobs

`lambda_Vc`: forgetting factor of the gradient method

`next_d`: variable giving when leakage detection is to be started (no. of sample)

`delay`: variable that adjusts for how long leakage detection is switched off after detecting leakage (no. of samples)

`gVc`: threshold value for leakage detection

Operation

The procedure is the implementation of the algorithm described in Eqns. (56)-(60). Measurements are filtered through a first order IIR filter. Leakage detection uses the signal `Vc_est` and it is based on Eqns. (152)-(155). The gain of the gradient method (variable `p`) is limited by variable `pmax`.

Simultaneous estimation of the HTC and the cold side liquid volume with jump- and leakage detection. The commented Matlab source code of the algorithm can be found in Appendix 7.3.

File name: AM_SIM.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco1, Tco2, Tco3`: cold side measured output temperatures of cells no. 3, 2 and 1 respectively

`Tho1, Tho2, Tho3`: hot side measured output temperatures of cells no. 1, 2 and 3 respectively

Outputs

`Vc_store`: contains the cold side liquid volume estimates corresponding to the different cells of the heat exchanger

`Vc_estimate`: cold side liquid volume estimates corresponding to the whole heat exchanger

`Uc_store, Uh_store`: HTC estimates obtained using the cold and hot side energy balance equation respectively

`U_store`: convex combination of `Uc_store` and `Uh_store`

`leak`: vector for the indication of leakage detection: if leakage is detected at the kth sampling instant, then `leak(k)` is set to -1, otherwise it is 0

`jump`: matrix for indicating jump detection in the HTC: if a jump in the HTC is detected at sampling instant k in the jth cell of the heat exchanger, then `jump(k,j)` is set to 1, otherwise it is 0

Tuning knobs

`lambda_Ucc, lambda_Uch`: forgetting factors of the gradient methods that estimates the HTC

`lambdamin, lambdamax`: lower and upper bounds for `lambda_Ucc` and `lambda_Uch`

`lambda0`: the rate at which `lambda_Ucc` and `lambda_Uch` approach their normal value after jump detection

`next_d`: variable giving when jump detection in the HTC is to be started (no. of sample)

`delay`: variable that adjusts for how long jump detection is switched off after detecting a jump in the HTC (no. of samples)

`gU`: threshold value for jump detection

`lambda_Vc`: forgetting factor of the gradient method that estimates the cold side liquid volume

`next_Vc_d`: gives when leakage detection is to be started (no. of sample)

`delay_Vc`: adjusts for how long leakage detection is switched off after detecting leakage (no. of samples)

`gVc`: threshold value for leakage detection

`gsize`: size of the queue used to store the cold side volume estimates (no. of samples)

Operation

The algorithm used in this case is the implementation of the method described in Eqns. (69)-(82). Measurements are filtered through a first order IIR filter. Leakage detection uses the signal `Vc_estimate` and it is based on Eqns. (152)-(155). Jump detection in the HTC uses the signal `U_store` and it is based on Eqns. (146)-(151). To enable the simultaneous detection of the two faults, a given number of previous cold side volume estimates corresponding to the cells of the heat exchanger are stored in the matrix `V_queue`. `V_queue` is updated at each sampling instant. When a jump in the HTC in the jth cell is detected, the cold side volume estimate of the jth cell is substituted by the mean value of the elements standing in the jth column of `V_queue`. Moreover, the cold

side volume estimates of the jth cell are excluded (i.e. they are switched off) from the estimation of the cold side liquid volume corresponding to the whole heat exchanger for 1500 samples. If all the cold side volume estimates are switched off at the same time (when the jumps in the HTC are so close to each other in time) then leakage detection is stopped for this interval. The gains of the gradient methods estimating the HTC and the cold side liquid volume are limited by variables `P_Umax` and `P_Vcmax` respectively.

5.3.2 Only cold side measurements available

Estimation of the cold side liquid volume with leakage detection. The commented Matlab source code of the algorithm is in Appendix 7.4.

File name: CSO_VC.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco1, Tco2, Tco3`: cold side measured output temperatures of cells no. 3, 2 and 1 respectively

Outputs

`Vc_store`: its colums contain the cold side liquid volume estimates corresponding to the heat exchanger's cells

`Vc_estimate`: cold side liquid volume estimates for the whole heat exchanger

`leak`: vector for indicating leakage: if leakage is detected at sampling instant k then `leak(k)` is set to -1, otherwise it is 0

Tuning knobs

`lambda`: forgetting factor of the gradient method

`next_d`: variable giving when leakage detection is to be started (no. of sample)

`delay`: variable that adjusts for how long leakage detection is switched off after detecting leakage (no. of samples)

`gVc`: threshold value for leakage detection

Operation

The procedure is the implementation of the algorithm described in Eqns. (119)-(126). Measurements are filtered through a second order IIR filter. Leakage detection uses the signal `Vc_estimate` and it is based on Eqns. (152)-(155). The gain of the gradient method (variable `p`) is limited by variable `pmax`.

5.3.3 Simplified model

Estimation of the HTC with jump detection. The Matlab source code of the algorithm is presented in Appendix 7.5.

File name: SIMP_HTC.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco3, Tho3`: cold and hot side measured outlet temperatures

Outputs

`Uc_store, Uh_store`: HTC estimates obtained using the cold and hot side energy balance equations respectively

`U_store`: convex combination of `Uc_store` and `Uh_store`

`jump`: vector for indicating jump detection: if a jump in the HTC is detected at sampling instant k then `jump(k)` is set to 1, otherwise it is 0

Tuning knobs

`lambdac, lambdah`: forgetting factors of the recursive least suares methods

`next_d`: variable giving when jump detection is to be started (no. of sample)

`delay`: variable that sets for how long the jump detection is switched off after detecting a jump in the HTC (no. of samples)

`g` : threshold value for jump detection in the HTC

`lambdamin, lambdamax`: variables giving lower and upper bounds for the forgetting factors

`lambda0`: gives the rate at which `lambdac` and `lambdah` approach their normal value after jump detection

Operation

The procedure is the implementation of the algorithm described in Eqns. (127)-(138). Measurements are filtered through a first order IIR filter. Jump detection uses the signal `U_store` and it is based on Eqns. (152)-(155). The gains of the recursive least squares methods (variables `pc` and `ph`) are limited by variable `pmax`.

Estimation of the cold side liquid volume with leakage detection. The Matlab source code of the algorithm is presented in Appendix 7.6.

File name: SIMP_VC.M

Inputs

`coldt, hott`: cold and hot measured inlet temperatures

`Tco3, Tho3`: cold and hot side measured outlet temperatures

Outputs

`Vc_store`: contains the cold side volume estimates corresponding to the whole heat exchanger

`leak`: vector for indicating leakage detection: if leakage is detected in the kth sampling instant then `leak(k)` is set to -1, otherwise it is 0

Tuning knobs

`lambda_Vc`: forgetting factor of the recursive least suares methods

`next_d`: variable giving when leakage detection is to be started (no. of sample)

`delay`: variable that sets for how long the leakage detection is switched off after detecting leakage in the heat exchanger (no. of samples)

`gVc`: threshold value for leakage detection

Operation

The procedure is the implementation of the algorithm described in Eqns. (139)-(145). Measurements are filtered through a first order IIR filter. Leakage detection uses the signal `Vc_store` and it is based on Eqns. (152)-(155). The gain of the recursive least squares method (variables `p`) is limited by variable `pmax`.

5.4 Presentation of the results

Due to the space limitations of this diploma work, only the figures regarding the slow heat exchanger will be shown in this section. However, the results of the evaluation of the algorithms will be presented for both models.

The simulated slow heat exchanger was sampled every 30 seconds on the basis of its dominant time constant (165 seconds). The simulated operating time was 108 hours which means 12960 samples. Gaussian type measurement noise was added to the simulated temperature data at each sampling instant with variance 0.01. The hot and cold side inlet temperatures were randomly generated: the simulation interval

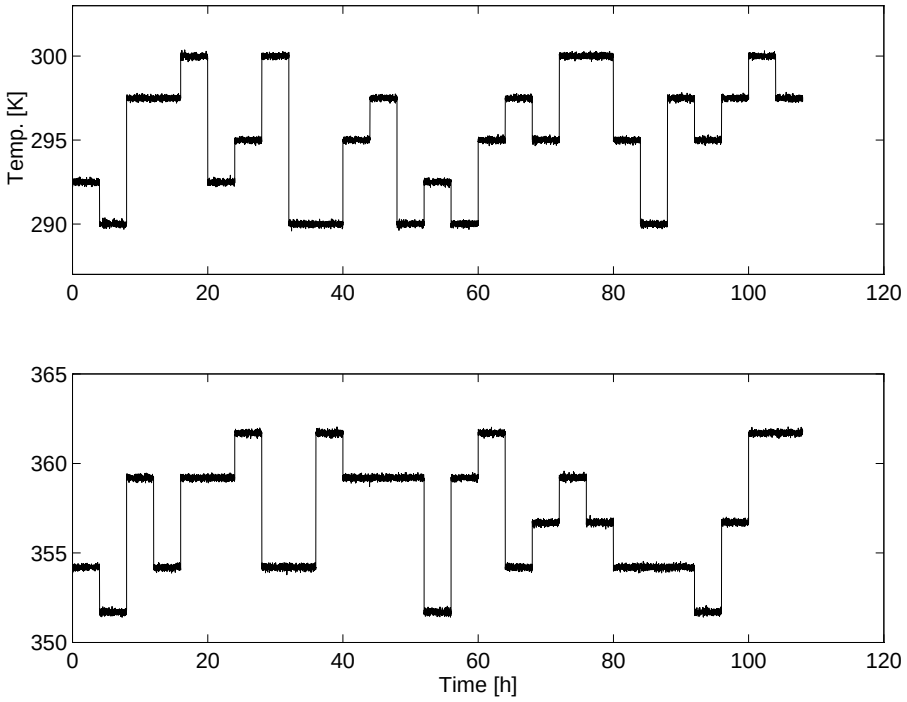

Figure 7: Cold and hot measured inlet temperatures

was divided into 27 subintervals of equal length. Within each subinterval the hot and cold inlet temperatures were constant and took their predefined mean values (353K and 293K respectively), ±2.5K, or ±5K with equal probability in order to provide good excitation for the system dynamics. Typical cold and hot inlet temperatures (T_{co} and T_{ho}) with measurement noise are visible in Fig. 7. The measured response of the slow heat exchanger to these inputs is shown in Fig. 8. The hot and cold side flow rates were constant during the simulations.

5.4.1 All measurements available

First order filtering was used for filtering the measurement data in the three algorithms belonging to this case. The filter coefficient e_0 was 0.0069.

Estimating the heat transfer coefficient with jump detection. The HTC profiles were different in each cell while the cold side volume was kept constant on its nominal value. The threshold value γ (150) for detecting jumps in the HTC was 5 $\frac{J \cdot m^2}{K \cdot s}$. After a jump was detected, the forgetting factors λ_c and λ_h were reduced to 0.2 and restored to 0.99 according to (151) with λ_0=0.995. The detector giving an

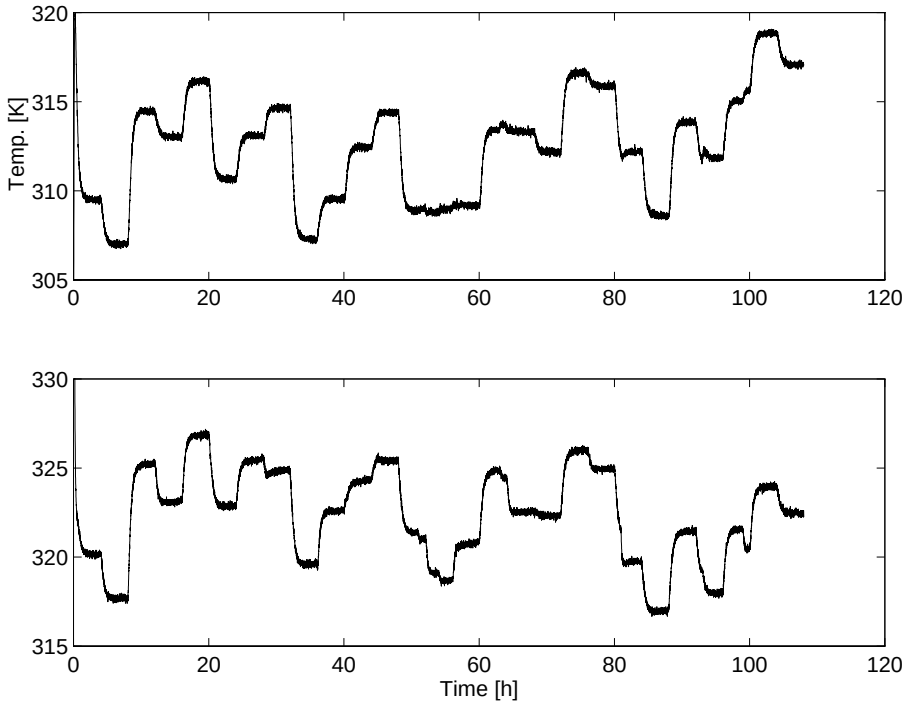

Figure 8: Cold and hot measured outlet temperatures

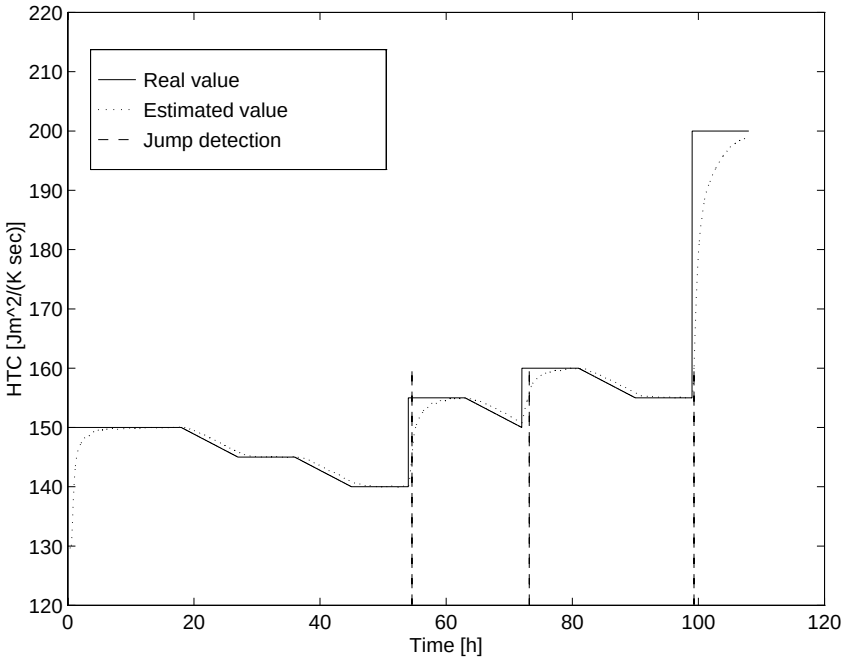

Figure 9: Real and estimated value of the HTC in the 2nd cell with jump detection – all measurements available

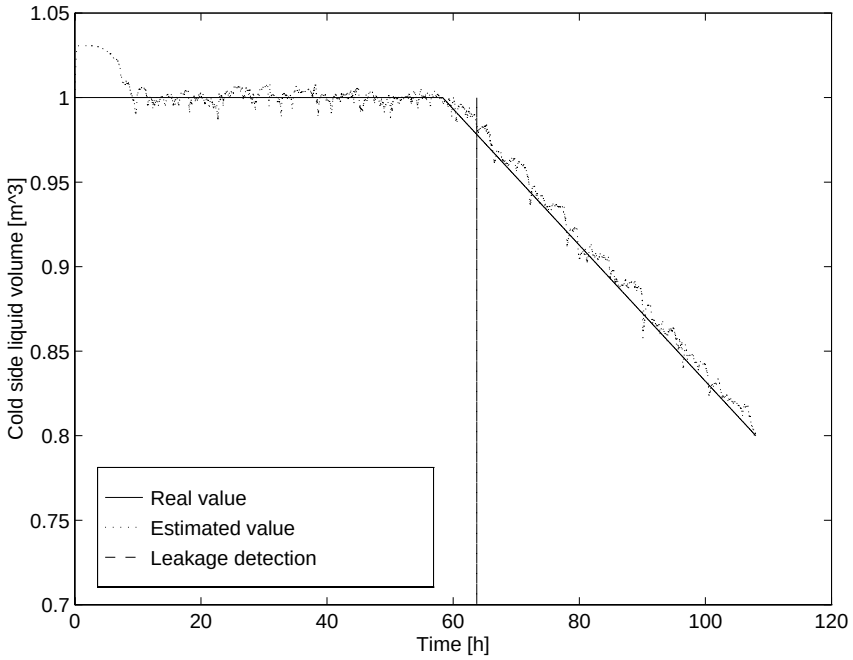

Figure 10: Real and estimated value of the cold side liquid volume with leakage detection – all measurements available

alarm was switched off for 500 samples after jump detection. Jump detection was not used for the first 1000 samples allowing the estimates to reach a "steady state". A typical result corresponding to the second (middle) cell of the heat exchanger is shown in Fig. 9. As it can be seen the results are quite good. All the jumps are detected and no false alarms occured. As it was expected the best tracking was achieved in this case.

Estimating the cold side liquid volume with leakage detection. Leakage was simulated with a linear decrease in the cold side liquid volume and the HTCs were constant and took their nominal values in each cell. The threshold value (see Eq.(151)) for leakage detection was 0.025 m^3 (which is 2.5% of the nominal value). After leakage was detected, leakage detection was switched off for the rest of the simulation time. The value of the forgetting factor λ was 0.99. Leakage detection was started after 500 samples. The result in this case is visible in Fig. 10. The estimates follow the true value well and the fault detection is safe, although it has noticeably big delay but it is due to the very slow decrease of the liquid volume.

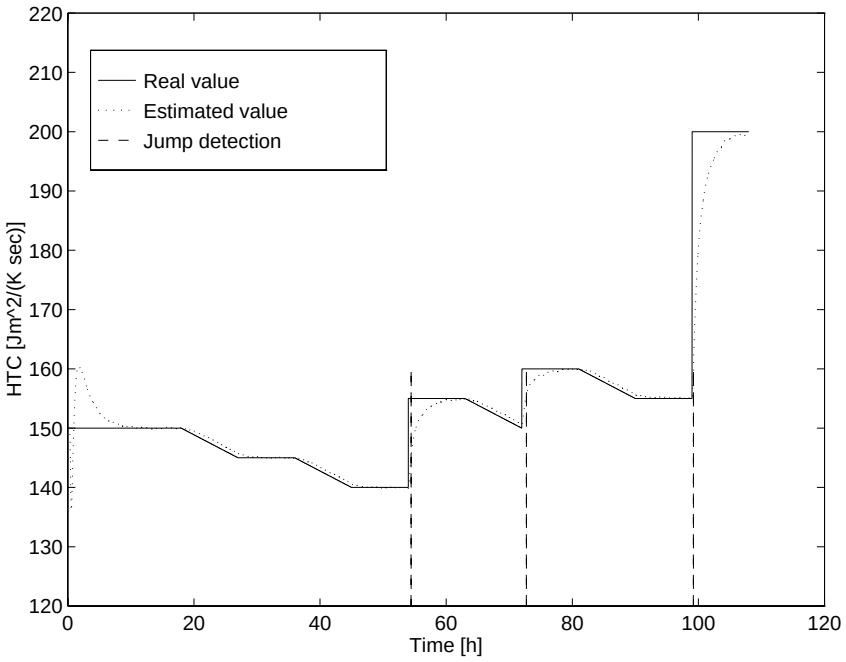

Figure 11: Real and estimated value of the HTC in the 2nd cell with jump detection – all measurements available, simultaneous estimation and fault detection

Simultaneous estimation of the HTC and the cold side liquid volume with jump- and leakage detection. In this case different heat transfer coefficients were generated for the hree heat exchanger cells and leakage was also simulated with a linear decrease of the cold side liquid volume. The threshold value for leakage detection was again 0.025 m^3 and for jump detection in the HTC it was 10 $\frac{J \cdot m^2}{K \cdot s}$. The estimation of the cold side volume was started after 1000 samples, the estimates were kept on their nominal value until that time and thus the HTC estimates could reach their true value more quickly. Jump- and leakage detection was started after 500 and 2000 samples respectively. The length of the queue for storing V_{jc} estimates was 1000 samples. As soon as a jump was detected in the HTC in the jth cell, the corresponding forgetting factors in the HTC estimator algorithms were set to 0.2 and then restored gradually to 0.99. The value of the forgetting factor used in the cold side volume estimation algorithm was 0.99. Jump detection was switched off for 500 samples after detecting a jump while leakage detection was switched off completely after setting a leakage alarm. The results corresponding to the estimation of the HTC and the cold side volume are visible in Figs. 9 and 10 respectively.

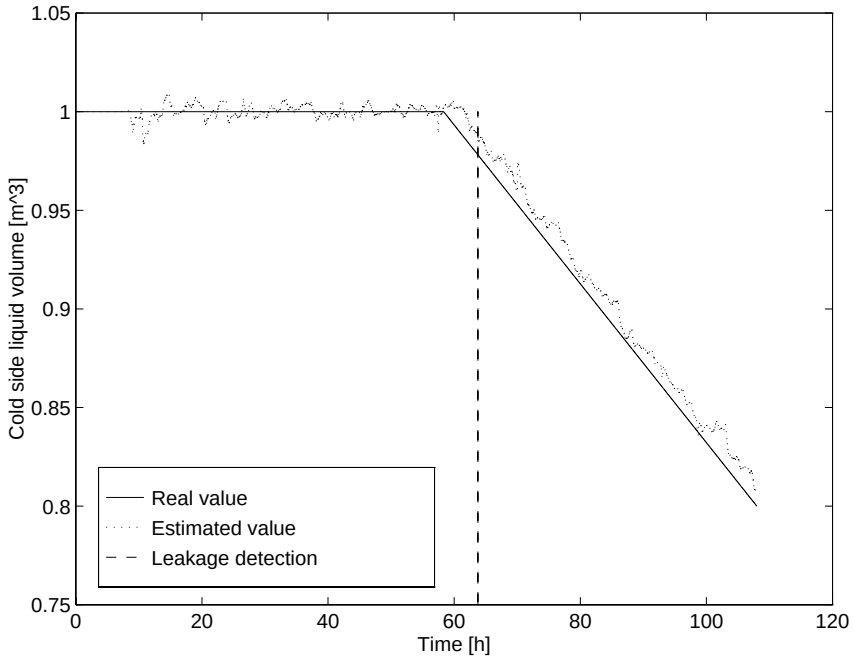

Figure 12: Real and estimated value of the cold side liquid volume with leakage detection – all measurements available, simultaneous estimation and fault detection

5.4.2 Only cold side measurements available

In this case the temperature measurements were filtered through a second order filter with coefficients $e_0 = 4.27 \cdot 10^{-5}$ and $e_1 = 0.0134$.

Estimation of the cold side liquid volume with leakage detection. The heat transfer coefficients were constant (nominal values) in this case in each cell, and leakage was simulated by a linear decrease in the cold side liquid volume. Leakage detection was started after the first 1000 samples, its threshold value was $0.025m^3$. After a leakage alarm was set, leakage detection was switched off completely. Fig. 13 shows the true and estimated parameter values in this case. As it is visible, the results are as accurate as in the 'all measurements available' case, but on the price of about twice as many numerical operations as we will see in Section 5.5.

5.4.3 Simplified model

In the forthcoming two algorithms first order filtering was applied on the measurement data with coefficient $e_0 = 0.0069$. A three-cell heat exchanger model was used

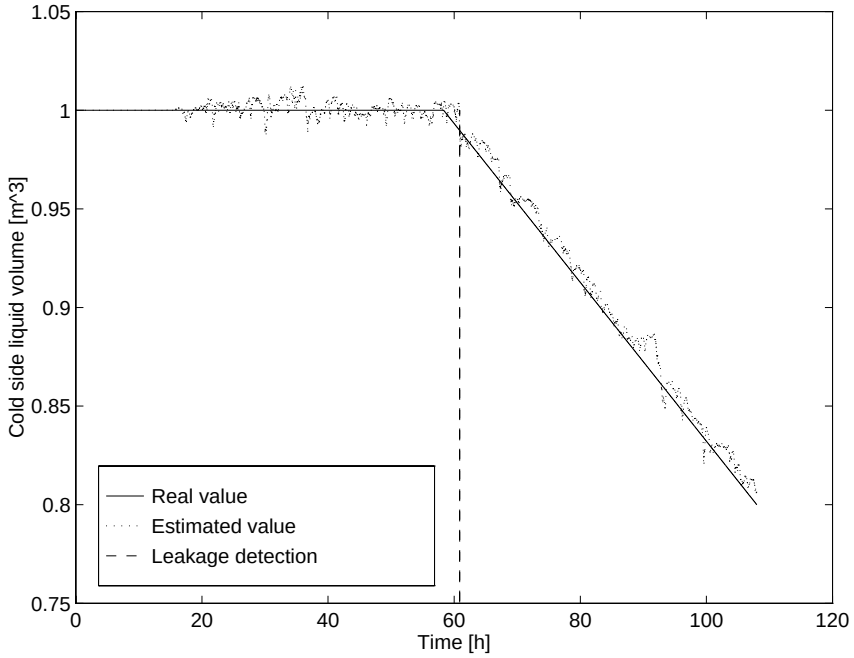

Figure 13: Real and estimated value of the cold side liquid volume with leakage detection – only cold side measurements available

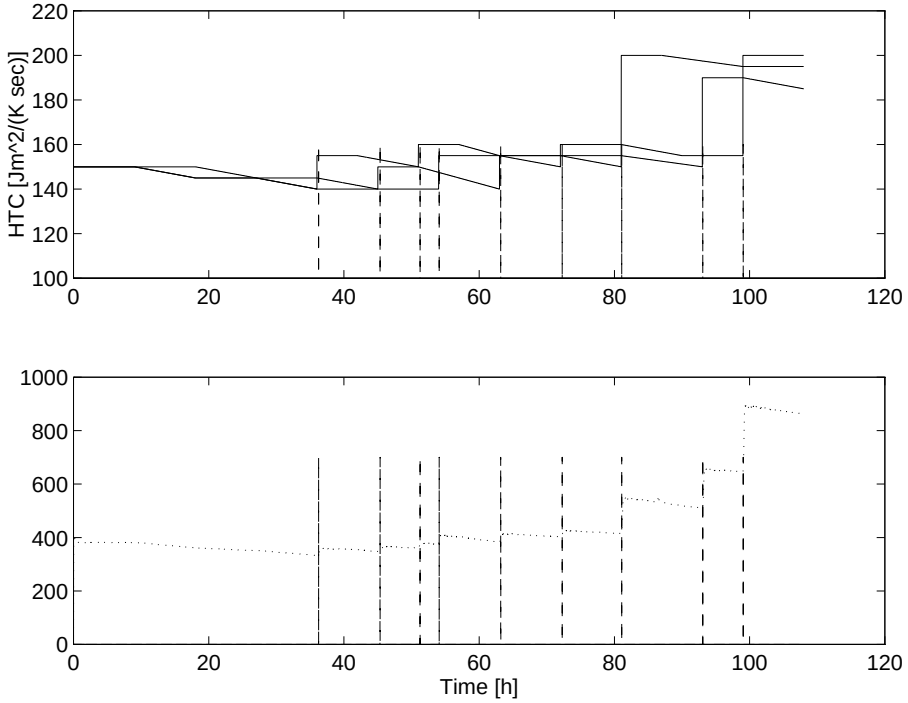

Figure 14: Real values of the HTCs in the three HE cells and the estimated average value with jump detection – simplified model

for generating the data but the algorithms used the simplified model containing only one cell (see Section 4.2.3). It means that temperature measurement data were needed only from the hot and cold input and output of the heat exchanger. The parameters of the one-cell model used in the identification algorithm were the same as the parameters of one simulated heat exchanger cell except the heat transfer area and the hot and cold side volumes which were multiplied obviously by the number of the simulated cells (i.e. by 3).

Estimating the HTC with jump detection. The threshold for jump detection was 10 $\frac{J \cdot m^2}{K \cdot s}$. Jump detection was turned on after the first 1500 samples. After a jump was detected the forgetting factors were set to 0.8 and then increased back to the normal value (0.88). The HTC profiles were different in the heat exchanger cells and altogether 9 jumps occured in the heat exchanger during the simulation. The real HTC profiles and the estimated average HTC value are shown in Fig. 14. As it is visible, jumps in the HTC occuring in any cell can be detected. From this it can be concluded that *the simplified dynamic model is sufficient for fault detection purposes*, though in this case the place where the fault occured cannot be localized physically. The probability of missed alarms is quite high in this case, because the jump detection is switched off for 200 samples after a jump is detected and this might prevent us from detecting consecutive jumps in different cells.

Estimating the cold side liquid volume with leakage detection. The threshold value for leakage detection was 0.025 m^3 in this case. The value of the forgetting factor was 0.99. After detecting leakage, further leakage detection was stopped for the rest of the simulation time. The real and estimated cold side liquid volumes are shown in Fig. 15. The difference between these values is significant in this case, but the trends in the estimated signal still contain enough information for safe fault detection.

5.5 Evaluation of the algorithms

The evaluation of the implemented algorithms was completed according to the definitions given in Section 2.7. During the simulations 3 jumps in the HTC in each cell were simulated. The same HTC profile was used for both heat exchanger models.

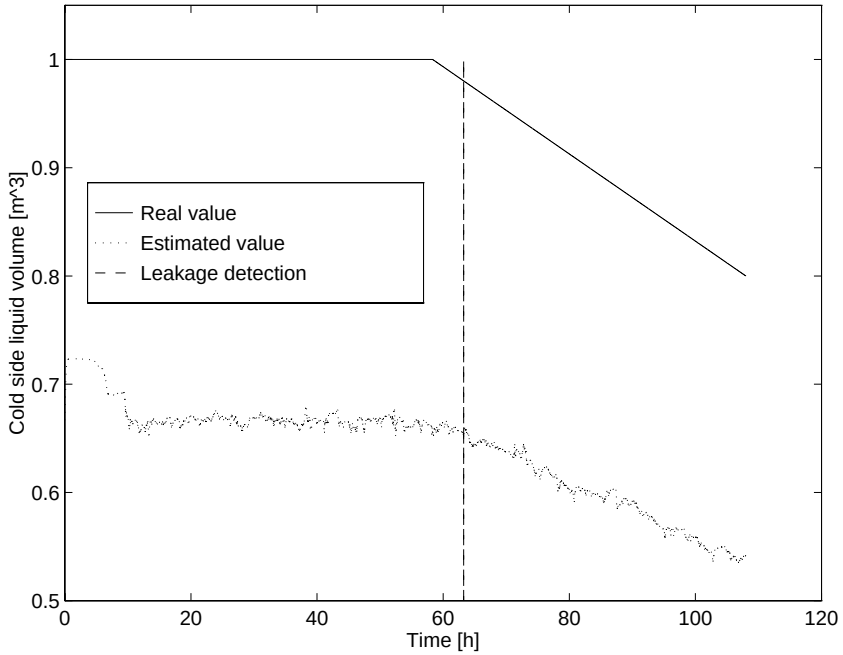

Figure 15: Real and estimated value of the cold side liquid volume with leakage detection – simplified model

Leakage began at sample no. 7000 and the cold side volume decreased linearly until 80% of the nominal value till the end of the simulation (see the true value of the cold side volume e.g. in Fig. 10). The tuning knobs of the algorithms were set equally proportionally to the parameters of the two models to produce comparable results. Tables 2-5 contain the results of the evaluation. For the sake of briefness the names of the Matlab files implementing a method were written in the first columns of the tables as identifiers of the methods.

Table 2 contains the results corresponding to the slow heat exchanger and HTC estimation with jump detection. As it can be seen from the data, no algorithm produced any false or missed alarms. The algorithm using the simplified model (i.e. `SIMP_HTC.M`) produced the best detection delays while there is no significant difference between the detection delays of `AM_HTC.M` and `AM_SIM.M`. Moreover, `SIMP_HTC.M` has the smallest computational complexity and the simultaneous fault detection method (`AM_SIM.M`) requires by far the greatest number of floating point operations.

Table 3 shows the results for the slow heat exchanger and the cold side volume estimation with leakage detection. There were again no false or missed alarms,

Table 2: Evaluation results of the fault detection and diagnosis methods – slow heat exchanger, HTC estimation and jump detection

	FD false alarms	FD missed alarms	DDs	Mean of DD	Mflops
AM_HTC.M	0	0	[50,41,24,50,87, 27,59,102,32]	52.444	4.258
AM_SIM.M	0	0	[70,45,20,53,80, 25,60,98,24]	52.778	25.581
SIMP_HTC.M	0	0	[31,41,32,14,24, 36,10,9,7]	22.667	1.259

Table 3: Evaluation results of the fault detection and diagnosis methods – slow heat exchanger, cold side liquid volume estimation and leakage detection

	FD false alarms	FD missed alarms	DDs	Mean of DD	Mflops
AM_VC.M	0	0	650	650	2.66
AM_SIM.M	0	0	653	653	25.581
CSO_VC.M	0	0	307	307	4.235
SIMP_VC.M	0	0	561	561	0.738

and the best detection delays were given by the algorithm using only cold side measurements (CSO_VC.M). The algorithm using the simplified model (SIMP_VC) is the most "economical" regarding the number of floating point operations with relatively good results. From this it can be concluded that if the assumptions make it possible, it is always a good choice to use the simplified model for leakage detection.

Tables 4 and 5 show the evaluation results obtained from the simulations of the fast heat exchanger. We can see that the results do not show significant difference compared to the evaluation data of the slow heat exchanger. The data in these tables strengthen the statements above in every respect.

Table 4: Evaluation results of the fault detection and diagnosis methods – fast heat exchanger, HTC estimation and jump detection

	FD false alarms	FD missed alarms	DDs	Mean of DD	Mflops
AM_HTC.M	0	0	[46,50,27,53,87, 37,61,101,28]	54.444	4.258
AM_SIM.M	0	0	[61,49,17,52,81, 16,57,96,15]	49.333	25.581
SIMP_HTC.M	0	0	[27,47,39,21,21, 31,16,17,11]	25.444	1.259

Table 5: Evaluation results of the fault detection and diagnosis methods – fast heat exchanger, cold side liquid volume estimation and leakage detection

	FD false alarms	FD missed alarms	DDs	Mean of DD	Mflops
AM_VC	0	0	621	621	2.66
AM_SIM	0	0	647	647	25.581
CSO_VC	0	0	357	357	4.235
SIMP_VC	0	0	512	512	0.738

6 Conclusions and possible future work

Model-based methods for fault diagnosis of heat exchangers have been proposed in this diploma work. The methods are based on a first principle model of the countercurrent heat exchanger, and on the white and grey box models of the faults: the leakage of the outer container and the deterioration of the heat transfer surface by ageing. We could see in the case of fault modelling, that prior knowledge of the investigated process plays an essential role in solving this engineering problem. Both faults are detected by estimating the parameters that carry the diagnostic information using recursive algorithms, and by detecting characteristic changes in the estimates applying the CUSUM method.

Six variants of the fault detection and diagnosis methods have been developed and implemented according to the availability of temperature measurement data. All the proposed fault detection methods showed good performance in the simulation experiments. The investigated cases were the following.

In the less realistic 'all measurements available' case we assumed that the hot and cold side temperatures were measurable along the heat exchanger. Three algorithms were created for this case: two for the separate detection of the two faults and one for the simultaneous estimation of them. When using the simultaneous fault detection method we could safely detect and isolate the two faults even if they occurred at the same time.

The 'only cold side measurements available' case is based on the assumption that we can measure the cold side temperatures along the heat exchanger in addition to the hot inlet temperature. Leakage detection is performed by approximating the heat exchanger cells with a LTI system between each sampling instant and thus eliminating the need of the measurement of the hot side temperatures. The algorithm developed for this case provides accurate parameter estimation and safe fault detection.

The most realistic case is when we only have access to the cold and hot side input and output temperatures. The two parameter estimation methods designed for these assumtions use a 'simplified model' of a countercurrent heat exchanger. This means that we approximate the dynamics of the operating unit with only one triplet of ordinary differential equations (i.e. one cell). The simulation results show

that this approach is very successful since it provides quick and safe fault detection. Moreover, it has small computational complexity, and this case stands closest to the availability of measurements in reality. These results do not necessarily show that it is always enough to only measure the inlet and outlet temperatures, but we can choose the size of the cells between certain (quite wide) boundaries, and a rough approximation of the dynamics is sufficient for fault detection purposes.

All algorithms have been created for the countercurrent heat exchanger, but they can be easily reformulated to detect faults in co-current heat exchangers, since we have very similar cascade model for this operating unit. It is a very important property of the fault detection algorithms that they do not require the linearity of the system model except leakage detection in the 'only cold side measurements' case. Therefore, it is quite easy to rewrite and test them on certain nonlinear cascade models of the heat exchanger. Having seen the good performance of the method for simultaneous fault detection in the 'all measurements available' case, it can also be an interesting aim of future work to try to simultaneously estimate the two parameters in the 'only cold side measurements' case. Investigating the optimal settings of the tuning knobs by defining different objective functions can also be of interest in further developments.

7 Appendix – Source code of the algorithms

7.1 Estimation of the heat transfer coefficient and jump detection - all measurements available

```
%File:  AM_HTC.M
%Description:  Algorithm for estimating the heat transfer coefficient
% and detecting jumps in it in a 3-cell countercurrent heat exchanger
%Measurements:  all measurements available
%Inputs:  hott,coldt,Tco1,Tco2,Tco3,Tho1,Tho2,Tho3
%Parameters needed:  Nocell,A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Uc_store, Uh_store, U_store, jump
%Author:  Gábor Szederkényi
%Last modified:  05 01 1998
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
e0=(vc)/(Vc/Nocell)+(U*(A/Nocell))/(cpc*rhoc*(Vc/Nocell));
alpha=0.5; %constant for convex combination of the estimates
lambdamin=0.8; %minimal value of lambdac and lambdah
lambda0=0.995; %growth rate of lambda after a jump was detected
lambdamax=0.99; %maximal value of lambdac and lambdah
d_length=length(Tco1); %number of measured data points
%-------------------------------------------------------------------------
%Temperature measurements:
Tci=[Tco2,Tco1,coldt]; %noisy cold inputs of cells 1,2,3 respectively
Tco=[Tco3,Tco2,Tco1]; %noisy cold outputs of cells 1,2,3
Thi=[hott,Tho1,Tho2]; %noisy hot inputs of cells 1,2,3
Tho=[Tho1,Tho2,Tho3]; %noisy hot outputs of cells 1,2,3
%-------------------------------------------------------------------------
%Allocating memory for the filtered temperature data
Tci_f=300*ones(d_length,Nocell);
Tco_f=300*ones(d_length,Nocell);
Thi_f=300*ones(d_length,Nocell);
Tho_f=300*ones(d_length,Nocell);
dTco_f=zeros(d_length,Nocell);
dTho_f=zeros(d_length,Nocell);
%-------------------------------------------------------------------------
%initialization of variables and parameters:  cold side equation
ec=zeros(1,Nocell); %prediction error estimate - cold side
psic=zeros(1,Nocell); %ec's negative gradient w.r.t.  U
Uc_store=zeros(d_length,Nocell); %storage for the HTC estimates - cold s.
lambdac=0.8*ones(1,Nocell); %lambda for the cold side estimates
pc=10*ones(1,Nocell); %gain in the identification scheme - cold side
pcmax=1e6; %maximum gain value
Uc_est=130*ones(1,Nocell); %recent estimate of U from the cold side
```

```
%------------------------------------------------------------------------
%initialization of variables and parameters:  hot side equation
eh=zeros(1,Nocell); %prediction error estimate - hot side
psih=zeros(1,Nocell); %eh's negative gradient w.r.t.  U
Uh_store=zeros(d_length,Nocell); %storage for the HTC estimates - hot s.
lambdah=0.8*ones(1,Nocell); %lambda for the hot side
ph=10*ones(1,Nocell); %gain in the identification scheme - hot side
phmax=1e6; %maximum gain value
Uh_est=130*ones(1,Nocell); %recent estimate of U from the hot side
U_store=zeros(d_length,Nocell); %storage for the HTC estimates
%------------------------------------------------------------------------
%Variables used in the jump detection algorithm
next_d=1000*ones(1,Nocell); %jump detection starts after 1000 samples
SU=zeros(1,Nocell); %CUSUM value
mU=zeros(1,Nocell); %min.value
delay=500; %jump detection is switched off for 500 samples after jump det.
gU=5; %threshold for jump detection
jump=zeros(d_length,1);
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%Beginning of the algorithm
for k=1:d_length-1
   for j=1:Nocell
      %filtering the temperature measurements
      Tci_f(k+1,j)=Tci_f(k,j)-sampint*e0*Tci_f(k,j)+sampint*e0*Tci(k,j);
      Tco_f(k+1,j)=Tco_f(k,j)-sampint*e0*Tco_f(k,j)+sampint*e0*Tco(k,j);
      Thi_f(k+1,j)=Thi_f(k,j)-sampint*e0*Thi_f(k,j)+sampint*e0*Thi(k,j);
      Tho_f(k+1,j)=Tho_f(k,j)-sampint*e0*Tho_f(k,j)+sampint*e0*Tho(k,j);
      dTco_f(k,j)=(Tco_f(k+1,j)-Tco_f(k,j))/sampint;
      dTho_f(k,j)=(Tho_f(k+1,j)-Tho_f(k,j))/sampint;
      if k>100
         lambdamin=0.99; %increase lambdamin after the first 100 samples
      end
%------------------------------------------------------------------------
      %Estimation from the cold side:
      ec(j)=dTco_f(k,j)-(vc/(Vc/Nocell))*(Tci_f(k,j)-Tco_f(k,j))-...
      ((Uc_est(j)*(A/Nocell))/(cpc*rhoc*(Vc/Nocell)))*(Tho_f(k,j)-Tco_f(k,j));
      psic(j)=((A/Nocell)/(cpc*rhoc*(Vc/Nocell)))*(Tho_f(k,j)-Tco_f(k,j));
      %Gradient method:
      pc(j)=pc(j)/(lambdac(j)+psic(j)^2*pc(j));
      pc(j)=min(pc(j),pcmax); %gain limiting
      Uc_est(j)=Uc_est(j)+pc(j)*psic(j)*ec(j);
      Uc_store(k,j)=Uc_est(j);
      %Setting the value of lambda
      lambdac(j)=min([lambdamin,lambdamax,lambda0*lambdac(j)+...
      lambdamax*(1-lambda0)]);
```

```
%-------------------------------------------------------------------------
     %Estimation from the hot side:
     eh(j)=dTho_f(k,j)-(vh/(Vh/Nocell))*(Thi_f(k,j)-Tho_f(k,j))-...
     ((Uh_est(j)*(A/Nocell))/(cph*rhoh*(Vh/Nocell)))*(Tco_f(k,j)-Tho_f(k,j));
     psih(j)=((A/Nocell)/(cph*rhoh*(Vh/Nocell)))*(Tco_f(k,j)-Tho_f(k,j));
     %Gradient method:
     ph(j)=ph(j)/(lambdah(j)+psih(j)^2*ph(j));
     ph(j)=min(ph(j),phmax); %gain limiting
     Uh_est(j)=Uh_est(j)+ph(j)*psih(j)*eh(j);
     Uh_store(k,j)=Uh_est(j);
%-------------------------------------------------------------------------
     %Setting the value of lambda (if needed):
     lambdah(j)=min([lambdamin,lambdamax,lambda0*lambdah(j)+...
     lambdamax*(1-lambda0)]);
     %convex combination of the cold and hot side estimates:
     U_store(k,j)=alpha*Uc_store(k,j)+(1-alpha)*Uh_store(k,j);
%-------------------------------------------------------------------------
     %Jump detection algorithm
     if(k>1 & k>next_d(j))
        SU(j)=SU(j)+U_store(k,j)-U_store(k-1,j);
        mU(j)=min(mU(j),SU(j));
        if (SU(j)-mU(j))>gU;
           lambdac(j)=0.2;
           lambdah(j)=0.2;
           jump(k,j)=1;
           SU(j)=0;
           mU(j)=0;
           next_d(j)=k+delay;
        end
     end %if
  end %for j=...
end %for k=...
%Setting the values of the last estimates (for plotting):
U_store(d_length,:)=U_store(d_length-1,:);
```

7.2 Estimation of the cold side liquid volume and leakage detection – all measurements available

```
%File:  AM_VC.M
%Description:  Algorithm for estimating the cold side liquid volume
% and detecting leakage in a 3-cell countercurrent heat exchanger
%Measurements:  all measurements available
%Inputs:  hott,coldt,Tco1,Tco2,Tco3,Tho1,Tho2,Tho3
%Parameters needed:  Nocell,A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Vc_store, Vc_est, leak
```

```
%Author:  Gábor Szederkényi
%Last modified:  05 01 1998
%-----------------------------------------------------------------
%-----------------------------------------------------------------
e0=(vc)/(Vc/Nocell)+(U*(A/Nocell))/(cpc*rhoc*(Vc/Nocell)); %filter coeff.
d_length=length(Tco1); %size of measurement records
%-----------------------------------------------------------------
%Temperature measurements
Tci=[Tco2,Tco1,coldt]; %noisy cold inputs of cells 1,2,3 respectively
Tco=[Tco3,Tco2,Tco1]; %noisy cold outputs of cells 1,2,3
Thi=[hott,Tho1,Tho2]; %noisy hot inputs of cells 1,2,3
Tho=[Tho1,Tho2,Tho3]; %noisy hot outputs of cells 1,2,3
%-----------------------------------------------------------------
%Allocating memory for the filtered temperatures
Tci_f=300*ones(d_length,Nocell);
Tco_f=300*ones(d_length,Nocell);
Thi_f=300*ones(d_length,Nocell);
Tho_f=300*ones(d_length,Nocell);
dTco_f=zeros(d_length,Nocell);
dTho_f=zeros(d_length,Nocell);
%-----------------------------------------------------------------
%Initializing variables used in the identification algorithm:
eps=zeros(1,Nocell); %prediction error estimate
psi=zeros(1,Nocell); %eps's negative gradient w.r.t.Vcrec
Vc_store=zeros(d_length,Nocell); %array for storing Vc estimates
lambda_Vc=0.99*ones(1,Nocell); %forgetting factor
p=10*ones(1,Nocell); %initial value of the gain
pmax=1e6; %maximum gain value
Vcr_est=Vc*Nocell*ones(1,Nocell); %initial values for recent 1/Vc and
Vc_est=zeros(d_length,1); %Vc estimates
%-----------------------------------------------------------------
%Variables used in the leakage detection algorithm
delay=10000; %no.of samples for which the leakage is switched off
%after an alarm was set
next_d=500; %beginning of leakage detection (no.of sample)
SVc=0; %CUSUM value for leakage detection
mVc=0; %minimum value for leakage detection
gVc=Vc*0.025; %threshold value:  5 percent decrease
leak=zeros(d_length,1); %ouput vector that contains alarm(s)
%-----------------------------------------------------------------
%-----------------------------------------------------------------
%Beginning of the main part of the algorithm
for k=1:d_length-1
   for j=1:Nocell
%-----------------------------------------------------------------
     %Filtering the temperature measurements:
```

```
      Tci_f(k+1,j)=Tci_f(k,j)-Ts*e0*Tci_f(k,j)+Ts*e0*Tci(k,j);
      Tco_f(k+1,j)=Tco_f(k,j)-Ts*e0*Tco_f(k,j)+Ts*e0*Tco(k,j);
      Thi_f(k+1,j)=Thi_f(k,j)-Ts*e0*Thi_f(k,j)+Ts*e0*Thi(k,j);
      Tho_f(k+1,j)=Tho_f(k,j)-Ts*e0*Tho_f(k,j)+Ts*e0*Tho(k,j);
      %Approximating derivatives with the Euler method:
      dTco_f(k,j)=(Tco_f(k+1,j)-Tco_f(k,j))/Ts;
      dTho_f(k,j)=(Tho_f(k+1,j)-Tho_f(k,j))/Ts;
%-------------------------------------------------------------------------
      %Identification part
      %Prediction error estimate:
      eps(j)=dTco_f(k,j)-(vc*Vcr_est(j))*(Tci_f(k,j)-Tco_f(k,j))-...
      ((U*(A/Nocell)*Vcr_est(j))/(cpc*rhoc))*(Tho_f(k,j)-Tco_f(k,j));
      %psi:  negative gradient of eps with respect to Vcr:
      psi(j)=vc*(Tci_f(k,j)-Tco_f(k,j))+...
      ((U*(A/Nocell))/(cpc*rhoc))*(Tho_f(k,j)-Tco_f(k,j));
      p(j)=p(j)/(lambda_Vc(j)+psi(j)^2*p(j));
      p(j)=min(p(j),pmax); %gain limiting
      Vcr_est(j)=Vcr_est(j)+p(j)*psi(j)*eps(j);
      if Vcr_est(j)==0 %protection against division by zero
         Vcr_est(j)=1/(Vc/Nocell);
      end %if
      Vc_store(k,j)=1/Vcr_est(j);
   end %for j=...
%-------------------------------------------------------------------------
   %Estimated cold side liquid volume in the whole HE:
   %Sum of the three estimates
   Vc_est(k)=Vc_store(k,1)+Vc_store(k,2)+Vc_store(k,3);
%-------------------------------------------------------------------------
   %leakage detection algorithm:  CUSUM test
   if(k>1 & k>next_d)
      SVc=SVc+Vc_est(k-1)-Vc_est(k);
      mVc=min(mVc,SVc);
      if (SVc-mVc)>gVc;
         leak(k)=-1; %leakage detected
         SVc=0;
         mVc=0;
         next_d=k+delay; %switch off detection for the next delay samples
      end
   end %if
end %for k=...
Vc_est(k+1)=Vc_est(k); %The last estimate (for plotting)
```

7.3 Simultaneous estimation of the heat transfer coefficient and the cold side liquid volume with jump- and leakage detection – all measurements available

```
%File:  AM_SIM.M
%Description:  Algorithm for simultaneously estimating the heat transfer
% coefficient and the cold side liquid volume and detecting jumps in
% the HTC and leakage in a 3-cell countercurrent heat exchanger
%Measurements:  all measurements available
%Inputs:  hott,coldt,Tco1,Tco2,Tco3,Tho1,Tho2,Tho3
%Parameters needed:  Nocell,A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Vc_store,Vc_estimate,Uc_store,Uh_store,U_store,leak,jump
%Author:  Gábor Szederkényi
%Last modified:  05 03 1998
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
e0=(vc)/(Vc/Nocell)+(U*(A/Nocell))/(cpc*rhoc*(Vc/Nocell)); %filter coeff.
d_length=length(Tco1); %length of measurement records
lambdamin=0.8; %initial minimal value of lambda_Ucc and lambda_Uch
lambda0=0.995; %growth rate of lambda after a jump was detected
lambdamax=0.99; %maximal value of lambda_Ucc and lambda_Uch
%-------------------------------------------------------------------------
%Temperature measurements:
Tci=[Tco2,Tco1,coldt]; %noisy cold inputs of cells 1,2,3 respectively
Tco=[Tco3,Tco2,Tco1]; %noisy cold outputs of cells 1,2,3
Thi=[hott,Tho1,Tho2]; %noisy hot inputs of cells 1,2,3
Tho=[Tho1,Tho2,Tho3]; %noisy hot ouputs of cells 1,2,3
%-------------------------------------------------------------------------
%Filtered measurements:
Tci_f=zeros(d_length,Nocell);
Tco_f=zeros(d_length,Nocell);
Thi_f=zeros(d_length,Nocell);
Tho_f=zeros(d_length,Nocell);
dTci_f=zeros(d_length,Nocell);
dTco_f=zeros(d_length,Nocell);
%-------------------------------------------------------------------------
%initialize variables for the HTC estimator algorithm:
qsize=1000; %queue size (samples)
V_queue=(Vc/Nocell)*ones(qsize,Nocell); %queue variable
e_Ucc=zeros(1,Nocell); %prediction error estimate
psi_Ucc=zeros(1,Nocell); %e_Ucc-s negative gradient w.r.t.U
Uc_store=zeros(d_length,Nocell); %storage for cold side HTC estimates
lambda_Ucc=0.8*ones(1,Nocell); %forgetting factor
p_Ucc=10*ones(1,Nocell); %gain
P_Umax=1e6; %max.gain value
```

```
Ucc_est=150*ones(1,Nocell); %recent cold side HTC estimates in the 3 cells
U_old=Ucc_est; %HTC estimates from the previous time step
e_Uch=zeros(1,Nocell); %prediction error estimate
psi_Uch=zeros(1,Nocell); %e_Uch-s negative gradient w.r.t.U
Uh_store=zeros(d_length,Nocell); %storage for hot side HTC estimates
lambda_Uch=0.8*ones(1,Nocell); %forgetting factor
p_Uch=10*ones(1,Nocell); %gain
Uch_est=150*ones(1,Nocell); %recent hot side HTC estimates in the 3 cells
alpha=0.5; %relative confidence for convex combination
U_store=zeros(d_length,Nocell); %storage for HTC estimates
%-------------------------------------------------------------------------
%Variables for the Vc estimator algorithm
e_Vc=zeros(1,Nocell); %prediction error estimate
psi_Vc=zeros(1,Nocell); %e_Vc-s negative gradient w.r.t.Vc
Vc_store=zeros(d_length,Nocell); %matrix for storing Vc estimates
lambda_Vc=0.8*ones(1,Nocell); %forgetting factor
P_Vc=10*ones(1,Nocell); %gain
P_Vcmax=1e6; %max.value of the gain
Vcrec_est=3*ones(1,Nocell); %recent (1/Vc) estimate
Vc_est=1/3*ones(1,Nocell); %recent Vc estimate
Vc_old=1/3*ones(1,Nocell); %previous value of Vc_est
Vc_estimate=zeros(d_length,1); %Vc estimates for the whole HE
Vc_e_f=ones(d_length,1); %filtered version of Vc_estimate
%-------------------------------------------------------------------------
%Variables for the jump detection algorithm:
SU=zeros(1,Nocell); %CUSUM
mU=zeros(1,Nocell); %minimal value
gU=5; %threshold for jump detection
jump=zeros(d_length,Nocell); %array for indicating jump detections
next_d=500*ones(1,Nocell); %jump detection starts after 500 samples
delay=500; %jump detection is switched off for 500 samples after a jump
switch=zeros(1,Nocell); %array for indicating in which cell a jump occured
%recently
sw_off=d_length*ones(1,Nocell); %variable for switching on/off Vc est.s
V_avg=zeros(1,Nocell); %average value of Vc estimates in the queue
%-------------------------------------------------------------------------
%Variables for the leakage detection algorithm:
SVc=0; %CUSUM
mVc=0; %min.value
gVc=0.025*Vc; %threshold for leakage detection:  2.5 percent decrease
leak=zeros(d_length,1); %array for indicating leakage detection
next_Vc_d=2000; %leakage detection starts after 2000 samples
delay_Vc=12000; %leak.detection is switched off for 12000 samples after
%a leakage alarm was set
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
```

```
%Beginning of the algorithm:
for k=1:d_length-1
   for j=1:Nocell
      if k>sw_off(j) %the Vc_estimate of the jth cell is switched on
         switch(j)=0;
      end
      %filtering the temperature measurements
      Tci_f(k+1,j)=Tci_f(k,j)-Ts*e0*Tci_f(k,j)+Ts*e0*Tci(k,j);
      Tco_f(k+1,j)=Tco_f(k,j)-Ts*e0*Tco_f(k,j)+Ts*e0*Tco(k,j);
      Thi_f(k+1,j)=Thi_f(k,j)-Ts*e0*Thi_f(k,j)+Ts*e0*Thi(k,j);
      Tho_f(k+1,j)=Tho_f(k,j)-Ts*e0*Tho_f(k,j)+Ts*e0*Tho(k,j);
      dTco_f(k,j)=(Tco_f(k+1,j)-Tco_f(k,j))/Ts;
      dTho_f(k,j)=(Tho_f(k+1,j)-Tho_f(k,j))/Ts;
      if k>500
         lambda_Vc=0.995*ones(1,Nocell);
         lambdamin=0.95;
      end
%-------------------------------------------------------------------------
      %HTC Estimation from the cold side:
      e_Ucc(j)=dTco_f(k,j)-(vc/(Vc_old(j)/Nocell))*(Tci_f(k,j)-Tco_f(k,j))-...
      ((Ucc_est(j)*(A/Nocell))/(cpc*rhoc*(Vc_old(j)/Nocell)))*...
      (Tho_f(k,j)-Tco_f(k,j));
      %psi_Ucc:  negative gradient of e_Ucc with respect to U
      psi_Ucc(j)=((A/Nocell)/(cpc*rhoc*(Vc_old(j)/Nocell)))*(Tho_f(k,j)-Tco_f(k,j));
      %Gradient method:
      p_Ucc(j)=p_Ucc(j)/(lambda_Ucc(j)+psi_Ucc(j)^2*p_Ucc(j));
      p_Ucc(j)=min(p_Ucc(j),P_Umax); %gain limiting
      Ucc_est(j)=Ucc_est(j)+p_Ucc(j)*psi_Ucc(j)*e_Ucc(j);
      Uc_store(k,j)=Ucc_est(j);
      lambda_Ucc(j)=min([lambdamin,lambdamax,lambda0*lambda_Ucc(j)+...
      lambdamax*(1-lambda0)]);
%-------------------------------------------------------------------------
      %HTC Estimation from the hot side:
      e_Uch(j)=dTho_f(k,j)-(vh/(Vh/Nocell))*(Thi_f(k,j)-Tho_f(k,j))-...
      ((Uch_est(j)*(A/Nocell))/(cph*rhoh*(Vh/Nocell)))*...
      (Tco_f(k,j)-Tho_f(k,j));
      psi_Uch(j)=((A/Nocell)/(cph*rhoh*(Vh/Nocell)))*...
      (Tco_f(k,j)-Tho_f(k,j));
      %Gradient method:
      p_Uch(j)=p_Uch(j)/(lambda_Uch(j)+psi_Uch(j)^2*p_Uch(j));
      p_Uch(j)=min(p_Uch(j),P_Umax); %gain limiting
      Uch_est(j)=Uch_est(j)+p_Uch(j)*psi_Uch(j)*e_Uch(j);
      Uh_store(k,j)=Uch_est(j);
      lambda_Uch(j)=min([lambdamin,lambdamax,lambda0*lambda_Uch(j)+...
      lambdamax*(1-lambda0)]);
      %Convex combination of Ucc_est and Uch_est
```

```
     U_store(k,j)=alpha*Ucc_est(j)+(1-alpha)*Uch_est(j);
%-------------------------------------------------------------------------
     %Vc estimation:
     e_Vc(j)=dTco_f(k,j)-(vc*Vcrec_est(j))*(Tci_f(k,j)-Tco_f(k,j))-...
     ((U_old(j)*(A/Nocell)*Vcrec_est(j))/(cpc*rhoc))*...
     (Tho_f(k,j)-Tco_f(k,j));
     psi_Vc(j)=vc*(Tci_f(k,j)-...
     Tco_f(k,j))+((U_old(j)*(A/Nocell))/(cpc*rhoc))*...
     (Tho_f(k,j)-Tco_f(k,j));
     P_Vc(j)=P_Vc(j)/(lambda_Vc(j)+psi_Vc(j)^2*P_Vc(j));
     P_Vc(j)=min(P_Vc(j),P_Vcmax); %gain limiting
     Vcrec_est(j)=Vcrec_est(j)+P_Vc(j)*psi_Vc(j)*e_Vc(j);
%-------------------------------------------------------------------------
     if switch(j)==1 %if a jump recently occured in the jth cell
        Vc_est(j)=V_avg(j); %then the average value in the queue is used
        Vcrec_est(j)=1/Vc_est(j);
     else
        Vc_est(j)=1/Vcrec_est(j);
     end %if switch...
     if k<1000
        Vc_est(j)=Vc/Nocell; %the nominal value of Vc is used for the
        Vcrec_est(j)=1/Vc_est(j); %first 1000 samples
     end %if k<...
     Vc_store(k,j)=Vc_est(j);
%-------------------------------------------------------------------------
     %update the FIFO queue:
     V_queue(1:qsize-1,j)=V_queue(2:qsize,j);
     V_queue(qsize,j)=Vc_est(j);
     V_avg(j)=mean(V_queue(1:qsize/2,j)); %calculate the average value
%-------------------------------------------------------------------------
     U_old(j)=U_store(k,j);
     Vc_old(j)=Vc_est(j)*Nocell;
%-------------------------------------------------------------------------
     %Jump detection in the HTC
     if(k>1 & k>next_d(j))
        SU(j)=SU(j)+U_store(k,j)-U_store(k-1,j);
        mU(j)=min(mU(j),SU(j));
        if (SU(j)-mU(j))>gU;
           lambda_Ucc(j)=0.2;
           lambda_Uch(j)=0.2;
           jump(k,j)=1;
           SU(j)=0;
           mU(j)=0;
           next_d(j)=k+delay;
           switch(j)=1;
           sw_off(j)=k+1500;
```

```
        end
      end %if
   end %for j=...
%-------------------------------------------------------------------------
   %Calculating the recent Vc estimate in the whole HE:
   if sum(switch)==0
      Vc_estimate(k)=Vc_est(1)+Vc_est(2)+Vc_est(3);
   elseif sum(switch)==3
      Vc_estimate(k)=sum(V_avg);
   else
      Vc_estimate(k)=((1-switch(1))*Vc_est(1)+(1-switch(2))*Vc_est(2)+...
      (1-switch(3))*Vc_est(3))*(Nocell/(Nocell-sum(switch)));
   end %if sum...
%-------------------------------------------------------------------------
   %Leakage detection:
   if(k>1 & k>next_Vc_d)
      SVc=SVc+Vc_e_f(k-1)-Vc_e_f(k); %The filtered Vc estimates are used
      mVc=min(mVc,SVc);
      if (SVc-mVc)>gVc & ...
         (sum(sum(V_queue(qsize-99:qsize,:)))/100)<Vc*0.99; %leakage detected
         leak(k)=-1;
         SVc=0;
         mVc=0;
         next_Vc_d=k+delay_Vc;
      end
   end %if
   Vc_e_f(k+1)=Vc_e_f(k)-Ts*e0*Vc_e_f(k)+Ts*e0*Vc_estimate(k); %filtering
end %for k=...
%-------------------------------------------------------------------------
%Updating the last estimates
Uc_store(k+1,:)=Uc_store(k,:);
Vc_estimate(k+1)=Vc_estimate(k);
```

7.4 Estimation of the cold side liquid volume with leakage detection – only cold side measurements available

```
%File:  CSO_VC.M
%Description:  Algorithm for estimating the heat transfer coefficient
% and detecting jumps in it in a 3-cell countercurrent heat exchanger
%Measurements:  all measurements available
%Inputs:  hott,coldt,Tco1,Tco2,Tco3,Tho1
%Parameters needed:  Nocell,A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Vc_store, Vc_estimate, leak
%Author:  Gábor Szederkényi
%Last modified:  05 04 1998
```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Temperature measurements:
Thi=hott; %hot inlet temperature + measurement noise
Tci=coldt; %cold inlet temp.  + noise
T1c=Tco3; %cold output temp.  of cell no.1 + noise
T2c=Tco2; %cold output temp.  of cell no.2 + noise
T3c=Tco1; %cold output temp.  of cell no.3 + noise
d_length=length(hott);
%-------------------------------------------------------------------------
%Filtered temperature data:
Thi_f=300*ones(d_length,1);
Tci_f=300*ones(d_length,1);
T1c_f=300*ones(d_length,1);
T2c_f=300*ones(d_length,1);
T3c_f=300*ones(d_length,1);
dTci_f=300*ones(d_length,1);
Thi_2ff=300*ones(d_length,1);
Thi_3ff=300*ones(d_length,1);
dT1c_f=zeros(d_length,1);
dT2c_f=zeros(d_length,1);
dT3c_f=zeros(d_length,1);
ddT1c_f=zeros(d_length,1);
ddT2c_f=zeros(d_length,1);
ddT3c_f=zeros(d_length,1);
%-------------------------------------------------------------------------
%Misc.constants:
convc=vc/(Vc/Nocell);
convh=vh/(Vh/Nocell);
Aj=A/Nocell; %heat transfer area in one cell
Vhj=Vh/Nocell; %hot side liquid volume in one cell
tc=U*Aj/(cpc*rhoc*Vc/Nocell);
th=U*Aj/(cph*rhoh*Vh/Nocell);
e1=convc+convh+tc+th; %second order filter coeff.
e0=convh*convc+convc*th+convh*tc; %second order filter coeff.
alphac=(U*Aj)/(cpc*rhoc);
tauh=(U*Aj)/(cph*rhoh*Vhj);
vhbar=vh/Vhj;
%-------------------------------------------------------------------------
%Variables needed by the identification algorithm:
lambda=0.99*ones(Nocell,1); %forgetting factor
p=10*ones(Nocell,1); %initial gain value
pmax=1e10;
Vcr_est=1/(Vc/Nocell)*ones(Nocell,1); %recent (1/Vc) estimate, jth cell
Vc_est=(Vc/Nocell)*ones(Nocell,1); %recent Vc estimate, jth cell
Vc_store=zeros(d_length,Nocell); %storage for Vc estimates, jth cell
```

```
Vc_estimate=zeros(d_length,1); %storage of Vc estimates in the whole HE
%-------------------------------------------------------------------------
%Variables used in the leakage detection algorithm:
delay=10000; %no.of samples for which the leakage is switched off
%after an alarm was set
next_d=0; %beginning of leakage detection (no.of sample)
SVc=0; %CUSUM value for leakage detection
mVc=0; %minimum value for leakage detection
gVc=Vc*0.025; %threshold value:  2.5 percent decrease
leak=zeros(d_length,1); %ouput vector that contains alarm(s)
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Beginning of the algorithm:
for k=1:d_length-2
   %Filtering the temperature measurements through a second order filter:
   Thi_f(k+2)=(2-Ts*e1)*Thi_f(k+1)-(1-Ts*e1+Ts^2*e0)*Thi_f(k)+Ts^2*e0*Thi(k);
   T2c_f(k+2)=(2-Ts*e1)*T2c_f(k+1)-(1-Ts*e1+Ts^2*e0)*T2c_f(k)+Ts^2*e0*T2c(k);
   T1c_f(k+2)=(2-Ts*e1)*T1c_f(k+1)-(1-Ts*e1+Ts^2*e0)*T1c_f(k)+Ts^2*e0*T1c(k);
   T3c_f(k+2)=(2-Ts*e1)*T3c_f(k+1)-(1-Ts*e1+Ts^2*e0)*T3c_f(k)+Ts^2*e0*T3c(k);
   Tci_f(k+2)=(2-Ts*e1)*Tci_f(k+1)-(1-Ts*e1+Ts^2*e0)*Tci_f(k)+Ts^2*e0*Tci(k);
%-------------------------------------------------------------------------
   %Approximating first derivatives:
   dT1c_f(k)=(T1c_f(k+1)-T1c_f(k))/Ts;
   dT2c_f(k)=(T2c_f(k+1)-T2c_f(k))/Ts;
   dT3c_f(k)=(T3c_f(k+1)-T3c_f(k))/Ts;
   dTci_f(k)=(Tci_f(k+1)-Tci_f(k))/Ts;
%-------------------------------------------------------------------------
   %Approximating second derivatives:
   ddT1c_f(k)=(T1c_f(k+2)-2*T1c_f(k+1)+T1c_f(k))/Ts^2;
   ddT2c_f(k)=(T2c_f(k+2)-2*T2c_f(k+1)+T2c_f(k))/Ts^2;
   ddT3c_f(k)=(T3c_f(k+2)-2*T3c_f(k+1)+T3c_f(k))/Ts^2;
%-------------------------------------------------------------------------
   %Identification:  cell no.1
   %Negative gradient of eps1 w.r.t.Vcr_est:
   psi1=-(vc+alphac)*dT1c_f(k)-(vc*vhbar+vc*tauh+vhbar*alphac)*T1c_f(k)...
   +vc*dT2c_f(k)+vc*(vhbar+tauh)*T2c_f(k)+alphac*vhbar*Thi_f(k);
   %Prediction error estimate:
   eps1=ddT1c_f(k)+(vhbar+Vcr_est(1)*vc+Vcr_est(1)*alphac+tauh)*...
   dT1c_f(k)+(Vcr_est(1)*vc*vhbar+Vcr_est(1)*vc*tauh+vhbar*...
   Vcr_est(1)*alphac)*T1c_f(k)-...
   Vcr_est(1)*vc*dT2c_f(k)-Vcr_est(1)*vc*(vhbar+tauh)*T2c_f(k)-...
   Vcr_est(1)*alphac*vhbar*Thi_f(k);
   %Gradient method:
   p(1)=p(1)/(lambda(1)+psi1^2*p(1));
   p(1)=min(p(1),pmax);
   Vcr_est(1)=Vcr_est(1)+p(1)*psi1*eps1;
```

```
   Vc_est(1)=1/Vcr_est(1);
   Vc_store(k,1)=Vc_est(1);
%-------------------------------------------------------------------------
   %Identification:  cell no.2
   %Calculating Thi_2ff according to the transfer function:
   Thi_2ff(k+1)=Thi_2ff(k)+Ts*(-vhbar*Thi_2ff(k)-...
   tauh*Thi_2ff(k)+vhbar*Thi_f(k)+tauh*T1c_f(k));
   %Negative gradient of eps2 w.r.t.Vcr_est:
   psi2=-(vc+alphac)*dT2c_f(k)-...
   (vc*vhbar+vc*tauh+vhbar*alphac)*T2c_f(k)+vc*dT3c_f(k)+vc*(vhbar+tauh)*...
   T3c_f(k)+alphac*vhbar*Thi_2ff(k);
   %Prediction error estimate:
   eps2=ddT2c_f(k)+(vhbar+Vcr_est(2)*vc+Vcr_est(2)*alphac+tauh)*...
   dT2c_f(k)+(Vcr_est(2)*vc*vhbar+Vcr_est(2)*vc*tauh+vhbar*...
   Vcr_est(2)*alphac)*T2c_f(k)-Vcr_est(2)*vc*dT3c_f(k)-Vcr_est(2)*...
   vc*(vhbar+tauh)*T3c_f(k)-Vcr_est(2)*alphac*vhbar*Thi_2ff(k);
   %Gradient method:
   p(2)=p(2)/(lambda(2)+psi2^2*p(2));
   p(2)=min(p(2),pmax);
   Vcr_est(2)=Vcr_est(2)+p(2)*psi2*eps2;
   Vc_est(2)=1/Vcr_est(2);
   Vc_store(k,2)=Vc_est(2);
   %-----------------------------------------------------------------------
   %Identification:  cell no.3
   %Calculating Thi_3ff according to the transfer function:
   Thi_3ff(k+1)=Thi_3ff(k)+Ts*(-vhbar*Thi_3ff(k)-...
   tauh*Thi_3ff(k)+vhbar*Thi_2ff(k)+tauh*T2c_f(k));
   %Negative gradient of eps3 w.r.t.Vcr_est:
   psi3=-(vc+alphac)*dT3c_f(k)-(vc*vhbar+vc*tauh+vhbar*alphac)*T3c_f(k)+...
   vc*dTci_f(k)+vc*(vhbar+tauh)*Tci_f(k)+alphac*vhbar*Thi_3ff(k);
   %Prediction error estimate:
   eps3=ddT3c_f(k)+(vhbar+Vcr_est(3)*vc+Vcr_est(3)*alphac+tauh)*...
   dT3c_f(k)+(Vcr_est(3)*vc*vhbar+Vcr_est(3)*vc*tauh+vhbar*...
   Vcr_est(3)*alphac)*T3c_f(k)-Vcr_est(3)*vc*dTci_f(k)-Vcr_est(3)*...
   vc*(vhbar+tauh)*Tci_f(k)-Vcr_est(3)*alphac*vhbar*Thi_3ff(k);
   %Gradient method:
   p(3)=p(3)/(lambda(3)+psi3^2*p(3));
   p(3)=min(p(3),pmax);
   Vcr_est(3)=Vcr_est(3)+p(3)*psi3*eps3;
   Vc_est(3)=1/Vcr_est(3);
   Vc_store(k,3)=Vc_est(3);
%-------------------------------------------------------------------------
   %Estimate for the cold side volume in the whole HE:
   Vc_estimate(k)=Vc_store(k,1)+Vc_store(k,2)+Vc_store(k,3);
%-----------------------------------------------------------------------
   %leakage detection algorithm:  CUSUM test
```

```
   if(k>1 & k>next_d)
      SVc=SVc+Vc_estimate(k-1)-Vc_estimate(k);
      mVc=min(mVc,SVc);
      if (SVc-mVc)>gVc;
         leak(k)=-1; %leakage detected
         SVc=0;
         mVc=0;
         next_d=k+delay; %switch off detection for the next delay samples
      end
   end %if
end %for
%-------------------------------------------------------------------------
%Setting the last values for plotting:
Vc_estimate(k+1)=Vc_estimate(k);
Vc_estimate(k+2)=Vc_estimate(k+1);
```

7.5 Estimation of the heat transfer coefficient with jump detection – simplified model

```
%File:  SIMP_HTC
%Description:  Algorithm for estimating the heat transfer coefficient
% and detecting jumps in it in a 3-cell countercurrent heat exchanger
%Measurements:  simplified model (only the hot and cold side inlet and
% outlet temperatures are available)
%Inputs:  hott,coldt,Tho3,Tco3
%Parameters needed:  A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Uc_store,Uh_store,U_store,jump;
%Author:  Gábor Szederkényi
%Last modified:  05 04 1998
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
Nocell=1; %number of cells in the model used for identification
e0=(vc)/(Vc/Nocell)+(U*(A/Nocell))/(cpc*rhoc*(Vc/Nocell)); %filter coeff.
d_length=length(coldt); %size of the measurement records
%Temperature measurements:
Tci=coldt; %cold inlet temperature
Tco=Tco3; %cold outlet temperature
Thi=hott; %hot inlet temperature
Tho=Tho3; %hot outlet temperature
%-------------------------------------------------------------------------
%Filtered temperatures:
Tci_f=300*ones(d_length,1);
Tco_f=300*ones(d_length,1);
Thi_f=300*ones(d_length,1);
Tho_f=300*ones(d_length,1);
```

```
dTco_f=zeros(d_length,1);
dTho_f=zeros(d_length,1);
%------------------------------------------------------------------------
%Variables for the identification algorithm:
yc=zeros(d_length,1); %output of the discrete time model - cold side
uc=zeros(d_length,1); %input of the discrete time model - cold side
Uc_store=zeros(d_length,1); %storage for Uc estimates
pc=10; %gain
tauc_est=0; %recent estimate for tauc
lambdac=0.96; %forgetting factor
yh=zeros(d_length,1); %output of the discrete time model - hot side
uh=zeros(d_length,1); %input of the discrete time model - hot side
Uh_store=zeros(d_length,1); %storage for Uh estimates
ph=10; %gain
tauh_est=0; %recent estimate for tauh
lambdah=0.96; %forgetting factor
lambdamin=0.98; %min.value of lambdac and lambdah
lambdamax=0.98; %max.value of lambdac and lambdah
lambda0=0.995; %growth rate of lambda after a jump
alpha=0.5; %constant for convex combination
U_store=zeros(d_length,1); %storage for Uc estimates
pmax=1e6; %max.gain value
%------------------------------------------------------------------------
%Variables for the jump detection algorithm
jump=zeros(d_length,1); %array for indicating jumps
next_d=1500; %jump detection begins after 1500 samples
delay=200; %jump detection is switched off for 200 samples after an alarm
She2=0; %CUSUM
mhe2=0; %min.value
g=10; %threshold for jump detection
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%Beginning of the main part
for k=1:d_length-1;
   %Filtering the measurement data
   Tci_f(k+1)=Tci_f(k)-Ts*e0*Tci_f(k)+Ts*e0*Tci(k);
   Tco_f(k+1)=Tco_f(k)-Ts*e0*Tco_f(k)+Ts*e0*Tco(k);
   Thi_f(k+1)=Thi_f(k)-Ts*e0*Thi_f(k)+Ts*e0*Thi(k);
   Tho_f(k+1)=Tho_f(k)-Ts*e0*Tho_f(k)+Ts*e0*Tho(k);
   dTco_f(k)=(Tco_f(k+1)-Tco_f(k))/Ts;
   dTho_f(k)=(Tho_f(k+1)-Tho_f(k))/Ts;
%------------------------------------------------------------------------
   %Recursive least squares method - cold side equation:
   yc(k+1)=dTco_f(k)-(vc/(Vc/Nocell))*(Tci_f(k)-Tco_f(k));
   uc(k)=Tho_f(k)-Tco_f(k);
   pc=pc/(lambdac+uc(k)^2*pc);
```

```
   pc=min(pmax,pc);
   tauc_est=tauc_est+pc*uc(k)*(yc(k+1)-tauc_est*uc(k));
   Uc_store(k)=(tauc_est*cpc*rhoc*(Vc/Nocell))/(A/Nocell);
   lambdac=min([lambdamin,lambdamax,lambda0*lambdac+lambdamax*(1-lambda0)]);
%-------------------------------------------------------------------------
%Recursive least squares method - hot side equation:
   yh(k+1)=dTho_f(k)-(vh/(Vh/Nocell))*(Thi_f(k)-Tho_f(k));
   uh(k)=Tco_f(k)-Tho_f(k);
   ph=ph/(lambdah+uh(k)^2*ph);
   pc=min(pmax,ph);
   tauh_est=tauh_est+ph*uh(k)*(yh(k+1)-tauh_est*uh(k));
   Uh_store(k)=(tauh_est*cph*rhoh*(Vh/Nocell))/(A/Nocell);
   lambdah=min([lambdamin,lambdamax,lambda0*lambdah+lambdamax*(1-lambda0)]);
   %Convex combination:
   U_store(k)=alpha*Uc_store(k)+(1-alpha)*Uh_store(k);
%-------------------------------------------------------------------------
   %Jump detection algorithm
   if k>1 & k>next_d
      %cumulative sum of updates of the value of the htc:
      She2=She2+U_store(k)-U_store(k-1);
      %minimum cumulative sum:
      mhe2=min(mhe2,She2);
      if She2-mhe2>g
         %jump detected, adjust forgetting factor, switch off jump detection
         %for the next 200 samples.
         jump(k)=1;
         She2=0;
         mhe2=0;
         next_d=k+delay;
         lambdac=0.8;
         lambdah=0.8;
      end; %if She2
   end; %if k>...
end %for k=1;
%-------------------------------------------------------------------------
U_store(k+1)=U_store(k); %setting the last value
```

7.6 Estimation of the cold side liquid volume with leakage detection – simplified model

```
%File:  SIMP_VC.M
%Description:  Algorithm for estimating the cold side liquid volume
% and detecting leakage in a 3-cell countercurrent heat exchanger
%Measurements:  simplified model (only the hot and cold side inlet and
% outlet temperatures are available)
```

```
%Inputs:  hott,coldt,Tho3,Tco3
%Parameters needed:  A,vc,vh,Vc,Vh,cpc,cph,rhoc,rhoh,Ts
%Outputs:  Vc_store,leak
%Author:  Gábor Szederkényi
%Last modified:  05 03 1998
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
Nocell=3; %number of cells used by the algorithm
e0=(vc)/(Vc/Nocell)+(U*(A/Nocell))/(cpc*rhoc*(Vc/Nocell)); %filter coeff.
d_length=length(Tco1); %length of measurement records
%-------------------------------------------------------------------------
%Temperature measurements:
Tci=[coldt]; %noisy cold inlet temperature
Tco=[Tco3]; %noisy cold outlet temperature
Thi=[hott]; %noisy hot inlet temperature
Tho=[Tho3]; %noisy hot outlet temperature
%-------------------------------------------------------------------------
%Allocating memory for filtered measurements:
Tci_f=300*ones(d_length,1);
Tco_f=300*ones(d_length,1);
Thi_f=300*ones(d_length,1);
Tho_f=300*ones(d_length,1);
dTco_f=zeros(d_length,1);
dTho_f=zeros(d_length,1);
%-------------------------------------------------------------------------
%Variables used by the identification algorithm:
lambda_Vc=0.99; %forgetting factor
p=10; %gain
pmax=1e6; %max.gain value
Vcrec_est=1/(Vc/Nocell); %initial value for (1/Vc) estimate
Vc_store=zeros(d_length,1); %storage for Vc estimates
y=0; %output of the discrete time model
u=0; %input of the discrete time model
%-------------------------------------------------------------------------
%Variables used by the leakage detection algorithm:
delay=10000; %no.of samples for which leakage detection is switched off
%after leakage detected
next_d=500; %leakage detection starts after 500 samples
SVc=0; %CUSUM value for leakage detection
mVc=0; %min.value
gVc=Vc*0.05; %threshold
leak=zeros(d_length,1); %array for indicating leakage detection
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Beginning of the main part
for k=1:d_length-1
```

```
   %filtering the temperature measurements:
   Tci_f(k+1)=Tci_f(k)-Ts*e0*Tci_f(k)+Ts*e0*Tci(k);
   Tco_f(k+1)=Tco_f(k)-Ts*e0*Tco_f(k)+Ts*e0*Tco(k);
   Thi_f(k+1)=Thi_f(k)-Ts*e0*Thi_f(k)+Ts*e0*Thi(k);
   Tho_f(k+1)=Tho_f(k)-Ts*e0*Tho_f(k)+Ts*e0*Tho(k);
   dTco_f(k)=(Tco_f(k+1)-Tco_f(k))/Ts; %approximating the derivatives
   dTho_f(k)=(Tho_f(k+1)-Tho_f(k))/Ts;
%--------------------------------------------------------------------------
   %Recursive least squares method:
   y=dTco_f(k);
   u=vc*(Tci_f(k)-Tco_f(k))+((U*(A/Nocell))/(cpc*rhoc))*(Tho_f(k)-Tco_f(k));
   p=p/(lambda_Vc+u^2*p);
   p=min(p,pmax);
   Vcrec_est=Vcrec_est+p*u*(y-Vcrec_est*u);
   if Vcrec_est==0
      Vcrec_est=1/Vc;
   end
   Vc_store(k)=1/Vcrec_est;
%--------------------------------------------------------------------------
   %Leakage detection algorithm:
   if(k>1 & k>next_d)
      SVc=SVc+Vc_store(k-1)-Vc_store(k);
      mVc=min(mVc,SVc);
      if (SVc-mVc)>gVc;
         leak(k)=-1; %leakage detected
         SVc=0;
         mVc=0;
         next_d=k+delay;
      end
   end %if
end %for k=...
%--------------------------------------------------------------------------
Vc_store(k+1)=Vc_store(k);
```

References

[1] Karl J. Aström and Björn Wittenmark. *Computer Controlled Systems*. Prentice Hall, New Jersey, 1990.

[2] M. Basseville and I.V. Nikiforov. *Detection of Abrupt Changes. Theory and Practice*. Prentice Hall, London, 1993.

[3] Katalin Hangos, József Bokor, and Miklós Gerzson. *Computer Controlled Systems*. Veszprémi Egyetemi Kiadó, Veszprém, 1995.

[4] Rolf Isermann. Process fault detection based on modeling and estimation methods - a survey. *Automatica*, 20:387–404, 1984.

[5] L. Ljung. *System Identification - Theory for the User*. University of Linköping, Sweden, Linköping, 1984.

[6] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. The MIT Press, Massachusetts, 1983.

[7] Mohammad-Ali Massoumnia, George C. Verghese, and Alan S. Willsky. Failure detection and identification. *IEEE Transactions on Automatic Control*, 34:316–321, 1989.

[8] R.H. Middleton and G.C. Goodwin. *Digital Control and Estimation. A Unified Approach*. Prentice Hall, London, 1990.

[9] László Schnell. *Jelek és rendszerek Méréstechnikája*. Tankönyvkiadó, Budapest, 1991.

[10] E.I. Varga, K.M. Hangos, and F. Szigeti. Controllability and observability of heat exchanger networks in the time varying parameter case. *Control Engineering Practice*, 3:1409–1419, 1995.

[11] E. Weyer and K.M. Hangos. Grey box fault detection in heat exchanger networks. In R.J. Patton and J. Chen, editors, *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS'97*, volume 1, pages 187–192. Pergamon, Hull, 1997.

[12] A.S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12:601–611, 1976.

[13] E.A. Wolff, K.W. Mathisen, and S. Skogestad. Dynamics and controllability of heat exchanger networks. In L. Puigjaner and A. Espuna, editors, *Proc. of Computer Oriented Process Engineering (COPE-91)*, pages 117–122. 1991.

[14] *MATLAB (V4.2c1) User's Guide*, 1994.

[15] *SIMULINK (V1.3c) User's Guide*, 1994.

[16] *Documentation for The Fault Detection and Isolation Matlab Toolbox, EU-Copernicus Project CT94-0237*, 1998.