UNIVERSITY OF ICELAND

Detection of Fouling: The Effectiveness Ratio Method

A dissertation by

Helga Ingimundardóttir

60 ECTS thesis submitted in partial fulfilment of a

Magister Scientiarum degree in Computational Engineering

Faculty of Engineering

School of Engineering and Natural Sciences

University of Iceland

Reykjavík, February 2008

Advisors: Dr. Tómas Philip Rúnarsson and Dr. Sylvain Lalot
Faculty Representative: Dr. Ólafur Pétur Pálsson

Detection of Fouling: The Effectiveness Ratio Method
60 ECTS thesis submitted in partial fulfilment of a *Magister Scientiarum* degree in Computational Engineering

University of Iceland
Faculty of Engineering
VR-II, Hjarðarhaga 2-6, IS-107 Reykjavík, Iceland
Phone +354 525 4648, Fax +354 525 4632
verk@hi.is
www.hi.is

ISBN: XX
Printed in Iceland by Háskólaprent hf, 2010

Abstract

The aim of this study is to investigate the possibility of using models to detect fouling in a cross-flow heat exchangers, by only using measurements that are attainable in normal operation of the heat exchanger, i.e., the hot and cold inlet and outlet temperatures and the mass flow rates for the hot and cold fluids. Real data of cross-flow heat exchangers is not easily attainable so simulated data was used. Half of the data is of a clean heat exchanger, and the other half is of a fouled heat exchanger. There should be no detection of fouling for a clean heat exchanger, but should be as soon as possible for a fouled one.

The on-line detection of fouling is used by a new and more general method that also takes into account that the input can be varying. The new method finds a threshold for fouling based on the estimate of the steady states of the effectiveness; this is called the *effectiveness ratio method* (ERM). The estimate of the steady states can be done by applying a wavelet transform since the transform is localised in both time and frequency. A simpler approach is also applied, that of moving average, to show how the modelling approach needs to be chosen carefully to get a good tool for detection of fouling.

The parameters of the method need to be chosen carefully, e.g. the step size of the algorithm and the compromise between the frequency and time localisation, thus a multiple objective genetic algorithm is implemented for the optimisation. The new ERM approach is also compared with published results for common means of detection of fouling using Kalman filters.

Ágrip

Markmið þessarar rannsóknar er að kanna möguleikann á að nota líkan til að greina útfellingar í krossflæðis varmaskiptum, með því að nota einvörðungu mælingar sem eru í hefðbundnum rekstri varmaskiptisins, þ.e. hitastig við inntak og úttak og streymi heita og kalda vatnsins. Erfitt er að nálgast raungögn fyrir krossflæðis varmaskipta og því byggir rannsóknin á hermdum gögnum. Helmingur gagnanna er miðaður við hreina varmaskipta og hinn helmingurinn við varmaskipta með útfellingum. Útfellingar ættu ekki að greinast í hreinum varmaskipti, en ætti að greina eins fljótt og auðið er í útfelldum varmaskipti.

Kynnt er ný og alhæfðari rauntíma aðferð til að greina útfellingar, en hún tekur tillit til síbreytilegra gagna. Nýja aðferðin, sem kölluð er *skilvirknis hlutfalla aðferðin* (ERM), finnur þröskuld byggðan á jafnvægisástandi skilvirkninnar. Nálgunin á jafnvægisástandinu er fundin með því að beita smábylgjuummyndun eða „wavelet“ ummyndun, sem er bæði bundin í tíma og tíðni. Jafnframt er litið á hlaupandi meðaltal til að sýna fram á að nálgunaraðferðin þarf að vera vandlega valin til að greina útfellingar nægilega vel.

Stikar aðferðarinnar þurfa að vera valdir gaumgæfilega, t.a.m. skrefstærð reikniritsins eða ákvörðunin um hvort leggja eigi áherslu á tíma eða tíðni nálgunarinnar. Stikarnir eru fundnir með fjölmarkmiða bestunarreikniriti með erfðafræðilegu ívafi. Nýja ERM aðferðafræðin er borin saman við birtar niðurstöður á greiningu útfellinga með hjálp Kalman sía.

Acknowledgments

This thesis owes its existence to the help, support, and inspiration of many. Firstly, I would like to express my sincere appreciation and gratitude to my supervisors, Prof. Dr. Tómas Philip Rúnarsson and Prof. Dr. Sylvain Lalot, who were abundantly patient and offered invaluable assistance and guidance. Deepest gratitude are also due to the faculty representative, Prof. Dr. Ólafur Pétur Pálsson, without whose assistance this study would not have been successful.

Special thanks to all my graduate friends, especially Oddgeir Guðmundsson for sharing literature, data and assistance.

I would also like to convey thanks to the French government, the *Erasmus* programme and the French-Icelandic *Jules Verne* programme for providing financial means that gave me the opportunity to live and study in Valenciennes, France for five months and to be able to attend the *Heat Exchanger Fouling and Cleaning* conference in Schladming, Austria last year.

Finally, I owe special gratitude to my beloved family for continuous and unconditional support of all my undertakings, scholastic or otherwise. Not forgetting my best friends who have always been there for me.

Reykjavík, February 2010

Helga Ingimundardóttir

Contents

List of Figures

List of Tables

Nomenclature

T	temperature	K
A	area of the convection surface	m^2
C	specific heat	$\frac{J}{kgK}$
M	mass of fluid in one cell	kg
$\dot{m}$	mass flow rate	$\frac{kg}{s}$
R_f	fouling factor	$\frac{m^2K}{W}$
U	overall heat transfer coefficient	$\frac{W}{m^2K}$
h	convection coefficient	$\frac{W}{m^2K}$
α	convection coefficient on the hot side	$\frac{W}{m^2K}$
β	convection coefficient on the cold side	$\frac{W}{m^2K}$
k	thermal conductivity	$\frac{W}{mK}$
ρ	density	$\frac{kg}{m^3}$
ν	kinematic viscosity	$\frac{m^2}{s}$
$\tau_{h,I,J}$	residence time in one cell	s
$\gamma_{I,J}$	inverse of the response time for the hot cell	$\frac{1}{s}$
$\delta_{I,J}$	inverse of the response time for the cold cell	$\frac{1}{s}$
Dh	hydraulic diameter	m
V	mean fluid velocity	$\frac{m}{s}$
$ntu^h_{I,J}$	number of transfer units for the hot side	
$ntu^c_{I,J}$	number of transfer units for the cold side	
Pr	Prandlt number	
Re	Reynolds number	
Nu	Nusselt number	

$\mathcal{L}\{f\}$	the Laplace transform of function f	
$\mathcal{N}(\mu, \Sigma)$	multivariate normal distribution with mean μ and covariance matrix Σ	
ζ	threshold for ERM	
θ	offset for ERM	
κ	dimension of WT's basis	
ξ	scale of WT	
ϱ	boundary coefficient of WT	
η	parameter for the hot side of EKF	$\frac{W}{m^2K}$
ν	parameter for the cold side of EKF	$\frac{W}{m^2K}$
ω	length of the sliding window for MA	
f_1	ratio of positive detection of fouling on fouled CFHE	
f_2	ratio of positive non detection of fouling on clean CFHE	
f_3	mean detection time of fouling on fouled CFHE	

Subscripts and superscript

c	cold side
h	hot side
p	separating plate
I	in the middle of the cell
i	on the left side of the cell number I
J	in the middle of the cell
j	on the bottom side of the cell number J
in	inlet
out	outlet
T	transpose

Abbreviations

Eq.	equation
Fig.	figure
CFHE	cross-flow heat exchanger
ERM	effectiveness ratio method
MA	moving average
WT	wavelet transform
MOOP	multi-objective optimisation problem
MOGA	multi-objective genetic algorithm
KF	Kalman filter
EKF	extended Kalman filter

Chapter 1

Introduction

Research concerning fouling in heat exchangers is very prolific. Studies are divided into different fields: principles of fouling; the mitigation of fouling; and monitoring of fouling. Biannually an international conference on Heat Exchanger Fouling and Cleaning is held to present the latest research and technological developments in fouling and cleaning strategies. Focus of this study is monitoring of fouling, specifically modelling fouling to detect it as soon as possible.

1.1 Motivation

Fouling is when unwanted materials accumulate on solid surfaces, e.g. bio film builds, or the surface starts to corrode, thus leading to an impediment to the performance of the system it is taking place. For heat exchangers, fouling results with worse hydraulic performance, and reduced thermal efficiency of the effected equipment since there is an additional resistance to the heat flow. For heat transfer between two fluids, fouling is inevitable. Once fouling is present either the use of a hotter fluid or an increased hot fluid mass flow rate

is required to get the same desired outlet temperature for the cold fluid. This would come at a greater cost to the consumer.

Besides the lack of thermal effectiveness, fouling can have dire consequences, e.g. according to (Casanueva-Robles and Bott, 2005) they can create blockages or cause unnecessary CO_2 exhaust. Hence it is very important to detect fouling as soon as possible both because of environmental standpoint and having the system working at full capacity for minimal cost.

The classical methods for detecting fouling according to (S. Lalot and Desmet, 2007) are:

1. Examination of the heat transfer coefficient (or effectiveness);
2. Simultaneous observations of pressure drops and mass flow rates;
3. Temperature measurements;
4. Ultrasonic or electrical measurements;
5. Weighing of heat exchanger plates.

There are some drawback to these methods: The first three methods require that the system has reached a steady state; the fourth monitors only local fouling; and the fifth requires the process to be stopped. Another approach would be modelling the heat exchanger and look for discrepancies between the model predictions and actual measurements.

Model-based methods that have been used can either be based on physical parameters such as extended Kalman filters (S. Lalot and Desmet, 2007); or *black-box* methods such as artificial neural networks (Lalot and Lecoeuche, 2003); fuzzy logic model (F. Delmotte and Dambrine, 2008); and a recursive

subspace identification algorithm (Lalot and Mercère, 2008). These papers either model electrical circulation heaters or tube-in-tube heat exchangers. All the proposed methods are focusing on reducing computational cost so on-line implementation for detection of fouling is possible, without requiring steady state and only using observable inputs and outputs. Depending on what kind of fouling is being focused on, this can be extremely important, e.g. in the dairy industry fouling can occur in a matter of a few hours; but for crude oil it may take a few days for fouling to exist in its preheat train exchangers. Estimation is thus feasible whilst the heat exchanger is in use.

1.2 Contribution and objectives

For this study the main contribution and objectives are:

1. Effectiveness ratio method (ERM);
2. Wavelet transform to model ERM;
3. Multiple objective genetic algorithm to estimate the parameters in ERM.

ERM uses the estimated steady state of the system's effectiveness, and at each time step the effectiveness's average is used as a reference value. The ratio between the estimated effectiveness and its average is then examined. If the ratio falls below a certain threshold, fouling is detected, if not then the heat exchanger is assumed to be still clean.

The system's effectiveness never reaches a steady state, since the inputs are always simultaneously varying. Thus an approximation of the effectiveness

is needed for ERM. Wavelet transform is implemented since it can be both localised in the time and frequency domain, depending on where the emphasis needs to lie. Instead of working in the wavelet transform paradigm an equivelent filter is applied to the instantaneous effectiveness signal. First a low-pass analysis filter and a down sampling by two are applied iteratively to the signal to obtain the approximation coefficients. Afterwards an up sampling by 2 and a low-pass synthesis filter is applied to the approximation coefficients iteratively, yielding the original signal without noise, which is the desired estimate of the effectiveness's steady state. This approach was first presented in (Ingimundardóttir and Lalot, 2009).

There are several parameters that need to be chosen for ERM. First there is the choice of ERM's step size, or offset. For the wavelet transform, e.g. with a Daubechies basis, the parameters are the dimension of the basis, the scale of the transform and its boundary condition coefficient. Finding the optimal set of decision variables is done via MOGA optimisation, where its objectives are: if CFHE is clean there should be no detection of fouling; and conversely if CFHE is fouled there should be detection of fouling and preferably as soon as possible.

1.3 Overview

In chapter 2 there is a general description of heat exchangers, focusing on cross-flow heat exchangers and how they can be modelled and subsequently simulated. With simulated data, for both clean and fouled cross-flow heat exchangers, detection of fouling is found via a model-based method called the

effectiveness ratio method (ERM), discussed in detail in chapter 3. ERM relies on a sub method for the approximation of the heat exchanger's steady state of the effectiveness, the sub methods introduced are either moving average or wavelet transform. The parameters of ERM are optimised using a *multiple objective genetic algorithm* (MOGA). Results obtained from MOGA and published results using an extended Kalman filter are presented in chapter 4. Comparison between the different methods of detection on fouling are discussed along with future work in chapter 5.

Chapter 2

Heat Exchangers: Modelling and Simulation

A heat exchanger is a device built for efficient heat transfer from one medium to another. They are widely used in domestic and industrial applications. There are several different types of heat exchangers. They may be classified according to their flow arrangement, namely parallel-flow; counter-flow; and cross-flow heat exchangers. Furthermore they can be classified by their construction. The most common types being shell-and-tube heat exchanger and plate heat exchanger.

A plate heat exchanger uses metal plates to transfer heat between two fluids. The fluids are exposed to a larger surface area, resulting in a higher heat transfer efficiency despite small physical size than compared to other conventional heat exchangers. They are generally compact; can be used for various types of fluids and conditions; can be easily dismantled for cleaning and/or replacing individual damaged plates; and have a low manufacturing cost. A typical plate

heat exchanger in the dairy industry is depicted in Figure 2.1. For further information the reader can refer to (Guðmundsson, 2008) and (Çengel and Boles, 2007).

Since the plate heat exchangers are both economic and efficient, and therefore common in practice, this study will focus on simple cross-flow plate heat exchanger (CFHE), having both fluids unmixed and flowing perpendicular to each other. The fluids are separated by a metal plate; the other side of the fluid channels being perfectly insulated, the cold fluid flows along the x-axis and the hot fluid flows along the y-axis (see Figure 2.2).

Figure 2.1. Principles of flow and heat transfer in a plate heat exchanger, figure from Tetra Pak's Dairy Processing Handbook (Bylund, 1995).

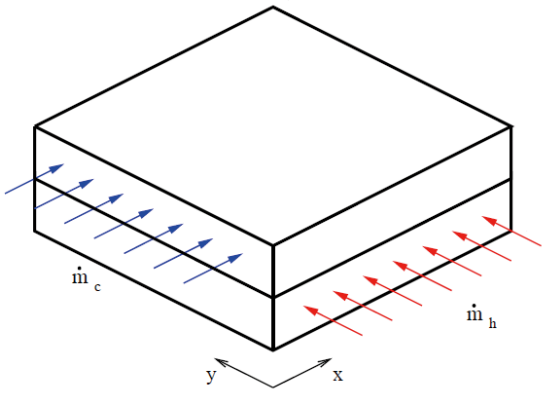

Figure 2.2. Cross-flow heat exchanger, having both fluids unmixed.

2.1 Simulation

The simulations are primarily to calculate the outflow temperatures for given mass flow rates and inflow temperatures. It is noted that in an accurate simulation there are no measurement errors that would be present in real data, thus need to be added if appropriate.

When it comes to modelling a heat exchanger, one must take into consideration the laws of thermodynamics. Especially the conservation of energy, in particular where heat is a form of energy. Entropy of an isolated macroscopic system never decreases, and consequently heat transfer is the transition of thermal energy from a hotter medium to a cooler medium, not vice versa.

One must also consider the means of how heat transfers. Heat transfer can either be due to conduction; convection and/or thermal radiation. Conduction is heat transfer by direct contact of particles of matter, i.e., it is due to com-

bination of vibrations between adjacent atoms and free electrons moving from atom to atom. Convection takes place through diffusion and advection in fluids. The greater the fluid motion, the greater the convection heat transfer. Radiation is heat transfer through electromagnetic waves, thus no medium is necessary for it to occur, (Çengel and Boles, 2007).

In a plate heat exchangers, heat is transferred by convection from the hot fluid to the separating plate, by conduction through the separating plate, and by convection from the separating plate to the cold fluid.

Last but not least, to model a heat exchanger, it is necessary to divide it into *cells* (i, j), $i \in \{1, ..., n_i\}, j \in \{1, ..., n_j\}$, as depicted in Figure 2.3.

For this study there are some assumptions for ease of calculations: The heat exchanger is perfectly insulated, i.e., heat loss to the surroundings is negligible; There is uniform temperature in each cell of the heat exchanger; The specific heat capacities are constant throughout the heat exchanger.

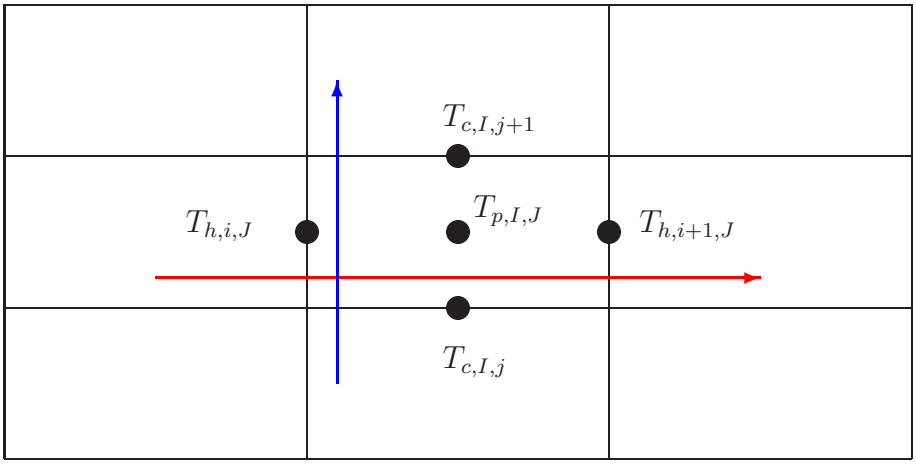

Figure 2.3. Discretisation of the CFHE.

Using the aforementioned laws of thermodynamics, the energy change in a cell is equal to the energy flow in the fluid in the cell and the energy that is transferred from or to the cell. Hence according to (Ingimundardóttir and Lalot, 2009), in each cell (I, J) there are three differential equations describing the energy balance that have to hold, namely, one for the hot fluid:

$$\begin{aligned} M_h c_h \frac{d}{dt} T_{h\cdot,i,J} &= \dot{m}_{h,J} c_h \left(T_{h,i,J} - T_{h,i+1,J}\right) \\ &\quad + \alpha_{I,J} A_{h,I,J} \left(T_{p,I,J} - \frac{T_{h,i,J} + T_{h,i+1,J}}{2}\right), \end{aligned} \tag{2.1}$$

where $T_{h\cdot,i,J} := \left(\frac{T_{h,i,J}+T_{h,i+1,J}}{2}\right)$. One for the cold fluid:

$$\begin{aligned} M_c c_c \frac{d}{dt} T_{c\cdot,I,J} &= \dot{m}_{c,I} c_c \left(T_{c,I,j} - T_{c,I,j+1}\right) \\ &\quad + \beta_{I,J} A_{c,I,J} \left(T_{p,I,J} - \frac{T_{c,I,j} + T_{c,I,j+1}}{2}\right), \end{aligned} \tag{2.2}$$

where $T_{c\cdot,I,J} := \left(\frac{T_{c,I,j}+T_{c,I,j+1}}{2}\right)$. Lastly one for the separating plate:

$$\begin{aligned} M_p c_p \frac{d}{dt} T_{p,I,J} &= -\alpha_{I,J} A_{h,I,J} \left(T_{p,I,J} - \frac{T_{h,i,J} + T_{h,i+1,J}}{2}\right) \\ &\quad - \beta_{I,J} A_{c,I,J} \left(T_{p,I,J} - \frac{T_{c,I,j} + T_{c,I,j+1}}{2}\right), \end{aligned} \tag{2.3}$$

for all $I \in \{1, ..., n_i\}$ and $J \in \{1, ..., n_j\}$, and each temperature is taken at a given time t.

Lets define the following variables:

$$ntu^h_{I,J} = \frac{\alpha_{I,J} A_{h,I,J}}{\dot{m}_{h,J} c_h} \tag{2.4}$$

$$ntu^c_{I,J} = \frac{\beta_{I,J} A_{c,I,J}}{\dot{m}_{c,I} c_c} \tag{2.5}$$

as the number of transfer units for the hot and cold cells respectively, and

$$\tau_{h,I,J} = \frac{M_h}{\dot{m}_{h,J}} \tag{2.6}$$

as the residence time in one cell and

$$\gamma_{I,J} = \frac{\alpha_{I,J} A_{h,I,J}}{M_p c_p} \tag{2.7}$$

$$\delta_{I,J} = \frac{\beta_{I,J} A_{c,I,J}}{M_p c_p} \tag{2.8}$$

as the inverses of response times for the hot and cold cells respectively.

Using these newly defined variables, it is possible to rearrange the terms of Equation (2.1), to get an equivalent differential equation for the hot fluid,

$$\begin{aligned} \frac{d}{dt} T_{h\cdot,i,J} &= \frac{1}{\tau_{I,J}} \left(2T_{h,i,J} - 2T_{h,i+1,J}\right) + \frac{ntu^h_{I,J}}{\tau_{I,J}} \left(2T_{p,I,J} - T_{h,i,J} - T_{h,i+1,J}\right), \\ &= \underbrace{\frac{1}{\tau_{I,J}} \left(2 - ntu^h_{I,J}\right) T_{h,i,J}}_{Product1} - \underbrace{\frac{1}{\tau_{I,J}} \left(2 + ntu^h_{I,J}\right) T_{h,i+1,J}}_{Product2} \\ &\quad + \underbrace{\frac{2ntu^h_{I,J}}{\tau_{I,J}} T_{p,I,J}}_{Product} \end{aligned} \tag{2.9}$$

With this rearrangement, and integrating rather than differentiating, it is easy to simulate the hot fluid in SIMULINK, and it is shown schematically in Figure 2.4. Equations (2.2) and (2.3) can be similarly manipulated to yield the SIMULINK schematics given in Figures 2.5 and 2.6, respectively.

Together these three modules define a single cell in a cross-flow heat exchanger, and its SIMULINK schematic is depicted in Figure 2.7. The hot fluid module is at the top; the separating plate module is in the middle; and the cold fluid module is at the bottom.

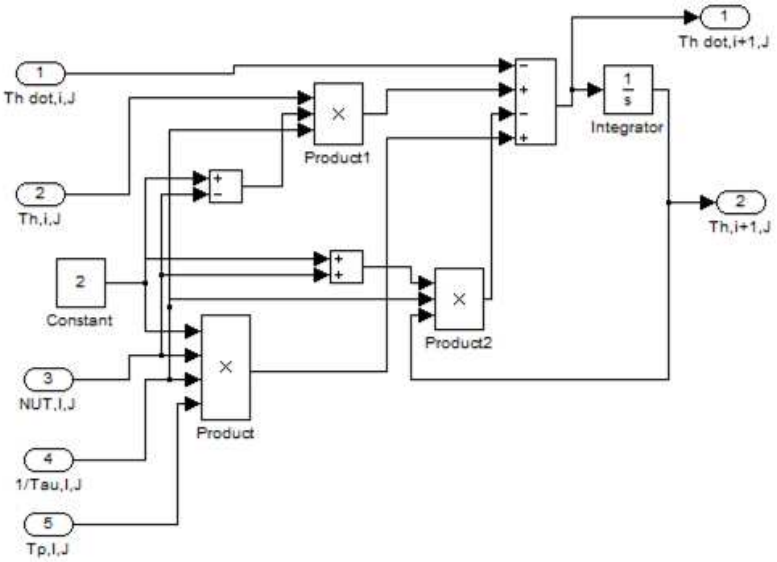

Figure 2.4. SIMULINK module for the hot fluid for Eq. (2.1) (one cell).

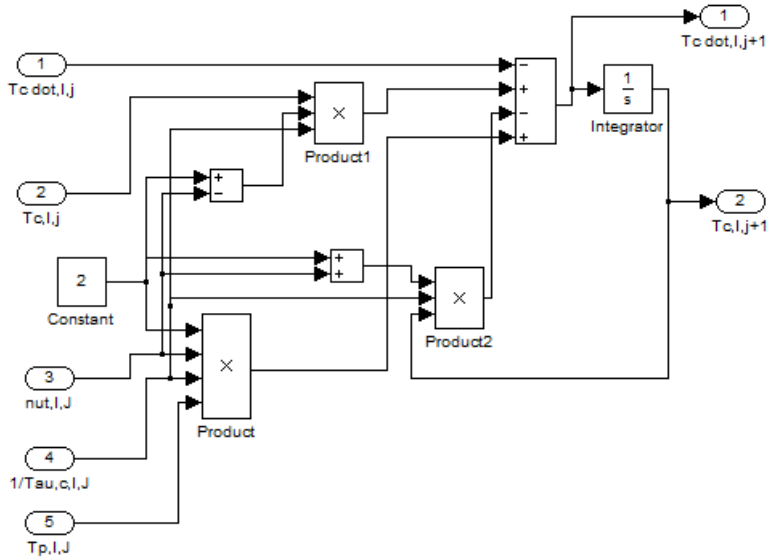

Figure 2.5. SIMULINK module for the cold fluid for Eq. (2.2) (one cell).

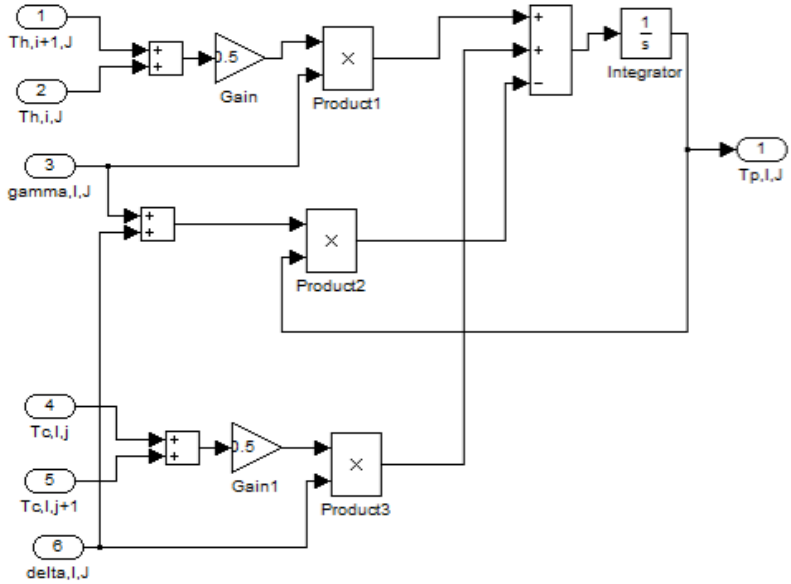

Figure 2.6. SIMULINK module for the separating plate for Eq. (2.3) (one cell).

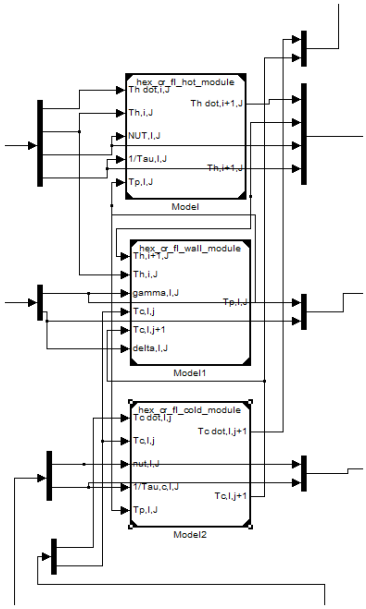

Figure 2.7. A single cell of a CFHE using SIMULINK.

2.2 Determination of cell numbers

To compute an accurate enough solution one has to determine how many cells are needed to divide the heat exchanger into. For this an electrical heater model is used, for which an analytical solution exists. This is appropriate because the electrical heater has to solve the same differential equations as for the cross-flow heat exchanger, except the boundary conditions are slightly different.

The electrical heater's geometry is a thin plate heated by Joule effect, so that a constant heat flux is generated all along the heater. The temperature of the plate is assumed to be homogeneous in the direction perpendicular to the flow. For this case, the exact solution in the Laplace space for the transfer function is, (Ingimundardóttir and Lalot, 2009),

$$\mathcal{L}\{T_h\} = \frac{\frac{ntu}{\tau}}{s\left(s+\frac{ntu}{\tau}+\gamma\right)} \times \left\{1 - \exp\left(-\tau s\right)\exp\left(-ntu\frac{s}{s+\gamma}\right)\right\} \tag{2.10}$$

where $ntu = \frac{hA}{\dot{m}_h c_h}$, $\tau = \frac{M_h}{\dot{m}_h}$, and $\gamma = \frac{hA}{M_p c_p}$.

When applying a Heaviside function for the heat flux, the inverse Laplace solution is given by:

$$\begin{aligned} T_h \;=\; & \frac{\frac{ntu}{\tau}}{\frac{ntu}{\tau}+\gamma} \times \left[1 - \exp\left(-\left(\frac{ntu}{\tau}+\gamma\right)t\right)\right] - \frac{ntu}{\tau}\exp\left(-ntu\right) \times \\ & \left\{\left(\exp\left(-\left(\frac{ntu}{\tau}+\gamma\right)t\right) \times \left[\frac{\frac{ntu}{\tau}}{\frac{ntu}{\tau}+\gamma}\right] + \frac{\gamma}{\frac{ntu}{\tau}+\gamma}\right)\right. \\ & \left. * \left(\exp(-\gamma t) \times I_0\left(2\sqrt{ntu\,\gamma t}\right)\right)\right\}(t-\tau) \end{aligned} \tag{2.11}$$

where I_0 is the modified Bessel function of the first kind and of order 0.

Comparing this analytical solution for electrical heater to the solution obtained

with SIMULINK's CFHE approximation one finds that 20 cells leads to a very accurate solution (Figure 2.8).

Conversely, studying a heat exchanger with constant plate temperature leads to the same conclusion: 20 cells are sufficient in the mass flow direction.

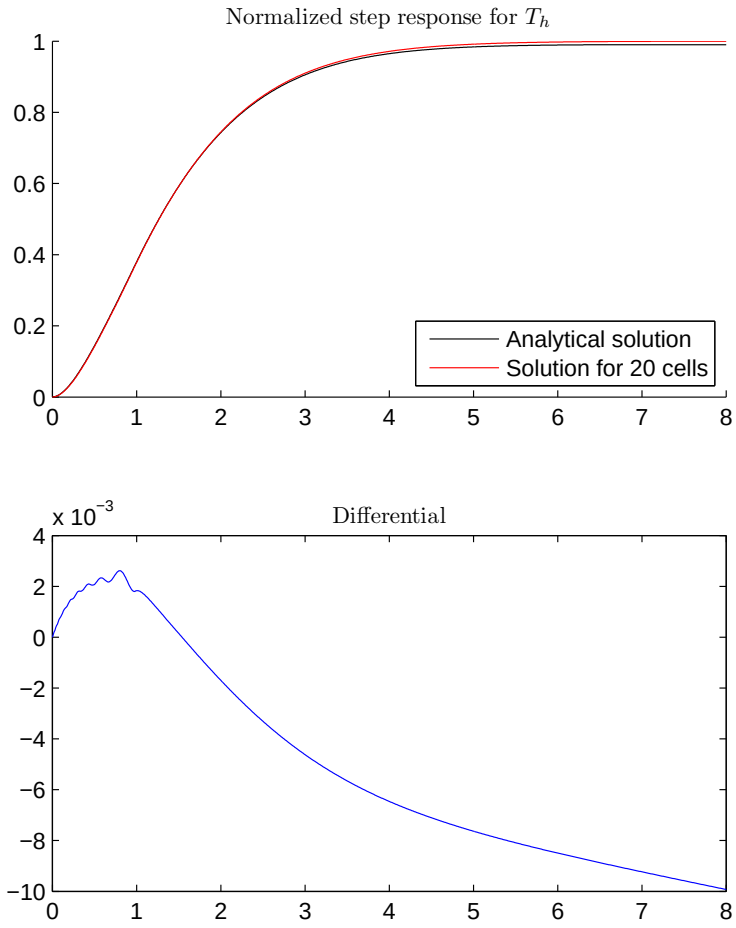

Figure 2.8. Comparison of the results obtained using the approximated model and analytical results (transient states).

To make sure the approximated SIMULINK model is correct, the model should reach a steady state regardless to the temperature level mass flow rates. This is done by comparing the effectiveness (defined by Equation (3.1)) for a large range of number of transfer units, ntu, and a large range of heat capacity rate ratios, C, for both the analytical solution and the approximated SIMULINK's model using 20×20 cell groups (or 1200 blocks). Figure 2.9 shows effectiveness as a function of the number of transfer units, and the approximated SIMULINK model comes to the same conclusion as the analytical results for steady states.

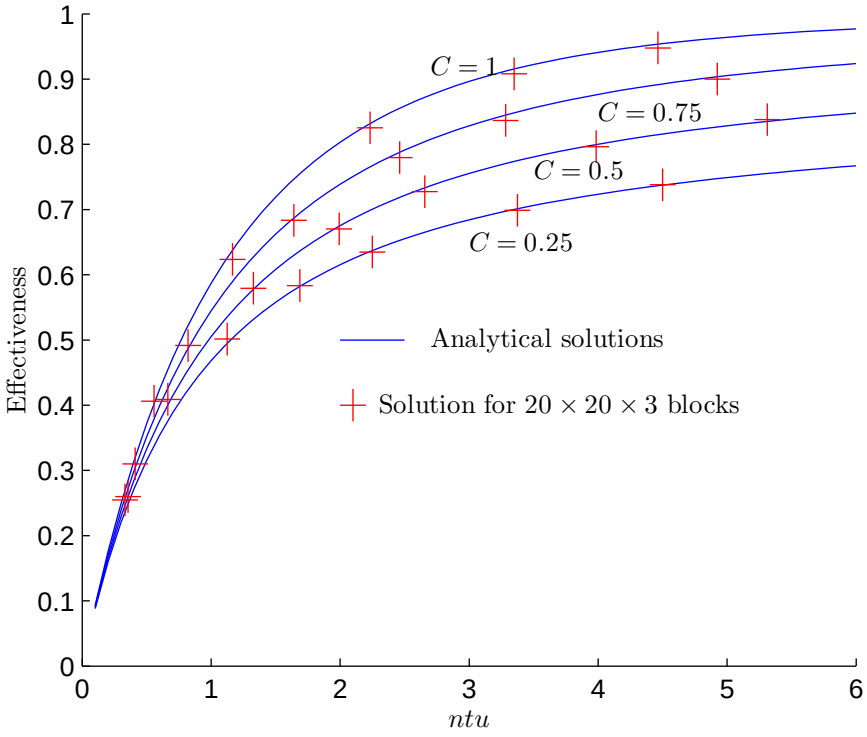

Figure 2.9. Comparison of the results obtained using the approximated model and analytical results (steady states).

2.3 Data

To be able to evaluate the efficiency of the methods proposed in chapter 3 it has been chosen to work with simulated data. In this case, it is possible to introduce an arbitrary time variation of the fouling factor, i.e. thermal resistance due to fouling, R_f; and consequently to know what this factor is when fouling is detected.

Generated are 200 sets of data, each time series going from time $t = 0$ to time $t = 1$ (dimensionless time). Each data set consists of six time series. Four of them being the inputs, $\{T^h_{in}, T^c_{in}, \dot{m}_h, \dot{m}_c\}$, namely the inlet temperatures and mass flow rates for the hot and cold fluids, and the second two being the simulated outputs, $\{T^h_{out}, T^c_{out}\}$, the outlet temperatures for the hot and cold fluids. First hundred data sets are of a clean heat exchanger, and the latter hundred sets are of a heat exchanger where fouling occurs. The clean and fouled data sets share the same inputs, except the output differs. For the fouled data sets a continuous fouling factor is applied to the heat exchanger's hot side and it is increased as time goes by with exponential growth. This is in accordance with Paul Watkinsons's findings on crude oil (Watkinson and Li, 2009). Figure 2.10 shows the evolution of the fouling factor.

For heat transfer the overall heat transfer coefficient U is generally studied. Its formula is given by Equation (2.12).

$$U_{\text{clean}} = \frac{1}{\frac{1}{A_c h_c} + \frac{1}{A_h h_h}} \Big/ A \tag{2.12}$$

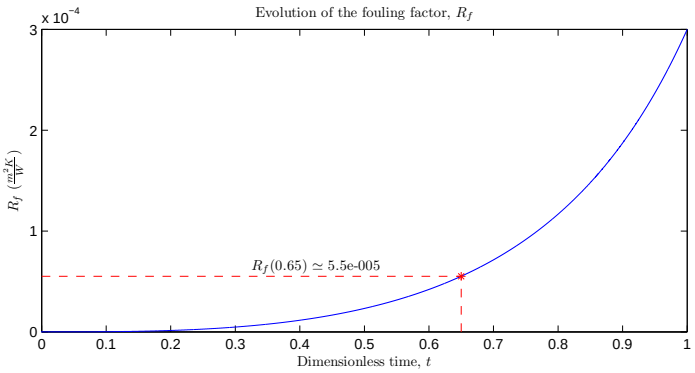

Figure 2.10. Evolution of the fouling factor R_f.

where

$$h_i = \frac{Nu_i k_i}{Dh_i} \tag{2.13}$$

$$Nu_i = 0.023 Re_i^{\frac{4}{5}} Pr_i^{\frac{1}{3}} \tag{2.14}$$

$$Pr_i = \frac{c_i \nu_i \rho_i}{k_i} \tag{2.15}$$

$$Re_i = \frac{V_i Dh_i}{\nu_i} \tag{2.16}$$

and $i = h, c$ denotes the hot and cold side of the heat exchanger, respectively.

When fouling accumulates in a heat exchanger surface coatings builds on the heat transfer surfaces during its operations. This adds an extra thermal resistance to the wall and may noticeably decrease the overall heat transfer coefficient and thus performance. The relationship for thermal resistance due to fouling deposits is usually as follows:

$$U_{\text{fouled}} = \frac{1}{\frac{1}{A_c h_c} + \frac{R_f}{A_c} + \frac{1}{A_h h_h}} \Big/ A \tag{2.17}$$

Input	min	max
Mass flow rates for hot and cold fluids:	0.6 $\frac{kg}{s}$	1.2 $\frac{kg}{s}$
Inlet temperatures for the cold fluid:	16 ℃	24 ℃
Inlet temperatures for the hot fluid:	56 ℃	64 ℃

Table 2.1. Ranges of the inputs

Since U is highly sensitive to the mass flow rates through the Reynolds numbers, Eq. (2.16), which in this study are not time invariant, it is impossible to observe U directly, hence it must be observed indirectly.

The ranges of the inputs are given in Table 2.1. Although in practise, the variation ranges would be much smaller, this is done to give a better generalization of the methods' abilities to detect fouling. The input values are randomly varying throughout the time series, as can be seen in Figure 2.11. To prevent too fast fluctuations in the input, a random permutation was implemented on several replicates of equally distributed values in the range of the inputs, that were then interpolated, yielding an input of 10,000 samples. The corresponding outlet temperatures are shown in Figure 2.13. Note that for a clean set, the corresponding fouling set uses exactly the same inputs, but of course the outputs vary.

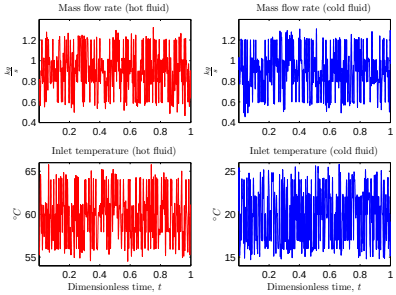

Figure 2.11. Inputs – inlet temperatures and mass flow rates.

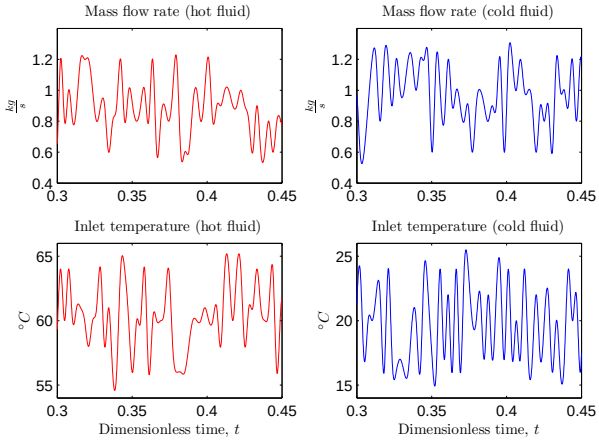

Figure 2.12. Partial view of the inputs.

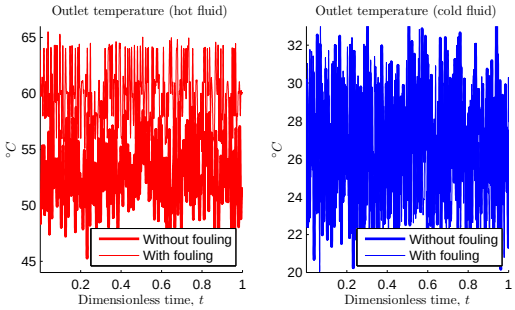

Figure 2.13. Outputs – outlet temperatures.

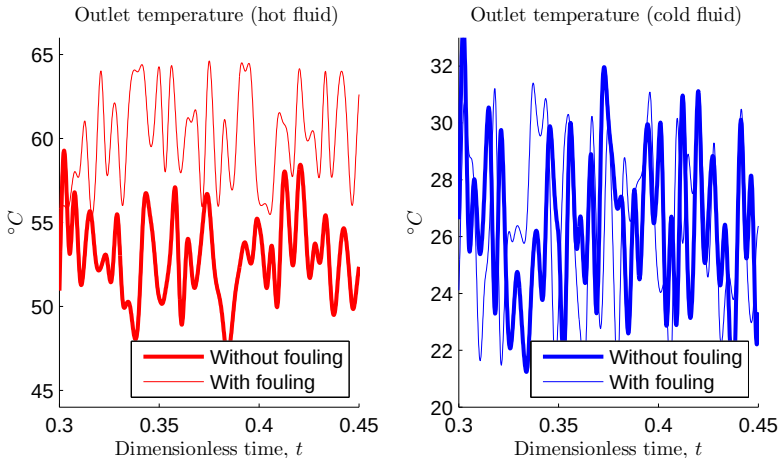

Figure 2.14. Partial view of the outputs.

2.4 Summary

In this chapter a cross-flow heat exchanger was introduced, and how it could be modelled using the laws of thermodynamics and subsequently simulated with SIMULINK. For each random input time series there were two time series simulated: one of a clean CFHE and one of a fouled CFHE, where the difference is due to a continuous exponentially growing fouling factor added to the fouled CFHE. All in all, there were 200 clean and 200 fouled data sets generated. Each data set was presented in dimensionless time, going from time $t_{start} = 0$ to time $t_{finish} = 1$. The each data set had 10,000 time steps.

Chapter 3

Effectiveness Ratio Method

On-line monitoring of fouling for *steady state* systems can be based on the evolution analysis of the thermal effectiveness of the heat exchanger. Where the effectiveness is defined in relation to the temperatures of the inlets and outlets of the hot and cold sides of the heat exchanger at time t:

$$E(t) = \frac{T^h_{in}(t) - T^h_{out}(t)}{T^h_{in}(t) - T^c_{in}(t)}. \tag{3.1}$$

In the best case scenario there would be a complete heat transfer if $T^h_{out} = T^c_{in}$ and $T^c_{out} = T^h_{in}$. When fouling is present the heat exchanger loses its efficiency and the T^h_{out} increases, so once the effectiveness starts to drop it shows an indication of presence of fouling. Equation (3.1) also takes into consideration the random deviations that occur between the inlet hot and cold temperatures.

For the constantly varying data generated in chapter 2 analysis has to be carried out. As Figure 3.1 shows, or Figure 3.2 shows more clearly, it is impossible to detect fouling using the raw evolution of the *instantaneous effectiveness* defined by Eq. (3.1). For the equation to be valid the time series needs to attain

a steady state, but with the inputs always simultaneously varying that is never possible.

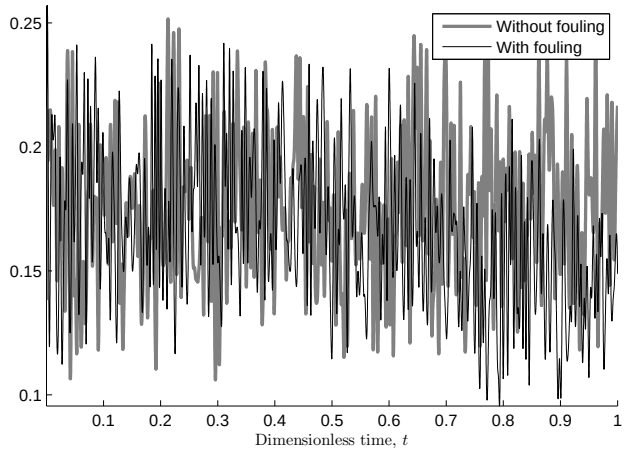

Figure 3.1. Evolution of the instantaneous effectiveness.

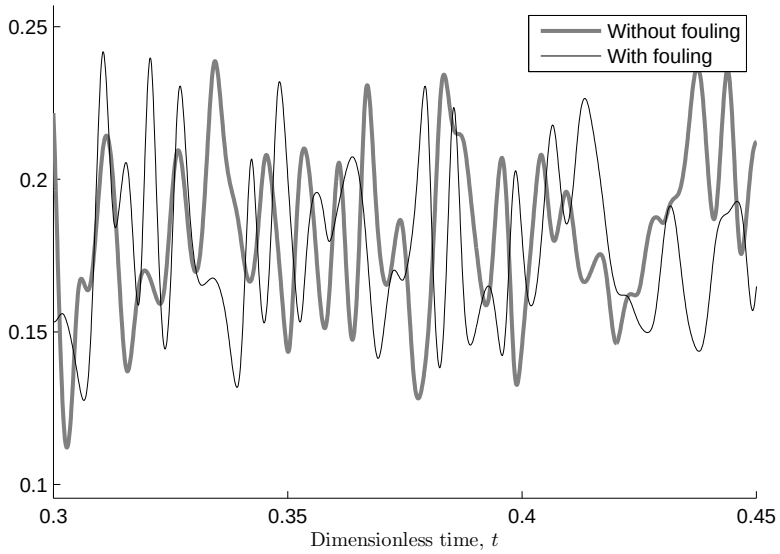

Figure 3.2. Partial view of the evolution of the instantaneous effectiveness.

To detect fouling, a *new* definition has been formed for the effectiveness, called the *effectiveness ratio method* (ERM) The method is quite generic for detection of fouling since it is not dependent of the model of the heat exchanger. Therefore, it can be utilised on different types of data than for the CFHE proposed in chapter 2.

The method skips the first $t = 0.1$ in dimensionless time of the samples in order to show that the analysis can be applied to an ongoing process. The method is dependent on the choice of its step size, or offset θ, in dimensionless time. The following θ samples make up the *expanding observation window* (see Figure 3.3), where an approximation of the effectiveness is computed. This is done by applying the methods discussed in the following subsections on the instantaneous effectiveness. For each observation window, the mean of the approximated effectiveness is used as a reference value. If the ratio between the approximated effectiveness and its mean falls below the predefined theshold, ζ, fouling is detected, otherwise the heat exchanger is assumed to be still clean. The algorithm for ERM is given in Table 3.1.

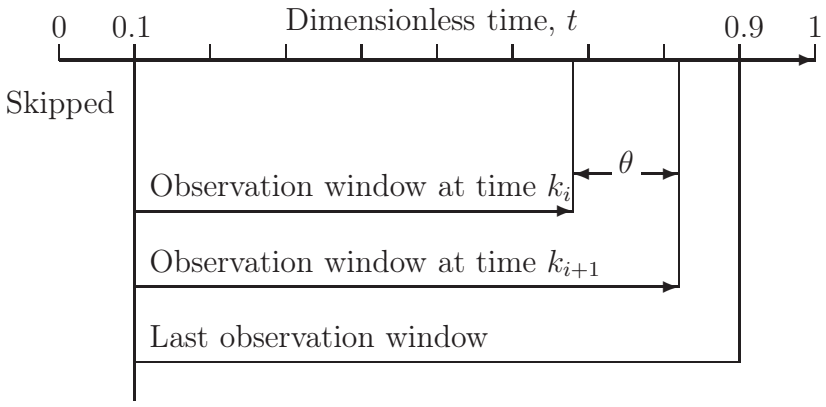

Figure 3.3. Expanding observation window scheme.

Initial step	Skip the first 10% of the samples.
Step 1	Expand the observation window by an offset of length θ.
Step 2	Compute the instantaneous effectiveness of this window.
Step 3	Approximate the effectiveness with a method introduced in the subsequent sections.
Step 4	Compute the effectiveness ratio for the window with respect to its average.
Step 5	If the ratio falls below the threshold, ζ, fouling is detected. Else return to step 1.

Table 3.1. Effectiveness ratio method algorithm

3.1 Threshold for ERM

The detection of fouling with ERM is fundamentally up to the choice of its threshold, ζ. For this the ratios, r, between the approximated effectiveness and its mean for the clean training sets, J, are used. Their median minimum value is chosen as the threshold for the fouled data sets, i.e.

$$\zeta = \text{median}\{\min_{j \in J} r(j)\} \tag{3.2}$$

Using the median minimum value of the ratios allows some false positive detection of fouling for clean CFHE, but this is done to reduce the likelihood of false negative detection of fouling for fouled CFHE.

3.2 Moving average

A common technique for smoothing out short-term fluctuations to focus on long-term trends in a time series is that of a *moving average*. A simple moving average with a sliding window of length ω, calculates the average of the previous ω points. For the instantaneous effectiveness defined in Eq. (3.1) its moving average is a sequence of values

$$\tilde{E}_t = \frac{1}{\omega} \sum_{i=t-\omega+1}^{t} E_i, \quad t = \omega, \omega+1, ... \tag{3.3}$$

For ease of computation this can be simplified to

$$\tilde{E}_t = \tilde{E}_{t-1} - \frac{E_{t-\omega}}{\omega} + \frac{E_t}{\omega}, \quad t = \omega+1, ... \tag{3.4}$$

3.3 Wavelet transform

When analysing non-stationary signals it is necessary to take careful consideration to the time and frequency domains and what compromises should be made between the two. Standard Fourier transform is only localised in frequency; the short-time Fourier transform is limited by its fixed window length. On the contrary, wavelets are localised both in time and in frequency; but it is possible to control the localisation, depending on where the emphasis needs to lie for a good estimate. In fact, wavelets separate data into frequency components and analyses each component with a resolution matched to its scale. The more the wavelet is similar to the signal components the larger is the corresponding wavelet coefficient.

Wavelet transform can be beneficial for feature extraction, e.g. fingerprints

recognition (J.A. Montoya Zegarra and da Silva Torres, 2009), or even for diagnosis, e.g. (C.K. Sung and Chen, 2000), (V. Belotti and Rossi, 2006) and (Saravanan and Ramachandran, 2009).

Wavelets are functions that satisfy certain requirements, e.g. they should integrate to zero, waving above and below the x-axis; be well localised; and other requirements are technical to insure quick and easy calculations to the direct and inverse wavelet transform.

Wavelets are structured basis in discrete or continuous time, and they permit different time versus frequency resolution trade-offs.

For a short introduction to wavelet transform one can refer to (Vidakovic and Mueller, 1994), or more extensively in (Vetterli and Kovaĉević, 1995) and (Pereyra and Mohlenkampy, 2004).

MATLAB code for the wavelet transform is given in Appendix B.

3.3.1 Orthonormal wavelet basis

For wavelet transforms an orthonormal bases has the form

$$\psi_{j,k}(x) = \{2^{j/2}\psi(2^j x - k) \ : \ j,k \in \mathbb{Z}\} \tag{3.5}$$

so each element of the basis is a translated and dilated version of a single wavelet. Usually $\psi_{j,k}$ are called daughter wavelets of the mother wavelet ψ.

Signal $f(t)$ can be expressed in terms of dilated and translated wavelets up to a scale j:

$$f(t) = \sum_k \lambda_j(k)\phi(2^j t - k) \tag{3.6}$$

If one would like to step to scale $j-1$, one has to add wavelets in order to keep the same level of detail, $f(t)$ is expressed as:

$$f(t) = \sum_k \lambda_{j-1}(k)\phi(2^{j-1}t - k) + \sum_k \gamma_{j-1}(k)\psi(2^{j-1}t - k) \tag{3.7}$$

Where

$$\lambda_{j-1}(k) = \langle f(t), \phi_{j,k}(t)\rangle \qquad \text{low-pass filter}$$
$$\gamma_{j-1}(k) = \langle f(t), \psi_{j,k}(t)\rangle \qquad \text{high-pass filter}$$

and ψ is the inverse transform of the wavelet ϕ. The low-pass filter is thought as the *averaging* filter whilst the high-pass filter produces the *detail*.

There is an important property for the wavelet series basis function called the *two-scale equation property*. The two-scaling equation for the scaling function is given by:

$$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} \lambda(n)\phi(2t - n) \tag{3.8}$$

that is to say that it is a series expansion of itself. The two-scale equation for the wavelet function is given by:

$$\psi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} \gamma(n)\phi(2t - n) \tag{3.9}$$

Thus the two-scaled relation states that the scaling function of a certain scale can be expressed in terms of translated scaling function at the next smaller scale.

Note that greater scale means more detail. Therefore, one can successively approximate a function starting from a coarse version going to a fine resolution version.

For this study concentrated was on on the Daubechies wavelet basis. It is compactly supported and can be designed with as much smoothness as desired. It relies on the iteration of the discrete filter bank that converges to a continuous time wavelet basis. Daubechies wavelets are designed so that they have the minimum length of support for a given number of vanishing points. Note that the shorter the support the fewer wavelets interact with a given singularity.

3.3.2 Filter bank tree

Once a basis has been decided, there exists an equivalent filter that can be worked with instead. Therefore, for the instantaneous effectiveness defined in Eq. (3.1) its wavelet transform $f(t) := E[n]$ can be implemented in the following manner:

A signal E is calculated by passing it through a series of filters: First the samples are passed through a low-pass analysis filter with impulse response λ resulting in a convolution of the two. The signal is also decomposed simultaneously using a high-pass filter γ. The output gives the detail coefficients (from the high-pass filter) and approximation coefficients (from the low-pass filter). Since half of the frequencies of the signal have now been removed, half of the samples can be disregarded according to Nyquist's rule, leading to the filter outputs being down sampled by 2. This decomposition has halved the time resolution since only half of the filter output characterise the signal. This

decomposition is repeated to further increase the frequency resolution and the approximation coefficients decomposed with high- and low-pass analysis filters and down sampling by 2. This is represented as a binary tree, known as a *filter-bank*, with nodes representing a subspace with different time-frequency localisation, shown in Figure 3.4. This process is repeated in accordance of the scale chosen. This results in the wavelet coefficients, consisting of the wavelet approximation coefficients and the detail coefficients. Now, to recover the original signal the filter outputs needs to be up sampled by 2 and passed through the synthesis filters as often as the scale dictates.

For this study only the wavelet transform is being used to get rid of the detail, i.e. the noise, of the effectiveness, thus only the low-pass filters are of interest so the high-pass filter do not need to be computed.

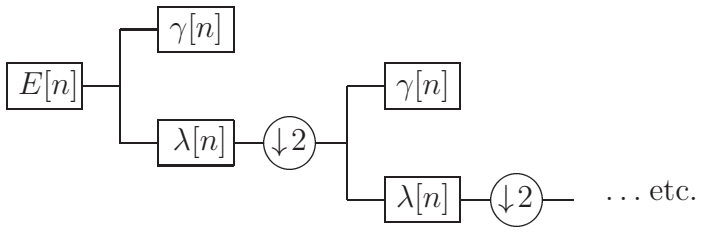

Figure 3.4. Filter bank tree with for signal E using analysis filters γ and λ.

3.4 Optimisation

For this optimisation problem there are multiple objectives: minimising the times of detection and maximising the correct classification for clean and fouled data. The decision variables are the offset of the effectiveness ratio method and the parameters of its sub method. To achieve this a multi objective genetic

algorithm (MOGA) implemented in C++ for MATLAB is used, (Sastry, 2007).

3.4.1 Multi-Objective Optimisation

The multi-objective optimisation problem (MOOP) for this study has three objective functions: the ratio of correctly classified fouled CFHE, f_1; the ratio of correctly classified clean heat CFHE, f_2; and the mean time of detection for fouled CFHE, f_3. The first two need to be maximised, and the thrid one needs to be minimised. The objective functions are dependant to some constraints of the decision variables, its mathematicall form is given as

$$\begin{aligned} \text{Maximise} \quad & f_j(\vec{x}) && j = 1, 2 \\ \text{Minimise} \quad & f_3(\vec{x}) && \\ \text{subject to} \quad & x_i^{(L)} \leq x_i \leq x_i^{(U)} && i = 1, 2, ..., n \end{aligned} \tag{3.10}$$

where a solution $\vec{x} = (x_1, x_2, ..., x_n)$ are the n decision variables; f_j are the objective functions; $x_i^{(L)}$ and $x_i^{(U)}$ are the lower and upper bounds for decision variable x_i, respectively. ERM always has the offset as its first decision variable, i.e. $x_1 = \theta$. Other decision variables depend on the modelling technique. For the wavelet transform they are the dimension of the basis, the scale of the transform, and the boundary condition coefficient, i.e. $x_2 = \kappa, x_3 = \xi, x_4 = \varrho$. For the moving average the additional decision variable is the length of the sliding window, i.e. $x_2 = \omega$.

Generally some of the objective functions are conflicting, thus there cannot be a single optimum solution that simultaneously optimises all of the objectives. Assume $\vec{f} : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^M$ is to be maximised in all its coordinates, then a solution $x \in \mathcal{D}$ dominates $z \in \mathcal{D}$ if each coordinate of $\vec{f}(\vec{x})$ is no greater than the corresponding coordinate of $\vec{f}(\vec{z})$, and at least one coordinate of $\vec{f}(\vec{x})$ is

strictly less than the corresponding coordinate of $\vec{f}(\vec{z})$. A set of dominated solutions are solutions that are not dominated by any other solution in the feasible region $\mathcal{D}$. None of these solutions can be chosen to be better than the other with respect to all of the objectives. This set of dominated solutions are called a Pareto set, or a Pareto frontier, (Deb, 2003).

Thus the goal in multi-objective optimisation is to find a set of solutions that are as close as possible to the Pareto frontier; and are as diverse as possible to be able to achieve global optimum or at least a very good local optimum. Depending on the importance of each objective function a single optimum solution is chosen from the Pareto set.

3.4.2 Optimisation using a Genetic Algorithm

A genetic algorithm (GA) is an evolution strategy that uses same fundamental principals that of Charles Darwin's natural selection in biology. Natural selection describes that genes that give an organism a higher chance of survival and reproduction rate are more common in a population over successive generations.

From an optimisation stand point each solution, or *individual*, $\vec{x} \in \mathcal{D} \subset \mathbb{R}^n$ for the MOOP described in Eq. (3.10) is characterised by its genotype that consists of n genes, which determine its vitality, i.e. its fitness. In GA an individual's genotype is represented as a binary bit string, so each objective parameter needs to be encoded accordingly.

For each iteration, or *generation*, of the optimisation process, the set of individuals used are referred to as its *population* of size λ.

Initial step	Random population for the first generation, chosen uniformly over the feasible region.
Step 1	Selection μ pairs of parents are chosen with probabilities proportional to their rank (via Pareto ranking) within the current population.
	Reproduction
Step 2	Recombination From each parental pair, given a recombination probability, p_c, there is a crossover of the two parental genotypes. There can be n_c number of crossover points per individual. See Figure 3.5. Of the two recombined offspring, only one is used. Repeated until there are λ new individuals for the following generation.
Step 3	Mutation For a given probability, p_m, a point mutation on an individual's gene occurs, independent to other individuals, by changing either a zero to one or vice versa.
Step 4	Return back to step 1.

Table 3.2. Multi objective genetic algorithm (MOGA) algorithm.

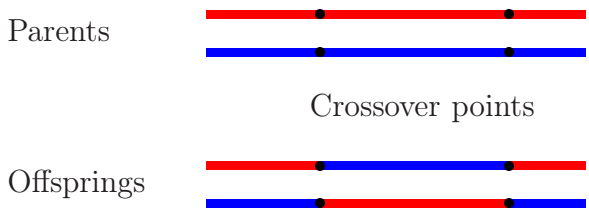

Figure 3.5. Recombination of a parental pair

After the initialisation process, the GA algorithm, given in Table 3.2, is a circular process of selection and reproduction, that consist of recombination and mutation, which literally yields the survival of the *fittest*.

According to (Schwefel, 1995) the canonical values for the recombination probability are $p_c = 0.6$; for the number of crossover points $n_c = 2$; and for the mutation probability $p_m = 0.001$.

3.5 Summary

On-line monitoring of fouling is generally done by examining the evolution of the effectiveness. This is only valid if the system reaches steady state. When the inputs are constantly simultaneously varying the system is always in transient state, thus an approximation of the instantaneous effectiveness is needed. The approximation schemes presented in this chapter were moving average and wavelet transform. The mean of the approximated effectiveness is then used as a reference. If the ratio between the approximation and its mean exceeded a certain threshold, it is implied that fouling has been detected. This method is referred to as the effectiveness ratio method. For the optimum

selection of the parameters in the approximation, a multiple objective genetic algorithm is applied to a training set consisting of clean and fouled data sets, with its goal being both correct detection of fouling for both clean and fouled CFHE and early detection time if CFHE is fouled.

Chapter 4

Results

In this chapter the main results for ERM will be given, using a MOGA for the optimal parameter selection. ERM will have both moving average and wavelet transform as its sub method. Published results for the Kalman filter are also presented.

4.1 Effectiveness ratio method

The effectiveness ratio method is dependent on its sub method, either moving average or wavelet transform. Both types have parameters that need to be optimised, for that reason the clean and corresponding fouled data sets are split into two equal parts, half for the *training* set and the latter half for the *test* set. Supervised learning takes place on the training set, yielding the optimal parameter estimate for the methods that can then be used on the test set, that has not yet been presented to the method.

			Training set			Test set			
	θ	ω	f_1	f_2	f_3	f_1	f_2	f_3	ζ
■	0.0054	0.070	0.90	0.77	0.769	1.00	0.82	0.762	1.01
♦	0.0312	0.081	1.00	0.94	0.783	1.00	0.84	0.776	1.03
▲	0.0052	0.082	1.00	0.84	0.765	1.00	0.84	0.759	1.03
▼	0.0163	0.073	1.00	0.92	0.780	1.00	0.86	0.776	1.02

Table 4.1. Solutions on the Pareto front for MA.

The parameters of ERM that need to be optimised are the offset, θ, and the parameters of its sub method. As discussed in (Ingimundardóttir and Lalot, 2009) the offset was chosen to be $t = 0.02$ since the fouling is relatively slow, so the range of θ in MOGA had a lower bound of $t = 0.01$ and an upper bound of $t = 0.04$.

4.1.1 Moving average

There is only one parameter involved with the moving average, the length of the sliding window, ω, that needs to be optimised. Its range was chosen to have a lower bound of $t = 0.01$ and an upper bound of $t = 0.1$.

The results from MOGA are given in Figure 4.1. The surface is the interpolation of the MOGA's evaluated population and its Pareto front is given in Table 4.1 along with a unique colour and character coding, and for clarity are moved slightly from the surface.

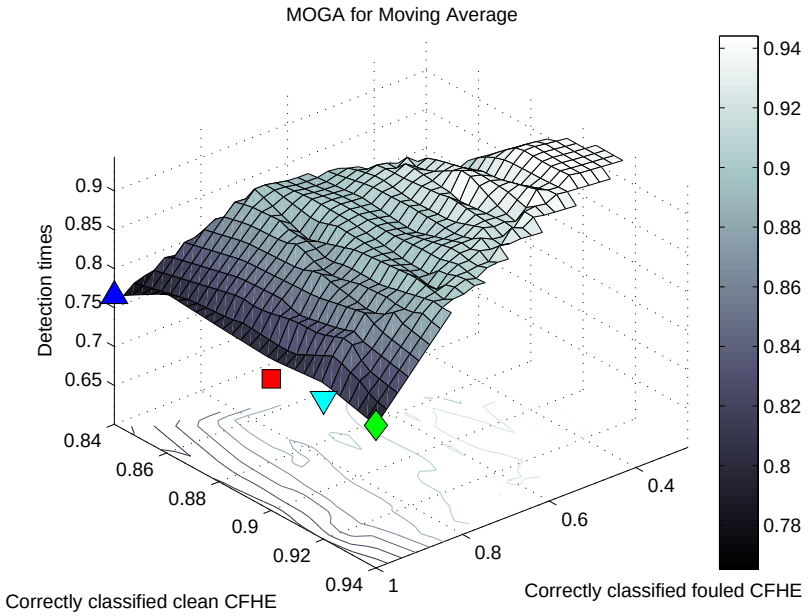

Figure 4.1. MOGA's Pareto front for MA.

4.1.2 Wavelet transform

For the Daubechies wavelet transform there are three parameters that need to be chosen, namely the dimension of the Daubechies basis, κ; the number of iterations for the approximation scheme, i.e. its scale, ξ; and the boundary coefficient, ϱ.

For the paper (Ingimundardóttir and Lalot, 2009) a low dimension of WT was preferred so the approximation would be similar to that of a step-function and consequently ERM too, which would be more easily interpreted for when fouling occurs. Lower dimension would also imply that singularities would be less as fewer wavelets were to interact, which is advantageous since the data is

quite noisy. For this reason the range of κ in MOGA had an upper bound of a dimension lower than 20, as well as being an even integer, so the basis would be well defined.

There were some boundary effects on the wavelet transform, and it was thought better to use real entries instead of adding pseudo entries, e.g. adding mirrored entries from both ends of the time series before approximating and that would would be immediately deleted afterwards, since the data is not periodic. Thus there are only the first ϱ entries considered in the approximation to be certain that the boundary effects did not effect the results. To make sure not too much information would be lost, the range of ϱ had a lower bound of 80% and an upper bound of 100% of the time series.

The results from MOGA are given in Fig. 4.2. The surface is the interpolation of the MOGA's evaluated population and its Pareto front is denoted by red asterisks. Selected solutions on the Pareto front are given in Table 4.2. Each solution is given a unique colour and character coding, and for clarity are moved slightly from the surface, so they can be easily identified in Fig. 4.3.

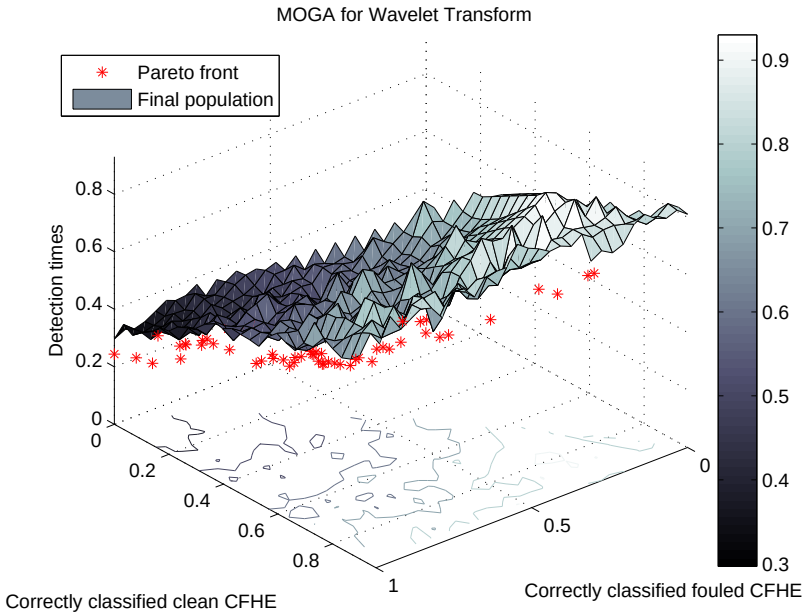

Figure 4.2. MOGA's evaluated solutions for WT.

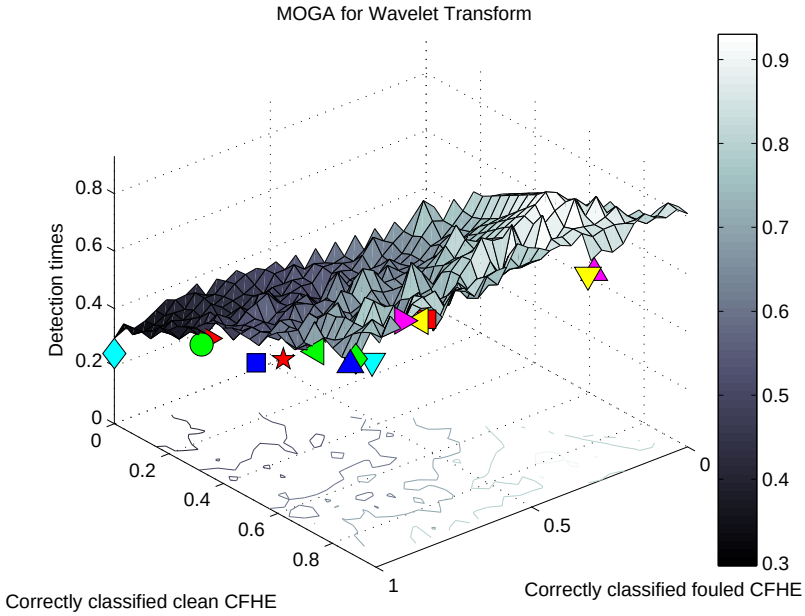

Figure 4.3. Selected solutions on the MOGA's Pareto front for WT.

					Training set			Test set			
	θ	κ	ξ	ϱ	f_1	f_2	f_3	f_1	f_2	f_3	ζ
■	0.0267	4	9	0.887	0.82	0.94	0.776	0.76	0.90	0.752	0.96
♦	0.0047	2	9	0.845	0.96	0.84	0.645	0.98	0.80	0.606	0.95
▲	0.0203	2	9	0.859	0.96	0.82	0.614	0.98	0.70	0.553	0.96
▼	0.0204	2	9	0.850	0.96	0.90	0.669	0.98	0.84	0.621	0.96
►	0.0015	2	8	0.916	0.88	0.92	0.786	0.90	0.96	0.776	0.92
◄	0.0310	4	9	0.889	0.84	0.94	0.779	0.78	0.92	0.729	0.95
★	0.0329	10	10	0.806	1.00	0.62	0.546	0.98	0.58	0.546	0.98
●	0.0135	14	10	0.989	1.00	0.32	0.442	1.00	0.20	0.445	0.59
■	0.0329	20	9	0.916	1.00	0.52	0.482	1.00	0.50	0.485	0.94
♦	0.0073	4	10	0.816	1.00	0.00	0.244	1.00	0.00	0.244	0.99
▲	0.0377	14	8	0.830	0.30	0.96	0.731	0.18	0.90	0.785	0.91
▼	0.0377	14	8	0.832	0.32	0.96	0.731	0.18	0.90	0.786	0.91
►	0.0179	10	10	0.994	1.00	0.34	0.473	1.00	0.28	0.499	0.55
◄	0.0126	8	9	0.891	1.00	0.74	0.636	1.00	0.64	0.593	0.95

Table 4.2. Selected solutions on the Pareto front for WT.

4.2 Kalman filter

For comparison of the new ERM to an established method of detection of fouling, this study focuses on detection of fouling using extended Kalman filters (EKF), which is a state space method.

Detection of fouling using EKF has been studied extensively at the University of Iceland. Oddgeir Guðmundsson, Halldór Pálsson and Ólafur Pétur Pálsson have published numerous papers on the subject. Their latest paper (O. Guðmundsson and Pálsson, 2009) was on a on-line detection of fouling using EKF for a CFHE using the exact same data as this study. See Appendix A for a short introduction to the Kalman filter and extended Kalman filter.

To use the EKF to detect fouling for CFHE, one needs to write Eq. (2.1), (2.2) and (2.3) in a state form, see (S. Lalot and Desmet, 2007) and (Guðmundsson, 2008).

$$\frac{d}{dt}\hat{T} = f(\hat{x}(t)). \tag{4.1}$$

In all of these studies, the parameters of interest are

$$\eta(t) = \frac{FA_hU_{ij}(t)}{\dot{m}_h(t)c_h} \tag{4.2}$$

$$\nu(t) = \frac{FA_cU_{ij}(t)}{\dot{m}_c(t)c_c} \tag{4.3}$$

where F is a correction factor. By observing the parametrisation $\theta = \{\eta, \nu\}$ one can observe U indirectly via Eq. (4.2), e.g. for U to decrease so must the parameters η and ν. To detect the shift in the parameters the Cumulative sum control chart is used. Their results were a mean detection of fouling at time $t = 0.83$ with a 95% confidence interval of $t \in [0.59, 0.98]$.

4.3 Summary

Optimal parameters for ERM are chosen using a MOGA optimisation. The bounds for the decision variables were chosen in accordance with the assumptions of the underlying data. The optimisation was done for both a moving average and a wavelet transform as the ERM's sub method. The resulting three dimensional Pareto front were plotted in Figures 4.1 and 4.2, and selected solutions listed in a Tables 4.1 and 4.2.

Chapter 5

Discussions and conclusion

MOGA gives a variety of optimal solutions, the Pareto front, for ERM since all of its objective functions are considered equally important. The selection of optimal parameters for ERM is therefore entirely up to the demands of the heat exchanger's operator, depending on where his emphasis lies, whether it is a lower detection time of fouling on average; or lower chance of false alarm on detection of fouling.

For instance, if there were the restriction of at least correctly classifying 80% of the fouled and clean CFHE, the evaluated solutions are those depicted in Figure 5.1, and its Pareto front is denoted using the same coding as before in Table 4.2.

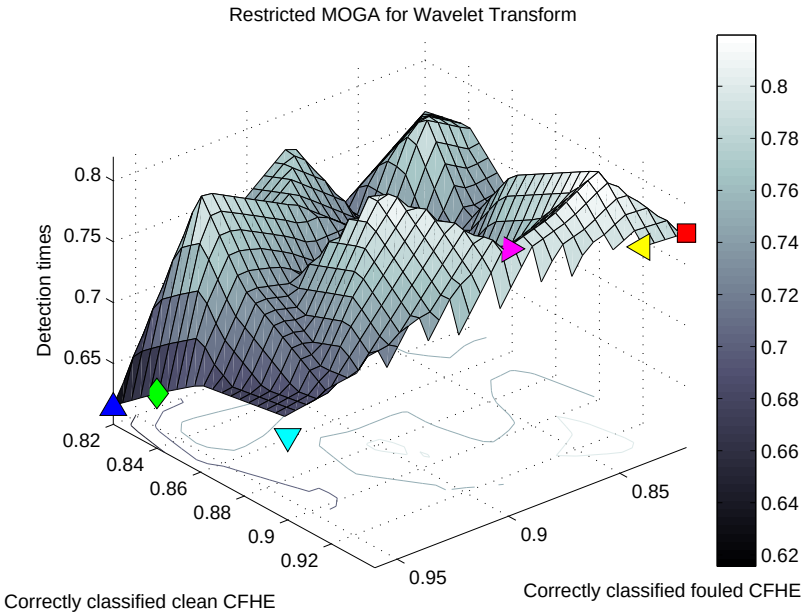

Figure 5.1. Retricted Pareto front of the MOGA for WT.

5.1 Improvements on ERM

For the paper (Ingimundardóttir and Lalot, 2009) the decision variables were chosen *ad hoc* as $(\theta, \kappa, \xi, \varrho) = (0.02, 2, 10, 0.85)$. The threshold was also chosen *ad hoc* as $\zeta = 0.85$ and the results were 100% correct detection of fouling for both fouled and clean CFHE and the mean detection time was $t = 0.6556$. If the threshold were chosen according to Eq. (3.2) it would be $\zeta = 0.96$, yielding: a 94% and 12% correct detection of fouling for fouled and clean CFHE respectively, and a mean detection time of $t = 0.354$ for the training set; a 90% and 8% correct detection of fouling for fouled and clean CFHE respectively, and a mean detection time of $t = 0.374$ for the test set. The great difference in the correct detection of fouling for the clean CFHE and the

mean detection times for the fouled CFHE between the results of the paper and this study is due to the fact the trials did not use the same data sets. The simulation of the CFHE was improved between the trials; also the paper did not split the data into training and test tests, so it is most likely that the data in the paper was overfitted; giving a loss of generality. On the other hand ERM is consistent, as the results from the training and test set do not vary greatly.

This particular solution, coded as ★ in Fig. 4.2, is not Pareto optimal. For instance $(\theta, \kappa, \xi, \varrho) = (0.0203, 2, 9, 0.859)$, coded as ▲ in Fig. 4.2, is believed to be Pareto optimal. Its threshold would be $\zeta = 0.96$, yielding: a 96% and 82% correct detection of fouling for fouled and clean CFHE respectively, and a mean detection time of $t = 0.614$ for the training set; 98% and 70% correct detection of fouling for fouled and clean CFHE respectively, and a mean detection time of $t = 0.5533$ for the test set. The detection of fouling for the test set is shown schematically in Figure 5.3, where the blue line denotes the threshold ζ, and the grey and black lines denote ERM ratios for the clean and fouled data sets, respectively. The first time the ratio falls below the threshold fouling is detected for that particular CFHE. False positive detection of fouling are the ratios of the clean CFHE that fall below the threshold; false negative detection of fouling are the ratios of the fouled CFHE that do not fall below the threshold. From the figure it is evident there is clean data set that doesn't fit with the others, and is therefore most likely an outlier. Removing it would yield a better correct classification rate for the clean CFHE, but outlier removal would be be too computationally expensive to implement along with MOGA.

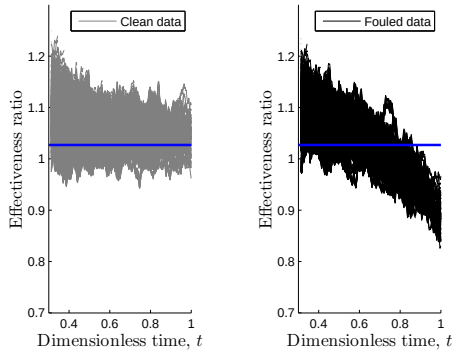

Figure 5.2. Detection of fouling using an ERM with MA on clean CFHE (left) and fouled CFHE (right).

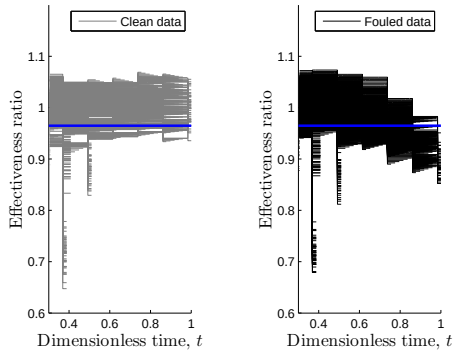

Figure 5.3. Detection of fouling using an ERM with WT on clean CFHE (left) and fouled CFHE (right).

5.2 Difference between methods

Comparing the Pareto optimal solutions for ERM using either MA or WT as its sub method, one can notice that WT can achieve a faster detection of fouling than using MA, and still retain similar correct detection rates for

fouling.

Lets look closer at the solutions: ♦ for MA, and ▼ for WT. Their mean detection time of fouling is $t = 0.7764$ and $t = 0.5526$, with a 95% confidence interval of $t \in [0.5618, 0.8816]$ and $t \in [0.2615, 0.8411]$, for MA and WT respectively, using the test set. WT outperforms MA in all aspects, it is on average doing better than the lower bound of MA's confidence interval and still WT manages to have a lower upper bound on its confidence interval. WT transform is then a better option for ERM's sub method.

Using an Extended Kalman filter the mean detection time of fouling is reported $t = 0.83$ with a 95% confidence interval of $t \in [0.59, 0.98]$. All of the methods fit well to previous research that typically the fouling factor are comprised in the range $[0.0001, 0.0007]$[1]. There is not much difference between the ERM with MA and EKF to infer one is better than the other, but ERM with WT would be a wiser choice.

The optimal solutions for MA do no vary much, the length of the window has a mean $t = 0.076$, and the outcome is invariant on the offset. WT on the other hand has more desicion variables, so it is hard to infer how the parameters infer with the outcome. Restricting the Pareto frontier to at least 80% correct classification of fouling a general rule of thumb for the selection of parameters would be a dimension of 2 or 4 and a scale of 9. The boundary condition and offset are not as convergent, they would need to be optimised with MOGA using a training set.

[1]Typical fouling factors can be found at: http://engineeringpage.com/technology/thermal/fouling_factors.html

5.3 Conclusion

ERM does not rely on any expert knowledge on the underlying data. It is robust in detection of fouling, but it is sensitive to its parameter selection. Clean training data is needed to find the threshold for ERM. For an optimal result MOGA is implemented, but that needs both clean and fouled data for the training set. MOGA is computationally expensive, and depending on the selection of parameters of ERM each run can be quite time consuming. Luckily the solutions on the Pareto front are of a low dimension and low scale for WT, so it relatively quick. Depending on the underlying data, e.g. dairy versus crude oil, time implementation can matter a great deal.

The confidence interval for ERM's mean detection time is much greater than the comparable state space model using the extended Kalman filter. Even if there is a lower mean detection time it still might not be applicable to use ERM instead of EKF; one should rather use a more consistent model like EKF.

EFK has more knowledge of the underlying system and can be further exploited. For EKF the relationship between detection time and corresponding fouling factor can be investigated, but this is not trivial using ERM since it is a black-box method.

5.4 Future work

The effectiveness ratio method is independent of the physical characteristics of underlying data, so it would be of interest to study detection of fouling in

other type of heat exchangers, whether it being different flow arrangement or different construction, e.g. shell-and-tube or parallel flow plate heat exchanger. Also it would of interest to try other sub methods to estimate the steady state trend for the effectiveness ratio method than those discussed in this study.

An investigation if there is a generalised link between the ERM's threshold and the its corresponding fouling factor for the mean detection time of fouling would be of interest. For this more varied type of data would be needed, but unfortunately access to real data is scarce.

Currently there is test rig cross-flow heat exchanger being constructed at *Université de Valenciennes et du Hainaut-Cambrésis* in Northern France. Once it is functional the methods discussed in this thesis will be put to the test: whether or not they can actually adequately detect fouling in the *real* world, or at least for dairy products.

Bibliography

G. Bylund. *Dairy Processing Handbook*. Tetra Pak Processing Systems AB, S-221 86 Lund, Sweden, 1995.

T. Casanueva-Robles and T.R. Bott. The environmental effect of heat exchanger fouling: a case study. In *Heat Exchanger Fouling and Cleaning – Challenges and Opportunities, Kloster Irsee, Germany*, 5-10 June 2005.

Y.A. Çengel and M.A. Boles. *Thermodynamics: and Engineering Approach*. McGraw-Hill, 6th edition, 2007.

H.M. Tai C.K. Sung and C.W. Chen. Locating defects of a gear system by the technique of wavelet transform. *Mechanism and Machine Theory*, 35: 1169–1182, 2000.

K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2003.

S. Lalot F. Delmotte, S. Delrot and M. Dambrine. Fouling detection in heat exchangers with fuzzy models. In *19th International Symposium on Transport Phenomena, Reykjavík, Iceland*, 17-21 August 2008. Paper #43.

O. Guðmundsson. Detection of fouling in heat exchangers. Master's thesis, University of Iceland, 2008.

H. Ingimundardóttir and S. Lalot. Detection of fouling in a cross-flow heat exchanger using wavelets. In *Heat Exchanger Fouling and Cleaning, Schladming, Austria*, 14-19 June 2009.

N.J. Leite J.A. Montoya Zegarra and R. da Silva Torres. Wavelet-based fingerprint image retrieval. *Journal of Computational and Applied Mathematics*, 227:294–307, 2009.

S. Lalot and S. Lecoeuche. Online fouling detection in electrical circulation heaters using neural networks. *International Journal of Heat and Mass Transfer*, 46:2445–2457, 2003.

S. Lalot and G. Mercère. Detection of fouling in a heat exchanger using a recursive subspace identification algorithm. In *ISTP-19, Reykjavík, Iceland*, 17-21 August 2008. Paper #37.

H. Pálsson O. Guðmundsson and Ó.P. Pálsson. Simulation of fouling in cross-flow heat exchanger and a fouling detection based on physical modeling. In *Modelling and Simulation of Energy Technology, Fredericia Denmark*, 7-8 October 2009.

M.C. Pereyra and M.J. Mohlenkampy. Wavelets, their friends, and what they can do for you. Retrieved September 23, 2009, from http://www.sci.sdsu.edu/compsciwork/IIIPASI/DOCUMENTS/COURSENOTES/waveletPASI-IIICPereyra.pdf, 2004.

G.R. Jónsson S. Lalot, Ó.P. Pálsson and B. Desmet. Use of extended kalman filtering in detecting fouling in heat exchangers. *International Journal of Heat and Mass Transfer*, pages 2643–2655, July 2007.

N. Saravanan and K.I. Ramachandran. Fault diagnosis of spur bevel gear

box using discrete wavelet features and decision tree classification. *Expert Systems with Applications*, 36:9564–9573, 2009.

K. Sastry. Single and multiobjective genetic algorithm toolbox for matlab in c++. Retrieved December 10, 2009, from http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2007017.pdf, June 2007.

H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley Interscience, New York, 1995.

R.C. Michelini V. Belotti, F. Crenna and G.B. Rossi. Wheel-flat diagnostic tool via wavelet transform. *Mechanical Systems and Signal Processing*, 20: 1953–1966, 2006.

M. Vetterli and J. Kovaĉević. *Wavelets and Subband Coding*. Prentice Hall PTR, Englewood Cliffs, New Jersey, reissued by the authors 2007 edition, 1995.

B. Vidakovic and P. Mueller. Wavelets for kids: A tutorial introduction. Retrieved September 23, 2009, from www2.isye.gatech.edu/~brani/wp/kidsA.ps, 1994.

A.P. Watkinson and Y.-H. Li. Fouling characteristics of a heavy vacuum gas oil in the presence of dissolved oxygen. In *Heat Exchanger Fouling and Cleaning, Schladming, Austria*, 14-19 June 2009.

G. Welch and G. Bishop. An introduction to the kalman filter. Retrieved November 25, 2009, from http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, 2006.

Appendix A

Kalman filter

The Kalman filter is a set of mathematical equations that recursively estimates a state of a process, such that it minimises the mean squared error. The filter is robust when estimating past, present, and future states, even if the precise nature of the modelled system is unknown.

A short introduction to the Kalman filter and the extended Kalman filter the reader can refer to (Welch and Bishop, 2006).

A.1 Standard Kalman filter

The standard Kalman filter estimates a state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the *linear* stochastic differential equation,

$$x_k = F_k x_{k-1} + B_k u_{k-1} + w_{k-1}, \tag{A.1}$$

with a measurement $z \in \mathbb{R}^n$ that is

$$z_k = Hx_k + v_k, \tag{A.2}$$

where the random variables w_k and v_k denote the process and measurement noise, respectively. They are assumed to be independent Gaussian white noise process, i.e.,

$$w_k \sim \mathcal{N}(0, Q_k), \tag{A.3}$$

$$v_k \sim \mathcal{N}(0, R_k). \tag{A.4}$$

As the equations above show, the Kalman filter filters out the noise from the measurements, yielding a desired estimate for the desired state noise free. The algorithm for a standard discrete Kalman filter is given in table A.1.

A.2 Extended Kalman filter

The extended Kalman filter, EKF, is the non-linear counterpart of the standard Kalman filter which linearises about the current mean and covariance. It estimates a state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the *non-linear* stochastic differential equation,

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}, \tag{A.5}$$

with a measurement $z \in \mathbb{R}^n$ that is

$$z_k = h(x_k) + v_k, \tag{A.6}$$

where the random variables w_k and v_k denote the process and measurement noise, respectively. They are assumed to be independent Gaussian white noise process, same as in equations A.3 and A.4.

Initial step	Set initial estimates.
	KF's prediction equations:
Step 1	Predicted state
	$\hat{x}_{k\|k-1} = F_k \hat{x}_{k-1\|k-1} + B_k u_{k-1}$
Step 2	Predicted estimate covariance
	$P_{k\|k-1} = F_k P_{k-1\|k-1} F_k^T + Q_{k-1}$
	KF's update equations:
Step 3	Innovation or measurement residual
	$\tilde{y}_k = z_k - H_k \hat{x}_{k\|k-1}$
Step 4	Innovation (or residual) covariance
	$S_k = H_k P_{k\|k-1} H_k^T + R_k$
Step 5	Optimal Kalman gain
	$K_k = P_{k\|k-1} H_k^T S_k^{-1}$
Step 6	Updated state estimate
	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_k \tilde{y}_k$
Step 5	Updated estimate covariance
	$P_{k\|k} = (I - K_k H_k) P_{k\|k-1}$
Step 6	Return back to step 1, with $k \leftarrow k + 1$

Table A.1. Standard Kalman filter algorithm

In this case the non-linear function f in equation A.5 relates the state at the previous time step t to the state of the current step t and an arbitrary driving function u_{k-1}. The non-linear function h in equation A.6 relates to the state x_k to the measurement z_k. The algorithm for an extended discrete Kalman filter is given in table A.2.

If f and h are both linear functions then equations A.5 and A.6 are equivalent to A.1 and A.2, respectively.

Unlike its linear counterpart, the extended Kalman filter is not necessarily an optimal estimator. It is dependent on its initial estimates, and if chosen poorly the filter my quickly diverge. Also it is noted that the EKF tends to underestimate the true covariance matrix.

Initial step Set initial estimates.

EKF's prediction equations:

Step 1 Predicted state

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$$

Step 2 Predicted estimate covariance

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{k-1}$$

EKF's update equations:

Step 3 Innovation or measurement residual

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$

Step 4 Innovation (or residual) covariance

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

Step 5 Optimal Kalman gain

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

Step 6 Updated state estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

Step 5 Updated estimate covariance

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Step 6 Return back to step 1, with $k \leftarrow k + 1$

where the state transition and observation matrices are defined to be the following Jacobians

$$F_k = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1|k-1}, u_k}$$

$$H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k-1|k-1}}$$

Table A.2. Extended Kalman filter algorithm

Appendix B

MATLAB code

```
1  function [objConst,correct,detect_times,th] = ERM(offset,
       submethod,subparm,train,test)
2  % Author: Helga Ingimundardóttir
3  % Call:   [objConst,correct,detect_times,th] = ERM(offset,
       submethod,subparm,train,test)
4  % Input:  offset = ERM offset
5  %         submethod = ERM sub method, either WT or MA
6  %         subparm = parameters for ERM's sub method
7  %         train = training data
8  %         test = test data
9  % Output: objConst = objective functions for ERM
10 %         correct = boolean, 1 if correct classifiction, 0
       otherwise
11 %         detect_times = dection times for fouling
12 %         th = ERM threshold
13
```

```
14      ratio.train = ratios(offset,@submethod,subparm,train,
            ts);
15      ratio.test = ratios(offset,@submethod,subparm,test,ts)
            ;
16
17      th=compute_threshold(ratio.train);
18
19      [detect_times.train,correct.train] = detection(ratio.
            train,th);
20      objConst.train(1) = mean(correct.train.f);
21      objConst.train(2) = mean(correct.train.c);
22      objConst.train(3) = mean(detect_times.train.f)/ts.
            final;
23
24      [detect_times.test,correct.test] = detection(ratio.
            test,th);
25      objConst.test(1) = mean(correct.test.f);
26      objConst.test(2) = mean(correct.test.c);
27      objConst.test(3) = mean(detect_times.test.f)/ts.final;
28
29  end
```

```
1   % Author: Helga Ingimundardóttir
2   % Call:   ratio = ratios(method,methodParm,data,ts)
3   % Input:  method: Function to approximate effectiveness
4   %         methodParm: Parameters for the method
5   %         data: Datasets to be used
6
7   % Output: ratio: Ratios for the effectiveness
```

```
%          detect: When ratio first falls below threshold

function ratio_burn = ratios(offset,method,methodParm,data
    ,ts)
if length(methodParm) == 4 % only the wavelet transfrom
    has 4 methodParm
     ts.boundarycondition=methodParm{4};
else
     ts.boundarycondition = 1;
end
ts.firstpt=1;

% Ratio param
ratio.offset = offset;
ratio.number_of_sets = data.nb;
ratio.intervals = floor((ts.final-ts.begin)/ratio.offset)
    +1;


for k=1:data.nb
  whom = k; where = ts.begin:ts.final;
  c = struct(...
      'Tho',data.(data.name{whom}).clean.Tho(where),...
      'Tco',data.(data.name{whom}).clean.Tco(where),...
      'Thi',data.(data.name{whom}).clean.Thi(where),...
      'Tci',data.(data.name{whom}).clean.Tci(where),...
      'm_h',data.(data.name{whom}).clean.m_h(where),...
      'm_c',data.(data.name{whom}).clean.m_c(where));
  f = struct(...
```

```
34        'Tho',data.(data.name{whom}).fouling.Tho(where),...
35        'Tco',data.(data.name{whom}).fouling.Tco(where),...
36        'Thi',data.(data.name{whom}).fouling.Thi(where),...
37        'Tci',data.(data.name{whom}).fouling.Tci(where),...
38        'm_h',data.(data.name{whom}).fouling.m_h(where),...
39        'm_c',data.(data.name{whom}).fouling.m_c(where));
40
41    % Lowpass approximation for the given number of
          iterations
42    j=0;
43    for i=ts.begin:ratio.offset:ts.final
44      j=j+1;
45      f_tmp = method(Eff(f.Thi(1:i),f.Tho(1:i),f.Tci(1:i)),
            methodParm);
46      c_tmp = method(Eff(c.Thi(1:i),c.Tho(1:i),c.Tci(1:i)),
            methodParm);
47
48      ts.lastpt=floor(length(c_tmp)*ts.boundarycondition);
49
50      f_tmp = f_tmp(ts.firstpt:ts.lastpt);
51      c_tmp = c_tmp(ts.firstpt:ts.lastpt);
52
53      ratio.f{k,j} = f_tmp/mean(f_tmp);
54      ratio.c{k,j} = c_tmp/mean(c_tmp);
55
56      ratio.N(j) = length(f_tmp);
57
58    end % end i for ratio
59  end % end k for set
```

```
60
61 ratio.final = ts.lastpt;
62 ratio.begin = floor(0.3*ratio.final);
63 ratio_burn = cut_burnin(ratio);
64 end % Main function finished
65
66 %% --- AUXILIARY FUNCTIONS ---
67
68 function E = Eff(Thi,Tho,Tci)
69   E = (Thi-Tho)./(Thi-Tci);
70 end
71 function ratio_burn = cut_burnin(ratio)
72     for j=1:ratio.intervals
73         if ratio.N(j) < ratio.begin
74             burn_set = j+1;
75         end
76     end
77     ratio.begin = ratio.N(burn_set);
78
79     ratio_burn.offset = ratio.offset;
80     ratio_burn.number_of_sets = ratio.number_of_sets;
81     ratio_burn.intervals = ratio.intervals-burn_set+1;
82     ratio_burn.begin = ratio.begin;
83     ratio_burn.final = ratio.final;
84     ratio_burn.N = ratio.N(burn_set:end);
85     for k=1:ratio.number_of_sets;
86         for j=burn_set:ratio.intervals
87             ratio_burn.c{k,j-burn_set+1}=ratio.c{k,j}(
                ratio.begin:end); % Henda burn-in svaedi
```

```
            ratio_burn.f{k,j-burn_set+1}=ratio.f{k,j}(
                ratio.begin:end); % Henda burn-in svaedi
        end
    end
end
```

```
function threshold=compute_threshold(ratio)
total_min = [];
for k=1:ratio.number_of_sets
    for j=1:ratio.intervals
        total_min = [total_min;min(ratio.c{k,j})];
    end
end
threshold.mean = mean(total_min);
threshold.quant = quantile(total_min,[0.025,0.975]);
%fprintf('Threshold: %6.4f, CI: [%6.4f,%6.4f]\n',threshold
    .mean,threshold.quant(1),threshold.quant(2));
end
```

```
function [detection_times,correct] = detection(ratio,th)
% Author: Helga Ingimundardóttir
% Call:    [detection_times,correct,detect_times] =
    detection(ratio,th)
% Input:   ratio = ratio is the time series of
    approximated effectiveness
%                  wrt to its average
%          th = threshold for ERM
% Output:  detection_times = all the detection time of
    fouling
%          correct = percentage of correctly classified
```

```
    detection

detection_times.c = ones(1,ratio.number_of_sets)*ratio.
    final;
detection_times.f = ones(1,ratio.number_of_sets)*ratio.
    final;

for j=1:ratio.intervals
  for k=1:ratio.number_of_sets
    if min(ratio.c{k,j}) < th.quant(1)
      if detection_times.c(k) == ratio.final
        detection_times.c(k) = ratio.N(j);
      end
    end
    if min(ratio.f{k,j}) < th.quant(1)
      if detection_times.f(k) == ratio.final
        detection_times.f(k) = ratio.N(j);
      end
    end
  end
end

correct.f = (detection_times.f ~= ratio.final);
correct.c = (detection_times.c == ratio.final);
end
```

```
% Author: Helga Ingimundardóttir
% Call:    filters = daub(N)
% Input:   N  = width of the support (number of
```

```
   coefficients) and must be an even number
% Output:  filters is a struct with the fields
%           filters.an.low = analysis lowpass filter
%           filters.an.high  = analysis highpass filter
%           filters.sy.low = synthesis lowpass filter
%           filters.sy.high = synthesis highpass filter
% ... for the Daubechies Wavelet basis
function filters = daub(N)

if rem(N,2)
  error( 'Error: width of the support must be of even
     length')
end
A = N/2;

% Have to implement the polynomial 'p' [see research book]
tmp = zeros(A,A+A-1);
tmp(1,A) = 1;

for k=1:A-1
  subtmp = [1 -2 1];
  for j=2:k
    subtmp = conv(subtmp,[1 -2 1]);
    % conv is inbuilt in MATLAB and if the inputs are
       vectors of polynomial coefficients, convolving them
        is equivalent to multiplying the two polynomials.
  end
  subtmp = nchoosek(A-1+k,k)*(-1/4)^k*subtmp;
  % Add the zeros on each side of the subtmp
```

```
29    tmp(k+1,:) = [zeros(1,A-k-1),subtmp,zeros(1,A-k-1)];
30  end
31
32  % The tmp-matrix has to be summed for each column (for the
        represent the same term in the polynomial 'p'
33  polyp = sum(tmp,1);
34
35  % Finding the roots/vanishing points of the polynomial 'p'
36  poly_zeros = roots(polyp); % inbuilt in MATLAB
37
38  % For the 'extremal phase' we choose the roots within the
       unit circle
39  zerosinside = poly_zeros(find(abs(poly_zeros)<=1));
40
41  % Need to sort the zeros for real and imaginary for ease
       of computation
42  imagzero = [];
43  realzero = [];
44  for i=1:length(zerosinside)
45    if imag(zerosinside(i))~=0
46      imagzero(end+1)=zerosinside(i);
47    else
48      realzero(end+1)=zerosinside(i);
49    end
50  end
51
52  % From the vanishing points we can make the synthesis
       lowpass filter 'rh'
53  rh = [1,1];
```

```
54 for i=2:A
55   rh = conv(rh,[1,1]);
56 end
57 for i=1:length(realzero)
58   rh = conv(rh,[1  -realzero(i)]);
59 end
60 for i=1:2:length(imagzero)
61     rh = conv(rh,[1  -2*real(imagzero(i))  (abs(imagzero(i
       )))^2 ]);
62     % Note: Conjugate roots are multiplied together before
         they're convolved with other roots. That's why we
       sorted out the real zeros from the imaginary ones.
63 end
64
65 % Normalizing term derived from the polynomial 'p' for the
     filter 'rh'
66 norm_term = 1;
67 for i=1:A-1
68     norm_term = norm_term * abs(zerosinside(i));
69 end
70 norm_term = abs(polyp(1))/norm_term;
71 norm_term = (1/2)^A * sqrt(2) * sqrt(norm_term);
72
73 rh = norm_term * rh;
74
75 % Since the Daubechies filter is orthogonal we get the
    other filters in the following manner:
76 for i=1:length(rh)
77     rg(i) = (-1)^(i+1)*rh(end-i+1);
```

```
78      % 'rg' is 'rh' reversed with everyother term
            multiplied by -1
79  end
80  % 'h' and 'g' are reversals of 'rh' and 'rg' respectively.
81  h = rh(length(rh):-1:1);
82  g = rg(length(rh):-1:1);
83
84  filters.an = struct('low',h,'high',g);
85  filters.sy = struct('low',rh,'high',rg);
86  end
```

```
1   % Author:  Helga Ingimundardóttir
2   % Call:    yhat = aprox(y,parm)
3   % Input:   y: time series that is to be approximated
4   %          parm: {h,rh,sc}; where
5   %            h: lowpass analysis filter
6   %            rh: lowpass synthesis filter
7   %            sc: how often the filters should be applied
8   % Output:  yhat: is the approximated result after using
9   %          lowpass analysis, and lowpass synthesis filter
10  %          on the time series y for a smoother trendline.
11  function yhat = aprox(y,parm)
12
13  h = parm{1}; rh = parm{2}; sc = parm{3};
14  % Make sure that 'y','h' and 'rh' are column vectors
15  y=y(:)'; h=h(:)'; rh=rh(:)';
16
17  % Using the theory on page 15 in wavelet_01.pdf
18  ytmp=y;
```

```
19 for i=1:sc
20   % Convolving the analysis filter with the detailed data
        (at scale i-1)
21   ytmp=conv(h,ytmp); % yielding data on a rougher scale (
        at scale i)
22   % Downsampling by 2
23   ytmp=ytmp(1:2:length(ytmp));
24 end
25
26 for i=1:sc
27   % Upsampling by 2
28   y_before = ytmp;
29   ytmp = zeros(2*length(y_before),1);
30   ytmp(1:2:end) = y_before;
31   % convolving the synthesis filter with the rough data
32   ytmp = conv(rh,ytmp);
33 end
34
35 % Have to cut off excess entries in the beginning and end
     of 'ytmp'
36 D=floor((length(h)+length(rh))/2)-1;
37 for i=1:sc
38   d(i) = D*2^(i-1);
39 end
40 d=sum(d);
41 yhat=ytmp(1+d:length(y)+d)';
42 end
```